CoinPress Extension: Linear Regression

Tatiana Ediger

April 2021

1 Introduction

With all the excitement around machine learning, there have been many concerns about privacy, and in particular, leaking information about the training data used for a particular model. We have seen attacks successfully reveal information about a training dataset from just the model. One of the most notable of these attacks being Membership Inference Attacks [SSSS17].

Linear Regression is a form of supervised machine learning used to generate a model that maps input data to a continuous set of outputs. The native method for solving linear regression finds parameters that minimize the distance between the predicted output and actual outputs. As it is, this native method has no privacy guarantees, i.e. information can be leveraged about the individuals used in the training dataset. This could be especially problematic if the model contains sensitive information about these individuals. The goal of this paper is to present a differentially-private algorithm to solve linear regression with a strong guarantee of protecting individuals in the training dataset.

Differential Privacy (DP) [DMNS06] is currently the most widely accepted solution to privacy in machine learning and statistics. CoinPress [BDKU20] is a private estimation confidence-interval based strategy that includes algorithms for finding the mean and covariance of a given dataset using DP. We use both of these estimation strategies to privately solve linear regression in the setting where the covariates and error terms are normally distributed.

2 Linear Regression

Linear regression is a form of supervised machine learning whose goal is to model the relationship between features and their labels, where the labels come from a continuous space.

Assume we have the distribution $(\vec{x}_i, y_i)$ with $\vec{x}_i \sim \mathcal{N}(\mu, \Sigma)$ a d-dimensional vector and $y_i|\vec{x}_i \sim \mathcal{N}(\langle \vec{x}, \beta \rangle, \sigma)$ a scalar. Let X, composed of $\vec{x}_{1,\cdots,n}$, be the training data, and y, composed of $y_{1,\cdots,n}$, be our training labels. Then to find the parameter vector, $\hat{\beta}$ which minimizes the square of the error between the predicted value βX and the actual output y, i.e. $\min_{\beta} \sum_{i=1}^{d} [\beta_i \cdot x_i - y_i]^2$, the

closed form solution is

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T y = \left(\frac{1}{n} X^T X\right)^{-1} \left(\frac{1}{n} X^T y\right)$$

Where $\frac{1}{n} X^T X$ converges to the inverse of the covariance matrix of X, and $\frac{1}{n} X^T y$ converges to the expected value $\mathbb{E}(x \cdot \langle x, \beta \rangle)$.

3 Our Algorithm

Theorem 3.1 (MVMRec [BDKU20])**.** *As proven in the cited paper, if we pass n samples $X_{1,\cdots,n}$ from $\mathcal{N} \sim (\mu, I_{d\times d})$, and $B_2(c, r)$ containing $\mu, \rho_s, \beta_s \geq 0$, to* `MVMRec`*, it returns a $(\Sigma_{i=1}^t \rho_i)$-zCDP estimate of μ.*

Theorem 3.2 (MVCRec [BDKU20])**.** *As proven in the cited paper, if we pass n samples $X_{1,\cdots,n}$ from $\mathcal{N} \sim (0, \Sigma)$, and L, u such that $L \preceq \Sigma \preceq uI$, $t \in \mathbb{N}^+, \rho_{1,\cdots,t}, \beta > 0$ to* `MVCRec`*, it returns a $(\Sigma_{i=1}^t \rho_i)$-zCDP estimate of Σ.*

From Theorem 2.1, we see that solving linear regression can be turned into a problem of estimating mean and covariance. Using MVMRec (Theorem 3.1) and MVCRec (Theorem 3.2), we can combine differentially private estimates for the mean and covariance to find a private estimate for $\hat{\beta}$.

Algorithm 1 LinReg Mean Estimation

Input: n samples of $z_{1,\cdots,n} \in \mathbb{R}^d$, $c \in \mathbb{R}^d, r, d$ and $\rho \geq 0$, $\|\beta\|_2^2$
Output: estimate of μ of z_i's

procedure LinRegMeanEst($z_{1,\cdots,n}, c, r, d, \|\beta\|_2^2, \rho$)
  Let $\vec{\rho} = [\frac{1}{4}\rho, \frac{3}{4}\rho]$
  Let $z_{1,\cdots,n} = \frac{1}{\sqrt{2*\|\beta\|_2^2+1}} \cdot z_{1,\cdots,n}$ ▷ Normalize z_i's by $\|\beta\|_2^2$
  return MVMRec($z_{1,\cdots,n}, c, r, 2, \vec{\rho}$) $\cdot\sqrt{2 \cdot \|\beta\|_2^2 + 1}$
end procedure

Algorithm 2 Beta L2-norm Estimation

Input: n samples from $y_{1,\cdots,n}$ from $\mathcal{N}(\langle \beta, \vec{x}_i \rangle, \sigma)$, and $d, t, \rho > 0$
Output: estimate of Σ of y_i's ▷ approximation for $\|\beta\|_2^2$

procedure BetaL2NormEst($y_{1,\cdots,n}, d, t, \rho$)
  Let $\vec{\rho} = [\frac{1}{4}\rho, \frac{3}{4}\rho]$
  Let $u = 100 \cdot d$
  return MVCRec($y_{1,\cdots,n}, n, d, u, \vec{\rho}, t$) - 1
end procedure

