EXPLORATION TOWARD STEIN VARIATIONAL GRADIENT DESCENT

CHANGHAO GE, ZHEMING CAO

Contents

1. Introduction 1
2. Preliminaries 2
3. SVGD 3
4. pSVGD 4
5. Experiments 6
5.1. Gaussian Mixture Model 7
5.2. Comparison with EM 8
5.3. Multivariate Normal Model 10
5.4. Conditional diffusion process 11
References 11

1. Introduction

The method in this report is mainly based on two papers of Bayesian inference: **Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm**[5] and **Projected Stein Variational Gradient Descent**[1]. The article [4] and [2] provide some ideas of experiments for our report.

Stein Variational Gradient Descent (SVGD) is a method to iterate a set of particles to its posterior distribution by regarding KL divergence as a metric and applying the corresponding functional gradient descent. However, SVGD faces the curse of dimensionality, and thus is not computationally efficient when the dimension of particles is too large. Projected Stein Variational Gradient Descent (pSVGD) tackles this problem by projecting the parameters to a lower-dimensional subspace. The subspace is constructed by a gradient information matrix of the log-likelihood, which reveals intrinsic low dimensionality of the data.

We briefly introduced the knowledge and theory needed for these two methods, finished the algorithms in these two articles with R, and improved the computational performance with Rcpp. In addition, we explored the rate of convergence of SVGD under some mild conditions, which is not mentioned in the two papers,

Date: March 28, 2021.

and proved it in the appendix. Besides, we implemented practical experiment with multivariate normal and Gaussian mixture models, and compared them with EM algorithm. The result shows SVGD and pSVGD algorithms have better performance. Finally, we made an R package and uploaded it on Github.

Contribution: Changhao Ge finished Algorithm 1 and 2 and experiments 5.1 and 5.2, rewrote Algorithm 3 by Rcpp, finished the proof of Theorem 3, and compiled the report. The contribution ratio is about 50%.

Zheming Cao finished Algorithm 3 and experiment 5.3 and 5.4, wrote the R package, and little part of the report. The contribution ratio is about 50%.

2. Preliminaries

Let $x \in \mathbb{R}^d$ is a random parameter of dimension d with a continuous prior distribution $p_0 : \mathbb{R}^d \to \mathbb{R}$.

Let $y = \{y_i\}_{i=1}^s$ denote the i.i.d observation data.

Let $f(x) := \prod_{i=1}^s p(y_i|x)$ denote a continuous likelyhood of y at given x(up to a irrelavant constant).

Then the posterior density of parameter x is given by Bayes' rule:

$$p(x) = \frac{1}{Z} f(x) p_0(x) \tag{1}$$

where Z is the normalization constant:

$$Z = \int_{\mathbb{R}^d} f(x) p_0(x) dx \tag{2}$$

Stein's Identity Let $\mathcal{A}_p \phi(x) = \phi(x) \nabla_x \log p(x)^\top + \nabla_x \phi(x)$,then we have

$$\mathbf{E}_{x \sim p}[\mathcal{A}_p \phi(x)] = 0 \tag{3}$$

when ϕ is good enough(we call in Stein class), and

$$\mathbf{E}_{x \sim q}[\mathcal{A}_p \phi(x)] \neq 0 \tag{4}$$

for general ϕ when $p \neq q$. So this could determine a metric between two distributions p and q.

KL divergence Similar to the metric above, the KL divergence:

$$\mathbf{D}_{KL}(q|p) := \mathbf{E}_{x \sim q}[\log(q/p)] \tag{5}$$

can also define a metric between two distributions p an q. Noticeably, the metric is not symmetric w.r.t p and q.

Functional Gradient For any functional $F[\boldsymbol{f}]$ of $\boldsymbol{f} \in \mathcal{H}^d$, its (functional) gradient $\nabla_{\boldsymbol{f}} F[\boldsymbol{f}]$is a function in $\mathcal{H}^d$ such that $F[\boldsymbol{f} + \epsilon \boldsymbol{g}(x)] = F[\boldsymbol{f}] + \epsilon \langle \nabla_{\boldsymbol{f}} F[\boldsymbol{f}], \boldsymbol{g} \rangle_{\mathcal{H}^d} + O(\epsilon^2)$ for any $\boldsymbol{g}$ in $\mathcal{H}^d$ and $\epsilon \in \mathbb{R}$.

Reducing Kernel Hilbert Space Suppose H is a Hilbert space whose elements are convex-value functions on set X. If H satisfies: $\forall x \in X$, exists a unique function $K_x(y) \in H$ such that

$$\langle f, K_x \rangle = f(x), \quad f \in H \tag{6}$$

then we call H a Reducing Kernel Hilbert Space(RKHS). Function $K(x, y) := K_y(x)$ is said to be the reproducing kernel of H.

Equation 6 is called the reproducing property of RKHS. Note that this property is one of the most important nature of RKHS.

3. SVGD

The key point of Stein Variational Gradient Descent(SVGD) is to estimate posterior distribution p by optimal $q* \in \mathcal{Q}$, such that

$$q* = \arg\min_{q\in\mathcal{Q}} \mathbf{D}_{KL}(q|p) \tag{7}$$

where $\mathcal{Q}$ is a family of distributions having good properties.

