3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–16

Neural Variational Gradient Descent

Anonymous Authors
Anonymous Institution

Abstract

Particle-based approximate Bayesian inference approaches such as Stein Variational Gradient Descent (SVGD) combine the flexibility and convergence guarantees of sampling methods with the computational benefits of variational inference. In practice, SVGD relies on the choice of an appropriate kernel function, which impacts its ability to model the target distribution—a challenging problem with only heuristic solutions. We propose Neural Variational Gradient Descent (NVGD), which is based on parametrizing the witness function of the Stein discrepancy by a deep neural network whose parameters are learned in parallel to the inference, mitigating the necessity to make any kernel choices whatsoever. We empirically validate our method on synthetic and real-world inference problems.

1. Introduction

This work is concerned with the problem of generating samples from an unnormalized Bayesian posterior. In particular, we are interested in the case where the target posterior is estimated from a large dataset, and only stochastic (minibatch) approximations are available. In this setting, standard MCMC algorithms such as Hamiltonian Monte Carlo (Neal, 2012) or MALA (Roberts and Rosenthal, 1998) face difficulties (Betancourt, 2015; Roberts and Rosenthal, 1998). As a consequence, practitioners tend to prefer simpler methods such as variational inference (VI) and stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011). A more recent method developed by Liu and Wang (2016), Stein variational gradient descent (SVGD), has been gaining popularity.

These methods have in common that they minimize the KL divergence $\mathrm{KL}(q \parallel p)$ between an approximating distribution q and the target p: VI does so explicitly by optimizing a parameterized density q_θ, thus necessitating the choice of a (more or less) limited variational family of densities, which can introduce bias. SGLD and SVGD do so non-parametrically by producing samples whose distribution follows a gradient-descent-like trajectory towards the target (Jordan et al., 1998; Liu, 2017). In particular, SGLD approximates a gradient flow in the Wasserstein space of probability measures. This gradient flow can also be approximated deterministically by evolving a set of particles using SVGD (Liu and Wang, 2016), which allows to take the geometry of the target space into account. However, this geometry has to be encoded in a kernel function within the Stein class, which is often challenging to choose in practice.

In this work, we aim to overcome those limitations by learning an update function f_θ parameterized by a neural network that is trained in parallel to the inference. The samples are then updated via $\boldsymbol{x} \leftarrow \boldsymbol{x} + \varepsilon f_\theta(\boldsymbol{x})$. This approach has the following benefits:

- Unlike VI, our method does not constrain inference to a fixed family of distributions.
- Unlike SGLD, our method is deterministic and adapts to the geometry of the target.

© A. Authors.

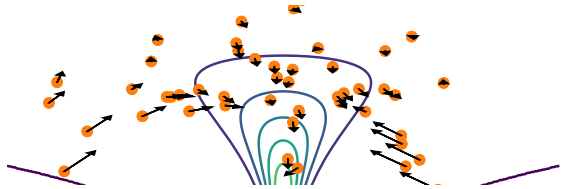


Figure 1: **Neural Variational Gradient Descent:** Initialize a set of samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ (orange). At each iteration, update samples via $\boldsymbol{x}_i = \boldsymbol{x}_i + \varepsilon f_\theta(\boldsymbol{x}_i)$ (black arrows), where the neural network f_θ is trained to minimize the KL divergence from the target posterior to the samples.

- Unlike SVGD, it is not dependent on a choice of kernel or kernel parameters and thus can *automatically* adapt to the geometry of the target posterior.

Empirically, we observe that our method outperforms or matches SVGD for a large array of choices of kernel parameters, while itself not relying on a kernel. In addition, we observe in synthetic experiments that our method exhibits less asymptotic bias than Langevin dynamics or SVGD.

2. Inference via KL-minimizing flows

Consider the problem of sampling from a posterior distribution p on $\mathbb{R}^d$. We recast this as an optimization problem, where the goal is to generate samples distributed according to the minimizer q^* of the KL divergence

$$q^* = \operatorname*{argmin}_{q \in \mathcal{P}'} \mathrm{KL}(q \parallel p),$$

where $\mathcal{P}'$ is some appropriate space of candidate measures. Concretely, we want to move a sample $\boldsymbol{x}_0 \sim q_0$ along a trajectory $(\boldsymbol{x}_k)_{k \in \mathbb{N}}$ such that the k-th iterate $\boldsymbol{x}_k \sim q_k$ converges to the posterior in the sense that $\lim_{k \to \infty} \mathrm{KL}(q_k \parallel p) = 0$. This is possible using the following deterministic, gradient-descent-like process: sample $\boldsymbol{x}_0 \sim q_0$, then iteratively apply

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \varepsilon \big(\nabla \log p(\boldsymbol{x}_k) - \nabla \log q_k(\boldsymbol{x}_k) \big). \tag{1}$$

