Coin Sampling: Gradient-Based Bayesian Inference without Learning Rates

Louis Sharrock [1] **Christopher Nemeth** [1]

Abstract

In recent years, particle-based variational inference (ParVI) methods such as Stein variational gradient descent (SVGD) have grown in popularity as scalable methods for Bayesian inference. Unfortunately, the properties of such methods invariably depend on hyperparameters such as the learning rate, which must be carefully tuned by the practitioner in order to ensure convergence to the target measure at a suitable rate. In this paper, we introduce a suite of new particle-based methods for scalable Bayesian inference based on coin betting, which are entirely learning-rate free. We illustrate the performance of our approach on a range of numerical examples, including several high-dimensional models and datasets, demonstrating comparable performance to other ParVI algorithms with no need to tune a learning rate.

1. Introduction

The task of sampling from complex, high-dimensional probability distributions is of fundamental importance to Bayesian inference (Robert & Casella, 2004; Gelman et al., 2013), machine learning (Neal, 1996; Andrieu et al., 2003; Wilson & Izmailov, 2020), molecular dynamics (Krauth, 2006; Lelièvre & Stoltz, 2016), and scientific computing (MacKay, 2003; Liu, 2009). In this paper, we consider the canonical task of sampling from a target probability distribution $\pi(\mathrm{d}x)$ on $\mathbb{R}^d$ with density $\pi(x)$ with respect to the Lebesgue measure of the form[1]

$$\pi(x) := \frac{e^{-U(x)}}{Z}, \tag{1}$$

where $U : \mathbb{R}^d \to \mathbb{R}$ is a continuously differentiable function known as the potential, and $Z = \int_{\mathbb{R}^d} e^{-U(x)} \mathrm{d}x$ is an unknown normalising constant.

[1]Department of Mathematics, Lancaster University, UK. Correspondence to: Louis Sharrock <l.sharrock@lancaster.ac.uk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1]In a slight abuse of notation, we use π to denote both the target distribution and its density.

Recently, there has been growing interest in hybrid sampling methods which combine the non-parametric nature of Markov chain Monte Carlo (MCMC) sampling with the parametric approach used in variational inference (VI). In particular, particle based variational inference (ParVI) methods (Liu & Wang, 2016; Chen et al., 2018a; Liu et al., 2019a) approximate the target distribution using an ensemble of interacting particles, which are deterministically updated by iteratively minimising a metric such as the Kullback-Leibler (KL) divergence.

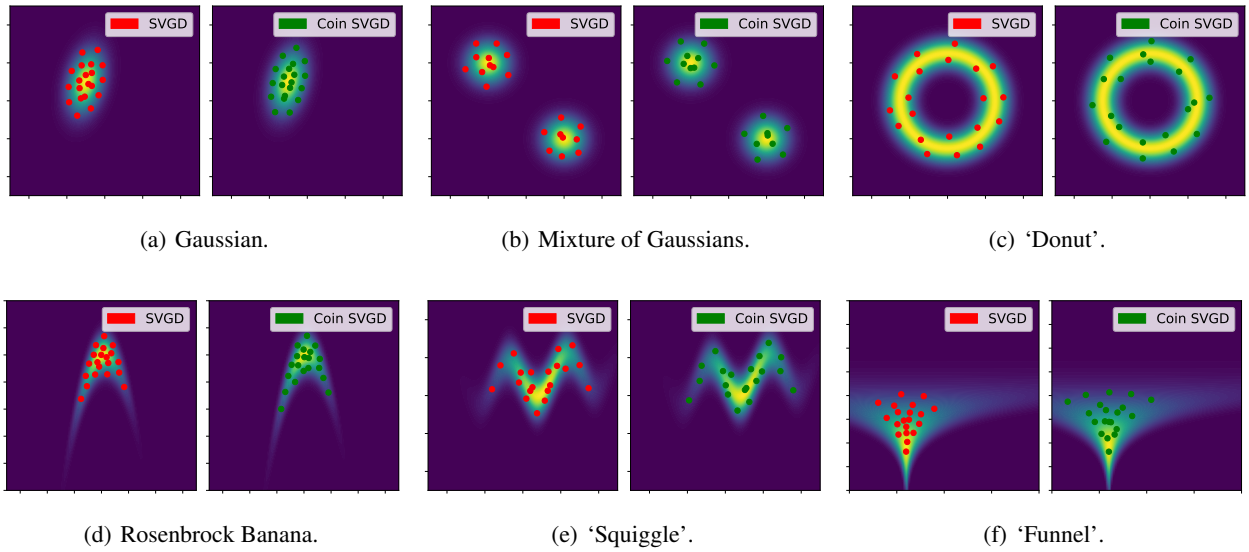
Perhaps the most well known of these methods is Stein variational gradient descent (SVGD) (Liu & Wang, 2016), which iteratively updates the particles according to a form of gradient descent on the KL divergence, with the descent direction restricted to belong to a unit ball in a reproducing kernel Hilbert space (RKHS). This approach has since given rise to several variants (Liu, 2017; Han & Liu, 2018; Liu & Zhu, 2018; Zhuo et al., 2018; Chen et al., 2018b; Detommaso et al., 2018; Futami et al., 2019a;b; Wang et al., 2019; Chen & Ghattas, 2020; Ye et al., 2020; Liu et al., 2022; Sun & Richtárik, 2022); and found success in a range of problems, including uncertainty quantification (Zhu & Zabaras, 2018), reinforcement learning (Haarnoja et al., 2017; Liu et al., 2017; Zhang et al., 2018), learning deep probabilistic models (Pu et al., 2017; Wang & Liu, 2017), and Bayesian meta-learning (Feng et al., 2017; Yoon et al., 2018).

In order to construct and analyse sampling algorithms of this type, one popular approach is to reformulate the sampling problem as an optimisation problem in the space of measures (Jordan et al., 1998; Wibisono, 2018; Cheng & Bartlett, 2018; Durmus et al., 2019). In this setting, one views the target π as the solution of an optimisation problem

$$\pi = \underset{\mu \in \mathcal{P}_2(\mathbb{R}^d)}{\arg\min}\, \mathcal{F}(\mu), \tag{2}$$

where $\mathcal{P}_2(\mathbb{R}^d)$ denotes the set of probability measures $\{\mu : \int_{\mathbb{R}^d} ||x||^2 \mu(\mathrm{d}x) < \infty\}$, and $\mathcal{F} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ is a functional which is uniquely minimised at π. A general strategy for solving this problem is then to simulate a time-discretisation of the gradient flow of $\mathcal{F}$ over $\mathcal{P}_2(\mathbb{R}^d)$, having equipped this space with a suitable metric (Ambrosio et al., 2008).

Many popular sampling algorithms can be understood from this perspective. For example, Langevin Monte Carlo (LMC), a popular MCMC algorithm, corresponds to the



(a) Gaussian. (b) Mixture of Gaussians. (c) 'Donut'.

(d) Rosenbrock Banana. (e) 'Squiggle'. (f) 'Funnel'.

Figure 1. **A comparison between SVGD (Liu & Wang, 2016) and its learning-rate free analogue, Coin SVGD (Alg. 2).** We plot the samples generated by both methods for several two-dimensional target distributions. Further details are provided in Sec. 4 and App. E.1.

so-called forward-flow discretisation of the gradient flow of the KL divergence with respect to the quadratic Wasserstein metric (Wibisono, 2018; Durmus et al., 2019).[2] Meanwhile, SVGD can be viewed as the explicit Euler discretisation of the gradient flow of the KL divergence with respect to a kernelised Wasserstein metric (Liu, 2017; Duncan et al., 2023). Other more recent examples, designed with this perspective in mind, include maximum mean discrepancy (MMD) gradient descent (Arbel et al., 2019), the Wasserstein proximal gradient algorithm (Salim et al., 2020), kernel Stein discrepancy descent (KSDD) (Korba et al., 2021), Laplacian adjusted Wasserstein gradient descent (LAWGD) (Chewi et al., 2020), mollified interaction energy descent (MIED) (Li et al., 2023), and the various other ParVI methods described in Chen et al. (2018a); Liu et al. (2019a;b).