We take $\mathcal{Q}$ to be the set of distributions of form $z = \mathbf{T}(x)$ where $\mathbf{T} : X \to X$ is a smooth one-to-one transfor, and x has density $q_0(x)$. By the chain rule, the density of z is

$$q_{[\mathbf{T}]}(z) = q(\mathbf{T}^{-1}(z)) \cdot |\det(\nabla_z \mathbf{T}^{-1}(z))|$$

where $\mathbf{T}^{-1}$ denotes the inverse map of $\mathbf{T}$ and $\nabla_z \mathbf{T}^{-1}$ the Jacobian matrix of $\mathbf{T}^{-1}$.

In order to implement gradient descent, we consider a small vibration of the identity map: $\mathbf{T}(x) = x + \epsilon\phi(x)$, where $\phi(x)$ is a smooth function characterizing the disturbance direction and the scalar ϵ represents the magnitude. In this work we take $|\epsilon|$ so small that the Jacobian of $\mathbf{T}$ is full rank to guarantee an one-to-one map by the inverse function theorem.

The following result in[5] draws a connection between Stein operator and the derivative of KL divergence w.r.t the disturbance magnitude ϵ.

Theorem 1. *Let $\mathbf{T}(x) = x + \epsilon\phi(x)$ and $q_{[\mathbf{T}]}(z)$ the density of $z = \mathbf{T}(x)$ when $x \sim q(x)$, we have*

$$\nabla_\epsilon \mathbf{D}_{KL}(q_{[\mathbf{T}]}(z)\|p)|_{\epsilon=0} = -\mathbf{E}_{x\sim q}[trace(\mathcal{A}_p\phi(x))] \tag{8}$$

where $\mathcal{A}_p\phi(x) = \nabla_x \log p(x)\phi(x)^\top + \nabla_x\phi(x)$ is the stein operator.

The theorem is describes the stepsize when the disturbance direction ϕ is fixed. The theorem below gives a criterion of optimal direction ϕ.

Theorem 2. *let $\boldsymbol{T}(x) = x + \boldsymbol{f}(x)$, where $\boldsymbol{f} \in \mathcal{H}^d$, and $q_{[\boldsymbol{T}]}$ the density of $z = \boldsymbol{T}(x)$ when $x \sim q$,*

$$\nabla_{\boldsymbol{f}} \mathbf{D}_{KL}(q_{[\boldsymbol{T}]}\|p)|_{\boldsymbol{f}=0} = -\phi^*_{q,p}(x) \tag{9}$$

where

$$\phi^*_{q,p}(\cdot) = \mathbf{E}_{x\sim q}[\mathcal{A}_p k(s, \cdot)] \tag{10}$$

Define KSD as below:

$$\mathbf{S}(q,p) = \max_{\phi\in\mathcal{H}^d}\{[\mathbf{E}_{x\sim q}(trace(\mathcal{A}_p\phi(x)))]^2, \quad s.t \quad \|\phi\|_{\mathcal{H}^d} \le 1\} \tag{11}$$

Applying **Theorem 2** to our problem, we have:

Lemma 1. *Assume the conditions in Theorem 5. Consider all the perturbation directions ϕ in the ball $\mathcal{B} = \{\phi \in \mathcal{H}^d : \|\phi\|^2_{\mathcal{H}^d} \leq \mathbf{S}(q,p)\}$ of vector-valued RKHS $\mathcal{H}^d$, the direction of steepest descent that maximizes the negative gradient in 9 is the $\phi^*_{q,p}$ in 10, i.e.,*

$$\phi^*_{q,p}(\cdot) = \mathbf{E}_{x\sim q}[k(x,\cdot)\nabla_x \log p(x) + \nabla_x k(x,\cdot)] \tag{12}$$

for which the negative gradient in 9 equals KSD, that is, $\nabla_\epsilon \mathbf{D}_{KL}(q_{[\mathbf{T}]}\|p)|_{\epsilon=0} = -\mathbf{S}(q,p)$

Then we can make Bayesian Inference via this gradient descent algorithm. Note that $\hat{\phi}^*(x)$ is the approximation of function $\phi^*_{q,p}(x)$ for the later one is hard to compute with an integral.

Algorithm 1 Stein Variational Gradient Descent

```
Input: A target distribution with density function $p(x)$ and a set of initial particles $\{x_i^0\}_n^{i=1}$.
Output: A set of particles $\{x_i\}_{i=1}^n$ that approximates the target distribution.
for iteration $\ell$ do
    (13)
    $x_i^{\ell+1} \leftarrow x_i^\ell + \epsilon_\ell \hat{\phi}^*(x_i^\ell) \quad \text{where} \quad \hat{\phi}^*(x) = \frac{1}{n}\sum_{j=1}^n [k(x_j^\ell, x)\nabla_{x_j^\ell} \log p(x_j^\ell) + \nabla_{x_j^\ell} k(x_j^\ell, x)]$
    where $\epsilon_\ell$ is the step size at the $\ell$-th iteration.
```

Convergence Property of SVGD The paper [5] doesn't show the convergence conditions of SVGD. However, we prove that under some mild conditions, the convergence of SVGD is guaranteed. Note that our choice of kernel is $k(x,y) := \exp(-\frac{1}{h}\|x-y\|_2^2)$, and suppose the domain of x is bounded, say $|x| \leq M$.

