Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm

Qiang Liu **Dilin Wang**
Department of Computer Science
Dartmouth College
Hanover, NH 03755
{qiang.liu, dilin.wang.gr}@dartmouth.edu

Abstract

We propose a general purpose variational inference algorithm that forms a natural counterpart of gradient descent for optimization. Our method iteratively transports a set of particles to match the target distribution, by applying a form of functional gradient descent that minimizes the KL divergence. Empirical studies are performed on various real world models and datasets, on which our method is competitive with existing state-of-the-art methods. The derivation of our method is based on a new theoretical result that connects the derivative of KL divergence under smooth transforms with Stein's identity and a recently proposed kernelized Stein discrepancy, which is of independent interest.

1 Introduction

Bayesian inference provides a powerful tool for modeling complex data and reasoning under uncertainty, but casts a long standing challenge on computing intractable posterior distributions. Markov chain Monte Carlo (MCMC) has been widely used to draw approximate posterior samples, but is often slow and has difficulty accessing the convergence. Variational inference instead frames the Bayesian inference problem into a deterministic optimization that approximates the target distribution with a simpler distribution by minimizing their KL divergence. This makes variational methods efficiently solvable by using off-the-shelf optimization techniques, and easily applicable to large datasets (i.e., "big data") using the stochastic gradient descent trick [e.g., 1]. In contrast, it is much more challenging to scale up MCMC to big data settings [see e.g., 2, 3].

Meanwhile, both the accuracy and computational cost of variational inference critically depend on the set of distributions in which the approximation is defined. Simple approximation sets, such as these used in the traditional mean field methods, are too restrictive to resemble the true posterior distributions, while more advanced choices cast more difficulties on the subsequent optimization tasks. For this reason, efficient variational methods often need to be derived on a model-by-model basis, causing is a major barrier for developing general purpose, user-friendly variational tools applicable for different kinds of models, and accessible to non-ML experts in application domains.

This case is in contrast with the maximum *a posteriori* (MAP) optimization tasks for finding the posterior mode (sometimes known as the *poor man's Bayesian estimator*, in contrast with the *full Bayesian inference* for approximating the full posterior distribution), for which variants of (stochastic) gradient descent serve as a simple, generic, yet extremely powerful toolbox. There has been a recent growth of interest in creating user-friendly variational inference tools [e.g., 4–7], but more efforts are still needed to develop more efficient general purpose algorithms.

In this work, we propose a new general purpose variational inference algorithm which can be treated as a natural counterpart of gradient descent for full Bayesian inference (see Algorithm 1). Our algorithm uses a set of particles for approximation, on which a form of (functional) gradient descent

is performed to minimize the KL divergence and drive the particles to fit the true posterior distribution. Our algorithm has a simple form, and can be applied whenever gradient descent can be applied. In fact, it reduces to gradient descent for MAP when using only a single particle, while automatically turns into a full Bayesian approach with more particles.

Underlying our algorithm is a new theoretical result that connects the derivative of KL divergence w.r.t. smooth variable transforms and a recently introduced kernelized Stein discrepancy [8–10], which allows us to derive a closed form solution for the optimal smooth perturbation direction that gives the steepest descent on the KL divergence within the unit ball of a reproducing kernel Hilbert space (RKHS). This new result is of independent interest, and can find wide application in machine learning and statistics beyond variational inference.

Outline This paper is organized as follows. Section 2 introduces backgrounds on kernelized Stein discrepancy (KSD). Our main results are presented in Section 3 in which we clarify the connection between KSD and KL divergence, and leverage it to develop our novel variational inference method. Section 4 discusses related works, and Section 5 presents numerical results. The paper is concluded in Section 6.

2 Background

Preliminary Let x be a continuous random variable or parameter of interest taking values in $\mathcal{X} \subset \mathbb{R}^d$, and $\{D_k\}$ is a set of i.i.d. observation. With prior $p_0(x)$, Bayesian inference of x involves reasoning with the posterior distribution $p(x) \coloneqq \bar{p}(x)/Z$ with $\bar{p}(x) \coloneqq p_0(x)\prod_{k=1}^N p(D_k|x)$, where $Z = \int \bar{p}(x)dx$ is the troublesome normalization constant. We have dropped the conditioning on data $\{D_k\}$ in $p(x)$ for convenience.

Let $k(x, x')\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel. The reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ of $k(x, x')$ is the closure of linear span $\{f\colon f(x) = \sum_{i=1}^m a_i k(x, x_i),\ \ a_i \in \mathbb{R},\ m \in \mathbb{N},\ x_i \in \mathcal{X}\}$, equipped with inner products $\langle f,\ g\rangle_{\mathcal{H}} = \sum_{ij} a_i b_j k(x_i, x_j)$ for $g(x) = \sum_i b_i k(x, x_i)$. Denote by $\mathcal{H}^d$ the space of vector functions $\boldsymbol{f} = [f_1, \ldots, f_d]$ with $f_i \in \mathcal{H}$, equipped with inner product $\langle \boldsymbol{f}, \boldsymbol{g}\rangle_{\mathcal{H}^d} = \sum_{i=1}^d \langle f_i, g_i\rangle_{\mathcal{H}}$. We assume all the vectors are column vectors.

Stein's Identity and Kernelized Stein Discrepancy Stein's identity plays a fundamental role in our framework. Let $p(x)$ be a continuously differentiable (also called smooth) density supported on $\mathcal{X} \subseteq \mathbb{R}^d$, and $\boldsymbol{\phi}(x) = [\phi_1(x), \cdots, \phi_d(x)]^\top$ a smooth vector function. Stein's identity states that for sufficiently regular $\boldsymbol{\phi}$, we have

$$\mathbb{E}_{x\sim p}[\mathcal{A}_p\boldsymbol{\phi}(x)] = 0, \qquad \text{where} \qquad \mathcal{A}_p\boldsymbol{\phi}(x) = \boldsymbol{\phi}(x)\nabla_x \log p(x)^\top + \nabla_x\boldsymbol{\phi}(x), \tag{1}$$

where $\mathcal{A}_p$ is called the Stein operator, which acts on function $\boldsymbol{\phi}$ and yields a zero mean function $\mathcal{A}_p\boldsymbol{\phi}(x)$ under $x \sim p$. This identity can be easily checked using integration by parts, assuming mild zero boundary conditions on $\boldsymbol{\phi}$, either $p(x)\boldsymbol{\phi}(x) = 0,\ \forall x \in \partial\mathcal{X}$ when $\mathcal{X}$ is compact, or $\lim_{||x||\to\infty} \boldsymbol{\phi}(x)p(x) = 0$ when $\mathcal{X} = \mathbb{R}^d$. We call that $\boldsymbol{\phi}$ is in the Stein class of p if Stein's identity (1) holds.

