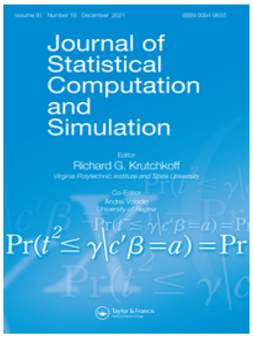

Taylor & Francis
Taylor & Francis Group

Journal of Statistical Computation and Simulation

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/gscs20

Stochastic gradient Langevin dynamics with adaptive drifts

Sehwan Kim, Qifan Song & Faming Liang

To cite this article: Sehwan Kim, Qifan Song & Faming Liang (2021): Stochastic gradient Langevin dynamics with adaptive drifts, Journal of Statistical Computation and Simulation, DOI: 10.1080/00949655.2021.1958812

To link to this article: https://doi.org/10.1080/00949655.2021.1958812

JOURNAL OF STATISTICAL COMPUTATION AND SIMULATION
https://doi.org/10.1080/00949655.2021.1958812

Stochastic gradient Langevin dynamics with adaptive drifts

Sehwan Kim, Qifan Song and Faming Liang

Department of Statistics, Purdue University, West Lafayette, IN, USA

ABSTRACT

We propose a class of adaptive stochastic gradient Markov chain Monte Carlo (SGMCMC) algorithms, where the drift function is adaptively adjusted according to the gradient of past samples to accelerate the convergence of the algorithm in simulations of the distributions with pathological curvatures. We establish the convergence of the proposed algorithms under mild conditions. The numerical examples indicate that the proposed algorithms can significantly outperform the popular SGMCMC algorithms, such as stochastic gradient Langevin dynamics (SGLD), stochastic gradient Hamiltonian Monte Carlo (SGHMC) and preconditioned SGLD, in both simulation and optimization tasks. In particular, the proposed algorithms can converge quickly for the distributions for which the energy landscape possesses pathological curvatures.

ARTICLE HISTORY
Received 8 April 2021
Accepted 19 July 2021

KEYWORDS
Adaptive MCMC; deep neural network; mini-batch data; momentum; stochastic gradient MCMC

1. Introduction

Bayesian learning is an important field in machine learning, which potentially is able to quantify uncertainty of the prediction, leading to safe decision-making for artificial intelligence (AI) machines. When the dataset is big, Bayesian learning is often conducted using the stochastic gradient Langevin dynamics (SGLD) algorithm [1], which is scalable by using a mini-batch of data to estimate the required gradient during iterations. However, SGLD suffers from poor mixing rates when pathological curvatures present on the energy landscape, i.e. the target density function has strong couplings and scales differently across dimensions. This difficulty is caused by the mismatch between the geometry of the energy landscape and the isotropic Langevin diffusion that SGLD relies on. As pointed out by Simsekli et al. [2], the isotropic Langevin diffusion implicitly assumes that different components of the underlying variable are uncorrelated and of the same scale, which is hardly satisfied in real applications.

To tackle this difficulty, the second-order gradient algorithms, such as stochastic gradient Fisher scoring (SGFS) [3] and stochastic gradient Riemannian Langevin dynamics (SGRLD) [4,5] have been developed. With the use of the Fisher information matrix of the target distribution, these algorithms rescale the stochastic gradient noise to be isotropic near stationary points, which accelerates the convergence of the simulation. However, calculation of the Fisher information matrix can be time-consuming, which makes

CONTACT F. Liang fmliang@purdue.edu Department of Statistics, Purdue University, West Lafayette, 47906 IN, USA

these algorithms lack scalability necessary for learning large-scale models such as deep neural networks (DNNs). Instead of using the exact Fisher information matrix, preconditioned SGLD (pSGLD) [6] approximates it by a diagonal matrix adaptively updated with the current gradient information, and stochastic quasi-Newton Langevin Monte Carlo (SQNLMC) [2] approximates the Fisher information matrix using a limited history of samples and their stochastic gradients. Orthogonal to the use of the Fisher information of the target distribution, stochastic gradient Hamiltonian Monte Carlo (SGHMC) [7], stochastic gradient Nosé-Hoover thermostats (SGNHT) [8] and relativistic Monte Carlo [9] improve the mixing of SGLD with momentum, where the moving direction can be adjusted by the momentum term in a similar way to the Fisher information matrix. Refer to Lu et al. [9] for more discussions on this issue.

Ma et al. [10] provides a general framework (see Section 2) for the existing SGMCMC algorithms based on the Wiener process, where the stochastic gradient is restricted to be unbiased. However, this restriction is unnecessary. As shown in the recent work, see, e.g. Dalalyan and Karagulyan [11], Song et al. [12] and Bhatia et al. [13], the stochastic gradient can be biased as long as its mean squared error can be upper bounded by an appropriate function of θ_t, the current sample of the stochastic gradient Markov chain. On the other hand, a variety of adaptive SGD algorithms, such as momentum [14], Adagrad [15], RMSprop [16], and Adam [17], have been proposed in the recent literature to accelerate the convergence of SGD. These algorithms adjust the moving direction at each iteration according to the current gradient as well as the past ones. It was shown in Staib et al. [18] that, compared to SGD, these algorithms escape saddle points faster and can converge faster overall to the second-order stationary points.

Motivated by the above two observations, we propose a class of adaptive SGLD algorithms, where a bias term is included in the drift function to enhance escape from saddle points and accelerate the convergence in the presence of pathological curvatures. The bias term can be adaptively adjusted based on the path of the sampler. In particular, we propose to adjust the bias term based on the past gradients in the flavour of adaptive SGD algorithms [19]. We establish the convergence of the proposed adaptive SGLD algorithms under mild conditions, and demonstrate via numerical examples that the adaptive SGLD algorithms can significantly outperform the popular SGMCMC algorithms, such as SGLD, SGHMC and pSGLD.

The remaining part of the paper is organized as follows. Section 2 gives a brief review for SGLD and its general formulation. Section 3 describes the proposed adaptive SGLD algorithms and studies its convergence. Section 4 illustrates the performance of the proposed algorithms via simulated examples. Section 5 applies the proposed algorithms to deep neural networks. Section 6 concludes the paper with a brief discussion.

2. A brief review of SGLD and its general formulation

Let $X_N = (X_1, X_2, \ldots, X_N)$ denote a set of N independent and identically distributed samples drawn from the distribution $f(x \mid \theta)$, where N is the sample size and θ is the vector of parameters. Let $p(X_N \mid \theta) = \prod_{i=1}^{N} f(X_i \mid \theta)$ denote the likelihood function, let $\pi(\theta)$ denote the prior distribution of θ, and let $U(\theta) = -\log p(X_N \mid \theta) - \log \pi(\theta)$ denote the energy function of the posterior distribution. If θ has a fixed dimension and $U(\theta)$ is differentiable

with respect to θ, then SGLD can be used to simulate from the posterior by iterating

$$\theta_{t+1} = \theta_t - \epsilon_{t+1}\nabla_\theta \tilde{U}(\theta_t) + \sqrt{2\epsilon_{t+1}\tau}\eta_{t+1}, \quad \eta_{t+1} \sim N(0, I_d),$$

where d is the dimension of θ, I_d is an $d \times d$-identity matrix, ϵ_{t+1} is the learning rate, τ is the temperature, and $\nabla_\theta \tilde{U}(\theta)$ denotes a noisy estimate of $\nabla_\theta U(\theta)$ based on a mini-batch of data. The learning rate can be kept as a constant or decay with iterations. For the former, the convergence of the algorithm has been studied in Sato and Nakagawa [20] and Dalalyan and Karagulyan [11]. For the latter, the convergence of the algorithm has been studied in Teh et al. [21].