CONDITION 1. $\log p(x)$ is concave and gradient Lipschitz continuous. L is the Lipschitz constant.

CONDITION 2. The gradient of perturbation family $\mathcal{F}$ is small enough such that for every $f \in \mathcal{F}$, $\|(I - \nabla_x f(x))^{-1}\|_F^2 \leq a$ for some $a \in \mathbb{R}^+$.

Theorem 3. *Under* CONDITION 1 *and* CONDITION 2*, the the sequence $\{f^k\}$ derived from SVGD converges to the optimal solution in the case that stepsize $\epsilon \leq 1/(K(L + \frac{4aM}{h}))$. Moreover, the convergence rate is $O(k)$ in the sense of functional value.*

4. pSVGD

Define $H \in \mathbb{R}^{d\times d}$, which is called the gradient information matrix representing the average outer product of the gradient of the log-likelihood w.r.t the posterior, as below:

$$H = \int_{\mathbb{R}^d} (\nabla_x \log f(x))(\nabla_x \log f(x))^\top p(x)dx \tag{14}$$

By $(\lambda_i, \psi_i)_{i=1}^r$ we denote the r eigen pairs of (H, Γ) having largest eigenvalues, with Γ representing the covariance of the parameter x w.r.t its prior.

$$H\psi_i = \lambda_i \Gamma \psi_i \tag{15}$$

Then define a projector (also can be seen as a project matrix) of rank r, $P_r : \mathbb{R}^d \to \mathbb{R}^d$ as

$$P_r x; = \sum_{i=1}^{r} \psi_i \psi_i^\top x = \Psi_r w, \quad \forall x \in \mathbb{R}^d \tag{16}$$

where $\Psi_r := (\psi_1, \psi_2, \ldots, \psi_r) \in \mathbb{R}^{d\times r}$ represents the projection matrix w.r.t w and $w := (w_1, w_2, \ldots, w_r)^\top \in \mathbb{R}^r$ is a coefficient vector with element $w_i := \psi_i^\top x$ for $i = 1, 2, \ldots, r$.

Given w, we want to find a function $g : \mathbb{R}^d \to \mathbb{R}$, which receive projected vector and is a good approximation of likelihood function $f(x)$,i.e. $g(P_r x) \approx f(x)$. Suppose we have found the optimal g, and define the projected density $p_r : \mathbb{R}^d \to \mathbb{R}$:

$$p_r(x) := \frac{1}{Z_r} g(P_r x) p_0(x) \tag{17}$$

where $Z_r := \mathbf{E}_{x\sim p_0}[g(P_r x)]$. It's proven that an optimal $g = g^*$ exists such that

$$\mathbf{D}_{KL}(p|p_r^*) \leq \mathbf{D}_{KL}(p|p_r) \tag{18}$$

when p_r^* is defined as in 17 with g^*.

Moreover, under some mild conditions one can show that

$$\mathbf{D}_{KL}(p|p_r^*) \leq \frac{\gamma}{2} \sum_{i=r+1}^{d} \lambda_i \tag{19}$$

for a constanc γ independent of r.

Now the fact is that the optimal profile function g^* is the marginal likelihood, i.e.

$$g^*(P_r x) = \int_{X_\perp} f(P_r x + \xi) p_0^\perp(\xi | P_r x) d\xi \tag{20}$$

where $X_\perp$ is the complement space of $X_r := span(\psi_1, \psi_2, \ldots, \psi_r)$ and

$$p_0^\perp(\xi|P_r x) = p_0(P_r x + \xi)/p_0^r(P_r x) \quad \text{with} \quad p_0^r(P_r x) = \int_{X_\perp} p_0(P_r x + \xi) d\xi \tag{21}$$

Then the prior distribution can be rewritten as

$$p_0(x) = p_0^r(x^r) p_0^\perp(x^\perp | x^r) \tag{22}$$

and we define

$$\pi_0(w) = p_0^r(\Psi_r w) \qquad \pi(w) = \frac{1}{Z_w} g^*(\Psi_r w) \pi_0(w) \tag{23}$$

where Z_w is a normalization constant.

Then the projected distribution in 17 can be expressed as

$$p_r(x) = \pi(w) p_0^{\perp}(x^{\top}|\Psi_r w) \tag{24}$$

Then we only need to optimize $\pi(w)$ and $p_0^{\perp}(x^{\top}|\Psi_r w)$ respectively. In fact, $\pi(w)$ can be processed by SVGD method, so we only need to think about the later one. Moreover, we have the properties below:

$$\nabla_w \log \pi(w) = \Psi_r^{\top}(\frac{\nabla_x g(P_r x)}{g(P_r x)} + \frac{\nabla_x p_0^r(P_r x)}{p_0^r(P_r x)}) \tag{25}$$

which links w in pSVGD and x in SVGD.

Now we deal with some difficulties in this algorithm.

g^* in 20 involve high-dimensional integrals. One possible solution is to approximate it by

$$g^*(P_r x_n^l) \approx f(P_r x_n^l + x_n^{\perp})$$

where $x_n^{\perp} = x_n^0 - P_r x_n^0$ is seen as a sample from distribution $p_0^{\perp}(x^{\perp}|P_r x)$.