One feature common to all of these approaches is the need to specify an appropriate learning rate (i.e., step size) γ, or a learning rate schedule $(\gamma_t)_{t\geq 1}$. This learning rate must be sufficiently small to ensure convergence to the target measure, or a close approximation thereof, but also large enough to ensure convergence within a reasonable time period. In theory, for a given target π, existing convergence rates allow one to derive an optimal learning rate (see, e.g., Korba et al., 2020; Salim et al., 2022; Sun & Richtárik, 2022 for SVGD; Dalalyan, 2017a;b; Durmus & Moulines, 2017; Dalalyan & Karagulyan, 2019; Durmus & Moulines, 2019 for LMC). Invariably, however, the optimal learning rate is a function of the unknown target measure (e.g., Corollary 6 in Korba et al., 2020; Theorem 9 in Durmus et al., 2019) and thus, in practice, cannot be computed.

With these considerations in mind, a natural question is whether one can obtain a gradient-based sampling method which does not require a learning rate. In this paper, we answer this question in the affirmative. In particular, inspired by the parameter-free optimisation methods developed by Orabona and coworkers (Orabona, 2014; Orabona & Pal, 2016; Orabona & Tommasi, 2017; Cutkosky & Orabona, 2018; Jun & Orabona, 2019; Chen et al., 2022a), and leveraging the view of sampling as an optimisation problem in the space of measures (Wibisono, 2018), we obtain a new suite of particle-based algorithms for scalable Bayesian inference which are entirely learning rate free. Similar to other ParVIs, our algorithms deterministically update an ensemble of interacting particles in order to approximate the target distribution. However, unlike other ParVIs, our algorithms do not correspond to the time-discretisation of any gradient flow, and thus bear little resemblance to existing methods.

Under the assumption of log-concavity, we outline how to establish convergence to the target measure in the infinite-particle regime and obtain a non-asymptotic convergence rate. We then illustrate the performance of our approach on a range of numerical examples, including both convex and non-convex targets. Our results indicate that the proposed methodology achieves comparable performance to existing particle-based sampling algorithms in a range of tasks, with no need to tune a learning rate.

[2]We note that the connection between the law of the over-damped Langevin diffusion (i.e., the continuous-time dynamics of LMC) and the gradient flow of the KL divergence dates back to Otto et al. (Jordan et al., 1998; Otto, 2001; Otto & Westdickenberg, 2005).

2. Preliminaries

2.1. Optimisation in Euclidean Space

We begin by reviewing optimisation in Euclidean spaces, focusing on the learning-rate free stochastic optimisation method introduced by Orabona & Pal (2016). This will later provide the foundation for our learning-rate free sampling algorithms.

2.1.1. Notation

Let $\mathcal{X} \subseteq \mathbb{R}^d$, and write $||\cdot||$ and $\langle \cdot, \cdot \rangle$ for the Euclidean norm and inner product in $\mathbb{R}^d$. Let $f : \mathcal{X} \to \mathbb{R} \cup \{-\infty, \infty\}$, and let $f^* : \mathcal{X}^* \to \mathbb{R} \cup \{-\infty, \infty\}$ denote the Fenchel conjugate of f, so that $f^*(u) = \sup_{x \in \mathcal{X}} [\langle u, x \rangle - f(x)]$.

Suppose that f is m-strongly convex, for some $m \geq 0$. Let $x \in \mathcal{X}$. We say that $g \in \mathcal{X}$ is a subgradient of f at x, and write $g \in \partial f(x)$ if, for any $z \in \mathcal{X}$,

$$f(z) - f(x) \geq \langle g, z - x \rangle + \frac{m}{2}||z - x||^2. \tag{3}$$

If f is differentiable at x, then the differential set $\partial f(x)$ contains a single element, $\partial f(x) = \{\nabla f(x)\}$, where $\nabla f(x)$ denotes the gradient of f at x.

2.1.2. Euclidean Gradient Flows

We begin by considering the optimisation problem

$$x^* = \arg\min_{x \in \mathcal{X}} f(x), \tag{4}$$

where $f : \mathcal{X} \to \mathbb{R}$ is m-strongly convex. We can solve this problem using the gradient flow of f, defined as the solution $x : [0, \infty) \to \mathbb{R}^d$ of the following differential inclusion

$$\dot{x}_t \in -\partial f(x_t), \tag{5}$$

initialised at $x_0 \in \mathcal{X}$. This inclusion admits a unique, absolutely continuous solution for almost all $t \geq 0$ (e.g., Theorem 3.1 in Brézis, 1973, Theorem 2.7 in Peypouquet & Sorin, 2010; Proposition 2.1 in Santambrogio, 2017). Moreover, the function $t \mapsto f(x_t)$ is decreasing, with $\lim_{t \to \infty} f(x_t) = \inf_{x \in \mathcal{X}} f(x)$ (Peypouquet & Sorin, 2010, Proposition 3.1).

In practice, it is necessary to use a time-discretisation of this gradient flow. One standard choice is a backward Euler discretisation, which results in the proximal point algorithm (Güler, 1991; De Giorgi, 1993). Alternatively, one can utilise a forward Euler discretisation, which results in the standard subgradient descent algorithm (Shor, 1985)

$$x_{t+1} = x_t - \gamma g_t \,, \quad g_t \in \partial f(x_t). \tag{6}$$

The properties of this algorithm depend, necessarily, on the choice of learning rate $\gamma > 0$. For example, given an L-Lipschitz function, it is well known that the average of the algorithm iterates $\bar{x}_T = \frac{1}{T}\sum_{t=1}^T x_t$ satisfies (e.g., Zinkevich, 2003)

$$f(\bar{x}_T) - f(x^*) \leq \frac{1}{T}\left[\frac{||x_1 - x^*||^2}{2\gamma} + \frac{L^2 T \gamma}{2}\right]. \tag{7}$$

Using this expression, one can obtain the 'ideal' learning rate as $\gamma_{\text{ideal}} = \frac{||x_1 - x^*||}{L\sqrt{T}}$, which implies the optimal error bound

$$f(\bar{x}_T) - f(x^*) \leq \frac{L||x_1 - x^*||}{\sqrt{T}}. \tag{8}$$

In practice, however, it is not possible to achieve this bound. Indeed, even in hindsight, one cannot compute the ideal learning rate γ_{ideal}, since it depends on the unknown $||x_1 - x^*||$.

2.1.3. Learning-Rate Free Gradient Descent

Following Orabona & Pal (2016), we now outline an alternative approach for solving the stochastic optimisation problem in (4) which is entirely learning-rate free. Consider a gambler who bets on the outcomes of a series of adversarial coin flips. Suppose that the gambler starts with an initial wealth $w_0 = \varepsilon > 0$. In the t^{th} round, the gambler bets on the outcome of a coin flip $c_t \in \{-1, 1\}$, where $+1$ denotes heads and -1 denotes tails. For now, we make no assumptions on how c_t is generated.