As mentioned previously, many variants of SGLD have been developed in the literature and they can be formulated under the same framework based on the Wiener process [10]. Let ξ denote an augmented state, which may include some auxiliary components. For example, SGHMC augments the state to $\xi = (\theta, \nu)$ by including an auxiliary velocity component denoted by ν. Then the general SGMCMC algorithm is given by

$$\theta_{t+1} = \theta_t - \epsilon_{t+1}[D(\xi) + Q(\xi)]\nabla_\xi \tilde{H}(\xi) + \Gamma(\xi) + \sqrt{2\epsilon_{t+1}\tau}Z_{t+1}, \tag{1}$$

where $Z_{t+1} \sim N(0, D(\xi_t))$, $H(\xi)$ is the energy function of the augmented system, $\nabla_\xi \tilde{H}(\xi)$ denotes an unbiased estimate of $\nabla_\xi H(\xi)$, $D(\xi)$ is a positive semi-definite diffusion matrix, $Q(\xi)$ is a skew-symmetric curl matrix, and $\Gamma_i(\xi) = \sum_{j=1}^{d} \frac{\partial}{\partial \xi_j}(D_{ij}(\xi) + Q_{ij}(\xi))$. The diffusion $D(\xi)$ and curl $Q(\xi)$ matrices can take various forms and the choice of the matrices will affect the rate of convergence of the sampler. For example, for the SGHMC algorithm, we have $H(\xi) = U(\theta) + \frac{1}{2}\nu^T\nu$, $D(\xi) = \left(\begin{smallmatrix} 0 & 0 \\ 0 & C \end{smallmatrix}\right)$ for some positive semi-definite matrix C, and $Q(\xi) = \left(\begin{smallmatrix} 0 & -I \\ I & 0 \end{smallmatrix}\right)$. For the SGRLD algorithm, we have $\xi = \theta, H(\xi) = U(\theta), D(\xi) = G(\theta)^{-1}$, $Q(\xi) = 0$, where $G(\theta)$ is the Fisher information matrix of the posterior distribution. Refer to Nemeth and Fearnhead [22] for more examples on this formulation.

3. Stochastic gradient Langevin dynamics with adaptive drifts

Motivated by the observations that the stochastic gradient used in SGLD is not necessarily unbiased and that the past gradients can be used to enhance escape from saddle points for SGD, we propose a class of adaptive SGLD algorithms, where the past gradients are used to accelerate the convergence of the sampler by forming a bias to the drift at each iteration. A general form of the adaptive SGLD algorithm is given by

$$\theta_{t+1} = \theta_t - \epsilon_{t+1}(\nabla_\theta \tilde{U}(\theta_t) + aA_t) + \sqrt{2\epsilon_{t+1}\tau}\eta_{t+1}, \tag{2}$$

where A_t is the adaptive bias term, a is called the bias factor, and $\eta_{t+1} \sim N(0, I_d)$. Two adaptive SGLD algorithms are given in what follows. In the first algorithm, the bias term is constructed based on the momentum algorithm [14]; and in the second algorithm, the bias term is constructed based on the Adam algorithm [17]. Due to the inclusion of the bias term, the proposed algorithms do not follow the framework provided by (1).

3.1. Momentum SGLD

It is known that SGD has trouble in navigating ravines, i.e. the regions where the energy surface curves much more steep in one dimension than in another, which are common

around local energy minima [19,23]. In this scenario, SGD oscillates across the slopes of the ravine while making hesitant progress towards the local energy minima. To accelerate SGD in the relevant direction and dampen oscillations, the momentum algorithm [14] updates the moving direction at each iteration by adding a fraction of the moving direction of the past iteration, the so-called momentum term, to the current gradient. By accumulation, the momentum term increases updates for the dimensions whose gradients pointing in the same directions and reduces updates for the dimensions whose gradients change directions. As a result, the oscillation is reduced and the convergence is accelerated.

As an analogy of the momentum algorithm in stochastic optimization, we propose the so-called momentum SGLD (MSGLD) algorithm, where the momentum is calculated as an exponentially decaying average of past stochastic gradients and added as a bias term to the drift of SGLD. The resulting algorithm is depicted in Algorithm 1. The ergodicity of the algorithm is established in Theorem 3.1, whose proof is given in the Appendix.

Algorithm 1 MSGLD.

Input: Data $\{x_i\}_{i=1}^N$, subsample size n, smoothing factor $0 < \beta_1 < 1$, bias factor a, temperature τ, and learning rate sequence $\{\epsilon_t : t = 1, 2, \ldots\}$;
Initialization: θ_0 from an appropriate distribution, and $m_0 = 0$.
for $t = 1, 2, \ldots,$ **do**
Draw a mini-batch of data $\{x_j^*\}_{j=1}^n$, and calculate
$\theta_{t+1} = \theta_t - \epsilon_{t+1}(\nabla \tilde{U}(\theta_t) + am_t) + e_{t+1}$,
$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla \tilde{U}(\theta_{t-1})$,
where $e_{t+1} \sim N(0, 2\tau\epsilon_{t+1} I_d)$, and d is the dimension of θ.
end for

Theorem 3.1 (Ergodicity of MSGLD): *Suppose that the conditions* (A.1)–(A.4) *hold (given in Appendix), and* $\beta_1 \in (0, 1]$ *is a constant. For any smooth function* $\phi(\theta)$*, let* $\hat{\phi}_L = \frac{\sum_{k=1}^L \epsilon_k \phi(\theta_k)}{\sum_{k=1}^L \epsilon_k}$ *and* $\bar{\phi} = \int_\Theta \phi(\theta)\pi_*(\theta)\,\mathrm{d}\theta$*, where* $\pi_*(\theta)$ *denotes the target posterior distribution.*

(i) *If a constant learning rate of* ϵ *is used, then* $E\|\hat{\phi}_L - \bar{\phi}\|^2 = O(\epsilon^2)$ *as* $L \to \infty$.
(ii) *If the learning rate sequence* $\{\epsilon_k : k = 1, 2, \ldots\}$ *decays and satisfies the conditions that* $\sum_{k=1}^\infty \epsilon_k = \infty$ *and* $\lim_{L\to\infty} \frac{\sum_{k=1}^L \epsilon_k^2}{\sum_{k=1}^L \epsilon_k} = 0$*, then* $E\|\hat{\phi}_L - \bar{\phi}\|^2 \to 0$ *as* $L \to \infty$.

Algorithm 1 contains a few parameters, including the subsample size n, smoothing factor β_1, bias factor a, temperature τ, and learning rate ϵ. Among these parameters, n, τ and ϵ are shared with SGLD and can be set as in SGLD. Refer to Nagapetyan et al. [24] and Nemeth and Fearnhead [22] for more discussions on their settings. The smoothing factor β_1 is a constant, which is typically set to 0.9. The bias factor a is also a constant, which is typically set to 1 or a slightly large value.

3.2. Adam SGLD

The Adam algorithm [17] has been widely used in deep learning, which typically converges much faster than SGD. Recently, Staib et al. [18] showed that Adam can be viewed as a preconditioned SGD algorithm, where the preconditioner is estimated in an on-line manner and it helps escape saddle points by rescaling the stochastic gradient noise to be isotropic near stationary points.

Motivated by this result, we propose the so-called Adam SGLD (ASGLD) algorithm. Ideally, we would construct the adaptive bias term as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla\tilde{U}(\theta_{t-1}),\\ \tilde{V}_t &= \beta_2 \tilde{V}_{t-1} + (1-\beta_2)\tilde{U}(\theta_{t-1})\tilde{U}(\theta_{t-1})^{\mathrm{T}},\\ \tilde{A}_t &= \tilde{V}_t^{-1/2} m_t, \end{aligned} \tag{3}$$

where β_1 and β_2 are smoothing factors for the first and second moments of stochastic gradients, respectively. Since $\tilde{V}_t$ can be viewed as an approximator of the true second moment matrix $E(\nabla_\theta \tilde{U}(\theta_{t-1})\nabla_\theta \tilde{U}(\theta_{t-1})^{\mathrm{T}})$ at iteration $t-1$, $\tilde{A}_t$ can viewed as the rescaled momentum which is isotropic near stationary points. If the bias factor a is chosen appropriately, ASGLD is expected to converge very fast. In particular, the bias term may guide the sampler to converge to a global optimal region quickly, similar to Adam in optimization. However, when the dimension of θ is high, calculation of $\tilde{V}_t$ and $\tilde{V}_t^{-1/2}$ can be time consuming. To accelerate computation, we propose to approximate $\tilde{V}_t$ using a diagonal matrix as in pSGLD. This leads to Algorithm 2. The convergence of the algorithm is established in Theorem 3.2, whose proof is given in the Appendix.

Algorithm 2 ASGLD.

