Communications in Applied Mathematics and Computational Science

A STOCHASTIC VERSION OF STEIN VARIATIONAL GRADIENT DESCENT FOR EFFICIENT SAMPLING

Lei Li, Yingzhou Li, Jian-Guo Liu, Zibu Liu and Jianfeng Lu

vol. 15 no. 1 2020

msp

COMM. APP. MATH. AND COMP. SCI.
Vol. 15, No. 1, 2020
dx.doi.org/10.2140/camcos.2020.15.37

A STOCHASTIC VERSION OF STEIN VARIATIONAL GRADIENT DESCENT FOR EFFICIENT SAMPLING

LEI LI, YINGZHOU LI, JIAN-GUO LIU, ZIBU LIU AND JIANFENG LU

We propose in this work RBM-SVGD, a stochastic version of the Stein variational gradient descent (SVGD) method for efficiently sampling from a given probability measure, which is thus useful for Bayesian inference. The method is to apply the random batch method (RBM) for interacting particle systems proposed by Jin et al. to the interacting particle systems in SVGD. While keeping the behaviors of SVGD, it reduces the computational cost, especially when the interacting kernel has long range. We prove that the one marginal distribution of the particles generated by this method converges to the one marginal of the interacting particle systems under Wasserstein-2 distance on fixed time interval $[0, T]$. Numerical examples verify the efficiency of this new version of SVGD.

1. Introduction

The empirical measure with samples from some probability measure (which might be known up to a multiplicative factor) has many applications in Bayesian inference [5; 3] and data assimilation [17]. A class of widely used sampling methods is the Markov chain Monte Carlo (MCMC) methods, where the trajectory of a particle is given by some constructed Markov chain with the desired distribution invariant. The trajectory of the particle is clearly stochastic, and the Monte Carlo methods take effect slowly for small number of samples. Unlike MCMC, the Stein variational gradient method (proposed by Liu and Wang in [20]) belongs to particle-based variational inference sampling methods (see also [22; 9]). These methods update particles by solving optimization problems, and each iteration is expected to make progress. As a nonparametric variational inference method, SVGD gives a deterministic way to generate points that approximate the desired probability distribution by solving an ODE system. Suppose that we are interested in some target probability distribution with density $\pi(x) \propto \exp(-V(x))$ $(x \in \mathbb{R}^d)$. In SVGD, one sets $V = -\log \pi$, chooses some symmetric positive definite kernel $\mathcal{K}(x, y)$,

MSC2010: 62D05, 65C35.
Keywords: random batch method, RBM-SVGD, nonparametric variational inference, KL divergence, reproducing kernel Hilbert space, MCMC.

and solves the following ODE system for given initial points $\{X_i(0)\}_{i=1}^N$ [20; 19]:

$$\dot{X}_i = \frac{1}{N}\sum_{j=1}^{N} \nabla_y \mathcal{K}(X_i, X_j) - \frac{1}{N}\sum_{j=1}^{N} \mathcal{K}(X_i, X_j)\nabla V(X_j), \quad i = 1, \dots, N, \quad (1\text{-}1)$$

where N is the number of particles for the sampling purpose. The subindex "y" in ∇_y means that the gradient is taken with respect to the second variable in $\mathcal{K}(\cdot, \cdot)$; i.e., $\nabla_y \mathcal{K}(X_i, X_j) := \nabla_y \mathcal{K}(x, y)|_{(x,y)=(X_i, X_j)}$. When t is large enough, the empirical measure constructed using $\{X_i(t)\}_{i=1}^N$ is expected to be close to π, i.e.,

$$\frac{1}{N}\sum_{i=1}^{N} \delta(x - X_i(t)) \approx \pi(x)\, dx, \quad t \gg 1.$$

Below, in Section 2, we will explain why this is expected to be true. Theoretic understanding of (1-1) is limited. For example, the convergence of the particle system (1-1) is still open. Recently, there have been a few attempts at understanding the limiting mean field PDE [19; 21]. In particular, Lu et al. [21] showed the convergence of the mean field PDE to the desired measure π.

In practice, SVGD seems to perform quite well, better compared with some typical Monte Carlo methods in some examples [19; 10]. It provides consistent estimation for generic distributions as Monte Carlo methods do, but with fewer samples. SVGD seems to be more efficient than some Monte Carlo methods in the particle level for approximating the desired measure, when the number of particles is small. Interestingly, it reduces to the maximum a posterior (MAP) method when $N = 1$ [20].

Though (1-1) behaves well when the particle number N is not very big, one sometimes still needs an efficient algorithm to simulate (1-1). For example, when the dimension of the problem is not very high, in a typical MCMC method, the number of particles is several millions, or $N \approx 10^6$, while in SVGD, one may have $N \approx 10^3$. Simulating (1-1) needs $O(N^2)$ work to compute the interactions for each iteration, especially for interaction kernels that are not superlocalized or particles that are not sparse. In fact, for such situations, to compute the interaction force for one particle, one must consider all the other $N - 1$ particles to have enough accuracy. There are N particles, so one must consider $O(N^2)$ interactions, which yields the $O(N^2)$ complexity for one iteration. Though $N \approx 10^2$–10^3 is not large, the $O(N^2)$ complexity makes the cost of SVGD for these cases comparable with MCMC with larger number of particles. Hence, it is highly motivated to develop a cheap version of SVGD.

In this work, we propose RBM-SVGD, a stochastic version of SVGD for sampling from a given probability measure. The idea is very natural: we apply the random batch method in [16] to the interacting particle system (1-1). Note that in the random

batch method, the "batch" refers to the set for computing the interaction forces, not to be confused with the "batch" of samples for computing gradient as in stochastic gradient descent (SGD). Of course, if V is the loss function corresponding to many samples, or the probability density in Bayesian inference corresponding to many observed data, the data-mini-batch idea can be used to compute ∇V in SVGD as well [20]. With the random batch idea for computing interaction, the complexity for each iteration now is only $O(N)$. Moreover, it inherits the advantages of SVGD (i.e., efficient for sampling when the number of particles is not large) since the random batch method is designed to approximate the particle system directly. In fact, we will prove that the one marginal of the random batch method converges to the one marginal of the interacting particle systems under Wasserstein-2 distance on a fixed time interval $[0, T]$. Note that the behavior of randomness in RBM-SVGD is different from that in MCMC. In MCMC, the randomness is required to ensure that the desired probability is invariant under the transition. The randomness in RBM-SVGD is simply due to the batch for computing the interaction forces, which is mainly for speeding up the computation. Though this randomness is not essential for sampling from the invariant measure, it may have other benefits. For example, it may lead to better ergodic properties for the particle system.

2. Mathematical background of SVGD

We now give a brief introduction to the SVGD proposed in [20] and provide some discussions. The derivation here is a continuous counterpart of that in [20].

Assume that random variable $X \in \mathbb{R}^d$ has density $p_0(x)$. Consider some mapping $\mathcal{T} : \mathbb{R}^d \to \mathbb{R}^d$, and we denote the distribution of $\mathcal{T}(X)$ by $p := \mathcal{T}_{\#} p_0$, which is called the push-forward of p_0 under $\mathcal{T}$. The goal is to make $\mathcal{T}_{\#} p_0$ closer to $\pi(x)$ in some sense. The way to measure the closeness of measures in [20] is taken to be the Kullback–Leibler (KL) divergence, which is also known as the relative entropy, defined by

$$\mathrm{KL}(\mu \parallel \nu) = \mathbb{E}_{Y\sim\mu} \log\left(\frac{d\mu}{d\nu}(Y)\right), \tag{2-1}$$

where $\frac{d\mu}{d\nu}$ is the well known Radon–Nikodym derivative. In [20, Theorem 3.1], it is shown that the Gateaux differential of $\mathcal{T} \mapsto G(\mathcal{T}) := \mathrm{KL}(p \parallel \pi)$ is given by

$$\left\langle \frac{\delta G}{\delta \mathcal{T}}, \phi \right\rangle = -\mathbb{E}_{Y\sim p} S_\pi \phi(Y) \quad \text{for all } \phi \in C_c^\infty(\mathbb{R}^d; \mathbb{R}^d) \tag{2-2}$$

where S_q associated with a probability density q is called the Stein operator given by

$$S_q \phi(x) = \nabla(\log q(x)) \cdot \phi(x) + \nabla \cdot \phi(x). \tag{2-3}$$

In fact, using the formula

$$\frac{d}{d\epsilon}(\mathcal{T}+\epsilon\phi\circ\mathcal{T})_{\#}p_0|_{\epsilon=0}=\frac{d}{d\epsilon}(I+\epsilon\phi)_{\#}p|_{\epsilon=0}=-\nabla\cdot(p\phi)=-pS_p\phi, \quad (2\text{-}4)$$

and $\frac{\delta \mathrm{KL}(p\|\pi)}{\delta p}=\log p+1-\log\pi$, one finds

$$\left\langle\frac{\delta G}{\delta\mathcal{T}},\phi\right\rangle=\left\langle\frac{\delta\,\mathrm{KL}(p\parallel\pi)}{\delta p},-\nabla\cdot(p\phi)\right\rangle=-\int_{\mathbb{R}^d}pS_\pi\phi\,dx. \quad (2\text{-}5)$$

The quantity $\langle\frac{\delta G}{\delta\mathcal{T}},\phi\rangle$ can be understood as the directional derivative of $G(\cdot)$ in the direction given by ϕ. The paring in the second term above is in $L^2(\mathbb{R}^d)$ sense.