The second problem is the calculation of H in 15. We also approximate it by

$$\hat{H} := \frac{1}{M}\sum_{m=1}^{M}(\nabla_x \log f(x_m))(\nabla_x \log f(x_m))^{\top} \tag{26}$$

Algorithm 2 projected Stein Variational Gradient Descent

```
Input: samples $\{x_n^0\}_{n=1}^N\}$, basis $\Psi_r$, maximum iteration $L_{\max}$,tolerance $w_{tol}$.
Output: posterior samples $\{x_n^*\}_{n=1}^N$.
Set $\ell = 0$, project $w_n^0 = \Psi_r^{\top} x_n^0, x_n^{\perp} = x_n^0 - \Psi_r w_n^0$.
repeat
    Compute gradients $\nabla_{w_n^\ell} \log \pi(w_n^\ell)$ by 25 for $n = 1, \cdots, N$.
    Compute the kernel values $k^r(w_n^\ell, w_m^\ell)$ and their gradients $\nabla_{w_n^\ell} k^r(w_n^\ell, w_m^\ell)$
for $n = 1, \cdots, N, m = 1, \cdots, N$.
    Update samples $w_m^{\ell+1}$ from $w_m^\ell$ by Algorithm 1, i.e.
(27)            $\{w_m^{\ell+1}\}_{n=1}^N = SVGD(\{w_m^\ell\}_{n=1}^N)$
    Set $\ell \leftarrow \ell + 1$
until $\ell \geq L_{\max}$ or mean$(\|w_m^\ell - w_m^{\ell+1}\|_2) \leq w_{tol}$.
Reconstruct samples $x_n^* = \Psi_r w_n^\ell + x_n^{\perp}$.
```

5. Experiments

In this chapter, we implemented numerical experiments with Gaussian Mixture Model and Multivariate Normal Model, and compared our methods with EM algorithm. It turns out that EM algorithm faces the challenge of sensitive dependence on initial conditions in Gaussian Mixture Model. In addition, we compared the

Algorithm 3 Adaptive projected Stein Variational Gradient Descent

```
**Input:** samples $\{x_n^0\}_{n=1}^N, L_{\max}^x, L_{\max}^w, x_{tol}, w_{tol}$.
**Output:** posterior samples $\{x_n^*\}_{n=1}^N$.
Set $\ell_x = 0$
**repeat**
    Compute $\nabla_x \log f(x_n^{\ell_x})$ in 26 for $n = 1, \cdots, N$.
    Solve 15 with $H$ approximated as in 26, to get bases $\Psi_r^{\ell_x}$.
    Apply the pSVGD Algorithm 2, i.e.
(28)        $\{x_n^*\}_{n=1}^N = \mathrm{pSVGD}(\{x_n^{\ell_x}\}_{n=1}^N, \Psi_r^{\ell_x}, L_{\max}^w, w_{tol})$.
    Set $\ell_x \leftarrow \ell_x + 1$ and $x_n^{\ell_x} = x_n^*, n = 1, \cdots, N$.
**until** $\ell_x \geq L_{\max}^x$ or mean$(\|x_m^{\ell_x} - x_m^{\ell_x - 1}\|_X) \leq x_{tol}$.
```

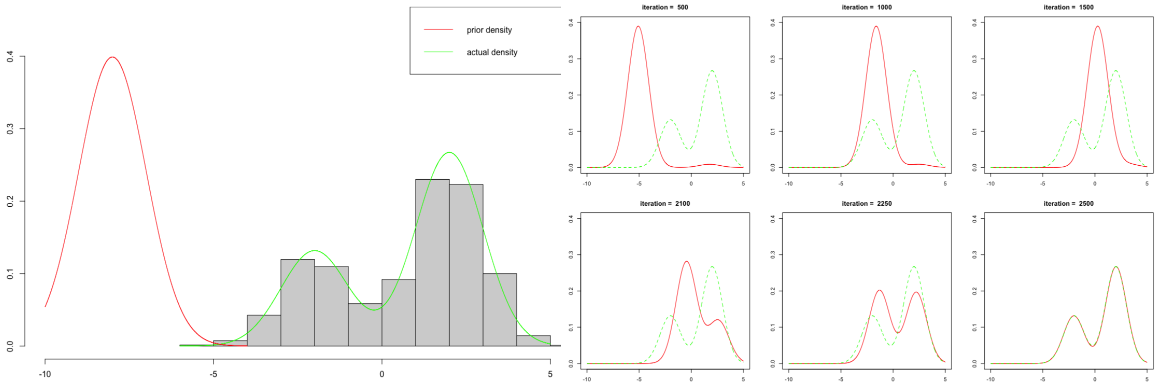
time efficiency of SVGD with pSVGD in two ways: in the same iterations and in the same tolerance.