Algorithm 3 Linear Regression using CoinPress

Input: n samples $X_{1,\cdots,n}$ from $\mathcal{N}(0, \mathbb{I}_{d\times d})$, n samples from $y_{1,\cdots,n}$ from $\mathcal{N}(\langle\beta, \vec{x}_i\rangle, \sigma)$, $c \in \mathbb{R}^d$, $\rho > 0$, and $r \geq \|\beta - c\|_2$

Output: $\hat{\beta}$

procedure CoinPressLinReg($X_{1,\cdots,n}, y_{1,\cdots,n}, c, r, \rho$)
  Let ρDivision $= [0.1 \cdot \rho, 0.45 \cdot \rho, 0.45 \cdot \rho]$
  Let $z_i = X_i \cdot y_i$
  Let $\|\beta\|_2^2 =$ `BetaL2NormEst`$(y_{1,\cdots,n}, d, t, \rho_s[0])$
  Let meanEst = `LinRegMeanEst`$(z_{1,\cdots,n}, c, r, d, \|\beta\|_2^2, \rho_s[1])$
  return meanEst
end procedure

Our algorithm considers only the simplified case where $\vec{x} \sim N(0, I_{d\times d})$, and $y|\vec{x} \sim N(\langle\vec{x}, \beta\rangle, \sigma^2)$. Therefore, $\frac{1}{n}X^TX \approx I_{d\times d}$, and

$$\hat{\beta} = \mathbb{I}\frac{1}{n}X^Ty = \frac{1}{n}X^Ty$$

We can use Algorithm 1, `LinRegMeanEst`, to estimate $\left(\frac{1}{n}X^Ty\right)$, which converges to the expected value $\mathbb{E}(\vec{x} \cdot \langle\vec{x}, \beta\rangle)$. Therefore, if we define a variable $z_i = \vec{x}_i y_i \approx \vec{x} \cdot \langle\vec{x}, \beta\rangle$, we can pass all the z_i's to MVMRec. However, this first requires some modifications. First of all, we must transform the $(\vec{x}_i, y_i)$ data into z_i before passing it to MVMRec. Furthermore, MVMRec is stated and implemented as an algorithm for a Gaussian with identity variance, but the same argument works for an arbitrary known covariance Σ if we rescale the data. The expected covariance of the z_i's is in fact not the identity so a rescale is necessary. In particular, from Lemma 3.1, we can see that a good approximate to Σ is $\sqrt{2\|\beta\|_2^2 + 1} \cdot \mathbb{I}_{d\times d}$.

Lemma 3.1. $\sqrt{2\|\beta\|_2^2+1}\cdot\mathbb{I}_{d\times d}$ *dominates the covariance matrix,* Σ*, of* $z_{1,\cdots,n}$*.*

Proof. The off-diagonal entries $Cov(z_j, z_k)$ take on the value $\beta_j\beta_k$ (Lemma 3.4), and the diagonal entries of $Cov(z_j, z_j)$ take on the value $\beta_j^2+\|\beta\|_2^2+1$ (Lemma 3.3). Combining these two results, we get that $\Sigma \in \mathbb{R}^{d\times d}$, with $\Sigma = \beta\beta^T + (\|\beta\|_2^2+1)\mathbb{I}_{d\times d}$. Furthermore, if we refer to Figure 1 below, we can see a pictorial representation of this proof. The inner red circle represents $\|\beta\|_2^2$, and the blue ellipse represents adding the term $\beta\beta^T$. Then we can see that the red circle which is approximately equal to $\sqrt{2\|\beta\|_2^2+1}\cdot\mathbb{I}_{d\times d}$, gives an overestimate for Σ. In using CoinPress, we would rather have an estimate of $\|\beta\|_2^2$ that is too large rather than too small. ■

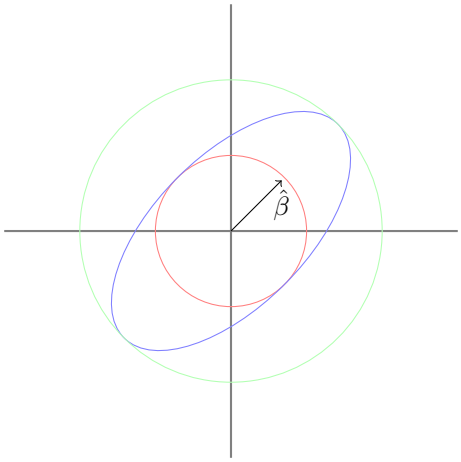

Figure 1: Using $\sqrt{2\|\beta\|_2^2+1}\cdot\mathbb{I}_{d\times d}$ to approximate Σ of $z_{1,\cdots,n}$.