The term $\nabla \log p - \nabla \log q_k$ is called the *Wasserstein gradient* of the KL divergence.[1] Applying update (1) directly is intractable because it usually not possible to efficiently evaluate the density q_k.[2] Instead, we evolve many particles *in parallel*, which allows us to approximate the gradient term using a neural network f_θ and perform the update

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \varepsilon f_\theta(\boldsymbol{x}_k). \tag{2}$$

1. A short overview of Wasserstein gradient flows is available in Appendix A. For more details see the textbooks (Villani, 2008; Ambrosio et al., 2008).
2. For an invertible, differentiable transformation T, the pushforward distribution of q under T can be computed via the change of variables formula $T_\# q(\boldsymbol{x}) = q(T^{-1}\boldsymbol{x}) \cdot \det |J_{T^{-1}}(\boldsymbol{x})|$. The computation of the determinant of a general $d \times d$ matrix needs $\mathcal{O}(d^3)$ operations; when d is large, this is too expensive.

3. Neural variational gradient descent

In this section, we will see how to train a neural network f_θ to approximate the gradient $\nabla \log p - \nabla \log q_k$. We build on the method of Stein variational inference developed by Liu and Wang (2016). In a slightly non-standard manner, let us define the Stein discrepancy (Gorham and Mackey, 2018) given an $L^2(q)$-measureable vector field $f : \mathbb{R}^n \to \mathbb{R}^n$ as

$$\begin{aligned} \mathrm{SD}(q \parallel p; f) &= \mathop{\mathrm{E}}_{\boldsymbol{x}\sim q}\Big[f(\boldsymbol{x})^T\big(\nabla \log p(\boldsymbol{x}) - \nabla \log q(\boldsymbol{x})\big)\Big] \\ &= \mathop{\mathrm{E}}_{\boldsymbol{x}\sim q}\Big[f(\boldsymbol{x})^T \nabla \log p(\boldsymbol{x}) + \operatorname{div} f(\boldsymbol{x})\Big], \end{aligned} \tag{3}$$

where the second equality follows via partial integration (details in Appendix B.1). Here, $\operatorname{div} f = \sum_{i=1}^d \frac{\partial f_i}{\partial x_i}$ denotes the divergence of f.

Algorithm 1: Neural variational gradient descent

Input: Learning rates η and ε, number of gradient updates P, initial samples $x_1, \dots, x_n$.
Output: Samples from the posterior.
while *not converged* **do**
  $X_{\text{train}},\ X_{\text{validation}} \leftarrow \text{RandomSplit}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$;
  for P *steps* **do**
    $\theta \leftarrow \theta + \eta \nabla_\theta \widehat{\mathrm{RSD}}(\theta; X_{\text{train}})$;
    if EarlyStopCondition($X_{validation}$) **then**
      break
    end
  end
  $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \varepsilon f_\theta(\boldsymbol{x}_i)$ for all $1 \le i \le n$;
end

There are two properties of the Stein discrepancy that deserve special emphasis: Firstly, it can be estimated using only samples from q, without access to the explicit form of the density. Secondly, a regularized Stein discrepancy is maximized by the Wasserstein gradient from Equation (7). Indeed, following Grathwohl et al. (2020), let us define the *regularized Stein discrepancy* as

$$\mathrm{RSD}(f_\theta) = \mathrm{SD}(q \parallel p; f_\theta) - \frac{1}{2}\|f_\theta\|^2_{L^2(q)}, \tag{4}$$

where $\|f_\theta\|^2_{L^2(q)} = \mathrm{E}_{\boldsymbol{x}\sim q}[f_\theta(\boldsymbol{x})^T f_\theta(\boldsymbol{x})]$. Now it is easy to confirm that

$$f^* = \operatorname*{argmax}_{f \in L^2(q)} \mathrm{RSD}(q \parallel p; f) = \nabla \log p - \nabla \log q. \tag{5}$$

This choice of regularization is equivalent to bounding the total squared distance by which the particles are moved. The regularization term is necessary because otherwise the optimization could scale f by an arbitrarily large factor.

3.1. Computation

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be a set of samples which we wish to transport towards the target posterior p. In this section, we describe how to compute a single iteration of our method; the entire procedure is summarized in Algorithm 1.

While it is intractable to compute $\mathrm{RSD}(f)$ exactly, we can approximate it with the Monte Carlo estimate

$$\widehat{\mathrm{RSD}}(f) = \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i)^T \nabla \log p(\boldsymbol{x}_i) + \operatorname{div} f(\boldsymbol{x}_i) - \frac{1}{2} f(\boldsymbol{x}_i)^T f(\boldsymbol{x}_i).$$

If the dimension d of the sample space is large, the $O(d^2)$ computation of the divergence $\operatorname{div} f$ might be too expensive. We follow Grathwohl et al. (2020) in using Hutchinson's estimator (Hutchinson, 1989), which gives an efficient and unbiased estimate of $\operatorname{div} f$ via

$$\operatorname{div} f(\boldsymbol{x}) \approx \boldsymbol{z}^T \nabla f(\boldsymbol{x}) \boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(0, 1),$$

which is derived from the identity $\operatorname{div} f(\boldsymbol{x}) = \mathrm{E}_{\boldsymbol{z} \sim \mathcal{N}(0,1)} \left[\boldsymbol{z}^T \nabla f(\boldsymbol{x}) \boldsymbol{z} \right]$, and has been used in a number of recent works (Grathwohl et al., 2020; Han et al., 2017; Grathwohl et al., 2018).