We will encode the gambler's bet in the t^{th} round by $x_t \in \mathbb{R}$. In particular, $\text{sign}(x_t) \in \{-1, 1\}$ will denote whether the bet is on heads or tails, and $|x_t| \in \mathbb{R}$ will denote the size of the bet. Thus, in the t^{th} round, the gambler wins $x_t c_t$ if $\text{sign}(c_t) = \text{sign}(x_t)$; and loses $x_t c_t$ otherwise. Finally, we will write w_t for the wealth of the gambler at the end of the t^{th} round. Clearly, we then have that

$$w_t = \varepsilon + \sum_{i=1}^t c_i x_i. \tag{9}$$

We will restrict our attention to the case in which the gambler's bets satisfy $x_t = \beta_t w_{t-1}$, for some betting fraction $\beta_t \in [-1, 1]$. This is equivalent to the assumption that the gambler cannot borrow any money.

We will now outline how to solve the convex optimisation problem $x^* = \arg\min_{x \in \mathbb{R}} f(x)$ using a coin-betting algorithm. For simplicity, we will restrict our attention to the simple one-dimensional function $f(x) = |x - 10|$. We note, however, that this approach can easily be extended to any convex function $f : \mathbb{R}^d \to \mathbb{R}$ (Orabona & Pal, 2016). Suppose we define the outcome of a coin flip $c_t \in \{-1, 1\}$ to be equal to $-g_t \in -\partial[f(x_t)]$, the negative subgradient of $f(x_t)$. In this case, under a certain assumption on the betting strategy $(\beta_t)_{t=1}^T$, Orabona & Pal (2016) show that the average of bets $f(\bar{x}_T)$ converges to $f(x^*)$, with a rate which depends on the quality of the betting strategy.

Lemma 2.1. *Suppose that the betting strategy $(\beta_t)_{t=1}^T$ guarantees that, for any sequence of coin flips $(c_t)_{t=1}^T \in \{-1, 1\}$, there exists a function $h : \mathbb{R} \to \mathbb{R}$ such that the wealth after T rounds satisfies $w_T \geq h(\sum_{t=1}^T c_t)$. Then*

$$f\left(\frac{1}{T}\sum_{t=1}^T x_t\right) - f(x^*) \leq \frac{h^*(x^*) + \varepsilon}{T}. \tag{10}$$

Proof. See App. B. □

We can thus use any suitable coin-betting algorithm to obtain $x^* = \arg\min_{x\in\mathbb{R}} f(x)$, given access to the subgradients of f. Moreover, any such algorithm will be entirely learning-rate free. There are various betting strategies which satisfy the requirement that $w_T \geq h(\sum_{t=1}^T c_t)$ (e.g., Orabona & Pal, 2016; Orabona & Tommasi, 2017; Chen et al., 2022a). Perhaps the simplest such strategy is one based on the Krichevsky-Trofimov (KT) estimator (Krichevsky & Trofimov, 1981), which defines the betting strategy to be equal to $\beta_t = \sum_{i=1}^{t-1} c_i/t$. This results in the coin betting algorithm

$$x_t = -\frac{\sum_{i=1}^{t-1} g_i}{t}\left(\varepsilon - \sum_{i=1}^{t-1} g_i x_i\right). \tag{11}$$

In this case, it is possible to show (Orabona & Pal, 2016, Lemma 14) that the wealth is lower bounded by

$$h\left(\sum_{t=1}^T c_t\right) = \frac{\varepsilon}{K\sqrt{T}}\exp\left(\frac{\left(\sum_{t=1}^T c_t\right)^2}{2T}\right), \tag{12}$$

where K is a universal constant. Thus, using Lemma 2.1 and an appropriate bound on the convex conjugate of h, one obtains (Orabona & Pal, 2016, Corollary 5)

$$f(\bar{x}_T) - f(x^*) \leq K\frac{||x^*||\sqrt{\log(1 + \frac{24T^2||x^*||^2}{\varepsilon^2})} + \varepsilon}{\sqrt{T}}. \tag{13}$$

It is instructive to compare this bound with (8), the corresponding bound for subgradient descent with an optimally chosen learning rate. Although the coin-betting approach does not quite achieve the optimal bound in (8), it comes close, containing only an additional log-factor. This can be viewed as the trade-off for the fact that the algorithm is now learning-rate free.

3. Coin Sampling for Bayesian Inference

Our approach, summarised in Alg. 1, can be viewed as a natural extension of the learning-rate free optimisation methods introduced in Sec. 2.1.3 to the Wasserstein space. In particular, coin sampling replaces Euclidean gradients with Wasserstein gradients in the coin-betting framework, and can thus be used to solve optimisation problems on the space of probability measures, i.e., for Bayesian inference.

3.1. Optimisation in Wasserstein Space

To extend coin betting to our setting, we will require some basic concepts from optimal transport, including the definition of the Wasserstein space, and of a Wasserstein gradient flow. We provide additional details on geodesic convexity and subdifferential calculus in App. A; see also the books of Ambrosio et al. (2008) and Villani (2008).

3.1.1. The Wasserstein Space

Let $\mathcal{P}_2(\mathbb{R}^d)$ denote the set of probability measures on $\mathbb{R}^d$ with finite 2nd moment: $\int_{\mathbb{R}^d} ||x||^2\mu(\mathrm{d}x) < \infty$. For any $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, let $L^2(\mu)$ denote the set of measurable functions $f : \mathbb{R}^d \to \mathbb{R}^d$ such that $\int_{\mathbb{R}^d} ||f(x)||^2\mu(\mathrm{d}x) < \infty$. We will write $||\cdot||^2_{L^2(\mu)}$ and $\langle\cdot,\cdot\rangle_{L^2(\mu)}$ to denote, respectively, the norm and the inner product of this space.

Given a probability measure $\mu \in \mathcal{P}_2(\mathbb{R}^d)$ and a measurable function $T : \mathbb{R}^d \to \mathbb{R}^d$, we write $T_\#\mu$ for the pushforward measure of μ under T, that is, the measure such that $T_\#\mu(B) = \mu(T^{-1}(B))$ for all Borel measurable $B \in \mathcal{B}(\mathbb{R}^d)$. For every $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^d)$, let $\Gamma(\mu, \nu)$ be the set of couplings (or transport plans) between μ and ν, defined as $\Gamma(\mu,\nu) = \{\gamma \in \mathcal{P}_2(\mathbb{R}^d) : Q^1_\#\gamma = \mu, Q^2_\#\gamma = \nu\}$, where Q^1 and Q^2 denote the projections onto the first and second components of $\mathbb{R}^d \times \mathbb{R}^d$. The Wasserstein 2-distance between μ and ν is then defined according to

$$W_2^2(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \int_{\mathbb{R}^d\times\mathbb{R}^d} ||x - y||^2\gamma(\mathrm{d}x, \mathrm{d}y). \tag{14}$$

The Wasserstein distance W_2 is a distance over $\mathcal{P}_2(\mathbb{R}^d)$. Thus $(\mathcal{P}_2(\mathbb{R}^d), W_2)$ is a metric space of probability measures, known as the Wasserstein space.

3.1.2. Wasserstein Gradient Flows

Recall the optimisation problem from Sec. 1,

$$\pi = \arg\min_{\mu\in\mathcal{P}_2(\mathbb{R}^d)} \mathcal{F}(\mu), \tag{15}$$

where $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to (-\infty, \infty]$ is a proper, lower semi-continuous functional uniquely minimised at π. There are various possible choices for the dissimilarity functional $\mathcal{F}$ (see, e.g., Simon-Gabriel, 2018). In the context of Bayesian inference, perhaps the most common choice is $\mathrm{KL}(\mu|\pi)$, the Kullback-Leibler (KL) divergence of μ with respect to π. Other possibilities include the chi-squared divergence $\mathcal{X}^2(\mu|\pi)$ (Chewi et al., 2020), and the maximum mean discrepancy $\mathrm{MMD}(\mu|\pi)$ (Arbel et al., 2019), of which the kernel Stein discrepancy $\mathrm{KSD}(\mu|\pi)$ (Korba et al., 2021) is a special case.

