Differentially Private Ridge Regression

The Cost of a Hyperparameter

Tyler Piazza

An undergraduate thesis submitted to the The School of Engineering and Applied Sciences in partial fulfillment of the requirements for the joint degree of Bachelor of Arts in Computer Science and Mathematics with Honors

Harvard University
Cambridge, Massachusetts
March 26, 2021.

Differentially Private Ridge Regression

The Cost of a Hyperparameter

Tyler Piazza

Abstract

Studying problems of interest, like finding trends in medical data, can require analyzing data which contains sensitive and personally identifying information. As a result, it is often infeasible to release these datasets to researchers or to the general public. In this paper, we study algorithms that are differentially private, where there are theoretical guarantees that the mechanisms studied will reveal only limited amounts of information about individual people while still providing insights about large groups. This thesis discusses various forms of linear and ridge regression in this differentially private setting, with the goal of studying sensitive data to make predictions about future sensitive data. In particular, we will discuss the internal privacy-loss budgeting of the differentially private ridge regression technique adaSSP. This thesis provides 3 contributions. First, we discuss the existing SSP and adaSSP algorithms, and provide detailed proofs that they are each differentially private. Second, we introduce the two new algorithms adaSSPbudget and constSSPfull and prove that these are each differentially private. Third, we conduct experiments using synthetic and real world data to explore whether the precise privacy-loss budgeting used within these algorithms could improve their performance. These experiments will explore the tradeoff between the accuracy of a hyperparameter and the accuracy of the other releases. Through the experimental results, we find that the performance is often insensitive to the particular privacy-loss budgeting and that for certain datasets, no choice of privacy-loss budget allows for the adaptive adaSSPbudget to outperform the standard SSP algorithm.

Acknowledgements

I would like to thank my thesis advisor, Professor Salil Vadhan, for introducing me to the wonderful world of differential privacy and for his continued patience and attention to detail. I also want to acknowledge Allison Choat, Daniel Alabi, and Yu-Xiang Wang for their support.

Lastly, I would like to give a sincere dedication to my parents for their lifelong support and encouragement in my education, from the first day of kindergarten at Hardy Elementary School through my senior year at Harvard College.

Contents

1 Introduction

Linear regression is an old technique, published by Legendre in 1805 and Gauss in 1809 [28]. 200 years later, linear regression remains a common form of data analysis, where coefficients are assigned to the different parameters of our dataset to reduce some pre-defined loss. In the case where linear regression is used to predict new, unseen data, the linear regression model is fit on a training set of data and is often evaluated on a different testing set of data. Using the existing least squares loss metric, one can analytically solve for the coefficients which minimize the loss on the training set [15].

The issue with this approach is that what is best for the training set may not be best for the testing set. In particular, the coefficients assigned to the linear regression model may be highly dependent on the particular configuration of the training set, in such a way that the coefficients do not generalize to data that was generated in a similar way. One way to prevent this over fitting is to penalize the size of the coefficients of the model in the loss function. This way, the coefficients may not grow to improbably large values in order to make marginal gains in the loss function on the training set. One way to perform this penalization is to use ridge regression [22] [16], which penalizes the model with the $L2$ norm of the linear regression coefficients. One challenge with ridge regression is that the $L2$ penalty needs to be scaled (relative to the squared error loss), and the optimal scalar depends on the data. This scalar in ridge regression is a special case of a machine learning hyperparameter, which is a parameter in a machine learning model that is not automatically learned along with the rest of the model, so it must be input to the model before training begins. In ridge regression, neural networks, support vector machines, and other machine learning techniques, the task of choosing the optimal hyperparameters can appear ad-hoc and may require several different interactions with the dataset [8].

Of course, data science is not always done in a vacuum, and we cannot interact with sensitive data as much as we may want for the model's accuracy. Machine learning has become popular, even in cases with sensitive data. For example, some hospitals use machine learning to decide the likelihood of a re admittance of a patient [30], and web mail providers detect spam using machine learning [27]. Furthermore, some datasets seem private but actually reveal sensitive information. As an example, consider Amazon's [17] product recommendation system, which informs users that customers who bought some particular item also bought some list of several other items. Researchers in 2011 were able to observe these recommendations change over time, cross-reference with customers' public reviews of items, and then infer if a particular customer bought some specific item on a given day, even if the customer did not review the item [17]. These aggregate recommendations and customer reviews at first seem to be private, but clearly some sensitive information was able to leak through. Therefore, a need exists for a formalized way to ensure that released values about a dataset do not reveal personal information.

Differential privacy (DP) [11] is a formalized approach at studying sensitive data, such that the study produces accurate results, but the identities of the data subjects remain obscured. This paper focuses on the definition of differential privacy with the (ϵ, δ) parameters to describe how much privacy is lost when the algorithm is called (see Definition 3). Differential privacy aims to capture some idea of "plausible deniability", where the results of an algorithm do not change by very much if one particular row of the dataset changes. Since differential privacy's inception, it has connected to a variety of fields. Applications of differential privacy include eye-tracking software [4] and graph theory [3], and differential privacy also has deep theoretical connections to cryptography [24], through fingerprinting codes, traitor tracing, and digital signatures.

One fortunate feature of differential privacy is that if multiple algorithms satisfy differential privacy, then calling all algorithms (independently) has privacy-loss parameters which are at most the sum of the individual privacy-loss parameters (see Theorems 4, 5). In other words, the privacy loss of the whole is the sum of the privacy losses of its parts. Another benefit of differential privacy is that we can start with a pre-defined privacy-loss budget (ϵ, δ), and construct multiple algorithms with their own budgets such that the composition of the algorithms satisfies the overall budget. This setup leads to situations where the

privacy loss is split between the different algorithms. In differential privacy, accuracy and privacy come at a trade off, so making one algorithm more accurate may cause a different algorithm to be less accurate, if the overall privacy-loss budget must be satisfied.

Tying these notions of ridge regression and privacy-loss budgeting together, Wang [26] presents a differentially private ridge regression technique adaSSP. The algorithm releases 3 quantities, each release output from standalone differentially private algorithms. Then, the algorithm combines the releases together to output regression coefficients. adaSSP builds on the foundations of Sufficient Statistics Perturbation (SSP) [25] [13] by releasing an extra quantity to adaptively compute the ridge regression hyperparameter. Intuitively, this one privately released quantity could let ridge regression implementers sidestep the issue of returning to the data multiple times to choose the best ridge regression coefficient. A potential downside to this approach is that the differentially private release of this value to help the hyperparameter will reduce the privacy-loss budget of the other releases of the algorithm. With this framework, the use of a well tuned hyperparameter has a cost to the accuracy of the quantities used in the original model.

1.1 Contributions of this Thesis

This thesis aims to study differentially private ridge regression algorithms and how the privacy-loss budget impacts the performance of the model on real and synthetic data. In particular, this paper surveys the proposed differentially private ridge regression algorithm adaSSP which splits its privacy-loss budget to compute an eigenvalue in the hopes of using an adaptive hyperparameter.

The theoretical contributions of this paper are two new proposed ridge regression algorithms, adaSSPbudget and constSSPfull, along with proofs that they are differentially private. This paper also provides detailed proofs for other differentially private regression techniques which have been studied elsewhere, adaSSP and SSP. The main tools involved in proving these claims come from quantifying the $l2$ sensitivity (Definition 6) of the internal releases of these algorithms. Then, with $l2$ sensitivity, we appeal to the Gaussian Mechanism (Lemma 7) to argue that each release is differentially private. Finally, we can combine Composition of DP and Post Processing of DP (Theorems 4 and 5) to conclude that the entire algorithm is differentially private.

The experimental contributions of this paper are the study of the privacy-loss budget of the ridge regression mechanism adaSSP under different data conditions, including synthetic data and real UCI (University of California Irvine) data. To measure performance, we use metrics of relative efficiency and prediction error. By exploring the performance of our algorithms under different privacy-loss parameters, data sizes, and datasets, these experiments aim to describe how and when the benefit of the adaptively chosen ridge regression hyperparameter of adaSSP is worth the corresponding privacy-loss budget cost to the other releases within the algorithm.

2 Notation and Setup

2.1 Matrix and Regression Notation

In this thesis, we will assume that there is a dataset $X \in \mathbb{R}^{n \times d}$, with n rows (i.e. n participants, n data points) and d columns (i.e. d features, d predictors). There is a corresponding response vector $\boldsymbol{y} \in \mathbb{R}^n$. $||\cdot||_2$ denotes the Euclidean norm for vector inputs. For matrix inputs, there are two primary norms used in this paper: $||\cdot||_F$ is the Frobenius norm (the square root of the sum of the squared entries of the matrix), and $||\cdot||_2$ for a matrix will be the spectral norm $||A||_2 = \sup_{x \in \mathbb{R}^d, ||\boldsymbol{x}||_2 = 1} ||A\boldsymbol{x}||_2$. In general, upper case letters (X, A, etc.) refer to matrices, bold letters ($\boldsymbol{y}, \boldsymbol{x_i}$, etc.) refer to vectors, and lower case letters (y, λ, etc.) refer to scalars in $\mathbb{R}$.

It is also the case that $||\cdot||$, when applied to subsets of Euclidean space, will refer to the radius of the smallest Euclidean ball containing that set. We are concerned with $\mathcal{X} \subset \mathbb{R}^d$, the set of all possible row vectors $\boldsymbol{x}$ of a dataset X, and $\mathcal{Y} \subset \mathbb{R}$, the set of all possible $y \in \mathbb{R}$ values. Thus, we assume finite bounds on these domains, $||\mathcal{X}|| = \sup_{\boldsymbol{x} \in \mathcal{X}} ||\boldsymbol{x}||$, $||\mathcal{Y}|| = \sup_{y \in \mathcal{Y}} |y|$. For this paper, the values of $||\mathcal{X}||$ and $||\mathcal{Y}||$ are fixed and known ahead of time.

When discussing eigenvalues, $\lambda_{min}(X^T X)$ (or λ_{min}) denotes the smallest eigenvalue of $X^T X$. Below are two propositions that provide the solution to ordinary least squares and ridge regression.

Proposition 1. $\Big($***Ordinary Least Squares Solution*** *[15]* $\Big)$ *Given* $X \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \mathbb{R}^n$,

$$\boldsymbol{\theta}^* = (X^T X)^{-1} X^T \boldsymbol{y}$$

is the (non-private) least squares solution, which is the vector $\boldsymbol{\theta} \in \mathbb{R}^d$ *that minimizes*

$$L(\boldsymbol{\theta}) = ||\boldsymbol{y} - X\boldsymbol{\theta}||_2^2$$

the ordinary least squares loss.

Proposition 2. $\Big($***Ridge Regression Solution*** *[15]* $\Big)$ *Given* $X \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \mathbb{R}^n, \lambda \geq 0$,

$$\boldsymbol{\theta}_\lambda^* = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}$$

is the (non-private) optimal solution to the ridge regression problem, minimizing

$$L_\lambda(\boldsymbol{\theta}) = ||\boldsymbol{y} - X\boldsymbol{\theta}||_2^2 + \lambda ||\boldsymbol{\theta}||_2^2$$

This value λ *here is the ridge regression* ***hyperparameter****.*

Note that ridge regression has the added benefit of numerical stability, which was one of the original motivations of ridge regression [22] [16]. The solution to ordinary least squares (Proposition 1) involves taking the inverse of $X^T X$, but there may not be guarantees ahead of time that this matrix is in fact invertible. As a result, by instead taking the inverse of $X^T X + \lambda I$ in Proposition 2, ridge regression can aim to avoid this invertibility problem by adding λ to the eigenvalues of $X^T X$.

$\boldsymbol{\theta}_0$ refers to the true $\boldsymbol{\theta}$ value, if it is known (such as in synthetic data, where the $\boldsymbol{\theta}$ value is known ahead of time and used to generate the data). In the procedure which is formally described in the experimental section, a random $\boldsymbol{\theta}_0 \in \mathbb{R}^d$ is generated alongside a random $X \in \mathbb{R}^{n \times d}$, and a $\boldsymbol{y} \in \mathbb{R}^n$ vector is generated with $X\boldsymbol{\theta}_0$ plus random Gaussian noise. $\widehat{\boldsymbol{\theta}}$ will refer to some predicted value of $\boldsymbol{\theta}$ (from one of the algorithms, for example).

In this paper, one metric for success of a linear regression (or ridge regression) algorithm will be relative efficiency:

$$\frac{\mathrm{E}\Big[||\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0||_2^2\Big]}{\mathrm{E}\Big[||\boldsymbol{\theta}^* - \boldsymbol{\theta}_0||_2^2\Big]}$$

This metric rewards being close to the true parameter, and normalizes by what the non-private linear regression could accomplish.

The randomness is over the cross validation trials. In the experiments, we use cross validation to split the data into training data and test data, where the algorithms perform procedures on the training data and then may do calculations using the remaining test data. Cross validation works by specifying the number of folds k in the cross validation, where the data is randomly divided into k equally sized chunks. Then, the

algorithm will use $k-1$ chunks to train, and then possibly the remaining last chunk to use as some sort of benchmark, as a test set. When using cross validation in this paper, we divide the data such that the training set is 90% of the data, and the test set is the remaining 10%. In the various experiments, we use 32 trials. This system means that 32 times, the data is split into a new, random train set and a test set, and the algorithms are used accordingly (with fresh randomness).

Another metric for success is prediction error, or mean squared error (MSE). For this metric, the data $X, \boldsymbol{y}$ must first be divided into an $X_{train}, \boldsymbol{y}_{train}$ and $X_{test}, \boldsymbol{y}_{test}$. This splitting would happen as described with cross validation. Next, $\widehat{\boldsymbol{\theta}}$ is computed using $X_{train}, \boldsymbol{y}_{train}$, and then define

$$\boldsymbol{y}_{pred} := X_{test}\widehat{\boldsymbol{\theta}}$$

Then, if there are n_{test} elements of $\boldsymbol{y}_{test}$, compute the prediction error (MSE) as

$$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (y_{pred,i} - y_{test,i})^2$$

Unlike relative efficiency, prediction error is not normalized with a value involving non-private ordinary least squares.

2.2 Principal Components and Eigenvalues

The algorithms described in later sections will involve $X^T X$, eigenvalues, and ridge regression, so to gain an intuition for how these quantities are connected, we can consider the singular value decomposition (SVD) of X. We have from SVD that

$$X = UDV^T$$

[15] where we still operate with $X \in \mathbb{R}^{n\times d}$, so $D \in \mathbb{R}^{d\times d}$ is a diagonal matrix, and $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{d\times d}$ are each orthogonal matrices. Let the diagonal entries of D be given by $d_1 \geq d_2 \geq \cdots \geq d_d \geq 0$. These are called the *singular values* of X. Let the column vectors of U and V be represented as $\boldsymbol{u}_j, \boldsymbol{v}_j$.

Using this decomposition, we can see what X applied to the ordinary least squares solution is. Recall that X applied to the ordinary least squares solution also corresponds to the predicted $\widehat{\boldsymbol{y}}$ values in this setting.

$$X\boldsymbol{\theta}^* = X(X^T X)^{-1}X^T\boldsymbol{y} = UU^T\boldsymbol{y}$$

Interpreting this result, we see that $U^T\boldsymbol{y}$ are the projected coordinates of $\boldsymbol{y}$ with respect to the orthogonal coordinates U. In contrast, when we try the same procedure for ridge regression with hyperparameter λ, we find

$$X\boldsymbol{\theta}^*_\lambda = X(X^T X + \lambda I)^{-1}X^T\boldsymbol{y} = UD(D^2 + \lambda I)^{-1}DU^T\boldsymbol{y}$$

$$= \sum_{j=1}^{d} (\boldsymbol{u_j} \frac{d_j^2}{d_j^2 + \lambda} \boldsymbol{u_j^T})\boldsymbol{y}$$

Because U is orthogonal, we see that $X\boldsymbol{\theta}^*_\lambda$ computes $\boldsymbol{y}$ in the new coordinate system U, and then shrinks the jth coordinate by $\frac{d_j^2}{d_j^2+\lambda}$. Notice that more shrinkage is applied to the coordinate with the smallest singular value (in this notation, d_d).

