Differentially Private Distributed Bayesian Linear Regression with MCMC

Barış Alparslan [1] **Sinan Yıldırım** [1] **Ş. İlker Birbil** [2]

Abstract

We propose a novel Bayesian inference framework for distributed differentially private linear regression. We consider a distributed setting where multiple parties hold parts of the data and share certain summary statistics of their portions in privacy-preserving noise. We develop a novel generative statistical model for privately shared statistics, which exploits a useful distributional relation between the summary statistics of linear regression. We propose Bayesian estimation of the regression coefficients, mainly using Markov chain Monte Carlo algorithms, while we also provide a fast version that performs approximate Bayesian estimation in one iteration. The proposed methods have computational advantages over their competitors. We provide numerical results on both real and simulated data, which demonstrate that the proposed algorithms provide well-rounded estimation and prediction.

1. Introduction

Linear regression is a mathematical method that lies at the core of statistical research. Many researchers have been working on linear regression since the 19th century, and hence, many well-known solution methods exist. On a separate note, privacy-preserving statistical learning has gained popularity and importance in recent years, with *differential privacy* prevailing as the most commonly used definition for privacy (Dwork, 2006; Dwork et al., 2014a; Dankar & El Emam, 2013). As a result, there is a recent but growing interest in differentially private linear regression.

Many works in the data privacy literature do not mainly focus on regression but are motivated by or can be applied to regression. As an example, differentially private empirical risk minimisation (Chaudhuri et al., 2011; Bassily et al., 2014; Abadi et al., 2016; Kuru et al., 2022) can be applied to regression once it is cast as a data-driven optimisation problem. Many general-purpose Bayesian differentially private estimation methods can also be used in regression problems. Williams & Mcsherry (2010) is one of the first works that considered a hierarchical model for the privatised data and Bayesian estimation for the model parameters. Zhang et al. (2016) analyse several differential privacy mechanisms for posterior sampling and suggest using these mechanisms also for linear regression. Dimitrakakis et al. (2017) developed a posterior sampling query algorithm to combine differential privacy and Bayesian inference. Contrary to those one-sample approaches, general-purpose differentially private Markov chain Monte Carlo (MCMC) algorithms, which aim to identify the whole posterior distribution via iterative sampling, can also be applied to regression (Wang et al., 2015; Foulds et al., 2016; Wang et al., 2015; Yıldırım & Ermiş, 2019; Heikkilä et al., 2019; Gong, 2022; Alparslan & Yıldırım, 2022; Ju et al., 2022).

Several works in the literature are somewhat more directly related to differentially private regression. Zhang et al. (2012) have suggested a functional mechanism method, which is based on perturbing polynomial objective functions with privacy-preserving noise. As an alternative, Dwork et al. (2014b); Wang (2018) considered perturbation of summary statistics. Alabi et al. (2022) provide a technical discussion on different point estimation methods for differentially private simple linear regression, (that is when we have a single feature). Ferrando et al. (2022) present a method to compute confidence intervals for the coefficients of linear regression. Cai et al. (2021) study the rates of convergence for parameter estimation with differential privacy via output perturbation, where a non-private estimator is perturbed. All those works consider point estimation of the linear regression parameters.

In this paper, we focus on differentially private distributed Bayesian inference for the parameters of linear regression. We use a novel hierarchical model that relies on a distributional relationship (Proposition 3.1) between the summary statistics of linear regression, which, to the best of our knowledge, has not been exploited so far. We propose Bayesian inference algorithms that take perturbations of summary statistics as observations. The general inferential

[1]Faculty of Engineering and Sciences, Sabancı University, Turkey [2]Amsterdam Business School, University of Amsterdam, The Netherlands. Correspondence to: Barış Alparslan <baris.alparslan@sabanciuniv.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

tool we pick in this paper is MCMC, a well-known framework for iterative sampling from posterior distributions. As we shall see, the proposed MCMC algorithms in this paper already have lower computational complexities per iteration than their closest competitors in Bernstein & Sheldon (2019). Additionally, we also propose much faster Bayesian estimation methods that perform estimation in one iteration. Finally, for the sake of generality, we assume a distributed setting where the total dataset is shared among multiple parties (data nodes), who want to collaborate for the inference of a common parameter, see *e.g.*, Heikkilä et al. (2017) for such a setting. The non-distributed setting is just a special case (single data holder) for our methodology.

This paper has connections with several works in the literature, yet it has significant differences from each of those, as we shall explain below.

For the privacy-preserving mechanism, we consider adding noise to summary statistics of linear regression, similarly to Wang (2018); Bernstein & Sheldon (2019). The adaSSP framework of Wang (2018) also motivates the fast Bayesian estimation methods developed in this paper. However, adaSSP is a point estimation method while we aim for a posterior distribution. The latter work, Bernstein & Sheldon (2019), is particularly related to this paper as they also study Bayesian linear regression with differential privacy using perturbed statistics of data. However, there are some important differences between our work and that of Bernstein & Sheldon (2019). These differences stem from the choice of summary statistics and the consequent hierarchical structure used for modelling linear regression. Those modelling differences lead to significant differences in the inference methods as well as significant computational advantages for our methods. Specifically, the computational complexity of our methods is $\mathcal{O}(d^3)$, where d is the number of features. This order is much less than $\mathcal{O}(d^6)$ of Bernstein & Sheldon (2019). Finally, neither Wang (2018) nor Bernstein & Sheldon (2019) has considered a distributed learning setting as we do in this paper, although both works can be modified for the distributed setting after moderate modifications.

Foulds et al. (2016) and Heikkilä et al. (2017) are other differentially Bayesian inference methods that target posterior distributions of perturbed summary statistics of sensitive data. Heikkilä et al. (2017) is particularly interesting because they consider a distributed setting and present linear regression as their showcase example. However, we differ from those works in the way we model the perturbed statistics and in the choice of inference methods. Specifically, Foulds et al. (2016); Heikkilä et al. (2017) treat the perturbed statistics as if not perturbed, while we correctly incorporate the effect of perturbation in our model.

Recently, Alparslan & Yıldırım (2022) and Ju et al. (2022) employ data augmentation for modelling sensitive and privatised data and propose MCMC for Bayesian inference, the latter having linear regression as a major application. Their methods have $\mathcal{O}(n)$ complexity per iteration in general where n is the number of instances in the data set, which can be slow when n is large. In contrast, our methods are scalable in data size since their computational complexities do not depend on n. We note that Alparslan & Yıldırım (2022, Section 4.2) also present an MCMC method scalable with n that exploits the approximate normality of additive summary statistics. However, a direct application of that would lead to an algorithm with $\mathcal{O}(d^6)$ computational complexity (per iteration), like in Bernstein & Sheldon (2019).

The paper is organised as follows: In Section 2, we review differential privacy. In Section 3, we lay out the hierarchical model for differentially private distributed linear regression with perturbed summary statistics. In Section 4, we present and discuss the aspects of the proposed inference algorithms. In Section 5, we provide numerical experiments. We conclude in Section 6.

Notation: Matrices and vectors are shown in bold-face notation. For a matrix $\boldsymbol{A}$, its transpose, trace, and determinant (if they exist) are $\boldsymbol{A}^T$, $\text{tr}(\boldsymbol{A})$, and $|\boldsymbol{A}|$, respectively. $\boldsymbol{I}_d$ is the $d \times d$ identity matrix. For any sequence $\{a_i\}_{i\geq 0}$, we write $a_{i:j}$ for $(a_i, \ldots, a_j)$. We write $x \sim P$ to mean the random variable x has distribution P. $\mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma})$ stands for the multivariate normal distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma}$. The Wishart and inverse-Wishart distributions, each with scale matrix $\boldsymbol{\Lambda}$ and κ degrees of freedom, are shown as $\mathcal{W}(\boldsymbol{\Lambda}, \kappa)$ and $\mathcal{IW}(\boldsymbol{\Lambda}, \kappa)$, respectively. $\mathcal{IG}(a, b)$ stands for the inverse-gamma distribution with shape and scale parameters a and b. We augment those notations, *e.g.,* with $\boldsymbol{x}$, to denote the respective probability density functions (pdf), *e.g.*, $\mathcal{N}(\boldsymbol{x}; \boldsymbol{m}, \boldsymbol{\Sigma})$.

2. Differential Privacy

Differential privacy (Dwork, 2006; 2008) concerns randomised algorithms that run on sensitive, or usually private, data. A randomised algorithm takes an input data set $D \in \mathcal{D}$ and returns a random output in $\mathcal{O}$ where the randomness is intrinsic to the algorithm. A differentially private algorithm constrains the difference between the probability distributions of the output values obtained from neighbouring data sets. We say two data sets are *neighbours* if they have the same size and differ by a single element, corresponding to a single individual's piece of data.

Definition 2.1 (Differential privacy)**.** A randomised algorithm $M : \mathcal{D} \mapsto \mathcal{O}$ is (ϵ, δ)-differentially private (DP) if for any pair of neighbouring data sets $D, D' \in \mathcal{D}$ and for any subset $O \subseteq \mathcal{O}$ of the of support domain, it satisfies

$$\mathbb{P}[M(D) \in O] \leq e^{\epsilon}\mathbb{P}[M(D') \in O] + \delta.$$

The definition implies that we have more privacy with smaller (ϵ, δ) pair. Privacy-preserving algorithms often use noise-adding mechanisms. A popular noise-adding mechanism is the *Gaussian mechanism* (Dwork et al., 2006), which perturbs a function $f : \mathcal{D} \mapsto \mathbb{R}^k$ of the sensitive data, for some $k \geq 1$, with a random noise drawn from the Gaussian distribution. The amount of the added noise depends on the L_2-*sensitivity* of the function, given by

$$\Delta_f = \max_{\text{neighbour} D_1, D_2 \in \mathcal{D}} \|f(D_1) - f(D_2)\|_2.$$

An (ϵ, δ)-DP Gaussian mechanism returns

$$f(D) + \Delta_f \sigma(\epsilon, \delta) \boldsymbol{v}, \quad \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_k) \tag{1}$$

upon taking D as the input, where the quantity $\sigma(\epsilon, \delta)$ ensures (ϵ, δ)-DP. In this work, we take $\sigma(\epsilon, \delta)$ as the analytical solution given in Balle & Wang (2018, Algorithm 1) due to its tightness. The Gaussian mechanism is also central to other forms of privacy, such as zero-concentrated DP (Bun & Steinke, 2016) and Gaussian DP (Dong et al., 2022).

This paper considers (ϵ, δ)-DP as the type of privacy and the Gaussian mechanism to generate noisy observations. Moreover, the proposed methods in this paper never use the sensitive data given the noisy observations generated using the Gaussian mechanism, hence exploiting the *post-processing* property of differential privacy (Dwork & Roth, 2014).

Theorem 2.2 (Post-processing)**.** *If $M : \mathcal{D} \mapsto \mathcal{O}$ be (ϵ, δ)-DP and let $f : O \to O'$ be another mapping independent of D given $M(D)$. Then $f_M : \mathcal{D} \mapsto \mathcal{O}'$ with $f_M(D) = f(M(D))$ is (ϵ, δ)-DP.*

3. Differentially Private Distributed Linear Regression

This section presents a new hierarchical model for differentially private distributed linear regression. For ease of exposition, we first present a model with a single data holder, then generalise the model for the distributed setting.