Similarly to the Euclidean case, typical solutions to (15) are based on the use of a gradient flow. In particular, one can now consider the Wasserstein gradient flow of $\mathcal{F}$, defined

as the weak solution $\mu : [0, \infty) \to \mathcal{P}_2(\mathbb{R}^d)$ of the continuity equation (Ambrosio et al., 2008, Chapter 11)

$$\partial_t \mu_t + \nabla \cdot (v_t \mu_t) = 0 \,, \quad v_t \in -\partial \mathcal{F}(\mu_t), \tag{16}$$

where $\partial \mathcal{F}(\mu)$ denotes the Fréchet subdifferential $\partial \mathcal{F}(\mu)$ of $\mathcal{F}$ at μ (see App. A for a precise definition). Under mild conditions, this equation admits a unique solution for any initial condition (e.g., Theorem 11.1.4 and Theorem 11.2.1 in Ambrosio et al., 2008; Proposition 4.13 in Santambrogio, 2017). In addition, the function $t \mapsto \mathcal{F}(\mu_t)$ is decreasing, so that $\lim_{t\to\infty} \mathcal{F}(\mu_t) = \inf_{\mu \in \mathcal{P}_2(\mathbb{R}^d)} \mathcal{F}(\mu)$ (Ambrosio et al., 2008, Chapter 11).

3.1.3. Discretised Wasserstein Gradient Flows

For practical purposes, it is once more necessary to discretise the gradient flow in (16). One option is the backward Euler discretisation, which corresponds to the minimising movement (Ambrosio et al., 2008, Definition 2.0.6) or JKO (Jordan et al., 1998) scheme. Another natural choice is the forward Euler scheme, which yields the Wasserstein (sub)gradient descent algorithm (e.g., Guo et al., 2022)

$$\mu_{t+1} = (\mathbf{id} - \gamma \xi_t)_{\#} \mu_t \,, \quad \xi_t \in \partial \mathcal{F}(\mu_t). \tag{17}$$

For different choices of the functional $\mathcal{F}$, this discretisation yields the population limit of several existing particle-based algorithms. These include MMD gradient descent (Arbel et al., 2019), KSDD (Korba et al., 2021), and, replacing the Wasserstein gradient (17) by a kernel approximation, SVGD (Liu & Wang, 2016) and LAWGD (Chewi et al., 2020).

Regardless of the choice of numerical discretisation, the properties of the resulting algorithm depend, necessarily, on the choice of learning rate $\gamma > 0$. To illustrate this point, we recall the following bound for the Wasserstein subgradient descent algorithm (Guo et al., 2022, Theorem 8)

$$\mathcal{F}\left(\bar{\mu}_T\right) - \mathcal{F}(\pi) \leq \frac{1}{T}\left[\frac{W_2^2(\mu_1, \pi)}{2\gamma} + \frac{L^2 T \gamma}{2}\right], \tag{18}$$

where $\bar{\mu}_T = \frac{1}{T}\sum_{t=1}^T \mu_t$, which holds under the assumption that the Wasserstein subgradients $||\xi_t||_{L^2(\mu_t)} \leq L$. We note that a similar bound also holds for the Langevin Monte Carlo (LMC) algorithm (Durmus et al., 2019, Sec. 3).

Based on (18), one can obtain the optimal worst case learning rate as $\gamma_{\text{ideal}} = \frac{W_2(\mu_1,\pi)}{L\sqrt{T}}$, and thus the optimal error bound is given by

$$\mathcal{F}\left(\bar{\mu}_T\right) - \mathcal{F}(\pi) \leq \frac{L W_2(\mu_1, \pi)}{\sqrt{T}}. \tag{19}$$

Similar to the Euclidean case, however, this rate cannot be achieved in practice. In particular, computing γ_{ideal} now depends on the unknown Wasserstein distance $W_2(\mu_1, \pi)$.

3.2. Coin Wasserstein Gradient Descent

We now introduce an alternative approach to solving (15) which is entirely learning rate free. Consider a gambler, indexed by some initial bet $x_0 \in \mathbb{R}^d$, who bets on a series of outcomes $c_t = c_t(x_0) \in [-1, 1]^d$. Similar to before, we assume that this gambler has initial wealth $w_0 > 0$. In the t^{th} round, we now suppose that this gambler bets $x_t - x_0 \in \mathbb{R}^d$ on the outcome $c_t \in \mathbb{R}^d$. The wealth $w_t = w_t(x_0)$ of the gambler thus accumulates as

$$w_t = w_0 + \sum_{s=1}^{t} \langle c_s, x_s - x_0 \rangle. \tag{20}$$

We will assume, similar to before, that the bets $x_t - x_0$ satisfy $x_t - x_0 = \beta_t w_{t-1}$, for some vector-valued betting fraction $\beta_t = \beta_t(x_0) \in [-1, 1]^d$. In fact, henceforth we will always assume that $\beta_t = \frac{1}{t}\sum_{s=1}^{t-1} c_s$, which corresponds to the KT betting strategy. The sequence of bets is thus given by

$$x_t = x_0 + \frac{\sum_{s=1}^{t-1} c_s}{t}\left(w_0 + \sum_{s=1}^{t-1} \langle c_s, x_s - x_0 \rangle\right). \tag{21}$$

Suppose, now, that in fact $x_0 \sim \mu_0$, for some 'initial betting distribution' $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$. In addition, suppose that we write $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$ for the function which maps $x_0 \mapsto x_t$. We can then define a sequence of 'betting distributions' $\mu_t \in \mathcal{P}_2(\mathbb{R}^d)$ as the push-forwards of μ_0, under $\varphi_t : \mathbb{R}^d \to \mathbb{R}^d$, viz,

$$\mu_t = (\varphi_t)_{\#} \mu_0. \tag{22}$$

This implies, in particular, that given $x_0 \sim \mu_0$, the random variable $x_t := \varphi_t(x_0)$ is distributed according to μ_t.

We propose to use this framework to solve the minimisation problem in (15). In particular, taking inspiration from Orabona & Pal (2016), we will consider a betting game in which the bets are given by (21), and the outcomes are given by $c_t = -\frac{1}{L}\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)$, where L is an upper bound on the Wasserstein gradients (see Assumption 3.2). We will refer to this betting game, summarised in Alg. 1, as *coin Wasserstein gradient descent* or *coin sampling*.

3.3. Theoretical Results

In this case, under a rather strong sufficient condition (see App. B), we can show that the average of the betting distributions $\frac{1}{T}\sum_{t=1}^T \mu_t$ converges to the target π, at a rate determined by the betting strategy. For this result to hold, we will also require the following assumptions.

Assumption 3.1. The functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to (-\infty, \infty]$ is (i) proper and lower semi-continuous, and (ii) geodesically convex.

Assumption 3.2. There exists $L > 0$ such that, for all $t \in [T]$, $||\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)|| \leq L$.

