# A FAST ASYNCHRONOUS MCMC SAMPLER FOR SPARSE BAYESIAN INFERENCE

YVES ATCHADÉ AND LIWEI WANG

(July 2023; First version May 2021)

Abstract. We propose a very fast approximate Markov Chain Monte Carlo (MCMC) sampling framework that is applicable to a large class of sparse Bayesian inference problems. The computational cost per iteration in several regression models is of order $O(n(s+J))$, where $n$ is the sample size, $s$ the underlying sparsity of the model, and $J$ is the size of a randomly selected subset of regressors. This cost can be further reduced by data sub-sampling when stochastic gradient Langevin dynamics are employed. The algorithm is an extension of the asynchronous Gibbs sampler of Johnson et al. (2013), but can be viewed from a statistical perspective as a form of Bayesian iterated sure independent screening (Fan et al. (2009)). We show that in high-dimensional linear regression problems, the Markov chain generated by the proposed algorithm admits an invariant distribution that recovers correctly the main signal with high probability under some statistical assumptions. Furthermore we show that its mixing time is at most linear in the number of regressors. We illustrate the algorithm with several models.

## 1. Introduction

There is a rich and extensive literature on high-dimensional sparse Bayesian inference built mainly around shrinkage priors and spike and slab priors (see e.g. Mitchell and Beauchamp (1988); George and McCulloch (1997); Castillo et al. (2015); Atchade (2017); Carvalho et al. (2010); Piironen and Vehtari (2017); Biswas et al. (2021) and the references therein). However the computational cost per iteration for sampling from the resulting posterior distributions grows at least as $O(np^2)$ (Bhattacharya

2010 *Mathematics Subject Classification.* 62F15, 62Jxx.

*Key words and phrases.* Sparse Bayesian inference, Asynchronous MCMC sampling, MCMC mixing, Bayesian deep learning.

This work is partially supported by the NSF grants DMS 1513040 and DMS 2210664. The authors have no conflicts of interest to declare.

Y. Atchadé: Boston University, 111 Cummington Mall, Boston 02215 MA, United States. *E-mail address:* yvesa@umich.edu.

L. Wang: Boston University, 111 Cummington Mall, Boston 02215 MA, United States. *E-mail address:* wlwfoo@bu.edu.

et al. (2016)) in Gaussian linear regression models with $n$ data samples and $p$ regressors ($n \leq p$), and becomes quickly prohibitive, particularly in non-Gaussian models. Indeed, computing high-dimensional integrals remains the main challenge in the practical implementation of Bayesian inference. The problem has grown much worse over the last decade or so with the rise of deep neural networks and other highly over-parameterized models (Bhadra et al. (2020)).

As a step forward, we propose herein a very fast but approximate MCMC scheme for sparse Bayesian models with spike and slab priors. The algorithm builds on a version of the spike and slab prior developed in Atchade and Bhattacharyya (2019). Suppose that we have a dataset $\mathcal{D}$ with a postulated statistical model $\{f_\theta,\ \theta \in \mathbb{R}^p\}$. The proposed prior for $\theta$ introduces a variable $\delta \in \Delta \stackrel{\text{def}}{=} \{0,1\}^p$ (called sparsity structure), with prior distribution $\{\pi(\delta),\ \delta \in \Delta\}$ of the form

$$\pi(\delta) \propto p^{-\mathsf{u}\|\delta\|_0}, \tag{1}$$

for some user-defined parameter $\mathsf{u} > 1$. Given $\delta$, the components of $\theta$ are assumed to be conditionally independent mean-zero Gaussian random variables, with variance $\rho_0^{-1}$ (resp. $\rho_1^{-1}$) if the corresponding component of $\delta$ is 0 (resp. 1), for user-defined parameters $\rho_0 > 0$ and $\rho_1 > 0$. The limiting case $\rho_0^{-1} = 0$ corresponds to the well-known spike and slab prior with a point mass at 0 (Mitchell and Beauchamp (1988)). Setting $\ell(\theta) \stackrel{\text{def}}{=} \log f_\theta(\mathcal{D})$, we consider the posterior distribution of $(\delta, \theta)$ given $\mathcal{D}$ with density on $\Delta \times \mathbb{R}^p$ given by

$$\Pi(\delta, \theta|\mathcal{D}) \propto \left(\frac{1}{p^{\mathsf{u}}}\sqrt{\frac{\rho_1}{\rho_0}}\right)^{\|\delta\|_0} \exp\left(-\frac{\rho_0}{2}\|\theta - \theta_\delta\|_2^2 - \frac{\rho_1}{2}\|\theta_\delta\|_2^2 + \ell(\theta_\delta)\right), \tag{2}$$

where $\theta_\delta \stackrel{\text{def}}{=} \theta \cdot \delta$ is the component-wise product of $\theta, \delta$. A key feature of this sparse Bayesian modeling is that $\theta$ enters the likelihood through $\theta_\delta$. As a result, the marginal posterior distribution of $(\delta, \theta_\delta)$ under (2) does not depend on $\rho_0$, and in particular is the same as with the corresponding spike and slab prior with point mass at the origin. Hence (2) incurs no loss of information in the estimation of $(\delta, \theta_\delta)$ compared with spike and slab priors with point mass at the origin. We refer the reader to Atchade and Bhattacharyya (2019) for more details on (2) including its posterior contraction properties.

1.1. **Main contributions.** We propose a fast MCMC method to sample approximately from (2) where the computational cost per iteration in generalized linear models is of order $O(n(s + J))$, where $n$ is the sample size, $s$ the underlying sparsity of the model, and $J$ is the update batch size (the number of components of $\delta$ that

are asunchronously updated per iteration). This cost can be further reduced by subsampling when stochastic gradient Langevin dynamics (Welling and Teh (2011)) is employed. Furthermore, we show that for linear regression models and well-selected values of $J$, the mixing time of the algorithm is $O(p)$, and the sampling approximation error due to the asynchronous update is small. We show by simulation that these results continue to hold more generally for posterior distributions with good contraction properties.

The algorithm can be viewed as a form of Bayesian sure independent screening (Fan and Lv (2008); Fan et al. (2009)) in the sense that, as in sure independent screening, the algorithm alternates between a fast component-wise variable screening step where the components of $\delta$ are sampled independently (conditionally on $\theta$), and a sparse model refit step where the parameter $\theta$ is re-estimated. From the MCMC viewpoint, the proposed algorithm is an extension of the asynchronous Gibbs sampler (Smola and Narayanamurthy (2010); Johnson et al. (2013); De Sa et al. (2016)) where several variables are updated asynchronously and in parallel.

We test the algorithm empirically on linear and logistic regression models, and with a deep neural network model (lenet-5 applied to the MNIST-FASHION dataset (Xiao et al. (2017))). The application to logistic regression models show that the algorithm is faster and more accurate than the mean-field variational approximation, and the skinny-Gibbs sampler of (Narisetty et al. (2018)). In deep neural network models the proposed algorithm combined with stochastic gradient Langevin dynamics can be easily implemented by modifying existing stochastic gradient descent implementation.

1.2. **Related work.** Sparse model estimation has been a major theme in statistics over the last two decades (Bühlmann and van de Geer (2011); Hastie et al. (2015); Wainwright (2019)), and several MCMC and approximate MCMC samplers have been proposed to deal with Bayesian implementations. In (Bhattacharya et al. (2016)), the authors proposed an exact algorithm with complexity $O(p^2 \min(n, p))$ per iteration for posterior sampling in Bayesian linear regression with Gaussian scale-mixture priors. Rajaratnam et al. (2019) proposes a two-step Gibbs sampler with blocking in a linear regression scenario, simultaneously drawing the coefficient and residual variance in the same step of the Gibbs sampling cycle. In Narisetty et al. (2018), the author focused on logistic regression and proposed an approximate Gibbs sampler called Skinny Gibbs algorithm, which replaces the high dimensional covariance matrix with a sparse approximation. Rockova and George (2018) developed the spike and slab LASSO, a fast optimization-based method to recover the mode of a posterior distribution of the form (2) but with the Gaussian distribution replaced by a

double-exponential. In Johndrow et al. (2020), the authors proposed an approximate sampler for dealing with linear regression models with horseshoe priors.

One important limitation of these prior computational methods is that the proposed samplers target specific models, mostly linear and logistic regression models. In contrast, the approach proposed here has a much broader applicability, yet remains competitive when applied to those basic models.

Variational approximation is another class of methods widely recommended for large scale Bayesian computation (Blei et al. (2016); Louizos et al. (2017); Ghosh et al. (2019); Tran et al. (2020)). However variational approximation is a general methodology, and the extra step of building a good variational approximation family for a given problem – an important requirement for the consistency of the method (Zhang and Gao (2020)) – is a challenging task. Furthermore, fitting variational approximation families that are not conjugate is generally a costly nonconvex problem. For instance, we observed on logistic regression models that the mean field variational approximation of (2) is computationally more expensive and less accurate than the algorithm proposed in this work.

Another related line of research for dealing with big datasets is a divide-and-conquer strategy developed in works such as (Neiswanger et al. (2014); Xue and Liang (2019); Srivastava and Xu (2021)). However, these approaches are fundamentally different from the one pursued here. Divide-and-conquer strategies deal with the statistical challenge of combining posterior distributions obtained by data splitting, whereas stochastic gradient Langevin dynamics is a specific algorithm to sample from the full posterior distribution using mini-batches.

1.3. **Outline.** The paper is organized as follows. We end the introduction with a compendium of our notations. The asynchronous sampler is developed in Section 2. Some theoretical insights are provided in Section 3, but to keep the focus on the main ideas we placed the proofs in the appendix. The numerical illustrations are collected in Section 4. The paper ends with some concluding remarks in Section 5. MATLAB code for the linear regression, logistic regression and deep neural network examples are available at https://github.com/odrinaryliwei/S-SGLD.

1.4. **Notations.** We introduce here some Markov chain notations that are used below, largely following Meyn and Tweedie (2009). A Markov kernel $P$ on some measurable space $(\mathbb{T}, \mathcal{B})$ acts both on bounded measurable functions $f$ on $\mathbb{T}$ and on $\sigma$-finite measures $\mu$ on $(\mathbb{T}, \mathcal{B})$ via $Pf(\cdot) \stackrel{\text{def}}{=} \int P(\cdot, \mathrm{d}y) f(y)$ and $\mu P(\cdot) \stackrel{\text{def}}{=} \int \mu(\mathrm{d}x) P(x, \cdot)$ respectively. If $W : \mathbb{T} \to [1, +\infty)$ is a function, the $W$-norm of a function $f : \mathbb{T} \to \mathbb{R}$ is defined as $|f|_W \stackrel{\text{def}}{=} \sup_{\mathbb{T}} |f|/W$. When $W = 1$, this is the supremum

norm. If $\mu$ is a signed measure on $(\mathbb{T}, \mathcal{B})$, the total variation norm $\|\mu\|_{\mathrm{tv}}$ is defined as $\|\mu\|_{\mathrm{tv}} \stackrel{\mathrm{def}}{=} \frac{1}{2}\sup_{\{f,|f|_1\leq 1\}} |\mu(f)|$, and the $W$-norm of $\mu$ is defined as $\|\mu\|_W \stackrel{\mathrm{def}}{=} \frac{1}{2}\sup_{\{g,|g|_W\leq 1\}} |\mu(g)|$, where $\mu(f)$ denotes the integral $\int_{\mathbb{T}} f(x)\mu(\mathrm{d}x)$. Given two Markov kernels $P, Q$ on $(\mathbb{T}, \mathcal{B})$, their product is the Markov kernel defined as $PQ(x, A) \stackrel{\mathrm{def}}{=} \int P(x, \mathrm{d}y)Q(y, A)$. In particular we define $P^n$, the $n$-th power of $P$, as $P^0(x, A) \stackrel{\mathrm{def}}{=} \delta_x(A)$ and $P^{n+1} = PP^n$, $n \geq 0$, where $\delta_x(\mathrm{d}t)$ stands for the Dirac mass at $x$. Note that $\mu(PQ) = (\mu P)Q$, and $(\mu P)(f) = \mu(Pf)$.

We also collect here our notations on sparse models. Throughout our parameter space is $\mathbb{R}^p$ equipped with its Euclidean norm $\|\cdot\|_2$ and inner product $\langle\cdot,\cdot\rangle$. We also use $\|\cdot\|_0$ which counts the number of non-zero elements, and $\|\cdot\|_\infty$ which returns the largest absolute value. We set $\Delta \stackrel{\mathrm{def}}{=} \{0,1\}^p$. Elements of $\Delta$ are called sparsity structures. For $\delta, \delta' \in \Delta$, we write $\delta \subseteq \delta'$ if $\delta_j \leq \delta'_j$ for all $1 \leq j \leq p$, and we write $\delta \supseteq \delta'$ if $\delta' \subseteq \delta$. Given $\delta \in \Delta$, and $\theta \in \mathbb{R}^p$, we write $\theta_\delta$ to denote the component-wise product of $\theta$ and $\delta$, and $\delta^c \stackrel{\mathrm{def}}{=} 1 - \delta$. We will also write $[\theta]_\delta = (\theta_j,\ j \in \{1 \leq k \leq p :\ \delta_k = 1\})$ which collect the components of $\theta$ with corresponding components of $\delta$ equal to 1. Conversely, assuming $\|\delta\|_0 > 0$, and for $u \in \mathbb{R}^{\|\delta\|_0}$, we define $(u, 0)_\delta$ as the element of $\mathbb{R}^p$ such that $[(u, 0)_\delta]_\delta = u$.