To interpret these d_j values, we can reason through principal component analysis. The sample covariance matrix is given by $S = \frac{X^T X}{n} \in \mathbb{R}^{d\times d}$, and we know

$$X^T X = VD^2V^T$$

This is the *eigen decomposition* of X^TX. The columns of $V \in \mathbb{R}^{d\times d}$, $\boldsymbol{v_j}$, happen to be the eigenvectors, and they are called the *principal components* of X. Notice that the eigenvalues of X^TX are precisely the d_j^2 values, so $\lambda_{min}(X^TX) = d_d^2$. It happens that $\boldsymbol{v_1}$ has the property such that $\boldsymbol{z_1} = X\boldsymbol{v_1}$ has the largest sample variance among all normalized linear combinations of the columns of X [15]. What we find is that for each j, if we set $\boldsymbol{z_j} = X\boldsymbol{v_j}$, then

$$\text{Var}(\boldsymbol{z_j}) = \text{Var}(X\boldsymbol{v_j}) = \frac{d_j^2}{n}$$

As a result, $\boldsymbol{z_d}$ has the minimum variance, so small eigenvalues values d_j^2 of X^TX correspond to directions in the column space of X having small variance, so ridge regression shrinks these directions the most. The interpretation for such a result is that ridge regression "assumes" that directions with higher variance of X will be more useful in predicting the $\boldsymbol{y}$ response.

With this framework, we can see that the minimum eigenvalue of X^TX, $\lambda_{min}(X^TX) = d_d^2$, is an indicator of how much shrinkage is applied to any of the coordinates. If this minimum eigenvalue is small (compared to the other eigenvalues), then at least one direction should be shrunk to amplify the effect of the other directions on the regression. Thus, a larger λ hyperparameter should be chosen. In the other direction, if this minimum eigenvalue value is large, then a smaller λ should be chosen because there is less of a need to dampen some directions so that others can play a larger role.

As we shall see in the algorithms adaSSP and adaSSPbudget, they operate in a framework where a larger $\lambda_{min}(X^TX)$ corresponds to a smaller λ hyperparameter in ridge regression, and vice versa for smaller $\lambda_{min}(X^TX)$.

2.3 Differential Privacy Results

Differential privacy is defined as follows:

Definition 3. (**Differential Privacy** [9]) Let $(X, \boldsymbol{y}) \sim (X', \boldsymbol{y'})$ denote when dataset $(X', \boldsymbol{y'})$ can be constructed by adding or removing one row $(\boldsymbol{x}, y)$ from $(X, \boldsymbol{y})$.

A randomized algorithm M satisfies (ϵ, δ)-*DP* if for all $X, \boldsymbol{y}, X', \boldsymbol{y'}$ such that $(X, \boldsymbol{y}) \sim (X', \boldsymbol{y'})$, and for any measurable set S, over the probability of the algorithm

$$P(M((X, \boldsymbol{y})) \in S) \leq e^{\epsilon} P(M((X', \boldsymbol{y'})) \in S) + \delta$$

The parameter ϵ represents the amount of privacy lost from running the algorithm, and δ denotes the small probability of failure. These are input by a user, and they are often considered to provide meaningful privacy protection if $\epsilon \leq 1, \delta \ll 1/n$.

Note that for small ϵ values, $e^{\epsilon} \approx 1 + \epsilon$, so this definition captures an idea that no differentially private algorithm (for reasonably small ϵ, δ values) can with high probability detect whether some individual is in the dataset. Our procedure may be intended for some other use, like computing a median value of our dataset, but this differential privacy definition provides statistical guarantees that individual information is not also released with high accuracy. One interpretation of this probability relation is in comparing the utility of joining or not joining an experiment [5].

Suppose that we could assign numeric value to the outcome of the experiment - perhaps you win or lose some amount of money depending on the mechanism's output. If the probability that the mechanism outputs some particular value is similar regardless of your choosing or abstaining from participating in the

experiment, it follows that the expected utility (money) will remain largely the same (if $\delta = 0$, the utilities are within a factor of e^{ϵ}). Thus, a participant stands to individually lose very little by joining the experiment. Observe that this behavior rules out mechanisms which would accurately assess "is John Doe in this dataset?", because the answer to this question changes with the addition or subtraction of one person.

Note that the definition of differential privacy only assures a statistical guarantee about related datasets - it does not guarantee that the mechanism actually releases a value that someone would find useful. A mechanism that always outputs 0 will satisfy differential privacy for arbitrary ϵ, δ values. As a result, when a new differentially private mechanism is proposed, great care must be taken in demonstrating or proving that it computes, with some accuracy, a quantity of interest.

When we have multiple DP algorithms, we may wonder if we can combine them and still have a DP algorithm. This next theorem describes a case where the answer is yes and we can describe new privacy-loss parameters.

Theorem 4. $\Big($***Composition of DP*** *[9]* $\big)$*. Let M_i be an (ϵ_i, δ_i)-DP algorithm for $i \in \{1, 2, \cdots, k\}$. Then if $M_{[k]}$ is defined as $M_{[k]}(X, \boldsymbol{y}) = (M_1(X, \boldsymbol{y}), \cdots, M_k(X, \boldsymbol{y}))$, using independent randomness in each algorithm M_i, then $M_{[k]}$ is $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.*

Theorem 5. $\Big($***Post-processing of DP*** *[32]*$\Big)$*. Let M be a (ϵ, δ)-DP algorithm, and let g be an arbitrary mapping from the set of possible outputs of M to an arbitrary set. Then $g \circ M$ is (ϵ, δ)-DP.*

With composition and post-processing, it is clear that any function of multiple independent releases is still DP (if the multiple releases are each DP).

To satisfy the constraints of differential privacy, many of the algorithms considered in this paper use the addition of Gaussian noise. Below is one such mechanism, which is mindful of the sensitivity of the function in question. Note that $\mathcal{N}(\mu, \sigma^2)$ denotes a Normal random variable with mean μ and variance σ^2 (and thus standard deviation σ). In higher dimensions, $\mathcal{N}(\mathbf{0}, I_d)$ corresponds to a set of d independent variables, each distributed as $\mathcal{N}(0, 1)$.

Definition 6. $\Big(l_2$ **sensitivity** [14] $\Big)$ Let f be a vector $\mathbb{R}^d$ valued function and let

$$\Delta_f = \sup_{(X, \boldsymbol{y}) \sim (X', \boldsymbol{y}')} ||f(X, \boldsymbol{y}) - f(X', \boldsymbol{y}')||_2$$

denote its l_2**-sensitivity**.

Lemma 7. $\Big($***Gaussian mechanism*** *[14]* $\Big)$ *Let f be a vector $\mathbb{R}^d$ valued function and let I_d denote the $d \times d$ identity matrix. The* ***Gaussian mechanism***

$$M(X, \boldsymbol{y}) = f(X, \boldsymbol{y}) + \frac{\sqrt{2 \log(2/\delta)}}{\epsilon} \Delta_f \mathcal{N}(\mathbf{0}, I_d)$$

satisfies (ϵ, δ)-differential privacy.

The proof that this Gaussian mechanism satisfies (ϵ, δ)-differential privacy can be found in [9]. Note that even though the definition of the Gaussian mechanism uses an exact $l2$ sensitivity, we can see that the result will hold even if we replace the Δ_f with an upper bound on the $l2$ sensitivity. This is the case because differential privacy (Definition 3) is a definition that applies over a set of possible datasets $X \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \mathbb{R}^n$. If differential privacy applies to some entire set S of datasets, then it follows that differential privacy (with the same ϵ, δ) will apply over a subset $S' \subset S$. The Gaussian mechanism's differential privacy parameters will

still hold if we replace Δ_f with an upper bound on this $l2$ sensitivity, because the $l2$ sensitivity on a subset S' will be at most the $l2$ sensitivity on the overall set S (given that $l2$ sensitivity has to do with supremums over the entire set S). Therefore, if we can say that a mechanism is (ϵ, δ)-DP over some set S of datasets that have some $l2$ sensitivity Δ_f, then it follows that the same mechanism is (ϵ, δ)-DP over a subset $S' \subset S$, even if that subset S' had a smaller $l2$ sensitivity.

3 Survey of Prior Work

Given that linear regression is a common tool to solve a variety of problems, differentially private linear regression has been studied in various contexts. Sheffet [19] discusses the use of ordinary least squares (OLS) as a descriptive tool to study a dataset. This work with ordinary least squares also extends to releasing t-values and confidence intervals. There has been work on the task of differentially private Bayesian linear regression [2], where the goal is to obtain a distribution for $\boldsymbol{\theta}$ using X and $\boldsymbol{y}$. However, this Bayesian approach requires distribution assumptions about $\boldsymbol{\theta}$ and various underlying independent variables. When considering various sizes of data, [6] provides a framework for computing OLS regression, especially if the data comes in pre-grouped "cells".

There has also been work [1] to study various approaches to linear regression in a single dimensional setting, such as by computing sufficient statistics for least squares linear regression or by finding the median slope between pairs of points in a dataset. Another approach from [31] involves a perturbation of the objective function itself. Instead of modifying the parameters that minimize some objective function, that approach would model the objective function using polynomials, and then perturb the coefficients in that polynomial representation of the objective function. Then, the algorithm returns the argument that minimizes this perturbed objective function, and it can be proven that the overall algorithm is differentially private. In this paper, we largely focus only on algorithms that release statistics privately to combine together, but it is clear that there are different techniques in differential privacy to compute a best fit line. Moving beyond linear regression, more Bayesian work was conducted in [18], where hyperparameters of a wider class of machine learning models are considered. In particular, their work with hyperparameters aims to find differentially private techniques to release the optimal hyperparameter and achieved accuracy of a model, and their techniques hinge on either a Gaussian process assumption or on Lipschitz and convexity assumptions.

The most direct influence on this paper comes from Wang's work with differentially private ridge regression techniques [26]. In his paper, Wang introduces differentially private algorithms that privately release the eigenvalue of the covariance matrix $X^T X$ of the data to be used to set the ridge regression hyperparameter. One such algorithm is Wang's adaSSP algorithm, which divides its privacy-loss budget between the release of an eigenvalue and the release of other statistics for ridge regression. In this paper, we will use Wang's adaSSP algorithm and study how it performs when its privacy-loss budget is altered from its current fixed setting.

In many differentially private linear regression techniques, the covariance matrix of the data is privately released. This paper will focus on the Analyze Gauss approach, discussed by Dwork [10]. The Analyze Gauss approach, as we shall see, simply adds symmetric Normal noise to $X^T X$. Sheffet discusses another approach to release the covariance matrix [20], the Johnson-Lindenstrauss Transform (JLT). In his paper, Sheffet proposes a new differentially private ridge regression mechanism which uses this JLT. Broadly speaking, the JLT will take as input some dimension r and a matrix A, generate a matrix R with r rows where each entry is from an independent Normal distribution, and output $\frac{1}{r}(RA)^T(RA)$ (which is supposed to approximate $A^T A$). This approach is a notably different way to generate a second moment of a matrix without just adding noise to the output. The JLT approach was proven to be differentially private in [3], with particular focus to a graph theory question involving cuts.

4 Algorithms

Two of the main algorithms presented, SSP and adaSSP, have been introduced elsewhere and have been studied theoretically and experimentally. We include the proofs of DP here for completeness and so that the proofs of DP for the novel algorithms, such as adaSSPbudget, make sense in context of these other algorithms. This paper includes detailed proofs of the $l2$ sensitivity of various releases within the SSP and adaSSP algorithms.

4.1 Existing Algorithms

4.1.1 SSP

Sufficient Statistics Perturbation (SSP) [25] [13] is one method of privately computing ridge regression, where we privately release two parts of the ridge regression formula (Proposition 2) with $\lambda = 1$ using the above Gaussian mechanism (Theorem 7). As we shall discuss, this algorithm is meant to represent an "ordinary least squares approach" to regression, even if we are technically using ridge regression with a fixed hyperparameter.

Algorithm 1: SSP

Input: $\epsilon \in \mathbb{R}_{>0}, \delta \in (0,1], X \in \mathcal{X}^n, \boldsymbol{y} \in \mathcal{Y}^n$ for $\mathcal{X} \subset \mathbb{R}^d, \mathcal{Y} \subset \mathbb{R}, ||\mathcal{X}|| \in \mathbb{R}_{\geq 0}, ||\mathcal{Y}|| \in \mathbb{R}_{\geq 0}$

1. Set $Z_{sym} \in \mathbb{R}^{d \times d}$ as a symmetric matrix where every element from the upper triangular matrix is distributed independently as $\mathcal{N}(0,1)$.
2. Privately release $\widehat{X^TX} := X^TX + \frac{\sqrt{2\log(4/\delta)} \cdot ||\mathcal{X}||^2}{\epsilon/2} Z_{sym}$
3. Set $\boldsymbol{z}_r \sim \mathcal{N}(0, I_d)$
4. Privately release $\widehat{X^T\boldsymbol{y}} = X^T\boldsymbol{y} + \frac{\sqrt{2\log(4/\delta)} \cdot ||\mathcal{X}|| \cdot ||\mathcal{Y}||}{\epsilon/2} \boldsymbol{z}_r$
5. Set $\widehat{\boldsymbol{\theta}} := (\widehat{X^TX} + I)^{-1} \widehat{X^T\boldsymbol{y}}$

Result: $\widehat{\boldsymbol{\theta}}$

Lemma 8. *SSP is (ϵ, δ)-DP.*

Before proving that SSP is (ϵ, δ)-DP, we must first reason about the $l2$ sensitivity of the releases within the algorithm:

Lemma 9. *The $l2$ sensitivity of $X \mapsto X^TX$ is $||\mathcal{X}||^2$, and thus Step 2 of SSP is $(\epsilon/2, \delta/2)$-DP.*

Proof. **Proof of $l2$ sensitivity of X^TX**

Let $f_1(X) := X^TX$. Consider adjacent datasets $X \in \mathbb{R}^{n \times d}, X' \in \mathbb{R}^{(n+1) \times d}$, where X' has an additional row entry $\boldsymbol{v} \in \mathcal{X} \subset \mathbb{R}^d$. Note that the Frobenius norm $|| \cdot ||_F$ treats a matrix as if we are performing the L2 norm upon its values, treated as a single vector.

We can see that

$$f_1(X') - f_1(X) = \boldsymbol{v}^T\boldsymbol{v} \in \mathbb{R}^{d \times d}$$

Then, we find that

$$||f_1(X') - f_1(X)||_F^2 = ||\boldsymbol{v}^T\boldsymbol{v}||_F^2 = \sum_{1 \leq i,j \leq d} (v_i v_j)^2 = \left(\sum_{1 \leq i \leq d} v_i^2 \right)^2 = (||\boldsymbol{v}||_2^2)^2$$

$$\Rightarrow \Delta_{f_1} = \sup_{X \sim X'} ||f_1(X') - f_1(X)||_F = \sup_{\boldsymbol{v} \in \mathcal{X}} ||\boldsymbol{v}||_2^2 = ||\mathcal{X}||^2$$

where, as we recall, $||\mathcal{X}|| = \sup_{\boldsymbol{v} \in \mathcal{X}} ||\boldsymbol{v}||_2$ is the greatest possible norm of a row vector.

Therefore, the $l2$ sensitivity of f_1 is $||\mathcal{X}||^2$, so step 2 of the SSP algorithm is $(\epsilon/2, \delta/2)$-DP by the Gaussian mechanism, Lemma 7.