Now let $q(x)$ be a different smooth density also supported in $\mathcal{X}$, and consider the expectation of $\mathcal{A}_p\boldsymbol{\phi}(x)$ under $x \sim q$, then $\mathbb{E}_{x\sim q}[\mathcal{A}_p\boldsymbol{\phi}(x)]$ would no longer equal zero for general $\boldsymbol{\phi}$. Instead, the magnitude of $\mathbb{E}_{x\sim q}[\mathcal{A}_p\boldsymbol{\phi}(x)]$ relates to how different p and q are, and can be leveraged to define a discrepancy measure, known as *Stein discrepancy*, by considering the "maximum violation of Stein's identity" for $\boldsymbol{\phi}$ in some proper function set $\mathcal{F}$:

$$\mathbb{S}(q,\ p) = \max_{\boldsymbol{\phi}\in\mathcal{F}} \left\{[\mathbb{E}_{x\sim q}\mathrm{trace}(\mathcal{A}_p\boldsymbol{\phi}(x))]^2\right\},$$

Here the choice of this function set $\mathcal{F}$ is critical, and decides the discriminative power and computational tractability of Stein discrepancy. Traditionally, $\mathcal{F}$ is taken to be sets of functions with bounded Lipschitz norms, which unfortunately casts a challenging functional optimization problem that is computationally intractable or requires special considerations (see Gorham and Mackey [11] and reference therein).

Kernelized Stein discrepancy bypasses this difficulty by maximizing $\boldsymbol{\phi}$ in the unit ball of a reproducing kernel Hilbert space (RKHS) for which the optimization has a closed form solution. Following Liu

et al. [8], KSD is defined as

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

where we assume the kernel $k(x, x')$ of RKHS $\mathcal{H}$ is in the Stein class of p as a function of x for any fixed $x' \in \mathcal{X}$. The optimal solution of (2) has been shown [8–10] to be $\boldsymbol{\phi}(x) = \boldsymbol{\phi}^*_{q,p}(x)/||\boldsymbol{\phi}^*_{q,p}||_{\mathcal{H}^d}$, where

$$\boldsymbol{\phi}^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}[\mathcal{A}_p k(x, \cdot)], \qquad \text{for which we have} \qquad \mathbb{S}(q,\ p) = ||\boldsymbol{\phi}^*_{q,p}||^2_{\mathcal{H}^d}. \tag{3}$$

One can further show that $\mathbb{S}(q,\ p)$ equals zero (and equivalently $\boldsymbol{\phi}^*_{q,p}(x) \equiv 0$) if and only if $p = q$ once $k(x, x')$ is strictly positive definite in a proper sense [See 8, 10], which is satisfied by commonly used kernels such as the RBF kernel $k(x, x') = \exp(-\frac{1}{h}||x - x'||_2^2)$. Note that the RBF kernel is also in the Stein class of smooth densities supported in $\mathcal{X} = \mathbb{R}^d$ because of its decaying property.

Both Stein operator and KSD depend on p only through the score function $\nabla_x \log p(x)$, which can be calculated without knowing the normalization constant of p, because we have $\nabla_x \log p(x) = \nabla_x \log \bar{p}(x)$ when $p(x) = \bar{p}(x)/Z$. This property makes Stein's identity a powerful tool for handling unnormalized distributions that appear widely in machine learning and statistics.

3 Variational Inference Using Smooth Transforms

Variational inference approximates the target distribution $p(x)$ using a simpler distribution $q^*(x)$ found in a predefined set $\mathcal{Q} = \{q(x)\}$ of distributions by minimizing the KL divergence, that is,

$$q^* = \arg\min_{q\in\mathcal{Q}} \big\{ \mathrm{KL}(q \,||\, p) \equiv \mathbb{E}_q[\log q(x)] - \mathbb{E}_q[\log \bar{p}(x)] + \log Z \big\}, \tag{4}$$

where we do not need to calculate the constant $\log Z$ for solving the optimization. The choice of set $\mathcal{Q}$ is critical and defines different types of variational inference methods. The best set $\mathcal{Q}$ should strike a balance between i) *accuracy*, broad enough to closely approximate a large class of target distributions, ii) *tractability*, consisting of simple distributions that are easy for inference, and iii) *solvability* so that the subsequent KL minimization problem can be efficiently solved.

In this work, we focus on the sets $\mathcal{Q}$ consisting of distributions obtained by smooth transforms from a tractable reference distribution, that is, we take $\mathcal{Q}$ to be the set of distributions of random variables of form $z = \boldsymbol{T}(x)$ where $\boldsymbol{T} \colon \mathcal{X} \to \mathcal{X}$ is a smooth one-to-one transform, and x is drawn from a tractable reference distribution $q_0(x)$. By the change of variables formula, the density of z is

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

where $\boldsymbol{T}^{-1}$ denotes the inverse map of $\boldsymbol{T}$ and $\nabla_z \boldsymbol{T}^{-1}$ the Jacobian matrix of $\boldsymbol{T}^{-1}$. Such distributions are computationally tractable, in the sense that the expectation under $q_{[\boldsymbol{T}]}$ can be easily evaluated by averaging $\{z_i\}$ when $z_i = \boldsymbol{T}(x_i)$ and $x_i \sim q_0$. Such $\mathcal{Q}$ can also in principle closely approximate almost arbitrary distributions: it can be shown that there always exists a measurable transform $\boldsymbol{T}$ between any two distributions without atoms (i.e. no single point carries a positive mass); in addition, for Lipschitz continuous densities p and q, there always exist transforms between them that are least as smooth as both p and q. We refer the readers to Villani [12] for in-depth discussion on this topic.

In practice, however, we need to restrict the set of transforms $\boldsymbol{T}$ properly to make the corresponding variational optimization in (4) practically solvable. One approach is to consider $\boldsymbol{T}$ with certain parametric form and optimize the corresponding parameters [e.g., 13, 14]. However, this introduces a difficult problem on selecting the proper parametric family to balance the accuracy, tractability and solvability, especially considering that $\boldsymbol{T}$ has to be an one-to-one map and has to have an efficiently computable Jacobian matrix.

Instead, we propose a new algorithm that iteratively constructs incremental transforms that effectively perform steepest descent on $\boldsymbol{T}$ in RKHS. Our algorithm does not require to explicitly specify parametric forms, nor to calculate the Jacobian matrix, and has a particularly simple form that mimics the typical gradient descent algorithm, making it easily implementable even for non-experts in variational inference.