Algorithm 1 Coin Wasserstein Gradient Descent

Input: initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}_d)$, initial parameter $x_0 \sim \mu_0$, initial wealth $w_0 \in \mathbb{R}_+$, dissimilarity functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to (-\infty, \infty]$, gradient upper bound L.
for $t = 1$ **to** T **do**
Compute

$$x_t = x_0 - \frac{\sum_{s=1}^{t-1} \nabla_{W_2}\mathcal{F}(\mu_s)(x_s)}{Lt}\left(w_0 - \sum_{s=1}^{t-1}\langle \frac{1}{L}\nabla_{W_2}\mathcal{F}(\mu_s)(x_s), x_s - x_0\rangle\right) \quad , \quad \mu_t = (\varphi_t)_{\#}\mu_0. \tag{23}$$

Output: μ_T or $\frac{1}{T}\sum_{t=1}^{T}\mu_t$.

Assumption 3.1(i) is a general technical condition satisfied in all relevant cases (e.g., Ambrosio et al., 2008, Sec. 10). Assumption 3.1(ii) is a standard condition used in the analysis of existing algorithms such as LMC (Wibisono, 2018; Durmus & Moulines, 2019). This assumptions holds, for example, if $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$, and the potential $U : \mathbb{R}^d \to \mathbb{R}$ is convex (Ambrosio et al., 2008, Sec. 9.4).

To our knowledge, Assumption 3.2 has also only explicitly appeared in the analysis of the Wasserstein subgradient descent algorithm in Guo et al. (2022). However, similar conditions have also been used to analyse the convergence of SVGD to its population limit (Liu et al., 2017, Theorem 3.2; Korba et al., 2020, Proposition 7). Meanwhile, convergence rates for SVGD (in the infinite particle regime) can be established under boundedness assumptions for the kernel function, as well as bounds on either the KSD (Liu et al., 2017), the Stein Fisher information (Korba et al., 2020), or the Hessian of the potential (Salim et al., 2022; Shi & Mackey, 2022) at each iteration.

Proposition 3.3. *Let Assumptions 3.1 - 3.2 and Assumption B.1 (see App. B) hold. Then*

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\Bigg[w_0 + \int_{\mathbb{R}^d} ||x|| \sqrt{T \ln\left(1 + \frac{96K^2T^2\,||x||^2}{w_0^2}\right)}\pi(\mathrm{d}x) + \int_{\mathbb{R}^d} ||x|| \sqrt{T \ln\left(1 + \frac{96T^2\,||x||^2}{w_0^2}\right)}\mu_0(\mathrm{d}x)\Bigg]. \tag{24}$$

Proof. See App. B. □

The proof of Proposition 3.3 closely follows the proof used to establish the convergence rate of the parameter-free optimisation algorithm in Orabona & Pal (2016). In our case, however, it is no longer evident how to convert a lower bound on the wealth into an upper bound on the regret (see Lemma 2.1). In App. B, we provide a technical sufficient condition (Assumption B.1) which allow us to obtain the rate in Proposition 3.3. It is unclear, however, how to verify this condition in practice. We leave as an open question whether it is possible to obtain more easily verifiable conditions under which this result still holds.

3.4. Practical Implementation

In principle, Alg. 1 requires knowledge of a bound on the Wasserstein gradients (see Assumption 3.2). If such a constant is unknown in advance, then it can be adaptively estimated using a similar approach to the one proposed in Orabona & Tommasi (2017). We provide full details of this adaptive approach, which in practice we use in all of our numerical experiments, in App. D.

Alg. 1 also assumes that it is possible to observe the sequence of vector fields $(\nabla_{W_2}\mathcal{F}(\mu_t)_{t\in[T]}$. In practice, this is unrealistic: these quantities depend on knowledge of the measures $(\mu_t)_{t\in[T]}$, which typically we cannot compute in closed form. Following existing ParVIs, a standard approach is to approximate these quantities using a set of interacting particles. In particular, suppose we initialise $(x_0^i)_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mu_0(\mathrm{d}x)$, with empirical law $\mu_0^N = \frac{1}{N}\sum_{i=1}^n \delta_{x_0^i}$. We can then update the particles according to an empirical version of (23). This yields, after each iteration, particles $(x_t^i)_{i=1}^N$, with empirical distribution $\mu_t^N = \frac{1}{N}\sum_{i=1}^N \delta_{x_t^i}$.

This approach relies, crucially, on being able to compute or approximate $(\nabla_{W_2}\mathcal{F}(\mu_t^N))_{t\in[T]}$, the Wasserstein gradients of $\mathcal{F}$ evaluated at $(\mu_t^N)_{t\in[T]}$. Fortunately, this is also central to existing particle-based sampling algorithms, including SVGD (Liu & Wang, 2016), KSDD (Korba et al., 2020), and LAWGD (Chewi et al., 2020). We can thus take inspiration from these methods to compute or to approximate the required terms. In fact, for different choices of $\mathcal{F}$, and different approximations of $\nabla_{W_2}\mathcal{F}(\mu_t^N)$, we obtain learning-rate free versions of SVGD (Alf. 2), LAWGD (App. C.1), and KSDD (App. C.2). We refer to these algorithms as Coin SVGD, Coin LAWGD, and Coin KSDD, respectively.

Coin Stein Variational Gradient Descent. We now provide further details on Coin SVGD. Let $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$, with $\nabla_{W_2}\mathcal{F}(\mu) = \nabla \ln \frac{\mu}{\pi}$. Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ denote a positive semi-definite kernel, and $\mathcal{H}_k$ the associated re-

Algorithm 2 Coin Stein Variational Gradient Descent (Coin SVGD)

Input: initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}_d)$, initial particles $(x_0^i)_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mu_0$, initial wealth of particles $(w_0^i)_{i=1}^N \in \mathbb{R}_+$, kernel k, gradient upper bound L.
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do**
    Compute, with $P_{\mu_s^N,k}\nabla_{W_2}\mathcal{F}(\mu_s^N)(\cdot)$ defined as in (26),

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1} P_{\mu_s^N,k}\nabla_{W_2}\mathcal{F}(\mu_s^N)(x_s^i)}{Lt}\left(w_0^i - \sum_{s=1}^{t-1}\left\langle \frac{1}{L}P_{\mu_s^N,k}\nabla_{W_2}\mathcal{F}(\mu_s^N)(x_s^i), x_s^i - x_0^i\right\rangle\right), \tag{25}$$

    Define $\mu_t^N = \frac{1}{N}\sum_{i=1}^N \delta_{x_t^i}$.
Output: μ_T^N or $\frac{1}{T}\sum_{t=1}^T \mu_t^N$.

producing kernel Hilbert space (RKHS). Finally, let $P_{\mu,k} : L^2(\mu) \to L^2(\mu)$ denote the integral operator defined according to $P_{\mu,k}f(\cdot) = \int_{\mathbb{R}^d} k(x,\cdot)f(x)\mu(\mathrm{d}x)$.

Following Liu & Wang (2016), suppose that we replace $\nabla_{W_2}\mathcal{F}(\mu)$ by $P_{\mu,k}\nabla_{W_2}\mathcal{F}(\mu)$ in Alg. 1, its image under the integral operator $P_{\mu,k}$. This essentially plays the role of the Wasserstein gradient in $\mathcal{H}_k$. Using integration by parts, and recalling that $\pi \propto e^{-U}$, it holds that $P_{\mu,k}\nabla_{W_2}\mathcal{F}(\mu) = \mathbb{E}_{x\sim\mu}\left[k(x,\cdot)\nabla U(x) - \nabla_1 k(x,\cdot)\right]$. Thus, in particular,

$$P_{\mu_t^N,k}\nabla_{W_2}\mathcal{F}(\mu_t^N)(x_t^i) = \frac{1}{N}\sum_{j=1}^N [k(x_t^j, x_t^i)\nabla U(x_t^j) - \nabla_1 k(x_t^j, x_t^i)]. \tag{26}$$

Substituting this expression into Alg. 1, we arrive at a learning-rate free analogue of SVGD. This algorithm is summarised in Alg. 2. We note this algorithm is not entirely tuning free, since it requires a choice of bandwidth for the kernel. In practice, however, this parameter can be tuned automatically using the median rule (Liu & Wang, 2016).