Based on this calculation, we now consider a continuously varying family of mappings $\mathcal{T}_\tau$ with $\tau\geq 0$ and

$$\frac{d}{d\tau}\mathcal{T}_\tau=\phi_\tau\circ\mathcal{T}_\tau.$$

Here, "$\circ$" means composition, i.e., for any given x, $\frac{d}{d\tau}\mathcal{T}_\tau(x)=\phi_\tau(\mathcal{T}_\tau(x))$. In this sense $x\mapsto X(\tau;x):=\mathcal{T}_\tau(x)$ is the trajectory of x under this mapping; x can be viewed as the so-called Lagrangian coordinate as in fluid mechanics while ϕ_τ is the flow field. We denote

$$p_\tau:=(\mathcal{T}_\tau)_{\#}p_0. \quad (2\text{-}6)$$

The idea is then to choose ϕ_τ such that the functional $\tau\mapsto G(\mathcal{T}_\tau)$ decays as fast as possible. Note that to optimize the direction, we must require the field to have bounded magnitude $\|\phi_\tau\|_H\leq 1$, where H is some subspace of the functions defined on $\mathbb{R}^d$. The optimized curve $\tau\mapsto\mathcal{T}_\tau$ is a constant-speed curve (in some manifold). Hence, the problem is reduced to the optimization problem

$$\sup\{\mathbb{E}_{Y\sim p}S_\pi\phi(Y)\mid\|\phi\|_H\leq 1\}. \quad (2\text{-}7)$$

It is observed in [20] that this optimization problem can be solved by a convenient closed formula if H is the so-called (vector) reproducing kernel Hilbert space (RKHS) [1; 2]. A (scalar) RKHS is a Hilbert space, denoted by $\mathcal{H}$, consisting of functions defined on some space Ω (in our case $\Omega=\mathbb{R}^d$) such that the evaluation function $f\mapsto E_x(f):=f(x)$ is continuous for all $x\in\Omega$. There thus exists $k_x\in\mathcal{H}$ such that $E_x(f)=\langle f,k_x\rangle_{\mathcal{H}}$. Then the kernel $\mathcal{K}(x,y):=\langle k_x,k_y\rangle_{\mathcal{H}}$ is symmetric and positive definite, meaning that $\sum_{i=1}^n\sum_{j=1}^n\mathcal{K}(x_i,x_j)c_ic_j\geq 0$ for any $x_i\in\Omega$ and $c_i\in\mathbb{R}$. Reversely, given any positive definite kernel, one can construct a RKHS consisting of functions $f(x)$ of the form $f(x)=\int\mathcal{K}(x,y)\psi(y)\,d\mu(y)$ where μ is some suitably given measure on Ω. For example, if μ is the counting measure, choosing $\psi(y)=\sum_{j=1}^\infty a_j1_{x_j}(y)$ ($a_j\in\mathbb{R}$) can recover the form of RKHS in [20]. All such constructions yield isomorphic RKHS as guaranteed by the Moore–Aronszajn

theorem [1]. Now, consider a given μ and $H = \mathcal{H}^d$ to be the vector RKHS:

$$H = \left\{ f = \int_{\mathbb{R}^d} \mathcal{K}(\cdot, y)\psi(y)\, d\mu(y) \;\middle|\; \psi : \mathbb{R}^d \to \mathbb{R}^d, \iint_{\mathbb{R}^d \times \mathbb{R}^d} \mathcal{K}(x, y)\psi(x) \cdot \psi(y)\, d\mu(x)\, d\mu(y) < \infty \right\}.$$

The inner product is defined as

$$\begin{aligned} \langle f^{(1)}, f^{(2)} \rangle_H &= \iint_{\mathbb{R}^d \times \mathbb{R}^d} \mathcal{K}(x, y)\psi^{(1)}(x) \cdot \psi^{(2)}(y)\, d\mu(x)\, d\mu(y) \\ &= \sum_{j=1}^{d} \iint_{\mathbb{R}^d \times \mathbb{R}^d} \mathcal{K}(x, y)\psi_j^{(1)}(x)\psi_j^{(2)}(y)\, d\mu(x)\, d\mu(y). \end{aligned} \tag{2-8}$$

This inner product therefore induces a norm $\|f\|_H = \sqrt{\langle f, f \rangle_H}$. Clearly, H consists of functions with $\|\cdot\|_H$ to be finite. The optimization problem (2-7) can be solved by the Lagrange multiplier method

$$\mathcal{L} = \int_{\mathbb{R}^d} (S_\pi \phi) p_\tau(y)\, dy - \lambda \iint_{\mathbb{R}^d \times \mathbb{R}^d} \mathcal{K}(x, y)\psi(x) \cdot \psi(y)\, d\mu(x)\, d\mu(y),$$

where dy means Lebesgue measure and $\phi(x) = \int_{\mathbb{R}^d} \mathcal{K}(x, y)\psi(y)\, d\mu(y)$. Using $\frac{\delta \mathcal{L}}{\delta \phi} = 0$, we find

$$2\lambda\phi = \int_{\mathbb{R}^d} \mathcal{K}(x, y)(S_\pi^* p_t)(y)\, dy =: \mathcal{V}(p_t), \tag{2-9}$$

where S_π^* is given by

$$S_\pi^*(f) = f(y)\nabla(\log \pi) - \nabla f(y) = -f(y)\nabla V(y) - \nabla f(y). \tag{2-10}$$

The ODE flow

$$\frac{d}{d\tau} \mathcal{T}_\tau = \frac{1}{2\lambda(\tau)} \mathcal{V}(p_\tau) \circ \mathcal{T}_\tau$$

gives the constant-speed optimal curve, so that the velocity is the unit vector in H along the gradient of G. Reparametrizing the curve $t = t(\tau)$ so that $\frac{d\tau}{dt} = 2\lambda$ and denoting $\rho_t := p_{\tau(t)}$, then

$$\frac{d}{dt} \mathcal{T}_t = \mathcal{V}(\rho_t) \circ \mathcal{T}_t. \tag{2-11}$$

Clearly, the curve of $\mathcal{T}_t$ is not changed by this reparametrization. Using (2-4), one finds that ρ satisfies the equation

$$\partial_t \rho = -\nabla \cdot (\mathcal{V}(\rho)\rho) = \nabla \cdot (\rho \mathcal{K} * (\rho \nabla V + \nabla \rho)). \tag{2-12}$$

Here, $\mathcal{K} * f(x) := \int \mathcal{K}(x, y) f(y)\, dy$. It is easy to see that $\exp(-V)$ is invariant under this PDE. According to the explanation here, the right-hand side gives the optimal decreasing direction of KL divergence if the transport flow is measured by RKHS. Hence, one expects it to be the negation of gradient of KL divergence in the manifold of probability densities with metric defined through RKHS. Indeed, Liu made the first attempt to justify this in [19, §3.4].

The above theory has a little trouble for empirical measures because the KL divergence is simply infinity. For empirical measure, $\nabla\rho$ must be in the distributional sense. The good thing for RKHS is that we can move the gradient from $\nabla\rho$ onto the kernel $\mathcal{K}(x, y)$ so that the flow (2-11) becomes (1-1), which makes perfect sense. In fact, if (1-1) holds, the empirical measure is a measure solution to (2-12) (by testing on smooth function φ) [21, Proposition 2.5]. Hence, the ODE system is justified in this level, and one expects that (1-1) will give an approximation for the desired density. The numerical tests in [20] indeed justify this expectation. In this sense, the ODE system is formally a gradient flow of KL divergence, though the KL divergence functional is infinity for empirical measures.

Typical examples of $\mathcal{K}(x, y)$ include $\mathcal{K}(x, y) = (\alpha x \cdot y + 1)^m$, Gaussian kernel $\mathcal{K}(x, y) = e^{-|x-y|^2/(2\sigma^2)}$ for $\mathbb{R}^d$, and $\mathcal{K}(x, y) = (\sin a(x - y))/(\pi(x - y))$ for 1D space $\mathbb{R}$. By Bochner's theorem [25], if a function K has a positive Fourier transform, then

$$\mathcal{K}(x, y) = K(x - y) \tag{2-13}$$

is a positive definite kernel. With this kernel, (1-1) becomes

$$\dot{X}_i = -\frac{1}{N}\sum_{j=1}^{N} \nabla K(X_i - X_j) - \frac{1}{N}\sum_{j=1}^{N} K(X_i - X_j)\nabla V(X_j), \tag{2-14}$$

as used in [21]. Both Gaussians and $1/|x|^\alpha$ with $\alpha \in (0, d)$ have positive Fourier transforms. The difference is that the Gaussian has a short range of interaction while the latter has a long range of interaction. One can smoothen $1/|x|^\alpha$ out by mollifying with Gaussian kernels, resulting in positive definite smooth kernels but with long-range interaction. Choosing localized kernels like Gaussians may have some issues in very high-dimensional spaces [12; 10]. Due to its simplicity, when the dimension is not very high, we choose Gaussian kernels in Section 4.

As a further comment, one may consider other metrics to gauge the closeness of probability measures, such as Wasserstein distances. Also, one can consider other norms for ϕ and get gradient flows in different spaces. These variants have been explored by some authors already [18; 8]. In general, computing the Frechét derivatives in closed form for these variants seems not that easy.

Remark. If we optimize (2-7) for ϕ in $L^2(\mathbb{R}^d;\mathbb{R}^d)$ spaces, the flow is then given by

$$\frac{d}{dt}\mathcal{T} = (S_\pi^*\rho)\circ\mathcal{T}. \tag{2-15}$$

The corresponding PDE is $\partial_t\rho = \nabla\cdot(\rho(\rho\nabla V + \nabla\rho)) = \nabla\cdot(\rho^2\nabla\log(\rho/\pi))$. This is in fact the case when we choose $\mathcal{K}(x,y)=\delta(x-y)$. This PDE, however, will not make sense for empirical measures since $\rho\nabla\rho$ is hard to justify (clearly, the equivalent ODE system has the same trouble). By using RKHS, the derivative on $\nabla\rho$ can be moved onto the kernel and then the ODE system makes sense.

3. The new sampling algorithm: RBM-SVGD

In this section, we introduce the "random batch" or "mini-batch" idea, which has already appeared in many places, and recall the random batch method for simulating interacting particle systems in [16]. By applying the random batch method to (1-1), we obtain a new algorithm, called RBM-SVGD. The proof that RBM-SVGD is close to SVGD on finite time interval is given in Section 3.2.

3.1. ***The algorithms.*** Before we present the random batch method and RBM-SVGD, let us briefly explain what the "random mini-batch" idea is. Let us consider a typical optimization problem in machine learning:

$$\min L(\omega) := \min\frac{1}{M}\sum_{j=1}^{M}\ell(g(z_j;\omega),y_j), \tag{3-1}$$

where $(z_j,y_j)_{j=1}^M$ are some given data set, $g(\cdot;\omega)$ is a model that takes z_j as an input and gives some prediction to y_j, and $\ell(\cdot,\cdot)$ is some function to gauge the discrepancy between $g(z_j;\omega)$ and y_j. Hence, the problem is to find ω such that the discrepancy is small enough. Often $\ell(\cdot,\cdot)$ is a neural network so that computing the gradient is not easy. Hence, if one aims to find the minimizer using gradient descent, the computation cost is high. The idea of "mini-batch" or "random batch" is to choose a small random subset ξ of $\{1,2,\dots,N\}$, and consider the unbiased random estimate

$$L_\xi(\omega) := \frac{1}{B}\sum_{j\in\xi}\ell(g(z_j;\omega),y_j),$$

with $B=|\xi|$, the size of ξ. Using this unbiased estimation L_ξ to replace the original true gradient $\nabla L_\xi \approx \nabla L$, one can form the so-called stochastic gradient descent (SGD) [4; 6]. Using a similar idea for Langevin dynamics, Welling and Teh obtained a Markov chain Monte Carlo method, called the stochastic gradient Langevin dynamics (SGLD), useful for Bayesian inference [29].

for m in $1 : N_T$ **do**
  Divide $\{1, 2, \ldots, pn\}$ into n batches randomly.
  for each batch $\mathcal{C}_q$ **do**
    Update X_i $(i \in \mathcal{C}_q)$ by solving the equation for $t \in [t_{m-1}, t_m)$:

$$\dot{X}_i = \frac{1}{N} F(X_i, X_i) + \left(1 - \frac{1}{N}\right) \frac{1}{p-1} \sum_{j \in \mathcal{C}_q,\, j \neq i} F(X_i, X_j). \tag{3-4}$$

  end for
end for

Algorithm 1. Random batch method without replacement.