## 2. The asynchronous sampler

Probability distributions of the form (2) are commonly handled using Metropolis-Hastings within Gibbs (Robert and Casella (2004); Brooks et al. (2011)). As a start we follow the same approach, and derive an asymptotically exact algorithm that alternates between an update of $\delta$ given $\theta$, and an update of $\theta$ given $\delta$. To update $\theta$ given $\delta$, we utilize the fact that given $\delta$ the selected components (denoted $[\theta]_\delta$) and the non-selected components (denoted $[\theta]_{\delta^c}$) of $\theta$ are conditionally independent, and that the components of $[\theta]_{\delta^c}$ are i.i.d. $\mathbf{N}(0, \rho_0^{-1})$. Assuming $\|\delta\|_0 > 0$, it is clear from (2) that the conditional distribution of $[\theta]_\delta$ given $\delta$ has density on $\mathbb{R}^{\|\delta\|_0}$ proportional to

$$u \mapsto \exp\left(-\frac{\rho_1}{2}\|u\|_2^2 + \ell((u, 0)_\delta)\right). \tag{3}$$

We then naturally update $[\theta]_\delta$ using a Markov kernel on $\mathbb{R}^{\|\delta\|}$ with invariant distribution proportional to (3) that we denote $P_\delta$. Any convenient MCMC algorithm can be used here (Random Walk Metropolis, Metropolis adjusted Langevin, Hamiltonian Monte Carlo, data-augmentation schemes, and many others), and one can leverage the sparsity of $\delta$ for a fast computation of $\ell((u, 0)_\delta)$.

We use a Gibbs sampler to update $\delta$ given $\theta$. It comes out from (2) that the conditional distribution of $\delta_j$ given $\theta, \delta_{-j}$ is the Bernoulli distribution $\mathbf{Ber}(q_j(\delta, \theta))$,

with probability of success given by

$$q_j(\delta,\theta) \stackrel{\text{def}}{=} \left(1+\exp\left(\mathsf{a}+\frac{1}{2}(\rho_1-\rho_0)\theta_j^2+\ell(\theta_{\delta^{(j,0)}})-\ell(\theta_{\delta^{(j,1)}})\right)\right)^{-1}, \tag{4}$$

where $\mathsf{a} \stackrel{\text{def}}{=} \mathsf{u}\log(p)+\frac{1}{2}\log(\rho_0/\rho_1)$, and where $\delta^{(j,0)}$ (resp $\delta^{(j,1)}$) is the same as $\delta$ except possibly at component $j$ where $\delta_j^{(j,0)}=0$ (resp. $\delta_j^{(j,1)}=1$). Naturally, $q_j(\delta,\theta)$ does not depend on $\delta_j$. We update $J$ randomly selected components of $\delta$ at each iteration. Together, these two steps form an asymptotically exact MCMC algorithm to sample from (2) that is our ideal sampler.

The computational cost of Algorithm 1 depends on the Markov kernel $P_\delta$ used in (STEP 1). Suppose, as in most regression models, that the cost of computing the log-likelihood of a $\delta$-sparse model is $O(n\|\delta\|_0)$. In that case, (STEP 2) of Algorithm 1 is performed at the cost $O(Jn\|\delta\|_0)$. If we assume in addition that the main computational cost of $P_\delta$ is (one) evaluation of the log-likelihood function, then ignoring the cost of generating univariate Gaussian random variables, we see that the computation cost of the $k$-th iteration of Algorithm 1 is of order $O(nJ\|\delta^{(k)}\|_0)$. This cost increases to $O(nJ\|\delta^{(k)}\|_0+n\|\delta^{(k)}\|_0^2)$ (assuming $\|\delta^{(k)}\|_0\le n$) when $P_\delta$ makes use of the gradient of the log-likelihood, or when $P_\delta$ is an exact draw from the conditional distribution of $\theta|\delta$ – as in the linear regression model.

**Algorithm 1.** [Asymptotically Exact Sampler]

Draw $(\delta^{(0)},\theta^{(0)})$ from some initial distribution, and repeat the following steps for $k=0,\ldots$. Given $(\delta^{(k)},\theta^{(k)})=(\delta,\theta)\in\Delta\times\mathbb{R}^p$:

- **(STEP 1):** For all $j$ such that $\delta_j=0$, draw independently $\bar\theta_j\sim\mathbf{N}(0,\rho_0^{-1})$. Provided that $\|\delta\|_0>0$, draw $[\bar\theta]_\delta\sim P_\delta([\theta]_\delta,\cdot)$, where $P_\delta$ is a Markov kernel on $\mathbb{R}^{\|\delta\|_0}$ with invariant density proportional to (3).
- **(STEP 2):** Set $\bar\delta=\delta$. Randomly and uniformly select a subset $\mathsf{J}\subset\{1,\ldots,p\}$ of size $J$.
  - Update $(\bar\delta_{\mathsf{J}_1},\ldots,\bar\delta_{\mathsf{J}_J})$ sequentially: for $j=1,\ldots,J$, draw $V_j\sim\mathbf{Ber}(q_{\mathsf{J}_j}(\bar\delta,\bar\theta))$, where $q_j$ is as in (4), and set $\bar\delta_{\mathsf{J}_j}=V_j$.
- **(STEP 3):** Set $(\delta^{(k+1)},\theta^{(k+1)})=(\bar\delta,\bar\theta)$.

2.1. **Asynchronous approximation.** Algorithm 1 becomes slow in problems where $n$ is large and there is no efficient way of computing the log-likelihood differences $\ell(\theta_{\delta^{(j,0)}})-\ell(\theta_{\delta^{(j,1)}})$ in (4). We propose to speed up this step of the algorithm by (a) replacing the log-likelihood difference by an approximation, and (b) performing the $J$ updates asynchronously, and in parallel. To provide the motivation behind the

method, consider a linear regression problem where $\ell(\theta) = -\frac{1}{2}\|y - X\theta\|_2^2$, $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n\times p}$, with columns normalized to $\|X_j\|_2 = \sqrt{n}$. In that case the second order Taylor approximation of $\ell$ is exact, and since $\theta_{\delta(j,1)} = \theta_{\delta(j,0)} + \theta_j(0,\ldots,0,1,0,\ldots,0)^\intercal$, we have

$$\ell(\theta_{\delta(j,0)}) - \ell(\theta_{\delta(j,1)}) = -\theta_j \langle X_j, y - X\theta_{\delta(j,0)}\rangle + \frac{\theta_j^2 n}{2}. \tag{5}$$

Now, suppose that $j$ is a relevant variable with true regression coefficient $\theta_{\star j}$, say. Suppose also that $j$ is currently not selected ($\delta_j = 0$). In that case, the corresponding regression parameter $\theta_j$ is of order $1/\sqrt{\rho_0}$, since it is drawn from $\mathbf{N}(0, \rho_0^{-1})$. Therefore, and assuming that the regression errors are sub-Gaussian, it can be shown that with $\rho_0 = n$ (as we advocate below),

$$-\theta_j \langle X_j, y - X\theta_{\delta(j,0)}\rangle \approx -\theta_j\theta_{\star j} n \approx \pm|\theta_{\star j}|\sqrt{n},$$

whereas the second order term $\theta_j^2 n/2$ is $O(n/\rho_0) = O(1)$. Furthermore, notice that for $\theta$ close to $\theta_\star$ and $\delta$ sparse, we can afford to replace $\delta^{((j,0)}$ by some $\delta'$ sparse that satisfies $\delta'_j = 0$, since

$$\begin{aligned} &-\theta_j \langle X_j, y - X\theta_{\delta(j,0)}\rangle \approx -\theta_{\star j}^2 n, \quad \text{and} \\ &\quad |\theta_j|\,|\langle X_j, X(\theta_{\delta'} - \theta_{\delta(j,0)})\rangle - \langle X_j, y - X\theta_{\delta'}\rangle| \lesssim \|\theta_{\delta'} - \theta_{\delta(j,0)}\|_1 \max_{j<k} |\langle X_j, X_k\rangle|, \end{aligned}$$

and the rightmost term in the last display can be made $o(n)$. This discussion informally establishes two things: (a) we can safely replace the log-likelihood difference $\ell(\theta_{\delta(j,0)}) - \ell(\theta_{\delta(j,1)})$ by $-\theta_j \langle X_j, y - X\theta_{\delta(j,0)}\rangle$, and (b) for $\theta$ close to $\theta_\star$, we can safely replace $\delta$ by an approximation or an "old copy" (asynchronous update).

We extend this idea to the general model by using the one-dimensional quadratic approximation:

$$\ell(\theta_{\delta(j,0)}) - \ell(\theta_{\delta(j,1)}) \approx -\theta_j \frac{\partial \ell}{\partial \theta_j}(\theta_{\delta(j,0)}) - \frac{\theta_j^2}{2}\left(\frac{\partial \ell}{\partial \theta_j}(\theta_{\delta(j,0)})\right)^2,$$

where we switch the sign of the quadratic term for increased sensitivity. In the linear regression example discussed above, this corresponds to replacing the right-hand side of (5) by

$$-\theta_j \langle X_j, y - X\theta_{\delta(j,0)}\rangle - \frac{\theta_j^2 n}{2}.$$

To see why this is helpful, recall from the discussion above that if $j$ is a significant variable, then

$$-\theta_j \langle X_j, y - X\theta_{\delta(j,0)}\rangle \approx -\theta_j\theta_{\star j} n,$$

which dominates the quadratic term $\theta_j^2 n/2$ in (5). However, replacing the quadratic term in (5) by $-\theta_j^2 n/2$ increases the probability of selecting $X_j$, but does not hurt

otherwise. This leads to the following approximation of $q_j(\delta,\theta)$ in (4) :

$$\tilde{q}_j(\delta,\theta) \stackrel{\text{def}}{=} \left(1+\exp\left(\mathsf{a}+\frac{1}{2}(\rho_1-\rho_0)\theta_j^2-\theta_j G_j(\theta_\delta)-\frac{\theta_j^2}{2}G_j(\theta_\delta)^2\right)\right)^{-1},$$

$$\text{where} \quad G_j(\cdot) \stackrel{\text{def}}{=} \frac{\partial \ell}{\partial \theta_j}(\cdot). \quad (6)$$

Suppose now that we randomly select a subset $\mathsf{J}$ of $\{1,\ldots,p\}$ as in (STEP 2) of Algorithm 1, and we need to approximate the $J$ terms $q_{\mathsf{J}_j}(\delta,\bar{\theta})$. Let $\vartheta=\vartheta(\mathsf{J},\delta)\in\{0,1\}^p$ defined as follows

$$\vartheta_j=\begin{cases}\delta_j & \text{if} \quad j\notin\mathsf{J}\\ 0 & \text{otherwise}\ .\end{cases} \quad (7)$$

We then approximate $q_j(\delta,\bar{\theta})$ by $\tilde{q}_j(\vartheta,\bar{\theta})$, leading to Algorithm 2. We first note that the model $\vartheta$ corresponds to removing from $\delta$, all the variables that are currently selected to be updated. As a result, all the $J$ probabilities $\tilde{q}_j(\vartheta,\theta)$ are computed from the same gradient $G(\theta_\vartheta)=\nabla\ell(\theta_\vartheta)$, which, crucially, does not depend on $(\delta_j)_{j\in\mathsf{J}}$. Therefore, the variables $(V_j)_{1\le j\le J}$ in (STEP 2) of Algorithm 2 are now conditional independent Bernoulli random variables, and can be sampled in parallel (instead of sequentially as in Algorithm 1).

**Algorithm 2.** [Asynchronous sampler]

Draw $(\delta^{(0)},\theta^{(0)})$ from some initial distribution, and repeat the following steps for $k=0,\ldots$. Given $(\delta^{(k)},\theta^{(k)})=(\delta,\theta)\in\Delta\times\mathbb{R}^p$:

**(STEP 1):** For all $j$ such that $\delta_j=0$, draw independently $\bar{\theta}_j\sim\mathbf{N}(0,\rho_0^{-1})$. Provided that $\|\delta\|_0>0$, draw $[\bar{\theta}]_\delta\sim P_\delta([\theta]_\delta,\cdot)$, where $P_\delta$ is a Markov kernel on $\mathbb{R}^{\|\delta\|_0}$ with invariant density proportional to (3).

**(STEP 2):** Set $\bar{\delta}=\delta$. Randomly and uniformly select a subset $\mathsf{J}\subset\{1,\ldots,p\}$ of size $J$. Form the vector $\vartheta=\vartheta(\mathsf{J},\delta)$ as in (7).

- For $1\le j\le J$, draw independently $V_j\sim\mathbf{Ber}(\tilde{q}_{\mathsf{J}_j}(\vartheta,\bar{\theta}))$, where $\tilde{q}_j$ is as in (6).
- For $1\le j\le J$, set $\bar{\delta}_{\mathsf{J}_j}=V_j$.

**(STEP 3):** Set $(\delta^{(k+1)},\theta^{(k+1)})=(\bar{\delta},\bar{\theta})$.

To evaluate the computational cost, suppose that at the $k$-th iteration, sampling from $P_\delta$ requires one log-likelihood function evaluation at the cost of order $O(n\|\delta^{(k)}\|_0)$. Suppose also, as with many regression models, that $J$ partial derivatives of the log-likelihood (the functions $G_j$) can be computed at the cost of order

$O(n(\|\delta^{(k)}\|_0 + J))$. In that case, ignoring the cost of sampling Gaussian random variables, the overall cost of the $k$-th iteration of Algorithm 2 is $O(n(\|\delta^{(k)}\|_0 + J))$ which can be substantially better than $O(nJ\|\delta^{(k)}\|_0)$ achieved by Algorithm 1.