3.1. Basic Model and Privacy Setup

Suppose we have a sequence of random variables $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$, where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^{d \times 1}$ are the feature vectors and $y_i \in \mathcal{Y} \subseteq \mathbb{R}$ is the i'th response variable. We consider the normal linear regression to model the dependency between $\boldsymbol{x}_i$ and y_i. Specifically,

$$y_i = \boldsymbol{x}_i^T \boldsymbol{\theta} + e_i, \quad e_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_y^2), \quad i = 1, \ldots, n,$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector of the linear regression coefficients. We assume that the feature vectors $\boldsymbol{x}_i$'s are i.i.d. with distribution $P_{\boldsymbol{x}}$. A particular case of interest will be one where $P_{\boldsymbol{x}}$ can be assumed to be a normal distribution. However, we will also present algorithms for general $P_{\boldsymbol{x}}$ that is not (even approximately) normal.

In matrix notation, the above can shortly be expressed as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{e}, \quad \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2 \boldsymbol{I}_n),$$

where $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T & \ldots & \boldsymbol{x}_n^T \end{bmatrix}^T$ is the so-called design matrix, $\boldsymbol{y} = \begin{bmatrix} y_1 & \ldots & y_n \end{bmatrix}^T$. Additionally, we also define the summary statistics of $\boldsymbol{X}$ and $\boldsymbol{y}$ given by

$$\boldsymbol{S} := \boldsymbol{X}^T \boldsymbol{X} \text{ and } \boldsymbol{z} := \boldsymbol{X}^T \boldsymbol{y},$$

respectively. In this paper, we assume a data-sharing scenario where $\boldsymbol{S}$ and $\boldsymbol{z}$ are privately released as the noisy summary statistics $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{z}}$, constructed as

$$\hat{\boldsymbol{S}} = \boldsymbol{S} + \sigma_s \boldsymbol{M}, \tag{2}$$

$$\hat{\boldsymbol{z}} = \boldsymbol{z} + \sigma_z \boldsymbol{v}, \quad \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d), \tag{3}$$

where $\boldsymbol{M}$ is a $d \times d$ symmetric matrix with its upper triangular elements drawn from $\mathcal{N}(0, 1)$. Dwork et al. (2014b) arrange σ_s and σ_z so that both (2) and (3) are $(\epsilon/2, \delta/2)$ differentially private, leading to (ϵ, δ)-DP overall. Differently than Dwork et al. (2014b), we set

$$\sigma_s = \sigma_z = \Delta_{sz} \sigma(\epsilon, \delta),$$

where $\sigma(\epsilon, \delta)$ is given in Balle & Wang (2018, Algorithm 1), and Δ_{sz} is the overall L_2 sensitivity of $[\boldsymbol{S}, \boldsymbol{z}]$, given by

$$\Delta_{sz} = \sqrt{\|X\|^4 + \|X\|^2 \|Y\|^2}$$

with $\|X\| = \max_{\boldsymbol{x} \in \mathcal{X}} \|\boldsymbol{x}\|_2$ and $\|Y\| = \max_{y \in \mathcal{Y}} |y|$.

Finally, we assign prior distributions for $\boldsymbol{\theta}, \sigma_y^2$ as

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{C}), \quad \sigma_y^2 \sim \mathcal{IG}(a, b). \tag{4}$$

Based on the above relations, we shall represent a hierarchical model that enables Bayesian inference of $\boldsymbol{\theta}$ given $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{z}}$. One important element of our modelling approach is the following result that establishes the conditional distribution of $\boldsymbol{z}$ given $\boldsymbol{S}$, $\boldsymbol{\theta}$, and σ_y^2.

Proposition 3.1. *For the normal linear regression model, we have*

$$\boldsymbol{z} | \boldsymbol{S}, \boldsymbol{\theta}, \sigma_y^2 \sim \mathcal{N}(\boldsymbol{S}\boldsymbol{\theta}, \boldsymbol{S}\sigma_y^2).$$

Proof. First, note that,

$$\mathbb{E}[\boldsymbol{z} | \boldsymbol{X}, \boldsymbol{\theta}, \sigma_y^2] = \mathbb{E}[\boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\theta} + \boldsymbol{X}^T \boldsymbol{e}]$$
$$\text{Cov}(\boldsymbol{z} | \boldsymbol{X}, \boldsymbol{\theta}, \sigma_y^2) = \boldsymbol{X}^T \boldsymbol{X} \sigma_y^2$$

Hence, the conditional density of $\boldsymbol{z}$ given $\boldsymbol{X}$, $\boldsymbol{\theta}$, and σ_y^2 is

$$p(\boldsymbol{z} | \boldsymbol{X}, \boldsymbol{\theta}, \sigma_y^2) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{\theta}, \boldsymbol{X}^T \boldsymbol{X} \sigma_y^2). \tag{5}$$

Let ν and ω denote the probability distributions of $\boldsymbol{x}$ and $\boldsymbol{S}$, respectively. By change of variables $\boldsymbol{S} = \boldsymbol{X}^T\boldsymbol{X}$, we have

$$\int f(\boldsymbol{S})\omega(\mathrm{d}\boldsymbol{S}) = \int f(\boldsymbol{X}^T\boldsymbol{X})\nu(\mathrm{d}\boldsymbol{X}) \tag{6}$$

for any real-valued measurable function $f : \mathcal{S}_d \mapsto \mathbb{R}$, from the set $\mathcal{S}_d$ of $d \times d$ positive definite matrices. Also, for any pair of sets $A \subseteq \mathcal{S}_d$ and $B \in \mathbb{R}^{d\times 1}$, we can write

$$\mathbb{P}(\boldsymbol{S} \in A, \boldsymbol{z} \in B) = \int \mathbb{I}_A(\boldsymbol{X}^T\boldsymbol{X})\mathbb{P}(\boldsymbol{z} \in B|\boldsymbol{X})\nu(\mathrm{d}\boldsymbol{X}).$$

The integrand above depends on $\boldsymbol{X}$ *through* $\boldsymbol{X}^T\boldsymbol{X}$ *only*, since $\mathbb{P}(\boldsymbol{z} \in B|\boldsymbol{X}) = \int_B \mathcal{N}(\boldsymbol{z}; \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\theta}, \boldsymbol{X}^T\boldsymbol{X}\sigma_y^2)\mathrm{d}\boldsymbol{z}$. Therefore, applying the relation in (6) to the RHS with the choice $f(\boldsymbol{S}) = \mathbb{I}_A(\boldsymbol{S})\int_B \mathcal{N}(\boldsymbol{z}; \boldsymbol{S}\boldsymbol{\theta}, \boldsymbol{S}\sigma_y^2)\mathrm{d}\boldsymbol{z}$, we can rewrite the joint probability as

$$\begin{aligned}\mathbb{P}(\boldsymbol{S} \in A, \boldsymbol{z} \in B) &= \int \mathbb{I}_A(\boldsymbol{S}) \int_B \mathcal{N}(\boldsymbol{z}; \boldsymbol{S}\boldsymbol{\theta}, \boldsymbol{S}\sigma_y^2)\mathrm{d}\boldsymbol{z}\omega(\mathrm{d}\boldsymbol{S}) \\ &= \int_A \int_B \mathcal{N}(\boldsymbol{z}; \boldsymbol{S}\boldsymbol{\theta}, \boldsymbol{S}\sigma_y^2)\mathrm{d}\boldsymbol{z}\omega(\mathrm{d}\boldsymbol{S}).\end{aligned}$$

This shows that the variables $\boldsymbol{z}, \boldsymbol{S}$ have the joint probability distribution $\omega(\mathrm{d}\boldsymbol{S})\mathcal{N}(\boldsymbol{z}; \boldsymbol{S}\boldsymbol{\theta}, \boldsymbol{S}\sigma_y^2)\mathrm{d}\boldsymbol{z}$ and that the conditional probability density is $p(\boldsymbol{z}|\boldsymbol{S}, \boldsymbol{\theta}, \sigma_y^2) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{S}\boldsymbol{\theta}, \boldsymbol{S}\sigma_y^2)$, as claimed. □

At this point, some important modelling differences between our work and Bernstein & Sheldon (2019) are worth discussing. In Bernstein & Sheldon (2019), the central limit theorem (CLT) is applied to $\left[\boldsymbol{S}, \boldsymbol{z}, \boldsymbol{y}^T\boldsymbol{y}\right]$, leading to a normality assumption for the whole vector. In contrast, we use the *exact* conditional distribution $p(\boldsymbol{z}|\boldsymbol{S}, \boldsymbol{\theta}, \sigma^2)$ thanks to Proposition 3.1. Moreover, unlike Bernstein & Sheldon (2019), we do *not* require a noisy version $\boldsymbol{y}^T\boldsymbol{y}$, hence have a slight advantage of using less privacy-preserving noise. In summary, our model has a different hierarchical structure and requires less privacy-preserving noise.

3.2. Distributed Setting

Here we extend our model to the distributed setting, where the total data are shared among $J \geq 1$ data holders as

$$(\boldsymbol{X}, \boldsymbol{y}) = \{(\boldsymbol{X}_j, \boldsymbol{y}_j); j = 1, \ldots, J\}. \tag{7}$$

We let n_i be number of rows in each $\boldsymbol{x}_i$, so that $n = n_1 + \ldots + n_J$. Each data holder j shares their own summary statistics $\boldsymbol{S}_j = \boldsymbol{X}_j^T\boldsymbol{X}_j$, $\boldsymbol{z}_j = \boldsymbol{X}_j^T\boldsymbol{y}_j$ with privacy-preserving noise

$$\begin{aligned}\hat{\boldsymbol{S}}_j &= \boldsymbol{S}_j + \sigma_s\boldsymbol{M}_j, \\ \hat{\boldsymbol{z}}_j &= \boldsymbol{z} + \sigma_z^2\boldsymbol{v}_j, \quad \boldsymbol{v}_j \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d).\end{aligned} \tag{8}$$

Note that, to preserve a given (ϵ, δ)-DP overall, each party must provide that level of privacy for their data, hence σ_s^2 and σ_z^2 are the same as before. The hierarchical structure of the overall model (specified for normally distributed $\boldsymbol{x}_i$'s) is shown in Figure 1.

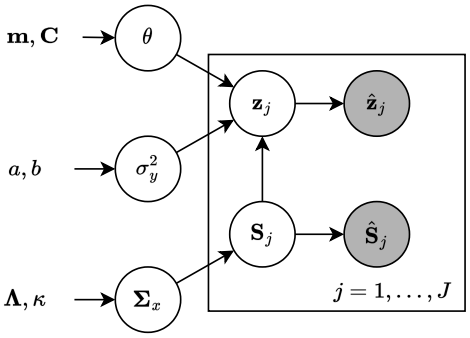

Figure 1. Differentially private distributed linear regression model (specified for normally distributed $\boldsymbol{x}_i$'s.)