Consider in general the interacting particle system of the form

$$\dot{X}_i = \frac{1}{N} \sum_{j=1}^{N} F(X_i, X_j) = \frac{1}{N} F(X_i, X_i) + \frac{1}{N} \sum_{j:j \neq i} F(X_i, X_j). \tag{3-2}$$

Here, $F(x, y)$ does not have to be symmetric, and also $F(x, x)$ is not necessarily zero. It is desirable to develop some cheap random approximation to the interacting forces so that the one-step $O(N^2)$ complexity can be reduced. One idea is to use the "random batch" idea, but how to develop the concrete "random batch" algorithm depends on the concrete applications. Regarding the interacting particle systems, Jin et al. proposed some random grouping approach to achieve this goal in [16].

Here, we adopt the random batch method in [16] to (3-2) and then obtain a stochastic version method for the SVGD ODE system (1-1). For this reason, we explain the random batch method a little bit. Choose a time step η. We define time grid points

$$t_m = m\eta. \tag{3-3}$$

At t_m, one divides the particles into groups randomly, and each group is called a "batch", and then turns on interactions inside batches only. As indicated in [16], the random division of the particles into n batches takes $O(N)$ operations (one can for example use random permutation). Depending on whether one does batches without or with replacement, one can have different versions (see Algorithms 1 and 2). For the ODEs in the algorithms, one can apply any suitable ODE solver. For example, one can use the forward Euler discretization if F is smooth like Gaussian kernels. If K is singular, one may take $p = 2$ and apply the splitting strategy in [16].

For the SVGD ODE system (1-1), the kernel F takes the form

$$F(x, y) = \nabla_y \mathcal{K}(x, y) - \mathcal{K}(x, y) \nabla V(y). \tag{3-6}$$

Applying the random batch method to this special kernel and using any suitable ODE solvers, we get a class of sampling algorithms, which we will call RBM-SVGD. In

for m in $1 : N_T * (N/p)$ **do**
Pick a set $\mathscr{C}$ of size p randomly.
Update X^i $(i \in \mathscr{C})$ by solving the following with pseudotime $s \in [s_{m-1}, s_m)$:

$$\dot{X}_i = \frac{1}{N} F(X_i, X_i) + \left(1 - \frac{1}{N}\right) \frac{1}{p-1} \sum_{j \in \mathscr{C},\, j \neq i} F(X_i, X_j). \tag{3-5}$$

end for

Algorithm 2. Random batch method with replacement.

for k in $0 : N_T - 1$ **do**
Divide $\{1, 2, \ldots, pn\}$ into n batches randomly.
for each batch $\mathscr{C}_q$ **do**
For all $i \in \mathscr{C}_q$,

$$X_i^{(k+1)} \leftarrow X_i^{(k)} + \frac{1}{N} (\nabla_y \mathscr{K}(X_i^{(k)}, X_i^{(k)}) - \mathscr{K}(X_i^{(k)}, X_i^{(k)}) \nabla V(X_i^{(k)})) \eta_k + \Phi_{k,i} \eta_k,$$

where

$$\Phi_{k,i} = \frac{N-1}{N(p-1)} \sum_{j \in \mathscr{C}_q,\, j \neq i} (\nabla_y \mathscr{K}(X_i^{(k)}, X_j^{(k)}) - \mathscr{K}(X_i^{(k)}, X_j^{(k)}) \nabla V(X_j^{(k)})). \tag{3-7}$$

end for
end for

Algorithm 3. RBM-SVGD.

this work, we will focus on the ones without replacement. The one with forward Euler discretization (with possible variant step size) is shown in Algorithm 3. Clearly, the complexity is $O(pN)$ for each iteration.

Here, N_T is the number of iterations and $\{\eta_k\}$ is the sequence of time steps, which play the same role as learning rate in SGD [4; 6]. For some applications, one may simply set $\eta_k = \eta \ll 1$ to be a constant and get relatively good results. However, in many high-dimensional problems, choosing η_k to be constant may yield divergent sequences [23]. One may decrease η_k to obtain convergent data sequences. For example, one may simply choose $\eta_k = 1/k$ as in SGD. Another frequently used strategy is the AdaGrad approach [11; 28].

3.2. ***Theoretic results.*** We now give convergence analysis regarding the time-continuous version of RBM-SVGD on torus $\mathbb{T}^d$ (i.e., choosing the particular force (3-6) for Algorithm 1 and $X_i \in \mathbb{T}^d$). The analysis in this section justifies the expectation that RBM-SVGD should give similar performance as the original SVGD, as confirmed by the numerical experiments in Section 4.

By "torus", we mean the domain is equipped with periodic boundary conditions. The derivation of SVGD clearly stays unchanged for the torus. The reason we consider a torus is that (1-1) is challenging to analyze in $\mathbb{R}^d$ because of the nonlocal effect of the external force. On the torus, all functions are smooth and bounded. Moreover, using bounded domains with periodic boundary condition can always approximate the problem in $\mathbb{R}^d$ in practice.

Consider the random force for $z = (x_1, \dots, x_N) \in \mathbb{T}^{Nd}$ defined by

$$f_i(z) := \left(1 - \frac{1}{N}\right) \frac{1}{p-1} \sum_{j: j \in \mathscr{C}} F(x_i, x_j), \tag{3-8}$$

where $\mathscr{C}$ is the random batch that contains i in the random batch method. Correspondingly, the exact force is given by

$$F_i(z) = \frac{1}{N} \sum_{j: j \neq i} F(x_i, x_j).$$

Define the "noise" by

$$\chi_i(z) := \frac{1}{N} \sum_{j: j \neq i} F(x_i, x_j) - f_i(z). \tag{3-9}$$

We have the following consistency result regarding the random batch.

Lemma 1. *For given $z = (x_1, \dots, x_N) \in \mathbb{T}^{Nd}$ (or $\mathbb{R}^{Nd}$), it holds that*

$$\mathbb{E}\chi_i(z) = 0. \tag{3-10}$$

Moreover, the second moment is given by

$$\mathbb{E}|\chi_i(z)|^2 = \left(1 - \frac{1}{N}\right)^2 \left(\frac{1}{p-1} - \frac{1}{N-1}\right) \Lambda_i(z), \tag{3-11}$$

where

$$\Lambda_i(z) = \frac{1}{N-2} \sum_{j: j \neq i} \left| F(x_i, x_j) - \frac{1}{N-1} \sum_{k: k \neq i} F(x_i, x_k) \right|^2. \tag{3-12}$$

The proof is similar to that in [16], but we also attach it in Appendix A for convenience.

We recall that the Wasserstein-2 distance is given by [26]

$$W_2(\mu, \nu) = \left(\inf_{\gamma \in \Pi(\mu, \nu)} \int_{\mathbb{T}^d \times \mathbb{T}^d} |x - y|^2 \, d\gamma \right)^{1/2}, \tag{3-13}$$

where $\Pi(\mu, \nu)$ is called the transport plan, consisting of all the joint distributions whose marginal distributions are μ and ν, respectively: i.e., for any Borel set $E \subset \mathbb{T}^d$, $\mu(E) = \iint_{x \in E, y \in \mathbb{T}^d} \gamma \, (dx, dy)$ and $\nu(E) = \int_{x \in \mathbb{T}^d, y \in E} \gamma \, (dx, dy)$.

We now state the convergence result for the time-continuous version of RBM-SVGD, where we recall that $F(x, y)$ given by (3-6). We use $\widetilde{X}$ to denote the process generated by the random algorithm while X is the process by (1-1). The particles are exchangeable if the initial values are sampled i.i.d. from the same distribution. Hence, the distributions of X_i are the same, and we call this one-particle distribution the one marginal distribution, which is a probability measure in $\mathbb{T}^d$ or $\mathbb{R}^d$. We denote it by $\mu_N^{(1)}$ for convenience. Similarly, we introduce the one marginal distribution for the particles generated by the random algorithm, denoted by $\tilde{\mu}_N^{(1)}$.

Theorem 2. *Assume V and K are smooth on torus $\mathbb{T}^d$. The initial data X_i^0 are drawn independently from the same initial distribution. Given $T > 0$, there exists $C(T) > 0$, such that*

$$\sup_{t\le T} \mathbb{E}|X_i(t) - \widetilde{X}_i(t)|^2 \le C(T)\frac{\eta}{p-1}.$$

Consequently, the one marginals $\mu_N^{(1)}$ and $\tilde{\mu}_N^{(1)}$ are close under Wasserstein-2 distance:

$$\sup_{t\le T} W_2(\mu_N^{(1)}(t), \tilde{\mu}_N^{(1)}(t)) \le C(T)\sqrt{\frac{\eta}{p-1}}.$$

Proof. In the proof below, the constant C will represent a general constant independent of N and p, but its concrete meaning can change for every occurrence.