Algorithm 2 has several interesting connections. From a statistical perspective it can be viewed as a Bayesian analog of the iterated sure independent screening (ISIS) of (Fan and Lv (2008); Fan et al. (2009)). Sure independent screening is a statistical inference algorithm that alternates between a fast component-wise variable screening step (based on marginal correlation thresholding, or marginal maximum likelihood estimate thresholding), and a model refit step. Algorithm 2 has the same structure: (STEP 2) corresponds to the variable screening step – which boils down to residual correlation in the linear regression case – followed by a refit step based on MCMC draws. Unlike SIS that relies on hard-thresholding, the variable screening step of Algorithm 2 uses the prior distribution to control sparsity.

Viewed through the lense of MCMC methods, Algorithm 2 appears as an approximate version of Algorithm 1 where the update of $\delta$ in (STEP 2) is replaced by (an inexact form of) the asynchronous Gibbs sampler aka Hogwild! (Smola and Narayanamurthy (2010); Johnson et al. (2013)), recently analyzed by De Sa et al. (2016); Daskalakis et al. (2018).

2.1.1. *On the update batch-size $J$.* The update batch-size $J$ plays a crucial role in Algorithm 2. In Algorithm 1 the choice of $J$ is not of great consequence: the invariant distribution remains $\Pi(\cdot|\mathcal{D})$ for all $J$, and increasing $J$ improves mixing (that is, reduces the number of iterations to convergence), but increases the computational cost per iteration. However in our proposed Algorithm 2, as $J$ increases, the mixing improves, but the limiting distribution diverges further away from $\Pi(\cdot|\mathcal{D})$, due to the accumulation of errors in the asynchronous sampling. We extensively evaluate the effect of $J$ in Section 4.1.4, and we recommend scaling $J$ as

$$J = \min(n, \alpha p), \quad \text{with} \quad \alpha \in (0, 1].$$

Since good approximation of the posterior is always important, we recommend conservative values of $\alpha$ ($\alpha = 0.1$ or $\alpha = 0.01$). For all the linear and logistic regression simulations where $p$ is in the range $1000 - 5000$, we actually use a fixed value of $J$ that we set conservatively to $J = 100$. For the deep learning example where $p \approx 300,000$ we set $J = 1,000$.

2.2. **Further extension using stochastic gradient Langevin dynamics.** Most statistical problems require a full pass through the dataset to evaluate the likelihood function and its derivatives. Therefore in big data problems the cost of computing the likelihood and its gradient in Algorithm 2 may become a computational bottleneck.

Stochastic gradient Langevin dynamics (SGLD) algorithms have recently emerged as very useful algorithms in such settings (Welling and Teh (2011); Ma et al. (2015)). To be more specific, we suppose here that the log-likelihood function has the form

$$\ell(\theta) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \ell_i(\theta), \qquad \theta \in \mathbb{R}^p.$$

In that case, provided that the log-likelihood functions has a Lipschitz gradient, one can naturally replace the Markov kernel $P_\delta$ in (STEP 1) of Algorithm 2 by SGLD or its variants. The resulting algorithm is presented in Algorithm 3.

**Algorithm 3.** [Sparse Asynchronous SGLD]

Draw $(\delta^{(0)}, \theta^{(0)})$ from some initial distribution, and repeat the following steps for $k = 0, \ldots$. Given $(\delta^{(k)}, \theta^{(k)}) = (\delta, \theta) \in \Delta \times \mathbb{R}^p$:

**(STEP 1):** For all $j$ such that $\delta_j = 0$, draw independently $\bar{\theta}_j \sim \mathbf{N}(0, \rho_0^{-1})$. Provided that $\|\delta\|_0 > 0$, randomly select a data mini-batch $\mathsf{I} \subset \{1, \ldots, n\}$ of size $B$, draw $Z \sim \mathbf{N}(0, I_{\|\delta\|_0})$, and set

$$[\bar{\theta}]_\delta = [\theta]_\delta + \gamma \left( -\rho_1 [\theta]_\delta + \frac{n}{B} \sum_{i \in \mathsf{I}} [\nabla \ell_i(\theta_\delta)]_\delta \right) + \sqrt{2\gamma} Z, \tag{8}$$

where $\gamma > 0$ is the step-size.

**(STEP 2):** Set $\bar{\delta} = \delta$. Randomly and uniformly select a subset $\mathsf{J} \subset \{1, \ldots, p\}$ of size $J$. Form the vector $\vartheta = \vartheta(\mathsf{J}, \delta)$ as in (7).

- For $1 \le j \le J$, draw independently $V_j \sim \mathbf{Ber}(\hat{q}_{\mathsf{J}_j}(\vartheta, \bar{\theta}))$, where $\hat{q}_j$ is as in (9).
- For $1 \le j \le J$, set $\bar{\delta}_{\mathsf{J}_j} = V_j$.

**(STEP 3):** Set $(\delta^{(k+1)}, \theta^{(k+1)}) = (\bar{\delta}, \bar{\theta})$.

Note that one can also approximate the Bernoulli probability $\tilde{q}_j$ in (6) using the selected mini-batch as follows. Given a mini-batch $\mathsf{I} \subset \{1, \ldots, n\}$ of size $B$, $1 \le j \le p$, and $(\delta, \theta) \in \Delta \times \mathbb{R}^p$, we set

$$\hat{q}_j(\delta, \theta) \stackrel{\text{def}}{=} \left( 1 + \exp\left( \mathsf{a} + \frac{1}{2}(\rho_1 - \rho_0)\theta_j^2 - \theta_j \hat{G}_j(\theta_\delta) - \frac{\theta_j^2}{2} \hat{G}_j(\theta_\delta)^2 \right) \right)^{-1},$$

$$\text{where} \quad \hat{G}_j(\cdot) = \frac{n}{B} \frac{\partial}{\partial \theta_j} \left[ \sum_{i \in \mathsf{I}} \ell_i \right](\cdot). \tag{9}$$

We systematically make that choice in Algorithm 3. Using the same cost computing assumption as above, and ignoring the cost of generating univariate Gaussian random

variables, we see that in the linear regression case, the computation cost of the $k$-th iteration of Algorithm 3 is now of order $O(B(\|\delta^{(k)}\|_0 + J))$.

## 3. Approximate correctness of Algorithm 2

In this section the dataset $\mathcal{D}$ is assumed fixed, and we shall omit the dependence of the Markov kernels on $\mathcal{D}$. Let $K$ (resp. $\tilde{K}$) be the transition kernel of the Markov chain generated by Algorithm 1 (resp. Algorithm 2). We first write the expression of $K$ and $\tilde{K}$ and introduce some useful notations in the process. Given $\theta \in \mathbb{R}^p$, and $j \in \{1, \dots, p\}$, let $Q_{\theta,j}$ be the Markov kernel on $\Delta$ which, given $\delta \in \Delta$, leaves $\delta_i$ unchanged for all $i \neq j$, and update $\delta_j$ using a draw from $\mathbf{Ber}(q_j(\delta, \theta))$:

$$Q_{\theta,j}(\delta, \delta') = q_j(\delta, \theta)^{\delta'_j}(1 - q_j(\delta, \theta))^{1-\delta'_j} \prod_{i \neq j} \mathbf{1}_{\{\delta_i = \delta'_i\}}, \quad \delta, \delta' \in \Delta.$$

Given $\mathsf{J} = \{j_1, \dots, j_J\} \subseteq \{1, \dots, p\}$, we multiply the Markov kernels $Q_{\theta,j_i}$ together to form $Q_{\theta,\mathsf{J}}$:

$$Q_{\theta,\mathsf{J}} \stackrel{\text{def}}{=} Q_{\theta,j_1} \times \cdots \times Q_{\theta,j_J},$$

where the Markov kernel multiplication is as defined in Section 1.4. We define similarly $\tilde{Q}_{\theta,\mathsf{J}}$ as

$$\tilde{Q}_{\theta,\mathsf{J}}(\delta, \delta') \stackrel{\text{def}}{=} \prod_{j \notin \mathsf{J}} \mathbf{1}_{\{\delta'_j = \delta_j\}} \prod_{i=1}^{J} \tilde{q}_{j_i}(\vartheta, \theta)^{\delta'_{j_i}}(1 - \tilde{q}_{j_i}(\vartheta, \theta))^{1-\delta'_{j_i}}, \quad \delta, \delta' \in \Delta,$$

where $\vartheta = \vartheta(\mathsf{J}, \delta)$ is as defined in (7), and depends on $\delta$. The Markov kernel $\tilde{K}$ of algorithm 2 can then be written as

$$\tilde{K}((\delta, \theta); (\mathrm{d}\delta', \mathrm{d}\theta')) = K_\delta(\theta, \mathrm{d}\theta') \sum_{\mathsf{J}:\, |\mathsf{J}| = J} \binom{p}{J}^{-1} \tilde{Q}_{\theta',\mathsf{J}}(\delta, \mathrm{d}\delta'),$$

where $K_\delta$ denotes the transition kernel of (STEP 1), which can be written as

$$K_\delta(\theta, \mathrm{d}\theta') \stackrel{\text{def}}{=} P_\delta([\theta]_\delta, \mathrm{d}[\theta']_\delta) \prod_{j:\, \delta_j = 0} \mathbf{N}(0, \rho_0^{-1})(\mathrm{d}\theta'_j),$$

where $\mathbf{N}(\mu, \sigma^2)(\mathrm{d}x)$ denotes the probability measure of the Gaussian distribution $\mathbf{N}(\mu, \sigma^2)$ on $\mathbb{R}$. The Markov kernel $K$ of Algorithm 1 has the same structure, but with $\tilde{Q}_{\theta,\mathsf{J}}$ replaced by $Q_{\theta,\mathsf{J}}$. The kernels $K$ and $\tilde{K}$ have the same structure, and therefore the same irreducibility and aperiodicity properties. By construction, the invariant distribution of $K$ is $\Pi(\cdot|\mathcal{D})$. In general, it is not possible to deduce the existence of an invariant distribution for $\tilde{K}$ from the shared similarity between $K$ and $\tilde{K}$. However, we will show that if $\tilde{K}$ has an invariant distribution $\tilde{\Pi}$, say, and is geometrically ergodic, then the proximity between $\tilde{\Pi}$ and $\Pi$ is guaranteed by the

proximity of $K$ and $\tilde{K}$. This is a standard Davis-Kahan type result for Markov chains that has also been derived by others in the literature (Pillai and Smith (2015); Rudolf and Schweizer (2018); Johndrow and Mattingly (2018)). What makes our version worth stating here is that it can leverage the posterior contraction of $\Pi$. Our approach is similar to (Pillai and Smith (2015)), but differs substantially in the details. We shall make the following geometric ergodicity assumption on $\tilde{K}$.

**H 1.** *The Markov kernel $\tilde{K}$ possesses a unique invariant distribution $\tilde{\Pi}$, and there exist $\tilde{\lambda} \in (0,1)$, a function $V : \Delta \times \mathbb{R}^p \to [1,\infty)$, a constant $C_0$ such that for all $(\delta,\theta) \in \Delta \times \mathbb{R}^p$, and all $k \geq 0$,*

$$\|\tilde{K}^k((\delta,\theta),\cdot) - \tilde{\Pi}\|_{\mathrm{tv}} \leq C_0 \tilde{\lambda}^k V^{1/2}(\delta,\theta). \tag{10}$$

**Theorem 1.** *Suppose that H1 holds, and $\Pi(V|\mathcal{D}) < \infty$. Suppose also that there exist $\mathsf{B} \subset \Delta \times \mathbb{R}^p$, and a finite constant $C_0$ such that*

$$\sup_{(\delta,\theta)\in\mathsf{B}} V(\delta,\theta) < \infty, \quad \textit{and} \quad \int_{\Delta} \tilde{Q}_{\theta,\mathsf{J}}(\delta, \mathrm{d}\delta') V(\delta',\theta) \leq C_0 V(\delta,\theta), \tag{11}$$

*for all $(\delta,\theta)$, and any subset $\mathsf{J}$, with $|\mathsf{J}| = J$. Then there exists a finite constant $C_1$ such that*

$$\begin{aligned} &\|\tilde{\Pi}(\cdot|\mathcal{D}) - \Pi(\cdot|\mathcal{D})\|_{\mathrm{tv}} \\ &\quad \leq \frac{C_1}{1-\tilde{\lambda}} \left( \Pi(\mathsf{B}^c|\mathcal{D})^{1/2} + \max_{\mathsf{J}:\,|\mathsf{J}|=J} \int_{\mathsf{B}} \Pi(\mathrm{d}\delta, \mathrm{d}\theta) \left\| Q_{\theta,\mathsf{J}}(\delta,\cdot) - \tilde{Q}_{\theta,\mathsf{J}}(\delta,\cdot) \right\|_{\mathrm{tv}} \right), \end{aligned} \tag{12}$$

*where $\tilde{\lambda}$ is as in H1, and $\mathsf{B}^c$ denotes the complement of $\mathsf{B}$ in $\Delta \times \mathbb{R}^p$.*

The proof of Theorem 1 is provided in Appendix Section A. When the set $\mathsf{B}$ is a sparse neighborhood of the true parameter $\theta_\star$, and $J$ small, we expect the total variation distance between $Q_{\theta,\mathsf{J}}$ and $\tilde{Q}_{\theta,\mathsf{J}}$ to be small (see Corollary 4 for a bound in the linear regression case). By posterior contraction the term $\Pi(\mathsf{B}^c|\mathcal{D})$ is also small. Hence, the main implication of the theorem is that for $J$ well-chosen, our proposed methodology works well for sparse modeling problems where the posterior distribution has good posterior contraction properties around the truth. However bounding the terms on the right-hand side of (12) is in general a non-trivial task. Another limitation of Theorem 1 is the reliance on the geometric ergodicity assumption H1, and the appearance of the spectral gap $1-\tilde{\lambda}$ (which depends in general on $n$ and $p$) on the right-hand side of (12). We leave these challenging technical issues for possible future research.