Input: Data $\{x_i\}_{i=1}^N$, subsample size n, smoothing factors β_1 and β_2, bias factor a, temperature τ, and learning rate sequence $\{\epsilon_t : t = 1, 2, \ldots\}$;
Initialization: θ_0 from appropriate distribution, $m_0 = 0$ and $V_0 = 0$;
for $t = 1, 2, \ldots,$ **do**
Draw a mini-batch of data $\{x_j^*\}_{j=1}^n$, and calculate
$\theta_{t+1} = \theta_t - \epsilon_{t+1}(\nabla\tilde{U}(\theta_t) + a m_t \oslash \sqrt{V_t + \lambda \mathbf{1}}) + e_{t+1}$,
$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla\tilde{U}(\theta_{t-1})$,
$V_t = \beta_2 V_{t-1} + (1-\beta_2)\nabla\tilde{U}(\theta_{t-1}) \odot \nabla\tilde{U}(\theta_{t-1})$,
where λ is a small constant added to avoid zero-divisors, $e_{t+1} \sim N(0, 2\tau\epsilon_{t+1} I_d)$, and d is the dimension of θ.
end for

Theorem 3.2: *Suppose that the conditions* (A.1)–(A.5) *hold (given in Appendix), and* $\beta_1^2 < \beta_2$ *are two constants between* 0 *and* 1. *For any smooth function* $\phi(\theta)$, *let* $\hat{\phi}_L = \frac{\sum_{k=1}^L \epsilon_k \phi(\theta_k)}{\sum_{k=1}^L \epsilon_k}$ *and* $\bar{\phi} = \int_\Theta \phi(\theta)\pi_*(\theta)\,\mathrm{d}\theta$, *where* $\pi_*(\theta)$ *denotes the target posterior distribution.*

(i) *If a constant learning rate of* ϵ *is used, then* $E\|\hat{\phi}_L - \bar{\phi}\|^2 = O(\epsilon^2)$ *as* $L \to \infty$.

(ii) *If the learning rate sequence* $\{\epsilon_k : k = 1, 2, \ldots\}$ *decays and satisfies the conditions that* $\sum_{k=1}^{\infty} \epsilon_k = \infty$ *and* $\lim_{L\to\infty} \frac{\sum_{k=1}^{L} \epsilon_k^2}{\sum_{k=1}^{L} \epsilon_k} = 0$, *then* $E\|\hat{\phi}_L - \bar{\phi}\|^2 \to 0$ *as* $L \to \infty$.

Compared to Algorithm 1, ASGLD contains one more parameter, β_2, which works as the smoothing factor for the second-moment term and is suggested to take a value of 0.999 in this paper.

3.3. Other adaptive SGLD algorithms

In addition to the Momentum and Adam algorithms, other optimization algorithms, such as AdaMax [17] and Adadelta [25], can also be incorporated into SGLD to accelerate the convergence of the simulation. Other than the drift term, the past gradients can also be used to construct an adaptive preconditioner matrix in a similar way to pSGLD [6] and SQNLMC [2]. Moreover, the adaptive drift and adaptive preconditioner matrix can be used together to accelerate the convergence of SGLD.

Rather than adapting the drift and/or preconditioner matrix, the contour SGLD algorithm [26] adapts the target density function in a dynamic importance sampling scheme as developed in [27]. For optimization tasks, the SANTA algorithm [28] extends further the SGNHT algorithm [8] with adaptive preconditioners and temperatures, where the preconditioner matrix is approximated by a diagonal matrix based on the current and past gradients as in pSGLD [6] and the temperature is updated in a simulated annealing scheme [29].

4. Illustrative examples

Before applying the adaptive SGLD algorithms to DNN models, we first illustrate their performance on two low-dimensional examples. The first example is a multi-modal distribution, which mimics the scenario with multiple local energy minima. The second example is more complicated, which mimics the scenario with pathological curvatures.

4.1. A multi-modal distribution

The target distribution is a 2-dimensional 5-component mixture Gaussian distribution, whose density function is given by $\pi(\theta) = \sum_{i=1}^{5} \frac{1}{10\pi} \exp(-\|\theta - \mu_i\|^2)$, where $\mu_1 = (-3, -3)^{\mathrm{T}}, \mu_2 = (-3, 0)^{\mathrm{T}}, \mu_3 = (0, 0)^{\mathrm{T}}, \mu_4 = (3, 0)^{\mathrm{T}}, \mu_5 = (3, 3)^{\mathrm{T}}$. For this example, we considered the natural gradient variational inference (NGVI) algorithm [30], which is known to converge very fast in the variational inference field, as the baseline algorithm for comparison.

Both MSGLD and ASGLD algorithms were applied to this example. We set $\nabla_\theta \tilde{U}(\theta) = \nabla U(\theta) + e$, where $e \sim N(0, I_2)$ and $U(\theta) = -\log \pi(\theta)$. For a fair comparison, each algorithm was run in 6.0 CPU minutes. The numerical results were summarized in Figure 1, which shows the contour of the energy function and its estimates by NGVI, MSGLD and ASGLD. The plots indicate that MSGLD and ASGLD are better at exploring the multi-modal distributions than NGVI.

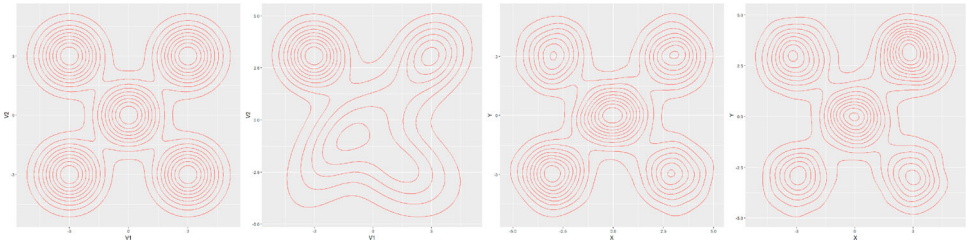

Figure 1. Contour plots of the true energy function (left) and its estimates by NGVI (middle left), MSGLD (middle right), and ASGLD (right).

4.2. A distribution with long narrow energy ravines

Consider a nonlinear regression

$$y = f_\theta(x) + \epsilon, \quad \epsilon \sim N(0,1),$$

where $x \sim \text{Unif}[-2,4]$, $\theta = (\theta_1,\theta_2)^{\mathrm{T}} \in \mathbb{R}^2$, and $f_\theta(x) = (x-1)^2 + 2\sin(\theta_1 x) + \frac{1}{30}\theta_1 + \cos(\theta_2 x - 1) - \frac{1}{20}\theta_2$. As θ increases, the function $f_\theta(x)$ fluctuates more severely. Figure 2(a) depicts the regression, where we set $\theta_1 = 20$ and $\theta_2 = 10$. Since the random error ϵ is relatively large compared to the local fluctuation of $f_\theta(x)$, i.e. $2\sin(\theta_1 x) + \frac{1}{30}\theta_1 + \cos(\theta_2 x - 1) - \frac{1}{20}\theta_2$, identification of the exact values of (θ_1,θ_2) can be very hard, especially when the mini-batch size n is small.

From this regression, we simulated 5 datasets with $(\theta_1,\theta_2) = (20,10)$ independently. Each dataset consisted of 10,000 samples. To conduct Bayesian analysis for the problem, we set the prior distribution: $\theta_1 \sim N(0,1)$ and $\theta_2 \sim N(0,1)$, which are *a priori* independent. This choice of the prior distribution makes the problem even harder, which discourages the convergence of the posterior simulation to the true value of θ. Instead, it encourages to estimate $f_\theta(x)$ by the global pattern $(x-1)^2$.

Both MSGLD and ASGLD were run for each of the five datasets. Each run consisted of 30,000 iterations, where the first 10,000 iterations were discarded for the burn-in process and the samples generated from the remaining 20,000 iterations were averaged as the

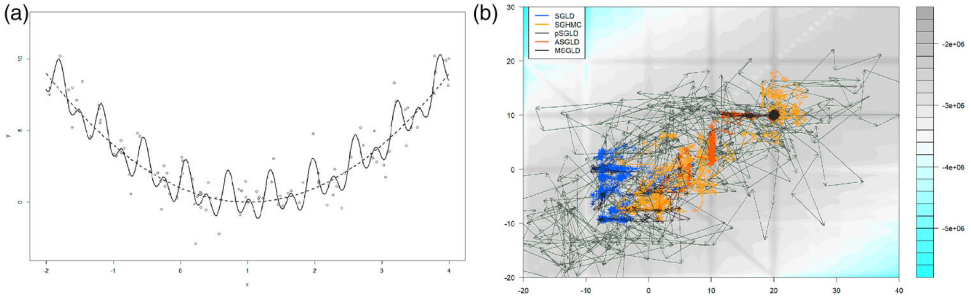

Figure 2. (a) The dashed line is for the global pattern $(x-1)^2$ and the solid line is for the regression function $f_\theta(x)$, where $\theta_1 = 20$ and $\theta_2 = 10$. The points represent 100 random samples from the regression. (b) Contour plot of the energy function for one dataset and the sample paths produced by SGLD, SGHMC, pSGLD, ASGLD and MSGLD in a run, where the sample paths have been thinned by a factor of 50 for readability of the plot.