3.1 Stein Operator as the Derivative of KL Divergence

To explain how we minimize the KL divergence in (4), we consider an incremental transform formed by a small perturbation of the identity map: $\boldsymbol{T}(x) = x + \epsilon\boldsymbol{\phi}(x)$, where $\boldsymbol{\phi}(x)$ is a smooth function that characterizes the perturbation direction and the scalar ϵ represents the perturbation magnitude. When $|\epsilon|$ is sufficiently small, the Jacobian of $\boldsymbol{T}$ is full rank (close to the identity matrix), and hence $\boldsymbol{T}$ is guaranteed to be an one-to-one map by the inverse function theorem.

The following result, which forms the foundation of our method, draws an insightful connection between Stein operator and the derivative of KL divergence w.r.t. the perturbation magnitude ϵ.

Theorem 3.1. *Let $\boldsymbol{T}(x) = x + \epsilon\boldsymbol{\phi}(x)$ and $q_{[\boldsymbol{T}]}(z)$ the density of $z = \boldsymbol{T}(x)$ when $x \sim q(x)$, we have*

$$\nabla_\epsilon \mathrm{KL}(q_{[\boldsymbol{T}]} \,||\, p) \,\big|_{\epsilon=0} = -\mathbb{E}_{x\sim q}[\mathrm{trace}(\mathcal{A}_p\boldsymbol{\phi}(x))], \tag{5}$$

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

Relating this to the definition of KSD in (2), we can identify the $\boldsymbol{\phi}^*_{q,p}$ in (3) as the optimal perturbation direction that gives the steepest descent on the KL divergence in zero-centered balls of $\mathcal{H}^d$.

Lemma 3.2. *Assume the conditions in Theorem 3.1. Consider all the perturbation directions $\boldsymbol{\phi}$ in the ball $\mathcal{B} = \{\boldsymbol{\phi} \in \mathcal{H}^d \colon ||\boldsymbol{\phi}||^2_{\mathcal{H}^d} \le \mathbb{S}(q,\ p)\}$ of vector-valued RKHS $\mathcal{H}^d$, the direction of steepest descent that maximizes the negative gradient in* (5) *is the $\boldsymbol{\phi}^*_{q,p}$ in* (3)*, i.e.,*

$$\boldsymbol{\phi}^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}[k(x,\cdot)\nabla_x \log p(x) + \nabla_x k(x,\cdot)], \tag{6}$$

for which the negative gradient in (5) *equals KSD, that is, $\nabla_\epsilon \mathrm{KL}(q_{[\boldsymbol{T}]} \,||\, p) \,\big|_{\epsilon=0} = -\mathbb{S}(q,\ p)$.*

The result in Lemma (3.2) suggests an iterative procedure that transforms an initial reference distribution q_0 to the target distribution p: we start with applying transform $\boldsymbol{T}^*_0(x) = x + \epsilon_0 \cdot \boldsymbol{\phi}^*_{q_0,p}(x)$ on q_0 which decreases the KL divergence by an amount of $\epsilon_0 \cdot \mathbb{S}(q_0,\ p)$, where ϵ_0 is a small step size; this would give a new distribution $q_1(x) = q_{0[\boldsymbol{T}_0]}(x)$, on which a further transform $\boldsymbol{T}^*_1(x) = x + \epsilon_1 \cdot \boldsymbol{\phi}^*_{q_1,p}(x)$ can further decrease the KL divergence by $\epsilon_1 \cdot \mathbb{S}(q_1,\ p)$. Repeating this process one constructs a path of distributions $\{q_\ell\}_{\ell=1}^n$ between q_0 and p via

$$q_{\ell+1} = q_{\ell[\boldsymbol{T}^*_\ell]}, \qquad \text{where} \qquad \boldsymbol{T}^*_\ell(x) = x + \epsilon_\ell \cdot \boldsymbol{\phi}^*_{q_\ell,p}(x). \tag{7}$$

This would eventually converge to the target p with sufficiently small step-size $\{\epsilon_\ell\}$, under which $\boldsymbol{\phi}^*_{p,q_\infty}(x) \equiv 0$ and $\boldsymbol{T}^*_\infty$ reduces to the identity map. Recall that $q_\infty = p$ if and only if $\boldsymbol{\phi}^*_{p,q_\infty}(x) \equiv 0$.

Functional Gradient To gain further intuition on this process, we now reinterpret (6) as a functional gradient in RKHS. 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 $\epsilon \in \mathbb{R}$.

Theorem 3.3. *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 $x \sim q$,*

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

whose squared RKHS norm is $||\boldsymbol{\phi}^*_{q,p}||^2_{\mathcal{H}^d} = \mathbb{S}(q,p)$.

This suggests that $\boldsymbol{T}^*(x) = x + \epsilon \cdot \boldsymbol{\phi}^*_{q,p}(x)$ is equivalent to a step of functional gradient descent in RKHS. However, what is critical in the iterative procedure (7) is that we also iteratively apply the variable transform so that every time we would only need to evaluate the functional gradient descent at zero perturbation $\boldsymbol{f} = 0$ on the identity map $\boldsymbol{T}(x) = x$. This brings a critical advantage since the gradient at $\boldsymbol{f} \neq 0$ is more complex and would require to calculate the inverse Jacobian matrix $[\nabla_x \boldsymbol{T}(x)]^{-1}$ that casts computational or implementation hurdles.

3.2 Stein Variational Gradient Descent

To implement the iterative procedure (7) in practice, one would need to approximate the expectation for calculating $\boldsymbol{\phi}^*_{q,p}(x)$ in (6). To do this, we can first draw a set of particles $\{x_i^0\}_{i=1}^n$ from the initial

Algorithm 1 Bayesian Inference via Variational Gradient Descent

Input: A target distribution with density function $p(x)$ and a set of initial particles $\{x_i^0\}_{i=1}^n$.
Output: A set of particles $\{x_i\}_{i=1}^n$ that approximates the target distribution.
for iteration ℓ **do**

$$x_i^{\ell+1} \leftarrow x_i^\ell + \epsilon_\ell \hat{\boldsymbol{\phi}}^*(x_i^\ell) \quad \text{where} \quad \hat{\boldsymbol{\phi}}^*(x) = \frac{1}{n}\sum_{j=1}^{n}\left[k(x_j^\ell, x)\nabla_{x_j^\ell}\log p(x_j^\ell) + \nabla_{x_j^\ell} k(x_j^\ell, x)\right], \quad (8)$$

where ϵ_ℓ is the step size at the ℓ-th iteration.
end for

distribution q_0, and then iteratively update the particles with an empirical version of the transform in (7) in which the expectation under q_ℓ in $\boldsymbol{\phi}^*_{q_\ell,p}$ is approximated by the empirical mean of particles $\{x_i^\ell\}_{i=1}^n$ at the ℓ-th iteration. This procedure is summarized in Algorithm 1, which allows us to (deterministically) transport a set of points to match our target distribution $p(x)$, effectively providing a sampling method for $p(x)$. We can see that this procedure does not depend on the initial distribution q_0 at all, meaning that we can apply this procedure starting with a set of arbitrary points $\{x_i\}_{i=1}^n$, possibly generated by a complex (randomly or deterministic) black-box procedure.