Assumption H1 is usually established by showing that $\tilde{K}$ satisfies a uniform minorization condition, or a geometric drift condition (Meyn and Tweedie (2009)).

Checking these conditions, in turn, boils down to analyzing the kernels $P_\delta$. For instance it can be easily shown that if each $P_\delta$ satisfies a uniform minorization condition, then H1 holds. To see this, suppose that for each $\delta \in \Delta$, with $\|\delta\|_0 > 0$, there exists a probability measure $\nu_\delta$ on $\mathbb{R}^{\|\delta\|_0}$ that is absolutely continuous with respect to the Lebesgue measure on $\mathbb{R}^{\|\delta\|_0}$ with a positive density, and $\epsilon_\delta > 0$ such that

$$P_\delta(u, \mathrm{d}v) \geq \epsilon_\delta \nu_\delta(\mathrm{d}v), \quad \text{for all} \quad u \in \mathbb{R}^{\|\delta\|_0}.$$

Then, we set

$$\bar{\nu}_\delta(\mathrm{d}\theta) \stackrel{\text{def}}{=} \nu_\delta(\mathrm{d}[\theta]_\delta) \prod_{j:\ \delta_j = 0} \mathbf{N}(0, \rho_0^{-1})(\mathrm{d}\theta_j), \quad \text{and} \quad \bar{\nu} \stackrel{\text{def}}{=} \min_\delta \bar{\nu}_\delta,$$

where for two measures $\mu_1, \mu_2$ with density $f_1, f_2$ respectively, $\min(\mu_1, \mu_2)$ is the measure with density $\min(f_1, f_2)$. For $\delta = (0, \ldots, 0)^\mathsf{T}$, we define $\bar{\nu}_\delta$ as $K_\delta(\theta, \mathrm{d}\theta') = \prod_{j=1}^p \mathbf{N}(0, \rho_0^{-1})(\mathrm{d}\theta'_j)$, and hence $\epsilon_\delta = 1$. Note that under the stated assumptions, each $\bar{\nu}_\delta$ has a positive density with respect to the Lebesgue measure on $\mathbb{R}^p$. Therefore, $\bar{\nu}$ is a non-trivial measure. It then follows from its definition that $\tilde{K}$ satisfies the minorization

$$\begin{aligned}
\tilde{K}(\delta, \theta; \bar{A}) &= \sum_{\mathsf{J}:\ |\mathsf{J}| = J} \binom{p}{J}^{-1} \int_{\mathbb{R}^p} K_\delta(\theta, \mathrm{d}\theta') \int_\Delta \tilde{Q}_{\theta', \mathsf{J}}(\delta, \mathrm{d}\delta') \mathbf{1}_{\bar{A}}(\delta', \theta') \\
&\geq \epsilon \sum_{\mathsf{J}:\ |\mathsf{J}| = J} \binom{p}{J}^{-1} \int_{\mathbb{R}^p} \bar{\nu}(\mathrm{d}\theta') \int_\Delta \min_{\delta \in \Delta} Q_{\theta', \mathsf{J}}(\delta, \mathrm{d}\delta') \mathbf{1}_{\bar{A}}(\delta', \theta') \\
&= \epsilon \mu(\bar{A}), \qquad (13)
\end{aligned}$$

where $\epsilon = \min_{\delta \in \Delta} \epsilon_\delta > 0$. Hence $\tilde{K}$ satisfies a uniform minorization, and by Theorem 16.0.2 of Meyn and Tweedie (2009), H1 holds with $V \equiv 1$. This uniform minorization analysis applies in the particular case (as in linear regression) where $P_\delta$ corresponds to taking an exact draw from the conditional distribution of $\theta|\delta$. Assumption H1 can also be shown to hold if the kernels $P_\delta$ satisfy a geometric drift condition as we show next. We refer the reader to Meyn and Tweedie (2009) Section 5.5.2 for the definition of a petite set. The proof of the following result is provided in Appendix Section B.

**Proposition 2.** *Suppose that $\tilde{K}$ is phi-irreducible and aperiodic, and for each $1 \leq j \leq p$ there exists $V_j : \mathbb{R} \to [1, \infty)$, with $\max_j \int V_j(x) e^{-\rho_0^{-1} x^2/2} \mathrm{d}x < \infty$, such that the following holds. For each $\delta \in \Delta$, with $\|\delta\|_0 > 0$, there exist $\lambda_\delta \in [0, 1)$, $b_\delta < \infty$ such that*

$$\tilde{P}_\delta V_\delta(u) \leq \lambda_\delta V_\delta(u) + b_\delta, \quad u \in \mathbb{R}^{\|\delta\|_0},$$

*where* $V_\delta(u) \stackrel{\text{def}}{=} V(\delta,(u,0)_\delta)$*, and* $V(\delta,\theta) \stackrel{\text{def}}{=} \sum_{j=1}^p \delta_j V_j(\theta_j)$*. Furthermore, suppose that the level sets* $\{(\delta,\theta)\in\Delta\times\mathbb{R}^p:\ V(\delta,\theta)\le b\}$ *for* $b\ge 0$ *are petite sets for* $\tilde{K}$*. Then H1 holds with* $V$*.*

3.1. **Approximate correctness for linear regression models.** In this section we take a closer look at Algorithm 2 in the case of linear regression models. Given some random response $Y\in\mathbb{R}^n$, and a nonrandom design matrix $X\in\mathbb{R}^{n\times p}$, we consider in this section a log-likelihood function given by

$$\ell(\theta) = -\frac{1}{2\sigma^2}\|Y-X\theta\|_2^2,\ \ \theta\in\mathbb{R}^p, \tag{14}$$

for a known constant $\sigma^2$. We write $X_j$ to denote the $j$-th column of $X$, and $X_\delta$ to denote the sub-matrix of $X$ comprised of the columns of $X$ for which $\delta_j=1$. Without any loss of generality we assume throughout that

$$\|X_j\|_2 = \sqrt{n}. \tag{15}$$

We also make the following assumption.

**H 2.** *(1) There exists an absolute constant* $c_0<\infty$ *such that*

$$\max_{j\neq k}|\langle X_j,X_k\rangle| \ \le\ \sqrt{c_0 n\log(p)}. \tag{16}$$

*(2) There exist a parameter value* $\theta_\star\in\mathbb{R}^p$ *(with sparsity support denoted* $\delta_\star$*, and* $\ell^0$ *norm* $s_\star \stackrel{\text{def}}{=} \|\delta_\star\|_0$*), such that*

$$\mathbb{E}_\star\left[Y-X\theta_\star\right]=0,\qquad \mathbb{P}_\star\left[|\langle u,Y-X\theta_\star\rangle|>\sigma t\right]\le c_1 e^{-\frac{t^2}{2\|u\|_2^2}}, \tag{17}$$

*for all* $u\in\mathbb{R}^n$*,* $t\in\mathbb{R}$*, and some absolute constant* $c_1$*.*

*(3) As* $n,p\to\infty$*, the ratios* $\|\theta_\star\|_\infty/\log(p)$ *and* $n/p$ *remain bounded from above by some absolute constant* $c_2$*.*

**Remark 3.** Assumption H2-(2) assumes that the regression errors are sub-Gaussian. (2)-(3) are mild assumptions. Assumption H2-(1) is also a standard assumption is sparse signal recovery and assumes a weak correlation between any two distinct columns of $X$ (Candès and Plan (2009)). □

In what follows we write $\mathbb{P}$ and $\mathbb{E}$ to denote the probability measure and expectation operator of the Markov chains defined by the algorithms, and we write $\mathbb{P}_\star$ and $\mathbb{E}_\star$ for the probability measure and expectation operator related to the data generating distribution as assumed in H2.

We set

$$\underline{\theta}_\star \stackrel{\text{def}}{=} \min_{j:\, \delta_{\star j}=1} |\theta_{\star j}|.$$

In the linear regression considered here, the conditional distribution of $\theta$ given $\delta$ has a closed-form Gaussian distribution. We can thus assume that (STEP 1) of Algorithm 2 is performed by taking a draw directly from the conditional distribution of $\theta$ given $\delta$. In this case, $\tilde{K}$ satisfies a uniform minorization as discussed in (13), and assumption H1 holds with $V \equiv 1$. Hence Theorem 5 directly applies and gives the following corollary. The proof is given in Section C of the supplement.

**Corollary 4.** *Consider the linear regression model presented above and assume H2. Suppose that*

$$\rho_1 = 1, \quad \textit{and} \quad \rho_0 = \frac{n}{\sigma^2}.$$

*Let* $\mathsf{B} = \{(\delta,\theta) : \ \delta_\star \subseteq \delta, \ \|\delta\|_0 \le s, \quad \textit{and } \|\theta_\delta - \theta_\star\|_\infty \le c_2\sqrt{\log(p)/n}\}$ *for some constant* $s \ge s_\star$ *such that* $n \ge s^2\log(p)$*, and some constant* $c_2$*. Then there exists a finite constants* $C_1, C_2$ *and* $\tilde{\lambda} \in [0,1)$ *such that with probability at least* $1 - 1/p$

$$\begin{aligned}\|\tilde{\Pi}(\cdot|\mathcal{D}) - \Pi(\cdot|\mathcal{D})\|_{\text{tv}} \le \frac{C_1}{1-\tilde{\lambda}}\Big[\Pi(\mathsf{B}^c|\mathcal{D})^{1/2} + J\Pi(\mathsf{B}^c|\mathcal{D})\Big] \\ + \frac{C_1 J}{1-\tilde{\lambda}}\max\left(e^{-C_2\left(n\underline{\theta}_\star^2 - (1+s_\star)\sqrt{n\log(p)}\right)}, \frac{1}{\sqrt{n}}e^{-(\mathsf{u}-C_2(1+s_\star))\log(p)}\right) \quad (18)\end{aligned}$$

The result captures our main intuition: if $\theta_\star$ is sparse and the posterior contracts towards $\theta_\star$, then the limiting distribution of the asynchronous sampler can remain close to $\Pi(\cdot|\mathcal{D})$ by choosing $\mathsf{u} \ge C_2(1+s_\star)$ and $n$ large enough such that $n\underline{\theta}_\star^2 > (1+s_\star)\sqrt{n\log(p)}$. However, even in the current linear regression setting, the bound in (18) is not fully informative. For instance we still need to establish that $\Pi(\mathsf{B}^c|\mathcal{D})$ is small[1]. Furthermore, bounding the spectral gap of $\tilde{K}$ that appears in the bound (18) remains challenging. To circumvent some of these limitations, in the next result we analyze directly the Markov chain $\{\delta^{(k)}, \ k \ge 0\}$ produced by Algorithm 2. The result has also the advantage of estimating the mixing time of Algorithm 2.

**Theorem 5.** *Consider the linear regression model presented above and assume H2. Suppose that*

$$\rho_1 = 1, \quad \textit{and} \quad \rho_0 = \frac{n}{\sigma^2}.$$

[1]Although the estimation rate in the sup-norm in high-dimensional linear regression is known to be of order $\sqrt{\log(p)/n}$, establishing this for posterior distribution is not trivial, and has not been done to the best of our knowledge.

*Let $\mathbb{P}$ denote the distribution of the Markov chain $\{\delta^{(k)},\ k \geq 0\}$ generated by Algorithm 2, and started from the null model ($\|\delta^{(0)}\|_0 = 0$). We can find constants $C_1, C_2, C_3, C_4$ that depend only on $\sigma^2, \|\theta_\star\|_\infty$, $c_0, c_1$, and $c_2$, such that for all $n, p \geq 2$, if*

$$n \geq C_1 \max\left(\underline{\theta}_\star^{-2}(1+s_\star^3)\log(p),\ (\log(p))^3\right), \quad \textit{and} \quad \mathsf{u} \geq C_2(1+s_\star)^2, \tag{19}$$

*it holds for all $k \geq 1$,*

$$\mathbb{E}_\star\left[\max_{j:\ \delta_{\star j}=1} \left|\mathbb{P}(\delta_j^{(k)}=1) - \Pi(\delta_j = 1|\mathcal{D})\right|\right] \leq \left(1 - \frac{3}{10}\frac{J}{p}\right)^k + \exp\left(-C_3\underline{\theta}_\star\sqrt{n} + C_4 J\sqrt{\log(p)}\right) + \frac{10}{p}. \tag{20}$$

The proof of Theorem 5 is provided in Appendix Section D. The bound in (20) gives a mixing time bound for the Markov chain $\{\delta^{(k)},\ k \geq 0\}$, where convergence to stationarity is measured using a (sliced) total variation distance on the relevant one-dimensional marginal distributions. Importantly, the result shows that Algorithm 2 converges faster for larger values of $J$. Furthermore, letting $k \to \infty$, (20) implies that

$$\mathbb{E}_\star\left[\max_{j:\ \delta_{\star j}=1} \left|\tilde{\Pi}(\delta_j = 1|\mathcal{D}) - \Pi(\delta_j = 1|\mathcal{D})\right|\right] \leq \exp\left(-C_3\underline{\theta}_\star\sqrt{n} + C_4 J\sqrt{\log(p)}\right) + \frac{10}{p}.$$

Hence, when $J = o(\sqrt{n/\log(p)})$, the result shows that in linear regression models the limiting distribution of Algorithm 2 recovers correctly the relevant components of the signal. However our numerical simulations suggest that the condition $J = o(\sqrt{n/\log(p)})$ is too restrictive, and in fact the algorithm continues to behave well for much larger values of $J$ (see Section 4.1.4 for more details).