Table 1. Bayesian estimates of θ produced by different algorithms for the long narrow energy ravines example in five independent runs, where the true value of θ is (20,10).

Method	θ	1	2	3	4	5
SGLD	θ_1	−5.79	−6.59	17.43	0.99	−3.01
	θ_2	−2.4	−1.76	9.02	−1.36	−4.74
SGHMC	θ_1	9.22	14.73	23.19	11.39	−4.22
	θ_2	1.27	6.23	11.03	2.65	−2.25
pSGLD	θ_1	1.27	−19.77	16.52	5.14	−9.08
	θ_2	−2.22	−15.44	8.17	0.65	3.75
ASGLD	θ_1	19.75	15.34	19.99	19.01	19.98
	θ_2	9.99	9.29	9.92	9.66	9.99
MSGLD	θ_1	20.01	20.07	19.99	20.00	19.99
	θ_2	9.99	10.00	9.99	9.99	9.99

Bayesian estimate of θ. In the simulations, we set the mini-batch size $n = 100$. The settings of other parameters are given in the Appendix. Table 1 shows the Bayesian estimates of θ produced by the two algorithms in each of the five runs. The MSGLD estimates converged to the true value in all five runs, while the ASGLD estimates converged to the true value in four of five runs. For comparison, SGLD, SGHMC and pSGLD were also applied to this example with the settings given in the Appendix. As implied by Table 1, all the three algorithms essentially failed for this example: none of their estimates converged to the true value!

For further exploration, Figure 2(b) shows the contour plot of the energy function for one dataset and the sample paths produced by SGLD, SGHMC, pSGLD, ASGLD and MSGLD for the dataset. As shown by the contour plot, the energy landscape contains multiple long narrow ravines, which make the existing SGMCMC algorithms essentially fail. However, due to the use of momentum information, ASGLD and MSGLD work extremely well for this example. As indicated by their sample paths, they can move along narrow ravines, and converge to the true value of θ very quickly. It is interesting to point out that pSGLD does not work well for this example, although it has used the past gradients in constructing the preconditioned matrix. A possible reason for this failure is that it only approximates the preconditioned matrix by a diagonal matrix and missed the correlation between different components of θ.

5. DNN applications

5.1. Landsat data

The dataset is available at the UCI Machine Learning Repository, which consists of 4435 training instances and 2000 testing instances. Each instance consists of 36 attributes that represent features of the earth surface images taken from a satellite. The training instances consist of six classes, and the goal of this study is to learn a classifier for the earth surface images.

We modelled this dataset by a fully connected DNN with structure 36-30-30-6 and *Relu* as the activation function. Let θ denote the vector of all parameters (i.e. connection weights) of the DNN. We let θ be subject to a Gaussian prior distribution: $\theta \sim N(0, I_d)$, where d is the dimension of θ. The SGLD, SGHMC, pSGLD, ASGLD and MSGLD algorithms were all applied to simulate from the posterior of the DNN. Each algorithm was run

for 3000 epochs with the mini-batch size $n = 50$ and a decaying learning rate

$$\epsilon_k = \epsilon_0 \gamma^{\lfloor k/L \rfloor}, \tag{4}$$

where k indexes epochs, the initial learning rate $\epsilon_0 = 0.1$, $\gamma = 0.5$, the step size $L = 300$, and $\lfloor z \rfloor$ denotes the maximum integer less than z. For the purpose of optimization, the temperature was set to $\tau = 0.01$. The settings for the specific parameters of each algorithm were given in the Appendix.

Each algorithm was run for five times for the example. In each run, the training and test classification accuracy were calculated by averaging over the last 200 samples, which were collected from the last 100,000 iterations with a thinning factor of 500. For each algorithm, Table 2 reports the mean classification accuracy, for both training and test, averaged over five runs and its standard deviation. The results indicate that MSGLD has significantly outperforms other algorithms in both training and test for this example. While ASGLD performs similarly to pSGLD for this example.

Finally, we note that for this example, the SGMCMC algorithms have been run excessively long. Figure 3(a) and Figure 3(b) show, respectively, the training and test classification errors produced by SGLD, pSGLD, SGHMC, ASGLD and MSGLD along with iterations. It indicates again that MSGLD significantly outperforms other algorithms in both training and test.

Table 2. Training and test classification accuracy produced by different SGMCMC algorithms for the Landsat data, where the accuracy and its standard error were calculated based on 5 independent runs.

Method	Training accuracy	Test accuracy
SGLD	93.163 ± 0.085	90.225 ± 0.153
pSGLD	93.857 ± 0.125	90.712 ± 0.090
SGHMC	94.015 ± 0.117	90.848 ± 0.089
ASGLD	93.827 ± 0.087	90.794 ± 0.087
MSGLD	**94.910** ± 0.105	**91.247** ± 0.141

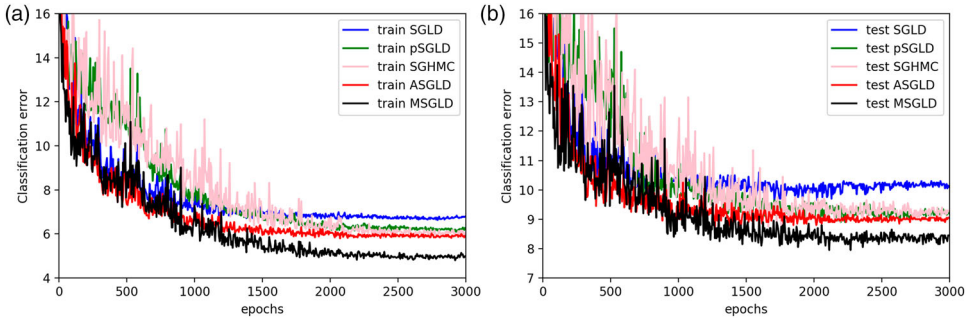

Figure 3. (a) Training classification errors produced by different SGMCMC algorithms for the Landsat data example. (b) Test classification errors produced by different SGMCMC algorithms for the Landsat data example.

5.2. MNIST data

The MNIST is a benchmark dataset of computer vision, which consists of 60,000 training instances and 10,000 test instances. Each instance is an image consisting of 28×28 attributes and representing a hand-written number of 0 to 9. For this data set, we tested whether ASGLD or MSGLD can be used to train sparse DNNs. For this purpose, we considered a mixture Gaussian prior for each of the connection weights:

$$\pi(\theta_k) \sim \lambda_k N(0, \sigma_{1,k}^2) + (1 - \lambda_k) N(0, \sigma_{0,k}^2), \tag{5}$$

where k is the index of hidden layers, and $\sigma_{0,k}$ is a relatively very small value compared to $\sigma_{1,k}$. In this paper, we set $\lambda_k = 10^{-7}$, $\sigma_{1,k}^2 = 0.02, \sigma_{0,k}^2 = 1 \times 10^{-5}$ for all k.

We trained a DNN with structure 784-800-800-10 using ASGLD and MSGLD for 250 epochs with subsample size 100. For the first 100 epochs, the DNN was trained as usual, i.e. with a Gaussian prior $N(0, I_d)$ imposed on each connection weight. Then the DNN was trained for 150 epochs with the prior (5). The settings for the other parameters of the algorithms were given in the Appendix. Each algorithm was run for 3 times. In each run, the training and prediction accuracy were calculated by averaging, respectively, the fitting and prediction results over the iterations of the last 5 epochs. The numerical results were summarized in Table 3, where 'Sparse ratio' is calculated as the percentage of the learned connection weights satisfying the inequality $|\theta_k| \geq \sqrt{\log\left(\frac{1-\lambda_k}{\lambda_k}\frac{\sigma_{1k}}{\sigma_{0k}}\right)\frac{2\sigma_{0k}^2\sigma_{1k}^2}{\sigma_{1k}^2-\sigma_{0k}^2}}$. The threshold is determined by solving the probability inequality $P\{\theta_k \sim N(0, \sigma_{0,k}^2) \mid \theta_k\} \leq P\{\theta_k \sim N(0, \sigma_{1,k}^2) \mid \theta_k\}$.

Adam is a well-known DNN optimization method for MNIST data. For comparison, it was also applied to train the sparse DNN. Interestingly, ASGLD outperforms Adam in both training and prediction accuracy, although its resulting network is a little more dense than that by Adam.