5.1. **Gaussian Mixture Model.** Let $\{y_i\}$ is a set of i.i.d samples generated by a Gaussian mixture model

$$f(y_i) = \alpha \mathcal{N}(y_i; \mu_1, 1) + (1 - \alpha)\mathcal{N}(y_i; \mu_2, 1), \quad i = 1, \cdots, 2000$$

where $\mathcal{N}$ represents the density of a normal distribution. Set $\mu_1 = -2, \mu_2 = 2, \alpha = 0.33$.

Set prior density of $\mu_1 \sim N(-8, 1), \mu_2 \sim N(4, 1)$. The number of parameter particles is 10.



(a) (b)

FIGURE 1. (a): initial conditions of prior density (b): posterior density during iteration

From figure 1(a) we can tell that the choose of initial condition is extraordinarily ill since the probability mass of $p(x)$ and $q_0(x)$ are far away each other (with almost

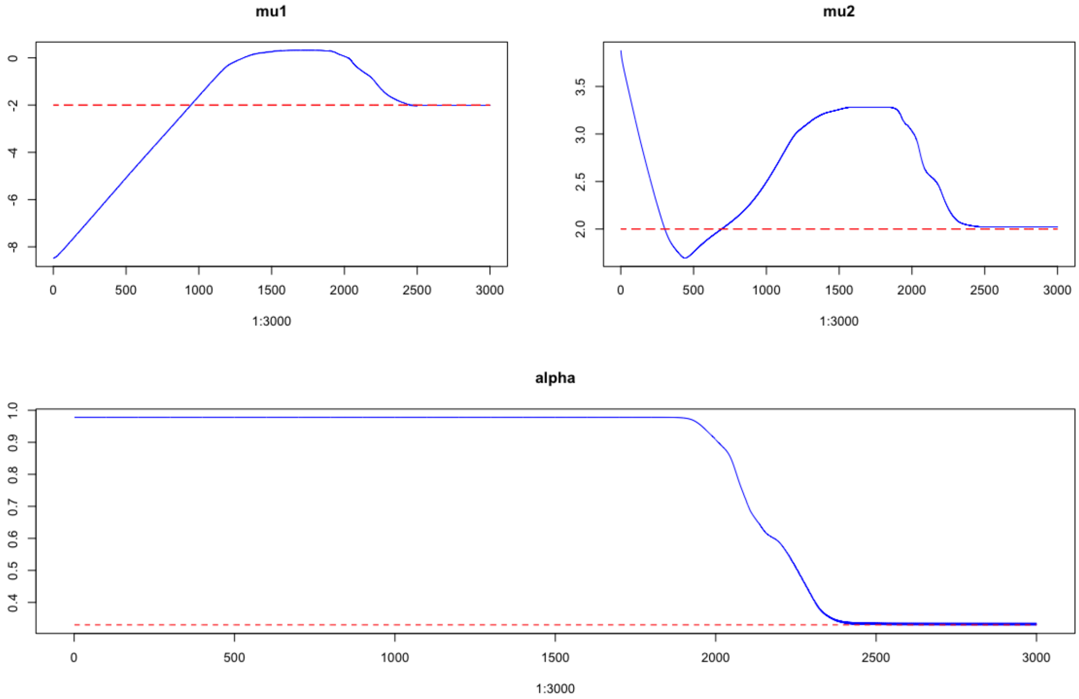


Figure 2. parameter value during iteration

zero overlap). Figure 1(b) showshow the distribution of the particles change as the iteration goes. We see that despite the small overlap between $q_0(x)$ and $p(x)$, SVGD can push the particle towards the target distribution, and the final result meets our expectation.

Figure 2 shows how the parameter changes step by step - first μ_1 and μ_2 and then α. After about 2500 iterations the parameter converges to real value.

5.2. Comparison with EM.

We set the same distribution of $\{y_i\}_{n=1}^{2000}$ as in former section, and did the same numerical experiment using EM algorithm.

From figure 3 we can tell that when the initial condition is good enough, the speed of EM algorithm in GMM model is really fast - after about thirty iterations, μ_1 and μ_2 converge to the real value. However, EM algorithm faces the challenge of sensitive dependence on initial condition. EM algorithm converges to a false value when we change the initial value of α from 0.9 to 0.91. This is because EM algorithm only maximizes the target function $Q(\theta|\theta_n) = \mathbf{E}[\log f(X|\theta)|Y = y, \theta_n]$ at each step. This procedure finds the optimal θ informing the maximum likelihood estimation in the sense of expectation, or in other words, mean value, rather than the distribution itself. So when the question is not convex, the result may be totally different from true value. However, SVGD measures two distributions with

KL divergence, and the metric vanishes iff the two distributions are equal. This property guarantees a better performance of SVGD method.