The first part of (19) imposes some minimum sample size requirement. We noted empirically that the mixing time of Algorithm 2 degrades for small values of the sample size $n$, particularly in logistic regression models. This suggests that (19) – which may not be optimal – represents some genuine information limit of the problem. In limited data settings we recommend combining Algorithm 2 with simulated tempering or related methods for improved mixing. However in the interest of space, we do not pursue these tempering ideas here. The dependence of (19) on $\underline{\theta}_\star$ is the so-called $\beta$-min condition that is commonly needed for correct model selection (see Meinshausen and Yu (2009) for discussion). It is similar to the high signal-to-noise ratio condition of Yang et al. (2016), and has also appeared elsewhere in the analysis of high-dimensional MCMC samplers (Atchadé (2022)).

With the same proof strategy, we believe that Theorem 5 can be extended to statistical models that possess the restricted strong concavity property (Negahban et al. (2012)), under the additional assumption that one can sample exactly from the conditional distribution of $\theta$ given $\delta$. We did not pursue this because of the limited

applicability: the exact sampling assumption is highly unrealistic for non-Gaussian models. Extending Theorem 5 to cases where a Markov kernel is used in (STEP 1) is more challenging, but is likely to still hold if the Markov kernel has a strong drift toward the level sets of the target distribution. We leave this for potential future research.

## 4. Numerical illustration

4.1. **Linear regression.** We investigate several aspects of Algorithm 1 and Algorithm 2 with a simulated linear regression example. Here is the simulation set up. We generate $X \in \mathbb{R}^{n\times p}$ with independent rows drawn from $\mathbf{N}_p(0,\Sigma)$, where $\Sigma_{ij} = \varrho^{|j-i|}$, where $\varrho \in \{0, 0.9\}$. Then we draw $Y \sim \mathbf{N}_n(X\theta_\star, \sigma I_n)$, with $\sigma = 1$, and with a sparse $\theta_\star$ with first 10 components uniformly drawn from $(-3,-2)\cup(2,3)$. We scale the sample size as $n = p/2$ (unless specified otherwise), for values of $p$ that depend on the experiments. For all the results we set

$$\rho_0 = n, \quad \rho_1 = 1, \text{ and } \quad \mathsf{u} = 1.5.$$

Unless stated otherwise, we set $J = 100$ for both samplers. We initialize all the MCMC samplers the same way and as follows. We randomly and uniformly select a subset $\mathsf{J} \subset \{1,\dots,p\}$ of size 100. For $j \in \mathsf{J}$ we set $\delta_j = 1$ and draw $\theta_j \sim \mathbf{N}(0, \rho_1^{-1})$, whereas for $j \notin \mathsf{J}$, we set $\delta_j = 0$ and $\theta_j = 0$..

We call Algorithm 1 the exact sampler (`Exact`), and we call Algorithm 2 the asynchronous sampler (`ASYN`).

4.1.1. *Comparison of mixing times.* We first compare the mixing times of the exact and the asynchronous samplers. We estimate empirically the mixing times using the coupling methodology of Biswas et al. (2019). We refer the reader to Appendix E for a description of the estimation method and the coupled chain used. To estimate the mixing times we replicated the coupled chains 50 times, and we estimate the mixing times for $p = 1,000$ to $p = 5,000$ by increment of 500. The estimated mixing times are given on Figure 1, and indeed shows a linear trend, which is consistent with the conclusion of Theorem 5. The results also show that the asynchronous sampler mixes slightly faster than Algorithm 1.

We also look at the sample path of the penalized log-likelihood

$$\bar{\ell}(\theta,\delta|Y',X') \stackrel{\text{def}}{=} -\frac{1}{2\sigma^2}\|Y' - X'\theta_\delta\|_2^2 - \frac{\rho_1}{2}\|\theta_\delta\|_2^2,$$

evaluated on a test dataset $(Y', X')$ (generated independently from the training set $(Y, X)$ but from the same model) along the MCMC iterations. By posterior contraction, we expect $\bar{\ell}(\theta^{(k)}, \delta^{(k)}|Y', X')$ to concentrate around $\bar{\ell}(\theta_\star, \delta_\star|Y', X')$ as $k \to \infty$. The speed with which $\bar{\ell}(\theta^{(k)}, \delta^{(k)}|Y', X')$ approaches $\bar{\ell}(\theta_\star, \delta_\star|Y', X')$ during the MCMC sampling is another empirical indication of mixing. Figures 2 and 3 show the averages of 50 penalized log-likelihood sample paths (for each MCMC sample we generate a new training and test datasets with the same $\theta_\star$). These averaged sample paths offer another look into the mixing of the samplers that is consistent with the empirical mixing times estimates.

4.1.2. *Comparison in terms of statistical performance.* Since the target distribution of the asynchronous sampler is biased, we investigate more systematically the bias by comparing the relative errors. On a given MCMC run we evaluate the accuracy of the parameter estimation by computing

$$\mathcal{E} \stackrel{\text{def}}{=} \frac{1}{\mathsf{Niter} - N_0} \sum_{k=N_0+1}^{\mathsf{Niter}} \frac{\|\theta^{(k)} \cdot \delta^{(k)} - \theta_\star\|_2}{\|\theta_\star\|_2}, \tag{21}$$

for a burn-in $N_0$ that we set at $N_0 = \mathsf{Niter} - 1000$, where $\mathsf{Niter}$ is the number of MCMC iterations. For this comparison we run the MCMC samplers for $\mathsf{Niter} = \max(2000, p-2000)$ number of iterations. Figure 4 and 5 show the distributions of the relative errors $\mathcal{E}$ produced by Algorithm 1 and 2 for $p \in \{1000, 5000\}$, and $\varrho \in \{0, 0.9\}$. These boxplots are based on 50 MCMC replications. Again, the difference remains small, even in the case $\varrho = 0.9$. We note that, since we scale $n = p/2$, and kept $s_\star = 10$, the simulation results are better for $p = 5,000$ than for $p = 1,000$.

4.1.3. *Comparison with* `Sparsevb`. We now compare our proposed sampler with the `Sparsevb` of Ray and Szabó (2021), a mean-field variational Bayes (VB) approximation for sparse high-dimensional linear regression. We generate the data as above with $p = 1000$, and $\varrho = 0.9$. We compare the algorithms in terms of the relative error $\mathcal{E}$ in (21), and in terms of running time, using 50 MCMC/VB replications (each replication is based on independently generated data from the same model). For the comparison we use the `R` package `Sparsevb` provided by the author with its default parametrization. We calculate the mixing time for `Sparsevb` using its default criterion, i.e. when maximum absolute differences between binary entropies of successive iterates is smaller than a prespecified tolerance. Figure 6 shows that `Asyn` tracks `Exact` better than `Sparsevb`, and is faster (although of course, part of the difference in running time may be software-induced such as difference in coding, or differences between `R` and `Matlab`).

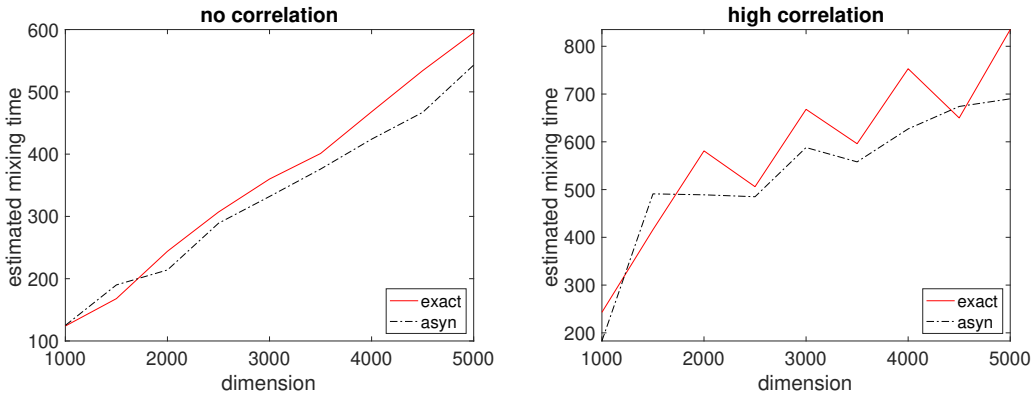


FIGURE 1. Estimated mixing time for a linear regression example. Figure on the left (resp. right) is $\varrho = 0$ (resp. $\varrho = 0.9$).

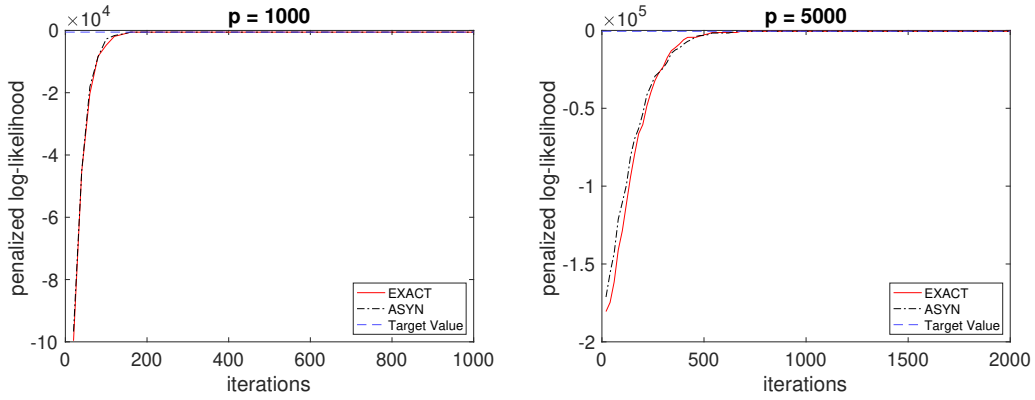


FIGURE 2. Averaged sample paths of penalized log-likelihood values in linear regression with $\varrho = 0$

4.1.4. *Dependence on J.* Here we explore how the update batch size parameter $J$ affects the sampler. We fix $p = 1000$, and $\varrho = 0.9$, and we vary $J$ from $J = 10$ to $J = 1000$ by increment of 10 until 100 and then by increment of 100. In this experiment we depart from the choice $n = p/2$, by setting either $n = 1000$ or $n = 150$. Given $J$, we evaluate the mixing of Algorithm 1 and Algorithm 2 using the coupled-chain methodology of Biswas et al. (2019) used above, that we describe in Appendix E, with 50 replications of the coupled chains (for a given dataset $Y, X$). And we evaluate the statistical performance of the sampler using

$$\mathcal{E}' \stackrel{\text{def}}{=} \frac{1}{\mathsf{Niter} - N_0} \sum_{k=N_0+1}^{\mathsf{Niter}} \|\theta^{(k)} \cdot \delta^{(k)} - \theta_\star\|_\infty.$$

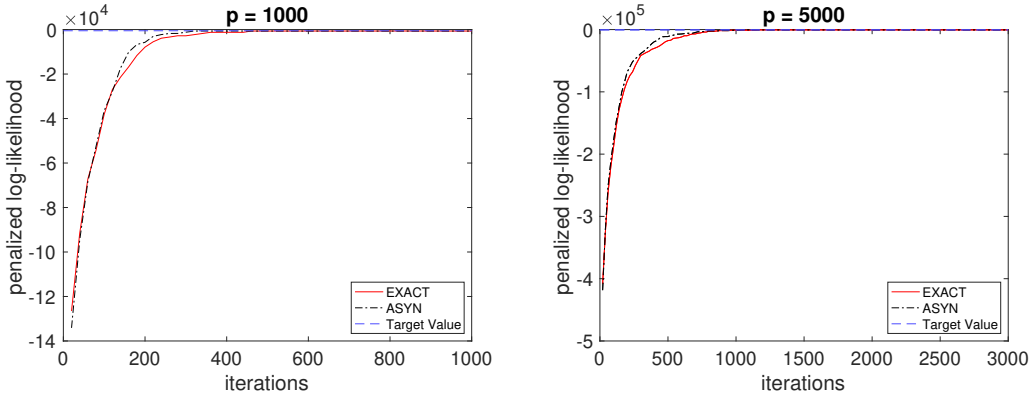


FIGURE 3. Averaged sample paths of penalized log-likelihood values in linear regression with $\varrho = 0.9$

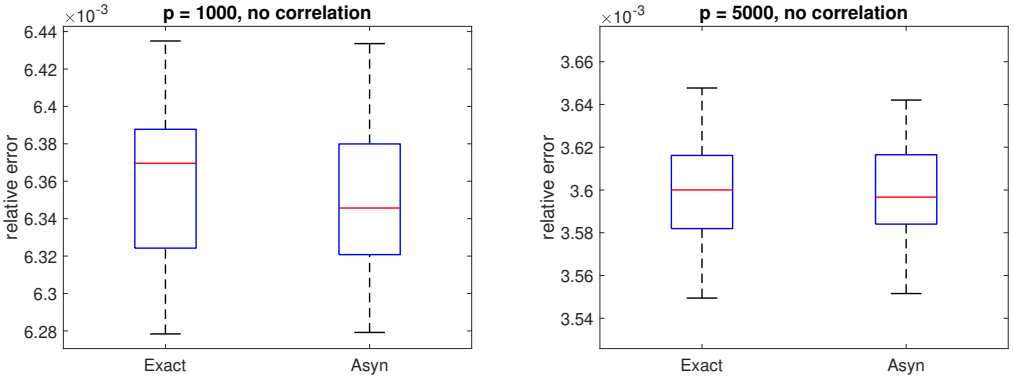


FIGURE 4. Distributions of the relative error (21) in linear regression with $\varrho = 0$. Based on 50 MCMC sample paths replications.