Technically, we are not adding $d \times d$ independent normal variables to $f_1(X)$, given that we add a symmetric normal matrix (in step 2 of SSP, for example), so we only add $\frac{d(d+1)}{2}$ independent normal variables. This is justified because $f_1(X) = X^T X$ is also a symmetric matrix, so we could restrict the output of f_1 to just the upper triangular entries: it is clear that the $l2$ sensitivity of the upper triangular entries are upper bound by the $l2$ sensitivity of the entire matrix, for

$$\sum_{1 \leq i \leq j \leq d} (v_i v_j)^2 \leq \sum_{1 \leq i,j \leq d} (v_i v_j)^2 \leq (||\mathcal{X}||^2)^2$$

As we see, we actually satisfy a potentially tighter $l2$ sensitivity with this symmetric matrix, because we can ignore everything below the diagonal, for example:

$$\sum_{1 \leq i \leq j \leq d} (v_i v_j)^2 \leq \frac{(||\boldsymbol{v}||_2^2)^2 + \sum_{j=1}^d v_j^4}{2} \leq \frac{(||\mathcal{X}||^2)^2 + \sum_{j=1}^d v_j^4}{2}$$

Thus, if you had some bound on the $L4$ norm of row vectors in $\mathcal{X}$, you could have a stronger overall $l2$ sensitivity. Note that in the worst case, the $l2$ sensitivity is still the same, even if we use this symmetric matrix. To see this worst case behavior, consider if $d = 2$ and we had vector $\boldsymbol{v} = [||\mathcal{X}||, 0]$. In this case, $\sum_{1 \leq i \leq j \leq d} (v_i v_j)^2 = \sum_{1 \leq i,j \leq d} (v_i v_j)^2 = (||\mathcal{X}||^2)^2$, so no improvement could be made on the $l2$ bound without stronger conditions on the elements of $\mathcal{X}$.

□

Lemma 10. *The $l2$ sensitivity of $(X, \boldsymbol{y}) \mapsto X^T y$ is $||\mathcal{X}|| \cdot ||\mathcal{Y}||$, and hence Step 4 of SSP is $(\epsilon/2, \delta/2)$-DP.*

Proof. **Proof of $l2$ sensitivity of $X^T \boldsymbol{y}$**

Let $f_2(X, y) := X^T \boldsymbol{y}$. Again, suppose that we have some dataset $X \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \mathbb{R}^n$, and adjacent datasets $X' \in \mathbb{R}^{(n+1) \times d}, \boldsymbol{y}' \in \mathbb{R}^{n+1}$, where $\boldsymbol{v} \in \mathcal{X} \subset \mathbb{R}^d$ is the new row, and $w \in \mathcal{Y} \subset \mathbb{R}$ is the new value in $\boldsymbol{y}'$. We then find that

$$f_2(X', \boldsymbol{y}') - f_2(X, \boldsymbol{y}) = X'^T \boldsymbol{y}' - X^T \boldsymbol{y} = w\boldsymbol{v} \in \mathbb{R}^d$$

As a result, it follows that

$$||f_2(X', \boldsymbol{y}') - f_2(X, \boldsymbol{y})||_2 = ||w\boldsymbol{v}||_2 = |w| \cdot ||\boldsymbol{v}||_2$$

$$\Rightarrow \Delta_{f_2} = \sup_{(X, \boldsymbol{y}) \sim (X', \boldsymbol{y}')} ||f_2(X', \boldsymbol{y}') - f_2(X, \boldsymbol{y})||_2 = \sup_{w \in \mathcal{Y}, \boldsymbol{v} \in \mathcal{X}} |w| \cdot ||\boldsymbol{v}||_2 = ||\mathcal{Y}|| \cdot ||\mathcal{X}||$$

$$\Rightarrow \Delta_{f_2} = ||\mathcal{Y}|| \cdot ||\mathcal{X}||$$

where, as we recall, $||\mathcal{X}|| = \sup_{\boldsymbol{x} \in \mathcal{X}} ||\boldsymbol{x}||_2$ and $||\mathcal{Y}|| = \sup_{y \in \mathcal{Y}} |y|$. □

Proof. **Proof that SSP is (ϵ, δ)-DP**

Lemma 8 follows from Lemmas 9 and 10 by the Composition of DP Theorem 4 and Post-Processing of DP Mechanisms Theorem 5, which tell us that the release of $\widehat{\boldsymbol{\theta}}$ is overall (ϵ, δ)-DP.

□

Note that for numerical stability, SSP actually adds the identity matrix I to $\widehat{X^TX}$ to help ensure that $\widehat{X^TX}$ had an inverse when computing $\widehat{\boldsymbol{\theta}}$. Observe that the eigenvalues of $A + I$ are precisely the eigenvalues of A plus 1, which we can see in the following argument: λ is an eigenvalue of A if and only if $\det(A - \lambda I) = 0 = \det((A + I) - (\lambda + 1)I)$, and λ is an eigenvalue of $(A + I)$ if and only if $\det((A + I) - \lambda I) = 0 = \det(A - (\lambda - 1)I)$. Thus, if X^TX plus a random matrix was not invertible, adding the identity matrix could help to remove zero eigenvalues. This technique of adding a multiple of the identity matrix to allow for numerical stability was one of the motivations for ridge regression [22] [16] in the first place. Here, SSP adheres to a ridge regression with hyperparameter equal to 1 (Proposition 2). All other algorithms studied in this paper (adaSSP, adaSSPbudget, constSSPfull) will use a ridge regression hyperparameter that is at least 1, so SSP is the closest among the studied algorithms to ordinary least squares (which would have a corresponding ridge regression hyperparameter of 0).

This convention of adding an identity matrix to X^TX was adopted throughout Wang's [26] implemented code, so we are using it here for consistency. Note that this added constant value does not change the theoretical properties that are proven in this paper. Observe in the later adaptive algorithms, such as adaSSP and adaSSPbudget, that the minimum eigenvalue computed is also of $X^TX + I$, which just corresponds to adding 1 to the minimum eigenvalue of X^TX. Future work could explore what happens when the identity matrix is not added to X^TX inside of these algorithms, where SSP would adhere to a literal ordinary least squares formula (Proposition 1). This cleaner algorithm would be more in line with ordinary least squares, but perhaps the algorithms would have more failure states if $\widehat{X^TX}$ were not invertible. As we discuss in the experiments, failures (such as a non-invertible matrix) return $\widehat{\boldsymbol{\theta}} = \mathbf{0}$, but with the algorithms as written this is an extremely rare event.

4.1.2 adaSSP

Wang [26] proposes a way to use ridge regression where the hyperparameter is chosen based on the smallest eigenvalue of X^TX. In particular, this is a variant of SSP with adaptive dampening.

Algorithm 2: adaSSP

Input: $\epsilon \in \mathbb{R}_{>0}, \delta \in (0, 1], X \in \mathcal{X}^n, \boldsymbol{y} \in \mathcal{Y}^n$ for $\mathcal{X} \subset \mathbb{R}^d, \mathcal{Y} \subset \mathbb{R}, ||\mathcal{X}|| \in \mathbb{R}_{\geq 0}, ||\mathcal{Y}|| \in \mathbb{R}_{\geq 0}, \rho \in (0, 1)$

1. Calculate the minimum eigenvalue $\lambda_{min}(X^TX + I)$
2. Set $z_e \sim \mathcal{N}(0, 1)$
3. Privately release $\tilde{\lambda}_{min} = \max\left\{\lambda_{min}(X^TX + I) + \frac{\sqrt{2\log(6/\delta)}}{\epsilon/3}||\mathcal{X}||^2 z_e - \frac{2\log(6/\delta)}{\epsilon/3}||\mathcal{X}||^2, 0\right\}$
4. Set $\lambda = \max\left\{0, \frac{\sqrt{2d\log(6/\delta)\log(2d^2/\rho)}\cdot||\mathcal{X}||^2}{\epsilon/3} - \tilde{\lambda}_{min}\right\}$
5. Set $Z_{sym} \in \mathbb{R}^{d\times d}$ as a symmetric matrix where every element from the upper triangular matrix is distributed independently as $\mathcal{N}(0, 1)$.
6. Privately release $\widehat{X^TX} := X^TX + \frac{\sqrt{2\log(6/\delta)}\cdot||\mathcal{X}||^2}{\epsilon/3} Z_{sym}$
7. Set $\boldsymbol{z}_r \sim \mathcal{N}(0, I_d)$
8. Privately release $\widehat{X^T\boldsymbol{y}} = X^T\boldsymbol{y} + \frac{\sqrt{2\log(6/\delta)}\cdot||\mathcal{X}||\cdot||\mathcal{Y}||}{\epsilon/3}\boldsymbol{z}_r$
9. Set $\widehat{\boldsymbol{\theta}} := (\widehat{X^TX} + (\lambda + 1)I)^{-1}\widehat{X^T\boldsymbol{y}}$

Result: $\widehat{\boldsymbol{\theta}}$

Lemma 11. *adaSSP is (ϵ, δ)-DP.*

Before proving that adaSSP is DP, we must argue about the $l2$ sensitivity of the eigenvalue $\lambda_{min}(X^T X)$, which will require some technical arguments about eigenvalues and matrix norms.

Lemma 12. $\Big($***Weyl's eigenvalue inequality***$\Big)$ *Let B and B' be symmetric real matrices in $\mathbb{R}^{d\times d}$, and let $\lambda_j(B), j = 1, \cdots, d$ (resp. $\lambda_j(B')$) denote the eigenvalues of B (resp. B'), with $\lambda_1(B) \geq \cdots \geq \lambda_d(B)$. Then*

$$\lambda_{i+j-1}(B + B') \leq \lambda_i(B) + \lambda_j(B')$$

whenever $i, j \geq 1, i + j - 1 \leq n$.

Proof found in [7].

Lemma 13. $\Big($***Relation of matrix norms***$\Big)$ *If $||\cdot||_F$ is the Frobenius norm of a matrix (the square root of the sum of the squared entries of the matrix), and $||\cdot||_2$ is defined for matrix $A \in \mathbb{R}^{n\times d}$ as*

$$||A||_2 = \max_{\boldsymbol{x}\in\mathbb{R}^d, ||\boldsymbol{x}||_2=1} ||A\boldsymbol{x}||_2$$

then

$$||A||_2 \leq ||A||_F$$

Note that the Frobenius norm is the same norm that we have been using on matrices earlier in this paper (ex. in proving $l2$ sensitivity of $X^T X$ in Lemma 9).

Proof. **Proof of relation of matrix norms**
Let $A \in \mathbb{R}^{n\times d}, \boldsymbol{x} \in \mathbb{R}^d$. Then

$$||A\boldsymbol{x}||_2^2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{d} A_{i,j} x_j \right)^2$$

Let $\boldsymbol{A}_{j,:} \in \mathbb{R}^d$ be the jth row vector of A. Then the Cauchy-Schwarz inequality gives that

$$\begin{aligned}
||A\boldsymbol{x}||_2^2 &= \sum_{i=1}^{n} \left(\sum_{j=1}^{d} A_{i,j} x_j \right)^2 \\
&= \sum_{i=1}^{n} \langle \boldsymbol{A}_{i,:}, \boldsymbol{x} \rangle^2 \\
&\leq \sum_{i=1}^{n} \langle \boldsymbol{A}_{i,:}, \boldsymbol{A}_{i,:} \rangle \langle \boldsymbol{x}, \boldsymbol{x} \rangle \\
&= \langle \boldsymbol{x}, \boldsymbol{x} \rangle \sum_{i=1}^{n} \sum_{j=1}^{d} A_{i,j}^2 \\
&= \langle \boldsymbol{x}, \boldsymbol{x} \rangle \cdot ||A||_F^2 \\
&= ||\boldsymbol{x}||_2^2 \cdot ||A||_F^2
\end{aligned}$$

Therefore,

$$||A||_2 = \max_{\boldsymbol{x}\in\mathbb{R}^d, ||\boldsymbol{x}||_2=1} ||A\boldsymbol{x}||_2 \leq (1) \cdot ||A||_F = ||A||_F$$

as desired. □

With these results, we can prove an upper bound on the $l2$ sensitivity of $\lambda_{min}(X^TX)$.

Lemma 14. *The $l2$ sensitivity of $\lambda_{min}(X^TX)$ is upper bound by $||\mathcal{X}||^2$*

Proof. **Proof of $l2$ sensitivity of $\lambda_{min}(X^TX)$**

Let $f_3(X) := \lambda_{min}(X^TX)$. Consider adjacent datasets $X \in \mathbb{R}^{n\times d}, X' \in \mathbb{R}^{(n+1)\times d}$, where X' has an additional row entry $\boldsymbol{v} \in \mathcal{X} \subset \mathbb{R}^d$.

In the language of Weyl's inequality (Lemma 12), set

$$B = X^TX, \quad B' = X'^TX' - X^TX, \quad i = d, j = 1$$

Thus,

$$\lambda_{min}(X'^TX') - \lambda_{min}(X^TX) \leq \lambda_{max}(X'^TX' - X^TX)$$

By symmetry, it follows that

$$\lambda_{min}(X^TX) - \lambda_{min}(X'^TX') \leq \lambda_{max}(X^TX - X'^TX')$$

It is either the case that $\lambda_{min}(X'^TX') - \lambda_{min}(X^TX) \geq 0$, or $\lambda_{min}(X^TX) - \lambda_{min}(X^TX) \geq 0$ (or possibly both, at 0). Thus,

$$|\lambda_{min}(X'^TX') - \lambda_{min}(X^TX)| \leq \max\Big\{|\lambda_{max}(X^TX - X'^TX')|, |\lambda_{max}(X'^TX' - X^TX)|\Big\}$$

Because the set of eigenvalues of $X^TX - X'^TX'$ is precisely the additive inverses of the set of eigenvalues of $X'^TX' - X^TX$, we have

$$\begin{aligned}|\lambda_{min}(X'^TX') - \lambda_{min}(X^TX)| &\leq \max\Big\{|\lambda_{max}(X'^TX' - X^TX)|, |\lambda_{min}(X'^TX' - X^TX)|\Big\} \\ &\leq \max_{\boldsymbol{x}\in\mathbb{R}^d, ||\boldsymbol{x}||_2=1} ||(X'^TX' - X^TX)\boldsymbol{x}||_2\end{aligned}$$

Thus

$$|f_3(X) - f_3(X')| \leq \max_{\boldsymbol{x}\in\mathbb{R}^d, ||\boldsymbol{x}||_2=1} ||(X'^TX' - X^TX)\boldsymbol{x}||_2 \leq ||X'^TX' - X^TX||_F$$

where this last inequality comes from Lemma 13. Hence,

$$\Delta_{f_3} = \sup_{X\sim X'} |f_3(X) - f_3(X')| \leq ||X'^TX' - X^TX||_F$$

$$\Rightarrow \Delta_{f_3} \leq ||\mathcal{X}||^2$$

where this last inequality is justified from the $l2$ sensitivity of $X \mapsto X^TX$, $||\mathcal{X}||^2$ (Lemma 9). □

Now that there are bounds on the $l2$ sensitivity of the eigenvalue release, we can prove the main result about adaSSP.

Proof. **Proof that adaSSP is (ϵ, δ)-DP.**

Like with SSP, this adaSSP algorithm can be seen as 3 releases, and we can prove that each release is $(\epsilon/3, \delta/3)$-DP.

Based on the $l2$ sensitivity of $\lambda_{min}(X^TX)$ (Lemma 14) it follows that the mechanism that releases the quantity

$$\lambda_{min}(X^TX + I) + \frac{\sqrt{2\log(6/\delta)}}{\epsilon/3}||\mathcal{X}||^2 z_e = \lambda_{min}(X^TX) + 1 + \frac{\sqrt{2\log(6/\delta)}}{\epsilon/3}||\mathcal{X}||^2 z_e$$

is $(\epsilon/3, \delta/3)$-DP by the Gaussian mechanism (and by Post Processing Theorem 5, where adding a 1 does not change the DP property). It then follows that the λ value from step 4 is likewise $(\epsilon/3, \delta/3)$-DP by post

processing (the other values are all constants, none depend on the data itself: we assume $||\mathcal{X}||$ is fixed and known ahead of time).