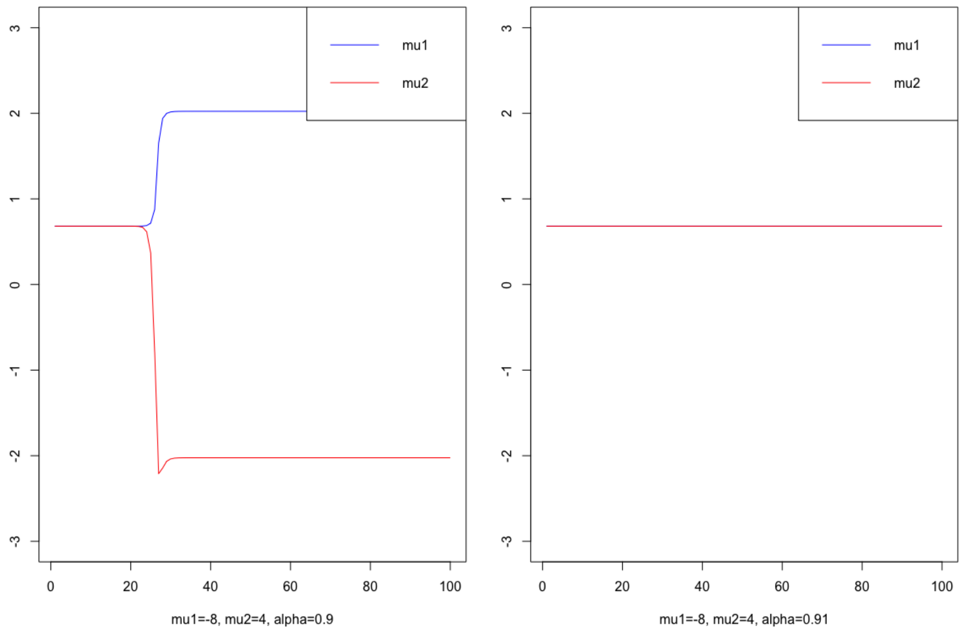
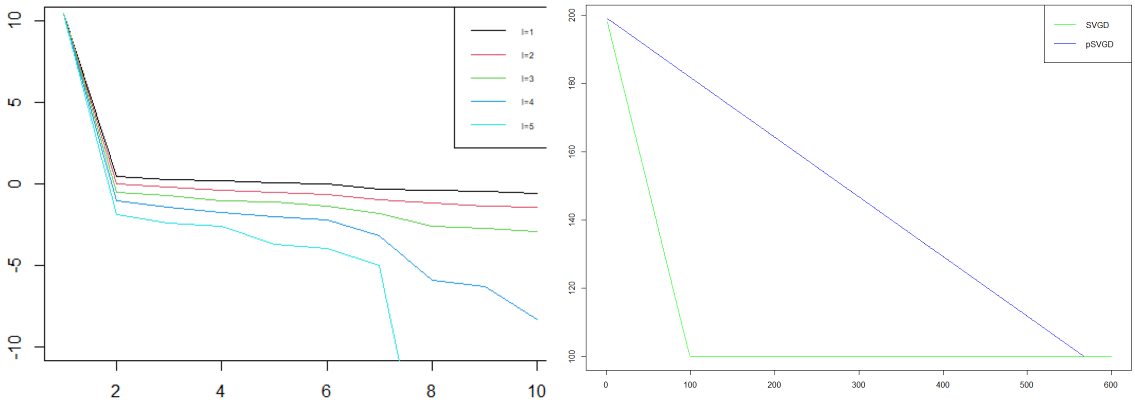


FIGURE 3. the performance of EM algorithm in different initial conditions



(a)

(b)

FIGURE 4. (a): top 10 log eigenvalues in first 5 iterations (b): iteration speed of SVGD and pSVGD

5.3. **Multivariate Normal Model.** We use both SVGD and pSVGD in Multivariate Normal Model, and compares their behavior.

Set Set $y \sim N(\mu, I_{100}), i = 1, 2, \cdots, n$, where $\mu = (1, 2, \cdots, 100)$.

Figure 4 (a) shows the biggest log eigenvalues in first 5 iterations. It's shown that the first eigenvalue remains relatively still, while smaller ones decrease fast, even in first several iterations. This gives us the information that it's feasible to project 100-dimensional parameter space to a 10-dimensional subspace. Figure 4(b) is a comparison of iteration speed of SVGD and pSVGD. SVGD is much quicker, for pSVGD sacrifices the number of iterations for the efficiency in per steps.

Note that Table 1 and 2 are based on **Rcpp** codes. Table 1 shows the time spent in the same number of iterations. The dimension of parameter is fixed, that is, 100. One can say in most cases pSVGD is better than SVGD, even in low-dimensional cases. This isn't surprising since pSVGD reduced the dimension of kernel from 100×100 to 10×10. Table 2 shows the time spent in the same tolerance 10^{-3}, where the dimension of parameters differs. When the dimension is relatively low - lower than 100, SVGD is better than pSVGD since it needs fewer iterations. However, as dimension increases, pSVGD performs better, due to the lower time spent in calculating the kernel matrix. When dealing with high-dimensional data, such as more than 1000 dimensions, pSVGD is extraordinarily faster than SVGD.

TABLE 1. CPU time of SVGD and pSVGD in same iterations

	time(s) l_w l_x	1	10	100	1000
SVGD	100	0.08	0.79	7.59	75.22
pSVGD		0.20	0.61	4.15	37.02
SVGD	500	0.37	3.81	37.63	376.55
pSVGD		0.96	2.32	18.26	179.5
SVGD	1000	0.79	7.59	75.22	752.11
pSVGD		2.04	5.45	39.08	329.14

TABLE 2. CPU time of SVGD and pSVGD in same tolerance

d time(s)	10	50	100	200	400
SVGD	4.43	42.39	124.7	424.31	1587.92
pSVGD	8.27	70.61	152.6	362.37	834.98

5.4. **Conditional diffusion process.** This part is finished with the direction of **Projected Stein Variational Gradient Descent**[1] and more detailed procedure in **A Stein variational Newton method**[4].