5.3. CIFAR 10 and CIFAR 100

The CIFAR-10 and CIFAR-100 are also benchmark datasets of computer vision. CIFAR-10 consists of 50,000 training images and 10,000 test images, and the images were classified into 10 classes. The CIFAR-100 dataset consists of 50,000 training images and 10,000 test images, but the images were classified into 100 classes. We modelled both datasets using a ResNet-18 [31], and trained the model for 250 epochs using various optimization and SGMCMC algorithms, including SGD, Adam, SGLD, SGHMC, pSGLD, ASGLD and MSGLD with data augmentation techniques. The temperature τ was set to 1.0e−5 for all

Table 3. Comparison of different algorithms for training sparse DNNs for the MNIST data, where the reported results for each algorithm were averaged based on three independent runs.

Method	Training accuracy	Test accuracy	Sparsity ratio
ASGLD	**99.542** $\pm$ 0.026	**98.417** $\pm$ 0.044	3.176 $\pm$ 0.155
MSGLD	99.494 $\pm$ 0.002	98.319 $\pm$ 0.019	3.369 $\pm$ 0.063
ADAM	99.383 $\pm$ 0.072	98.332 $\pm$ 0.003	2.169 $\pm$ 0.013

Table 4. Mean training and test classification accuracy (averaged over 5 runs) of the Bayesian ResNet-18 for CIFAR-10 and CIFAR-100 data.

	CIFAR-10		CIFAR-100	
Method	Training	Test	Training	Test
SGD	99.721 ± 0.058	92.896 ± 0.141	98.211 ± 0.208	72.274 ± 0.145
ADAM	99.515 ± 0.087	92.144 ± 0.154	96.744 ± 0.390	67.564 ± 0.501
SGLD	99.713 ± 0.034	92.908 ± 0.114	97.650 ± 0.131	71.908 ± 0.149
pSGLD	97.209 ± 0.465	89.398 ± 0.379	99.034 ± 0.101	69.234 ± 0.162
SGHMC	99.152 ± 0.060	93.470 ± 0.073	96.92 ± 0.091	73.112 ± 0.139
ASGLD	99.738 ± 0.052	93.018 ± 0.187	96.938 ± 0.268	73.252 ± 0.681
MSGLD	99.702 ± 0.066	**93.512** ± 0.081	96.801 ± 0.150	**73.670** ± 0.144

SGMCMC methods. The subsample sample size was set to 100. For CIFAR-10, the learning rate was set as in (4) with $L = 40$ and $\gamma = 0.5$; and the weight prior was set to $N(0, \frac{1}{25}I_d)$ for all SGMCMC algorithms. For CIFAR-100, the learning rate was set as in (4) with $L = 90$ and $\gamma = 0.1$; and the weight prior was set to $N(0, \frac{1}{25}I_d)$ for SGLD, pSGLD, and SGHMC, and $N(0, \frac{1}{75}I_d)$ for ASGLD and MSGLD. For SGD and Adam, we set the objective function as

$$-\frac{1}{n}\sum_{i=1}^{n}\log f(x_i \mid \theta) + \frac{1}{2}\lambda\|\theta\|^2, \tag{6}$$

where $f(x_i \mid \theta)$ denotes the likelihood function of observation i, and λ is the regularization parameter. For SGD, we set $\lambda = 5.0\text{e} - 4$; and for Adam we set $\lambda = 0$. For Adam, we have also tried the case $\lambda \neq 0$, but the results were inferior to those reported below.

For each dataset, each algorithm was run for five times. In each run, the training and test classification errors were calculated by averaging over the iterations of the last five epochs. Table 4 reported the mean training and test classification accuracy (averaged over 5 runs) of the Bayesian ResNet-18 for CIFAR-10 and CIFAR-100. The comparison indicates that MSGLD outperforms all other algorithms in test accuracy for the two datasets. In terms of training accuracy, ASGLD and MSGLD work more like the existing momentum-based algorithms such as ADAM and SGHMC, but tend to generalize better than them. This result has been very remarkable, as all algorithms were run with a small number of epochs; that is, the Monte Carlo algorithms are not necessarily slower than the stochastic optimization algorithms in deep learning.

6. Conclusion

This paper has developed a class of adaptive SGMCMC algorithms by including a bias term to the drift of SGLD, where the bias term is allowed to be adaptively adjusted with past samples, past gradients, etc. The numerical results indicate that the proposed algorithms have inherited many attractive properties, such as quick convergence in the scenarios with pathological curvatures from their counterpart optimization algorithms, while ensuring more extensive exploration of the sample space than the optimization algorithms due to their simulation nature. As a result, the proposed algorithms can outperform the existing SGMCMC algorithms in both simulation and optimization.

The performance of the proposed algorithms can be further improved by incorporating other advanced SGMCMC techniques into simulations. For example, cyclical SGMCMC [32] employed a cyclical step size schedule, where large step sizes are used for discovering new modes, and small step sizes are used for characterizing each mode. This technique can be easily incorporated into the adaptive SGLD algorithms to improve their convergence to the stationary distribution.

Finally, we would mention that other than the SGLD algorithm and its variants, there exist other classes of MCMC algorithms that make use of subsamples in Bayesian simulations for big data problems. They include the split-and-merge algorithms [33–35], mini-batch Metropolis–Hastings algorithms [36–40], and some algorithms based on non-reversible Markov process [41–43]. These algorithms have achieved great successes for some low-dimensional statistical models, but are rarely used for high-dimensional DNN models. A further exploration for the performance of these algorithms in large-scale machine learning models is of great interest.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by Division of Mathematical Sciences [grant numbers DMS-1811812, DMS-2015498] and the National Institute of General Medical Sciences [grant numbers R01-GM117597, R01-GM126089].

References

[1] Welling M, Teh YW.. Bayesian learning via stochastic gradient Langevin dynamics. Bellevue, Washington, USA; Proceedings of the 28th International Conference on International Conference on Machine Learning, 2011.

[2] Simsekli U, Badeau R, Cemgil T, et al. Stochastic quasi-Newton Langevin Monte Carlo. In: Balcan MF, Weinberger KQ, editors. Proceedings of The 33rd International Conference on Machine Learning; Jun 20–22; New York, NY, USA. PMLR; 2016. p. 642–651. (Proceedings of Machine Learning Research; vol. 48).

[3] Ahn S, Balan AK, Welling M. Bayesian Posterior Sampling via Stochastic Gradient Fisher Scoring, Edinburgh Scotland, Proceedings of the 31st International Conference on Machine Learning, 2012.

[4] Girolami M, Calderhead B. Riemann manifold Langevin and Hamiltonian Monte Carlo methods (with discussion). J R Stat Soc Ser B. 2011;73(2):123–214.

[5] Patterson S, Teh YW. Stochastic Gradient Riemannian Langevin Dynamics on the Probability Simplex. Lake Tahoe Nevada. Advances in Neural Information Processing Systems. 2013.

[6] Li C, Chen C, Carlson DE. Preconditioned Stochastic Gradient Langevin Dynamics for deep neural networks. Phoenix Arizona. AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. 2016.

[7] Chen T, Fox EB, Guestrin C. Stochastic Gradient Hamiltonian Monte Carlo. Bejing, China. Proceedings of the 31st International Conference on Machine Learning. 2014.

[8] Ding N, Fang Y, Babbush R, et al. Bayesian Sampling Using Stochastic Gradient Thermostats. Montreal Canada. Advances in Neural Information Processing Systems. 2014.