The releases in step 6 and 8 are similarly each $(\epsilon/3, \delta/3)$-DP by the same reasoning that was used in SSP (Lemmas 9, 10).
Therefore, the Composition Theorem 4 tells us that the release of these 3 quantities, $\lambda, \widehat{X^TX}, \widehat{X^T\boldsymbol{y}}$, is overall (ϵ, δ)-DP. We can then use the Post Processing Theorem 5 to see that the final release of $\widehat{\boldsymbol{\theta}}$ is still (ϵ, δ)-DP. □

It is clear that adaSSP contains many terms that are not cleanly explained through the Gaussian mechanism bounds or through the formula for the ridge regression coefficients. To understand certain properties of adaSSP and where the constants come from, below we prove the following lemma about the eigenvalue release:

Lemma 15. *Let $\delta' > 0, \epsilon' > 0, Z \sim \mathcal{N}(0,1)$. Then*

$$P\left(\frac{\sqrt{2\log(2/\delta')}}{\epsilon'}||\mathcal{X}||^2 Z - \frac{2\log(2/\delta')}{\epsilon'}||\mathcal{X}||^2 > 0\right) \leq \frac{\delta'}{4\sqrt{\pi \cdot \log(2/\delta')}}$$

In other words, this Lemma 15 asserts that with high probability the eigenvalue release in adaSSP's step 3 is an underestimate. Hence, the value

$$P(\tilde{\lambda}_{min} \leq \lambda_{min}(X^TX + I))$$

is close to 1. We will soon quantity exactly how close to 1 this probability is.

Proof. First, we can rephrase this problem as

$$P\left(\frac{\sqrt{2\log(2/\delta')}}{\epsilon'}||\mathcal{X}||^2 Z - \frac{2\log(2/\delta')}{\epsilon'}||\mathcal{X}||^2 > 0\right) = P(Z - \sqrt{2\log(2/\delta')} > 0) = P(Z > \sqrt{2\log(2/\delta')})$$

At this point, we can turn to a common tail bound for Normal variables, where for $x > 0, Z \sim \mathcal{N}(0,1)$,

$$P(Z > x) \leq \frac{e^{-x^2/2}}{x\sqrt{2\pi}}$$

(Because we know that $\frac{t}{x} \geq 1$ for $t \geq x$,

$$P(Z > x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt \leq \int_x^\infty \frac{t}{x\sqrt{2\pi}} e^{-t^2/2} dt = \frac{e^{-x^2/2}}{x\sqrt{2\pi}}$$

The proof for this Normal tail bound and for more general results of this form can be found at [21].)

With this tail bound, we have

$$\begin{aligned}
P(Z > \sqrt{2\log(2/\delta')}) &\leq \frac{\exp(-2\log(2/\delta')/2)}{\sqrt{2\log(2/\delta')}\sqrt{2\pi}} \\
&\leq \frac{\exp(-2\ln(2/\delta')/2)}{\sqrt{2\log(2/\delta')}\sqrt{2\pi}} \\
&= \frac{1}{(2/\delta')\sqrt{2\log(2/\delta')}\sqrt{2\pi}} \\
&= \frac{\delta'}{4\sqrt{\pi \cdot \log(2/\delta')}}
\end{aligned}$$

Therefore,

$$\Rightarrow P\left(\frac{\sqrt{2\log(2/\delta')}}{\epsilon'}||\mathcal{X}||^2 Z - \frac{2\log(2/\delta')}{\epsilon'}||\mathcal{X}||^2 > 0\right) \leq \frac{\delta'}{4\sqrt{\pi \cdot \log(2/\delta')}}$$

as desired. □

In the context of adaSSP, this Lemma 15 has $\delta' = \delta/3$. In adaSSP and the soon-to-be described adaSSP-budget, this Lemma 15 tells us about the probability that the minimum eigenvalue is an overestimate. As we shall see with the adaSSPbudget algorithm, the greatest such δ' value is no greater than $\frac{1}{300^{1.1}}$ (the smallest dataset size considered has more than 300 entries). We see that this bound monotonically decreases to 0 as δ' goes to zero, so the probability that the minimum eigenvalue released is an overestimate is upper bound by

$$\frac{\frac{1}{300^{1.1}}}{4\sqrt{\pi \cdot \log(2/\frac{1}{300^{1.1}})}} \approx 0.000084 < 10^{-4}$$

As a result, the eigenvalue release in adaSSP (and, as we will see, in adaSSPbudget) is almost always an underestimate. Following through adaSSP, this eigenvalue underestimate corresponds to a larger λ hyperparamater used.

Parsing the other elements of adaSSP, the value ρ that appears in line 4 has to do with a claim [26] that with probability $1-\rho$, a bound can be made on the difference between $L(\boldsymbol{\theta}^*)$ and $L(\widehat{\boldsymbol{\theta}})$. In particular, Wang proves the following (if $X^TX + I$ is replaced by X^TX in the adaSSP algorithm):

Proposition 16. *[26] Assuming $||\mathcal{Y}|| \lesssim ||\mathcal{X}|| \cdot ||\boldsymbol{\theta}^*||$, with probability at least $1-\rho$,*

$$L(\widehat{\boldsymbol{\theta}}) - L(\boldsymbol{\theta}^*) \leq O\Big(\max\Big\{\frac{\sqrt{d\log(\frac{d^2}{\rho})} \cdot ||\mathcal{X}||^2 \cdot ||\boldsymbol{\theta}^*||^2}{\epsilon/\sqrt{\log(\frac{6}{\rho})}}, \frac{||\mathcal{X}||^4 \cdot ||\boldsymbol{\theta}^*||^2 tr[(X^TX)^{-1}]}{\epsilon^2/(\log(\frac{6}{\rho})\log(\frac{d^2}{\rho}))}\Big\}\Big)$$

where $\widehat{\boldsymbol{\theta}}$ is the output of adaSSP, and $\boldsymbol{\theta}^*$ is the non-private linear regression output.

Proof. Proof is omitted here, but it can be found in [26]. □

Because the non-private linear regression value $\boldsymbol{\theta}^*$ is defined as the value achieving the minimum of L, $L(\widehat{\boldsymbol{\theta}}) - L(\boldsymbol{\theta}^*)$ in this theorem is nonnegative. Proving this Proposition 16 requires various technical results which we will not fully explore in this paper. In broad terms, the proof first asserts that for any $\boldsymbol{\theta} \in \mathbb{R}^d$,

$$L(\boldsymbol{\theta}) - L(\boldsymbol{\theta}^*) = (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T X^T X(\boldsymbol{\theta} - \boldsymbol{\theta}^*) = ||\boldsymbol{\theta} - \boldsymbol{\theta}^*||^2_{X^TX}$$

using a Taylor expansion of $L(\boldsymbol{\theta}) = ||\boldsymbol{y} - X\boldsymbol{\theta}||_2^2$ at $\boldsymbol{\theta}^*$. With this result, the proof creates upper bounds on $||\boldsymbol{\theta} - \boldsymbol{\theta}^*||^2_{X^TX}$ using quantities involving X^TX. Next, $\boldsymbol{\theta}$ is replaced by the random output of adaSSP $\widehat{\boldsymbol{\theta}}$, so the upper bound involves various high probability bounds on random symmetric matrices. At the end, there is a bound on $L(\widehat{\boldsymbol{\theta}}) - L(\boldsymbol{\theta}^*)$ which involves the true value of $\lambda_{min}(X^TX)$ and the value of λ chosen by adaSSP. In order to minimize the upper bound on $L(\widehat{\boldsymbol{\theta}}) - L(\boldsymbol{\theta}^*)$ without knowing ahead of time the true $\lambda_{min}(X^TX)$, Wang uses results like the bound on the probability that the released eigenvalue is an overestimate of $\lambda_{min}(X^TX)$. Additionally, various additive constants are included, like $\frac{\sqrt{2d\log(6/\delta)\log(2d^2/\rho)} \cdot ||\mathcal{X}||^2}{\epsilon/3}$, because they are the functions of $\epsilon, \delta, ||\mathcal{X}||, \mathcal{Y}||, \rho$ (up to constant factor) that happen to help minimize this upper bound on $L(\widehat{\boldsymbol{\theta}}) - L(\boldsymbol{\theta}^*)$.

At a high level, the various maximum operations applied to the minimum eigenvalue $\lambda_{min}(X^TX)$ are meant to sanitize the eventual λ parameter in the ridge regression. Invertibility of $(\widehat{X^TX} + (\lambda+1)I)^{-1}$ could

break down if $\lambda < 0$, for example. The reason that λ shrinks as λ_{min} grows (up to a point) is that if $X^T X$ is suitably well conditioned (with little correlation in the data), then its smallest eigenvalue is not too close to zero; therefore, in this "good" condition, $\tilde{\lambda}_{min}$ would be larger, so λ would be smaller (or 0). If $\lambda = 0$, then $\widehat{\boldsymbol{\theta}}$ reduces to SSP (but with different privacy-loss parameters for the released quantities), which relies on ridge regression with a small hyperparameter. On the other hand, if $X^T X$ has a small eigenvalue, then we may believe that the data has correlations that ridge regression with larger hyperparameters can address. In this case, λ is likely large when $\lambda_{min}(X^T X)$ is small, so the hyperparameter in the ridge regression plays a larger role in handling covariances.

4.2 New Algorithms

In this paper, we propose and evaluate two new variants of adaSSP, where the privacy-loss budget is not necessarily evenly split between the three privately released quantities. In particular, the privacy-loss budget is divided to go to the release of $\tilde{\lambda}_{min}$, or to the releases of $\widehat{X^T X}, \widehat{X^T \boldsymbol{y}}$ (with these two quantities receiving equal privacy-loss budget). The motivation for this split of the budget was to compare the "necessary" parts of linear regression, such as the release of $X^T X$ and $X^T \boldsymbol{y}$, with the "unnecessary" part that had to do with the hyperparameter of ridge regression, like the release of the minimum eigenvalue of $X^T X$. The parameter γ, an element of $(0, 1)$, partitions the privacy-loss budget. γ is directly proportional to how much of the privacy-loss budget goes to the release of the eigenvalue.

4.2.1 adaSSPbudget

Algorithm 3: adaSSPbudget

Input: $\epsilon \in \mathbb{R}_{>0}, \delta \in (0, 1], \gamma \in (0, 1), X \in \mathcal{X}^n, \boldsymbol{y} \in \mathcal{Y}^n$ for $\mathcal{X} \subset \mathbb{R}^d, \mathcal{Y} \subset \mathbb{R}, ||\mathcal{X}|| \in \mathbb{R}_{\geq 0}, ||\mathcal{Y}|| \in \mathbb{R}_{\geq 0}, \rho \in (0, 1)$

1. Set $\epsilon_{hyp} = \gamma\epsilon, \delta_{hyp} = \gamma\delta$.
2. Calculate the minimum eigenvalue $\lambda_{min}(X^T X + I)$
3. Set $z_e \sim \mathcal{N}(0, 1)$
4. Privately release $\tilde{\lambda}_{min} = \max\left\{\lambda_{min}(X^T X + I) + \frac{\sqrt{2\log(2/\delta_{hyp})}}{\epsilon_{hyp}}||\mathcal{X}||^2 z_e - \frac{2\log(2/\delta_{hyp})}{\epsilon_{hyp}}||\mathcal{X}||^2, 0\right\}$
5. Set $\lambda = \max\left\{0, \frac{\sqrt{2d\log(2/\delta_{nec})\log(2d^2/\rho)}\cdot||\mathcal{X}||^2}{\epsilon_{nec}} - \tilde{\lambda}_{min}\right\}$
6. Set $\epsilon_{nec} = \frac{(1-\gamma)}{2}\epsilon, \delta_{nec} = \frac{(1-\gamma)}{2}\delta$
7. Set $Z_{sym} \in \mathbb{R}^{d\times d}$ as a symmetric matrix where every element from the upper triangular matrix is independently distributed as $\mathcal{N}(0, 1)$.
8. Privately release $\widehat{X^T X} := X^T X + \frac{\sqrt{2\log(2/\delta_{nec})}\cdot||\mathcal{X}||^2}{\epsilon_{nec}} Z_{sym}$
9. Set $\boldsymbol{z}_r \sim \mathcal{N}(0, I_d)$
10. Privately release $\widehat{X^T \boldsymbol{y}} = X^T \boldsymbol{y} + \frac{\sqrt{2\log(2/\delta_{nec})}\cdot||\mathcal{X}||\cdot||\mathcal{Y}||}{\epsilon_{nec}} \boldsymbol{z}_r$
11. Set $\widehat{\boldsymbol{\theta}} := (\widehat{X^T X} + (\lambda + 1)I)^{-1}\widehat{X^T \boldsymbol{y}}$

Result: $\widehat{\boldsymbol{\theta}}$

Lemma 17. *adaSSPbudget is* (ϵ, δ)*-DP.*

Proof. **Proof that adaSSPbudget is (ϵ, δ)-DP.**

Like the adaSSP algorithm, this adaSSPbudget algorithm consists of 3 differentially private releases, and each release is differentially private by similar reasoning as with adaSSP. This time, instead of an even split of $(\epsilon/3, \delta/3)$ between the three releases, we have that the release of $\tilde{\lambda}_{min}$ had a privacy-loss budget of $(\gamma\epsilon, \gamma\delta)$, and the releases of $\widehat{X^TX}$ and $\widehat{X^T\boldsymbol{y}}$ are each $(\frac{1-\gamma}{2}\epsilon, \frac{1-\gamma}{2}\delta)$-DP. Again, based on the Composition Theorem 4 and Post Processing Theorem 5, the adaSSPbudget algorithm is (ϵ, δ)-differentially private ($\gamma + 2\frac{1-\gamma}{2} = 1$). □

Notice that if $\gamma = 1/3$, then adaSSPbudget reduces to adaSSP. However, it is not the case that as γ goes to 0, adaSSPbudget goes to SSP (which would correspond to the ridge regression hyperparameter coefficient being 1): one main difference is that adaSSPbudget (and adaSSP) use a ridge regression parameter $\lambda + 1$, where λ is only partly determined by the release of $\lambda_{min}(X^TX)$. Indeed, if γ goes to 0, then the $\epsilon_{hyp}, \delta_{hyp}$ values used for the release of $\lambda_{min}(X^TX)$ go to zero, which causes for high instability in the release of $\tilde{\lambda}_{min}$ (step 4) which is truncated at 0. Then, the ridge regression parameter λ may still just use the constant value within step 5 (which in particular does not set λ to be 0 in the ridge regression algorithm). This constant in step 5 shrinks down to some nonzero constant as γ goes to zero (because $\epsilon_{nec}, \delta_{nec}$ will increase to ϵ, δ) which depends on $\epsilon, \delta, d, \rho, ||\mathcal{X}||$. As γ goes to 1, this constant in step 5 goes to infinity, because $\epsilon_{nec}, \delta_{nec}$ will shrink to zero. Noticing how often λ is equal to this constant value in step 5 of adaSSPbudget, or how often λ is set to 0, may help us understand the circumstances when adaSSPbudget outperforms SSP. We shall explore this behavior in Figure 4.

4.2.2 constSSPfull

In order to examine the behavior of adaSSPbudget, we introduce constSSPfull below, which ignores the eigenvalue calculations of adaSSPbudget, and instead defaults to the constant value from step 5 of adaSSPbudget (which corresponds to if the released value $\tilde{\lambda}_{min}$ were 0).