The distributed setting deserves separate consideration than the single data holder case for a couple of reasons: Firstly, the node-specific observations $(\hat{\boldsymbol{S}}_1, \hat{\boldsymbol{z}}_1), \ldots, (\hat{\boldsymbol{S}}_J, \hat{\boldsymbol{z}}_J)$ are altogether statistically *more informative* on θ than their aggregates $\sum_{j=1}^J \hat{\boldsymbol{S}}_j$ and $\sum_{j=1}^J \hat{\boldsymbol{z}}_j$. This is because the aggregate versions are *not* sufficient statistics of the node-specific observations $(\hat{\boldsymbol{S}}_1, \hat{\boldsymbol{z}}_1), \ldots, (\hat{\boldsymbol{S}}_J, \hat{\boldsymbol{z}}_J)$ with respect to θ (even when σ_y^2 is known.) Therefore, when the node-specific observations are available, one should not, in principle, trivially aggregate them and apply an inference method designed for $J = 1$ using those aggregates.

Secondly, the partitioning of data as in (7) can be relevant to data privacy applications even *outside* the distributed learning framework, rendering the methodology in Section 4 useful in a broader sense. For example, batches of $(\boldsymbol{x}, y)$-type of data may be donated to a common data collector as in (8). At this point, a particular and interesting relation exists with pan-privacy applications (Dwork et al., 2010). Imagine that sensitive data from individuals are collected sequentially in time and the data holder is concerned about possible intrusions into the memory where the sensitive data are stored. Then, one possible way to ensure the privacy of the data against such possible intrusions, which is the promise of pan-privacy, is to store the noisy statistics of every new batch of data and erase the original sensitive data. Then, at any time, the data collector has data of the form $(\hat{\boldsymbol{S}}_1, \hat{\boldsymbol{z}}_1), \ldots, (\hat{\boldsymbol{S}}_J, \hat{\boldsymbol{z}}_J)$, each pair corresponding to a batch, the Jth pair being the current batch. As a result, inference algorithms in Section 4 can be applied.

4. Algorithms for Bayesian Inference

Bayesian inference targets the posterior distribution of the latent variables of the model, in particular $\boldsymbol{\theta}$, given the observations $\hat{\boldsymbol{S}}_{1:J}$ and $\hat{\boldsymbol{z}}_{1:J}$. We present several Bayesian

inference algorithms for the hierarchical model described in the previous section. In addition to other concerns like computational budget, the choice among those approaches mainly depends on the specification of $P_{\boldsymbol{x}}$ as the distribution of $\boldsymbol{S}$ directly depends on it. In this paper, we consider the following two cases and devise algorithms for each of them:

1. In some cases it may be adequate to specify $P_{\boldsymbol{x}} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_x)$. This leads to $\boldsymbol{S}|\boldsymbol{\Sigma}_x \sim \mathcal{W}(\boldsymbol{\Sigma}_x, n)$. Further, to account for the uncertainty about the covariance $\boldsymbol{\Sigma}_x$, one can treat it as a random variable with $\boldsymbol{\Sigma}_x \sim \mathcal{IW}(\boldsymbol{\Lambda}, \kappa)$. Figure 1 shows the hierarchical structure of the distributed setting with those specifications. We defer discussing the conflict between the normality and boundedness assumptions to Remark 4.1 towards the end of Section 4.1.

2. As the second case, we assume a general (non-normal) $P_{\boldsymbol{x}}$. A normal approximation, based on the CLT, could be considered for the distribution $\boldsymbol{S}$ (Wilson & Ghahramani, 2011). However, this would require the knowledge (or accurate estimation) of up to the fourth moments of $P_{\boldsymbol{x}}$ as well as expensive computations for sampling $\boldsymbol{S}$. We circumvent those difficulties by plugging in a point estimate of $\boldsymbol{S}$ given $\hat{\boldsymbol{S}}$ and use it during the sampling process as if it is the true $\boldsymbol{S}$ itself. Then, we develop two different algorithms for inference of $\boldsymbol{\theta}$, one being an MCMC algorithm and the other providing a closed form-solution for the posterior of $\boldsymbol{\theta}$ following a rough point-wise estimation of σ_y^2. Note that these algorithms with fixed $\boldsymbol{S}$ do not require a distribution for $\boldsymbol{x}$.

Next, we provide the details of our approaches and the resulting algorithms.

4.1. MCMC for Normally Distributed Features

In this section, we present an MCMC algorithm for Bayesian inference for the differentially private distributed linear regression model when $P_{\boldsymbol{x}} = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_x)$ and $\boldsymbol{\Sigma}_x \sim \mathcal{IW}(\Lambda, \kappa)$. The latent variables involved in this variant are $\boldsymbol{\theta}, \boldsymbol{\Sigma}_x, \sigma_y^2, \boldsymbol{S}_{1:J}, \boldsymbol{z}_{1:J}$. Their posterior distribution given $\hat{\boldsymbol{S}}_{1:J}, \hat{\boldsymbol{z}}_{1:J}$ can be written as

$$p(\boldsymbol{\theta},\sigma_y^2, \boldsymbol{\Sigma}_x, \boldsymbol{z}_{1:J}, \boldsymbol{S}_{1:J}|\hat{\boldsymbol{z}}_{1:J}, \hat{\boldsymbol{S}}_{1:J}) \propto p(\boldsymbol{\theta})p(\sigma_y^2)p(\boldsymbol{\Sigma}_x) \prod_{j=1}^{J} p(\boldsymbol{z}_j|\boldsymbol{\theta}, \sigma_y^2, \boldsymbol{S})p(\boldsymbol{S}_j|\boldsymbol{\Sigma}_x)p(\hat{\boldsymbol{S}}_j|\boldsymbol{S}_j)p(\hat{\boldsymbol{z}}_j|\boldsymbol{z}_j). \quad (9)$$

One could design an MCMC algorithm for this posterior distribution that updates $\boldsymbol{\theta}$, σ_y^2, $\boldsymbol{\Sigma}_x$, $\boldsymbol{z}_{1:J}$, $\boldsymbol{S}_{1:J}$ in turn based on their full conditional distributions. However, such an algorithm suffers from poor convergence because of a high posterior correlation between $\boldsymbol{\theta}$ and $\boldsymbol{z}_{1:J}$ (as verified in our numerical studies). It is well known that highly correlated variables result in poor convergence if they are updated one conditional on the other. To alleviate that problem, we work with the reduced model where $\boldsymbol{z}_{1:J}$ is integrated out. The reduced model has $\boldsymbol{\theta}, \boldsymbol{\Sigma}_x, \sigma_y^2$ as its latent variables, whose joint posterior distribution can be written as

$$p(\boldsymbol{\theta}, \sigma_y^2,\boldsymbol{\Sigma}_x, \boldsymbol{S}|\hat{\boldsymbol{z}}, \hat{\boldsymbol{S}}) \propto p(\boldsymbol{\theta})p(\sigma_y^2)p(\boldsymbol{\Sigma}_x) \prod_{j=1}^{J} p(\boldsymbol{S}_j|\boldsymbol{\Sigma}_x)p(\hat{\boldsymbol{S}}_j|\boldsymbol{S}_j)p(\hat{\boldsymbol{z}}_j|\boldsymbol{S}_j, \boldsymbol{\theta}, \sigma_y^2), \quad (10)$$

where $p(\hat{\boldsymbol{z}}|\boldsymbol{S}, \boldsymbol{\theta}, \sigma_y^2) = \mathcal{N}(\hat{\boldsymbol{z}}; \boldsymbol{S}\boldsymbol{\theta}, \sigma_y^2\boldsymbol{S}\boldsymbol{\theta} + \sigma_z^2\boldsymbol{I}_d)$.

We would like to sample from the posterior distribution in (10) via MCMC that updates $\boldsymbol{\theta}$, σ_y^2, $\boldsymbol{\Sigma}_x$, $\boldsymbol{S}_{1:J}$ in turn based on their full conditional distributions. The variables $\boldsymbol{\theta}$ and $\boldsymbol{\Sigma}_x$ enjoy closed-form full conditional distributions (see Appendix A for the derivations):

$$\boldsymbol{\Sigma}_x|\boldsymbol{S}_{1:J}, \hat{\boldsymbol{S}}_{1:J}, \hat{\boldsymbol{z}}_{1:J} \sim \mathcal{IW}\left(\boldsymbol{\Lambda} + \sum_{j=1}^{J} \boldsymbol{S}_j, \kappa + n\right), \quad (11)$$

$$\boldsymbol{\theta}|\sigma_y^2, \hat{\boldsymbol{z}}, \boldsymbol{S}_{1:J} \sim \mathcal{N}(\boldsymbol{m}_p, \boldsymbol{\Sigma}_p), \quad (12)$$

where the posterior moments for $\boldsymbol{\theta}$ are

$$\boldsymbol{\Sigma}_p^{-1} = \sum_{j=1}^{J} \boldsymbol{S}_j(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\boldsymbol{S}_j + \boldsymbol{C}^{-1},$$

$$\boldsymbol{m}_p = \boldsymbol{\Sigma}_p\left(\sum_{j=1}^{J} \boldsymbol{S}_j(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\hat{\boldsymbol{z}}_j + \boldsymbol{C}^{-1}\boldsymbol{m}\right).$$

The full-conditional distributions of $\boldsymbol{S}_{1:J}$ and σ_y^2 have no closed form; hence, we design Metropolis-Hastings (MH) moves to update them. For σ_y^2, one can simply use a random-walk MH move targeting $p(\sigma_y^2|\boldsymbol{\theta}, \boldsymbol{S}_{1:J}, \hat{\boldsymbol{z}}_{1:J})$. For $\boldsymbol{S}_{1:J}$, their full conditional distribution can be factorised as

$$p(\boldsymbol{S}_{1:J}|\hat{\boldsymbol{S}}_{1:J}, \hat{\boldsymbol{z}}_{1:J}, \boldsymbol{\Sigma}_x, \sigma_y^2, \boldsymbol{\theta}) = \prod_{j=1}^{J} p(\boldsymbol{S}_j|\hat{\boldsymbol{S}}_j, \hat{\boldsymbol{z}}_j, \boldsymbol{\Sigma}_x, \sigma_y^2, \boldsymbol{\theta}),$$

where each factor is given by

$$p(\boldsymbol{S}_j|\hat{\boldsymbol{S}}_j, \hat{\boldsymbol{z}}_j, \boldsymbol{\Sigma}_x, \sigma_y^2, \boldsymbol{\theta}) \propto p(\hat{\boldsymbol{z}}_j|\boldsymbol{S}_j, \boldsymbol{\theta}, \sigma_y^2)p(\boldsymbol{S}_j|\boldsymbol{\Sigma}_x)p(\hat{\boldsymbol{S}}_j|\boldsymbol{S}_j).$$

Thanks to that factorised form, each $\boldsymbol{S}_j$ can be updated with an MH move independently and in parallel. For the MH algorithm to update one $\boldsymbol{S}_j$, we propose a new value from a Wishart distribution as $\boldsymbol{S}'_j \sim \mathcal{W}(\boldsymbol{S}_j/\alpha, \alpha)$, which has mean $\boldsymbol{S}_j$ and variance determined by $\alpha > 0$. In our experiments, we adjust α using ideas from the adaptive

Algorithm 1 `MCMC-normalX` - one iteration

Input: Current values of $\boldsymbol{S}_{1:J}$, $\boldsymbol{\theta}$, σ_y^2, $\boldsymbol{\Sigma}_x$; observations $\hat{\boldsymbol{S}}_{1:J}$,$\hat{\boldsymbol{z}}_{1:J}$; noise variances σ_s^2, σ_z^2; proposal parameters a, σ_q^2; hyperparameters $a, b, \kappa, \boldsymbol{\Lambda}, \boldsymbol{m}, \boldsymbol{C}$.
Output: New sample of $\boldsymbol{\Sigma}_x, \boldsymbol{S}, \sigma_y^2, \boldsymbol{\theta}$
Sample $\boldsymbol{\Sigma}_x$ using (11).
for $j = 1, 2, \dots J$ **do**
  Update $\boldsymbol{S}_j$ via an MH move targeting $p(\boldsymbol{S}_j|\boldsymbol{\Sigma}_x, \boldsymbol{\theta}, \hat{\boldsymbol{z}}_j)$.
Sample $\boldsymbol{\theta}$ using (12).
Update σ_y^2 via an MH move targeting $p(\sigma_y^2|\boldsymbol{\theta}, \boldsymbol{S}_{1:J}, \hat{\boldsymbol{z}}_{1:J})$.