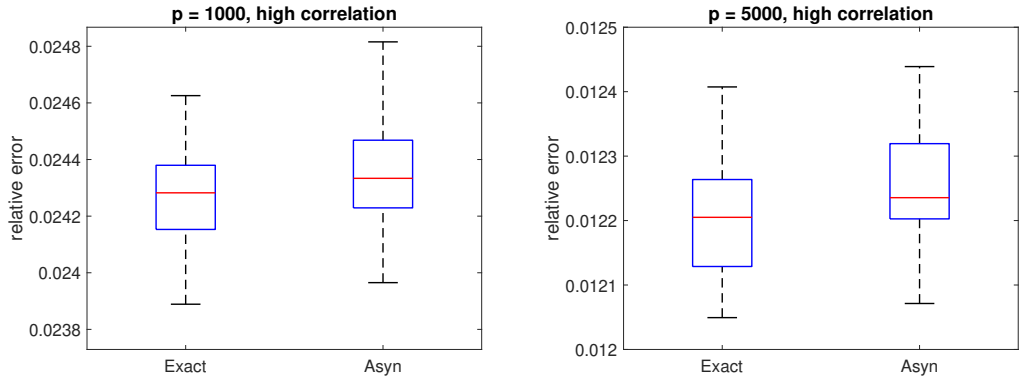


FIGURE 5. Distributions of the relative error (21) in linear regression with $\varrho = 0.9$. Based on 50 MCMC sample paths replications.

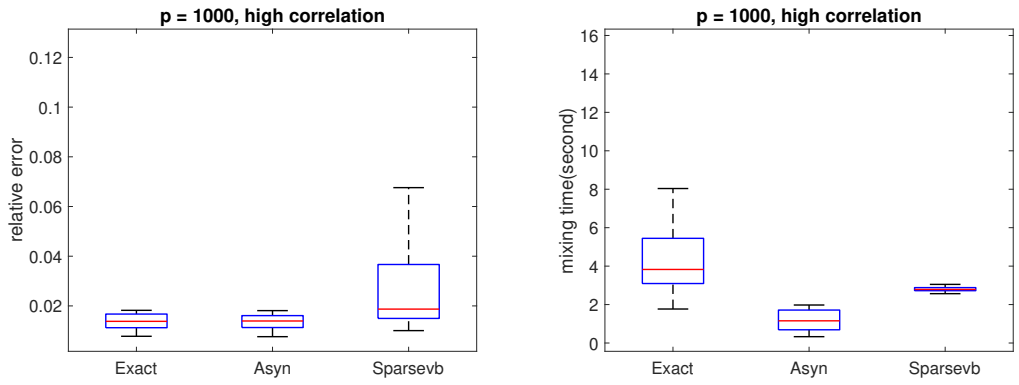


FIGURE 6. Relative error and Estimated mixing time for a linear regression example. Figure on the left (resp. right) is relative error (resp. Mixing time).

We obtain the distribution of $\mathcal{E}'$ and the distribution of the mixing time from 50 replications (each with a new dataset drawn from the same model). Figure 7 and 9 show the behavior of Algorithm 1 (the exact sampler) as we change $J$. As expected the statistical performance of the sampler is unchanged, and the mixing improved as $J$ increases (with increased computational cost per iteration, not reported here). Figure 8 and 10 show the behavior of the asynchronous sampler (Algorithm 2) as we vary $J$. Here also the mixing improves as $J$ increases, as predicted by Theorem 5, but the statistical performance degrades as $J$ increases, as expected and as predicted by Theorem 5. Comparing Figure 8 and 10, we note also that the bias of the asynchronous sampler increases as the sample sample size $n$ decreases, as predicted by the bound in Theorem 5. We note however that there is a large range of values from $J = 40$ to $J = 400$ where the performance of the algorithm remains very good for both values of $n$. In other words, $J$ is not hard to tune.

In summary, the parameter $J$ allows the user to trade statistical accuracy for computational speed. In all our simulations, we have conservatively set $J = 100$. A more aggressive scaling that works also well is to set $J = \min(\alpha p, n)$, where we recommend $\alpha \in (0, 0.1]$.

4.2. **Logistic regression.** We also illustrate the behavior of the proposed algorithms on logistic regression models. We use the same data generating set up for the regressors $X \in \mathbb{R}^{n\times p}$ and true signal $\theta_\star$ as above with $\varrho \in \{0, 0.9\}$, and we set $n = p/2$. We draw the response as $Y_i \sim \mathbf{Ber}(p_i)$, with $p_i = \left(1 + \exp^{-\langle x_i, \theta_\star\rangle}\right)^{-1}$, where $x_i$ denotes the $i$-th row of $X$. In this model we cannot draw exactly from the posterior conditional distribution of $\theta$ given $\delta$. Hence we implemented Algorithm 1 and Algorithm

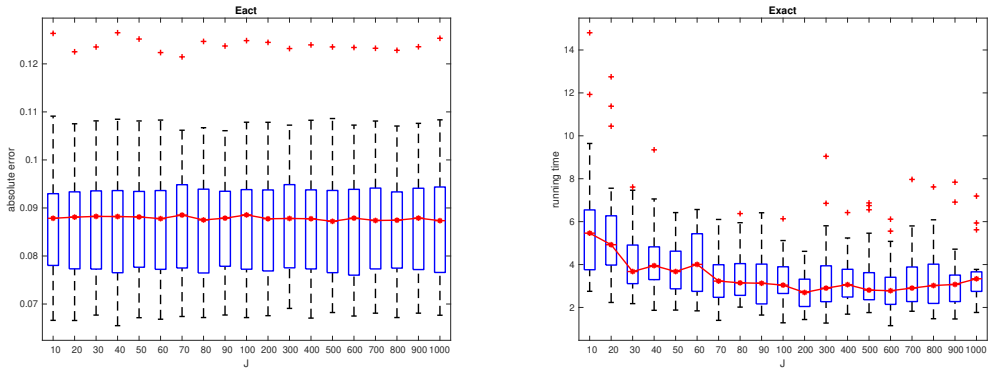


FIGURE 7. Absolute error and mixing time of Exact sampler with different $J$, linear regression, $p = 1000, n = 1000, \varrho = 0.9$

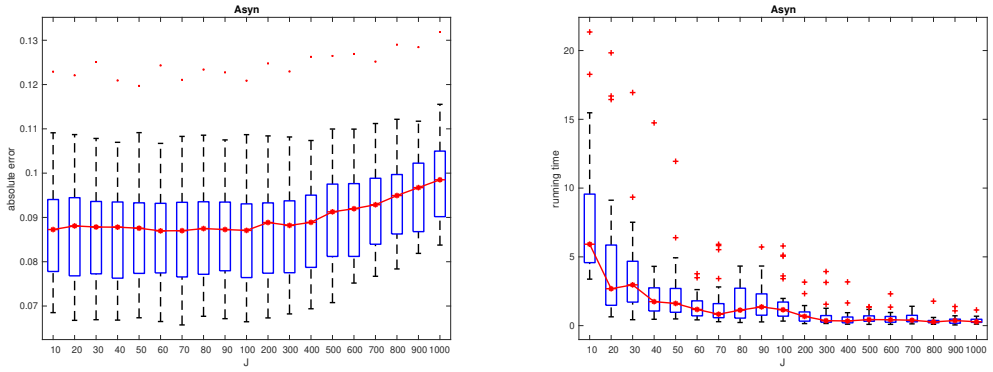


FIGURE 8. Absolute error and mixing time of Asyn sampler with different $J$, linear regression, $p = 1000, n = 1000, \varrho = 0.9$

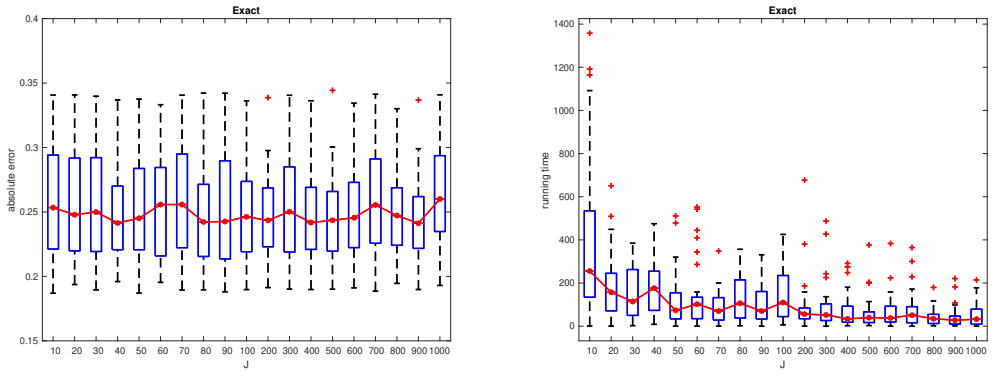


FIGURE 9. Absolute error and mixing time of Exact sampler with different $J$, linear regression, $p = 1000, n = 150, \varrho = 0.9$

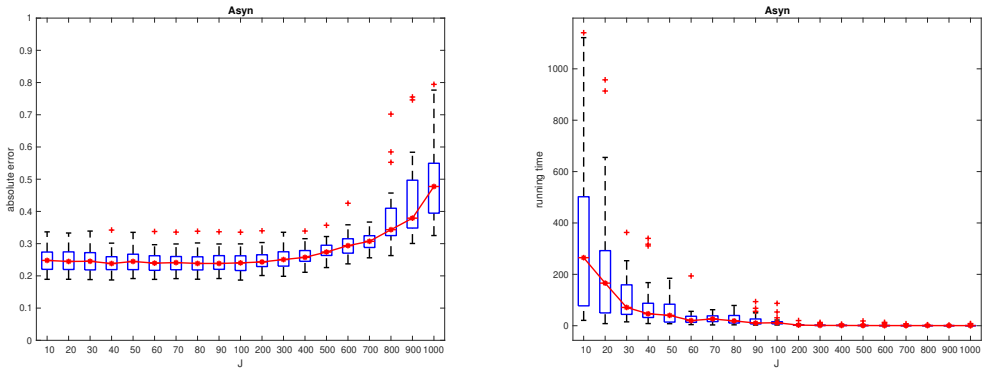


FIGURE 10. Absolute error and mixing time of Asyn sampler with different $J$, linear regression, $p = 1000, n = 150, \varrho = 0.9$

2 with $P_\delta$ taken as (one iteration of) the Metropolis Adjusted Langevin (MaLa) algorithm (Roberts and Tweedie (1996)), with a step-size fixed to $\gamma = 0.01$. In this example we also implement Algorithm 3 using a mini-batch of size $B = 100$, and a step-size fixed to $\gamma = 0.005$. These step-size and mini-batch size values were selected by trial and error. As with most MCMC samplers, implementing these algorithms on different models would require some tuning of step-size values. Adaptive MCMC methods may be considered (Atchadé et al. (2011)), but we do not pursue this here. For all the algorithms, we set $J = 100$, and we initialize all the algorithms from the lasso estimate of $\theta$. We call Algorithm 3 the sparse asynchronous SGLD sampler (`SA-SGLD`).

We compare the proposed algorithms with a mean-field variational approximation of the posterior distribution (2), and with the skinny-Gibbs sampler of (Narisetty et al. (2018)). Before presenting the results, we briefly describe these methods.

4.2.1. *Variational approximation.* We compare the proposed algorithms with a mean field variational approximation (VA) of (2), using the VA family

$$\prod_{j=1}^{p} \mathbf{Ber}(\alpha_j)(\mathrm{d}\delta_j)\mathbf{N}(\mu_j, v_j^2)(\mathrm{d}\theta_j),$$

with parameter $(\alpha_j, \mu_j, v_j^2)_{1\le j\le p}$, that we estimate by minimizing the ELBO objective function using stochastic gradient descent, following (Kingma and Welling (2014)). In the stochastic gradient descent we use a constant step-size of 0.001, we estimate the gradient by drawing small sample of size 100 from the VA family and small mini-batch of size 100 from the dataset. We stop the stochastic gradient descent when the

maximum relative change satisfies

$$\max\left(\frac{\|\alpha^{(k)}-\alpha^{(k-1)}\|_2}{\|\alpha^{(k)}\|_2}, \frac{\|\mu^{(k)}-\mu^{(k-1)}\|_2}{\|\mu^{(k)}\|_2}, \frac{\|v^{(k)}-v^{(k-1)}\|_2}{\|v^{(k)}\|_2}\right) \leq 0.0025. \tag{22}$$

And we evaluate the accuracy of the produced solution by computing on the last iteration the relative error

$$\frac{\|\mu^{(k)}\cdot\alpha^{(k)}-\theta_\star\|_2}{\|\theta_\star\|_2}.$$

For a fair comparison we also initialize $\mu^{(0)}$ from the same lasso estimate. The finding reported below are robust to the choices of mini-batch size, and stopping criterion in (22).