[9] Lu X, Perrone V, Hasenclever L, et al. Relativistic Monte Carlo. In: Singh A, Zhu J, editors. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics;

April 20–22; Fort Lauderdale, FL, USA. PMLR; 2017. p. 1236–1245. (Proceedings of Machine Learning Research; vol. 54).
[10] Ma YA, Chen T, Fox EB. A Complete Recipe for Stochastic Gradient MCMC. Montreal, Canada. Advances in Neural Information Processing Systems. 2015.
[11] Dalalyan AS, Karagulyan AG. User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. CoRR; 2017. Available from: arXiv:1710.00095.
[12] Song Q, Sun Y, Ye M, et al. Extended stochastic gradient MCMC for large-scale Bayesian variable selection. Preprint; 2020. Available from: arXiv:2002.02919v1.
[13] Bhatia K, Ma YA, Dragan AD, et al. Bayesian robustness: a nonasymptotic viewpoint. Preprint; 2019. Available from: arXiv:190711826.
[14] Qian N. On the momentum term in gradient descent learning algorithms. Neural Netw. 1999;12(1):145–151.
[15] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res. 2011;12:2121–2159.
[16] Tieleman T, Hinton G. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: Neural Netw Mach Learn. 2012;4(2):26–31.
[17] Kingma D, Ba J. Adam: a Method for Stochastic Optimization. San Diego, CA, USA. 3rd International Conference on Learning Representation. 2015.
[18] Staib M, Reddi S, Kale S, et al. Escaping Saddle Points with Adaptive Gradient Methods. Long Beach, California. Proceedings of the 36th International Conference on Machine Learning. 2019.
[19] Ruder S. An overview of gradient descent optimization algorithms. CoRR; 2016. Available from arXiv:abs/1609.04747.
[20] Sato I, Nakagawa H. Approximation Analysis of Stochastic Gradient Langevin Dynamics by using Fokker-Planck Equation and Ito Process. Bejing, China. Proceedings of the 31st International Conference on Machine Learning. 2014.
[21] Teh YW, Thiery AH, Vollmer SJ. Consistency and fluctuations for stochastic gradient Langevin dynamics. J Mach Learn Res. 2016;17(1):193–225.
[22] Nemeth C, Fearnhead P. Stochastic gradient Markov chain Monte Carlo. Preprint; 2019. Available from: arXiv:190706986.
[23] Sutton RS. Two problems with backpropagation and other steepest-descent learning procedures for networks. In: Proceedings of the 8th Annual Conference of the Cognitive Science Society. Hillsdale (NJ): Erlbaum; 1986.
[24] Nagapetyan T, Duncan A, Hasenclever L, et al. The true cost of SGLD. Preprint; 2017. Available from: arXiv:170602692v1.
[25] Zeiler MD. ADADELTA: an adaptive learning rate method. CoRR; 2012. Available from: arXiv:abs/1212.5701.
[26] Deng W, Lin G, Liang F. A Contour Stochastic Gradient Langevin Dynamics Algorithm for Simulations of Multi-modal Distributions. Virtual. The 34st Conference on Neural Information Processing System. 2020.
[27] Liang F, Liu C, Carroll R. Stochastic approximation in Monte Carlo computation. J Am Stat Assoc. 2007;102:305–320.
[28] Chen C, Carlson D, Gan Z, et al. Bridging the gap between stochastic gradient MCMC and stochastic optimization. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS); Cadiz, Spain; 2016.
[29] Kirkpatrick S, Gelatt Jr C, Vecchi M. Optimization by simulated annealing. Sciences. 1983;220(4598):671–680.
[30] Lin W, Khan ME, Schmidt M. Fast and simple natural-gradient variational inference with mixture of exponential-family approximations. In: ICML. PMLR; 2019. (Proceedings of Machine Learning Research).
[31] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. CVPR; 2015.
[32] Zhang R, Li C, Zhang J, et al. Cyclical stochastic gradient MCMC for Bayesian deep learning. Preprint; 2019. Available from: arXiv:190203932.

[33] Scott SL, Blocker AW, Bonassi FV, et al. Bayes and big data: the consensus Monte Carlo algorithm. Int J Manag Sci Eng Manag. 2016;11(2):78–88.
[34] Srivastava S, Li C, Dunson DB. Scalable Bayes via Barycenter in Wasserstein space. J Mach Learn Res. 2018;19:1–35.
[35] Xue J, Liang F. Double-parallel Monte Carlo for Bayesian analysis of big data. Stat Comput. 2019;29:23–32.
[36] Chen H, Seita D, Pan X, et al. An efficient minibatch acceptance test for Metropolis-Hastings. Preprint; 2016. Available from: arXiv:161006848.
[37] Korattikara A, Chen Y, Welling M. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. Bejing, China. Proceedings of the 31st International Conference on Machine Learning. 2014.
[38] Bardenet R, Doucet A, Holmes C. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. Bejing, China. Proceedings of the 31st International Conference on Machine Learning. 2014.
[39] Maclaurin D, Adams RP. Firefly Monte Carlo: Exact MCMC with Subsets of Data. Quebec City, Quebec, Canada. Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence. 2014.
[40] Bardenet R, Doucet A, Holmes CC. On Markov chain Monte Carlo methods for tall data. J Mach Learn Res. 2017;18(47):1–47. 43.
[41] Bierkens J, Fearnhead P, Roberts G. The zig–zag process and super-efficient Monte Carlo for Bayesian analysis of big data. Ann Stat. 2019;47(3):1288–1320.
[42] Bouchard Coté A, Vollmer S, Doucet A. The bouncy particle sampler: a nonreversible rejection-free Markov chain Monte Carlo method. J Am Stat Assoc. 2018;113:855–867.
[43] Pakman A, Gilboa D, Carlson D, et al. Stochastic bouncy particle sampler. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning; Aug 6–11. PMLR; 2017. p. 2741–2750. (Proceedings of Machine Learning Research; vol. 70).
[44] Vollmer SJ, Zygalakis KC, Teh YW. Exploration of the (non-)asymptotic bias and variance of stochastic gradient Langevin dynamics. J Mach Learn Res. 2016;17(159):1–48.
[45] Chen C, Ding N, Carin L. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. Montreal, Canada. Proceedings of the 28th International Conference on Neural Information Processing Systems. 2015.
[46] Raginsky M, Rakhlin A, Telgarsky M. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. Proc Mach Learn Res. 2017;65:1–30.
[47] Xu P, Chen J, Zou D, et al. Global convergence of Langevin dynamics based algorithms for nonconvex optimization. Red Hook, NY, United States. Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2015.
[48] Mattingly J, Stuartb A, Highamc D. Ergodicity for SDEs and approximations: locally Lipschitz vector fields and degenerate noise. Stoch Process Appl. 2002;101:185–232.
[49] Deng W, Zhang X, Liang F, et al. An adaptive empirical Bayesian method for sparse deep learning. Vancouver, Canada. 33rd Conference on Neural Information Processing System. 2019.

Appendices

Appendix 1. Proofs of Theorems 3.1 and 3.2

Without loss of generality and for notational simplicity, we assume in the proof that the temperature τ in both the MSGLD and ASGLD algorithms takes a value of 1. Consider a generalized SGLD algorithm with a biased drift term for simulating from the target distribution $\pi_*(\theta) \propto \exp\{-U(\theta)\}$. Let θ_{k+1} and θ_k be two random vectors in Θ satisfying

$$\theta_{k+1} = \theta_k - \epsilon_{k+1}[\nabla U(\theta_k) + \zeta_{k+1}] + \sqrt{2\epsilon_{k+1}}e_{k+1}, \tag{A1}$$

where $e_{k+1} \sim N(0, I_p)$, and $\zeta_{k+1} = \nabla\widehat{U}(\theta_k) - \nabla U(\theta_k)$ denotes deviation between the drift $\nabla\widehat{U}(\theta_k)$ used in simulations and the ideal drift $\nabla U(\theta_k) = -\nabla \log \pi_*(\theta_k)$. For example, in Equation (2), we have $\nabla\widehat{U}(\theta_k) = \nabla_\theta \widetilde{U}(\theta_k) + aA_k$.

For the generalized SGLD algorithm (A1), we aim to analyse the deviation of the averaging estimate $\hat{\phi}_L = \frac{\sum_{k=1}^L \epsilon_k \phi(\theta_k)}{\sum_{k=1}^L \epsilon_k}$ from the posterior mean $\bar{\phi} = \int_\Theta \phi(\theta)\pi_*(d\theta)$ for a bounded smooth function $\phi(\theta)$ of interest. The key tool we employed in the analysis is the Poisson equation which is used to characterize the fluctuation between ϕ and $\bar{\phi}$:

$$\mathcal{L}g(\theta) = \phi(\theta) - \bar{\phi}, \tag{A2}$$

where $g(\theta)$ is the solution to the Poisson equation, and $\mathcal{L}$ is the infinitesimal generator of the Langevin diffusion

$$\mathcal{L}g := \langle \nabla g, \nabla U(\cdot)\rangle + \tau \Delta g.$$

By imposing the following regularity conditions on the function $g(\theta)$, we can control the fluctuation of $\hat{\phi}_L - \bar{\phi}$, which enables convergence of the sample average.