One iteration of our method then takes two steps: First train the model f_θ to maximize $\widehat{\mathrm{RSD}}(f_\theta)$ for a fixed number of steps, then update the particles using the trained model. The full algorithm is described in Algorithm 1. To prevent the model from overfitting, we track the value of the $\widehat{\mathrm{RSD}}$ on a validation set of particles and early-stop the training if the validation $\widehat{\mathrm{RSD}}$ stops increasing.

4. Related work

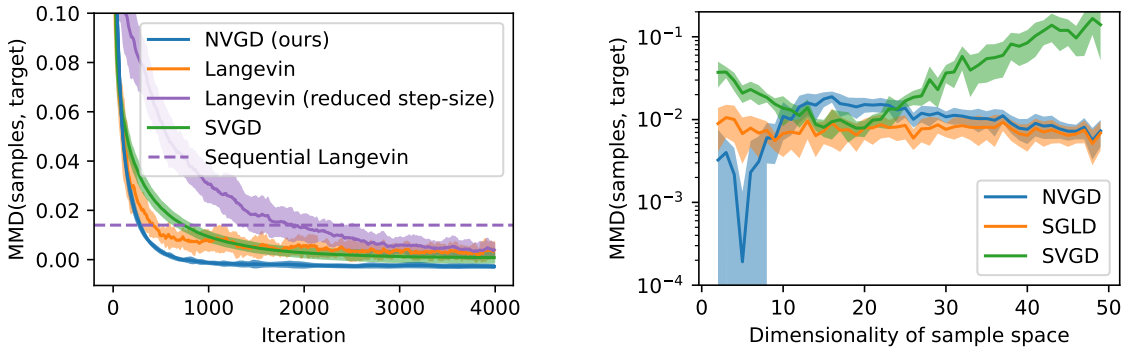
SVGD. Our method is based on the SVGD algorithm (Liu and Wang, 2016). It has been previously noted that this algorithm often fails when the target distribution is high-dimensional and has a complex shape. Attempted remedies include projections (Zhuo et al., 2018; Gong et al., 2020; Chen and Ghattas, 2020) or modifications to the update equations (Gallego and Insua, 2018; Chewi et al., 2020; D'Angelo and Fortuin, 2021a). The main difference between our method and these ones is that they only consider perturbations within an RKHS, thus requiring the explicit choice of a kernel function and restricting the flexibility of the method, while our method does not require any kernel choice whatsoever.

Learning the witness function. The approach used in this paper for learning the Stein discrepancy is modeled after the approach used by Grathwohl et al. (2020) to train and evaluate energy-based models. Another work that uses a similar approach to learn a Stein discrepancy is Hu et al. (2018). Both of these works directly minimize the Stein discrepancy between an approximating distribution[3] and a target. In contrast, our method minimizes the KL divergence by leveraging the relation (5).

Extensions. Many extensions have been proposed for SVGD, including using projections (Chen and Ghattas, 2020), stochastic disrepancies (Gorham et al., 2020), and more. While we focus on the simplest setting as a proof of concept, most of these ideas do not rely on the kernelized form of the update and could also be readily applied to our method.

3. An energy-based model and a set of samples, respectively

5. Experiments



(a) Sampling from the 2-D funnel. (b) Sampling the funnel in dimensions 2–40.

Figure 2: We compare NVGD, SVGD, and Langevin dynamics on the funnel density. We plot the mean and standard deviation of the MMD error across ten runs. Lower is better.

In this section, we test our NVGD on various synthetic benchmarks and one large-data logistic regression task. [4] For most experiments we choose f_θ to be a simple neural network with two linear hidden layers of size 32 and an output layer of size d, where d is the dimensionality of the sample space. More information on the technical setup as well as further experiments are detailed in Appendix C and D.

We will compare NVGD with two closely related sampling algorithms: SVGD (Liu and Wang, 2016) and the unadjusted Langevin algorithm (ULA). Both are approximations of the Wasserstein gradient flow of the KL divergence: SVGD is a particle-based algorithm that at every step applies the particle transformation in a reproducing kernel Hilbert space that locally minimizes the KL divergence. ULA is an MCMC algorithm that has been shown to also minimize the KL divergence. When estimating the posterior density using minibatches, ULA is referred to as stochastic gradient Langevin dynamics (SGLD) Welling and Teh (2011).