Algorithm 4: constSSPfull

$\epsilon \in \mathbb{R}_{>0}, \delta \in (0, 1], X \in \mathcal{X}^n, \boldsymbol{y} \in \mathcal{Y}^n$ for $\mathcal{X} \subset \mathbb{R}^d, \mathcal{Y} \subset \mathbb{R}, ||\mathcal{X}|| \in \mathbb{R}_{\geq 0}, ||\mathcal{Y}|| \in \mathbb{R}_{\geq 0}, \rho \in (0, 1)$

1. Set $\epsilon_{nec} = \frac{1}{2}\epsilon, \delta_{nec} = \frac{1}{2}\delta$
2. Set $\lambda = \frac{\sqrt{2d\log(2/\delta_{nec})\log(2d^2/\rho)}||\mathcal{X}||^2}{\epsilon_{nec}}$
3. Set $Z_{sym} \in \mathbb{R}^{d\times d}$ as a symmetric matrix where every element from the upper triangular matrix is independently distributed as $\mathcal{N}(0, 1)$.
4. Privately release $\widehat{X^TX} := X^TX + \frac{\sqrt{2\log(2/\delta_{nec})}||\mathcal{X}||^2}{\epsilon_{nec}} Z_{sym}$
5. Set $\boldsymbol{z}_r \sim \mathcal{N}(0, I_d)$
6. Privately release $\widehat{X^T\boldsymbol{y}} = X^T\boldsymbol{y} + \frac{\sqrt{2\log(2/\delta_{nec})}\cdot||\mathcal{X}||\cdot||\mathcal{Y}||}{\epsilon_{nec}}\boldsymbol{z}_r$
7. Set $\widehat{\boldsymbol{\theta}} := (\widehat{X^TX} + (\lambda + 1)I)^{-1}\widehat{X^T\boldsymbol{y}}$

Result: $\widehat{\boldsymbol{\theta}}$

Lemma 18. *constSSPfull is (ϵ, δ)-DP.*

Proof. **Proof of constSSPfull is (ϵ, δ)-DP.**

This proof follows similarly as for adaSSPbudget. Now that the ϵ, δ values used in the releases of X^TX and $X^T\boldsymbol{y}$ are full (i.e. $\epsilon_{nec} = \frac{\epsilon}{2}$ and not $\epsilon_{nec} = \frac{1-\gamma}{2}\epsilon$), each of these two releases is $(\frac{\epsilon}{2}, \frac{\delta}{2})$-DP (Lemmas 9, 10). Thus, from Composition Theorem 4 and Post-Processing Theorem 5, the overall mechanism is (ϵ, δ)-DP. □

Note that constSSPfull does not take γ as an input. No privacy-loss budget is spent on an eigenvalue release.

This algorithm is proposed so that, experimentally, we can see if or when the release of the eigenvalue is worth the privacy-loss budget that could go to the releases of X^TX and $X^T\boldsymbol{y}$. In other words, maybe the default constant used is all that is useful about adaSSPbudget. This constSSPfull also stands in as a proxy for an algorithm which would compute ridge regression (with some standard large ridge regression hyperparameter), and we can compare this algorithm to SSP, which we might think of as a standard for an ordinary least squares approach. Yes, it is true that the implementation of SSP includes a ridge regression parameter of 1, but given that the other algorithms adaSSP, adaSSPbudget, constSSPfull all have a ridge regression hyperparameter of at least 1, this SSP is the "closest" to the ordinary least squares defined in Proposition 1.

5 Experimental Outline

The goal of this experimental work is to understand the role of dividing the privacy-loss budget in adaSSP-budget between the eigenvalue release, used for the ridge regression parameter, and the other regression components ($\widehat{X^TX}$ and $\widehat{X^T\boldsymbol{y}}$). Throughout these discussions, the "necessary" components of ridge regression are the $\widehat{X^TX}$ and $\widehat{X^T\boldsymbol{y}}$ releases, and the "hyperparameter" component is the released minimum eigenvalue of X^TX used in the ridge regression hyperparameter.

All code used for these experiments can be found on GitHub at `https://github.com/TPiazza21/Thesis`.

5.1 Motivating Questions

Below is a list of motivating questions that guide the experiments and the surrounding discussion. The finer questions are grouped into broader categories, concerning the sensitivity of the performance to the setting of γ, the comparisons of the various algorithms, and the performance of adaSSPbudget in different circumstances.

(I) Is the performance of adaSSPbudget sensitive to the choice of γ, or does adaSSPbudget perform similarly for a wide range of γ?

- (a) Is the choice of $\gamma = 1/3$ (corresponding to adaSSP) a particularly strong choice of γ?
- (b) What happens near the extreme values of $\gamma \in \{0, 1\}$ for adaSSPbudget?

(II) How does adaSSPbudget compare against SSP and constSSPfull?

- (a) When does adaSSPbudget outperform the vanilla SSP algorithm?
- (b) Does the release of the eigenvalue in adaSSPbudget provide a benefit over just using a default ridge regression parameter (like in constSSPfull, or SSP with a smaller hyperparameter of 1)?
- (c) When does adaSSPbudget reduce to the same ridge regression calculation as SSP in the release of $\widehat{\boldsymbol{\theta}}$ (by setting $\lambda = 0$ in step 5 of the algorithm)?

(III) How does the performance of adaSSPbudget change under different data and privacy conditions?

- (a) How does adaSSPbudget perform under different dataset sizes n?
- (b) How does adaSSPbudget perform under different ϵ values?

(c) How does adaSSPbudget perform under synthetic data and UCI data?

There were two types of data used in the experiments, which are referred to as "synthetic data" and "UCI data" (or "real data").

5.2 Synthetic Data

The synthetic data was generated as follows:

1. Set $d = 10$
2. Set $\boldsymbol{\theta}_0 \in \mathbb{R}^d$ as a vector where every entry is independently from $\mathcal{N}(0, 1)$
3. Standardize the $\boldsymbol{\theta}_0$ by dividing by its vector norm, $\boldsymbol{\theta}_0 := \frac{\boldsymbol{\theta}_0}{||\boldsymbol{\theta}_0||_2}$.
4. Repeat the remaining steps for $n \in \{1280, 20480, 327680\}$:
   (a) Set $X \in \mathbb{R}^{n \times d}$ as a matrix where every entry is independently from $\mathcal{N}(0, 1)$.
   (b) Standardize the X such that each row has a vector norm of 1.
   (c) Set $\boldsymbol{Z} \in \mathbb{R}^n$ to be a vector where each element is independently from $\mathcal{N}(0.0, (0.1)^2)$.
   (d) Set
   $$\boldsymbol{y} = X\boldsymbol{\theta}_0 + \boldsymbol{Z}$$

In other words, once $\boldsymbol{\theta}_0$ and X are fixed, the $\boldsymbol{y}$ is distributed as $\mathcal{N}(X\boldsymbol{\theta}_0, (0.1)^2 I_n)$. The point of this data is that there should be no strong correlations between the input dimensions of each row vector of X. This is the problem that (ordinary) linear regression is supposed to solve, so we can analyze what happens when a ridge regression approach is used instead. We might thus expect that SSP would perform better than adaSSPbudget (for most γ values) in this setting.

This setting is useful also because we know ahead of time the true $\boldsymbol{\theta}_0$ value, and so we can directly compare this value to a predicted $\widehat{\boldsymbol{\theta}}$.

5.3 UCI Data

Following Wang [26], we use the UCI (University of California Irvine) datasets [29]. These datasets are often used as a standard machine learning benchmark, and they contain real world data. As an example, the "bike" dataset [23] has 17 attributes like "temperature", "hour", and "humidity", with a response variable of how many bikes were rented on a given day in Maryland.

In contrast to the synthetic data, we would expect UCI data to have correlations within the dataset (like "humidity" and "temperature"), so these datasets may respond better to a ridge regression approach (instead of a linear regression approach). Therefore, we may expect adaSSPbudget to perform better than SSP here. Because we do not have an underlying "true" $\boldsymbol{\theta}_0$ value, we cannot perform a relative efficiency calculation with this dataset.

We arbitrarily chose 3 datasets to explore the behavior of the algorithms on this potentially correlated data. These results are not meant to be exhaustive of all possible UCI datasets, but they are supposed to provide a meaningful alternative to the synthetic data. Future work could include exploring the properties of the other UCI datasets, in particular as the dimension d changes.

5.4 Procedures Taken With Both Datasets

Even with different datasets and different performance metrics, a similar pattern is followed in all cases. The number of cross validation trials is set to 32, with 90% training size and 10% test size. The variable ρ used in the algorithms (such as in adaSSP, adaSSPbudget, and constSSPfull) is set to 0.05.

For both synthetic data and UCI data, we repeat experiments for $\epsilon \in \{0.001, 0.01, 0.1, 1.0\}$ and for various γ values between 0 and 1. In all cases, we set $\delta = 1/n^{1.1}$ (where n is the number of rows in the given dataset). For synthetic data, the performance metric is relative efficiency and prediction error, and for UCI data, the performance metric is just prediction error. The plots for these values represent the mean values between the cross validation trials, with error bars representing the standard deviation across these cross validation trials (note that the error bar is only shown on top). If the y-axis is a log scale, the size of the standard deviation corresponds to the numeric difference between the top and bottom of the standard deviation line according to the y-axis.

The algorithms tested for these two datasets for different ϵ and γ values are adaSSPbudget, constSSPfull, and SSP. Note that SSP and constSSPfull do not change with γ, so their roles in the graph are copied for various γ values. Within adaSSPbudget, step 5 sets the value λ to be used as the ridge regression parameter (and then plus 1). To study the behavior of how the algorithm uses this ridge regression parameter, we count the number of times across the cross validation trials (for a certain ϵ, γ, n combination) that λ was set to zero within the algorithm. This is the plotted "Lambda Zero Proportion" values. As we will explore, there is no such plot for the UCI data because λ was never set to zero for the UCI data in any of our experiments.

6 Results

6.1 Existing Results

One of the motivating experiments for the work done in this thesis is seen in Figure 4 of Wang's paper on adaSSP [26] (represented here, in Figure 2). In his paper, the adaSSP algorithm was introduced as an adaptive and potentially more effective version of SSP. Using similarly defined synthetic and UCI data, Wang found that the adaSSP algorithm outperformed the SSP algorithm in various UCI datasets (Wang's Figure 3, reproduced here as Figure 1).

In this Figure 1, we see that for small ϵ values (less than $\epsilon = 0.1$), adaSSP outperforms SSP on this UCI data. Note that similar results hold for almost all of the 36 UCI datasets that Wang tested, see his Tables 3 and 4 [26]. This dominance of adaSSP could be explained by the fact that this UCI data may have several correlations that adaSSP's eigenvalue release helps to tackle with a ridge regression approach.

However, in Wang's Figure 4 (this paper's Figure 2), SSP outperforms adaSSP on the synthetic data in relative efficiency.

He proposes that this unexpected dominance of SSP over adaSSP in this setting was perhaps due to the privacy-loss budget within the algorithms. After all, adaSSP spends 1/3 of its privacy-loss budget releasing λ_{min}, but SSP wastes none of its privacy-loss budget on a similar calculation. Wang suggests that, perhaps if the privacy-loss budget within adaSSP were different, then the adaptive ridge regression approach may outperform the SSP approach. The purported beauty of the adaSSP algorithm is its adaptive property: it can scale up or down the ridge regression coefficient dynamically with the data. In this situation, the synthetic data was generated such that its datasets were relatively well behaved (compared to the UCI data), so an ordinary least squares approach would probably perform as well as a ridge regression approach. If, however, the cost of having this adaptive property takes away accuracy from the $X^T X$ and $X^T \boldsymbol{y}$ releases, then the eigenvalue release of adaSSP may not be worth the privacy-loss budget cost. Therefore, this thesis aims to explore

Wang's Prediction Error in UCI Datasets

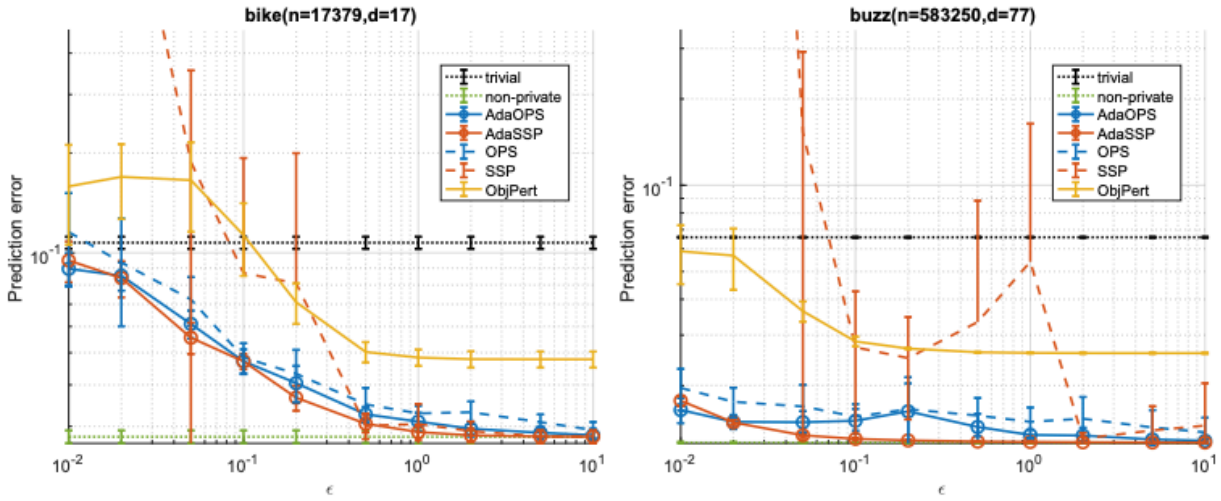
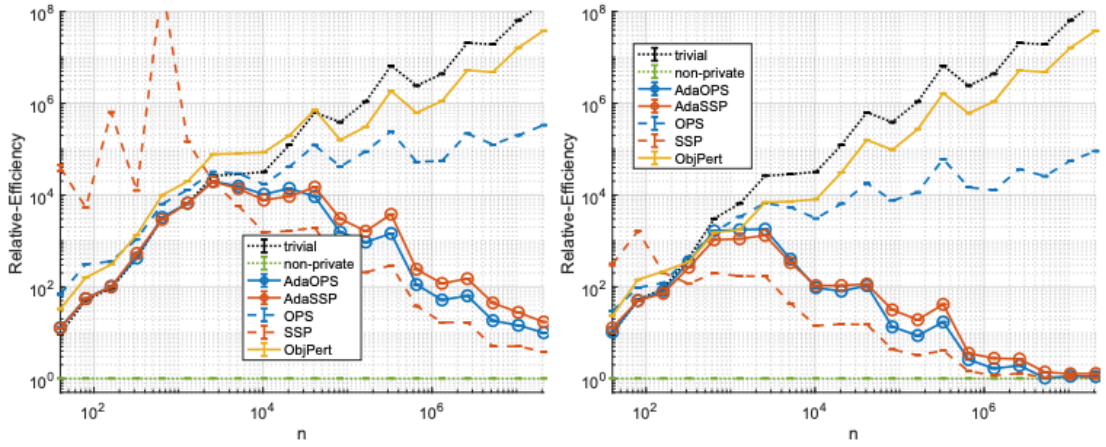


Figure 1: **Existing Figures, Figure 3 in [26]**. Wang operated using the same adaSSP and SSP definitions, as well as data from the UCI dataset ("bike" is the same that we use later). $\delta = 1/n^2$. He studies a larger selection of regression algorithms, such as AdaOPS and OPS. For this paper, we focus primarily on adaSSP and SSP. Standard deviation over trials is shown as vertical bars. Both the y-axis and x-axis are log scale.

Wang's Relative Efficiency of adaSSP and SSP in Synthetic Data



(c) Rel. efficiency at $\epsilon = 0.1$ (d) Rel. efficiency at $\epsilon = 1$

Figure 2: **Existing Figures, Figures 4c,d in [26]**. Wang operated using the same adaSSP and SSP definitions, as well as a similar construction of synthetic data. $d = 10$ for all synthetic data, and the dataset size n is on the x-axis. $\delta = 1/n^2$. He studies a larger selection of regression algorithms, such as AdaOPS and OPS. For this paper, we focus primarily on adaSSP and SSP. By the definition of relative efficiency, non-private linear regression is at 1. Both the x-axis and y-axis are log scale.

whether a new privacy-loss budget actually could demonstrate if adaSSP generally outperforms SSP. Yes, it may be the case that if every quantity in each algorithm had the same privacy-loss budget (ex. the $X^T X$ and $X^T \boldsymbol{y}$ releases), then having extra privacy-loss budget to release an eigenvalue may outperform SSP. However,

this paper is concerned with the situation where a maximum privacy-loss budget is set, and adaSSP or SSP must be chosen, meaning that choices must be made about the privacy-loss budget allocated to each of the released quantities within the algorithms. This budget concern motivates Question (I), and Question (II) aims to solve whether SSP or an adaptive variant (adaSSP or adaSSPbudget) is always preferable. constSSPfull is meant to be a stand-in for an algorithm that is always using a ridge regression approach with large hyperparameter, so comparisons to constSSPfull help tease out when a ridge regression approach, adaptive or otherwise, is preferable. Finally, Question (III) hopes to discuss the subtlety of whether the answer to the other questions depends on some features of the situation, like dataset size or overall privacy-loss budget.