4.2.2. *Skinny Gibbs.* We also use this example to compare our method with the skinny Gibbs sampler of (Narisetty et al. (2018)), an approximate Gibbs sampler for high-dimensional logistic regression models which replaces the high dimensional covariance matrix with a sparse approximation. The comparison is performed using the `R` package `skinnybasad` provided by the authors, with its default parametrization. However, for a fair comparison we use the same lasso initialization as above.

4.2.3. *Evaluation metrics and results.* As with the linear regression example we compare the quality of the approximation by computing the relative error of each method using (21). The boxplots of these relative errors for different values of $p$ and $\varrho$ are shown on Figure 11-12. We observe from these results that our method remains on a par with the exact method, and outperforms VA and the skinny Gibbs, particularly in the high correlation case.

Next, we compare the computational cost and mixing times of the MCMC algorithms. We estimate empirically the mixing time of a sampler as the first time when its relative error (21) becomes smaller than a prescribed value $\tau$. However, since the algorithms have different relative errors in the limit, we proceed as follows. We run each sampler for a large number of iterations (equals to $5p$), and calculate its average relative error (21) after burn-in, that we then use as $\tau$. We then estimate the mixing time of the sampler as the first time where the relative error (21) is smaller or equal to $\tau$. For this comparison, we focus on the low correlation case where $\varrho = 0.0$ (in the high correlation setting VA and skinny Gibbs do not provide accurate approximation of the posterior). We estimate these mixing times for the dimension $p$ ranging from 1000 to 5000, with an increment of 500. For each $p$ we run 50 simulations and take the median.

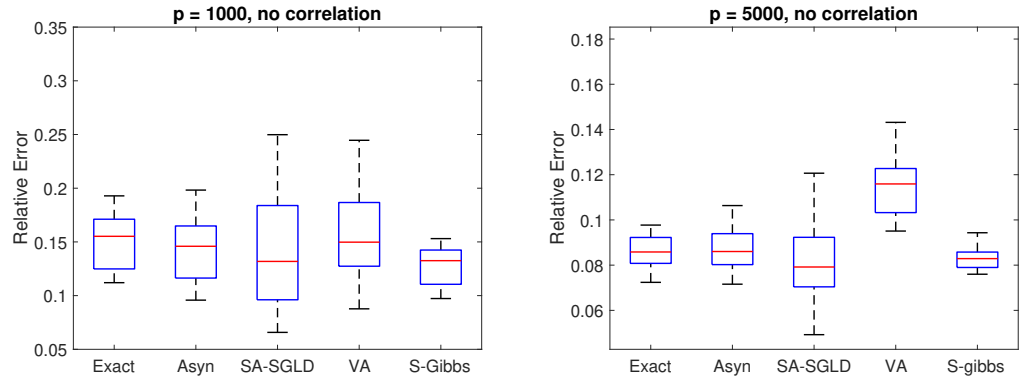


FIGURE 11. Relative error for logistic regression, with $\varrho = 0$

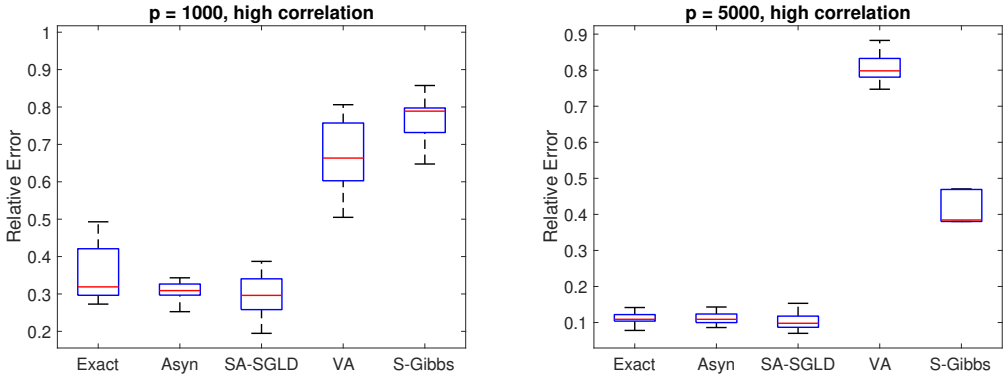


FIGURE 12. Relative error for logistic Regression, with $\varrho = 0.9$

| p/n | Complexity/iteration | 1000/500 | 2000/1000 | 5000/2500 |
|---|---|---|---|---|
| Exact | $O(nJ\|\delta^{(k)}\|_0)$ | 5.25s | 35.13s | 1360.09s |
| Asyn | $O(n(\|\delta^{(k)}\|_0 + J))$ | 0.71s | 2.19s | 99.04s |
| SA-SGLD | $O(B(\|\delta^{(k)}\|_0 + J))$ | 0.24s | 1.44s | 30.12s |
| Skinny-Gibbs | $O(n(p \vee \|\delta^{(k)}\|_0^2))$ | 10.50s | 87.27s | 1154.40s |
| VA | $O(B \cdot J \cdot p)$ | 4.05s | 34.42s | 1243.82s |

TABLE 1. Running times per iterations

Figure 13 shows the mixing times in terms of the number of iterations and the actual running times, and Table 1 shows the running times per iteration. We observe that the skinny-Gibbs sampler has very short mixing times. However, its cost per iteration is of order $O(np)$. As a results its overall running time has the same scaling as the running time of the exact sampler, and both are more expensive than the asynchronous and SA-SGLD samplers. We also notice that the mixing times (in terms of number of

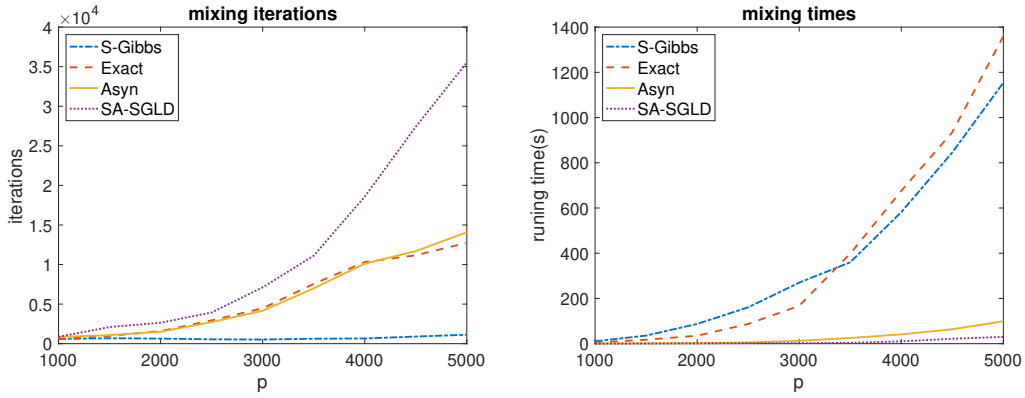


FIGURE 13. Mixing times in number of iterations (left), and in total running time. A mixing time is defined as first time when relative error is smaller than $\tau$, with **lasso** initialization, where $\tau$ is obtained from an initial long run.

iterations) for the `Exact` and `Asyn` samplers are similar, which is consistent with the results in the linear regression scenario. We also notice that overall the SA-SGLD sampler is not significantly faster than the asynchronous sampler. This is because, although the cost per iteration of SA-SGLD is much smaller, the algorithm typically requires a much longer mixing time.

4.3. **Illustration with a deep neural network model.** There is a growing interest in sparse deep learning (Gale et al. (2019); Frankle and Carbin (2019)). Most existing approach for estimating sparse deep learning models are frequentist. Bayesian deep learning can greatly facilitate uncertainty quantification in model predictions. As a proof of concept we apply Algorithm 3 to an image classification problem using the MNIST-FASHION dataset (Xiao et al. (2017)), and the deep neural network Lenet-5 (LeCun et al. (2010)). The MNIST-FASHION dataset[2] consists of $60,000$ data points $(y_i, \mathbf{x}_i)$ (plus another set of $10,000$ test samples), where $y_i \in \{1, \ldots, 10\}$ encodes the class of a fashion item (T-shirt, trouser, etc), and $\mathbf{x}_i$ is a $28 \times 28$ image of the item. We model the class outcome $y_i$ as independent random variables draws from a multinomial distribution:

$$y_i \sim \mathcal{M}(\mathcal{F}_\theta(\mathbf{x}_i)), \quad i = 1, \ldots, n,$$

with class probabilities proportional to $\exp(\mathcal{F}_\theta(\mathbf{x}_i))$, where $\mathcal{F}_\theta : \mathbb{R}^{28\times 28} \to \mathbb{R}^{10}$ is a lenet-5 neural network. We actually use a slightly modified lenet-5 architecture

[2]The data can be downloaded from `https://www.kaggle.com/datasets/zalando-research/fashionmnist`, or from the python package `keras`.

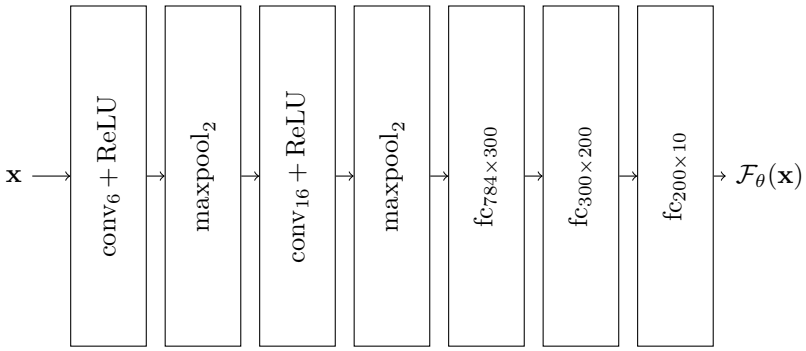


FIGURE 14. Illustration of the lenet-5 architecture.

obtained by replacing the tanh activation function by the ReLU function, and by enlarging the fully-connected layers. We refer the reader to Figure 14 for the architecture of the network, and to Zhang et al. (2019) for an introduction to neural network modeling. The total number of parameter is $p = 298,650$. For stability in the learned structures we did not sparsify the convolutional layers (specifically, we keep their corresponding $\delta_j$ set to 1).

For the Bayesian inference we use the hyper-parameter $\mathsf{u} = 50$, $\rho_0 = n$, $\rho_1 = 1$. We apply Algorithm 3 with SGLD on the selected components of $\theta$ with a fixed step-size $\gamma = 10^{-7}$. We set $J = 1,000$ (with stratified sampling accross the layer), and a batch size $B = 100$. We initialize the sampler from the full model with all components active, and the parameter $\theta$ initialized using the default initialization in Matlab. We then run Algorithm 3 for $\mathsf{Niter} = 250,000$ iterations and we use the first $150,000$ as burn-in. The running time took about 4.9 hours on a 8-core computer node with a NVIDIA TESLA V100 GPU system with 384 GB GPU memory, using MATLAB 2021a.

During the MCMC, at each iteration $k$, and for each $i$ in the test sample we define the prediction accurary as $\mathsf{A}_i^{(k)} \stackrel{\text{def}}{=} \mathbf{1}_{\{\hat{y}_i^{(k)}=y_i\}}$, where $\hat{y}_i^{(k)} \sim \mathcal{M}(\mathcal{F}_{\theta^{(k)}\cdot\delta^{(k)}}(\mathbf{x}_i))$. We average these prediction accuracies to get $\bar{\mathsf{A}}^{(k)}$. We also average the prediction accuracies within each group of items to get $\bar{\mathsf{A}}^{(k)}(g)$, $g = 1,\ldots,10$. To save time we actually compute these statistics only every 100 iterations. Figure 15 plots $\{\bar{\mathsf{A}}^{(k)},\ k\}$ and the model sparsity $\{\|\delta^{(k)}\|_0/p,\ k\}$ along the MCMC iterations, and Figure 16 shows the boxplots of the $\{\bar{\mathsf{A}}^{(k)}(g),\ k\}$ for each $g$. Table 2 shows the posterior sparsity and posterior average accuracy, and includes a comparison to Monte Carlo dropout (Gal and Ghahramani (2016)). The results shows that it is possible to significantly compress deep learning models with only modest loss of performance.