For this section,we compare pSVGD and SVGD with the convergence we consider the discretized data points from a conditional diffusion process:

$$du_t = \frac{10u(1-u^2)}{1+u^2}dt + dx_t \tag{29}$$

with zero initial condition $u_0 = 0$. The forcing term $(x_t)_{t\geq 0}$ is a Brownian motion, whose prior is the Gauss Distribution with mean 0 and covariance $C(t1, t2) = \min(t1, t2)$, with t from 0 to 1. To solve the process, we use the Euler-Maruyama scheme and set stepsize $\delta_t = 0.01$, leading to 100 dimension data. Those 100 dimension data generate the u_t and observation data $y = (y_1, \cdots, y_{20})$. For each y, y_i is generated by $y_i = u_{t_i} + \xi_i$. u_t is the diffusion process pushed forward by the x_t Brownian Motion. ξ_i is the addictive noise from $N(0, \sigma^2)$ with $\sigma = 0.1$.

Our work is to infer x_{true} out of observation data y by using SVGD and pSVGD. Then we compare the convergence rate as well as the precision of the final result.Final results should be close to the x_{true}. The $x_{initial}$ and x_{true} are in the figure 5.

For the results:

1.Eigenvalues: The eigenvalues decreases very fast as been shown in the multivariate Normal. Here it is confirmed again.

2.Convergence Characteristic: Figure 6 shows the result of pSVGD and SVGD in their 1200 iteration. Figure(a) shows pSVGD pushes forward the data in a relatively coherent pace, while figure(b) indicates that SVGD delays in updating some parts of data. In conclusion, pSVGD is much closer to the real value at 1200th iteration than SVGD.

3.Convergence Iteration: As in figure 7, SVGD repeats 2600 times before convergence, and pSVGD repeats 1400 times as a great leap in efficiency. In SVGD's 2000th iteration, small part of data is still needed to be updated, and it finally gets to convergence at 2600th iteration.

4.Data Precision: In figure 7 ,when reaching to the convergence criterion, we have SVGD and pSVGD almost the same data updated, as an explanation–pSVGD is just accelerated from SVGD.

References

[1] Peng Chen and Omar Ghattas. *Projected Stein Variational Gradient Descent.* 2020. arXiv: 2002.03469 [cs.LG].

[2] Peng Chen, Keyi Wu, and Omar Ghattas. *Bayesian inference of heterogeneous epidemic models: Application to COVID-19 spread accounting for long-term care facilities.* 2020. arXiv: 2011.01058 [stat.ME].

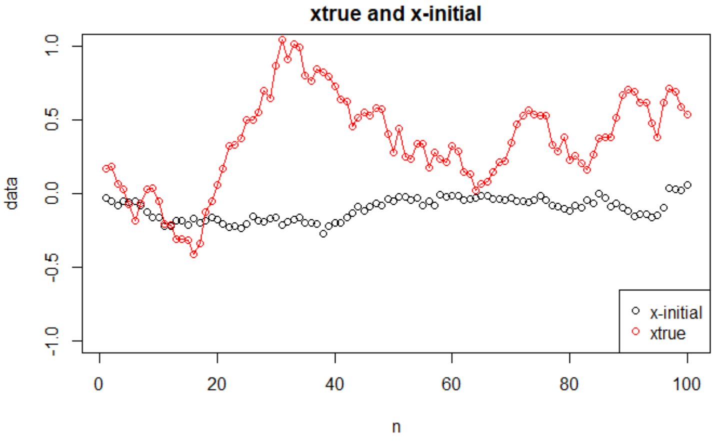
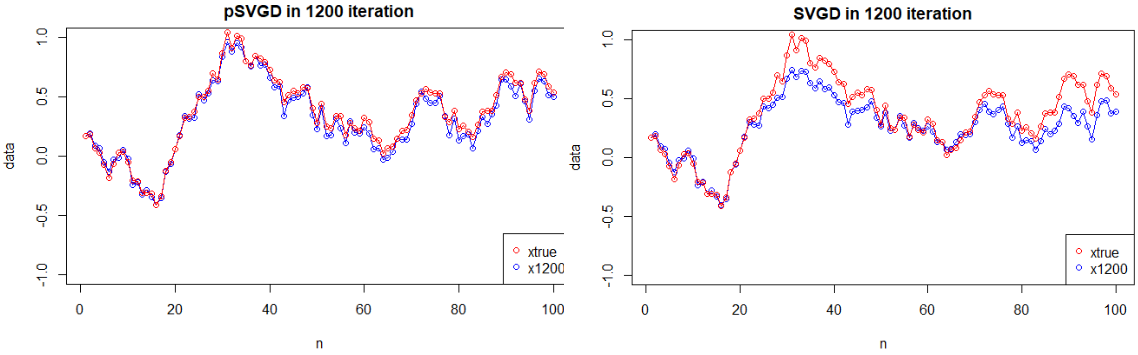


FIGURE 5. xtrue to aproach and the x-initial generated



(a) (b)