Also, in Figure 2 we see that for small n values, adaSSP still does outperform SSP. We will see similar behavior in the later synthetic data experiments, which will help answer Question (III). In this graph, this behavior may be explained if small n leads to datasets that are not well behaved: if X^TX has small eigenvalues for low n, then a ridge regression approach used in adaSSP may be more valuable than the limited privacy-loss budget cost in the release of X^TX and $X^T\boldsymbol{y}$. As n goes to infinity, it appears that both SSP and adaSSP approach the non-private linear regression. Similarly, it appears that larger ϵ values mean that adaSSP and SSP each are closer to the non-private linear regression, which makes sense given that larger ϵ corresponds to less random noise. In the synthetic dataset, we would expect for adaSSP to generate a large eigenvalue and set a low ridge regression coefficient - this claim will be explored in later experiments (discussed with Figure 4). In this case, adaSSP and SSP would be computing a similar algorithm, just with different privacy budgets for the other quantities. Thus, as ϵ increases, both algorithms are dealing with smaller random noise, it just happens that SSP has slightly less noise per released quantity.

6.2 Synthetic Data

6.2.1 Synthetic Data: Relative Efficiency

To tackle these questions, Figure 3 explores how the privacy-loss budget within adaSSPbudget effects its performance on the synthetic data. Recall that γ corresponds to the proportion of the overall privacy-loss budget that is allocated to the eigenvalue release in adaSSPbudget. There is one plot per ϵ value, where different colored lines correspond to different sized datasets (i.e. the n value). Because neither SSP nor constSSPfull releases an eigenvalue, their budgets do not vary with γ so their performances do not change with γ. As a result, each of these two algorithms was computed once for every n, ϵ value, and their results were repeatedly displayed at each γ value. The red vertical line at $\gamma = 1/3$ marks where adaSSP would perform (adaSSP is a special case of adaSSPbudget). Based on the definition of relative efficiency, the baseline non-private linear regression is at $1 = 10^0$ because the numerator and denominator of relative efficiency would be the same.

Moving question by question, we can discuss the sensitivity of the performance of adaSSPbudet to the choice of γ in Figure 3, reminding us of Question (I). For small ϵ, it appears that the performance of adaSSPbudget does not vary significantly as γ changes (see the curves when $\epsilon = 0.001$). That said, as ϵ increases, a more pronounced shape in the adaSSPbudget curves emerge, where the performance is worse for γ near 1, and the optimal γ seems to be close to 0 (see the green curve in $\epsilon = 1$). As long as γ is a "reasonable" value ($\gamma \leq 0.5$), the performance appears to be within one order of magnitude away from the optimal value achieved by adaSSPbudget for each various n, ϵ combination. Thus, while $\gamma = 1/3$ is not always an optimal choice, in this case it is not that far away from the optimal relative efficiency.

However, once γ grows towards 1 (see again the green curve in $\epsilon = 1$, Figure 3), this worse performance could be explained by the fact that the benefit of increasing the accuracy of the releases of X^TX and $X^T\boldsymbol{y}$ exceeds some benefit of improving the accuracy of the eigenvalue release. Additionally, the additive constant used in step 5 of the adaSSPbudget algorithm will increase as γ goes to 1, so perhaps this behavior would cause for the output $\widehat{\boldsymbol{\theta}}$ to be closer and closer to 0 as γ approaches 1. For some (but not all) combinations of ϵ, n, it appears that as γ approaches 0, the relative efficiency increases (and thus performance worsens),

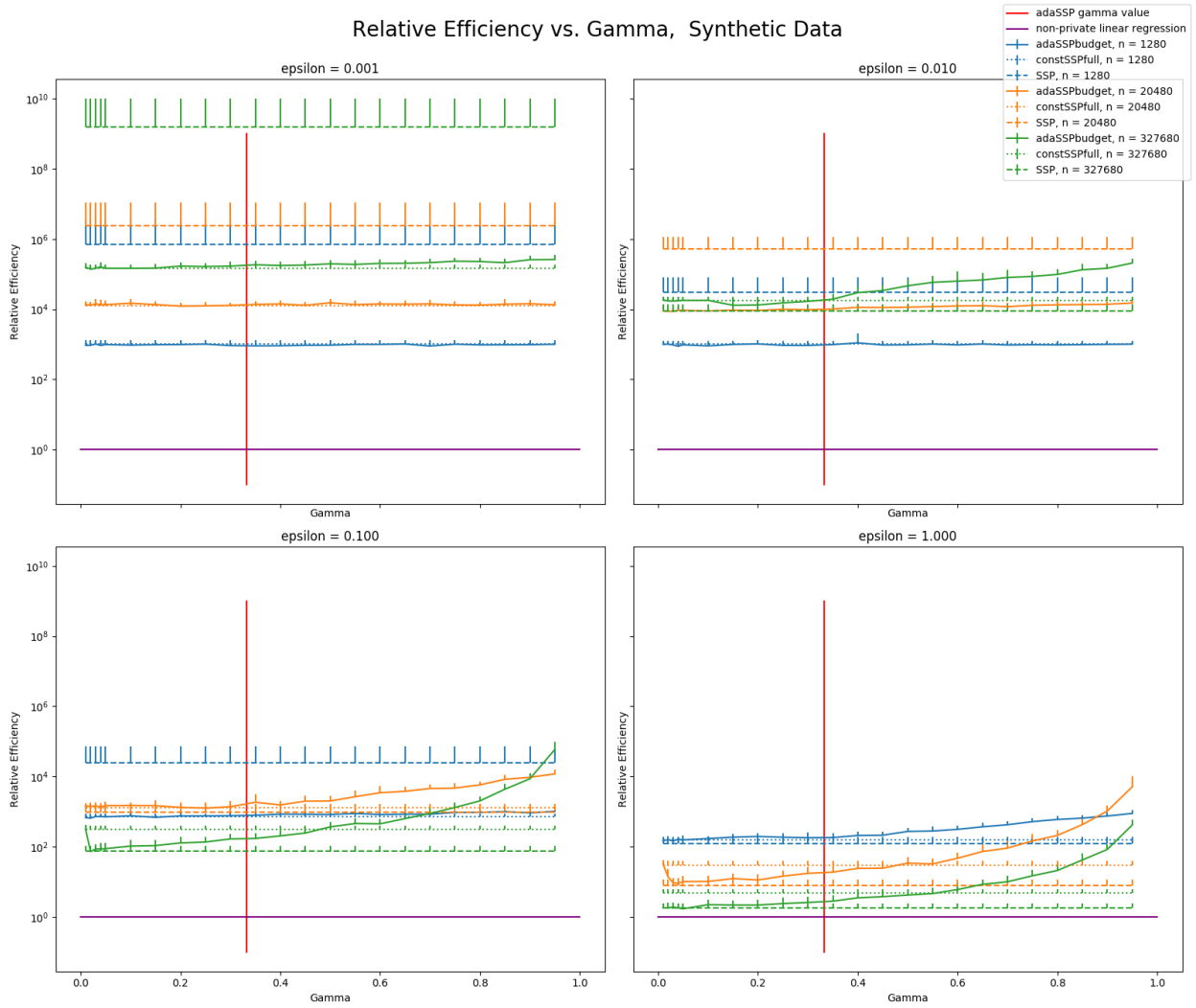


Figure 3: **Relative Efficiency for Synthetic Data**. $d = 10$ for all synthetic data. $\delta = 1/n^{1.1}$. 32 cross validation trials, split randomly into training size of 90%, test size of 10%. The lambda zero proportions from Figure 4 correspond to the ϵ, γ, n values from this experiment. Vertical error bars indicate standard deviation across the cross validation trials, where in the log scale, the numeric value of the error bar corresponds to the numeric difference on the y-axis between the top and bottom values of the error bar.

which may be explained by unreliable eigenvalue releases with a small privacy-loss budget allocated to its release (see the orange curve in $\epsilon = 1$).

Thinking of Question (II) (which also covers Question (III) when considering n, ϵ), we see that for small ϵ and small n values, adaSSPbudget consistently outperforms SSP. See this performance in all curves when $\epsilon = 0.001$, in the blue and orange curves when $\epsilon = 0.01$, and the blue curve when $\epsilon = 0.1$ (Figure 3). This result is coherent with Wang's results that we saw in Figure 2. Perhaps low n corresponds to unstable $X^T X$ values which benefit from a ridge regression approach (like adaSSPbudget) instead of an approach like SSP which uses the smallest ridge regression coefficient among the discussed algorithms, and is thus closest to

ordinary least squares. Also, for low ϵ, the SSP values seem to have high variance (see $\epsilon = 0.001$, or the orange and blue curve in $\epsilon = 0.01$), indicating that these small ϵ trials cause for large amounts of noise within the SSP and adaSSPbudget algorithms, so a ridge regression approach could help to dampen such noise (especially in the release of X^TX). The idea that adaSSPbudget performs more like ridge regression (with a large ridge regression coefficient λ) is evidenced by the fact that for low ϵ, n values, the constSSPfull and adaSSPbudget values are closely aligned. Thus, even though constSSPfull allocates more privacy-loss budget to the $X^TX, X^T\boldsymbol{y}$ releases than adaSSPbudget, its ridge regression coefficient is the main similarity with adaSSPbudget.

However, as n and ϵ grow, a new pattern emerges where the adaSSPbudget algorithm is consistently performing worse than SSP, where adaSSPbudget becomes within a margin of error of SSP at some optimal value of γ (which may depend on n, ϵ). See this behavior in the $\epsilon = 1$ curves, or the green and orange curves of $\epsilon = 0.1$ of Figure 3. This pattern confirms what we saw in Wang's results in Figure 2: that in many cases in the synthetic data, SSP actually outperforms adaSSPbudget. These results in Figure 3 go further, saying that the particular choice of $\gamma = 1/3$ is not the only budget with this behavior. Indeed, once n or ϵ are large enough, there appears to be no such γ value where adaSSPbudget outperforms SSP. This result indicates that adaSSP's poor performance against SSP in Figure 2 is not due to the restrictions of some arbitrary privacy-loss budgeting, but perhaps due to a feature of the data and the general approach of the algorithm that no budget could resolve.

As we consider general patterns in n, ϵ that have not been discussed to answer Question (III), we see that all algorithms (SSP, adaSSPbudget, constSSPfull) benefit from increases in ϵ, where it appears that each algorithm approaches the non-private linear regression benchmark as more privacy-loss budget is allotted (follow where the green curves are along the y-axis as ϵ increases, for example). It is also interesting to note that the ordering of the lines by dataset size n does not cleanly appear until larger ϵ values, again highlighting the increased random noise for smaller ϵ values.

For small ϵ or n, we see that in these events where adaSSPbudget outperforms SSP, constSSPfull similarly outperforms SSP (see $\epsilon = 0.001$ in Figure 3). This behavior may again be explained by a stability argument, where the larger ridge coefficient of constSSPfull helps to stabilize the low privacy-loss budget releases of the other quantities when ϵ is overall small. This argument is further supported by noticing that in these early ϵ values, the standard deviation of the SSP graph is much larger than for adaSSPbudget or constSSPfull, implying that SSP (with its smaller ridge regression coefficient) faces unstable releases and would benefit from a larger regularization term in computing its $\widehat{\boldsymbol{\theta}}$. Furthermore, we see that constSSPfull and adaSSPbudget in Figure 3 begin to move apart as ϵ and n increase, where adaSSPbudget can outperform constSSPfull (see in particular the green curve when $\epsilon = 1$). This behavior again suggests that adaSSPbudget eventually (for large enough n, ϵ) uses a small λ value in its ridge regression approach, whereas constSSPfull uses a larger constant coefficient.

To verify this internal behavior of adaSSPbudget, we turn to Figure 4.

6.2.2 Synthetic Data: Lambda Zero Proportion

To understand the behavior of adaSSPbudget in Figure 3, especially as it relates to constSSPfull, we can look at the proportion of times that λ (used in ridge regression of adaSSPbudget, where the ridge hyperparameter is this λ plus 1) was set to zero within adaSSPbudget's step 5, as we see in Figure 4. This exploration will help explain the internal behavior of the algorithm adaSSPbudget, so we can see how and when it uses the eigenvalue release to influence the ridge regression calculation. As stated earlier, the synthetic data was generated with the understanding that its X^TX value would be relatively well behaved, meaning that its smallest eigenvalue should be large. Thinking through the steps of adaSSPbudget, a large minimum eigenvalue release corresponds to a small λ value used within adaSSPbudget. Thus, a "Lambda Zero Proportion" of 1 corresponds to a setting where every trial of adaSSPbudget released a large minimum eigenvalue, and

a proportion of 0 corresponds to every trial of adaSSPbudget releasing a small minimum eigenvalue.

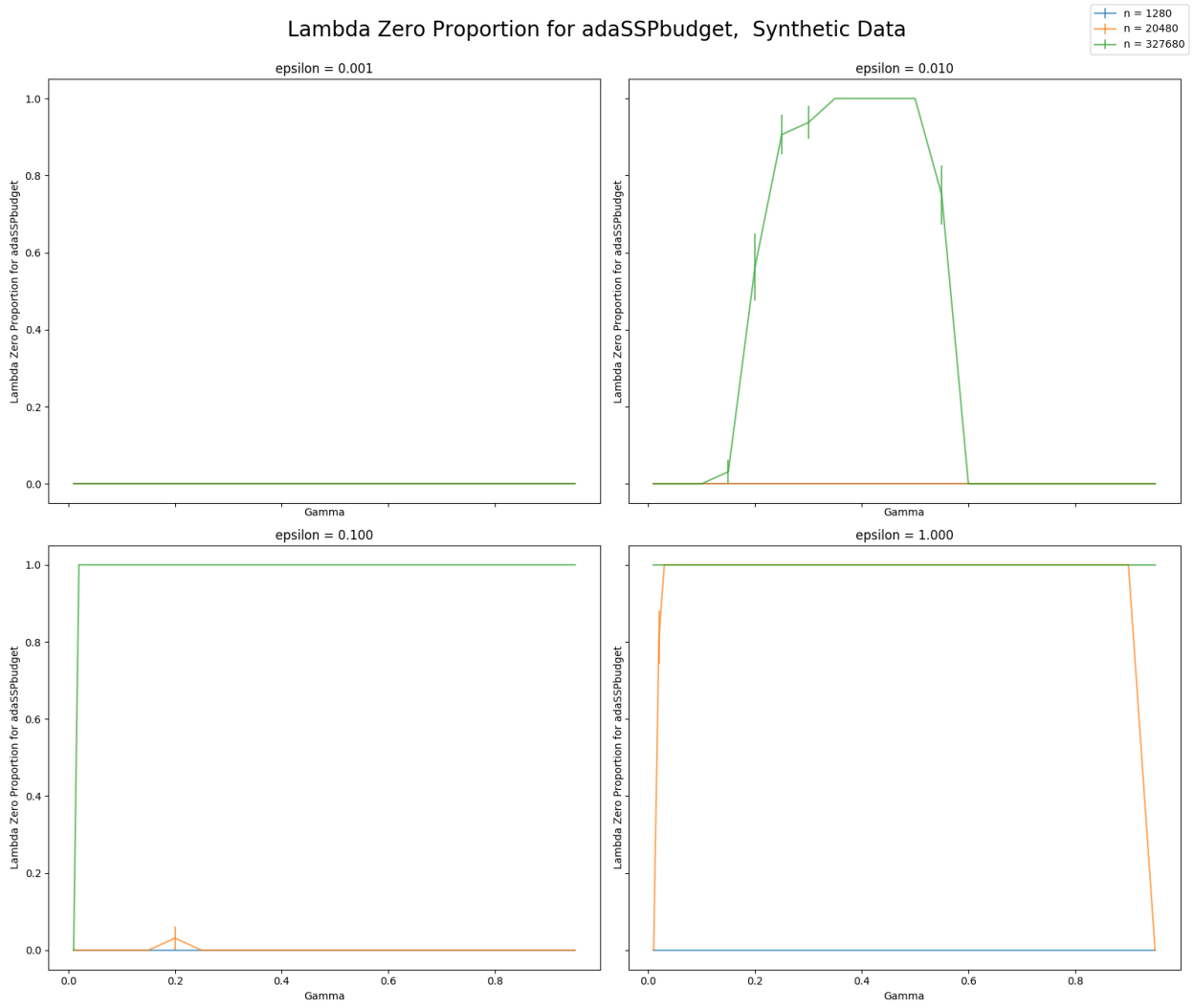


Figure 4: **Lambda Zero Proportions**. These graphs demonstrate how many times λ from step 5 of adaSSPbudget was set to 0 while running the algorithm on various γ, ϵ, n values on the synthetic data (so these γ, ϵ, n values correspond to those in Figure 3, in the same situation; $\delta = 1/n^{1.1}$). The y-axis is a proportion (between 0 and 1) of how many of the 32 cross-validation trials had the behavior where the adaSSPbudget algorithm set this λ to zero. Standard deviation for estimated proportion $\hat{p}$ is $\sqrt{\hat{p}(1-\hat{p})/32}$, [12].