Consider the corresponding two processes and $t \in [t_{m-1}, t_m]$:

$$\frac{d}{dt}\widetilde{X}_i = \frac{1}{N}(\nabla_y \mathcal{K}(\widetilde{X}_i, \widetilde{X}_i) - \mathcal{K}(\widetilde{X}_i, \widetilde{X}_i)\nabla V(\widetilde{X}_i)) + \frac{1-1/N}{p-1}\sum_{j:j\in\mathscr{C}} (\nabla_y \mathcal{K}(\widetilde{X}_i, \widetilde{X}_j) - \mathcal{K}(\widetilde{X}_i, \widetilde{X}_j)\nabla V(\widetilde{X}_j)) \quad (3\text{-}14)$$

and

$$\frac{d}{dt}X_i = \frac{1}{N}(\nabla_y \mathcal{K}(X_i, X_i) - \mathcal{K}(X_i, X_i)\nabla V(X_i)) + \frac{1}{N}\sum_{j:j\neq i} (\nabla_y \mathcal{K}(X_i, X_j) - \mathcal{K}(X_i, X_j)\nabla V(X_j)). \quad (3\text{-}15)$$

Taking the difference and dotting with $\widetilde{X}_i - X_i$, one has

$$(\widetilde{X}_i - X_i)\cdot\frac{d}{dt}(\widetilde{X}_i(t) - X_i(t)) \le \frac{C}{N}|\widetilde{X}_i(t) - X_i(t)|^2 + (\widetilde{X}_i(t) - X_i(t))\cdot(I_1 + I_2)$$

where

$$I_1 = \frac{1-1/N}{p-1}\Bigg(\sum_{j:j\in\mathscr{C}}(\nabla_y\mathcal{K}(\widetilde{X}_i,\widetilde{X}_j)-\mathcal{K}(\widetilde{X}_i,\widetilde{X}_j)\nabla V(\widetilde{X}_j)) - \sum_{j:j\in\mathscr{C}}(\nabla_y\mathcal{K}(X_i,X_j)-\mathcal{K}(X_i,X_j)\nabla V(X_j))\Bigg),$$

$$I_2 = \frac{1-1/N}{p-1}\sum_{j:j\in\mathscr{C}}(\nabla_y\mathcal{K}(X_i,X_j)-\mathcal{K}(X_i,X_j)\nabla V(X_j)) - \frac{1}{N}\sum_{j:j\neq i}(\nabla_y\mathcal{K}(X_i,X_j)-\mathcal{K}(X_i,X_j)\nabla V(X_j)).$$

Hence, introducing

$$u(t) = \mathbb{E}|X_i(t)-\widetilde{X}_i(t)|^2 = \mathbb{E}|X_1(t)-\widetilde{X}_1(t)|^2,$$

we have

$$\frac{d}{dt}u \le \frac{C}{N}u(t) + \mathbb{E}(X_i-\widetilde{X}_i)\cdot I_1 + \mathbb{E}(X_i-\widetilde{X}_i)\cdot I_2.$$

Due to the smoothness of K and V on the torus, we easily find

$$|I_1| \le C\frac{1}{p-1}\sum_{j\in\mathscr{C},\, j\neq i}(|X_i-\widetilde{X}_i|+|X_j-\widetilde{X}_j|) = C|X_i-\widetilde{X}_i| + C\frac{1}{p-1}\sum_{j\in\mathscr{C},\, j\neq i}|X_j-\widetilde{X}_j|,$$

where C is independent of N. Note that $\mathscr{C}$ is not independent of $X_j(t)$ for $t > t_{m-1}$, so to continue we must consider conditional expectation. Let $\mathscr{F}_{m-1}$ be the σ-algebra generated by $X_i(\tau)$, $\widetilde{X}_i(\tau)$ for $\tau \le t_{m-1}$ (including the initial data drawn independently) and the random division of the batches at t_{m-1}. Then (3-14) directly implies almost surely that

$$\mathbb{E}(|X_j(t)-X_j(t_{m-1})|\,|\,\mathscr{F}_{m-1}) \le C\eta, \quad \mathbb{E}(|\widetilde{X}_j(t)-\widetilde{X}_j(t_{m-1})|\,|\,\mathscr{F}_{m-1}) \le C\eta. \tag{3-16}$$

Thus, defining the error process

$$Y_i(t) = \widetilde{X}_i(t) - X_i(t), \tag{3-17}$$

we have $\mathbb{E}(|Y_i(t)-Y_i(t_{m-1})|) \le C\eta$, yielding

$$|\sqrt{u}(t)-\sqrt{u}(t_{m-1})| \le C\eta. \tag{3-18}$$

Note that

$$\mathbb{E}\left(|X_i-\widetilde{X}_i|\frac{1}{p-1}\sum_{j\in\mathscr{C},\, j\neq i}|X_j-\widetilde{X}_j|\right) \le \sqrt{u}\left(\frac{1}{p-1}\mathbb{E}\sum_{j\in\mathscr{C},\, j\neq i}|X_j-\widetilde{X}_j|^2\right)^{1/2}.$$

The inside of the parentheses can be estimated as

$$\frac{1}{p-1}\mathbb{E}\sum_{j\in\mathscr{C},\, j\neq i}|X_j-\widetilde{X}_j|^2=\frac{1}{p-1}\mathbb{E}\sum_{j\in\mathscr{C},\, j\neq i}|X_j(t_{m-1})-\widetilde{X}_j(t_{m-1})|^2$$
$$+\frac{1}{p-1}\mathbb{E}(\mathbb{E}((|X_j-\widetilde{X}_j|^2-|X_j(t_{m-1})-\widetilde{X}_j(t_{m-1})|^2)\mid\mathscr{F}_{m-1})).$$

The first term on the right-hand side then becomes $u(t_{m-1})$ by Lemma 1. By (3-16), it is clear that

$$\mathbb{E}((|X_j-\widetilde{X}_j|^2-|X_j(t_{m-1})-\widetilde{X}_j(t_{m-1})|^2)\mid\mathscr{F}_{m-1})$$
$$\leq 2|X_j(t_{m-1})-\widetilde{X}_j(t_{m-1})|C\eta+C\eta^2.$$

Hence,

$$\mathbb{E}(X_i-\widetilde{X}_i)\cdot I_1\leq Cu(t)+Cu(t_{m-1})+C\sqrt{u(t_{m-1})}\eta+C\eta^2,$$

where C is independent of N. Since $u(t_{m-1})\leq Cu(t)+C\eta^2$ by (3-18), then

$$\mathbb{E}(X_i-\widetilde{X}_i)\cdot I_1\leq Cu(t)+C\eta^2.$$

Letting $Z=(X_1,\ldots,X_N)$, one sees easily that $I_2=\chi_i(Z(t))$. Then, we find

$$Y_i(t)\cdot I_2(t)=(Y_i(t)-Y_i(t_{m-1}))\cdot\chi_i(Z(t))+Y_i(t_{m-1})\cdot\chi_i(Z(t))=:J_1+J_2.$$

In J_2, $Y_i(t_{m-1})$ is independent of the random batch division at t_{m-1}. Then, Lemma 1 tells us that

$$\mathbb{E}J_2=0.$$

Using (3-14), we have

$$Y_i(t)-Y_i(t_{m-1})=-\int_{t_{m-1}}^{t}\chi_i(Z(s))\,ds+\int_{t_{m-1}}^{t}f_i(\widetilde{Z}(s))-f_i(Z(s))\,ds. \quad (3\text{-}19)$$

Since χ_i is bounded,

$$\left|\mathbb{E}\int_{t_{m-1}}^{t}\chi_i(Z(s))\cdot\chi_i(Z(t))\,ds\right|\leq\|\Lambda_i\|_\infty\eta\leq 2\|F\|_\infty\frac{\eta}{p-1}, \quad (3\text{-}20)$$

where C is related to the infinity norm of the variance of $\chi_i(t)$. This is the main term in the local truncation error. Just as we did for I_1,

$$|f_i(\widetilde{Z}(s))-f_i(Z(s))|\leq C\frac{1}{p-1}\sum_{j\in\mathscr{C},\, j\neq i}(|X_i-\widetilde{X}_i|+|X_j-\widetilde{X}_j|)$$
$$=C|X_i-\widetilde{X}_i|+\frac{C}{p-1}\sum_{j\in\mathscr{C},\, j\neq i}|X_j-\widetilde{X}_j|.$$

Since

$$\mathbb{E}\frac{1}{p-1}\sum_{j\in\mathcal{C},\, j\neq i}|X_j-\widetilde{X}_j| \le \mathbb{E}\frac{1}{p-1}\sum_{j\in\mathcal{C},\, j\neq i}|X_j(t_{m-1})-\widetilde{X}_j(t_{m-1})| + \mathbb{E}\left(\frac{1}{p-1}\sum_{j\in\mathcal{C},\, j\neq i}\mathbb{E}(|X_j(s)-\widetilde{X}_j(s)-(X_j(t_{m-1})-\widetilde{X}_j(t_{m-1}))| \mid \mathcal{F}_{m-1})\right),$$

this is controlled by $C\sqrt{u(t_{m-1})}+C\eta$. Hence,

$$\mathbb{E}J_1 \le 2\|F\|_\infty\frac{\eta}{p-1}+C\sqrt{u(t_{m-1})}\eta+C\eta^2.$$

Using the fact that $u(t_{m-1})\le u(t)+C\eta$, one eventually has that

$$\frac{d}{dt}u \le Cu+2\|F\|_\infty\frac{\eta}{p-1}+C\eta^2.$$

Applying Grönwall's inequality, we find

$$\sup_{t\le T}u(t) \le C(T)\frac{\eta}{p-1}.$$

The last claim for W_2 distance follows from the definition of W_2. □

Note that the one marginal $\mu_N^{(1)}(t)$ is the distribution of $X_i(t)$ for any i, which is deterministic. This should be distinguished from the empirical measure $\mu_N = (1/N)\sum_i \delta(x-X_i(t))$ which is random. As can be seen from the proof, the main contribution in the local truncation error comes from the variance of the noise χ_i.

As can be seen, the error bound is given by the square root of variance of the random force times $\sqrt{\eta}=\sqrt{T/N_T}$ with N_T being the number of steps. Hence, the result is a type of law of large number convergence result (see [16] for more details). The bigness of the variance on one hand depends on the batch size as $1/(p-1)-1/(N-1)$, while on the other hand depends on the bigness of the interaction. As long as the variance is bounded, the convergence of random batch method is ensured.

One crucial part is that the bigness of the variance depends on the bigness of the interaction, instead of the range of the interaction. This means that the random batch version of the algorithm is particularly useful when the interaction has long range or when the particles are not sparse. In fact, if the interaction has short range and the particles are sparse, one can use some data structure like cell-list [13, Appendix F] to reduce the computation of the interactions from $O(N^2)$ to $O(N)$. However, when the interaction has long range or is not sparse (like the case in the example in Section 4.2), those data structures cannot be used any more, and RBM-SVGD becomes useful: it can still reduce the cost from $O(N^2)$ to $O(N)$.