4. Numerical Results

In this section, we evaluate the numerical performance of Coin SVGD (Alg. 2). We provide additional results for Coin LAWGD (Alg. 3) and Coin KSDD (Alg. 4) in App. E.1.2 and E.1.3. In all experiments, we implement the adaptive version of Coin SVGD (see App. D). For both Coin SVGD and SVGD, we use the RBF kernel $k(x,x') = \exp(-\frac{1}{h}||x-x'||_2^2)$, with bandwidth chosen using the median heuristic in Liu & Wang (2016). Additional implementation details and results are provided in App. E. Code to reproduce our numerical results can be found at `https://github.com/louissharrock/Coin-SVGD`.

4.1. Toy Examples

We first illustrate the performance of Coin SVGD on a series of toy examples (see App. E.1 for full details). In Fig. 1 (see Sec. 1), we plot the samples generated by Coin SVGD and SVGD after $T = 1000$ iterations, using $N = 20$ particles. In all examples, Coin SVGD qualitatively appears to converge to the correct target distribution.

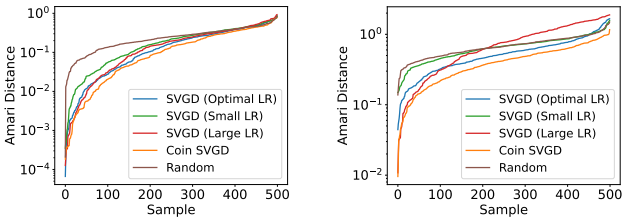
In Fig. 6 - 9 (App. E.1), we provide a more quantitative assessment of our method, plotting the KSD and the energy distance (Székely & Rizzo, 2013) between the targets in Fig. 1, and the approximations obtained by Coin SVGD and SVGD. Our results indicate that the performance of Coin SVGD is competitive with the best performance of SVGD (i.e., the performance of SVGD using the optimal but a priori unknown learning rate) (Fig. 6) and that Coin SVGD often converges more rapidly to the target distribution (Fig. 7). They also confirm, as expected, that Coin SVGD generates increasingly accurate posterior approximations as the number of particles N increases (Fig. 8 and Fig. 9).

4.2. Bayesian Independent Component Analysis

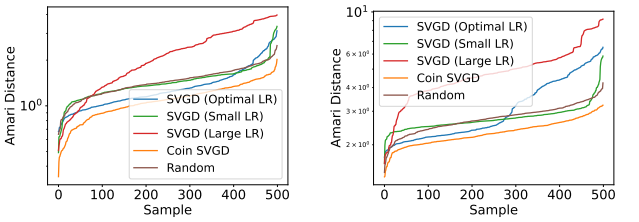
We next consider a Bayesian independent component analysis (ICA) model (e.g., Comon, 1994). Suppose we observe $\boldsymbol{x} = (x_1, \ldots, x_p) \in \mathbb{R}^p$. The task of ICA is to infer the 'unmixing matrix' $\mathbf{W} \in \mathbb{R}^{p\times p}$ such that $\boldsymbol{x} = \mathbf{W}^{-1}\boldsymbol{s}$, where $\boldsymbol{s} = (s_1, \ldots, s_p) \in \mathbb{R}^p$ denote the latent independent sources. We will assume each s_i has the same density: $s_i \sim p_s$. The log-likelihood of this model is then given by $\log p(\boldsymbol{x}|\mathbf{W}) = \log|\mathbf{W}| + \sum_{i=1}^p p_s([\mathbf{W}\boldsymbol{x}]_i)$. For the prior, we assume that the entries of $\mathbf{W}$ are i.i.d., with law $\mathcal{N}(0,1)$. The posterior is then $p(\mathbf{W}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\mathbf{W})p(\mathbf{W})$, with

$$\nabla_{\mathbf{W}} \log p(\mathbf{W}|\boldsymbol{x}) = (\mathbf{W}^{-1})^\top - \frac{p_s'(\mathbf{W}\boldsymbol{x})}{p_s(\mathbf{W}\boldsymbol{x})}\boldsymbol{x}^\top - \mathbf{W}. \tag{27}$$

Following Korba et al. (2021), we choose p_s such that $p_s'(\cdot)/p_s(\cdot) = \tanh(\cdot)$. We are interested in sampling from $p(\mathbf{W}|\boldsymbol{x})$. In our experiments, we generate 1000 samples of $\boldsymbol{x}$ from the ICA model, for $p \in \{2, 4, 8, 16\}$. We use $N = 10$ particles, so that each algorithm returns 10 estimated unmixing matrices $(\hat{\boldsymbol{W}}_i)_{i=1}^{10}$. We then repeat each experiment 50 times, thus obtaining 500 estimates for each method. To assess convergence, we compute the Amari

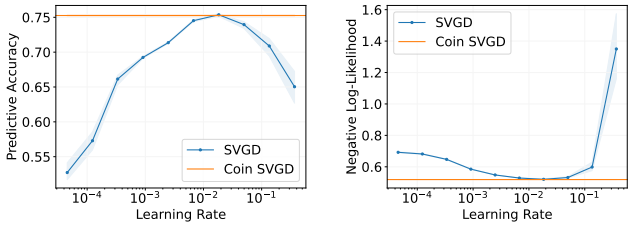


(a) $p = 2$. (b) $p = 4$.

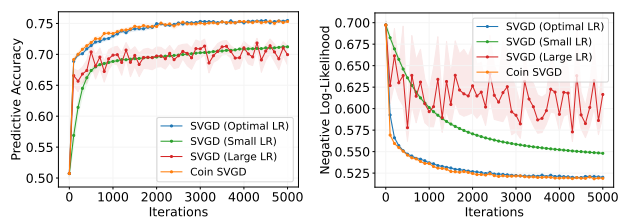


(c) $p = 8$. (d) $p = 16$.

Figure 2. **Results for the Bayesian ICA model**. Amari distances between the true unmixing matrix $\boldsymbol{W}$, and the 500 approximate unmixing matrices output by Coin SVGD and SVGD, for three different values of the learning rate.



(a) Test Accuracy. (b) Negative Log-Likelihood.



(c) Test Accuracy. (d) Negative Log-Likelihood.

Figure 3. **Results for the Bayesian logistic regression model**. (a)-(b). Test accuracy and negative log-likelihood for Coin SVGD and SVGD, as a function of the learning rate. (c)-(d). Test accuracy and negative log-likelihood for Coin SVGD and SVGD (three learning rates) as a function of the number of iterations.

distance (Amari et al., 1995) between the true $\mathbf{W}$ and the estimates generated by each algorithm. This is equal to zero if and only if the two matrices are the same up to scale and permutation. We run SVGD for three learning rates: an 'optimal' rate, which we determine by running SVGD over a range of six candidate values $\gamma \in [1 \times 10^{-5}, 1 \times 10^{0}]$, and selecting the one which returns the lowest average Amari distance, a smaller rate, and a larger rate. We also include the results of a random output, where the estimated matrices have entries which are generated i.i.d. $\mathcal{N}(0, 1)$.