For SVGD, one has to choose a kernel. The most frequent choice in the literature is the squared-exponential kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2h^2))$ with bandwidth h chosen according to the median heuristic

$$h^2 = \frac{\mathrm{med}^2}{2 \log n},$$

where med^2 is the median of all squared pairwise distances $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ between samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. We follow this choice in all experiments.

4. Code for all experiments is available at https://github.com/langosco/neural-variational-gradient-descent

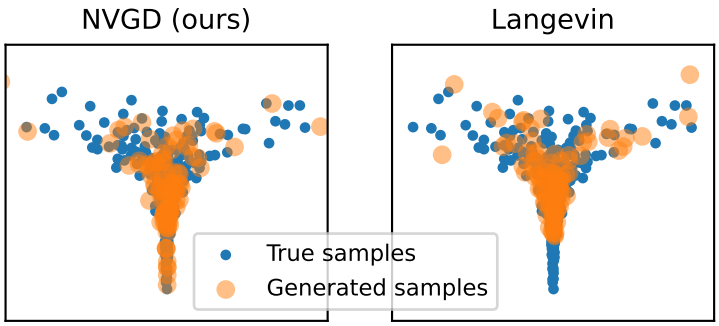


Figure 3: Samples generated for the Funnel density (same setting as Figure 2(a)). The difficulty arises from the funnel geometry: too wide at the top, and too narrow at the bottom. In particular, Langevin dynamics (pULA) is not able to sample from the narrow strait if the step size is too large.

Neal's Funnel (Neal, 2003) is a distribution on $\mathbb{R}^d$ defined via the density

$$p(\boldsymbol{x}) = N(x_1; 0, 3) \cdot \prod_{i=2}^{d} N(x_i; 0, \exp(x_1)). \tag{6}$$

This density is a common benchmark for sampling algorithms because it is easy to sample from directly (and thus easy to track convergence) but also challenging due to its geometry.

We choose $d = 2$ and initialize a set of 100 particles by sampling from a standard Gaussian and develop them along the approximate gradient flows given by the different methods. Figure 2(a) shows how the methods converge, measured by MMD, and Figure 3 is a scatterplot that shows how NVGD and SGLD perform, thus demonstrating why the funnel is a challenging target.

Higher-dimensional funnels. We test performance of NVGD for the Funnel distribution across 40 dimensions. Figure 2(b) shows how the performance of SVGD degrades as the dimensionality increases.

5.1. Bayesian logistic regression

We sample from a Bayesian logistic regression model trained on the binary Covertype dataset(Dua and Graff, 2017). As previously, we compare the three methods SVGD, SGLD, and NVGD, sampling 100 particles for each method. All three methods perform similarly. Test accuracy averaged over ten runs is reported in Appendix C.

6. Conclusion

We introduced NVGD, a novel particle-based variational inference method.We have performed comparisons to the two most relevant competitor methods (SVGD and SGLD), and have demonstrated that our method performs as well as (or better than) them across different tasks, while not requiring the choice of a kernel function.

References

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media, 2008.

Michael Betancourt. The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling. In *International Conference on Machine Learning*, pages 533–540, 2015.

Peng Chen and Omar Ghattas. Projected stein variational gradient descent. *arXiv preprint arXiv:2002.03469*, 2020.

Sinho Chewi, Thibaut Le Gouic, Chen Lu, Tyler Maunu, and Philippe Rigollet. Svgd as a kernelized wasserstein gradient flow of the chi-squared divergence. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, pages 2098–2109, 2020.

Francesco D'Angelo and Vincent Fortuin. Annealed stein variational gradient descent. *arXiv preprint arXiv:2101.09815*, 2021a.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. *arXiv preprint arXiv:2106.11642*, 2021b.

Francesco D'Angelo, Vincent Fortuin, and Florian Wenzel. On stein variational neural network ensembles. *arXiv preprint arXiv:2106.10760*, 2021.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Vincent Fortuin. Priors in bayesian deep learning: A review. *arXiv preprint arXiv:2105.06868*, 2021.

Vincent Fortuin, Adrià Garriga-Alonso, Mark van der Wilk, and Laurence Aitchison. Bnnpriors: A library for bayesian neural network inference with different prior distributions. *Software Impacts*, page 100079, 2021a.

Vincent Fortuin, Adrià Garriga-Alonso, Florian Wenzel, Gunnar Rätsch, Richard Turner, Mark van der Wilk, and Laurence Aitchison. Bayesian neural network priors revisited. *arXiv preprint arXiv:2102.06571*, 2021b.

Victor Gallego and David Rios Insua. Stochastic gradient mcmc with repulsive forces. *arXiv preprint arXiv:1812.00071*, 2018.

Adrià Garriga-Alonso and Vincent Fortuin. Exact langevin dynamics with stochastic gradients. *arXiv preprint arXiv:2102.01691*, 2021.

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Sliced kernelized stein discrepancy. *arXiv e-prints*, pages arXiv–2006, 2020.