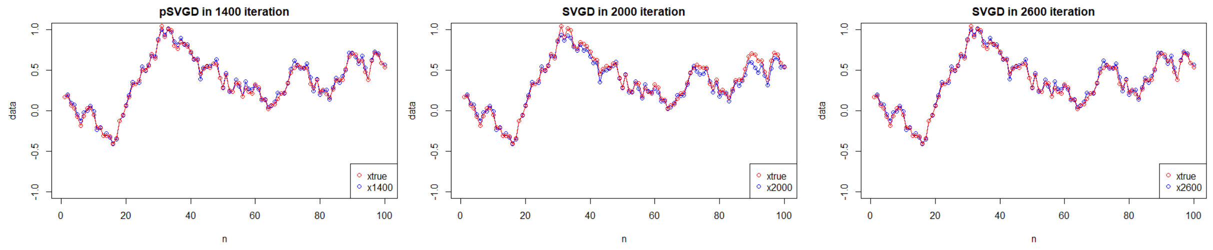
FIGURE 6. (a): pSVGD in 1200th iteration for x updating (b): SVGD in 1200th iteration

[3] Yukuang Chiu. “Theory of reproducing kernels”. In: *Transactions of the American Mathematical Society* 68.3 (1997), pp. 337–404.

[4] Gianluca Detommaso et al. *A Stein variational Newton method*. 2018. arXiv: 1806.03085 [stat.ML].

[5] Qiang Liu and Dilin Wang. *Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm*. 2019. arXiv: 1608.04471 [stat.ML].

[6] Yurii Nesterov. *Lectures on Convex Optimization*. Springer, 2018.



(a) (b) (c)

FIGURE 7. (a): pSVGD in 1400th iteration for x updating (b): SVGD in 2000th iteration (c)SVGD in 2600th iteration

Appendices

Proof of Theorem 3

Proof. From Appendix B in [5], we have

$$F[f] = \mathbf{E}_q[\log q(x) - \log p(x + f(x)) - \log\det(I + \nabla_x f(x))] \tag{30}$$

and

$$\nabla_f F[f] = -\mathbf{E}_q[\nabla_x \log p(x + f(x)) + trace((I + \nabla_x f(x))^{-1} \cdot \nabla_x k(x, \cdot)] \tag{31}$$

it's sufficient to prove that F is convex and ∇F is Lipschitz continuous. However, due to CONDITION 1 and the concaveness of $\log\det(X)$, the former one is obvious. We only need to prove the Lipschitz continuity of ∇F.

Suppose

$$\nabla F[f] - \nabla F[g] = -\Delta_1 - \Delta_2 \tag{32}$$

where

$$\Delta_1 = \mathbf{E}_q[\nabla_x \log p(x + f(x)) - \nabla_x \log p(x + g(x))] \tag{33}$$

$$\Delta_2 = \mathbf{E}_q[trace((I + \nabla_x f(x))^{-1} - (I + \nabla_x g(x))^{-1}) \cdot \nabla_x k(x, \cdot)] \tag{34}$$

For the first term in the above equation, we have

$$\|\Delta_1\|_{\mathcal{H}^d} = \|\mathbf{E}_q[\nabla_x \log p(x + f(x)) - \nabla_x \log p(x + g(x))]\|_{\mathcal{H}^d} \tag{35}$$

$$\leq \mathbf{E}_q|\nabla_x \log p(x + f(x)) - \nabla_x \log p(x + g(x))| \tag{36}$$

$$\leq \mathbf{E}_q L|f(x) - g(x)| \tag{37}$$

$$= \mathbf{E}_q L|\langle f - g, k(x, \cdot)\rangle| \tag{38}$$

$$\leq L\|f - g\|_{\mathcal{H}^d} \mathbf{E}_q \|k(x,)\|_{\mathcal{H}^d} \tag{39}$$

$$= LK\|f - g\|_{\mathcal{H}^d} \tag{40}$$

where the second inequality is due to gradient Lipschitz continuity of $\log p(x)$ and the last inequality is derived by C-S inequality.

For the second term, we have

$$\|\Delta_2\|_{\mathcal{H}^d} = \|\mathbf{E}_q[trace((I+\nabla_x g(x))^{-1}(I+\nabla_x f(x))^{-1}(\nabla_x(g-f))\cdot\nabla_x k(x,\cdot)]\|_{\mathcal{H}^d} \tag{41}$$

$$\leq \mathbf{E}_q|\frac{4aM}{h}|f(x)-g(x)|\cdot\nabla_x k(x,\cdot)| \tag{42}$$

$$\leq \frac{4aM}{h}\|f-g\|_{\mathcal{H}^d}\mathbf{E}_q\|k(x,)\|_{\mathcal{H}^d} \tag{43}$$

$$= \frac{4aMK}{h}\|f-g\|_{\mathcal{H}^d} \tag{44}$$

In combination of the former two results, we have

$$\|\nabla F[f]-\nabla F[g]\|_{\mathcal{H}^d} \leq K(L+\frac{4aM}{h})\|f-g\|_{\mathcal{H}^d} \tag{45}$$

Then the convergence theorem of gradient descent in [6] gives us the desirable result. □