Algorithm 2 `MCMC-fixedS` - one iteration

Input: Current values of $\boldsymbol{\theta}$, σ_y^2; estimates $\widetilde{\boldsymbol{S}}_{1:J}$, observations $\hat{\boldsymbol{z}}_{1:J}$; noise variance σ_z^2, and hyperparameters a, b, $\boldsymbol{m}$, $\boldsymbol{C}$.
Output: New sample of $\sigma_y^2, \boldsymbol{\theta}$.
Use $\boldsymbol{S}_{1:J} = \widetilde{\boldsymbol{S}}_{1:J}$ throughout.
Sample $\boldsymbol{\theta}$ using (12).
Update σ_y^2 via an MH move targeting $p(\sigma_y^2|\boldsymbol{\theta}, \boldsymbol{S}_{1:J}, \hat{\boldsymbol{z}}_{1:J})$.

MCMC framework (Andrieu & Thoms, 2008) to target an acceptance rate of around 0.2.

Algorithm 1 represents the overall MCMC algorithm for the hierarchical model for differentially Bayesian distributed linear regression when $P_{\boldsymbol{x}}$ is a normal distribution with a random covariance matrix having an inverse-Wishart distribution. We call this algorithm `MCMC-normalX`.

Remark 4.1. Admittedly, a potential concern is a conflict between the normality and boundedness assumptions (both for $\boldsymbol{x}$ and y). However, we also note that the collected data often happen to have some natural boundaries (which can be exploited to determine the sensitivity of the shared statistics), and yet the normal distribution is still used for modelling and subsequent inference mainly for the sake of tractability. With the normality assumption, one can implement computationally efficient algorithms at the expense of minor modelling inaccuracies. While we acknowledge the methodologies in Alparslan & Yıldırım (2022, Section 4.2) and Ju et al. (2022) that can correctly incorporate the effect of truncation into inference, we remark that those methods pay the price of exactness by having $\mathcal{O}(n)$ computational complexity per iteration.

4.2. MCMC for Non-normally Distributed Features

The normality assumption for $\boldsymbol{x}_i$'s in Section 4.1 may not be adequate for some data sets. Moreover, when d is large, updating $\boldsymbol{S}_j$'s can be the bottleneck of `MCMC-normalX` in Algorithm 1 in terms of computation time and convergence. We propose two algorithms to address both of those concerns. As it turns out, those algorithms provide accurate estimations even for the case of normally distributed features; see Section 5.1.

Our approach for non-normal $\boldsymbol{x}_i$'s is based on estimating $\boldsymbol{S}_j$'s from $\hat{\boldsymbol{S}}_j$s at the beginning, using some principled estimation method, and fixing $\boldsymbol{S}_j$'s to those estimates during the whole course of the inference procedure. In that way, we obtain a faster version of `MCMC-normalX` that is also well-suited for non-normal $\boldsymbol{x}_i$'s. Indeed, we observed in our experiments that this method outperforms the other methods for most of the cases, especially when the total number of nodes J increases. We call this variant `MCMC-fixedS` and present it in Algorithm 2.

As for estimating $\boldsymbol{S}_j$'s, one could simply take the privately shared $\hat{\boldsymbol{S}}_j$ as an estimator for $\boldsymbol{S}_j$, but $\hat{\boldsymbol{S}}_j$ is not necessarily a positive (semi-)definite matrix. Instead, we consider the nearest positive semi-definite matrix to $\hat{\boldsymbol{S}}_j$ in terms of the Frobenius norm as the estimator of $\boldsymbol{S}_j$. (The nearest *positive* definite matrix to $\hat{\boldsymbol{S}}_j$ does not exist.) To find the nearest positive semi-definite matrix, we follow Higham (1988) and apply the following procedure for each $j = 1, \dots, J$: (i) Calculate the eigendecomposition $\widehat{\boldsymbol{S}}_j = \boldsymbol{E}\boldsymbol{D}\boldsymbol{E}^T$, where $\boldsymbol{E}$ is a matrix of eigenvectors, and $\boldsymbol{D}$ is a diagonal matrix consisting of the eigenvalues λ_i. (ii) The nearest symmetric positive semi-definite matrix is $\widetilde{\boldsymbol{S}}_j = \boldsymbol{E}\boldsymbol{D}_+\boldsymbol{E}^T$, where $\boldsymbol{D}_+$ is a diagonal matrix with $\boldsymbol{D}_+(i,i) = \max\{\boldsymbol{D}(i,i), 0\}$.

Note that $\widetilde{\boldsymbol{S}}_j$ found above is the maximum likelihood estimator of $\boldsymbol{S}_j$ given $\hat{\boldsymbol{S}}_j$ (over the set of positive semi-definite matrices) since the conditional distribution of $\hat{\boldsymbol{S}}_j$ given $\boldsymbol{S}_j$ is a normal distribution with mean $\boldsymbol{S}_j$.

Algorithm 2 is faster than Algorithm 1, since it avoids the step to update $\boldsymbol{S}_j$'s, which constitutes the main computational burden on Algorithm 1. However, Algorithm 2 can be made even faster by fixing σ_y^2 also. As a crude estimator, we used $\tilde{\sigma}_y^2 = \|Y\|/3$ throughout the experiments. We call the resulting algorithm `Bayes-fixedS-fast` and present it in Algorithm 3. Algorithm 3 does nothing but calculate the moments of the posterior distribution of $\boldsymbol{\theta}$ given $\sigma_y^2 = \tilde{\sigma}_y^2$, $\boldsymbol{S}_j = \widetilde{\boldsymbol{S}}_j$ and $\hat{\boldsymbol{z}}_j$ for $j = 1, \dots, J$, and the prior parameters for $\boldsymbol{\theta}$.

4.3. Computational Cost

All our methods described in this section require $\mathcal{O}(d^3)$ computation (either per iteration for the iterative ones in Algorithms 1 and 2 or as a whole for the fast version in Algorithm 3) since they deal with $d \times d$ matrices. In contrast, since Bernstein & Sheldon (2019) apply CLT to the vector $[\boldsymbol{S}, \boldsymbol{z}, \boldsymbol{y}^T\boldsymbol{y}]$, their methods deal with covariance matrices of size $(d^2 + d + 1)$ *explicitly*, which leads to $\mathcal{O}(d^6)$ computation per MCMC iteration. For even moderate d, this computational difference between $\mathcal{O}(d^6)$ and $\mathcal{O}(d^3)$

Algorithm 3 `Bayes-fixedS-fast`

Input: Observations $\hat{\boldsymbol{S}}_{1:J}$, $\hat{\boldsymbol{z}}_{1:J}$; noise variance: σ_z^2; estimate $\tilde{\sigma}_y^2$ of σ_y^2; hyperparameters: $\boldsymbol{m}$, $\boldsymbol{C}$.
Output: Estimate $\hat{\boldsymbol{\theta}}$.
for $j = 1, 2, \dots J$ **do**
  Calculate the estimate $\widetilde{\boldsymbol{S}}_j$ for $\boldsymbol{S}_j$ using $\hat{\boldsymbol{S}}_j$.
  Calculate $\boldsymbol{U}_j = \widetilde{\boldsymbol{S}}_j(\tilde{\sigma}_y^2\widetilde{\boldsymbol{S}}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\widetilde{\boldsymbol{S}}_j$.
  Calculate $\boldsymbol{u}_j = \widetilde{\boldsymbol{S}}_j(\tilde{\sigma}_y^2\widetilde{\boldsymbol{S}}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\hat{\boldsymbol{z}}_j$.
return Posterior moments of $\boldsymbol{\theta}$: $\boldsymbol{\Sigma}_{\text{post}}^{-1} = \sum_{j=1}^{J}\boldsymbol{U}_j + \boldsymbol{C}^{-1}$, $\boldsymbol{m}_{\text{post}} = \boldsymbol{\Sigma}_{\text{post}}\left(\boldsymbol{C}^{-1}\boldsymbol{m} + \sum_{j=1}^{J}\boldsymbol{u}_j\right)$.

becomes dramatic and the former may be prohibitive in practice.

We also note that the complexity of our methods does not depend on the data size n. This is in contrast to the $\mathcal{O}(n)$ complexity of general-purpose methods applicable to linear regression, such as Alparslan & Yıldırım (2022, Section 4.3) and Ju et al. (2022).

4.4. Extensions

We mention two other variants of our methodology, deferring the details to Appendix B. One solution for dealing with non-normal $P_{\boldsymbol{x}}$ could be to average the feature vectors in $\boldsymbol{X}$ and the corresponding response variables in $\boldsymbol{y}$, so that the averaged rows of $\boldsymbol{X}$ can be modelled as approximately normal, due to CLT. This enables using the methods devised for normally distributed features. For the details of this approach, see Appendix B.1.

Secondly, if the features are normally distributed but not centred, we need to include the intercept parameter, which corresponds to appending $\boldsymbol{x}_i$ with a one from the left, and `MCMC-normalX` does not directly apply. In that case, we can modify the hierarchical model that accommodates the non-centralised features and the intercept parameter, and we still benefit from the sampling techniques involved in `MCMC-normalX` in Algorithm 1. Appendix B.2 contains the details of the modified hierarchical model.

5. Numerical Experiments

We present several numerical evaluations of the proposed methods, `MCMC-normalX`, `MCMC-fixedS`, and `Bayes-fixedS-fast`, with simulated and real data. We compare our algorithms with two methods: `adaSSP` of Wang (2018) and the MCMC method of Bernstein & Sheldon (2019) for differentially private linear regression, which we will call `MCMC-B&S`. Note that `adaSSP` and `MCMC-B&S` were originally proposed for the non-distributed setting, that is, $J = 1$. For a comprehensive comparison, we implemented their extensions for $J \geq 1$. The details of those extensions are provided in Appendix C. In particular, we carefully generalised the model in Bernstein & Sheldon (2019) for $J \geq 1$ (and for (ϵ, δ)-DP) in a similar fashion as we did to our model in Section 3.2. `MCMC-B&S` is the adaptation of Bernstein & Sheldon (2019, Algorithm 1) for this generalised model. Both for simulated and real data, we set $\|X\|$ and $\|Y\|$ to the maximum of the norms over the whole dataset (for real data, we perform centring and normalising first.) This procedure is equivalent to scaling the data to an interval and standard in existing work. The link for the code and data to replicate all of the experiments in this section is given at the end of Section 6.