Jackson Gorham and Lester Mackey. Measuring Sample Quality with Stein's Method. *arXiv:1506.03039 [cs, math, stat]*, December 2018. URL http://arxiv.org/abs/1506.03039. arXiv: 1506.03039.

Jackson Gorham, Anant Raj, and Lester Mackey. Stochastic stein discrepancies. *arXiv preprint arXiv:2007.02857*, 2020.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Will Grathwohl, Kuan-Chieh Wang, Jorn-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the Stein Discrepancy for Training and Evaluating Energy-Based Models without Sampling. *arXiv:2002.05616 [cs, stat]*, August 2020. URL http://arxiv.org/abs/2002.05616. arXiv: 2002.05616.

Insu Han, Dmitry Malioutov, Haim Avron, and Jinwoo Shin. Approximating spectral sums of large-scale matrices using stochastic chebyshev approximations. *SIAM Journal on Scientific Computing*, 39(4):A1558–A1585, 2017. Publisher: SIAM.

Tianyang Hu, Zixiang Chen, Hanxi Sun, Jincheng Bai, Mao Ye, and Guang Cheng. Stein neural sampler. *arXiv preprint arXiv:1810.03545*, 2018.

Xinyu Hu, Paul Szerlip, Theofanis Karaletsos, and Rohit Singh. Applying svgd to bayesian neural networks for cyclical time-series prediction and inference. *arXiv preprint arXiv:1901.05906*, 2019.

Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989. Publisher: Taylor & Francis.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning. *arXiv preprint arXiv:2104.04975*, 2021a.

Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pages 703–711. PMLR, 2021b.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.

Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998. Publisher: SIAM.

Qiang Liu. Stein variational gradient descent as gradient flow. In *Advances in neural information processing systems*, pages 3115–3123, 2017.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2378–2386. Curran Associates, Inc., 2016.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Radford Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 06 2012. doi: 10.1201/b10905-6.

Radford M. Neal. Bayesian training of backpropagation networks by the Hybrid Monte Carlo method. Technical report, University of Toronto, 1992.

Radford M. Neal. Slice Sampling. *The Annals of Statistics*, 31(3):705–741, 2003. ISSN 0090-5364. URL https://www.jstor.org/stable/3448413. Publisher: Institute of Mathematical Statistics.

Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, February 1998. ISSN 1369-7412, 1467-9868. doi: 10.1111/1467-9868.00123. URL http://doi.wiley.com/10.1111/1467-9868.00123.

Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted langevin algorithm: Isoperimetry suffices. In *Advances in Neural Information Processing Systems*, pages 8094–8106, 2019.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for bayesian neural networks. In *International Conference on Learning Representations*, 2018.

Max Welling and Yee W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing Stein variational gradient descent. In *International Conference on Machine Learning*, pages 6018–6027. PMLR, 2018.

Appendix A. Wasserstein gradient flows

Consider the space $\mathcal{P}$ of probability measures on $\mathbb{R}^n$ equipped with the 2-Wasserstein metric. On this space it is possible to define a *Wasserstein gradient flow*: a continuous trajectory $(q_t)_{t\geq 0}$ in $\mathcal{P}$ that locally (with respect to the Wasserstein metric) minimizes a functional $F(q_t)$.[5] We are interested in the case where this functional is the KL divergence: $F(q) = \mathrm{KL}(q \parallel p)$, where p is a fixed target posterior distribution. In terms of samples $X_t \sim q_t$, this KL-minimizing flow is characterized by the deterministic Markov process $(X_t)_{t\geq 0}$ given by

$$\frac{dX_t}{dt} = \nabla \log p(X_t) - \nabla \log q_t(X_t), \tag{7}$$

where $X_0 \sim q_0$ is initialized according to some initial distribution q_0, and q_t denotes the density of X_t. Under reasonable conditions, this process converges in the sense that $\lim_{t\to\infty} \mathrm{KL}(q_t \parallel p) = 0$ (Vempala and Wibisono, 2019). Thus for large t, the sample X_t is approximately distributed as p.

To generate such a sample we need to discretize the differential equation (7). The direct application of the Euler discretization

$$x_{k+1} = x_k + \varepsilon\Big(\nabla \log p(x_k) - \nabla \log q_k(x_k)\Big)$$

as a sampling method is unfortunately intractable because it is usually not possible to efficiently compute $q_k(x)$.[6] Instead, the discretization that is usually used is Langevin dynamics (SGLD), which Jordan et al. (1998) have shown to be equivalent (in the limit of infinitesimal step-size) to the flow (7) in the sense that the marginal distributions $(q_t)_{t\geq 0}$ are equal. Langevin dynamics requires adding Gaussian noise to the samples. In contrast, our method (based on approximating the Euler discretization (1)) is entirely deterministic.