As we see in Figure 4, as ϵ increases it becomes more frequent for the λ to be set at 0 (follow what happens with the green curve as ϵ increases). A similar phenomenon seems to happen as n increases (larger n values, like the green curves, achieve this 1 proportion faster than the orange curve). All of these behaviors make sense in the context of the synthetic data: it was generated with the hypothesis that ordinary least squares linear regression would handle the data better, so with higher ϵ, n, values, the privately released minimum eigenvalue will be more accurate to the correct large value of the data. The flat-line zero result in Figure 4

for the smallest $\epsilon = 0.001$ may indicate that there was, for no γ value, a sufficient amount of the privacy-loss budget ever devoted to the eigenvalue for the released eigenvalue to be reliably close to the correct large $\lambda_{min}(X^T X)$ value in this low privacy-loss budget setting. For higher ϵ, n values, when we trace through the adaSSPbudget algorithm, a large released minimum eigenvalue corresponds to a small λ parameter (possibly zero) used in the ridge regression. In a sense, one should hope that the algorithm has a high zero proportion on this synthetic data, given that a zero ridge regression hyperparameter will mimic simple linear regression. In the context of Question (I), this behavior implies that in this synthetic data, adaSSPbudget (with large enough ϵ, n) uses the same approach that SSP does (ridge regression with small hyperparameter), with the caveat that SSP did not waste any privacy-loss budget releasing an eigenvalue. This result would explain why, for large enough ϵ, n, adaSSPbudget seems to never outperform SSP in Figure 3.

Interestingly, thinking of Question (I), this behavior of adaSSPbudget setting $\lambda = 0$ seems to be mostly insensitive to the choice of γ; the only curve that suggests otherwise is the green plot from $\epsilon = 0.01$ in Figure 4, where small changes in γ may correspond to releasing mostly zero or nonzero λ values. In most of the plots, the behavior is either a proportion of 0 or 1 for all but the most extreme (near $\gamma = 0$ or $\gamma = 1$) values of γ. This behavior again suggests that the behavior of adaSSPbudget is not highly sensitive to the choice of γ, where the behavior of the adaSSPbudget is relatively consistent, provided that γ is in some reasonable range ($\gamma \leq 0.5$).

6.2.3 Synthetic Data: Prediction Error

In addition to testing the relative efficiency of adaSSPbudget, we also tested the prediction error of adaSSPbudget on the synthetic data in Figure 5. As we shall see, many of the same conclusions can be made for relative efficiency and prediction error with synthetic data. Furthermore, prediction error allows us to compare performance on synthetic data and UCI data, given that prediction error does not require that we have knowledge of the "true" $\boldsymbol{\theta_0}$. The set up to Figure 5 is the same as that of Figure 3, with similar graph conventions. Note that by the way that the data was standardized, a prediction error of $1 = 10^0$ should correspond to a prediction of the mean $\boldsymbol{y}$ value.

Considering Question (II), Figure 5 illustrates again a behavior where, for large enough ϵ, n values, adaSSPbudget can never outperform SSP (see $\epsilon = 1$, or the orange and green curves in $\epsilon = 0.1$). It is interesting to note that in this setting of prediction error, the adaSSPbudget curve no longer strictly adheres to the shape it had in Figure 3 where adaSSPbudget would come into contact with SSP only at some particular γ value. As we saw in Figure 3, adaSSPbudget's $\widehat{\boldsymbol{\theta}}$ was relatively close to the SSP value of $\widehat{\boldsymbol{\theta}}$ for a large range of γ values. Therefore, thinking of Question (I), the performance of adaSSPbudget may not be highly sensitive to γ, provided it is within some acceptable range, where the returned values of $\widehat{\boldsymbol{\theta}}$ from adaSSPbudget and SSP are close enough together that their predictions are close. We also see in Figure 5 that constSSPfull's returned $\widehat{\boldsymbol{\theta}}$ may also be within some close range of the SSP's returned $\widehat{\boldsymbol{\theta}}$, given that it too performs similarly to SSP in prediction error for large enough n, ϵ.

Concerning Question (III), we see that the prediction error improves for increased n and ϵ values across the different regression algorithms. This behavior again makes sense if the data is better behaved for larger n and the algorithms have less internal random noise for larger ϵ. In the next section, we consider the effect of a new dataset on the performance of adaSSPbudget.

6.3 UCI Data

Next, we consider the performance of these regression algorithms on UCI data, with prediction error in Figure 6. Note that in this case, each color corresponds to an entirely different dataset, which each has a unique n and d value. Recall that in Figure 1, Wang showed that for large enough ϵ values, adaSSP

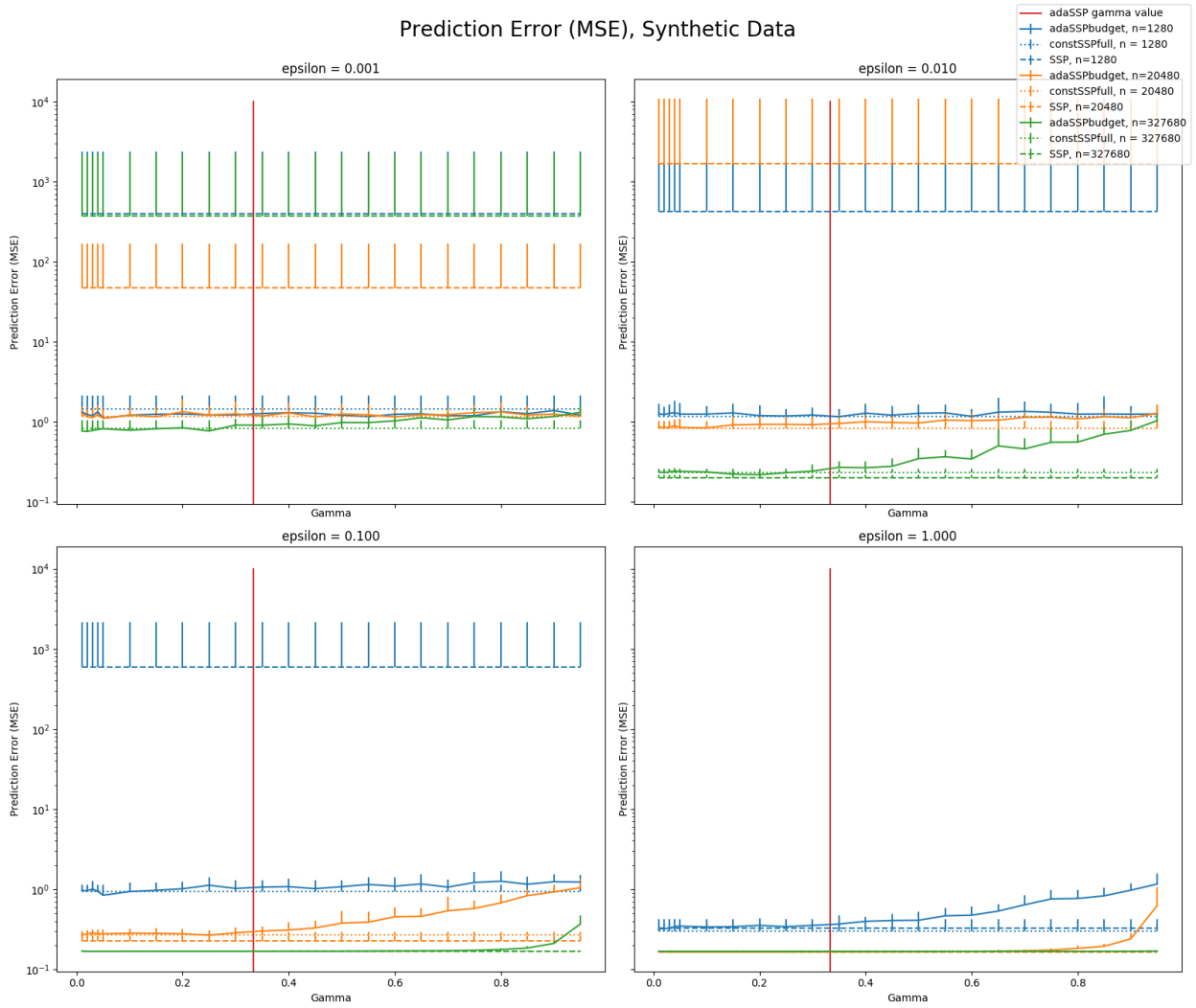


Figure 5: **Prediction Error for Synthetic Data**. $d = 10$ for all synthetic data. 32 cross validation trials, with a random split into a training batch size of 90%, test size of 10%. $\delta = 1/n^{1.1}$. By the way that the $\boldsymbol{y}$ values were standardized, a prediction error of 1 corresponds to a model that only predicts the mean of the $\boldsymbol{y}$ values. Vertical error bars indicate standard deviation across the cross validation trials. y-axis is log scale.

outperformed SSP in these UCI datasets (including the "bike" dataset seen here in Figure 6). The reasoning for this behavior was because in the UCI datasets, we expected for there to be correlations in the X matrix, such as between columns like "temperature" and "humidity" in the "bike" dataset.

What we see for the UCI data in Figure 6 is that the adaSSPbudget algorithm almost always outperforms the SSP version, even for large n, ϵ values. The fact that adaSSPbudget fares better here (under more conditions) than it did in the synthetic data makes sense, given that the UCI data would likely have more covariances in the data and would thus respond better to a ridge regression approach with large hyperparameter. Thus, thinking of Questions (II) and (III), adaSSPbudget outperforms SSP at least in this UCI

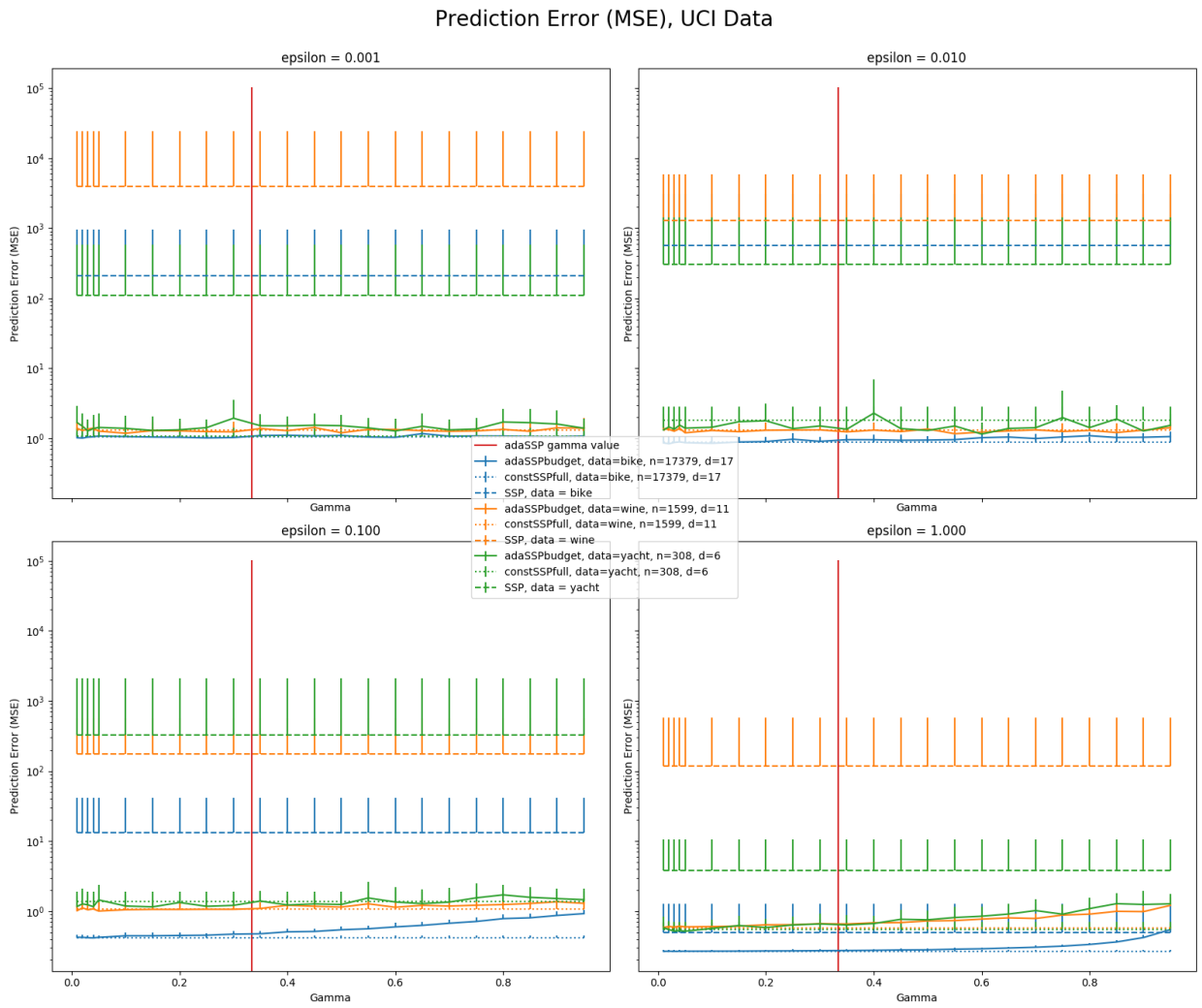


Figure 6: **UCI for different Gamma values**. One plot per ϵ value. Different colors correspond to different datasets, which have their own n and d values (d is no longer constant across different dataset sizes n). $\delta = 1/n^{1.1}$. The error bars correspond to standard deviation between the 32 cross validation trials. Note that this paper does not include the corresponding "lambda zero proportion" graphs for the UCI data because they were always 0 (for all γ, ϵ, data combinations); that would have been the corresponding UCI figure for Figure 4.

data setting.

The fact that the Lambda Zero Proportions for all UCI data results were always zero (as stated in Figure 6) drives home the idea that the UCI data would have generated very small minimum eigenvalues (within the adaSSPbudget algorithm), corresponding to high correlation. As a result, in this UCI data setting where we expect high correlation in the data, adaSSPbudget and constSSPfull should actually be close (closer than in the synthetic data setting), because both algorithms would reduce to a ridge regression approach with a similarly large λ value. This understanding of the data is indeed illustrated in Figure 6, where constSSPfull

seems to almost always be a lower bound for the prediction error of adaSSPbudget.

Also, the performance of adaSSPbudget on UCI data seems to not vary significantly, except for γ values close to 1 (see the curves in $\epsilon = 0.1$ and $\epsilon = 1$). This trend is similar for the synthetic data in Figure 5, where the adaSSPbudget performance was worse for these very large γ values. Considering Question (I), the fact that the performance worsens for large γ values makes sense, where the releases of $X^T X$ and $X^T \boldsymbol{y}$ have more internal randomness with large γ values. Also, as γ approaches 1, the coefficient in step 5 of adaSSPbudget approaches infinity, which would set $\widehat{\boldsymbol{\theta}}$ to zero, thus reducing the prediction error to a prediction of the mean.