5.1. Experiments with Simulated Data

We considered two different configurations, $(n = 10^5, d = 2)$ and $(n = 10^5, d = 5)$, for the problem size. For each (n, d) we simulated the data as follows. We generated $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$, $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_x)$ where $\boldsymbol{\Sigma}_x \sim \mathcal{IW}(\boldsymbol{\Lambda}, \kappa)$, with $\kappa = d + 1$ and the scale matrix selected randomly as $\boldsymbol{\Lambda} = \boldsymbol{V}^T\boldsymbol{V}$, where $\boldsymbol{V}$ is a $d \times d$ matrix of i.i.d. variables from $\mathcal{N}(0, 1)$. The response variables $\boldsymbol{y}$ are generated with $\sigma_y^2 = 1$. For inference, we used the same $\boldsymbol{\Lambda}$, κ as above and $a = 20$, $b = 0.5$, $\boldsymbol{m} = \boldsymbol{0}_{d\times 1}$, $\boldsymbol{C} = b/(a-1)\boldsymbol{I}_d$.

We evaluated the methods at all combinations of $J \in \{1, 5, 10\}$ and $\epsilon \in \{0.1, 0.2, 0.5, 1, 2, 5, 10\}$. All the MCMC algorithms were run for 10^4 iterations. For each (J, ϵ) pair, we ran each method for 50 times (each with different noisy observations) to obtain average performances.

For performance metrics, we looked at the mean squared errors (MSE) of (i) the estimates $\hat{\boldsymbol{\theta}}$ and (ii) the predictions $\hat{y}(\boldsymbol{x}_{\text{test}})$ generated by the methods. For the Bayesian methods, $\hat{\boldsymbol{\theta}}$ is taken as the posterior mean, which can be numerically estimated for the MCMC algorithms. For prediction performance, we calculated $\mathbb{E}[\hat{y}(\boldsymbol{x}_{\text{test}}) - y_{\text{test}}]^2$. For the Bayesian methods, $\hat{y}(\boldsymbol{x}_{\text{test}})$ is the posterior predictive expectation of y_{test} at $\boldsymbol{x}_{\text{test}}$. For `adaSSP`, we simply take $\hat{y}(\boldsymbol{x}_{\text{test}}) = \boldsymbol{x}_{\text{test}}^T\hat{\boldsymbol{\theta}}$.

The results are summarised in Figure 2. We observe that `MCMC-fixedS` and `Bayes-fixedS-fast` outperform `adaSSP` and `MCMC-B&S` in almost all cases both in terms of estimation and prediction. Comparing the full-scale algorithms `MCMC-normalX` and `MCMC-B&S` (that involve updates of $\boldsymbol{S}$), we observe a clear advantage of `MCMC-normalX` at $d = 2$ but `MCMC-B&S` becomes more competitive at $d = 5$. This can be attributed to the fact that `MCMC-B&S` requires the extra statistic $\boldsymbol{y}^T\boldsymbol{y}$, unlike `MCMC-normalX`, which causes `MCMC-B&S` to use more noisy statistics. This difference becomes more significant at small d, where the relative effect of the presence of $\boldsymbol{y}^T\boldsymbol{y}$ on the sensitivity is more significant. Finally, all methods improve as ϵ grows, as expected.

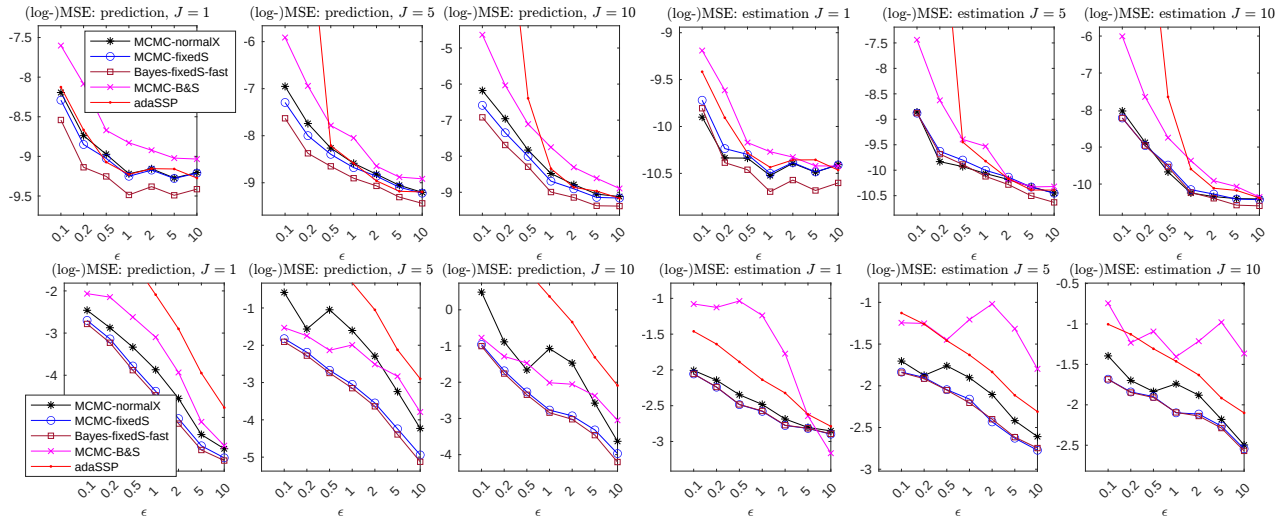

Figure 2. Averaged prediction and estimation performances (over 50 runs). Top row: $n = 10^5, d = 2$, Bottom row: $n = 10^5, d = 5$. For the non-private posterior, for $d = 2$ ($d = 5$, resp.), the obtained MSE values for estimation are 2.92×10^{-5} (1.54×10^{-5}, resp.) and for prediction are 9.60×10^{-5} (4.19×10^{-5}, resp.)

Table 1. Averaged prediction performances (over 50 runs) for the real datasets - $\epsilon = 1$

J	**data sets**	`MCMC-normalX`	`MCMC-fixedS`	`Bayes-fixedS-fast`	`MCMC-B&S`	`adaSSP`
$J = 1$	PowerPlant	0.0129	0.0129	0.0129	**0.0128**	0.0139
	BikeSharing	0.0024	0.0021	0.0021	**0.0020**	0.0107
	AirQuality	0.0060	**0.0057**	**0.0057**	0.0062	0.0066
	3droad	0.0229	0.0229	0.0229	0.0229	0.0229
$J = 5$	PowerPlant	**0.0133**	0.0134	0.0134	0.0136	0.0235
	BikeSharing	0.0174	**0.0045**	**0.0045**	0.0086	0.0382
	AirQuality	0.0142	0.0100	**0.0099**	0.0130	0.0227
	3droad	0.0229	0.0229	0.0229	0.0229	0.0229
$J = 10$	PowerPlant	**0.0142**	0.0143	0.0143	0.0143	0.0351
	BikeSharing	0.0812	**0.0082**	**0.0082**	0.0137	0.0526
	AirQuality	0.0985	**0.0117**	**0.0117**	0.0216	0.0314
	3droad	0.0229	0.0229	0.0229	0.0229	0.0229

We also compare the computation times of the MCMC algorithms `MCMC-normalX`, `MCMC-fixedS`, and `MCMC-B&S`[1]. Figure 3 shows the run times of the algorithms vs. d. The drastic difference in computational loads explained in Section 4.3 is also visible in the figure. While `MCMC-B&S` may be improved in terms of accuracy as d increases, the $\mathcal{O}(d^6)$ dramatically slows it down.

In addition to the MSE, we also consider maximum mean discrepancy (MMD) for evaluating the calibration of the learned posteriors following the method in (Gretton et al., 2012). The estimated MMD values in Figure 4 in Appendix D show that all the methods converge to the non-private posterior as ϵ increases.

[1]The algorithms were run in MATLAB 2021b on an Apple M1 chip with 8 cores and 16 GB LPDDR4 memory.

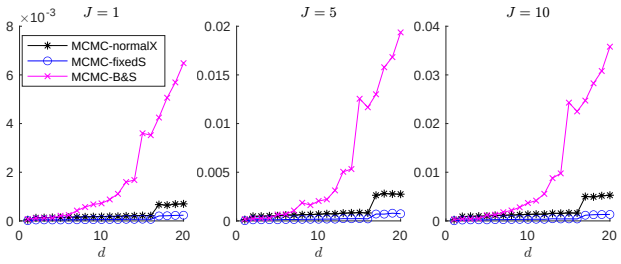

Figure 3. Run times per iteration for MCMC algorithms

5.2. Experiments with Real Data

For the real data case, we use four different data sets from the UCI Machine Learning Repository. We disregard the columns including string data or key values (ID, name, date, etc.), and we consider the most right-hand col-

umn as $\boldsymbol{y}$. The finalised data sets are summarised below.

data set	n	d	hyperlinks
power plant energy	7655	4	view link
bike sharing	13904	14	view link
air quality	7486	12	view link
3d road	347900	3	view link

For prediction, we took 80% of the data for training and the rest for testing. We present the average prediction performances (out of 50 runs) in Table 1 for each dataset and J with $\epsilon = 1$. We observe that the prediction performances of the compared methods are close, while `MCMC-fixed-S` and `Bayes-fixed-S` are arguably the most stable ones. When $J > 1$, (the distributed data setting) those two methods beat `adaSSP` and `MCMC-B&S` more satisfactorily. To show the robustness of these conclusions to variation, we provide 90% confidence intervals for the prediction performances in Table 2 in Appendix D.

6. Conclusion

We propose a novel Bayesian inference framework, with MCMC being its main workhorse, for a differentially private distributed linear regression setting where the data is partitioned among the data holders and shared using summary statistics. We provide several Bayesian inference algorithms suited to the developed hierarchical model for linear regression. Those algorithms can be preferred one over the other depending on the computational budget, model specifics, or how much we know about the underlying statistical facts of the data. We exploit the conditional structure between the summary statistics of linear regression (Proposition 3.1), which leads to feasible algorithms with computational advantages over their competitors. The numerical experiments show that the proposed methods are competitive with their state-of-the-art alternatives in terms of accuracy. The extensions mentioned in Section 4.4 indicate potential future directions.

The experiments demonstrate that `MCMCM-fixedS` and `Bayes-fixedS-fast` are competitive even under normality. Hence, we can suggest users try those fast and effective versions on their first attempts. However, `MCMC-normalX` can potentially provide more insight since it infers $P_{\boldsymbol{x}}$ as well. There is also room for improvement of `MCMC-normalX`. We chose the most common MH moves to update σ_y^2 and $\boldsymbol{S}_j$'s, without paying much attention to their efficiencies. Especially for large d, more advanced techniques such as Hamiltonian Monte Carlo or pseudo-marginal MCMC may be employed to facilitate the mixing of the algorithm.