Appendix B. Proofs

B.1. The Stein Discrepancy

In Equation 3 we defined the Stein discrepancy given f as

$$\begin{aligned}\mathrm{SD}(q \parallel p; f) =& E_{x\sim q}[f(x)^T(\nabla \log p(x) - \nabla \log q(x))] \\ =& E_{x\sim q}[f(x)^T \nabla \log p(x) + \operatorname{div} f(x)].\end{aligned} \tag{8}$$

We will now provide the assumptions under which the second equality holds, and show how it follows from integration by parts.

Proposition 1 *Let q be a probability density on $\mathcal{X}$ and $f : \mathcal{X} \to \mathbb{R}^d$, $f \in L^2(q)$ a differentiable function. Assume that*

$$\int_{\partial\mathcal{X}} f(x)q(x)\,\mathrm{d}x = 0, \tag{9}$$

5. For details on Wasserstein gradient flows we refer to the textbooks (Villani, 2008; Ambrosio et al., 2008).
6. For an invertible, differentiable transformation T, the pushforward distribution of q under T can be computed via the change of variables formula $T_{\#}q(x) = q(T^{-1}x) \cdot \det |J_{T^{-1}}(x)|$. The computation of the determinant of a general $d \times d$ matrix needs $\mathcal{O}(d^3)$ operations; when d is large, this is too expensive.

where $\partial\mathcal{X}$ denotes the boundary of $\mathcal{X}$. If $\mathcal{X}$ is instead all of $\mathbb{R}^d$, then the condition must hold in the limit $r \to \infty$ for integral over the ball B_r of radius r centered at the origin. Then

$$E_{x\sim q}[\operatorname{div} f(x)] = -E_{x\sim q}[f(x)^T \nabla \log q(x)].$$

Proof The proposition follows directly from integration by parts. If $n(x)$ is the outward-pointing unit vector on the boundary of $\mathcal{X}$, then

$$\begin{aligned} E[f(x)^T \nabla \log q(x)] &= \int_{\mathcal{X}} f(x)^T \nabla q(x)\,\mathrm{d}x \\ &= \int_{\partial\mathcal{X}} f(x)^T n(x) q(x)\,\mathrm{d}x - \int_{\mathcal{X}} \operatorname{div} f(x) q(x)\,\mathrm{d}x \\ &= -E[\operatorname{div} f(x)]. \end{aligned}$$

Our desired equality follows. ■

Proposition 1 is used in many problems throughout statistics and deep learning. It is the basis of the policy gradient algorithm in reinforcement learning and the score matching algorithm for density estimation Hyvärinen (2005).

Appendix C. Experiment details

C.1. Synthetic benchmark distributions

ULA and NVGD are discretizations (stochastic and deterministic, respectively) of the Wasserstein gradient flow (7). It is known that in the limit of infinitesimally small step-size, ULA perfectly approximates this flow. Any hope of outperforming ULA must therefore lie in a better approximation of the flow *relative to the step-size*, that is, we need to show that ULA with a step-size $\eta > 0$ is either more biased than NVGD (if η is too large) or takes more steps to converge than NVGD (if η is too small).

When comparing the methods on this task, where it is possible to track convergence exactly, we find that NVGD outperforms ULA. In particular, we compare with two versions of ULA: the standard single-chain ULA, and a parallel version pULA where n chains are initialized and evolved simultaneously. The parallel version is more similar to NVGD, which also transports a number of particles in parallel.

To reduce correlation between sequential ULA samples, most are discarded (e.g., only every 100th sample is retained). We develop a single chain for $5000 \cdot 100 = 5 \cdot 10^5$ steps, corresponding to the same number of likelihood evaluations as 5000 steps for 100 parallel chains. However, even after discarding all but every 100th sample, the resulting samples are still correlated. This results in a high MMD error (Figure 2(a)). For sequential ULA, we use the same learning rate as the pULA run that achieved lower MMD error. When using a higher learning rate instead, we found that the samples were less correlated at the cost of worse coverage of the posterior.

In many applications, ULA is augmented with an accept/reject step that adjusts for its asymptotic bias. We compare to the *unadjusted* version because the accept/reject step is not used in large-data applications (recent work by Garriga-Alonso and Fortuin (2021) has shown it possible to do so using custom integrators, but this is beyond the scope of this