Our results are plotted in Fig. 2. For lower dimensional data, Coin SVGD performs similarly to SVGD with the optimal learning rate (see Fig. 2(a)). In fact, in this case, using a smaller or larger learning rate does not have a significant effect on the performance of SVGD. On the other hand, for higher dimensions, the gap between the two algorithms increases, as does the importance of choosing a good learning rate for SVGD. In particular, for $p \in \{4, 8, 16\}$, Coin SVGD increasingly outperforms SVGD, for any choice of the learning rate. These results perfectly illustrate the robustness of Coin SVGD: our algorithm performs consistently well across these experiments, even as the dimension varies.

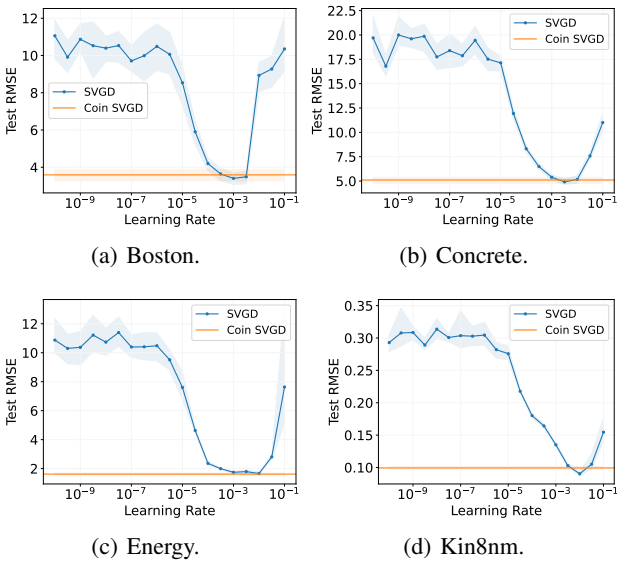
4.3. Bayesian Logistic Regression

We next consider the Bayesian logistic regression model for binary classification, as described in Gershman et al. (2012). Let $\mathcal{D} = (\boldsymbol{x}_i, y_i)_{i=1}^N$ be a dataset with feature vectors $\boldsymbol{x}_i \in \mathbb{R}^p$, and binary labels $y_i \in \{-1, 1\}$. We assume that $p(y_i = 1|\boldsymbol{x}_i, \boldsymbol{w}) = (1 + \exp(-\boldsymbol{w}^T \boldsymbol{x}_i))^{-1}$, for some $\boldsymbol{w} \in \mathbb{R}^p$. We place a Gaussian prior $p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|0, \alpha^{-1})$ on the regression weights $\boldsymbol{w}$, and a Gamma prior $p(\alpha) = \text{Gamma}(\alpha|1, 0.01)$ on $\alpha \in \mathbb{R}_+$. We would like to sample from $p(\boldsymbol{\theta}|\mathcal{D})$, where the parameter of interest is $\boldsymbol{\theta} = [\boldsymbol{w}, \log \alpha]^T \in \mathbb{R}^{p+1}$. We test our algorithm using the Covertype dataset, which consists of 581,012 data points and 54 features. We randomly partition the data into a training dataset (70%), validation dataset (10%), and testing dataset (20%). We run each algorithm with $N = 50$ particles for $T = 5000$ iterations, and compute stochastic gradients using mini-batches of size 100. The results are averaged over 20 random train-test splits.

In Fig. 3(a) - 3(b), we plot the test accuracy and the negative log-likelihood for Coin SVGD, and for SVGD as a function of the step size. Meanwhile, in Fig. 3(c) - 3(d), we plot the test accuracy and the negative log-likelihood against the number of iterations. Once again, we consider three learning rates for SVGD: the optimal learning rate as determined by the results in Fig. 3(a), a smaller learning rate, and a larger learning rate. Similar to before, the performance of Coin SVGD is similar to the best performance of SVGD. On the other hand, when the learning rate is too small or too large, SVGD either converges slowly (green lines, Fig. 3(c) - 3(d)) or is unstable (red lines, Fig. 3(c) - 3(d)).

4.4. Bayesian Neural Network

We next consider a Bayesian neural network model. Our settings are identical to those given in Liu & Wang (2016); see also Hernandez-Lobato & Adams (2015). In particular, we use a two-layer neural network with 50 hidden units with $\text{RELU}(x) = \max(0, x)$ as the activation function. We assume the output is normal, and place a $\text{Gamma}(1, 0.1)$



(a) Boston. (b) Concrete. (c) Energy. (d) Kin8nm.

Figure 4. **Results for the Bayesian neural network**. Average test RMSE for Coin SVGD and SVGD, as a function of the learning rate, after $T = 2000$ iterations, for several UCI datasets.

prior on the inverse covariance. We then assign an isotropic Gaussian prior to the neural network weights. We test the performance of our algorithms on several UCI datasets. The datasets are partitioned into 90% for training and 10% for testing, and our results are averaged over 20 random train-test splits. Finally, we use $N = 20$ particles, and consider a snapshot of the performance after $T = 2000$ iterations.

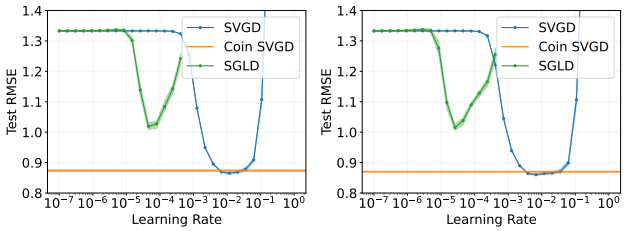
Our results, shown in Fig. 4 (see also Fig. 15 in App. E.4), indicate that SVGD slightly outperforms Coin SVGD for well chosen learning rates, but significantly under-performs Coin SVGD when the learning rate is too small or too large. For certain datasets, the performance of Coin SVGD is close to the optimal performance of SVGD, while for others, there remains a reasonable performance gap. We expect that this could be reduced using recent advancements in parameter-free stochastic optimisation (e.g. Chen et al., 2022a;b).

4.5. Bayesian Probabilistic Matrix Factorisation

Finally, we consider a Bayesian probabilistic matrix factorisation (PMF) model (Salakhutdinov & Mnih, 2008). This model is defined as follows. Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be a matrix of ratings for N users and M movies, where R_{ij} is the rating user i gave to movie j. Define matrices $\mathbf{U}$ and $\mathbf{V}$ for users and movies, respectively, where $\mathbf{U}_i \in \mathbb{R}^d$ and $\mathbf{V}_j \in \mathbb{R}^d$ are d-dimensional latent feature vectors for user i and movie j. The likelihood for the rating matrix is given by

$$p(\mathbf{R}|\mathbf{U}, \mathbf{V}, \alpha) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij}|\mathbf{U}_i^T \mathbf{V}_j, \alpha^{-1})\right]^{I_{ij}}, \quad (28)$$

where I_{ij} denotes an indicator variable which equals 1 if users i gave a rating for movie j. The priors for the users and



(a) $T = 1000$. (b) $T = 2000$.