Code for the experiments: Link to the code and the data for the experiments: `https://github.com/sinanyildirim/Bayesian_DP_dist_LR.git`.

Acknowledgements

The study was funded by the Scientific and Technological Research Council of Turkey (TÜBİTAK) ARDEB Grant No 120E534.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978318. URL http://doi.acm.org/10.1145/2976749.2978318.

Alabi, D., McMillan, A., Sarathy, J., Smith, A., and Vadhan, S. Differentially private simple linear regression. *Proceedings on Privacy Enhancing Technologies*, 2022: 184–204, 04 2022. doi: 10.2478/popets-2022-0041.

Alparslan, B. and Yıldırım, S. Statistic selection and MCMC for differentially private Bayesian estimation. *Statistics and Computing*, 32(5):66, 2022. doi: 10.1007/s11222-022-10129-8. URL https://doi.org/10.1007/s11222-022-10129-8.

Andrieu, C. and Thoms, J. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, 2008. doi: 10.1007/s11222-008-9110-y. URL https://doi.org/10.1007/s11222-008-9110-y.

Balle, B. and Wang, Y.-X. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 394–403. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/balle18a.html.

Bassily, R., Smith, A., and Thakurta, A. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 464–473, Oct 2014. doi: 10.1109/FOCS.2014.56.

Bernstein, G. and Sheldon, D. R. Differentially private bayesian linear regression. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f90f2aca5c640289d0a29417bcb63a37-Paper.pdf.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*, pp. 635–658, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-662-53640-7. doi: 10.1007/978-3-662-53641-4_24. URL https://doi.org/10.1007/978-3-662-53641-4_24.

Cai, T. T., Wang, Y., and Zhang, L. The cost of privacy: Optimal rates of convergence for parameter estimation with differential privacy. *The Annals of Statistics*, 49(5): 2825 – 2850, 2021. doi: 10.1214/21-AOS2058. URL https://doi.org/10.1214/21-AOS2058.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, jul 2011. ISSN 1532-4435.

Dankar, F. K. and El Emam, K. Practicing differential privacy in health care: A review. *Trans. Data Priv.*, 6(1): 35–67, 2013.

Dimitrakakis, C., Nelson, B., Zhang, Z., Mitrokotsa, A., and Rubinstein, B. I. Differential privacy for bayesian inference through posterior sampling. *Journal of machine learning research*, 18(11):1–39, 2017.

Dong, J., Roth, A., and Su, W. J. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B*, 84(1):3–37, February 2022. doi: 10.1111/rssb.12454. URL https://ideas.repec.org/a/bla/jorssb/v84y2022i1p3-37.html.

Dwork, C. Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., and Wegener, I. (eds.), *Automata, Languages and Programming*, pp. 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

Dwork, C. Differential privacy: A survey of results. In Agrawal, M., Du, D., Duan, Z., and Li, A. (eds.), *Theory and Applications of Models of Computation*, pp. 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-79228-4.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of Cyrptography*, pp. 265–284. Springer, 2006.

Dwork, C., Naor, M., Pitassi, T., Rothblum, G. N., and Yekhanin, S. Pan-private streaming algorithms. In *ICS*, pp. 66–80, 2010.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014a.

Dwork, C., Talwar, K., Thakurta, A., and Zhang, L. Analyze gauss: Optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pp. 11–20, New York, NY, USA, 2014b. Association for Computing Machinery. ISBN 9781450327107. doi: 10.1145/2591796.2591883. URL https://doi.org/10.1145/2591796.2591883.

Ferrando, C., Wang, S., and Sheldon, D. Parametric bootstrap for differentially private confidence intervals. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 1598–1618. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/ferrando22a.html.

Foulds, J., Geumlek, J., and an Kamalika Chaudhuri, M. W. On the theory and practice of privacy-preserving Bayesian data analysis. Technical report, arxiv:1603.07294, mar 2016.

Gong, R. Exact inference with approximate computation for differentially private data via perturbations. *Journal of Privacy and Confidentiality*, 12(2), 2022. doi: 10.29012/jpc.797. URL https://par.nsf.gov/biblio/10387759.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Heikkilä, M., Lagerspetz, E., Kaski, S., Shimizu, K., Tarkoma, S., and Honkela, A. Differentially private Bayesian learning on distributed data. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/dfce06801e1a85d6d06f1fdd4475dacd-Paper.pdf.

Heikkilä, M. A., Jälkö, J., Dikmen, O., and Honkela, A. Differentially private Markov chain Monte Carlo. In *NeurIPS*, 2019.

Higham, N. J. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its*

Applications, 103:103–118, 1988. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(88)90223-6. URL https://www.sciencedirect.com/science/article/pii/0024379588902236.

Ju, N., Awan, J., Gong, R., and Rao, V. Data augmentation MCMC for bayesian inference from privatized data. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=tTWCQrgjuM.

Kuru, N., Birbil, S. I., Gürbüzbalaban, M., and Yıldırım, S. Differentially private accelerated optimization algorithms. *SIAM Journal on Optimization*, 32(2):795–821, 2022. doi: 10.1137/20M1355847. URL https://doi.org/10.1137/20M1355847.

Wang, Y.-X. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. In *UAI*, 2018.

Wang, Y.-X., Fienberg, S., and Smola, A. Privacy for free: Posterior sampling and stochastic gradient Monte Carlo. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2493–2502, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/wangg15.html.

Williams, O. and Mcsherry, F. Probabilistic inference and differential privacy. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

Wilson, A. G. and Ghahramani, Z. Generalised Wishart processes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pp. 736–744, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

Yıldırım, S. and Ermiş, B. Exact MCMC with differentially private moves. *Statistics and Computing*, 29(5):947–963, 2019. doi: 10.1007/s11222-018-9847-x. URL https://doi.org/10.1007/s11222-018-9847-x.

Zhang, J., Zhang, Z., Xiao, X., Yang, Y., and Winslett, M. Functional mechanism: Regression analysis under differential privacy. *Proc. VLDB Endow.*, 5(11):1364–1375, jul 2012. ISSN 2150-8097. doi: 10.14778/2350229.2350253. URL https://doi.org/10.14778/2350229.2350253.

Zhang, Z., Rubinstein, B., and Dimitrakakis, C. On the differential privacy of Bayesian inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016. doi: 10.1609/aaai.v30i1.10254. URL https://ojs.aaai.org/index.php/AAAI/article/view/10254.

A. Derivations for `MCMC-normalX`

We reserve this section for the derivations for our algorithm `MCMC-normalX`.

A.1. Full Conditional Distribution of $\boldsymbol{\Sigma}_x$

We can write the full conditional distribution for $\boldsymbol{\Sigma}_x$ as

$$\begin{aligned}
p(\boldsymbol{\Sigma}_x|\boldsymbol{S_{1:J}}, \hat{\boldsymbol{S}}_{1:J}, \hat{\boldsymbol{z}}_{\boldsymbol{1:J}}) &\propto p(\boldsymbol{\Sigma}_x)\prod_{j=1}^{J} p(\boldsymbol{S}_j|\boldsymbol{\Sigma}_x) \\
&= \frac{|\boldsymbol{\Lambda}|^{dk/2}}{2^{dk/2}\boldsymbol{\Gamma}_d(\frac{k}{2})}|\boldsymbol{\Sigma}_x|^{-(d+\boldsymbol{\kappa}+1)/2}e^{-\frac{1}{2}\text{tr}(\boldsymbol{\Lambda}\boldsymbol{\Sigma}_x^{-1})}\prod_{j=1}^{J}\frac{|\boldsymbol{S}_j|^{(n_j-d-1)/2}e^{-\frac{1}{2}\text{tr}(\boldsymbol{\Sigma}_x^{-1}\boldsymbol{S}_j)}}{2^{n_jd/2}|\boldsymbol{\Sigma}_x|^{n_j/2}\boldsymbol{\Gamma}_d(n_j/2)} \\
&\propto |\boldsymbol{\Sigma}_x|^{-\frac{n}{2}-\frac{(d+\boldsymbol{\kappa}+1)}{2}}e^{-\frac{1}{2}(\sum\text{tr}(\boldsymbol{\Sigma}_x^{-1}\boldsymbol{S}_j)+\text{tr}(\boldsymbol{\Lambda}\boldsymbol{\Sigma}_x^{-1}))} \\
&\propto |\boldsymbol{\Sigma}_x|^{-\frac{(d+\boldsymbol{\kappa}+n+1)}{2}}e^{-\frac{1}{2}\text{tr}((\sum\boldsymbol{S_j}+\boldsymbol{\Lambda})\boldsymbol{\Sigma}_x^{-1})}.
\end{aligned}$$

Therefore, we have

$$\boldsymbol{\Sigma}_x|\boldsymbol{S}_{1:J}, \hat{\boldsymbol{S}}_{\boldsymbol{1:J}}, \hat{\boldsymbol{z}}_{1:J} \sim \mathcal{IW}\left(\boldsymbol{\Lambda}+\sum_{j=1}^{J}\boldsymbol{S}_j, \boldsymbol{\kappa}+n\right).$$

A.2. Full Conditional Distribution of $\boldsymbol{\theta}$

The full conditional distribution for θ can be written as

$$p(\boldsymbol{\theta}|\boldsymbol{S}_{1:J}, \boldsymbol{\sigma}_y^2, \hat{\boldsymbol{z}}_{1:J}) \propto \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{m}, \boldsymbol{C})p(\hat{\boldsymbol{z}}_{1:J}|\boldsymbol{S}_{1:J}, \boldsymbol{\theta}, \boldsymbol{\Sigma}_y^2).$$

For the second factor, we have

$$\begin{aligned}
p(\hat{\boldsymbol{z}}_{\boldsymbol{1:J}}|\boldsymbol{S}_{1:J}, \boldsymbol{\theta}, \boldsymbol{\sigma}_y^2) &\propto \prod_{i=1}^{J} p(\hat{\boldsymbol{z}}_j|\boldsymbol{S}_j, \boldsymbol{\theta}, \sigma_y^2) = \prod_{i=1}^{J} p(\hat{\boldsymbol{z}}_j; \boldsymbol{S}_j\boldsymbol{\theta}, \sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d) \\
&\propto \prod_{i=1}^{J}\exp\left\{-\frac{1}{2}(\hat{\boldsymbol{z}}_j - \boldsymbol{S}_j\boldsymbol{\theta})^T(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}(\hat{\boldsymbol{z}}_j - \boldsymbol{S}_j\boldsymbol{\theta})\right\} \\
&\propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\theta}^T\left(\sum_j \boldsymbol{S}_j(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\boldsymbol{S}_j\right)\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\left(\sum_j \boldsymbol{S}_j(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\right)\hat{\boldsymbol{z}}_j\right]\right\}.
\end{aligned}$$

Reorganising the terms, we end up with

$$p(\boldsymbol{\theta}|\boldsymbol{S}_{1:J}, \boldsymbol{\sigma}_y^2, \hat{\boldsymbol{z}}_{1:J}) \propto \exp\left\{-\frac{1}{2}\left[\boldsymbol{\theta}^T\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\theta} - 2\boldsymbol{\theta}^T\boldsymbol{\Sigma}_p^{-1}\boldsymbol{m}_p\right]\right\},$$

where $\boldsymbol{\Sigma}_p = [\sum_j \boldsymbol{S}_j(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1}\boldsymbol{S}_j + \boldsymbol{C}^{-1}]^{-1}$ and $m_p = \boldsymbol{\Sigma}_p[\sum_j \boldsymbol{S}_j(\sigma_y^2\boldsymbol{S}_j + \sigma_z^2\boldsymbol{I}_d)^{-1})\hat{\boldsymbol{z}}_j + \boldsymbol{C}^{-1}\boldsymbol{m}]$. Therefore, $\boldsymbol{\theta}|\boldsymbol{S}_{1:J}, \boldsymbol{\sigma}_y^2, \hat{\boldsymbol{z}}_{1:J} \sim \mathcal{N}(\boldsymbol{\theta}; \boldsymbol{m}_p, \boldsymbol{\Sigma}_p)$.