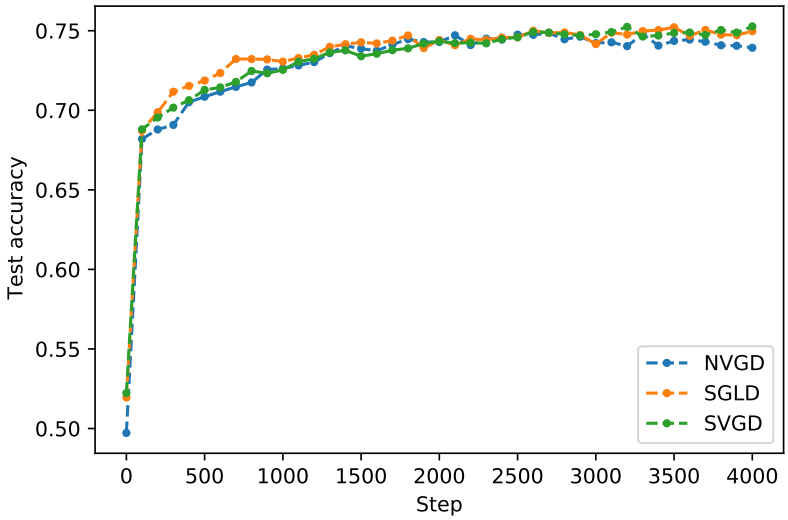


Figure 4: Bayesian logistic regression on the Covertype dataset. We use 100 particles and a mini-batch size of 128, so 4000 steps corresponds to one epoch. Results are averaged over ten runs.

work). Instead, the stochastic gradient variant of ULA (SGLD) is preferred, as for instance in Wenzel et al. (2020). Where an accept/reject step is feasible, NVGD can be similarly augmented, though we do not study this here.

Higher-dimensional funnels In other settings, it has already been shown that high-dimensional sample spaces are often challenging for SVGD (Zhuo et al., 2018; Gong et al., 2020), though in general this depends on the choice of kernel function. Figure 2(b) also seems to show that the SGLD and NVGD are immune to the difficulties of increasing dimensionality. Unfortunately, visual inspection of samples shows that this is just an artefact of the MMD measure, and the performance of the methods does get worse as the dimensionality increases.

One thing that is surprising about Figure 2(b) is that ULA performs equally or better than NVGD, which seems to contradict the results in Figure 2(a). This discrepancy likely stems from two sources: 1) ideally, the learning rate of f_θ and the size of its hidden layers should be adapted as the dimension increases, but we keep it fixed; and 2) the advantage of NVGD—better coverage along the narrow end of the y-axis—becomes less relevant for the MMD measure as the dimension increases, since all $d-1$ other axes are symmetric.

C.2. Bayesian logistic regression

The Covertype dataset consists of $581,012$ observations of 55 features and a binary label. We split off a proportion of 0.2 for use as test set. We follow Liu and Wang (2016) in using

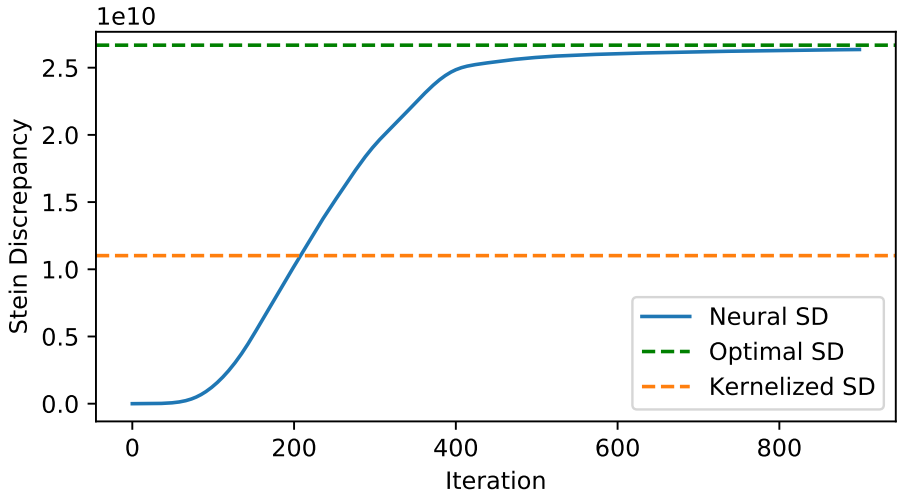


Figure 5: The Stein discrepancy over the course of gradient descent on θ. Larger is better, since the Stein discrepancy is proportional to the amount that one particle iteration reduces the KL divergence (Equation 3). Our proposed NVGD converges closely to the optimal Stein discrepancy.

a hierarchical prior on β:

$$\alpha \sim \text{Gamma}(a_0, b_0),$$
$$\beta \sim \mathcal{N}(0, \alpha^{-1}),$$

choosing the rate and inverse scale parameters as $a_0 = 1$ and $b_0 = 0.01$. The logistic model for (x, y) is then

$$y \sim \text{Bernoulli}\left(\frac{\exp(x^\top \beta)}{1 + \exp(x^\top \beta)}\right).$$

To test our method in the stochastic gradient setting, we estimate the likelihood using minibatches of size 128 and train for one epoch (4085 steps).

We compare again the three methods NVGD, SGLD, and SVGD. For all methods, we sample 100 particles and choose the particle step-size via tuning on a validation set of size 0.1 subsampled from the training set. We see in Figure 4 that all three methods perform similarly. The test accuracy is averaged over ten runs.