As another observation, according to (3-11) and (3-12), the bigness of the variance depends on the bigness of the interaction kernel. As long as the variance stays controlled, the convergence of RBM-SVGD to SVGD is guaranteed. In this sense, the range of the interaction kernel is not sensitive to RBM-SVGD, so it can intrinsically be used for kernels that have long range. The choice of kernels clearly affects the performance of SVGD, but it seems not so significant for RBM-SVGD to approximate SVGD. In other words, we expect RBM-SVGD to work well when the kernel is chosen such that SVGD behaves well. In fact, our experience in Section 4 confirmed this.

Remark. We believe the error bound in Theorem 2 can be made independent of T due to the intrinsic structure of SVGD discussed above in Section 2. Then RBM-SVGD can be used as the efficient sampling algorithm from the desired distribution π. Such long time estimates are often established by some contracting properties of the ODE flows, so one may want to find the intrinsic converging structure of (1-1). However, rigorously establishing such results seems nontrivial due to the nonlocal effects of the external forces (∇V terms).

4. Numerical experiments

We consider some test examples in [19] to validate RBM-SVGD algorithm and compare with the original SVGD algorithm. In particular, in a toy example for 1D Gaussian mixture, RBM-SVGD is proved to be effective in the sense that the particle system converges to the expected distribution with less running time than the original SVGD method. A more practical example, namely Bayesian logistic regression, is also considered to verify the effectiveness of RBM-SVGD on large data sets in high dimension. Competitive prediction accuracy is presented by RBM-SVGD, and less time is needed. Hence, RBM-SVGD seems to be a more efficient method.

All numerical results in this section are implemented with Matlab R2018a and performed on a machine with Intel Xeon CPU E5-1650v2 at 3.50 GHz with 64 GB memory.

4.1. ***1D Gaussian mixture.*** As a first example, we use the Gaussian mixture probability in [20] for RBM-SVGD. The initial distribution is $\mathcal{N}(-10, 1)$, Gaussian with mean -10 and variance 1. The target density is given by the Gaussian mixture

$$\pi(x) = \frac{1}{3} \cdot \frac{1}{\sqrt{2\pi}} e^{-(x+2)^2/2} + \frac{2}{3} \cdot \frac{1}{\sqrt{2\pi}} e^{-(x-2)^2/2}. \tag{4-1}$$

The kernel for the RKHS is the Gaussian kernel

$$K(x) = \frac{1}{\sqrt{2\pi h}} e^{-x^2/2h}, \tag{4-2}$$

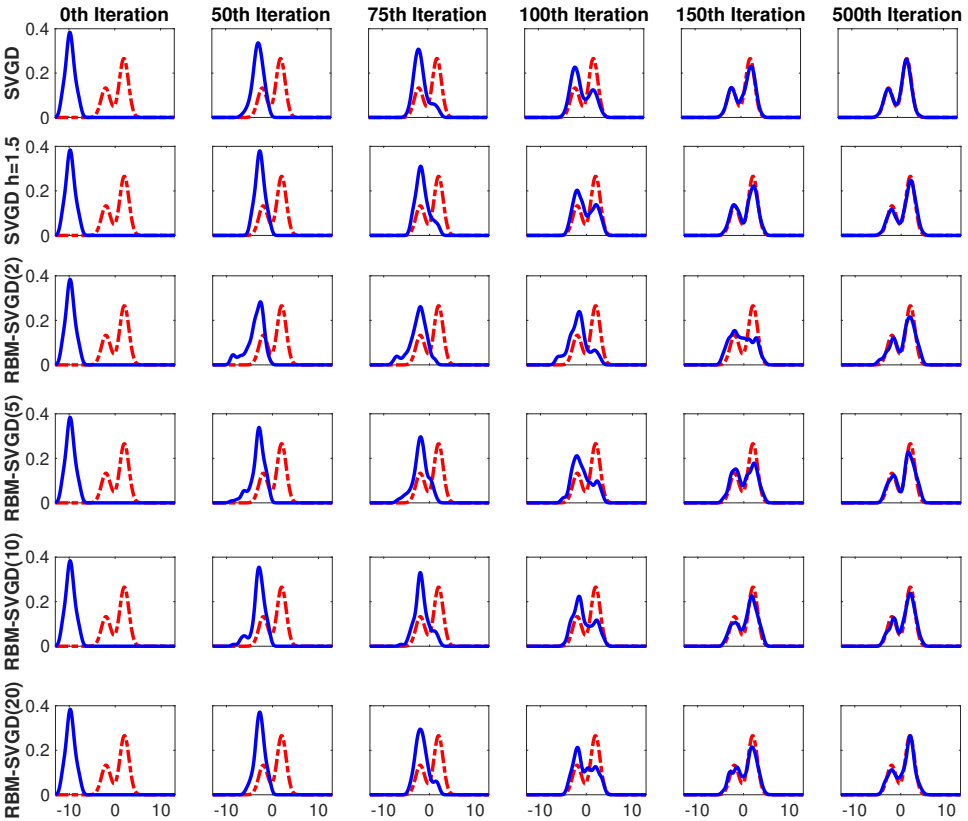

Figure 1. Comparison between SVGD and RBM-SVGD with different batch sizes using $N = 100$ particles. The first row reproduces results in [20]; the second row uses a fixed bandwidth $h = 2$ with other settings being the same as in the first row; the third to fifth rows apply RBM-SVGD with batch sizes 2, 5, and 20, respectively, and other settings are the same as in the second row. In all figures, red dashed curves indicate target density functions whereas blue curves are empirical density estimators (estimated using the kernel density estimator).

where h is the bandwidth parameter. For a fair comparison with the numerical results in [20], we first reproduce their results using $N = 100$ particles and dynamic bandwidth parameter $h = \text{med}^2/(2 \log N)$, where med is the median of the pairwise distance between the current points. Since dynamic bandwidth is infeasible for RBM-SVGD, we produce the results with fixed bandwidth $h = 2$ for the comparison between SVGD and RBM-SVGD. The RBM-SVGD uses Algorithm 3 with initial step size 0.2 and the following step sizes generated from AdaGrad. Different batch sizes are tested to demonstrate the efficiency of RBM-SVGD. Numerical results are illustrated in Figure 1 with the same initial random positions of particles following an $\mathcal{N}(-10, 1)$ distribution.

As stated in [20], the difficulty lies in the strong disagreement between the initial density function and the target density $\pi(x)$. According to the first and second rows in Figure 1, SVGD with and without the fixed bandwidth parameter capture the target density efficiently and the corresponding convergence behaviors are similar to each other. Reading from the last column of Figure 1, we observe that RBM-SVGD inherits the advantage of SVGD in the sense that it can conquer the challenge and also show compelling result with SVGD. When the batch size is small, e.g., $p = 2$ or $p = 5$, the estimated densities differ from that of SVGD, and according to our experience, the estimated densities are not very stable across several executions while, in theory, RBM-SVGD runs N/p times faster than SVGD. Hence, RBM-SVGD with $p = 5$ at the 500-th iteration costs the same as 50 iterations of SVGD. According to Figure 1, RBM-SVGD(2) at the 500-th iteration significantly outperforms the 50-th iteration of SVGD. As we increase the batch size, as in the last two rows of Figure 1, more stable and similar behavior to SVGD is observed.

Provided the good performance of RBM-SVGD, we also check the sampling power and its computational cost. We conduct the following simulations with $N = 256$ particles for 500 iterations with the Gaussian kernel (4-2). For RBM-SVGD, we use fixed bandwidth $h = 2$ whereas SVGD uses the aforementioned dynamic bandwidth strategy. When we apply SVGD or RBM-SVGD with different batch sizes, the same initial random positions of particles is used. For a given test function $h(x)$, we compute the estimated expectation $\bar{h} = (1/N)\sum_{i=1}^{N} h(X_i(T))$ and the sampling accuracy is measured via the minimum square error (MSE) over 100 random initializations following the same distribution as before:

$$\mathrm{MSE} = \frac{1}{100}\sum_{j=1}^{100}(\bar{h}_j - \mathbb{E}_{X\sim\pi}h(X))^2,$$

where $\mathbb{E}_{X\sim\pi}h(X)$ denotes the underlying truth. Three test functions are explored, $h_1(x) = x$, $h_2(x) = x^2$, and $h_3(x) = \cos 2x$, with their corresponding true expectations being $\frac{2}{3}$, 5, and $(\cos 4)/e^2$. The reported run time is also averaged over 100 random initializations.

Figure 2 shows the MSE against different batch sizes for $h_1(x)$, $h_2(x)$, and $h_3(x)$, respectively. The results of RBM-SVGD with different batch sizes are connected by lines, whereas the results of SVGD are the isolated points with batch size $p = 256$. In general, the estimations of $h_1(x)$ and $h_2(x)$ are better than that of $h_3(x)$, which agrees with the difficulty of the problems. Table 1 shows the averaged run time of RBM-SVGD and SVGD for different batch sizes under two different implementations in Matlab. RBM-SVGD is faster than SVGD for all choices of batch size. With respect to the two implementations in Matlab, for the first block row, within each batch, a matrix operation is adopted in computing the kernel matrix

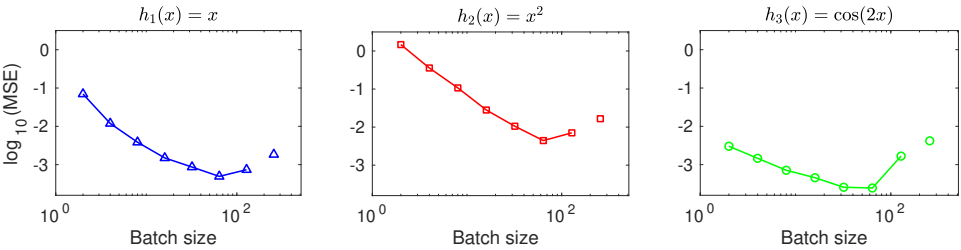

Figure 2. MSEs of (left) $h_1(x) = x$, (center) $h_2(x) = x^2$, and (right) $h_3(x) = \cos 2x$, against different batch sizes.

		RBM-SVGD							SVGD
Matlab	batch size	2	4	8	16	32	64	128	256
matrix op.	run time (s)	0.055	0.095	0.178	0.341	0.270	0.238	0.314	0.733
	speedup (×)	13.3	7.7	4.1	2.1	2.7	3.1	2.3	
row op.	run time (s)	0.055	0.095	0.175	0.332	0.646	1.274	2.527	4.968
	speedup (×)	91.0	52.5	28.4	15.0	7.7	3.9	2.0	

Table 1. Averaged run time for different batch sizes.

whereas for the second block row, the kernel matrix is computed row by row. Matlab naturally is more favorable in the first implementation, which hence achieves fastest run time for all different batch sizes. For other programming languages, e.g., C++, Fortran, etc., the speedup of the second block row is excepted, which is close to ideal case as we predicted earlier.