Lemma 3.2. *The diagonal entries of $Cov(z_j, z_j)$ take on the value $\beta_j^2 + \|\beta\|_2^2 + 1$.*

Proof. For every d-dimensional vector, $\vec{z} = \vec{x} \cdot y$, if we assume $y = \langle x, \beta \rangle + v$, where v is random noise with zero mean:

$$\begin{aligned}
\mathbb{E}\left(z_j^2\right) - \mathbb{E}\left(z_j\right)^2 &= \mathbb{E}\left(z_j^2\right) - \beta_j^2 && (1)\\
&= \mathbb{E}\left((x_j y)^2\right) - \beta_j^2 && (2)\\
&= \mathbb{E}\left(\left(x_j \cdot \left(\langle \vec{x}, \vec{\beta} \rangle + v\right)\right)^2\right) - \beta_j^2 && (3)\\
&= \mathbb{E}\left(x_j^2 \cdot \left(\langle \vec{x}, \vec{\beta} \rangle + v^2 + 2v\langle \vec{x}, \vec{\beta} \rangle\right)\right) - \beta_j^2 && (4)\\
&= \mathbb{E}\left(x_j^2 \langle \vec{x}, \vec{\beta} \rangle + x_j^2 v^2 + 2x_j^2 v \langle \vec{x}, \vec{\beta} \rangle\right) - \beta_j^2 && (5)\\
&= \mathbb{E}\left(x_j^2 \langle \vec{x}, \vec{\beta} \rangle\right) + 1 - \beta_j^2 && (6)\\
&= \mathbb{E}\left(x_j^2 \left(\sum_{k=1}^{d} x_k \beta_k\right)^2\right) + 1 - \beta_j^2 && (7)\\
&= \mathbb{E}\left(x_j^2 \sum_{k=1}^{d} \sum_{l=1}^{d} x_k x_l \beta_k \beta_l\right) + 1 - \beta_j^2 && (8)\\
&= \mathbb{E}\left(x_j^4 \beta_j^2\right) + \sum_{k \neq j} \mathbb{E}\left(x_k^2 \beta_k^2\right) + 1 - \beta_j^2 && (9)\\
&= \beta_j^2 \mathbb{E}\left(\mathcal{N}(0,1)\right) + \sum_{k \neq j} \mathbb{E}\left(x_k^2 \beta_k^2\right) + 1 - \beta_j^2 && (10)\\
&= 3\beta_j^2 + \sum_{k \neq j} \beta_k^2 + 1 - \beta_j^2 && (11)\\
&= 2\beta_j^2 + \sum_{k \neq j} \beta_k^2 + 1 && (12)\\
&= \beta_j^2 + \sum_{k=1}^{d} \beta_k^2 + 1 && (13)\\
&= \beta_j^2 + \|\beta\|_2^2 + 1 && (14)
\end{aligned}$$

■

Lemma 3.3. *The off-diagonal entries $Cov(z_j, z_k)$ take on the value $\beta_j\beta_k$.*

Proof. For every d-dimensional vector, $\vec{z} = \vec{x} \cdot y$, if we assume $y = \langle x, \beta \rangle + v$, where v is random noise with zero mean, then for entry C_{jk}, we want $\text{Cov}(z_j, z_k)$:

$$\begin{aligned}
\text{Cov}(z_j, z_k) &= \mathbb{E}\left((z_j - \mathbb{E}(z_j))(z_k - \mathbb{E}(z_k))\right) && (15)\\
&= \mathbb{E}\left(z_j z_k - z_j\mathbb{E}(z_k) - z_k\mathbb{E}(z_j) + \mathbb{E}(z_j)\mathbb{E}(z_k)\right) && (16)\\
&= \mathbb{E}\left(z_j z_k\right) - \mathbb{E}\left(z_j\mathbb{E}\left(z_k\right)\right) - \mathbb{E}\left(z_k\mathbb{E}\left(z_j\right)\right) + \mathbb{E}\left(\mathbb{E}\left(z_j\right)\mathbb{E}\left(z_k\right)\right) && (17)\\
&= \mathbb{E}\left(z_j z_k\right) - \mathbb{E}\left(z_j\mathbb{E}\left(z_k\right)\right) && (18)\\
&= \mathbb{E}\left(z_j z_k\right) - \beta_j\beta_k && (19)\\
&= \mathbb{E}\left(\left(x_j y\right)\left(x_k y\right)\right) - \beta_j\beta_k && (20)\\
&= \mathbb{E}\left(\left(x_j \cdot \left(\langle \vec{x}, \vec{\beta}\rangle + v\right)\right)\left(x_k \cdot \left(\langle \vec{x}, \vec{\beta}\rangle + v\right)\right)\right) - \beta_j\beta_k && (21)\\
&= \mathbb{E}\left(\left(x_j\langle \vec{x}, \vec{\beta}\rangle + x_j v\right)\left(x_k\langle \vec{x}, \vec{\beta}\rangle + x_k v\right)\right) - \beta_j\beta_k && (22)\\
&= \mathbb{E}\left(x_j x_k\langle \vec{x}, \vec{\beta}\rangle^2 + 2x_j x_k\rangle\vec{x}, \vec{\beta}\rangle v + x_j x_k v^2\right) - \beta_j\beta_k && (23)\\
&= \mathbb{E}\left(x_j x_k\langle \vec{x}, \vec{\beta}^2\right) + \mathbb{E}\left(2x_j x_k\langle \vec{x}, \beta\rangle v\right) - \mathbb{E}\left(x_j x_k v^2\right) - \beta_j\beta_k && (24)\\
&= \mathbb{E}\left(x_j x_k\langle \vec{x}, \vec{\beta}^2\right) - \beta_j\beta_k && (25)\\
&= \mathbb{E}\left(x_j x_k\left(\sum_{i=1}^{d} x_i\beta_i\right)^2\right) - \beta_j\beta_k && (26)\\
&= \mathbb{E}\left(x_j x_k\sum_{h=1}^{d}\sum_{i=1}^{d} x_h x_i\beta_h\beta_i\right) - \beta_j\beta_k && (27)\\
&= 2\mathbb{E}\left(x_j^2 x_k^2\beta_j\beta_k\right) - \beta_j\beta_k && (28)\\
&= 2\beta_j\beta_k - \beta_j\beta_k && (29)\\
&= \beta_j\beta_k && (30)
\end{aligned}$$