We can expect that $\{x_i^\ell\}_{i=1}^n$ forms increasingly better approximation for q_ℓ as n increases. To see this, denote by Φ the nonlinear map that takes the measure of q_ℓ and outputs that of $q_{\ell+1}$ in (7), that is, $q_{\ell+1} = \Phi_\ell(q_\ell)$, where q_ℓ enters the map through both $q_{\ell[\boldsymbol{T}_\ell^*]}$ and $\boldsymbol{\phi}^*_{q_\ell,p}$. Then, the updates in Algorithm 1 can be seen as applying the same map Φ on the empirical measure $\hat{q}_\ell$ of particles $\{x_i^\ell\}$ to get the empirical measure $\hat{q}_{\ell+1}$ of particles $\{x_i^{\ell+1}\}$ at the next iteration, that is, $\hat{q}_{\ell+1} = \Phi_\ell(\hat{q}_\ell)$. Since $\hat{q}_0$ converges to q_0 as n increases, $\hat{q}_\ell$ should also converge to q_ℓ when the map Φ is "continuous" in a proper sense. Rigorous theoretical results on such convergence have been established in the mean field theory of interacting particle systems [e.g., 15], which in general guarantee that $\sum_{i=1}^n h(x_i^\ell)/n - \mathbb{E}_{q_\ell}[h(x)] = \mathcal{O}(1/\sqrt{n})$ for bounded testing functions h. In addition, the distribution of each particle $x_{i_0}^\ell$, for any fixed i_0, also tends to q_ℓ, and is independent with any other finite subset of particles as $n \to \infty$, a phenomenon called *propagation of chaos* [16]. We leave concrete theoretical analysis for future work.

Algorithm 1 mimics a gradient dynamics at the particle level, where the two terms in $\hat{\boldsymbol{\phi}}^*(x)$ in (8) play different roles: the first term drives the particles towards the high probability areas of $p(x)$ by following a *smoothed* gradient direction, which is the weighted sum of the gradients of all the points weighted by the kernel function. The second term acts as a *repulsive force* that prevents all the points to collapse together into local modes of $p(x)$; to see this, consider the RBF kernel $k(x, x') = \exp(-\frac{1}{h}||x - x'||^2)$, the second term reduces to $\sum_j \frac{2}{h}(x - x_j)k(x_j, x)$, which drives x away from its neighboring points x_j that have large $k(x_j, x)$. If we let bandwidth $h \to 0$, the repulsive term vanishes, and update (8) reduces to a set of independent chains of typical gradient ascent for maximizing $\log p(x)$ (i.e., MAP) and all the particles would collapse into the local modes.

Another interesting case is when we use only a single particle ($n = 1$), in which case Algorithm 1 reduces to a single chain of typical gradient ascent for MAP for any kernel that satisfies $\nabla_x k(x, x) = 0$ (for which RBF holds). This suggests that our algorithm can generalize well for supervised learning tasks even with a very small number n of particles, since gradient ascent for MAP ($n = 1$) has been shown to be very successful in practice. This property distinguishes our particle method with the typical Monte Carlo methods that requires to average over many points. The key difference here is that we use a deterministic repulsive force, other than Monte Carlo randomness, to get diverse points for distributional approximation.

Complexity and Efficient Implementation The major computation bottleneck in (8) lies on calculating the gradient $\nabla_x \log p(x)$ for all the points $\{x_i\}_{i=1}^n$; this is especially the case in big data settings when $p(x) \propto p_0(x)\prod_{k=1}^N p(D_k|x)$ with a very large N. We can conveniently address this problem by approximating $\nabla_x \log p(x)$ with subsampled mini-batches $\Omega \subset \{1, \ldots, N\}$ of the data

$$\log p(x) \approx \log p_0(x) + \frac{N}{|\Omega|}\sum_{k\in\Omega}\log p(D_k \mid x). \quad (9)$$

Additional speedup can be obtained by parallelizing the gradient evaluation of the n particles.

The update (8) also requires to compute the kernel matrix $\{k(x_i, x_j)\}$ which costs $\mathcal{O}(n^2)$; in practice, this cost can be relatively small compared with the cost of gradient evaluation, since it can be sufficient to use a relatively small n (e.g., several hundreds) in practice. If there is a need for very large n, one can approximate the summation $\sum_{i=1}^n$ in (8) by subsampling the particles, or using a random feature expansion of the kernel $k(x, x')$ [17].

4 Related Works

Our work is mostly related to Rezende and Mohamed [13], which also considers variational inference over the set of transformed random variables, but focuses on transforms of parametric form $T(x) = f_\ell(\cdots(f_1(f_0(x))))$ where $f_i(\cdot)$ is a predefined simple parametric transform and ℓ a predefined length; this essentially creates a feedforward neural network with ℓ layers, whose invertibility requires further conditions on the parameters and need to be established case by case. The similar idea is also discussed in Marzouk et al. [14], which also considers transforms parameterized in special ways to ensure the invertible and the computational tractability of the Jacobian matrix. Recently, Tran et al. [18] constructed a variational family that achieves universal approximation based on Gaussian process (equivalent to a single-layer, infinitely-wide neural network), which does not have a Jacobian matrix but needs to calculate the inverse of the kernel matrix of the Gaussian process. Our algorithm has a simpler form, and does not require to calculate any matrix determinant or inversion. Several other works also leverage variable transforms in variational inference, but with more limited forms; examples include affine transforms [19, 20], and recently the copula models that correspond to element-wise transforms over the individual variables [21, 22].

Our algorithm maintains and updates a set of particles, and is of similar style with the Gaussian mixture variation inference methods whose mean parameters can be treated as a set of particles. [23–26, 5]. Optimizing such mixture KL objectives often requires certain approximation, and this was done most recently in Gershman et al. [5] by approximating the entropy using Jensen's inequality and the expectation term using Taylor approximation. There is also a large set of particle-based Monte Carlo methods, including variants of sequential Monte Carlo [e.g., 27, 28], as well as a recent particle mirror descent for optimizing the variational objective function [7]; compared with these methods, our method does not have the weight degeneration problem, and is much more "particle-efficient" in that we reduce to MAP with only one single particle.

5 Experiments

We test our algorithm on both toy and real world examples, on which we find our method tends to outperform a variety of baseline methods. Our code is available at `https://github.com/DartML/Stein-Variational-Gradient-Descent`.