4.2. ***Double banana.*** In this section, we will compare RBM-SVGD with MCMC, specifically Metropolis–Hastings (MH) [15]. The algorithmic detail of Metropolis–Hastings is available in Appendix B. The performance of RBM-SVGD and MH on a Bayesian inference task is compared to illustrate the advantage of RBM-SVGD. When the number of particles is not very large and desired accuracy is not high, RBM-SVGD can be more efficient.

We run MH and RBM-SVGD on a Bayesian inference task which is exactly the experiment in [10]. In this inference problem, our unknown parameter x is in $\mathbb{R}^2$. The observational data y is a real number which is determined by the forward map $\mathcal{F}(x)$ and the observational noise, i.e., $y = \mathcal{F}(x) + \xi$, where the forward map is a scalar logarithmic Rosenbrock function [24] $\mathcal{F}(x) = \log((1-x_1)^2 + 100(x_2 - x_1^2)^2)$ for $x = (x_1, x_2)$ and the Gaussian noise ξ satisfies $\xi \sim \mathcal{N}(0, \sigma^2)$ for $\sigma = 0.3$. The relationship between parameter x and observation y implies that the likelihood function is $p(y \mid x) = \mathcal{N}(F(x), \sigma^2)$. Finally, we set the prior distribution for x to be Gaussian, i.e., $\pi_0(x) = \mathcal{N}(0, \tau^2 I_2)$, where I_2 is the identity matrix and τ will be

specified later. Thus, the unnormalized posterior density is given by

$$\pi(x) = \pi_0(x)p(y \mid x) = \exp\left(-\frac{\|x\|^2}{2\tau^2} - \frac{(y - F(x))^2}{2\sigma^2}\right). \tag{4-3}$$

$N = 512$ particles are sampled in RBM-SVGD, and the maximum iteration number is 800. Different batch sizes are tested for performance, and the bandwidth parameter is fixed to be $h = 0.1$. To make MH comparable with RBM-SVGD regarding the number of sampling points, we viewed MH as a method with batch size 1, so the total number of iterations we performed for MH was $N \cdot 200$. We apply burn-in technique by only considering the second-half iterations. To reduce correlation, only 1 sample is drawn from every 100 iterations. Therefore, a total number of N samples are selected from MH, which agrees with the number of particles we employ in RBM-SVGD. According to the performance test in Appendix B, we compare RBM-SVGD with MH by choosing $\tau = 5 \cdot 10^{-3}$, which is tested to be convergent and presents the best visual performance among different choices of τ. For both RBM-SVGD and MH, the initial points are sampled from a Gaussian distribution $\mathcal{N}(0, 0.4^2)$. The target distribution is double banana with centers near $(0, 0.5)$ and $(0, -0.5)$. Hence, we adopt two test functions as $h_1(x_1, x_2) = \exp(-(x_1^2 + (x_2 - 0.5)^2)/(2 \cdot 0.5^2))$ and $h_2(x_1, x_2) = \exp(-(x_1^2 + (x_2 + 0.5)^2)/(2 \cdot 0.5^2))$.

In Figure 3, we plotted the position of each particle after RBM-SVGD iteration or MH together with the contour map of the target distribution. From the picture we can tell that both MH and RBM-SVGD can recover the shape of the target density and produce persuasive samplings. Although RBM-SVGD slightly harmed the aggregation of particles around the true distribution (which also paid off with a much shorter running time) compared to the original SVGD (RBM-SVGD with batch size $= 512$), it can still provide a convincing sampling by almost recovering the shape of the target density. In Table 2, we give further quantitative comparison. All numbers in the table are averaged over 100 different initializations. The run time for any RBM-SVGD with different batch sizes is faster than that of MH, and RBM-SVGD with batch size 2 is more than $20\times$ faster while, regarding the MSE for both h_1 and h_2, RBM-SVGD is much better than MH for h_1 and better than MH for h_2. Hence, we conclude, for this example, SVGD outperforms MH both in run time and accuracy. RBM-SVGD further significantly reduces the run time of regular SVGD without loss of accuracy.

4.3. ***Bayesian logistic regression.*** In this experiment, we apply RBM-SVGD to conduct Bayesian logistic regression for binary classification for the Covertype data set with 581012 data points and 54 features [14]. Under the same setting as Gershman [14; 20], the regression weights w of dimension 54 are assigned with a Gaussian prior $p_0(\omega \mid \alpha) = \mathcal{N}(w, \alpha^{-1})$, and the variance satisfies $p_0(\alpha) =$

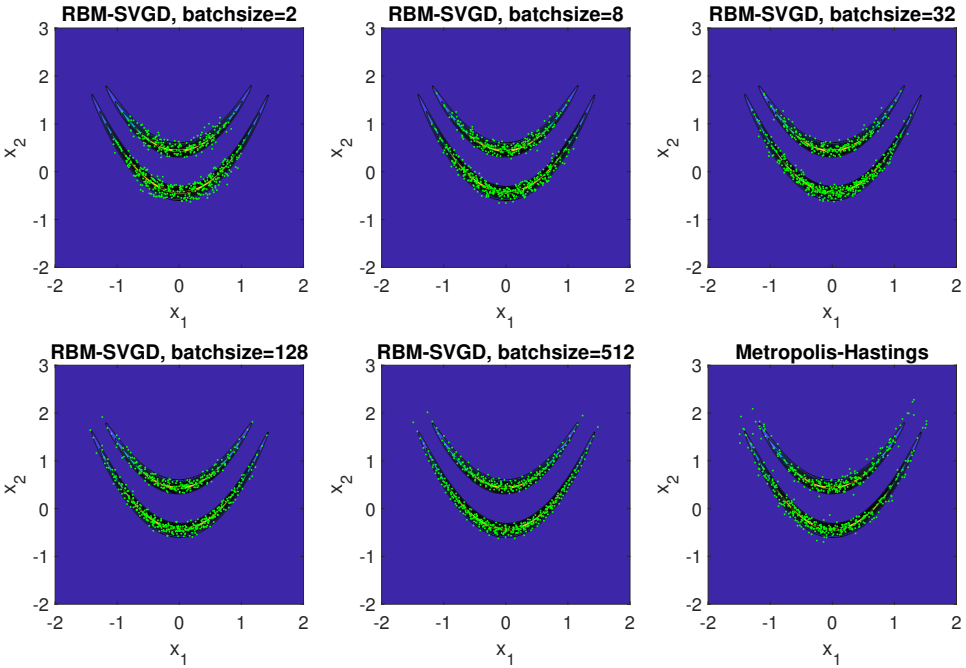

Figure 3. Comparison between RBM-SVGD and Metropolis–Hastings.

	RBM-SVGD					MH
batch size	2	8	32	128	512	
run time (s)	0.1321	0.3960	0.8268	0.9732	1.6086	2.9459
h_1 MSE $\times 10^3$	0.0942	0.1862	0.2270	0.2910	0.3850	6.0689
h_2 MSE $\times 10^3$	0.9559	2.2466	2.0151	1.0617	0.5634	3.5240

Table 2. Run time and MSE of h_1 and h_2 for RBM-SVGD and Metropolis–Hastings.

$\Gamma(\alpha, 1, 0.01)$, where Γ represents the density of Gamma distribution. The inference is applied on posterior $p(x \mid D)$ with $x = [w, \log \alpha]$ of dimension 55. The kernel $K(\cdot)$ is taken again to be the same Gaussian kernel as (4-2).

Since the problem is in high dimension, we adopt $N = 512$ particles in this experiment, which also create more space for the selection of batch sizes. The training is done on 80% of the data set, and the other 20% is used as the test data set. For particle system (1-1), the computation of $-\nabla V = \nabla \log p(x)$ is expensive. Hence, we use the same strategy as mentioned in [20, §3.2], i.e., using data-mini-batch[1] of the data to form a stochastic approximation of $p(x)$ with the data-mini-batch size being 100. Since $\nabla \log p$ depends only on x as in Algorithm 3,

[1]To avoid confusion with our batch of particles, we call it data-mini-batch instead.

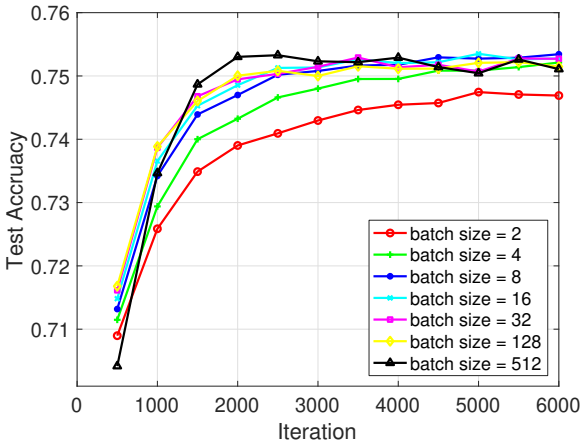

Figure 4. Test accuracy under different batch sizes of RBM-SVGD.

	RBM-SVGD						SVGD
batch size	2	4	8	16	32	128	512
run time (s)	8.59	11.24	16.28	26.15	21.66	19.42	47.01
speedup (×)	5.5	4.2	2.9	1.8	2.2	2.4	

Table 3. Average run time of 6000 iterations.

at each time step, we call this function only once and compute $\nabla \log p$ for all particles, which means the same data-mini-batches are used for $\nabla \log p$ of all particles. In this experiment, we use fixed bandwidth $h = 256$ for RBM-SVGD and dynamic bandwidth strategy for SVGD. The RBM-SVGD uses Algorithm 3 with initial step size being 0.05, and the following step sizes are generated from AdaGrad. Large h is used here for the reason of high dimensionality. Different batch sizes are tested to demonstrate the efficiency of RBM-SVGD. Each configuration is executed on 50 random initializations. The averaged test accuracies for different batch sizes are illustrated in Figure 4.