■

Lemma 3.4. `BetaL2NormEst` *gives a ρ-zCDP approximation of $\|\beta\|_2^2$*

Proof. We know that $y_i \sim \mathcal{N}\left(0, \|\beta\|_2^2 + 1\right)$. So calling `MVCRec` with $y_{1,\cdots,n}$, which is 1-dimensional data yields a scalar representing the covariance, $\tilde{v}$. Then this estimate should approximately equal $\|\beta\|_2^2 + 1$, and therefore $\|\beta\|_2^2 \approx \tilde{v} - 1$.

With respect to privacy, we pass $y_{1,\cdots,n}$ into CoinPress's `MVCRec`, which returns a $(\Sigma_{i=1}^{t}\rho_i)$-zCDP estimate of Σ. Therefore `BetaL2NormEst` gives a ρ-zCDP approximation of $\|\beta\|_2^2$ ■

Once we have pass in the z_i's, and pass in the center as a $d \times 1$ vector of all zeros, and $r = 100\sqrt{d}$, then we get an estimate for $\frac{1}{n}X^T y = \hat{\beta}$.

Theorem 3.3 (H). *Our algorithm,* `CoinPressLinReg`*, is* ρ*-zCDP.*

Proof. We know that ρ-zCDP can be composed with smooth degradation over all of it's privacy parameters [BS16]. Since `CoinPressLinReg` composes two ρ_i-zCDP algorithms, and $\sum_{i=1}^{2} \rho_i = \rho$, then `CoinPressLinReg`, is ρ-zCDP. ■

4 Experimental Results

To test the results of our algorithm, we computed the population loss of our estimated $\hat{\beta}$. The population loss is found by

$$\mathbb{E}\left(\left(\langle \vec{x}, \vec{\beta} \rangle - y\right)^2\right)$$

For all of the following experiments, we generate β, our underlying distribution, as a d-dimensional vector where each entry is sampled from $\mathcal{N}(0, 1)$. We sample the entries in $X_{n \times d}$ from $\mathcal{N}(0, 1)$, and generate the n-dimensional vector y by taking the dot product $\langle X, \beta \rangle$ and adding random noise proportional to $\mathcal{N}(0, 1)$. Each experiment is run 50 times, for each value of n or ρ. In each iteration we generate new data, find the error between our predicted $\hat{\beta}$ and β on newly generated X and y data with $n = 1000$, and then at the end we take the average error over every trial.

We first compared the accuracy of our algorithm at different dimensions. From the graphs below, we can see that in lower dimensions, our algorithm is accurate even with smaller values of n.

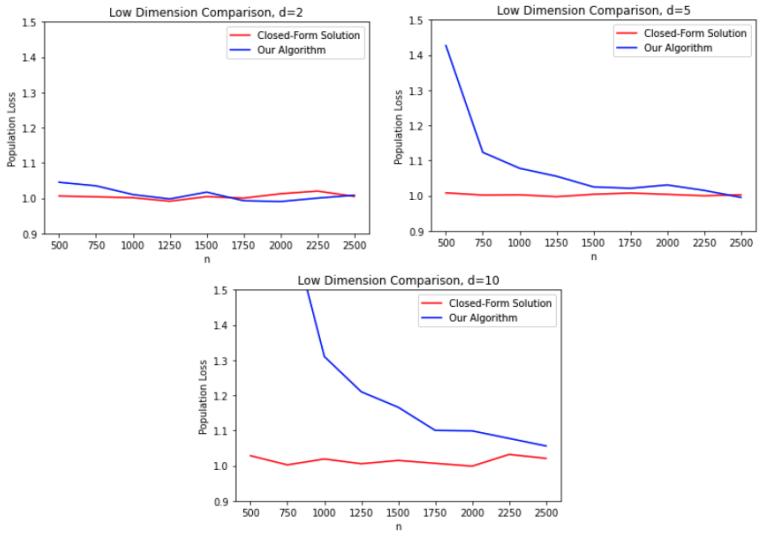

Figure 2: Low Dimensional Comparison ($\rho = 0.5$)

We also compared the accuracy of our algorithm at higher dimensions, and saw that for larger values of $n \geq 4000$, give good estimates when $d \leq 35$.