A.3. Acceptance Ratio for the MH Update of $\boldsymbol{S}_j$

We drop the index j from $\boldsymbol{S}_j$ for simplicity. When $\boldsymbol{S}' \sim \mathcal{W}(\boldsymbol{S}/\alpha, \alpha)$, the proposal density is

$$q(\boldsymbol{S}'|\boldsymbol{S}) = \frac{|\boldsymbol{S}'|^{(\alpha-d-1)/2}e^{-\text{tr}[\alpha\boldsymbol{S}^{-1}\boldsymbol{S}']/2}}{|\boldsymbol{S}/\alpha|^{\alpha/2}2^{\alpha d/2}\Gamma_d(\frac{\alpha}{2})} = \frac{|\boldsymbol{S}'|^{(\alpha-d-1)/2}e^{-\text{tr}[\alpha\boldsymbol{S}^{-1}\boldsymbol{S}']/2}}{|\boldsymbol{S}|^{\alpha/2}2^{\alpha d/2}\Gamma_d(\frac{\alpha}{2})}\alpha^{\alpha/2}.$$

Therefore, the acceptance ratio corresponding to this proposal is

$$\min\left\{1, \frac{q(\boldsymbol{S}|\boldsymbol{S}')}{q(\boldsymbol{S}'|\boldsymbol{S})} \frac{\mathcal{W}(\boldsymbol{S}'; n_j\boldsymbol{\Sigma}_x, \kappa)p(\hat{\boldsymbol{S}}|\hat{\boldsymbol{S}}')\mathcal{N}(\hat{\boldsymbol{z}}; \boldsymbol{S}'\boldsymbol{\theta}, \sigma_y^2\boldsymbol{S}\boldsymbol{\theta} + \sigma_z^2\boldsymbol{I}_d)}{\mathcal{W}(\boldsymbol{S}; n_j\boldsymbol{\Sigma}_x, \kappa)p(\hat{\boldsymbol{S}}|\hat{\boldsymbol{S}})\mathcal{N}(\hat{\boldsymbol{z}}; \boldsymbol{S}\boldsymbol{\theta}, \sigma_y^2\boldsymbol{S}\boldsymbol{\theta} + \sigma_z^2\boldsymbol{I}_d)}\right\},$$

where the ratio of proposals becomes

$$\frac{q(\boldsymbol{S}|\boldsymbol{S}')}{q(\boldsymbol{S}'|\boldsymbol{S})} = \frac{|\boldsymbol{S}|^{(\alpha-d-1)/2}|\boldsymbol{S}|^{\alpha/2}e^{-\mathrm{tr}[\alpha\boldsymbol{S}'^{-1}\boldsymbol{S}]/2}}{|\boldsymbol{S}'|^{(\alpha-d-1)/2}|\boldsymbol{S}'|^{\alpha/2}e^{-\mathrm{tr}[\alpha\boldsymbol{S}^{-1}\boldsymbol{S}']/2}} = \left(\frac{|\boldsymbol{S}|}{|\boldsymbol{S}'|}\right)^{\alpha-(d+1)/2} e^{\alpha(\mathrm{tr}[\boldsymbol{S}^{-1}\boldsymbol{S}']-\mathrm{tr}[\boldsymbol{S}'^{-1}\boldsymbol{S}])/2}.$$

A.4. Acceptance Ratio for the MH Update of σ_y^2

To update σ_y^2, we use a random walk proposal $\sigma_y^{2\prime} \sim \mathcal{N}(\sigma_y^2, \sigma_q^2)$. The resulting acceptance ratio is

$$\min\left\{1, \frac{\mathcal{IG}(\sigma_y^{2\prime}; a, b)\prod_{j=1}^J \mathcal{N}(\hat{\boldsymbol{z}}_j; \boldsymbol{S}_j\boldsymbol{\theta}, \sigma_y^{2\prime}\boldsymbol{S}_j\boldsymbol{\theta} + \sigma_z^2\boldsymbol{I}_d)}{\mathcal{IG}(\sigma_y^2; a, b)\prod_{j=1}^J \mathcal{N}(\hat{\boldsymbol{z}}_j; \boldsymbol{S}_j\boldsymbol{\theta}, \sigma_y^2\boldsymbol{S}_j\boldsymbol{\theta} + \sigma_z^2\boldsymbol{I}_d)}\right\}.$$

B. Other Variants

This appendix is reserved for the details of the other variants mentioned in Section 4.4.

B.1. Approximating Normality by Averaging

Assume $J = 1$ for simplicity. When the feature vectors $\boldsymbol{x}_i$, $i = 1, \ldots, n$ are not normal, another approach that we consider is based on modifying the data to make the rows of the feature matrix averages of multiple original features, and therefore, approximately normal, by the CLT. Specifically, let $k > 1$ be an integer that divides n so that $m = n/k$ is also an integer. Consider the $m \times n$ matrix

$$\boldsymbol{A} = \frac{1}{\sqrt{k}}\begin{bmatrix} \mathbf{1}_{1\times k} & \mathbf{0}_{1\times k} & \ldots & \mathbf{0}_{1\times k} \\ \mathbf{0}_{1\times k} & \mathbf{1}_{1\times k} & \ldots & \mathbf{0}_{1\times k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1\times k} & \mathbf{0}_{1\times k} & \ldots & \mathbf{1}_{1\times k} \end{bmatrix}_{m\times n},$$

where k is some number that divides n so that $m = n/k$ is an integer. Then, the matrix $\boldsymbol{X}_{\text{av}} = \boldsymbol{A}\boldsymbol{X}$ corresponds to constructing a shorter $m \times d$ matrix whose i'th column is the average of the rows $(i-1)k+1, \ldots, ik$ of $\boldsymbol{X}$ (scaled by $1/\sqrt{k}$). When k is large enough, we can make normality assumptions for the rows of $\boldsymbol{X}_{\text{av}}$. Further, consider

$$\boldsymbol{y}_{\text{av}} := \boldsymbol{A}\boldsymbol{y} = \boldsymbol{X}_{\text{av}}\boldsymbol{\theta} + \boldsymbol{A}\boldsymbol{e},$$

whose mean is $\boldsymbol{X}_{\text{av}}\boldsymbol{\theta}$ and covariance $\boldsymbol{A}\boldsymbol{A}^T\sigma_y^2$. But we have $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}_m$, so the covariance is $\sigma_y^2\boldsymbol{I}_m$. Therefore, the same hierarchical model in Figure 1 can be used for $\boldsymbol{X}_{\text{av}}$, $\boldsymbol{y}_{\text{av}}$ with their respective summary statistics

$$\boldsymbol{z}_{\text{av}} = (\boldsymbol{X}_{\text{av}})^T\boldsymbol{y}_{\text{av}}, \quad \boldsymbol{S}_{\text{av}} = (\boldsymbol{X}_{\text{av}})^T\boldsymbol{X}_{\text{av}},$$

as well as the noisy versions of those summary statistics to provide a given level of privacy. Note that $\boldsymbol{S}_{\text{av}}$ and $\boldsymbol{z}_{\text{av}}$ have the same sensitivities as $\boldsymbol{S}$ and $\boldsymbol{z}$, hence the same noise variances are needed for privacy. However, $\boldsymbol{S}_{\text{av}}$ and $\boldsymbol{z}_{\text{av}}$ bear less information about $\boldsymbol{\theta}$ than $\boldsymbol{S}$ and $\boldsymbol{z}$ due to averaging.

B.2. Including the Intercept

Again, assume $J = 1$ for simplicity. If we include the intercept parameter, which corresponds to appending $\boldsymbol{x}_i$ with a 1 from the left, the design matrix will be changed from $\boldsymbol{S}$ to $\boldsymbol{S}_0 = \begin{bmatrix} n & n\bar{\boldsymbol{x}}^T \\ n\bar{\boldsymbol{x}} & \boldsymbol{S} \end{bmatrix}$, where $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$. Also, note that $\boldsymbol{S} = (n-1)\widehat{\boldsymbol{\Sigma}}_x + n\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T$ where $\widehat{\boldsymbol{\Sigma}}_x$ is the sample covariance of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Under the normality assumption for $\boldsymbol{x}_i$'s, we have $\bar{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma}_x/n)$ and $\widehat{\boldsymbol{\Sigma}}_x \sim \mathcal{W}(n-1, \boldsymbol{\Sigma}_x)$ are independent and have known distributions. Therefore, we can

write a model that includes the additional variables $(\boldsymbol{b} = \bar{\boldsymbol{x}}, \widehat{\boldsymbol{\Sigma}}_x, \boldsymbol{S}_0)$ such that $\boldsymbol{b}$ and $\widehat{\boldsymbol{\Sigma}}_x$ are independent and have known distributions and

$$\boldsymbol{S}_0 = \begin{bmatrix} n & n\boldsymbol{b}^T \\ n\boldsymbol{b} & (n-1)\widehat{\boldsymbol{\Sigma}} + n\boldsymbol{b}\boldsymbol{b}^T \end{bmatrix}$$

replaces $\boldsymbol{S}$ in the standard model. More specifically, we have the following hierarchical model:

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{C}), \quad \boldsymbol{\Sigma}_x \sim \mathcal{IW}(\boldsymbol{\Lambda}, \kappa), \quad \widehat{\boldsymbol{\Sigma}}_x | \boldsymbol{\Sigma}_x \sim \mathcal{W}(n-1, \boldsymbol{\Sigma}_x), \quad \boldsymbol{b} | \boldsymbol{\Sigma}_x \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}_x/n),$$
$$\boldsymbol{z} | \boldsymbol{\theta}, \sigma_y^2, \widehat{\boldsymbol{\Sigma}}_x, \boldsymbol{b} \sim \mathcal{N}(\boldsymbol{S}_0\boldsymbol{\theta}, \boldsymbol{S}_0\sigma_y^2), \quad \hat{\boldsymbol{S}} | \widehat{\boldsymbol{\Sigma}}_x, \boldsymbol{b} = \mathcal{N}(\boldsymbol{S}_0, \sigma_{s,\epsilon}^2 \boldsymbol{I}_{d+1}), \quad \hat{\boldsymbol{z}} | \boldsymbol{z} = \mathcal{N}(\boldsymbol{z}, \sigma_z^2 \boldsymbol{I}_{d+1}).$$

C. Compared Methods

Here, we provide the details of the methods which we compare with the proposed methods in this paper. Those methods are originally proposed for $J = 1$. However, for comparison, we implemented their natural extensions to the general (distributed) case $J \geq 1$. The implementations of those methods can be found in the code package provided for this paper.