As shown in Figure 4, RBM-SVGD is almost as efficient as SVGD even for small batch sizes. When $p = 2$, the test accuracy converges to a value slightly off that of SVGD. RBM-SVGD with $p = 4$ converges to the same accuracy as SVGD but at a slower convergent rate. For RBM-SVGD with batch size greater than 4, we observe similar convergence behavior as that of SVGD. The run time of RBM-SVGD, as shown in Table 3, is lower than that of SVGD, where the run time of 6000 iterations is reported. Comparing to the similar run time table for the 1D Gaussian mixture example (Table 1), the acceleration of RBM-SVGD is not as significant as before. This is due to the linear but expensive evaluation of $\nabla \log p$,

	iteration	1000	2000	3000	4000	5000	6000
RBM-SVGD	mean	0.7090	0.7349	0.7409	0.7446	0.7457	0.7471
$p = 2$	std	0.0045	0.0040	0.0040	0.0034	0.0034	0.0038
RBM-SVGD	mean	0.7342	0.7470	0.7508	0.7518	0.7527	0.7534
$p = 8$	std	0.0073	0.0056	0.0041	0.0045	0.0039	0.0033
SVGD	mean	0.7347	0.7530	0.7523	0.7529	0.7504	0.7511
	std	0.0068	0.0048	0.0071	0.0048	0.0061	0.0062

Table 4. Statistics of RBM-SVGD and SVGD.

where RBM-SVGD and SVGD spend the same amount of time in the evaluation each iteration. Although the evaluation of $\nabla \log p$ is expensive, it is linear in N. As N increases, the advantage of RBM-SVGD would be more significant. In Table 4, we list the mean and standard deviation of RBM-SVGD with $p = 2$ and $p = 8$ and SVGD of different iterations. Based on the statistics, we conclude that RBM-SVGD and SVGD are of similar prediction power and RBM-SVGD is efficient also in high-dimensional particle systems as well.

5. Conclusion

We have applied the random batch method for interacting particle systems to SVGD, resulting in RBM-SVGD, which turns out to be a cheap sampling algorithm and inherits the efficiency of the original SVGD algorithm. Theory and numerical experiments have validated the algorithm, and hence, it can potentially have many applications, like Bayesian inference. Moreover, as a hybrid strategy, one may increase the batch size as time goes on to increase the accuracy, or apply some variance reduction approach.

Appendix A: Proof of Lemma 1

Proof of Lemma 1. The proof is pretty much like the one in [16]. We use the random variable $I(i, j)$ to indicate whether i and j are in a common batch. In particular, $I(i, j) = 1$ if i and j are in a common batch while $I(i, j) = 0$ otherwise. Then it is not hard to compute [16]

$$\begin{aligned} \mathbb{E}1_{I(i,j)=1} &= \frac{p-1}{N-1}, \\ \mathbb{P}(I(i,j)I(j,k)=1) &= \frac{(p-1)(p-2)}{(N-1)(N-2)}. \end{aligned} \tag{A-1}$$

We note

$$\chi_i(x) = \frac{1}{N}\sum_{j:j\neq i}\left(1 - \frac{N-1}{p-1}I(i,j)\right)F(x_i, x_j). \tag{A-2}$$

The first equation in (A-1) clearly implies that $\mathbb{E}\chi_i(x) = 0$. Using (A-1), we can compute directly that

$$\mathbb{E}|\chi_i(x)|^2 = \frac{1}{N^2}\Bigg(\sum_{j:j\neq i}\left(\frac{N-1}{p-1}-1\right)|F(x_i,x_j)|^2 + \sum_{j,k:j\neq i,\,k\neq i,\,j\neq k}\left(\frac{(N-1)(p-2)}{(N-2)(p-1)}-1\right)F(x_i,x_k)\cdot F(x_i,x_j)\Bigg).$$

Rearranging this, we get the claimed expression. □

Appendix B: Metropolis–Hastings method and performance

Metropolis–Hastings (MH) is a method of MCMC which produces a reversible Markov chain where the unnormalized target distribution π is invariant. This reversibility is realized by its "accept or reject" machinery. Roughly speaking, MH first generates a candidate according to a proposal distribution (which is always chosen as a normal distribution) and then determines whether to accept or reject the candidate according to the unnormalized target distribution π [15]. In detail, the algorithm has four steps:

(1) *initialization*. Draw X_0 according to a given prior distribution π_0.

(2) *generate a candidate*. Given X_n, draw candidate X' through a normal distribution with mean X_n and covariance C, i.e.,

$$X' = X_n + \mathcal{N}(0, C).$$

(3) *calculate the acceptance rate*. Acceptance rate α is set as

$$\alpha = \min\left\{1, \frac{\pi(X')}{\pi(X)}\right\}.$$

(4) *accept or reject*. Then we accept X' with probability α and reject it with probability $1-\alpha$, i.e., $X_{n+1} = X'$ with probability α and $X_{n+1} = X_n$ with probability $1-\alpha$.

The Markov chain constructed in this algorithm has transition kernel $h(x, y)$ which can be written as

$$h(x,y) = \exp\left(-\frac{(x-y)^T C^{-1}(x-y)}{2}\right)\cdot\min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}.$$

Direct calculation indicates that $h(x, y)\pi(x) = h(y, x)\pi(y)$. Thus, this Markov chain satisfies the detailed balance condition and hence has invariant measure π.

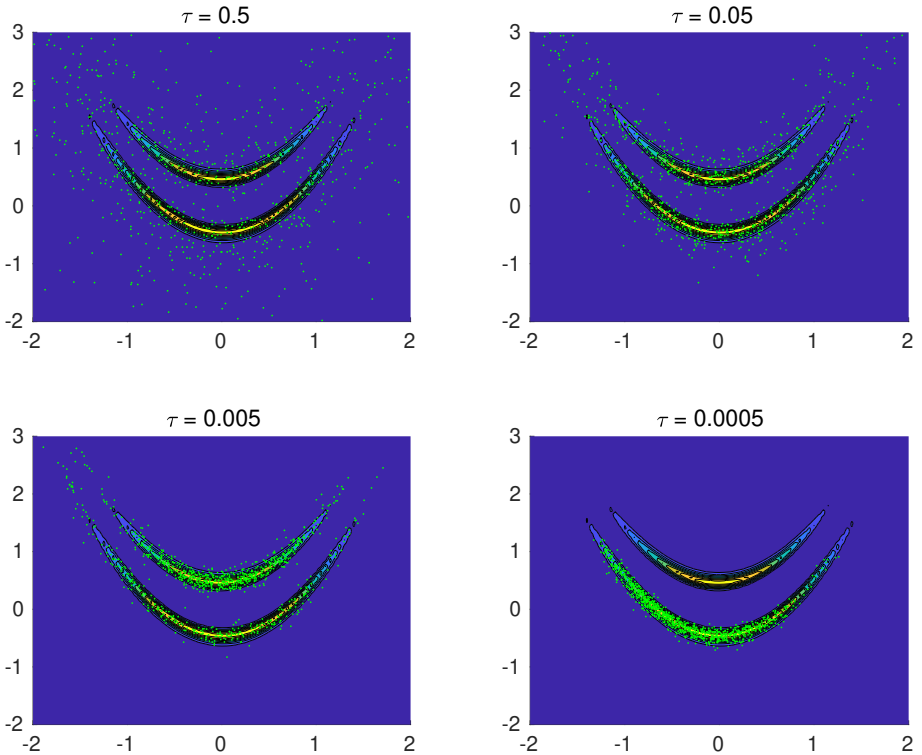

Figure 5. Samples from MH with different τ.

Given the prior and target distribution as in (4-3), we first test the performance of MH among different values of parameter τ, which represents the variance of the proposal distribution.

Figure 5 illustrates the samples together with the contour map of target distribution for $\tau = 5 \cdot 10^{-1}, 5 \cdot 10^{-2}, 5 \cdot 10^{-3}, 5 \cdot 10^{-4}$. Clearly, the best performance was attained when $\tau = 5 \cdot 10^{-3}$. For τ greater than $5 \cdot 10^{-3}$, samples are still wandering around the true distribution without accumulating due to a high variance, whereas for smaller τ, samples are merely aggregating around the "lower" banana rather than the "upper" banana. This phenomenon can be explained by the small variance of proposal distribution, which confines the particles around upper banana.

Moreover, a convergence diagnosis for MCMC was also conducted by computing the auto-correlation [7]. A lower auto-correlation always implies better convergence because a higher auto-correlation indicates that effective sampling size is smaller and more iteration is necessary [27]. Figure 6 plots the auto-correlation of the first coordinate of samples at different time lag κ. For $\tau = 5 \cdot 10^{-1}, 5 \cdot 10^{-2}$, auto-correlation at $\kappa \leq 5 \cdot 10^3$ (2.5% of the number of samples) is plotted, while for $\tau = 5 \cdot 10^{-3}, 5 \cdot 10^{-4}$, auto-correlation at $\kappa \leq 5 \cdot 10^4$ (25% of the number of

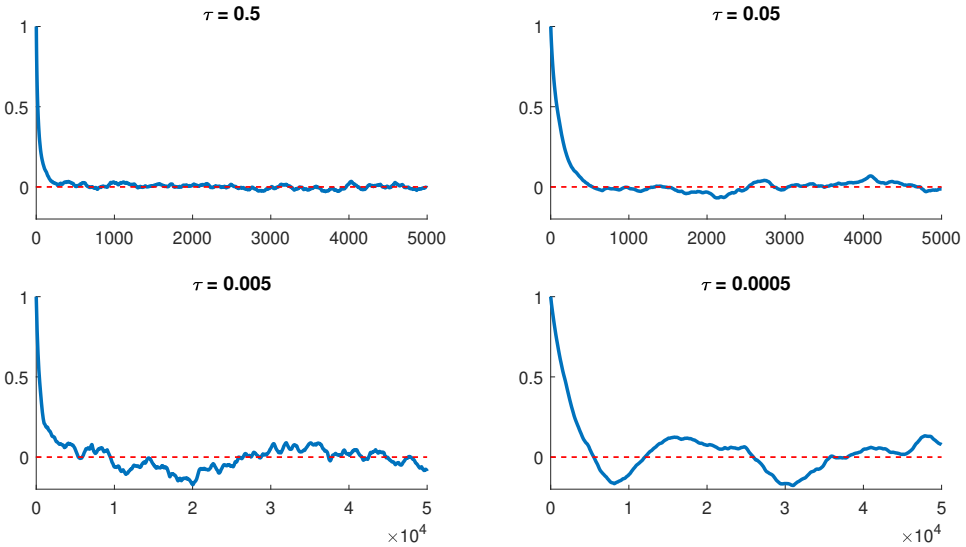

Figure 6. Auto-correlation curve of samples from MH with different τ.

samples) is plotted. This figure shows that auto-correlation decays rapidly for $\tau = 5 \cdot 10^{-1}, 5 \cdot 10^{-2}$ and oscillates around 0 with small magnitude. For $\tau = 5 \cdot 10^{-3}$, although it decays quickly, its oscillation has a greater magnitude. For $\tau = 5 \cdot 10^{-4}$, it does not converge to 0 at all. In conclusion, the convergence of MH with $\tau = 5 \cdot 10^{-1}, 5 \cdot 10^{-2}, 5 \cdot 10^{-4}$ is acceptable. Hence, in this paper, we use MH with $\tau = 5 \cdot 10^{-3}$ as a reference.