Appendix D. Further experiments

D.1. Minimizing the KL in one step

Our first experiment is a sanity check: we confirm that, on a synthetic task, our method approximates the true gradient of the KL, that is $f_\theta(x) \approx f^*(x) = \nabla \log p(x) - \nabla \log q(x)$.

In the synthetic examples below we have access to both p and q (which is not the case in real applications) and can thus compute f^* directly. We confirm that our method successfully learns to approximate f^*.

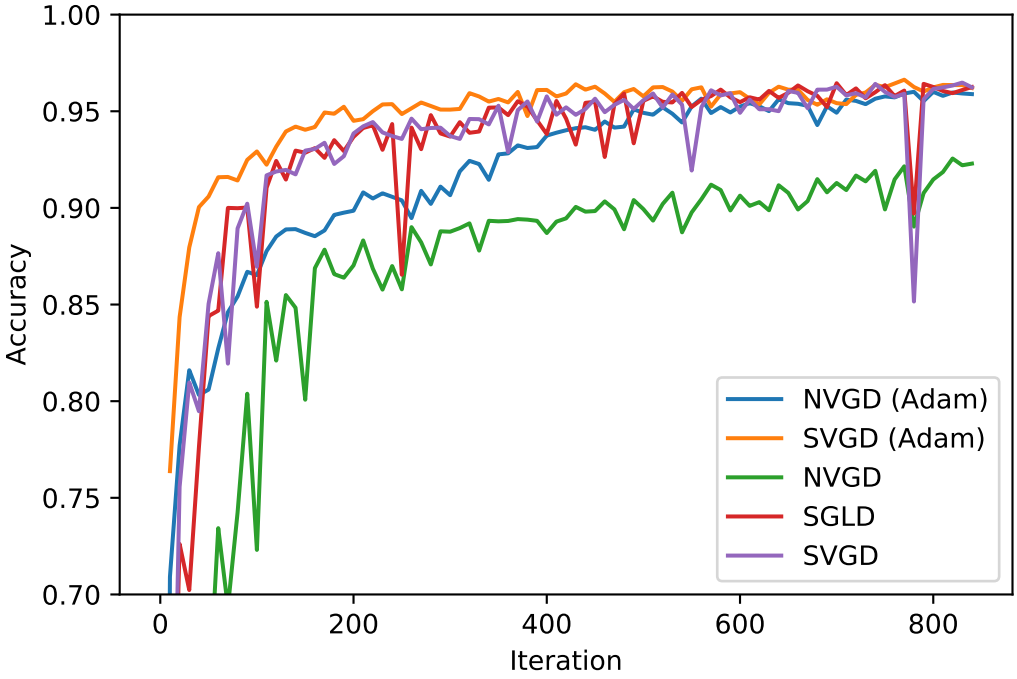


Figure 6: Bayesian neural network inference on MNIST. The vanilla version of our method (NVGD) performs poorly; when combined with the Adam optimizer, it performs better, but still converges more slowly than either SVGD and SGLD. One epoch corresponds to 420 iterations.

Results are shown in Figure 5. For comparison, we also plot the kernelized Stein discrepancy, rescaled such that the kernelized (SVGD) update has L_2-norm equal to f^* (rescaling is necessary because the scale of the update, which corresponds to the particle step size, is a priori arbitrary).

We choose the target posterior p to be an ill-conditioned Gaussian in $\mathbb{R}^{50}$ with mean zero and a diagonal covariance matrix with entries spaced logarithmically from 10^{-4} to 1.

We sample a batch of 1000 particles from a 'proposal' distribution q, here chosen to be a standard Gaussian, and train the network f_θ to maximize the rescaled Stein discrepancy (4) for 1000 iterations. As visible in Figure 5, the network quickly learns to match the Wasserstein gradient.

D.2. Bayesian neural network inference

While Bayesian neural networks (BNNs) (MacKay, 1992; Neal, 1992) have gained a lot of popularity recently (Immer et al., 2021b,a; Fortuin et al., 2021b,a; Fortuin, 2021; Izmailov et al., 2021; Wenzel et al., 2020), SVGD has been used for their inference on only few occasions (Wang et al., 2018; Hu et al., 2019; D'Angelo et al., 2021; D'Angelo and Fortuin, 2021b), possibly due to the difficulty to adapt to their high-dimensional posterior geometry.

We train a BNN on the MNIST dataset and again compare against SVGD and SGLD. The Bayesian classifier network consists of two convolutional and one fully-connected layer with 4594 parameters in total. Results are shown in Figure 6.

For MNIST we use a batch size of 128. We tune the BNN learning rate to maximize accuracy on a validation set comprising 10% of the training set, and apply the Adam optimizer to the SVGD and NVGD updates (this cannot easily be done for the SGLD gradients, so we use vanilla SGLD; for comparison, we also include results for vanilla SVGD and NVGD).

While all three methods reach similar accuracy, NVGD (our method) converges more slowly. Closer observation suggests that the gradient learner f_θ underfits; future work will consider architectures other than the simple linear feedforward network used here.