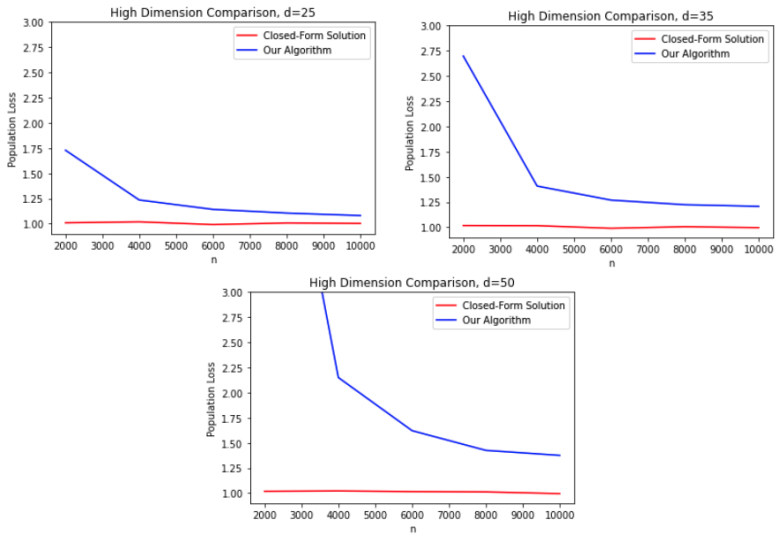

Figure 3: High Dimensional Comparison ($\rho = 0.5, d = \{25, 35, 50\}$)

We also compared the accuracy of our algorithm while varying ρ, the total privacy budget. We saw that with fewer dimensions, our algorithm is still accurate for low values of ρ.

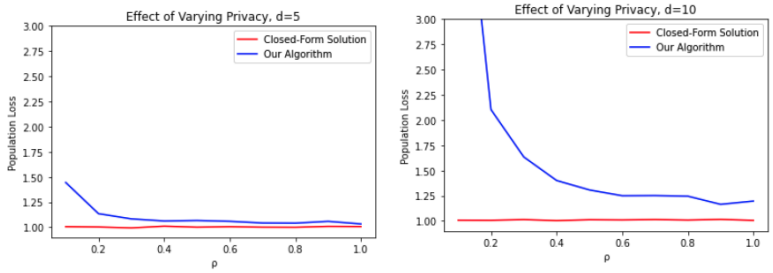

Figure 4: Measuring Effects of Privacy ($d = \{5, 10\}, n = 1000$)

We also wanted to analyze different error functions when measuring the accuracy of our algorithm.

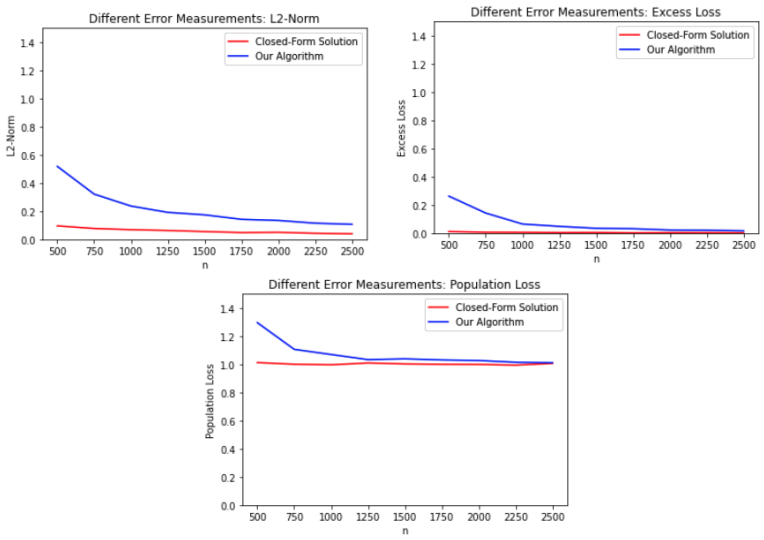

Figure 5: Different Error Functions ($d = 5, \rho = 0.5$)

5 Extension: $X \sim \mathcal{N}(0, \Sigma)$

Throughout the previous 4 sections, we worked under the assumption that $X \sim \mathcal{N}(0, \mathbb{I}_{d\times d})$. We will now relax those assumptions, and extend our algorithm to work in the case where $X \sim \mathcal{N}(0, \Sigma)$. In order to do this we will reduce this problem into our original algorithm. To do this, we must transform our data to appear normally distributed with $\mathcal{N}(0, \mathbb{I}_{d\times d})$. If $y|x \sim \mathcal{N}(\langle \vec{x}, \beta \rangle, 1)$, then $y|\Sigma^{-1/2}x \sim \mathcal{N}(\langle \vec{x}, \beta \rangle, 1) = \mathcal{N}(\langle \Sigma^{-1/2}\vec{x}, \Sigma^{1/2}\beta \rangle, 1)$. So we can run our original algorithm using $\Sigma^{-1/2}X$, to get a resultant $\Sigma^{1/2}\hat{\beta}$. Then, we can rescale by $\Sigma^{-1/2}$ to get $\hat{\beta}$.