Acknowledgements

This work is supported by KI-Net, NSF RNMS11-07444. The work of L. Li was partially sponsored by Shanghai Sailing Program 19YF1421300, the work of Y. Li was partially supported by OAC-1450280, the work of J.-G. Liu was partially supported by NSF DMS-1812573, and the work of J. Lu was supported in part by NSF DMS-1454939.

References

[1] N. Aronszajn, *Theory of reproducing kernels*, Trans. Amer. Math. Soc. **68** (1950), 337–404. MR Zbl

[2] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*, Kluwer, Boston, 2004. MR Zbl

[3] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Variational inference: a review for statisticians*, J. Amer. Statist. Assoc. **112** (2017), no. 518, 859–877. MR

[4] L. Bottou, *On-line learning and stochastic approximations*, On-line learning in neural networks (D. Saad, ed.), Cambridge University, 1998, pp. 9–42. Zbl

[5] G. E. P. Box and G. C. Tiao, *Bayesian inference in statistical analysis*, Addison-Wesley, Reading, MA, 1973. MR Zbl

[6] S. Bubeck, *Convex optimization: algorithms and complexity*, Found. Trends Machine Learn. **8** (2015), no. 3–4, 231–357. Zbl

[7] C. Chatfield, *The analysis of time series: an introduction*, 6th ed., Chapman & Hall/CRC Texts in Statistical Science Series, Chapman & Hall/CRC, Boca Raton, FL, 2004. MR Zbl

[8] C. Chen, R. Zhang, W. Wang, B. Li, and L. Chen, *A unified particle-optimization framework for scalable Bayesian sampling*, Uncertainty in Artificial Intelligence: proceedings of the thirty-fourth conference (Monterey, CA, 2018) (A. Globerson and R. Silva, eds.), AUAI, Corvallis, OR, 2018, p. 263.

[9] B. Dai, N. He, H. Dai, and L. Song, *Provable Bayesian inference via particle mirror descent*, Artificial intelligence and statistics (Cádiz, Spain, 2016) (A. Gretton and C. C. Robert, eds.), Proceedings of Machine Learning Research, no. 51, 2016, pp. 985–994.

[10] G. Detommaso, T. Cui, Y. Marzouk, A. Spantini, and R. Scheichl, *A Stein variational Newton method*, Advances in Neural Information Processing Systems 31 (Montréal, 2018), NIPS Proceedings, 2018.

[11] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, J. Mach. Learn. Res. **12** (2011), 2121–2159. MR Zbl

[12] D. Francois, V. Wertz, and M. Verleysen, *About the locality of kernels in high-dimensional spaces*, Applied Stochastic Models and Data Analysis (Brest, France, 2005) (J. Janssen and P. Lenca, eds.), ENST Bretagne, 2005, pp. 238–245.

[13] D. Frenkel and B. Smit, *Understanding molecular simulation: from algorithms to applications*, 2nd ed., Academic, 2002. Zbl

[14] S. Gershman, M. Hoffman, and D. Blei, *Nonparametric variational inference*, Proceedings of the 29th International Conference on Machine Learning (Edinburgh, 2012), Omnipress, Madison, WI, 2012, pp. 235–242.

[15] W. K. Hastings, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika **57** (1970), no. 1, 97–109. MR Zbl

[16] S. Jin, L. Li, and J.-G. Liu, *Random batch methods (RBM) for interacting particle systems*, J. Comput. Phys. **400** (2020), art. id. 108877. MR

[17] K. Law, A. Stuart, and K. Zygalakis, *Data assimilation: a mathematical introduction*, Texts in Applied Mathematics, no. 62, Springer, 2015. MR Zbl

[18] C. Liu and J. Zhu, *Riemannian Stein variational gradient descent for Bayesian inference*, Thirty-Second AAAI Conference on Artificial Intelligence (New Orleans, 2018), Association for the Advancement of Artificial Intelligence, 2018, pp. 3627–3634.

[19] Q. Liu, *Stein variational gradient descent as gradient flow*, Advances in Neural Information Processing Systems 30 (Long Beach, CA, 2017), NIPS Proceedings, 2017.

[20] Q. Liu and D. Wang, *Stein variational gradient descent: a general purpose Bayesian inference algorithm*, Advances in Neural Information Processing Systems 29 (Barcelona, 2016), NIPS Proceedings, 2016.

[21] J. Lu, Y. Lu, and J. Nolen, *Scaling limit of the Stein variational gradient descent: the mean field regime*, SIAM J. Math. Anal. **51** (2019), no. 2, 648–671. MR Zbl

[22] D. Rezende and S. Mohamed, *Variational inference with normalizing flows*, International Conference on Machine Learning (Lille, France, 2015) (F. Bach and D. Blei, eds.), Proceedings of Machine Learning Research, no. 37, 2015, pp. 1530–1538.

[23] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Statistics **22** (1951), 400–407. MR Zbl

[24] H. H. Rosenbrock, *An automatic method for finding the greatest or least value of a function*, Comput. J. **3** (1960), 175–184. MR

[25] W. Rudin, *Fourier analysis on groups*, Interscience Tracts in Pure and Applied Mathematics, no. 12, Interscience, New York, 1962. MR Zbl

[26] F. Santambrogio, *Optimal transport for applied mathematicians: calculus of variations, PDEs, and modeling*, Progress in Nonlinear Differential Equations and their Applications, no. 87, Springer, 2015. MR Zbl

[27] A. Sokal, *Monte Carlo methods in statistical mechanics: foundations and new algorithms*, Functional integration (Cargèse, 1996) (C. DeWitt-Morette, P. Cartier, and A. Folacci, eds.), NATO Adv. Sci. Inst. Ser. B Phys., no. 361, Plenum, New York, 1997, pp. 131–192. MR Zbl

[28] R. Ward, X. Wu, and L. Bottou, *AdaGrad stepsizes: sharp convergence over nonconvex landscapes*, International Conference on Machine Learning (Long Beach, CA, 2019) (K. Chaudhuri and R. Salakhutdinov, eds.), Proceedings of Machine Learning Research, no. 97, 2019, pp. 6677–6686.

[29] M. Welling and Y. W. Teh, *Bayesian learning via stochastic gradient Langevin dynamics*, Proceedings of the 28th International Conference on International Conference on Machine Learning (L. Getoor and T. Scheffer, eds.), Omnipress, Madison, WI, 2011, pp. 681–688.

Received April 10, 2019. Revised November 12, 2019.

LEI LI: leili2010@sjtu.edu.cn
School of Mathematical Sciences, Institute of Natural Sciences, Key Lab of Scientific and Engineering Computing, Ministry of Education, Shanghai Jiao Tong University, Shanghai, China

YINGZHOU LI: yingzhou.li@duke.edu
Department of Mathematics, Duke University, Durham, NC, United States

JIAN-GUO LIU: jliu@phy.duke.edu
Department of Mathematics, Department of Physics, Duke University, Durham, NC, United States

ZIBU LIU: zibu.liu@duke.edu
Department of Mathematics, Duke University, Durham, NC, United States

JIANFENG LU: jianfeng@math.duke.edu
Department of Mathematics, Department of Physics, Department of Chemistry, Duke University, Durham, NC, United States

Communications in Applied Mathematics and Computational Science

msp.org/camcos

EDITORS

MANAGING EDITOR

John B. Bell
Lawrence Berkeley National Laboratory, USA
jbbell@lbl.gov

BOARD OF EDITORS

Marsha Berger	New York University berger@cs.nyu.edu	Ahmed Ghoniem	Massachusetts Inst. of Technology, USA ghoniem@mit.edu
Alexandre Chorin	University of California, Berkeley, USA chorin@math.berkeley.edu	Raz Kupferman	The Hebrew University, Israel raz@math.huji.ac.il
Phil Colella	Lawrence Berkeley Nat. Lab., USA pcolella@lbl.gov	Randall J. LeVeque	University of Washington, USA rjl@amath.washington.edu
Peter Constantin	University of Chicago, USA const@cs.uchicago.edu	Mitchell Luskin	University of Minnesota, USA luskin@umn.edu
Maksymilian Dryja	Warsaw University, Poland maksymilian.dryja@acn.waw.pl	Yvon Maday	Université Pierre et Marie Curie, France maday@ann.jussieu.fr
M. Gregory Forest	University of North Carolina, USA forest@amath.unc.edu	James Sethian	University of California, Berkeley, USA sethian@math.berkeley.edu
Leslie Greengard	New York University, USA greengard@cims.nyu.edu	Juan Luis Vázquez	Universidad Autónoma de Madrid, Spain juanluis.vazquez@uam.es
Rupert Klein	Freie Universität Berlin, Germany rupert.klein@pik-potsdam.de	Alfio Quarteroni	Politecnico di Milano, Italy alfio.quarteroni@polimi.it
Nigel Goldenfeld	University of Illinois, USA nigel@uiuc.edu	Eitan Tadmor	University of Maryland, USA etadmor@cscamm.umd.edu
		Denis Talay	INRIA, France denis.talay@inria.fr

PRODUCTION

production@msp.org

Silvio Levy, Scientific Editor

See inside back cover or msp.org/camcos for submission instructions.

The subscription price for 2020 is US $110/year for the electronic version, and $165/year (+$15, if shipping outside the US) for print and electronic. Subscriptions, requests for back issues from the last three years and changes of subscriber address should be sent to MSP.

Communications in Applied Mathematics and Computational Science (ISSN 2157-5452 electronic, 1559-3940 printed) at Mathematical Sciences Publishers, 798 Evans Hall #3840, c/o University of California, Berkeley, CA 94720-3840, is published continuously online. Periodical rate postage paid at Berkeley, CA 94704, and additional mailing offices.

CAMCoS peer review and production are managed by EditFLOW® from MSP.

PUBLISHED BY

mathematical sciences publishers

nonprofit scientific publishing

http://msp.org/

© 2020 Mathematical Sciences Publishers

Communications in Applied Mathematics and Computational Science

vol. 15 no. 1 2020