It is also interesting that in the UCI data in Figure 6, the standard deviation of the SSP performance is consistently large, even for large n, ϵ values. This too may be an indicator that an approach closer to ordinary least squares has structural limitations in a setting where ridge regression with a large hyperparameter may perform better. Compare this high standard deviation to the seemingly stable values of SSP in Figure 5 with the synthetic data.

7 Discussion

To structure the discussion, we return to the motivating questions with answers from the data.

7.1 Question (I): Sensitivity of Performance to γ

With Question (I) concerning the sensitivity of performance to γ in adaSSPbudget, it appears that as long as γ is some "reasonable" value (perhaps $0.05 \leq \gamma \leq 0.5$), the performance of adaSSPbudget does not change significantly, as we see in Figures 3, 5, 6. Even in Figure 4, we see that the behavior of setting the λ value from step 5 of adaSSPbudget to 0 is consistent for most γ values in this range. As a result, the choice of $\gamma = 1/3$, when adaSSPbudget and adaSSP are the same, is a sensible choice of γ.

The fact that performance may worsen as γ approaches 1 makes sense, where as γ goes to 1, the internal randomness of the releases of $X^T X, X^T \boldsymbol{y}$ increases; thus, if these releases are unreliable, a well chosen hyperparameter may not be able to overcome the high variance of these other components. In Figure 3, we see that for large enough n, ϵ, increasing γ towards 1 drastically worsens the performance of adaSSPbudget. Another explanation for this behavior may be that the constant used in Step 5 of the adaSSPbudget algorithm will go to infinity as γ approaches 1. In this case, adaSSPbudget may be computing a $\widehat{\boldsymbol{\theta}}$ with smaller and smaller vector norm.

At first glance, it is perhaps surprising that, generally speaking, performance does not worsen significantly as γ approaches 0. One might think that an unstable eigenvalue release would effect the other aspects of the program. However, because adaSSPbudget contains two maximum operations to assert that the final λ is in a certain nonnegative range (from 0 to the constant in step 5 of adaSSPbudget), the instability coinciding with this eigenvalue release is dampened. Note that the releases of $X^T X$ and $X^T \boldsymbol{y}$ do not have any such flattening operations (applying a minimum or maximum), so instability in their releases can potentially destabilize the output $\widehat{\boldsymbol{\theta}}$ from adaSSPbudget.

7.2 Question (II): adaSSPbudget Compared to SSP and to constSSPfull

Now considering Question (II) and the comparison of adaSSPbudget against the other algorithms SSP and constSSPfull, there is a clear distinction between the different datasets. In the synthetic data, adaSSPbudget initially outperforms SSP for small ϵ, n values, as we see in Figures 3 and 5. However, once ϵ or n increase, a general pattern emerges in these plots that the performance of SSP in synthetic data is better

than adaSSPbudget for all γ values.

An explanation for this behavior is that with the synthetic data, where little covariance is expected in the X^TX matrix, an ordinary least squares approach may be preferred to a ridge regression approach. Indeed, as we saw in Figure 4, adaSSPbudget does set its λ value to 0 (for large enough ϵ, n), so its behavior should mimic the behavior of SSP. The primary difference between the two algorithms at this point is that SSP did not have to waste any of its privacy-loss budget on an eigenvalue release, so its releases of $X^TX, X^T\boldsymbol{y}$ were on balance more accurate than the same releases from adaSSPbudget. Notice also in the synthetic data in Figure 3 that, for large enough n, ϵ, the constSSPfull algorithm performs worse than adaSSPbudget for the previously mentioned "reasonable" choices of γ. This result again indicates that an approach closest to ordinary least squares fared better than a ridge regression approach with larger hyperparameter, because constSSPfull and SSP release $X^TX, X^T\boldsymbol{y}$ with the same privacy-loss parameters.

The behavior in the UCI data tells a different story. In Figure 6, we see that adaSSPbudget outperforms SSP for almost all γ values, even γ at extreme ends. We also observe that constSSPfull outperforms adaSSPbudget here, where for large enough n, ϵ, adaSSPbudget can achieve the value that constSSPfull does only at a narrow range of γ values. Both of these results with the UCI data suggest that in this setting where high covariance is expected, a ridge regression approach indeed outperforms an approach close to ordinary least squares.

There is an interesting parallel between the synthetic data and UCI data results, where for large enough n, ϵ values, adaSSPbudget can barely achieve the better performance of SSP or constSSPfull, respectively. This result demonstrates the potential power of the adaptive hyperparameter calculation inherent to adaSSPbudget and adaSSP. When the regression algorithm would benefit from a large ridge regression parameter, adaSSPbudget can adaptively set the λ accordingly. Similarly, when the regression algorithm would benefit from an approach close to ordinary least squares, the adaptive eigenvalue calculation allows for adaSSPbudget to set $\lambda = 0$.

However, the fact that the performance of adaSSPbudget compared to SSP differs in synthetic data and UCI data informs us that, contrary to Wang's suggestion about the sensitivity of the performance of adaSSP to its privacy-loss budgeting, there may be features inherent to the data that prevent adaSSPbudget from outperforming SSP. This result has potential implications to other differentially private algorithms that spend part of their privacy-loss budgets to improve a hyperparameter. Sure, having a well-tuned hyperparameter in a model may be preferable to the original model, "all else equal", but because a fixed privacy-loss budget must be divided between the releases of the algorithm, it is not the case that all else is equal if various releases have a now smaller privacy-loss budget. Faced with the choice to privately release a quantity to make a hyperparameter well tuned, it may be more worthwhile to improve the accuracy of the values used in the base model. In this problem, this has to do with ordinary least squares versus ridge regression, but one might find similar results in other machine learning techniques which take advantage of well tuned hyperparameters on top of an existing model.

7.3 Question (III): adaSSPbudget Under Different Conditions

When we considered Question (III) as we discussed the synthetic data in Figures 3, 4, 5, a similar trend appeared: for large enough n or ϵ values, the algorithms behaved in the way that one might expect given the dataset descriptions. The observation that as ϵ increases, the behavior of the algorithms becomes more coherent makes sense given that as ϵ increases, the magnitude of the internal randomness inside adaSSPbudget, SSP, and constSSPfull will decrease. The idea that similar behavior occurs as n increases may also correspond to the generated X^TX matrix for the synthetic data being better behaved when n increases.

It also appears to be a general trend, even with the UCI data in Figure 6, that as n increases, all 3 of adaSSPbudget, SSP, and constSSPfull perform better (compared to a smaller n). Note that this comparison in UCI comes with the caveat that these are 3 different datasets, and the number of features d happens to increase with n in these datasets. This ordering by size (again, once ϵ is large enough) may be another reflection of this idea that X^TX may more accurately represent the entire dataset for larger n. After all, these figures are computed using cross validation: if the training set is too small, its X^TX value may not actually match the corresponding X^TX from the test set.

When considering differences between how these algorithms perform on the synthetic data and UCI data, we also noted that the variance of SSP in Figure 6 is much higher in the UCI data than in the synthetic data in Figure 5. This observation again highlights the earlier idea that the UCI data may benefit from a ridge regression approach instead of an approach close to ordinary least squares.

8 Conclusion

In summary, this paper provided 3 contributions. First, we discussed the existing SSP and adaSSP algorithms and filled in the details of the proofs that they were each DP using $l2$ sensitivity and the composition of differential privacy. Second, we introduced the two new algorithms adaSSPbudget and constSSPfull and proved that these were each differentially private, again by appealing to $l2$ sensitivity and composition. Third, we conducted experiments using synthetic and UCI data to explore whether the precise privacy-loss budget used within adaSSP could help adaSSP outperform SSP.

Adaptive differentially private ridge regression techniques do indeed perform well for a robust choice of privacy-loss budgets in synthetic and in real UCI data, not just $\gamma = 1/3$ as adaSSP was originally set. This ridge regression technique performs better under conditions where high covariance in the data is likely, like in the UCI data. However, in settings where the original ordinary least squares approach worked better than a ridge regression approach, like in the synthetic data, no new budget could help the adaSSPbudget outperform SSP. This bound on performance demonstrates that the cost of the hyperparameter calculation may in fact not be universally worthwhile. Thus, implementers of differentially private algorithms should be mindful of their dataset before using a technique which spends part of its privacy-loss budget on a potentially wasteful hyperparameter calculation.

Given that this paper assesses only one adaptive ridge regression algorithm, adaSSP, future work would include the study of other differentially private ridge regression techniques, like Sheffet's Algorithm 2 [20]. The problem can be generalized to include other differentially private regression algorithms that consist of various private releases, where there is a distinction in the releases between "necessary" components and hyperparameters.

Within the context of the study of adaSSP, we still made the arbitrary choice to provide the same privacy-loss budget to the release of X^TX and $X^T\boldsymbol{y}$, putting both in the same bucket as "necessary". Perhaps one of these releases, especially for small ϵ, exhibits behavior where any marginal increase to its privacy-loss budget has more benefit than the accuracy cost to the other releases. Given that much attention in this paper was paid to the properties of X^TX, it may be the case that X^TX and the hyperparameter should have comparable privacy-loss budgets. Also, the various constants (in terms of $\epsilon, \delta, \gamma, \rho$) used within adaSSP, adaSSPbudget, and constSSPfull should be assessed.

Lastly, we tested various dataset sizes for the synthetic data, but we never changed the other dimension of the synthetic data (the d variable). Studying changes in d (and not just changes in n for $X \in \mathbb{R}^{n \times d}$) could be a worthwhile follow up to this work. Similarly, it may make sense to change some of the particulars of the set up to these calculations, such as the choice of $\gamma = 1/n^{1.1}$. Also, the synthetic data now is the

standard bearer for "clean" datasets, but future work could include synthetically generated data that had high correlations. This correlated synthetic data could be studied using relative efficiency, given that the true $\boldsymbol{\theta_0}$ would be known ahead of time.

References

[1] Daniel Alabi et al. "Differentially private Simple linear regression." In: *arXiv: 2007.05157 [cs.LG]* (2020). URL: https://arxiv.org/abs/2007.05157.

[2] Garrett Bernstein and Daniel Sheldon. "Differentially Private Bayesian Linear Regression." In: *CoRR* abs/1910.13153 (2019). arXiv: 1910.13153. URL: http://arxiv.org/abs/1910.13153.

[3] Jeremiah Blocki et al. "The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy." In: *CoRR* abs/1204.2136 (2012). arXiv: 1204.2136. URL: http://arxiv.org/abs/1204.2136.

[4] Efe Bozkir et al. *Differential Privacy for Eye Tracking with Temporal Correlations*. 2020. arXiv: 2002.08972 [cs.CR].

[5] Mark Bun. *A Teaser For Differential Privacy*. Princeton Lecture Slides. 2017. URL: https://www.cs.princeton.edu/~smattw/Teaching/521fa17lec22.pdf.

[6] Raj Chetty and John N Friedman. *A Practical Method to Reduce Privacy Loss when Disclosing Statistics Based on Small Samples*. Working Paper 25626. National Bureau of Economic Research, Mar. 2019. DOI: 10.3386/w25626. URL: http://www.nber.org/papers/w25626.

[7] Stephane Chretien and Sebastien Darses. *Perturbation bounds on the extremal singular values of a matrix after appending a column*. https://arxiv.org/abs/1406.5441. 2014. arXiv: 1406.5441 [math.SP].

[8] Marc Claesen and Bart De Moor. *Hyperparameter Search in Machine Learning*. 2015. arXiv: 1502.02127 [cs.LG].

[9] Cynthia Dwork and Aaron Roth. *The algorithmic foundations of differential privacy*. Foundations and Trends in Theoretical Computer Science. https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf. 2014.

[10] Cynthia Dwork et al. "Analyze Gauss: Optimal bounds for privacy-preserving principal component analysis." In: *Proceedings of the Annual ACM Symposium on Theory of Computing* (May 2014), pp. 11–20. DOI: 10.1145/2591796.2591883.

[11] Cynthia Dwork et al. "Calibrating Noise to Sensitivity in Private Data Analysis." In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32732-5.

[12] University of Florida Health. *Sampling Distribution of the Sample Proportion, p-hat*. https://bolt.mph.ufl.edu/6050-6052/module-9/sampling-distribution-of-p-hat/. 2021.

[13] James Foulds et al. *On the Theory and Practice of Privacy-Preserving Bayesian Data Analysis*. 2016. arXiv: 1603.07294 [cs.LG].

[14] Jonathan Ullman Gautam Kamath. *A Primer on Private Statistics*. arXiv preprint arXiv:2005.00010. https://arxiv.org/abs/2005.00010. 2020.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer. https://web.stanford.edu/~hastie/Papers/ESLII.pdf. 2009.

[16] A. E. Hoerl and R. W. Kennard. *Ridge regression: Biased estimation for nonorthogonal problems*. Technometrics, 12:55–67. 1970.

[17] Elica Klarreich. *Privacy By the Numbers: A New Approach to Safeguarding Data*. Quanta Magazine. 2012. URL: https://www.scientificamerican.com/article/privacy-by-the-numbers-a-new-approach-to-safeguarding-data/.

[18] Matt J. Kusner et al. *Differentially Private Bayesian Optimization*. 2015. arXiv: 1501.04080 [stat.ML].

[19] Or Sheffet. *Differentially Private Ordinary Least Squares*. 2017. arXiv: 1507.02482 [cs.DS].

[20] Or Sheffet. *Private Approximations of the 2nd-Moment Matrix Using Existing Techniques in Linear Regression*. 2015. arXiv: 1507.00056 [cs.DS].

[21] MIT: 18.S997 High-dimensional Statistics. *Sub-Gaussian Random Variables*. https://ocw.mit.edu/courses/mathematics/18-s997-high-dimensional-statistics-spring-2015/lecture-notes/MIT18_S997S15_Chapter1.pdf. 2015.

[22] A. N. Tikhonov. *Solution of incorrectly formulated problems and the regularization method.* Soviet Math. Dokl., 4. 1963.

[23] Machine Learning Repository University of California Irvine. *Bike Sharing Dataset Data Set*. https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset. 2013.

[24] Salil Vadhan. "The Complexity of Differential Privacy." In: *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Ed. by Yehuda Lindell. Cham: Springer International Publishing, 2017, pp. 347–450. ISBN: 978-3-319-57048-8. DOI: 10.1007/978-3-319-57048-8_7. URL: https://doi.org/10.1007/978-3-319-57048-8_7.

[25] D. Vu and A. Slavkovic. "Differential Privacy for Clinical Trial Data: Preliminary Evaluations." In: *2009 IEEE International Conference on Data Mining Workshops*. 2009, pp. 138–143. DOI: 10.1109/ICDMW.2009.52.

[26] Yu-Xiang Wang. *Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain*. Conference on Uncertainty in Artificial Intelligence (UAI). https://arxiv.org/abs/1803.02596. 2018.

[27] Kilian Weinberger et al. *Feature Hashing for Large Scale Multitask Learning*. 2010. arXiv: 0902.2206 [cs.AI].

[28] Xin Yan and Xiao Gang Su. *Linear Regression Analysis: Theory and Computation*. World Scientific. 2009. URL: https://books.google.com/books?id=MjNv6rGv8NIC&pg=PA1#v=onepage&q&f=false.

[29] Zichao Yang et al. "A la Carte - Learning Fast Kernels." In: *CoRR* abs/1412.6493 (2014). arXiv: 1412.6493. URL: http://arxiv.org/abs/1412.6493.

[30] Shipeng Yu et al. *Predicting readmission risk with institution specific prediction models*. In IEEE International Conference on Healthcare Informatics (ICHI), pp. 415–420. 2013.

[31] Jun Zhang et al. "Functional Mechanism: Regression Analysis under Differential Privacy." In: *CoRR* abs/1208.0219 (2012). arXiv: 1208.0219. URL: http://arxiv.org/abs/1208.0219.

[32] Keyu Zhu, Pascal Van Hentenryck, and Ferdinando Fioretto. *Bias and Variance of Post-processing in Differential Privacy*. 2020. arXiv: 2010.04327 [cs.LG].