(A.1) Given a sufficiently smooth function $g(\theta)$ as defined in (A2) and a function $\mathcal{V}(\theta)$ such that the derivatives satisfy the inequality $\|D^j g\| \lesssim \mathcal{V}^{p_j}(\theta)$ for some constant $p_j > 0$, where $j \in \{0, 1, 2, 3\}$. In addition, $\mathcal{V}^p$ has a bounded expectation, i.e. $\sup_k E[\mathcal{V}^p(\theta_k)] < \infty$; and $\mathcal{V}^p$ is smooth, i.e. $\sup_{s\in(0,1)} \mathcal{V}^p(s\theta + (1-s)\vartheta) \lesssim \mathcal{V}^p(\theta) + \mathcal{V}^p(\vartheta)$ for all $\theta, \vartheta \in \Theta$ and $p \le 2\max_j\{p_j\}$.

For a stronger but verifiable version of the condition, we refer readers to Vollmer et al. [44]. In what follows, we present a lemma which is adapted from Theorem 3 of [45] with a fixed learning rate ϵ. Note that [45] requires $\{\zeta_k : k = 1, 2, \ldots\}$ to be a zero mean sequence, while in our case $\{\zeta_k : k = 1, 2, \ldots\}$ forms an auto-regressive sequence which makes the proof of Chen et al. [45] still go through.

Lemma A.1: *Assume the condition* (A.1) *hold and a constant learning rate ϵ is used. For a smooth function ϕ, the mean square error (MSE) of the generalized SGLD algorithm* (A1) *at time $S_L = \epsilon L$ is bounded as*

$$E\|\hat{\phi}_L - \bar{\phi}\|^2 \le C\left(\frac{1}{L^2}\sum_{k=1}^L E\|\zeta_k\|^2 + \frac{1}{L\epsilon} + \epsilon^2\right), \tag{A3}$$

for some constant C.

Lemma A.2 is established for a decaying learning rate sequence. Refer to Theorem 5 of [45] for the proof.

Lemma A.2: *Assume the condition* (A.1) *hold, and the learning rate ϵ_k is decreasing and satisfies the conditions that $\sum_{k=1}^\infty \epsilon_k = \infty$ and $\lim_{L\to\infty} \frac{\sum_{k=1}^L \epsilon_k^2}{\sum_{k=1}^L \epsilon_k} = 0$. For a smooth function ϕ, the mean square error (MSE) of the generalized SGLD algorithm* (A1) *at time $S_L = \sum_{k=1}^L \epsilon_k$ is bounded as*

$$E\|\hat{\phi}_L - \bar{\phi}\|^2 \le C\left(\sum_{k=1}^L \frac{\epsilon_k^2}{S_L^2} E\|\zeta_k\|^2 + \frac{1}{S_L} + \frac{(\sum_{k=1}^L \epsilon_k^2)^2}{S_L^2}\right), \tag{A4}$$

for some constant C.

To prove Theorems 3.1 and 3.2, we further make the following assumptions:

(A.2) (smoothness) $U(\theta)$ is M-smooth; that is, there exists a constant $M > 0$ such that for any $\theta, \theta' \in \Theta$,

$$\|\nabla U(\theta) - \nabla U(\theta')\| \le M\|\theta - \theta'\|. \tag{A5}$$

The smoothness of $\nabla U(\theta)$ is a standard assumption in studying the convergence of SGLD, and it has been used in a few work, see, e.g. Raginsky et al. [46] and Xu et al. [47].

(A.3) (Dissipativity) There exist constants $m > 0$ and $b \geq 0$ such that for any $\theta \in \Theta$,

$$\langle \nabla U(\theta), \theta \rangle \geq m\|\theta\|^2 - b. \tag{A6}$$

This assumption has been widely used in proving the geometric ergodicity of dynamical systems [46–48]. It ensures the sampler to move towards the origin regardless the position of the current point.

(A.4) (Gradient noise) The stochastic gradient $\xi(\theta) = \nabla\widetilde{U}(\theta) - \nabla U(\theta)$ is unbiased; that is, for any $\theta \in \Theta$, $E[\xi(\theta)] = 0$. In addition, there exists some constant $B > 0$ such that the second moment of the stochastic gradient is bounded by $E\|\xi(\theta)\|^2 \leq M^2\|\theta\|^2 + B^2$, where the expectation is taken with respect to the distribution of the gradient noise.

Lemma A.3 (Uniform L^2 bound): *Assume the conditions* (A.2)–(A.4) *hold. For any learning rate sequence with* $0 < \epsilon_1 < Re(\frac{m-\sqrt{m^2-4M^2(M^2+1)}}{4M^2(M^2+1)})$, *there exists a constant* $G > 0$ *such that* $E\|\theta_k\|^2 \leq G$, *where* $G = \|\theta_0\|^2 + \frac{1}{m}(b + 2\epsilon_1 B^2(M^2+1) + p)$, p *denotes the dimension of* θ, *and* $Re(\cdot)$ *denotes the real part of a complex number.*

Proof: The proof follows that of Lemma 1 in Deng et al. [49]. To make use of the proof in Deng et al. [49], we can rewrite Equation (A1) as

$$\theta_{k+1} = \theta_k - \epsilon_{k+1}\nabla_\theta L(\theta_k, \zeta_{k+1}) + \sqrt{2\epsilon_{k+1}}e_{k+1},$$

where $\nabla_\theta L(\theta_k, \zeta_{k+1}) = [\nabla_\theta U(\theta_k) + \zeta_{k+1}]$ by viewing ζ_{k+1} as an argument of the function $L(\cdot,\cdot)$. Then, it is easy to verify that the conditions (A.2)–(A.4) imply the conditions of Lemma 1 of Deng et al. [49], and thus the uniform L^2 bound holds. Note that given the condition (A.3), the inequality $E\langle \nabla_\theta L(\theta_k, \zeta_{k+1}), \theta_k\rangle \geq mE\|\theta_k\|^2 - b$, required by Deng et al. [49] in its proof, will hold as long as ϵ_1 is sufficiently small or β_1 is not very close to 1. ■

Let θ_* denote the minimizer of $U(\theta)$. Therefore, $\nabla U(\theta_*) = 0$. Then, by Lemma A.3 and condition (A.2), there exists a constant C_1 such that

$$E\|\nabla U(\theta_k)\|^2 \leq 2M^2(G + \|\theta_*\|^2) := C_1 < \infty. \tag{A7}$$

Let $\xi_{k+1} := \nabla\tilde{U}(\theta_k) - \nabla U(\theta_k)$ be the gradient estimation error. By Lemma A.3 and condition (A.4), there exists a constant C_2 such that

$$E(\|\xi_k\|^2) \leq M^2 G + B^2 := C_2 < \infty. \tag{A8}$$

A.1 Proof of Theorem 3.1

Proof: The update of the MSGLD algorithm can be rewritten as $\theta_{k+1} = \theta_k - \epsilon_{k+1}[\nabla U(\theta_k) + \zeta_{k+1}] + \sqrt{2\epsilon_{k+1}}e_{k+1}$, where $\zeta_{k+1} = am_k + \xi_{k+1}$ and $m_0 = 0$.

First, we study the bias of ζ_k. According to the recursive update rule of m_i, we have

$$\begin{aligned} E(\zeta_{k+1} \mid \mathcal{F}_k)/a &= E(m_k \mid \mathcal{F}_k) = (1-\beta_1)\nabla U(\theta_{k-1}) + \beta_1 E(m_{k-1} \mid \mathcal{F}_k) \\ &= (1-\beta_1)\nabla U(\theta_{k-1}) + (1-\beta_1)\beta_1\nabla U(\theta_{k-2}) + \beta_1^2 E(m_{k-2} \mid \mathcal{F}_k) = \cdots \\ &= \sum_{i=1}^{k}(1-\beta_1)\beta_1^{i-1}\nabla U(\theta_{k-i}) + \beta_1^k E(m_0 \mid \mathcal{F}_k) = \sum_{i=1}^{k}(1-\beta_1)\beta_1^{i-1}\nabla U(\theta_{k-i}). \end{aligned}$$

Hence, by Jensen's inequality

$$\|E(\zeta_{k+1}\,|\,\mathcal{F}_k)\| \le a\sum_{i=1}^{k}(1-\beta_1)\beta_1^{i-1}\|\nabla U(\theta_{k-i})\| \le a\sqrt{\sum_{i=1}^{k}(1-\beta_1)\beta_1^{i-1}\|\nabla U(\theta_{k-i})\|^2}.$$

By (A7), the bias is further bounded by

$$E\|E(\zeta_{k+1}\,|\,\mathcal{F}_k)\|^2 \le a^2\sum_{i=1}^{k}(1-\beta_1)\beta_1^{i-1}E\|\nabla U(\theta_{k-i})\|^2 \le a^2C_1. \tag{A9}$$