5.1 Algorithm

Algorithm 4 LinReg Covariance Estimation

Input: $x_{1,\ldots,n}$, d, t, ρ
Input: n samples $X_{1,\cdots,n}$ from $\mathcal{N}(0, \Sigma)$, and $d, t, \rho > 0$
Output: estimate of Σ of X_i's

procedure LinRegCovarianceEst($x_{1,\ldots,n}, d, t, \rho$)
  Let $\rho_s = [\frac{1}{4}\rho, \frac{3}{4}\rho]$
  Let $u = 10 \cdot \sqrt{d}$
  return MVCRec($x_{1,\cdots,n}, n, d, u, \rho_s, t$)
end procedure

Algorithm 5 Linear Regression using CoinPress Extension (Non-identity covariance)

Input: n samples $X_{1,\cdots,n}$ from $\mathcal{N}(0, \Sigma)$, n samples from $y_{1,\cdots,n}$ from $\mathcal{N}(\langle \beta, \vec{x}_i \rangle, \sigma)$, $c \in \mathbb{R}^d$, $\rho > 0$, and $r \geq \|\beta - c\|_2$
Output: $\hat{\beta}$

procedure CoinPressLinRegExtCov($X_{1,\cdots,n}, y_{1,\cdots,n}, c, r, \rho$)
  Let $\hat{\Sigma} =$ `LinRegCovarianceEst`$(x_{1,\ldots,n}, d, t, \frac{\rho}{3})$
  Let $\tilde{X} = X \cdot \hat{\Sigma}^{-1/2}$
  Let $\tilde{\beta} =$ `CoinPressLinReg(`$\tilde{X}, y_{1,\cdots,n}$`,` $c, r, \frac{2\rho}{3}$`)`
  Let meanEst $= \hat{\Sigma}^{-1/2} \cdot \tilde{\beta}$
  return meanEst
end procedure

5.2 Experimental Results

To test the results of our extended algorithm, we once again computed the population loss of our estimated $\hat{\beta}$. The population loss is found by

$$\mathbb{E}\left(\left(\langle \vec{x}, \vec{\beta} \rangle - y\right)^2\right)$$

...

For all of the following experiments, we generate β, our underlying distribution, as a d-dimensional vector where each entry is sampled from $\mathcal{N}(0, 1)$. We sample the entries in $X_{n \times d}$ from $\mathcal{N}(0, \Sigma)$, and generate the n-dimensional vector y by taking the dot product $\langle X, \beta \rangle$ and adding random noise proportional to $\mathcal{N}(0, 1)$. Each experiment is run 50 times, for each value of n or ρ. In each iteration we generate new data, find the error between our predicted $\hat{\beta}$ and β on newly generated X and y data with $n = 1000$, and then at the end we take the average error over every trial.

We first compared the accuracy of our algorithm at different dimensions using different covariances. First, we look at the case where two variables are very highly correlated: the top right corner of the covariance matrix of all dimensions is

$$\begin{bmatrix} 6 & 2 \\ 2 & 1 \end{bmatrix}$$

and the rest is the identity.

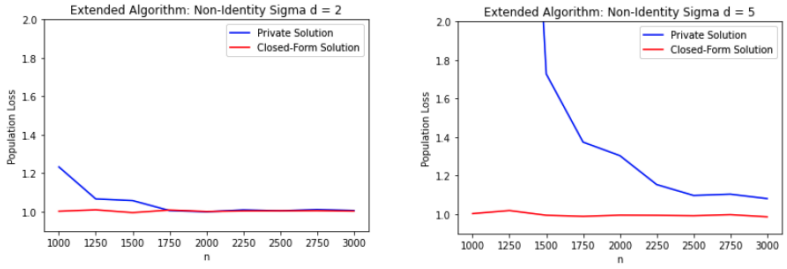

Figure 6: Low Dimensional Comparison - 2 Variables Highly Correlated ($\rho = 0.5$)

Next, we look at the case where the covariance is a high multiple of the identity matrix: $10 \cdot \mathbb{I}_{d \times d}$.

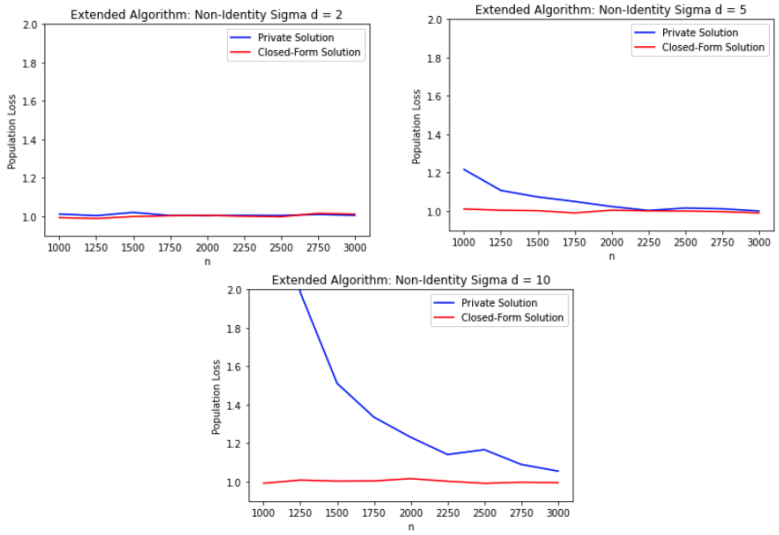

Figure 7: Low Dimensional Comparison - Multiple of Identity ($\rho = 0.5$)