For all our experiments, we use RBF kernel $k(x, x') = \exp(-\frac{1}{h}||x - x'||_2^2)$, and take the bandwidth to be $h = \text{med}^2/\log n$, where med is the median of the pairwise distance between the current points $\{x_i\}_{i=1}^n$; this is based on the intuition that we would have $\sum_j k(x_i, x_j) \approx n \exp(-\frac{1}{h}\text{med}^2) = 1$, so that for each x_i the contribution from its own gradient and the influence from the other points balance with each other. Note that in this way, the bandwidth h actually changes adaptively across the iterations. We use AdaGrad for step size and initialize the particles using the prior distribution unless otherwise specified.

Toy Example on 1D Gaussian Mixture We set our target distribution to be $p(x) = 1/3\mathcal{N}(x;\ -2, 1) + 2/3\mathcal{N}(x;\ 2, 1)$, and initialize the particles using $q_0(x) = \mathcal{N}(x; -10, 1)$. This creates a challenging situation since the probability mass of $p(x)$ and $q_0(x)$ are far away each other (with almost zero overlap). Figure 1 shows how the distribution of the particles $(n = 1)$ of our method evolve at different iterations. We see that despite the small overlap between $q_0(x)$ and $p(x)$, our method can push the particles towards the target distribution, and even recover the mode that is further away from the initial point. We found that other particle based algorithms, such as Dai et al. [7], tend to experience weight degeneracy on this toy example due to the ill choice of $q_0(x)$.

Figure 2 compares our method with Monte Carlo sampling when using the obtained particles to estimate expectation $\mathbb{E}_p(h(x))$ with different test functions $h(\cdot)$. We see that the MSE of our method tends to perform similarly or better than the exact Monte Carlo sampling. This may be because our

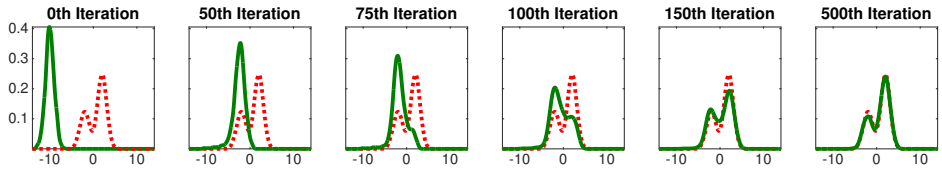
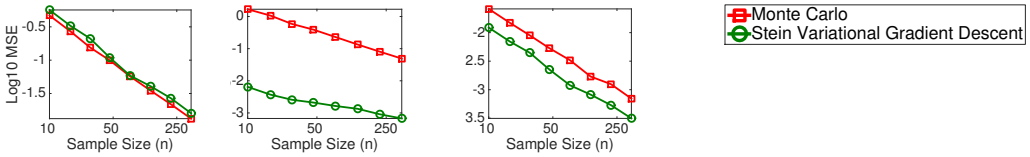


Figure 1: Toy example with 1D Gaussian mixture. The red dashed lines are the target density function and the solid green lines are the densities of the particles at different iterations of our algorithm (estimated using kernel density estimator) . Note that the initial distribution is set to have almost zero overlap with the target distribution, and our method demonstrates the ability of escaping the local mode on the left to recover the mode on the left that is further away. We use $n = 100$ particles.



(a) Estimating $\mathbb{E}(x)$ *(b) Estimating* $\mathbb{E}(x^2)$ *(c) Estimating* $\mathbb{E}(\cos(\omega x + b))$

Figure 2: We use the same setting as Figure 1, except varying the number n of particles. (a)-(c) show the mean square errors when using the obtained particles to estimate expectation $\mathbb{E}_p(h(x))$ for $h(x) = x$, x^2, and $\cos(\omega x + b)$; for $\cos(\omega x + b)$, we random draw $\omega \sim \mathcal{N}(0, 1)$ and $b \sim \text{Uniform}([0, 2\pi])$ and report the average MSE over 20 random draws of ω and b.

particles are more spread out than i.i.d. samples due to the repulsive force, and hence give higher estimation accuracy. It remains an open question to formally establish the error rate of our method.

Bayesian Logistic Regression We consider Bayesian logistic regression for binary classification using the same setting as Gershman et al. [5], which assigns the regression weights w with a Gaussian prior $p_0(w|\alpha) = \mathcal{N}(w, \alpha^{-1})$ and $p_0(\alpha) = Gamma(\alpha, 1, 0.01)$. The inference is applied on posterior $p(x|D)$ with $x = [w, \log \alpha]$. We compared our algorithm with the no-U-turn sampler (NUTS)[1] [29] and non-parametric variational inference (NPV)[2] [5] on the 8 datasets ($N > 500$) used in Gershman et al. [5], and find they tend to give very similar results on these (relatively simple) datasets; see Appendix for more details.

We further test the binary Covertype dataset[3] with 581,012 data points and 54 features. This dataset is too large, and a stochastic gradient descent is needed for speed. Because NUTS and NPV do not have mini-batch option in their code, we instead compare with the stochastic gradient Langevin dynamics (SGLD) by Welling and Teh [2], the particle mirror descent (PMD) by Dai et al. [7], and the doubly stochastic variational inference (DSVI) by Titsias and Lázaro-Gredilla [19].[4] We also compare with a parallel version of SGLD that runs n parallel chains and take the last point of each chain as the result. This parallel SGLD is similar with our method and we use the same step-size of $\epsilon_\ell = a/(t + 1)^{.55}$ for both as suggested by Welling and Teh [2] for fair comparison; [5] we select a using a validation set within the training set. For PMD, we use a step size of $\frac{a}{N}/(100 + \sqrt{t})$, and RBF kernel $k(x, x') = \exp(-||x - x'||^2/h)$ with bandwidth $h = 0.002 \times \text{med}^2$ which is based on the guidance of Dai et al. [7] which we find works most efficiently for PMD. Figure 3(a)-(b) shows the results when we initialize our method and both versions of SGLD using the prior $p_0(\alpha)p_0(w|\alpha)$; we find that PMD tends to be unstable with this initialization because it generates weights w with large magnitudes, so we divided the initialized weights by 10 for PMD; as shown in Figure 3(a), this gives some advantage to PMD in the initial stage. We find our method generally performs the best, followed with the parallel SGLD, which is much better than its sequential counterpart; this comparison is of course in favor of parallel SGLD, since each iteration of it requires $n = 100$ times of

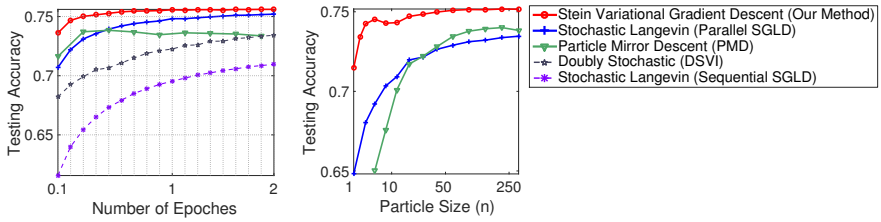
[1] code: http://www.cs.princeton.edu/ mdhoffma/

[2] code: http://gershmanlab.webfactional.com/pubs/npv.v1.zip

[3] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

[4] code: `http://www.aueb.gr/users/mtitsias/code/dsvi_matlabv1.zip`.