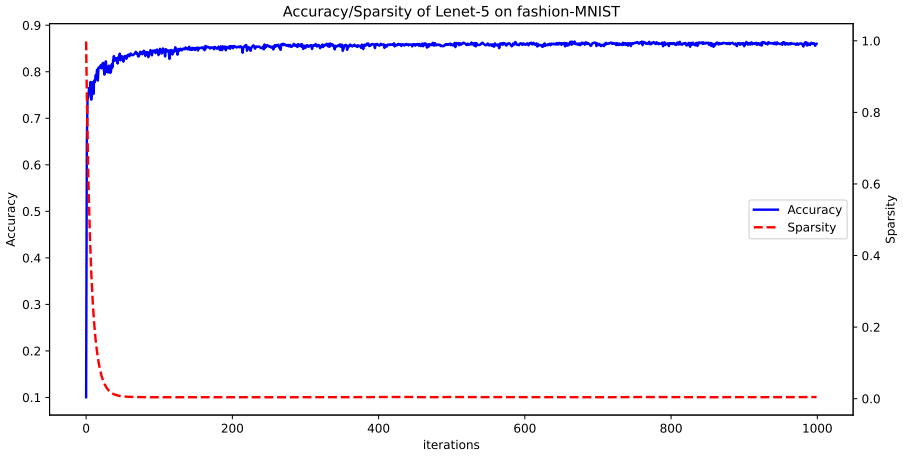


FIGURE 15. Prediction accuracy on test sample and sparsity along MCMC run.

| | Sparsity | Accuracy |
|---|---|---|
| SA-SGLD | 1.00 (0.00) | 86.5 (0.45) |
| MC Dropout | 100 | 88.65 (0.06) |

TABLE 2. Estimated posterior sparsity and prediction accuracy on test sample (in percentage)

The computational cost (per iteration) of the algorithm is roughly twice that of stochastic gradient descent, its frequentist counterpart. Note however that this cost can potentially be further reduced by exploiting sparsity (as we did with linear and logistic regression models). We did not pursue this here because MATLAB 2021a that we used for this project does not support sparse deep learning computation.

We end with some words of caution. We are presenting this example mainly as an exploratory exercise in the potential of the proposed framework, without much theoretical guarantee. In particular, due to the poor general understanding of deep neural network models, we currently cannot say much about the properties of the limiting distribution of Algorithm 3. Furthermore, due to the highly multimodal nature of the likelihood surface of deep neural network models, we cannot guarantee either that the algorithm has mixed and is correctly sampling from its limiting distribution. More research is needed on these issues.

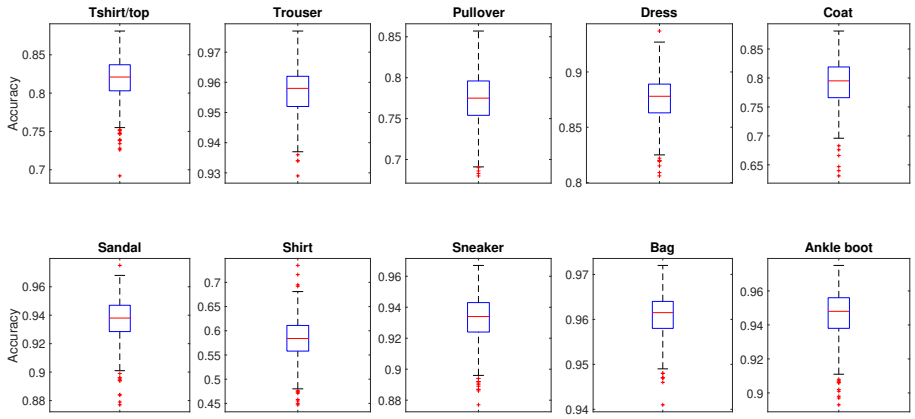


FIGURE 16. Distribution of posterior predictive accuracy on MNIST-Fashion test samples. Averaged within each class of item.

## 5. Some concluding remarks

We have proposed a fast MCMC algorithm for the Bayesian analysis of sparse high-dimensional models. The algorithm operates as a form of Bayesian iterated sure independent screening, resulting in tremendous speed. In linear regression models we show that the algorithm mixes quickly to a limiting distribution that recovers correctly the main underlying signal. In limited sample size problems the algorithm can be advantageously combined with tempering techniques (such as simulated tempering or related ideas) for better mixing properties. Such extensions could also be particularly useful in deep learning where the resulting posterior distributions are known to be highly multimodal. One interesting aspect of the MCMC analysis in this work (which extends from De Sa et al. (2016)), is the use of a metric weaker than the total variation metric and more directly pertinent for the statistical analysis. Exploring more systematically these ideas could be of broader theoretical interest.

## References

Atchadé, Y. (2022). Approximate spectral gaps for markov chains in high-dimensions. *SIAM Journal on Mathematics of Data Science* .

Atchade, Y. and Bhattacharyya, A. (2019). An approach to large-scale quasi-bayesian inference with spike-and-slab priors.

Atchadé, Y., Fort, G., Moulines, E. and Priouret, P. (2011). Adaptive Markov chain Monte Carlo: theory and methods. In *Bayesian time series models*. Cambridge Univ. Press, Cambridge, 32–51.

Atchade, Y. A. (2017). On the contraction properties of some high-dimensional quasi-posterior distributions. *Ann. Statist.* **45** 2248–2273.

Bhadra, A., Datta, J., Li, Y. and Polson, N. (2020). Horseshoe regularisation for machine learning in complex and deep models1. *International Statistical Review* **88** 302–320.

Bhattacharya, A., Chakraborty, A. and Mallick, B. K. (2016). Fast sampling with gaussian scale mixture priors in high-dimensional regression. *Biometrika* asw042.

Biswas, N., Bhattacharya, A., Jacob, P. E. and Johndrow, J. E. (2021). Coupled markov chain monte carlo for high-dimensional regression with half-t priors.

Biswas, N., Jacob, P. E. and Vanetti, P. (2019). Estimating convergence of markov chains with l-lag couplings.

Blei, D. M., Kucukelbir, A. and McAuliffe, J. D. (2016). Variational Inference: A Review for Statisticians. *ArXiv e-prints* arXiv:1601.00670.

Brooks, S., Gelman, A., Jones, G. and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Bühlmann, P. and van de Geer, S. (2011). *Statistics for high-dimensional data*. Springer Series in Statistics, Springer, Heidelberg. Methods, theory and applications.

Candès, E. J. and Plan, Y. (2009). Near-ideal model selection by l1 minimization. *The Annals of Statistics* **37** 2145 – 2177.

Carvalho, C. M., Polson, N. G. and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika* **97** 465–480.

Castillo, I., Schmidt-Hieber, J. and van der Vaart, A. (2015). Bayesian linear regression with sparse priors. *Ann. Statist.* **43** 1986–2018.

Daskalakis, C., Dikkala, N. and Jayanti, S. (2018). Hogwild!-gibbs can be panaccurate.

De Sa, C., Olukotun, K. and Ré, C. (2016). Ensuring rapid mixing and low bias for asynchronous gibbs sampling. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16.

Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70** 849–911.
URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2008.00674.x`

Fan, J., Samworth, R. and Wu, Y. (2009). Ultrahigh dimensional feature selection: Beyond the linear model. *J. Mach. Learn. Res.* **10** 2013–2038.

FRANKLE, J. and CARBIN, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

GAL, Y. and GHAHRAMANI, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*. PMLR.

GALE, T., ELSEN, E. and HOOKER, S. (2019). The state of sparsity in deep neural networks. *ArXiv* **abs/1902.09574**.

GEORGE, E. I. and MCCULLOCH, R. E. (1997). Approaches to bayesian variable selection. *Statist. Sinica* **7** 339–373.

GHOSH, S., YAO, J. and DOSHI-VELEZ, F. (2019). Model selection in bayesian neural networks via horseshoe priors. *Journal of Machine Learning Research* **20** 1–46.

HASTIE, T., TIBSHIRANI, R. and WAINWRIGHT, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC.

JOHNDROW, J., ORENSTEIN, P. and BHATTACHARYA, A. (2020). Scalable approximate mcmc algorithms for the horseshoe prior. *Journal of Machine Learning Research* **21**.

JOHNDROW, J. E. and MATTINGLY, J. C. (2018). Error bounds for approximations of markov chains used in bayesian sampling.

JOHNSON, M. J., SAUNDERSON, J. and WILLSKY, A. S. (2013). Analyzing hogwild parallel gaussian gibbs sampling. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13.

KINGMA, D. P. and WELLING, M. (2014). Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

LECUN, Y., CORTES, C. and BURGES, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* **2**.

LOUIZOS, C., ULLRICH, K. and WELLING, M. (2017). Bayesian compression for deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17.

MA, Y.-A., CHEN, T. and FOX, E. B. (2015). A complete recipe for stochastic gradient mcmc. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15, MIT Press, Cambridge, MA, USA.

MEINSHAUSEN, N. and YU, B. (2009). Lasso-type recovery of sparse representations for high-dimensional data. *Ann. Statist.* **37** 246–270.

MEYN, S. and TWEEDIE, R. L. (2009). *Markov chains and stochastic stability*. 2nd ed. Cambridge University Press, Cambridge.

MITCHELL, T. J. and BEAUCHAMP, J. J. (1988). Bayesian variable selection in linear regression. *Journal of the american statistical association* **83** 1023–1032.

NARISETTY, N. N., SHEN, J. and HE, X. (2018). Skinny gibbs: A consistent and scalable gibbs sampler for model selection. *Journal of the American Statistical Association* .

NEGAHBAN, S. N., RAVIKUMAR, P., WAINWRIGHT, M. J. and YU, B. (2012). A unified framework for high-dimensional analysis of $m$-estimators with decomposable regularizers. *Statistical Science* **27** 538–557.

NEISWANGER, W., WANG, C. and XING, E. P. (2014). Asymptotically exact, embarrassingly parallel mcmc. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, USA.

PIIRONEN, J. and VEHTARI, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics* **11** 5018 – 5051.

PILLAI, N. S. and SMITH, A. (2015). Ergodicity of approximate mcmc chains with applications to large data sets.

RAJARATNAM, B., SPARKS, D., KHARE, K. and ZHANG, L. (2019). Uncertainty quantification for modern high-dimensional regression via scalable bayesian methods. *Journal of Computational and Graphical Statistics* **28** 174–184.

RAY, K. and SZABÓ, B. (2021). Variational bayes for high-dimensional linear regression with sparse priors. *Journal of the American Statistical Association* 1–12.

ROBERT, C. P. and CASELLA, G. (2004). *Monte Carlo statistical methods*. 2nd ed. Springer Texts in Statistics, Springer-Verlag, New York.

ROBERTS, G. and TWEEDIE, R. (1996). Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli* **2** 341–363.

ROCKOVA, V. and GEORGE, E. I. (2018). The spike-and-slab lasso. *Journal of the American Statistical Association* **113** 431–444.

RUDOLF, D. and SCHWEIZER, N. (2018). Perturbation theory for Markov chains via Wasserstein distance. *Bernoulli* **24** 2610 – 2639.

SMOLA, A. and NARAYANAMURTHY, S. (2010). An architecture for parallel topic models. *Proc. VLDB Endow.* **3** 703–710.

SRIVASTAVA, S. and XU, Y. (2021). Distributed bayesian inference in linear mixed-effects models. *Journal of Computational and Graphical Statistics* **30** 594–611.

TRAN, M.-N., NGUYEN, N., NOTT, D. and KOHN, R. (2020). Bayesian deep net glm and glmm. *Journal of Computational and Graphical Statistics* **29** 97–113.

WAINWRIGHT, M. J. (2019). *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

WELLING, M. and TEH, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML'11, Omnipress, USA.

XIAO, H., RASUL, K. and VOLLGRAF, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* .

XUE, J. and LIANG, F. (2019). Double-parallel monte carlo for bayesian analysis of big data. *Statistics and Computing* **29**.

YANG, Y., WAINWRIGHT, M. J. and JORDAN, M. I. (2016). On the computational complexity of high-dimensional bayesian variable selection. *Ann. Statist.* **44** 2497–2532.

ZHANG, A., LIPTON, Z. C., LI, M. and SMOLA, A. J. (2019). *Dive into Deep Learning*. `http://www.d2l.ai`.

ZHANG, F. and GAO, C. (2020). Convergence rates of variational posterior distributions. *The Annals of Statistics* **48** 2180 – 2207.

(1) Figure 1: Estimated mixing time for a linear regression example. Figure on the left (resp. right) is $\varrho = 0$ (resp. $\varrho = 0.9$).
(2) Figure 2: Averaged sample paths of penalized log-likelihood values in linear regression with $\varrho = 0$
(3) Figure 3: Averaged sample paths of penalized log-likelihood values in linear regression with $\varrho = 0.9$
(4) Figure 4: Distributions of the relative error (21) in linear regression with $\varrho = 0$. Based on 50 MCMC sample paths replications.
(5) Figure 5: Distributions of the relative error (21) in linear regression with $\varrho = 0.9$. Based on 50 MCMC sample paths replications.
(6) Figure 6: Relative error and Estimated mixing time for a linear regression example. Figure on the left (resp. right) is relative error (resp. Mixing time).
(7) Figure 7: Absolute error and mixing time of Exact sampler with different $J$, linear regression, $p = 1000, n = 1000, \varrho = 0.9$
(8) Figure 8: Absolute error and mixing time of Asyn sampler with different $J$, linear regression, $p = 1000, n = 1000, \varrho = 0.9$
(9) Figure 9: Absolute error and mixing time of Exact sampler with different $J$, linear regression, $p = 1000, n = 150, \varrho = 0.9$
(10) Figure 10: Absolute error and mixing time of Asyn sampler with different $J$, linear regression, $p = 1000, n = 150, \varrho = 0.9$
(11) Figure 11: Relative error for logistic regression, with $\varrho = 0$
(12) Figure 12: Relative error for logistic Regression, with $\varrho = 0.9$
(13) Figure 13: Mixing times in number of iterations (left), and in total running time. A mixing time is defined as first time when relative error is smaller than $\tau$, with **lasso** initialization, where $\tau$ is obtained from an initial long run.
(14) Figure 14: Illustration of the lenet-5 architecture.
(15) Figure 15: Prediction accuracy on test sample and sparsity along MCMC run.
(16) Figure 16: Distribution of posterior predictive accuracy on MNIST-Fashion test samples. Averaged within each class of item.