We also compared the accuracy of our algorithm while varying ρ, the total privacy budget. We saw that with fewer dimensions, our algorithm is still

accurate for low values of ρ. This uses the same covariance as figure 6.

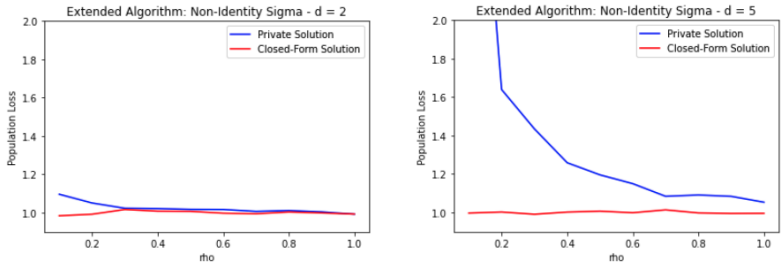

Figure 8: Measuring Effects of Privacy ($d = \{2, 5\}, n = 1000$)

6 Extension: $X \sim \mathcal{N}(\mu, \mathbb{I}_{d\times d})$

Throughout the first 4 sections, we worked under the assumption that $X \sim \mathcal{N}(0, \mathbb{I}_{d\times d})$. We will now relax those assumptions, and extend our algorithm to work in the case where $X \sim \mathcal{N}(\mu, \mathbb{I}_{d\times d})$. In order to do this we will reduce this problem into our original algorithm. To do this, we must transform our data to appear normally distributed with $\mathcal{N}(0, \mathbb{I}_{d\times d})$. If we start with the original algorithm, and then add an additional column of 1's to X, we are working with the following system of linear equations:

$$\left[\begin{array}{c|c} X & \vec{1} \end{array}\right] \left[\begin{array}{c} \vec{\beta} \\ \hline 0 \end{array}\right] = \left[y\right] + \left[z\right]$$

Where this updated X has mean $\big[\mu|1\big]$, and covariance $\left[\begin{array}{c|c} \Sigma & \vec{0} \\ \hline \vec{0} & 1 \end{array}\right]$.

In the case where we have a nonzero mean, this can be rewritten as

$$\left[\begin{array}{c|c} (x_1 - \bar{\mu}) & \vec{1} \\ (x_2 - \bar{\mu} & \\ \ldots & \end{array}\right] \left[\begin{array}{c} \vec{\beta} \\ \hline \langle \bar{\mu}, \vec{\beta} \rangle \end{array}\right] = \left[y\right] + \left[z\right]$$

Where the updated X has mean $\big[\mu - \mu = 0|1\big]$, and covariance $\left[\begin{array}{c|c} \Sigma^{-1/2}\Sigma^{1]2} = \mathbb{I}_{d\times d} & \vec{0} \\ \hline \vec{0} & 1 \end{array}\right]$.

We see that we are now dealing with normal data which has zero mean and identity covariance, so we can run our original algorithm. When we run our original algorithm on

$$X' = \left[\begin{array}{c|c} (x_1 - \bar{\mu}) & \vec{1} \\ (x_2 - \bar{\mu} & \\ \ldots & \end{array}\right]$$

we get an estimate for $\tilde{\beta} = \begin{bmatrix} \vec{\beta} \\ \langle \vec{\mu}, \vec{\beta} \rangle \end{bmatrix}$. And finally, to get $\hat{\beta}$, we can take all but the last element of $\tilde{\beta}$, $\hat{\beta} = \tilde{\beta}[:-1]$.

6.1 Algorithm

Algorithm 6 Linear Regression using CoinPress Extension (Nonzero mean)

Input: n samples $X_{1,\cdots,n}$ from $\mathcal{N}(\mu, \mathbb{I}_{d\times d})$, n samples from $y_{1,\cdots,n}$ from $\mathcal{N}(\langle \beta, \vec{x}_i \rangle, \sigma)$, $c \in \mathbb{R}^d$, $\rho > 0$, and $r \geq \|\beta - c\|_2$