[5] We scale the gradient of SGLD by a factor of $1/n$ to make it match with the scale of our gradient in (8).



(a) Particle size $n = 100$ (b) Results at 3000 iteration (≈ 0.32 epoches)

Figure 3: Results on Bayesian logistic regression on Covertype dataset w.r.t. epochs and the particle size n. We use $n = 100$ particles for our method, parallel SGLD and PMD, and average the last 100 points for the sequential SGLD. The "particle-based" methods (solid lines) in principle require 100 times of likelihood evaluations compare with DVSI and sequential SGLD (dash lines) per iteration, but are implemented efficiently using Matlab matrix operation (e.g., each iteration of parallel SGLD is about 3 times slower than sequential SGLD). We partition the data into 80% for training and 20% for testing and average on 50 random trials. A mini-batch size of 50 is used for all the algorithms.

likelihood evaluations compared with sequential SGLD. However, by leveraging the matrix operation in MATLAB, we find that each iteration of parallel SGLD is only 3 times more expensive than sequential SGLD.

Bayesian Neural Network We compare our algorithm with the probabilistic back-propagation (PBP) algorithm by Hernández-Lobato and Adams [30] on Bayesian neural networks. Our experiment settings are almost identity, except that we use a $\mathrm{Gamma}(1, 0.1)$ prior for the inverse covariances and do not use the trick of scaling the input of the output layer. We use neural networks with one hidden layers, and take 50 hidden units for most datasets, except that we take 100 units for Protein and Year which are relatively large; all the datasets are randomly partitioned into 90% for training and 10% for testing, and the results are averaged over 20 random trials, except for Protein and Year on which 5 and 1 trials are repeated, respectively. We use $\mathrm{RELU}(x) = \max(0, x)$ as the active function, whose weak derivative is $\mathbb{I}[x > 0]$ (Stein's identity also holds for weak derivatives; see Stein et al. [31]). PBP is repeated using the default setting of the authors' code[6]. For our algorithm, we only use 20 particles, and use AdaGrad with momentum as what is standard in deep learning. The mini-batch size is 100 except for Year on which we use 1000.

We find our algorithm consistently improves over PBP both in terms of the accuracy and speed (except on Yacht); this is encouraging since PBP were specifically designed for Bayesian neural network. We also find that our results are comparable with the more recent results reported on the same datasets [e.g., 32–34] which leverage some advanced techniques that we can also benefit from.

	Avg. Test RMSE		Avg. Test LL		Avg. Time (Secs)	
Dataset	**PBP**	**Our Method**	**PBP**	**Our Method**	**PBP**	**Ours**
Boston	2.977 ± 0.093	$\mathbf{2.957 \pm 0.099}$	-2.579 ± 0.052	$\mathbf{-2.504 \pm 0.029}$	18	**16**
Concrete	5.506 ± 0.103	$\mathbf{5.324 \pm 0.104}$	-3.137 ± 0.021	$\mathbf{-3.082 \pm 0.018}$	33	**24**
Energy	1.734 ± 0.051	$\mathbf{1.374 \pm 0.045}$	-1.981 ± 0.028	$\mathbf{-1.767 \pm 0.024}$	25	**21**
Kin8nm	0.098 ± 0.001	$\mathbf{0.090 \pm 0.001}$	0.901 ± 0.010	$\mathbf{0.984 \pm 0.008}$	118	**41**
Naval	0.006 ± 0.000	$\mathbf{0.004 \pm 0.000}$	3.735 ± 0.004	$\mathbf{4.089 \pm 0.012}$	173	**49**
Combined	4.052 ± 0.031	$\mathbf{4.033 \pm 0.033}$	-2.819 ± 0.008	$\mathbf{-2.815 \pm 0.008}$	136	**51**
Protein	4.623 ± 0.009	$\mathbf{4.606 \pm 0.013}$	-2.950 ± 0.002	$\mathbf{-2.947 \pm 0.003}$	682	**68**
Wine	0.614 ± 0.008	$\mathbf{0.609 \pm 0.010}$	-0.931 ± 0.014	$\mathbf{-0.925 \pm 0.014}$	26	**22**
Yacht	$\mathbf{0.778 \pm 0.042}$	0.864 ± 0.052	$\mathbf{-1.211 \pm 0.044}$	-1.225 ± 0.042	25	25
Year	$8.733 \pm \mathrm{NA}$	$\mathbf{8.684 \pm NA}$	$-3.586 \pm \mathrm{NA}$	$\mathbf{-3.580 \pm NA}$	7777	**684**

6 Conclusion

We propose a simple general purpose variational inference algorithm for fast and scalable Bayesian inference. Future directions include more theoretical understanding on our method, more practical applications in deep learning models, and other potential applications of our basic Theorem in Section 3.1.

[6]https://github.com/HIPS/Probabilistic-Backpropagation

References

[1] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *JMLR*, 2013.

[2] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 2011.

[3] D. Maclaurin and R. P. Adams. Firefly Monte Carlo: Exact MCMC with subsets of data. In *UAI*, 2014.

[4] R. Ranganath, S. Gerrish, and D. M. Blei. Black box variational inference. In *AISTATS*, 2014.

[5] S. Gershman, M. Hoffman, and D. Blei. Nonparametric variational inference. In *ICML*, 2012.

[6] A. Kucukelbir, R. Ranganath, A. Gelman, and D. Blei. Automatic variational inference in STAN. In *NIPS*, 2015.

[7] B. Dai, N. He, H. Dai, and L. Song. Provable Bayesian inference via particle mirror descent. In *AISTATS*, 2016.

[8] Q. Liu, J. D. Lee, and M. I. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. *ICML*, 2016.

[9] C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society, Series B*, 2016.

[10] K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness-of-fit. *ICML*, 2016.

[11] J. Gorham and L. Mackey. Measuring sample quality with Stein's method. In *NIPS*, pages 226–234, 2015.

[12] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[13] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.