C.1. MCMC of Bernstein & Sheldon (2019) Adapted to the Distributed Setting

In Bernstein & Sheldon (2019), $J = 1$ is considered only and the vector $\boldsymbol{ss} = [\text{vec}(\boldsymbol{S}), \boldsymbol{z} = \boldsymbol{X}^T\boldsymbol{y}, u = \boldsymbol{y}^T\boldsymbol{y}]$ is perturbed with privacy-preserving noise to generate the observations of the model. For $J \geq 1$, we consider the following natural extension for generating perturbed observations $\widehat{\boldsymbol{ss}} = [\text{vec}(\hat{\boldsymbol{S}}_j), \hat{\boldsymbol{z}}_j, \hat{u}_j]$, where

$$\hat{\boldsymbol{S}}_j = \boldsymbol{S}_j + \sigma_{dp}\boldsymbol{M}_j, \quad \hat{\boldsymbol{z}}_j = \boldsymbol{z}_j + \boldsymbol{v}_j, \quad \boldsymbol{v}_j \sim \mathcal{N}(\boldsymbol{0}, \sigma_{dp}^2\boldsymbol{I}_d), \quad \hat{u}_j = u_j + w_j, \quad w_j \sim \mathcal{N}(0, \sigma_{dp}^2), \tag{13}$$

where $\sigma_{dp} = \sigma(\epsilon, \delta)\Delta_{\boldsymbol{ss}}$ with $\Delta_{\boldsymbol{ss}} = \sqrt{\|X\|^4 + \|X\|^2\|Y\|^2 + \|Y\|^4}$.

For completeness, we provide the further specifics of the model: We take $(\boldsymbol{\theta}, \sigma_y^2) \sim \mathcal{NIG}(a_0, b_0, \boldsymbol{m}, \boldsymbol{C})$ and $P_{\boldsymbol{x}} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_x)$, where $\boldsymbol{\Sigma}_x \sim \mathcal{IW}(\boldsymbol{\Lambda}, \kappa)$.

During the comparisons, we set the $a_0, b_0, \boldsymbol{m}, \boldsymbol{C}, \boldsymbol{\Lambda}, \kappa$ to the same values for both this model and our proposed model that assumes normally distributed features, *i.e.*, $P_{\boldsymbol{x}} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_x)$. Then, we apply an extension of the method of Bernstein & Sheldon (2019) suited to those observations. One iteration of that algorithm includes the following steps in order:

- Calculate the $D \times 1$ mean vector and $D \times D$ covariance matrix

  $$\boldsymbol{\mu}_{\text{prior}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{ss}], \quad \boldsymbol{\Sigma}_{\text{prior}}(\boldsymbol{\theta}) = \text{Cov}_{\boldsymbol{\theta}}[\boldsymbol{ss}]$$

  This step requires the fourth moments $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_x)$.

- For $j = 1, \ldots, J$, sample $\boldsymbol{ss}_j \sim \mathcal{N}(\boldsymbol{\mu}_{\text{post}}^{(j)}, \boldsymbol{\Sigma}_{\text{post}}^{(j)})$ with

  $$\boldsymbol{\Sigma}_{\text{post}}^{(j)} = ([n_j\boldsymbol{\Sigma}_{\text{prior}}(\theta)]^{-1} + (1/\sigma_{dp}^2)\boldsymbol{I}_D)^{-1}, \quad \text{and} \quad \boldsymbol{\mu}_{\text{post}}^{(j)} = \boldsymbol{\Sigma}_{\text{post}}^{(j)}(\boldsymbol{\Sigma}_{\text{prior}}(\boldsymbol{\theta})^{-1}\boldsymbol{\mu}_{\text{prior}}(\boldsymbol{\theta}) + \widehat{\boldsymbol{ss}}_j/\sigma_{dp}^2).$$

- Sample $\boldsymbol{\Sigma}_x \sim \mathcal{IW}\left(\sum_{j=1}^J \boldsymbol{S}_j + \boldsymbol{\Lambda}, n + \kappa\right)$.

- Sample $(\boldsymbol{\theta}, \sigma^2) \sim \mathcal{NIG}(a_n, b_n, \boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n)$ by sampling $\sigma^2 \sim \mathcal{IG}(a_n, b_n)$, followed by sampling $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_n, \sigma_y^2\boldsymbol{\Lambda}_n^{-1})$ with

  $$\boldsymbol{\Lambda}_n = \sum_{j=1}^J \boldsymbol{S}_j + \boldsymbol{\Lambda}_0, \quad \boldsymbol{\mu}_n = \boldsymbol{\Lambda}_n\left(\sum_{j=1}^J \boldsymbol{z}_j + \boldsymbol{\Lambda}_0\mu_0\right), \quad a_n = a_0 + n/2, \quad b_n = 0.5u + \boldsymbol{\mu}_0^T\boldsymbol{\Lambda}_0\boldsymbol{\mu}_0 - \boldsymbol{\mu}_n^T\boldsymbol{\Lambda}_n\boldsymbol{\mu}_n.$$

C.2. A Variant of `adaSSP` for the Distributed Setting

The `adaSSP` algorithm of (Wang, 2018) is originally designed for a single data holder. In `adaSSP`, a differentially private estimate of $\boldsymbol{\theta}$ is released as

$$\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{S}} + \lambda\boldsymbol{I}_d)^{-1}\hat{\boldsymbol{z}}. \tag{14}$$

Here $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{z}}$ are the privatised versions of $\boldsymbol{S}$ and $\boldsymbol{z}$ as in (2) and (3), except that ϵ and δ must be changed to $2\epsilon/3$ and $2\delta/3$ in those equations to provide ϵ, δ-DP. This is because `adaSSP` uses another parameter, λ, which is also calculated from the sensitive data and a third of the privacy budget is spent for privatising that calculation. With $v \sim \mathcal{N}(0,1)$, λ is specifically calculated as

$$\lambda = \max\left\{0, \sigma\sqrt{d\ln(6/\delta)\ln(2d^2/\rho)} - \tilde{\lambda}_{\min}\right\}$$

with $\sigma = \|X\|^2/(\epsilon/3)$, $\lambda_{\min} = \min(\text{eig}(\boldsymbol{S}))$, and $\tilde{\lambda}_{\min} = \max\left\{\lambda_{\min} + \sqrt{\ln(6/\delta)}\sigma v - \ln(6/\delta)\sigma v, 0\right\}$. We consider an extension of (Wang, 2018) for $J \geq 1$. To perform the extension, we reflect on its tendency to approximate a (regularised) least square solution and consider the following estimate

$$\hat{\boldsymbol{\theta}} = \left(\sum_{j=1}^{J}\hat{\boldsymbol{S}}_j + \boldsymbol{I}_d\sum_{j=1}^{J}\lambda_j\right)^{-1}\left(\sum_{j=1}^{J}\hat{\boldsymbol{z}}_j\right). \tag{15}$$

Here $\hat{\boldsymbol{S}}_j$, $\hat{\boldsymbol{z}}_j$ and λ_j are calculated in data node j separately from the other nodes. The estimation procedure in (15) does not properly account for the Bayesian paradigm but aggregates the shared $\hat{\boldsymbol{S}}_j$'s and $\hat{\boldsymbol{z}}_j$'s to approximate the (regulated) least squares solution. Note that each node has separate λ_j because it depends on δ_j, and so the number of rows for each node.

D. Additional results

Figure 4 shows the MMD estimates for $d = 2$. The (squared) MMD between two distributions can be estimated unbiasedly using i.i.d. samples from those distributions. Non-private posterior and private posteriors of `Bayes-fixedS-fast` are in closed form and can be sampled easily. For the MCMC models, we use every 50th sample of the chain to avoid autocorrelation and thus obtain nearly independent samples.

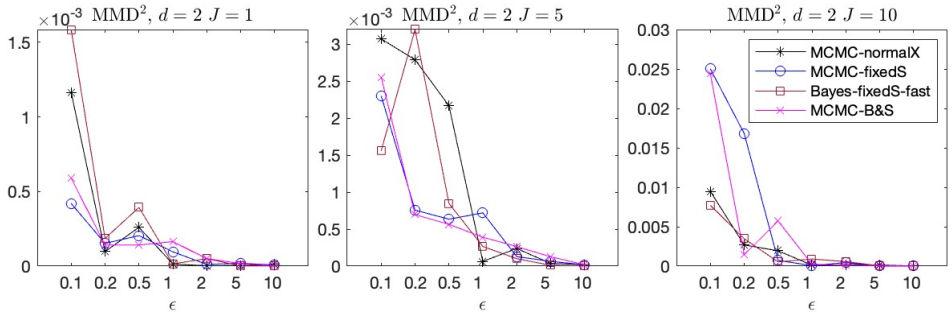

Figure 4. MMD^2 estimates for each J and $d = 2$.

Table 2 shows the confidence intervals for the prediction MSE for the real-data experiments.

Table 2. 90% confidence interval for mean prediction MSE (over 50 runs) for the real-data experiments - $\epsilon = 1$

J	**data sets**	`MCMC-normalX`	`MCMC-fixedS`	`Bayes-fixedS-fast`	`MCMC-B&S`	`adaSSP`
$J = 1$	PowerPlant	[0.0128, 0.0129]	[0.0128, 0.0129]	[0.0128, 0.0129]	[0.0128, 0.0129]	[0.0137, 0.0140]
	BikeSharing	[0.0021, 0.0027]	[0.0018, 0.0024]	[0.0018, 0.0024]	[0.0017, 0.0022]	[0.0106, 0.0108]
	AirQuality	[0.0051, 0.0069]	[0.0048, 0.0066]	[0.0048, 0.0066]	[0.0053, 0.0071]	[0.0065, 0.0067]
	3droad	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]
$J = 5$	PowerPlant	[0.0132, 0.0135]	[0.0132, 0.0136]	[0.0132, 0.0136]	[0.0135, 0.0138]	[0.0234, 0.0236]
	BikeSharing	[0.0137, 0.0210]	[0.0041, 0.0049]	[0.0040, 0.0049]	[0.0076, 0.0095]	[0.0380, 0.0383]
	AirQuality	[0.0109, 0.0175]	[0.0089, 0.010]	[0.0089, 0.0109]	[0.0109, 0.0151]	[0.0226, 0.0229]
	3droad	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]
$J = 10$	PowerPlant	[0.0139, 0.0145]	[0.0140, 0.0146]	[0.0140, 0.0146]	[0.0141, 0.0146]	[0.0349, 0.0353]
	BikeSharing	[0.0671, 0.0954]	[0.0072, 0.0092]	[0.0072, 0.0092]	[0.0116, 0.0158]	[0.0524, 0.0527]
	AirQuality	[0.0733, 0.1236]	[0.0099, 0.0135]	[0.0099, 0.0135]	[0.0175, 0.0257]	[0.0313, 0.0315]
	3droad	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]	[0.0229, 0.0229]