Output: $\hat{\beta}$

procedure CoinPressLinRegExtMean($X_{1,\cdots,n}, y_{1,\cdots,n}, c, r, \rho$)
  Let $\vec{\rho}_1 = [\frac{1}{4} \cdot \frac{\rho}{3}, \frac{3}{4} \cdot \frac{\rho}{3}]$
  Let $\hat{\mu} = \texttt{MVMRec}(x_{1,\cdots,n}, c, r, 2, \vec{\rho_1})$
  Let $X' = X - [\mu]^n$
  Let $\tilde{X} = \begin{bmatrix} & X' & | & \vec{1} & \end{bmatrix}$
  Let $\tilde{\beta} = \texttt{CoinPressLinReg}(\tilde{X}, y_{1,\cdots,n}, c, r, \frac{2\rho}{3})$
  Let meanEst $= \tilde{\beta}[:-1]$
  return meanEst
end procedure

6.2 Experimental Results

To test the results of our extended algorithm, we once again computed the population loss of our estimated $\hat{\beta}$. The population loss is found by

$$\mathbb{E}\left(\left(\langle \vec{x}, \vec{\beta} \rangle - y\right)^2\right)$$

For all of the following experiments, we generate β, our underlying distribution, as a d-dimensional vector where each entry is sampled from $\mathcal{N}(0, 1)$. We sample the entries in $X_{n\times d}$ from $\mathcal{N}(\mu, 1)$, and generate the n-dimensional vector y by taking the dot product $\langle X, \beta \rangle$ and adding random noise proportional to $\mathcal{N}(0, 1)$. Each experiment is run 50 times, for each value of n or ρ. In each iteration we generate new data, find the error between our predicted $\hat{\beta}$ and β on newly generated X and y data with $n = 1000$, and then at the end we take the average error over every trial.

We first compared the accuracy of our algorithm at different dimensions using different values for μ. First, we look at the case where $\mu = 2$.

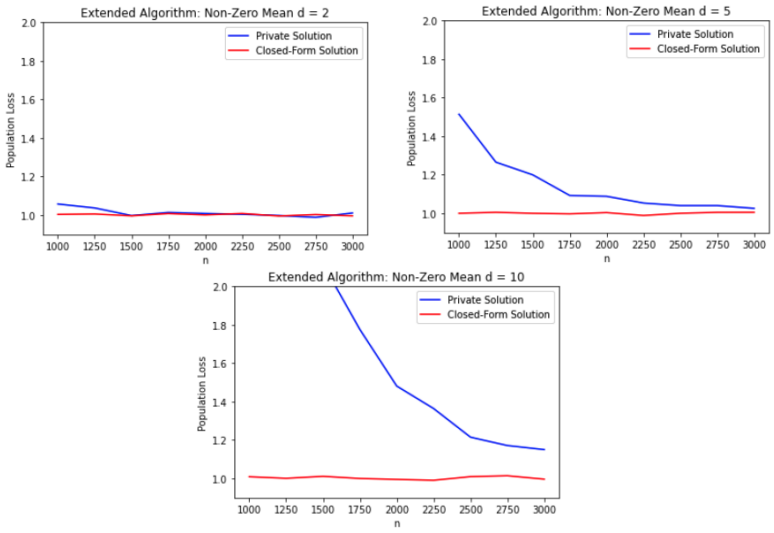

Figure 9: Low Dimensional Comparison: $\mu = 2$ ($\rho = 0.5$)

Next, we look at the case where μ is higher: $\mu = 10$.

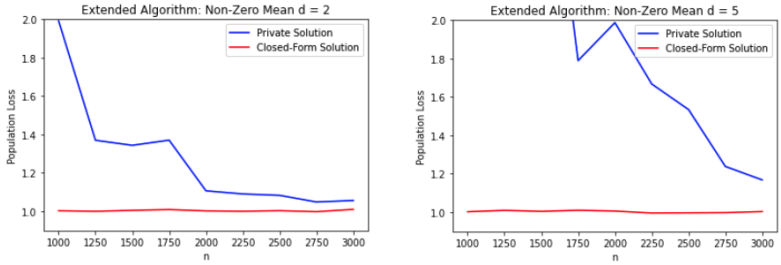

Figure 10: Low Dimensional Comparison: $\mu = 10$ ($\rho = 0.5$)

We also compared the accuracy of our algorithm while varying ρ, the total privacy budget. We saw that with fewer dimensions, our algorithm is still accurate for low values of ρ. This uses the same mean ($\mu = 2$) as figure 9.

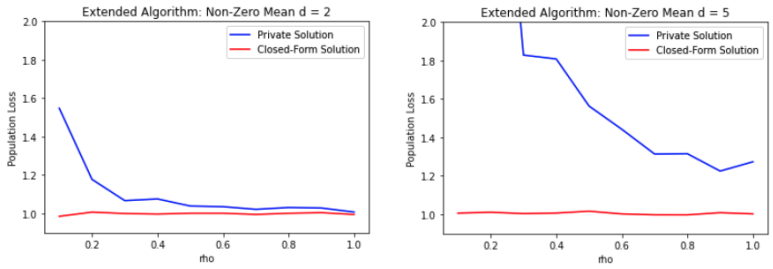

Figure 11: Measuring Effects of Privacy ($d = \{2, 5\}, n = 1000$)

References

[BDKU20] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan Ullman. Coinpress: Practical private mean and covariance estimation. *arXiv preprint arXiv:2006.06618*, 2020.

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.