[14] Y. Marzouk, T. Moselhy, M. Parno, and A. Spantini. An introduction to sampling via measure transport. *arXiv preprint arXiv:1602.05023*, 2016.

[15] P. Del Moral. *Mean field simulation for Monte Carlo integration*. CRC Press, 2013.

[16] M. Kac. *Probability and related topics in physical sciences*, volume 1. American Mathematical Soc., 1959.

[17] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[18] D. Tran, R. Ranganath, and D. M. Blei. Variational Gaussian process. In *ICLR*, 2016.

[19] M. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *ICML*, pages 1971–1979, 2014.

[20] E. Challis and D. Barber. Affine independent variational inference. In *NIPS*, 2012.

[21] S. Han, X. Liao, D. B. Dunson, and L. Carin. Variational Gaussian copula inference. In *AISTATS*, 2016.

[22] D. Tran, D. M. Blei, and E. M. Airoldi. Copula variational inference. In *NIPS*, 2015.

[23] C. M. B. N. Lawrence and T. J. M. I. Jordan. Approximating posterior distributions in belief networks using mixtures. In *NIPS*, 1998.

[24] T. S. Jaakkola and M. I. Jordon. Improving the mean field approximation via the use of mixture distributions. In *Learning in graphical models*, pages 163–173. MIT Press, 1999.

[25] N. D. Lawrence. *Variational inference in probabilistic models*. PhD thesis, University of Cambridge, 2001.

[26] T. D. Kulkarni, A. Saeedi, and S. Gershman. Variational particle approximations. *arXiv preprint arXiv:1402.5715*, 2014.

[27] C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[28] A. Smith, A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[29] M. D. Hoffman and A. Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *The Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[30] J. M. Hernández-Lobato and R. P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.

[31] C. Stein, P. Diaconis, S. Holmes, G. Reinert, et al. Use of exchangeable pairs in the analysis of simulations. In *Stein's Method*, pages 1–25. Institute of Mathematical Statistics, 2004.

[32] Y. Li, J. M. Hernández-Lobato, and R. E. Turner. Stochastic expectation propagation. In *NIPS*, 2015.

[33] Y. Li and R. E. Turner. Variational inference with Renyi divergence. *arXiv preprint arXiv:1602.02311*, 2016.

[34] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

[35] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[36] S. Lyu. Interpretation and generalization of score matching. In *UAI*, pages 359–366, 2009.

A Proof of Theorem 3.1

Lemma A.1. *Let q and p be two smooth densities, and $\boldsymbol{T} = \boldsymbol{T}_\epsilon(x)$ an one-to-one transform on $\mathcal{X}$ indexed by parameter ϵ, and $\boldsymbol{T}$ is differentiable w.r.t. both x and ϵ. Define $q_{[\boldsymbol{T}]}$ to be the density of $z = \boldsymbol{T}_\epsilon(x)$ when $x \sim q$, and $\boldsymbol{s}_p = \nabla_x \log p(x)$, we have*

$$\nabla_\epsilon \mathrm{KL}(q_{[\boldsymbol{T}]} \,||\, p) = \mathbb{E}_q\big[\boldsymbol{s}_p(\boldsymbol{T}(x))^\top \nabla_\epsilon \boldsymbol{T}(x) + \mathrm{trace}((\nabla_x \boldsymbol{T}(x))^{-1} \cdot \nabla_\epsilon \nabla_x \boldsymbol{T}(x))\big].$$

Proof. Denote by $p_{[\boldsymbol{T}^{-1}]}(z)$ the density of $z = \boldsymbol{T}^{-1}(x)$ when $x \sim p(x)$, then

$$q_{[\boldsymbol{T}^{-1}]}(x) = q(\boldsymbol{T}(x)) \cdot |\det(\nabla_x \boldsymbol{T}(x))|.$$

By the change of variable, we have

$$\mathrm{KL}(q_{[\boldsymbol{T}]} \,||\, p) = \mathrm{KL}(q \,||\, p_{[\boldsymbol{T}^{-1}]}),$$

and hence

$$\nabla_\epsilon \mathrm{KL}(q_{[\boldsymbol{T}]} \,||\, p) = -\mathbb{E}_{x\sim q}[\nabla_\epsilon \log p_{[\boldsymbol{T}^{-1}]}(x)].$$

We just need to calculate $\log p_{[\boldsymbol{T}^{-1}]}(x)$; define $\boldsymbol{s}_p(x) = \nabla_x \log p(x)$, we get

$$\nabla_\epsilon \log p_{[\boldsymbol{T}^{-1}]}(x) = \boldsymbol{s}_p(\boldsymbol{T}(x))^\top \nabla_\epsilon \boldsymbol{T}(x) + \mathrm{trace}((\nabla_x \boldsymbol{T}(x))^{-1} \cdot \nabla_\epsilon \nabla_x \boldsymbol{T}(x)).$$

□

Proof of Theorem 3.1. When $\boldsymbol{T}(x) = x + \epsilon\boldsymbol{\phi}(x)$ and $\epsilon = 0$, we have

$$\boldsymbol{T}(x) = x, \qquad \nabla_\epsilon \boldsymbol{T}(x) = \boldsymbol{\phi}(x), \qquad \nabla_x \boldsymbol{T}(x) = I, \qquad \nabla_\epsilon \nabla_x \boldsymbol{T}(x) = \nabla_x \boldsymbol{\phi}(x),$$

where I is the identity matrix. Using Lemma A.1 gives the result. □

B Proof of Theorem 3.3

Let $\mathcal{H}^d = \mathcal{H} \times \cdots \times \mathcal{H}$ be a vector-valued RKHS, and $F[f]$ be a functional on f. The gradient $\nabla_f F[f]$ of $F[\cdot]$ is a function in $\mathcal{H}^d$ that satisfies

$$F[f + \epsilon g] = F[f] + \epsilon \,\langle \nabla_f F[f],\, g\rangle_{\mathcal{H}^d} + O(\epsilon^2).$$

Proof. Define $F[f] = \mathrm{KL}(q_{[x+f(x)]} \,||\, p) = \mathrm{KL}(q \,||\, p_{[(x+f(x))^{-1}]})$, we have

$$\begin{aligned} F[f + \epsilon g] &= \mathrm{KL}(q \,||\, p_{[(x+f(x)+\epsilon g(x))^{-1}]}) \\ &= \mathbb{E}_q[\log q(x) - \log p(x + f(x) + \epsilon g(x)) - \log\det(I + \nabla_x f(x) + \epsilon \nabla_x g(x))], \end{aligned}$$

and hence we have

$$F(f + \epsilon g) - F[f] = -\Delta_1 - \Delta_2,$$

where

$$\begin{aligned} \Delta_1 &= \mathbb{E}_q[\log p(x + f(x) + \epsilon g(x))] - \mathbb{E}_q[\log p(x + f(x))], \\ \Delta_2 &= \mathbb{E}_q[\log\det(I + \nabla_x f(x) + \epsilon \nabla_x g(x))] - \mathbb{E}_q[\log\det(I + \nabla_x f(x))]. \end{aligned}$$