For the variance of ζ_{k+1}, we have

$$\begin{aligned}
&E\|\zeta_{k+1}-E(\zeta_{k+1}\,|\,\mathcal{F}_k)\|^2 = E\|\xi_{k+1}+am_k-E(am_k\,|\,\mathcal{F}_k)\|^2\\
&\quad= E\|\xi_{k+1}+a(1-\beta_1)\tilde{\nabla}U(\theta_{k-1})+a\beta_1m_{k-1}-a(1-\beta_1)\nabla U(\theta_{k-1})-a\beta_1E(m_{k-1}\,|\,\mathcal{F}_k)\|^2\\
&\quad= E\|\xi_{k+1}+a(1-\beta_1)\xi_k+a\beta_1m_{k-1}-a\beta_1E(m_{k-1}\,|\,\mathcal{F}_k)\|^2=\cdots\\
&\quad= E\|\xi_{k+1}+\sum_{i=1}^{k}a(1-\beta_1)\beta_1^{i-1}\xi_{k-i+1}\|^2.
\end{aligned}$$

Due to the independence among ξ_k's, we have

$$\begin{aligned}
E\|\zeta_{k+1}-E(\zeta_{k+1}\,|\,\mathcal{F}_k)\|^2 &\le E\|\xi_{k+1}\|^2+\sum_{i=1}^{k}a^2(1-\beta_1)^2\beta_1^{2i-2}E\|\xi_{k-i+1}\|^2\\
&\le C_2[1+a^2(1-\beta_1)/(1+\beta_1)],
\end{aligned} \tag{A10}$$

where the last inequality follows from (A8).

Combining (A9) and (A10), we have

$$E\|\zeta_{k+1}\|^2 \le a^2C_1+C_2[1+a^2(1-\beta_1)/(1+\beta_1)]<\infty,$$

which conclude the proof by applying Lemmas A.1 and A.2. ■

A.2 Proof of Theorem 3.2

Proof: The update of the ASGLD algorithm can be rewritten as $\theta_{k+1}=\theta_k-\epsilon_{k+1}[\nabla U(\theta_k)+\zeta_{k+1}]+\sqrt{2\epsilon_{k+1}}e_{k+1}$, where $\zeta_{k+1}=am_k\oslash\sqrt{v_k+\lambda\mathbf{1}}+\xi_{k+1}$.

According to the recursive update rule of m_i and v_i, we have

$$\begin{aligned}
m_i &= (1-\beta_1)\tilde{U}(\theta_{i-1})+(1-\beta_1)\beta_1\tilde{U}(\theta_{i-2})+(1-\beta_1)\beta_1^2\tilde{U}(\theta_{i-3})+\cdots,\\
v_i &= (1-\beta_2)\tilde{U}(\theta_{i-1})\odot\tilde{U}(\theta_{i-1})+(1-\beta_2)\beta_2\tilde{U}(\theta_{i-2})\odot\tilde{U}(\theta_{i-2})\\
&\quad+(1-\beta_2)\beta_2^2\tilde{U}(\theta_{i-3})\odot\tilde{U}(\theta_{i-3})+\cdots.
\end{aligned}$$

Therefore, by Cauchy–Schwarz inequality, when $\beta_1^2<\beta_2$, we have

$$\|m_{i-1}\oslash\sqrt{v_{i-1}}\|_\infty \le \sqrt{\sum_{j=1}^{i-1}\frac{(1-\beta_1)^2\beta_1^{2j-2}}{(1-\beta_2)\beta_2^{j-1}}} \le \sqrt{\frac{(1-\beta_1)^2}{1-\beta_2}\frac{1}{1-\beta_1^2/\beta_2}} := C.$$

It implies that $\|m_{i-1}\oslash\sqrt{v_{i-1}+\lambda\mathbf{1}}\|\le\sqrt{p}C$ almost surely, and in consequence,

$$E\|E(\zeta_{k+1}\,|\,\mathcal{F}_k)\|^2 \le a^2C^2p, \tag{A11}$$

and, by (A8),

$$E\|\zeta_{k+1}-E(\zeta_{k+1}\,|\,\mathcal{F}_k)\|^2 \le E\|\zeta_{k+1}\|^2 \le E\|\xi_{k+1}\|^2+a^2E\|m_k\oslash\sqrt{v_k+\lambda\mathbf{1}}\|^2$$

$$\leq a^2C^2p + C_2. \tag{A12}$$

Combining (A11) and (A12), we have

$$E\|\zeta_{k+1}\|^2 \leq 2a^2C^2p + C_2 < \infty,$$

which concludes the proof by applying Lemmas A.1 and A.2. ■

Appendix 2. Experimental setup

All numerical experiments on deep learning were done with pytorch. For all SGMCMC algorithms, the initial learning rates were set at the order of $O(1/N)$ in all experiments except for in MNIST training. For the optimization methods such as SGD and Adam, the objective function was set to (6), where $f(x_i \mid \theta)$ denotes the likelihood function of observation i, and λ is the regularization parameter whose value varies for different datasets.

A.3 Multi-modal distribution

For NGVI method, we chose a 5-component mixture Gaussian distribution, and set the initial parameters as $\pi_i = \frac{1}{5}$, $\mu_i \sim N(0_2, 1.5I_2)$, $\Sigma_i = I_2$ for $i = 1, \ldots, 5$, and the learning rate $\epsilon = 0.005$. For MSGLD, we set $(a, \beta_1) = (10, 0.9)$ and the learning rate $\epsilon = 0.05$. For Adam SGLD, we set $(a, \beta_1, \beta_2) = (1, 0.9, 0.999)$ and the learning rate $\epsilon = 0.05$. The CPU time limit was set to 6 min.

A.4 Distribution with long narrow ravines

Each algorithm was run for 30,000 iterations with the settings of specific parameters given in Table A1.

Table A1. Parameter setting for the distribution with long narrow ravines.

Method	Initial value	β_1	β_2	a	λ
SGLD	1e−4				
SGHMC	1e−5	0.9			
pSGLD	1e−4	0.9			1e−6
ASGLD	1e−4	0.9	0.999	1000	1e−5
MSGLD	1e−4	0.99		10	

A.5 Landsat

Each algorithm was run for 3000 epochs with the settings of specific parameters given in Table A2.

Table A2. Parameter setting for the Landsat data example.

Method	Initial value	β_1	β_2	a	λ
SGLD	0.1/4435				
SGHMC	0.1/4435	0.9			
pSGLD	0.1/4435	0.9			1e−5
ASGLD	0.1/4435	0.9	0.999	10	1e−5
MSGLD	0.1/4435	0.9		5	

A.6 MNIST

Each algorithm was run for 250 epochs, where the first 100 epochs were run with the conventional Gaussian prior $N(0, 1)$, and the followed 150 epochs were run with the mixture Gaussian prior given in the paper. For Adam, the objective function was set as (6) with $\lambda = 0$ for all 250 epochs. The settings of the specific parameters were given in Table A3.

Table A3. Parameter settings for MNIST before (stage I) and after (stage II) sparse learning.

	Stage I						Stage II					
Method	Initial	β_1	β_2	a	λ	τ	Initial	β_1	β_2	a	λ	τ
ADAM	0.001	0.9	0.999		1e−8		0.0001	0.9	0.999		1e−8	
ASGLD	0.5	0.9	0.999	10	1e−6	1e−2	0.5/10	0.9	0.999	1	1e−8	1e−5
MSGLD	0.5	0.99		1		1e−3	0.5/10	0.99		1		1e−5

A.7 Cifar-10 and Cifar-100

For SGD, the objective function was set as (6) with $\lambda = 5.0\text{e}{-4}$. For Adam, the objective function was set as (6) with $\lambda = 0$. The settings of specific parameters are given in Table A4.

Table A4. Parameter settings for CIFAR-10 and CIFAR-100.

	CIFAR-10					CIFAR-100				
Method	Initial	β_1	β_2	a	λ	Initial	β_1	β_2	a	λ
SGD	0.1					0.1				
ADAM	0.001	0.9	0.999		1e−8	0.001	0.9	0.999		1e−8
SGLD	0.1/50000					0.1/50,000				
SGHMC	0.1/50000	0.9				0.1/50,000	0.9			
pSGLD	0.001/50000	0.99			1e−6	0.0001/50,000	0.99			1e−6
ASGLD	0.1/50000	0.9	0.999	20	1e−6	0.1/50,000	0.9	0.999	20	1e−8
MSGLD	0.1/50000	0.9		1		0.1/50,000	0.9		1	