For the terms in the above equation, we have

$$\begin{aligned} \Delta_1 &= \mathbb{E}_q[\log p(x + f(x) + \epsilon g(x))] - \mathbb{E}_q[\log p(x + f(x))] \\ &= \epsilon\, \mathbb{E}_q[\nabla_x \log p(x + f(x)) \cdot g(x)] + O(\epsilon^2) \\ &= \epsilon\, \mathbb{E}_q[\nabla_x \log p(x + f(x)) \cdot \langle k(x, \cdot),\, g\rangle_{\mathcal{H}^d}] + O(\epsilon^2) \\ &= \epsilon\, \langle \mathbb{E}_q[\nabla_x \log p(x + f(x)) \cdot k(x, \cdot)],\, g\rangle_{\mathcal{H}^d} + O(\epsilon^2), \end{aligned}$$

and

$$\begin{aligned} \Delta_2 &= \mathbb{E}_q[\log\det(I + \nabla_x f(x) + \epsilon \nabla_x g(x))] - \mathbb{E}_q[\log\det(I + \nabla_x f(x))] \\ &= \epsilon\, \mathbb{E}_q[\mathrm{trace}((I + \nabla_x f(x))^{-1} \cdot \nabla_x g(x))] \;+\; O(\epsilon^2) \\ &= \epsilon\, \mathbb{E}_q[\mathrm{trace}((I + \nabla_x f(x))^{-1} \cdot \langle \nabla_x k(x, \cdot),\, g\rangle_{\mathcal{H}^d}] \;+\; O(\epsilon^2) \\ &= \epsilon\, \langle \mathbb{E}_q[\mathrm{trace}((I + \nabla_x f(x))^{-1} \cdot \nabla_x k(x, \cdot)],\, g\rangle_{\mathcal{H}^d} \;+\; O(\epsilon^2) \end{aligned}$$

and hence

$$F(f + \epsilon g) - F[f] = \epsilon \, \langle \nabla_f F[f], \, g \rangle_{\mathcal{H}^d} \, + \, O(\epsilon^2),$$

where

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

Taking $f = 0$ then gives the desirable result. □

C Connection with de Bruijn's identity and Fisher Divergence

If we take $\phi_{q,p}(x) = \nabla_x \log p(x) - \nabla_x \log q(x)$ in (5), we can show that (5) reduces to

$$\nabla_\epsilon \mathrm{KL}(q_{[\boldsymbol{T}]} \, || \, p)\big|_{\epsilon=0} = -\mathcal{F}(q, \, p),$$

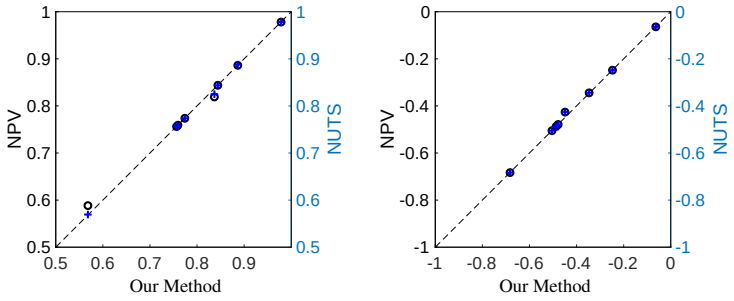
where $\mathcal{F}(q, \, p)$ is the Fisher divergence between p and q, defined as

$$\mathcal{F}(q, \, p) = \mathbb{E}_q[||\nabla_x \log p - \nabla_x \log q||_2^2].$$

Note that this can be treated as a deterministic version of *de Bruijn's identity* [35, 36], which draws similar connection between KL and Fisher divergence, but uses randomized linear transform $\boldsymbol{T}(x) = x + \sqrt{\epsilon} \cdot \xi$, where ξ is a standard Gaussian noise.

D Additional Experiments

We collect additional experimental results that can not fitted into the main paper.



(a) Testing Accuracy (b) Testing Log-Likelihood

Figure 4: Bayesian logistic regression on the 8 datasets studied in Gershman et al. [5]. We find our method performs similarly as NPV and NUTS on all the 8 datasets.

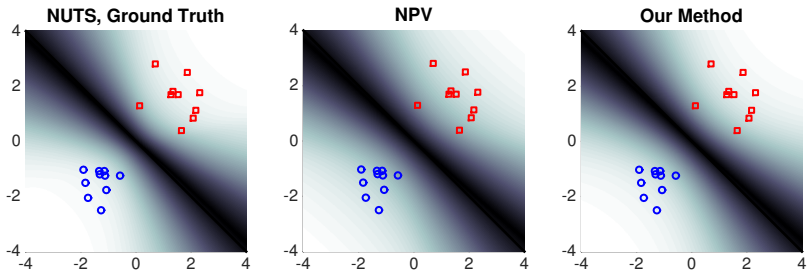


Figure 5: Bayesian logistic regression. The posterior prediction uncertainty as inferred by different approaches on a toy data.

D.1 Bayesian Logistic Regression on Small Datasets

We consider the Bayesian logistic regression model for binary classification, on which the regression weights w is assigned with a Gaussian prior $p_0(w) = \mathcal{N}(w, \alpha^{-1})$ and $p_0(\alpha) = \Gamma(\alpha, a, b)$, and apply

inference on posterior $p(x \mid D)$, where $x = [w, \log \alpha]$. The hyper-parameter is taken to be $a = 1$ and $b = 0.01$. This setting is the same as that in Gershman et al. [5]. We compared our algorithm with the no-U-turn sampler (NUTS)[7] [29] and non-parametric variational inference (NPV)[8] on the 8 datasets ($N > 500$) as used in Gershman et al. [5], in which we use 100 particles, NPV uses 100 mixture components, and NUTS uses 1000 draws with 1000 burnin period. We find that all these three algorithms almost always performs the same across the 8 datasets (See Figure in Appendix), and this is consistent with Figure 2 of Gershman et al. [5].

We further experimented on a toy dataset with only two features and visualize the prediction probability of the three algorithms in Figure D. We again find that all the three algorithms tend to perform similarly. Note, however, that NPV is relatively inconvenient to use since it requires the Hessian matrix, and NUTS tends to be very small when applied on massive datasets.

[7] code: http://www.cs.princeton.edu/ mdhoffma/

[8] code: http://gershmanlab.webfactional.com/pubs/npv.v1.zip