Figure 5. **Results for the Bayesian probabilistic matrix factorisation model**. Test RMSE for Coin SVGD, SVGD, and SGLD, as a function of the learning rate, after $T \in \{1000, 2000\}$ iterations.

movies are $p(\mathbf{U}|\mu_\mathbf{U}, \Lambda_\mathbf{U}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{U}_i|\mu_\mathbf{U}, \Lambda_\mathbf{U}^{-1})$ and $p(\mathbf{V}|\mu_\mathbf{V}, \Lambda_\mathbf{V}) = \prod_{j=1}^{M} \mathcal{N}(\mathbf{V}_j|\mu_\mathbf{U}, \Lambda_\mathbf{U}^{-1})$, with prior distributions on the hyper-parameters, for $\mathbf{W} = \mathbf{U}$ or $\mathbf{V}$, given by $\mu_\mathbf{W} \sim \mathcal{N}(\mu_\mathbf{W}|\mu_0, \Lambda_\mathbf{W})$ and $\Lambda_\mathbf{W} \sim \Gamma(a_0, b_0)$. The parameters of interest are then $\theta = (\mathbf{U}, \mu_\mathbf{U}, \Lambda_\mathbf{U}, \mathbf{V}, \mu_\mathbf{V}, \Lambda_\mathbf{V})$. In our experiments, we use hyper-parameters $(\alpha, \mu_0, a_0, b_0) = (3, 0, 4, 5)$, and set the latent dimension $d = 20$.

We test our algorithm on the MovieLens dataset (Harper & Konstan, 2015), which consists of 100,000 ratings, taking values {1,2,3,4,5}, for 1,682 movies from 943 users. The data are split into 80% for training and 20% for testing. We use $N = 50$ particles; a batch size of 1000 for stochastic gradients; and average the results over 10 random seeds. Our results are shown in Fig. 5, where we plot the RMSE for SVGD and Coin SVGD, as a function of the learning rate, after $T = 1000$ and $T = 2000$ iterations. We also compare against the stochastic gradient Langevin dynamics (SGLD) algorithm (Welling & Teh, 2011). In this case, Coin SVGD outperforms SVGD for almost all learning rates, and significantly outperforms SGLD.

5. Discussion

In this paper, we introduced a suite of new algorithms for Bayesian inference which are entirely learning-rate free, inspired by coin betting techniques from convex optimisation. In empirical experiments, our coin sampling algorithms - most notably Coin SVGD - demonstrated comparable performance to their learning-rate dependent counterparts, with no need for any hyperparameter tuning.

We highlight several opportunities for future work. In terms of theory, the main open challenge is to establish the convergence of our algorithms under more easily verifiable assumptions. In this paper, we were able to obtain a rather technical condition which was sufficient for convergence. However, it remains unclear how to verify this condition in practice, even for relatively simple target distributions. In terms of methodology, a natural extension of this work is to apply a similar treatment to the many recent variants of SVGD (e.g. Detommaso et al., 2018; Wang et al., 2019; Chen & Ghattas, 2020; Gong et al., 2021).

References

Amari, S., Cichocki, A., and Yang, H. H. A New Learning Algorithm for Blind Signal Separation. In *Proceedings of the 8th International Conference on Neural Information Processing Systems (NIPS 1995)*, pp. 757–763, Denver, CO, 1995. doi: 10.5555/2998828.2998935. 8

Ambrosio, L., Gigli, N., and Giuseppe Savaré. *Gradient Flows: In Metric Spaces and in the Space of Probability Measures*. Birkhäuser, Basel, 2008. ISBN 978-3-7643-8721-1. doi: 10.1007/978-3-7643-8722-8. 1, 4, 5, 6, 15, 23

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, 2003. ISSN 1573-0565. doi: 10.1023/A:1020281327116. 1

Arbel, M., Korba, A., Salim, A., and Gretton, A. Maximum Mean Discrepancy Gradient Flow. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019. 2, 4, 5

Bauschke, H. H. and Combettes, P. L. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, New York, NY, 2011. doi: 10.1007/978-1-4419-9467-7. 19, 22

Brenier, Y. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417, jun 1991. ISSN 0010-3640. doi: 10.1002/cpa.3160440402. 15

Brézis, H. *Opérateurs maximaux monotones et semi-groupes de contractions dans les espaces de Hilbert*. Elsevier Science, Burlington, MA, 1973. 3

Chen, C., Zhang, R., Wang, W., Li, B., and Chen, L. A Unified Particle Optimization Framework for Scalable Bayesian Sampling. In *Uncertainty in Artificial Intelligence*, Monterey, CA, 2018a. 1, 2

Chen, K., Cutkosky, A., and Orabona, F. Implicit Parameter-free Online Learning with Truncated Linear Models. In *Proceedings of the 33rd International Conference on Algorithmic Learning Theory (ALT 2022)*, Paris, France, 2022a. 2, 4, 9, 32

Chen, K., Langford, J., and Orabona, F. Better Parameter-Free Stochastic Optimization with ODE Updates for Coin-Betting. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, Online, 2022b. 9, 32

Chen, P. and Ghattas, O. Projected Stein Variational Gradient Descent. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020. 1, 9

Chen, W. Y., Mackey, L., Gorham, J., Briol, F.-X., and Oates, C. Stein Points. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, Stockholm, Sweden, 2018b. 1

Chen, Y., Georgiou, T. T., and Tannenbaum, A. Optimal Transport for Gaussian Mixture Models. *IEEE Access*, 7:6269–6278, 2019. ISSN 2169-3536 VO - 7. doi: 10.1109/ACCESS.2018.2889838. 23

Cheng, X. and Bartlett, P. Convergence of Langevin MCMC in KL-divergence. *Algorithmic Learning Theory*, pp. 186–211, 2018. 1

Chewi, S., Le Gouic, T., Lu, C., Maunu, T., and Rigollet, P. SVGD as a kernelized Wasserstein gradient flow of the chi-squared divergence. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, pp. 2098–2109, 2020. 2, 4, 5, 6, 24, 26, 29, 30

Chwialkowski, K., Strathmann, H., and Gretton, A. A Kernel Test of Goodness of Fit. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, New York, NY, 2016. 24

Comon, P. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314, 1994. ISSN 0165-1684. doi: 10.1016/0165-1684(94)90029-9. 7

Cutkosky, A. and Orabona, F. Black-Box Reductions for Parameter-free Online Learning in Banach Spaces. In *Proceedings of the 31st Annual Conference on Learning Theory (COLT 2018)*, Stockholm, Sweden, 2018. 2

Dalalyan, A. S. Further and stronger analogy between sampling and optimization: Langevin Monte Carlo and gradient descent. In *Proceedings of the 30th Conference on Learning Theory (COLT 2017)*, Amsterdam, The Netherlands, 2017a. 2

Dalalyan, A. S. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 79(3):651–676, 2017b. ISSN 13697412, 14679868. 2

Dalalyan, A. S. and Karagulyan, A. User-friendly guarantees for the Langevin Monte Carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019. ISSN 0304-4149. doi: 10.1016/j.spa.2019.02.016. 2

De Giorgi, E. New problems on minimizing movements. In *Boundary Value Problems for PDE and Applications*, pp. 81–98. Masson, Paris, 1993. 3

Detommaso, G., Cui, T., Spantini, A., Marzouk, Y., and Scheichl, R. A Stein variational Newton method. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal, Canada, 2018. 1, 9

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159, 2011. 26, 27, 32

Duncan, A., Nüsken, N., and Szpruch, L. On the geometry of Stein variational gradient descent. *Journal of Machine Learning Research*, 24:1–40, 2023. doi: 10.48550/arXiv.1912.00894. 2

Durmus, A. and Moulines, É. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017. doi: 10.1214/16-AAP1238. 2

Durmus, A. and Moulines, É. High-dimensional Bayesian inference via the unadjusted Langevin algorithm. *Bernoulli*, 25(4A):2854–2882, 2019. doi: 10.3150/18-BEJ1073. 2, 6

Durmus, A., Majewski, S., and Miasojedow, B. Analysis of Langevin Monte Carlo via Convex Optimization. *Journal of Machine Learning Research*, 20(1):2666–2711, 2019. 1, 2, 5

Feng, Y., Wang, D., and Liu, Q. Learning to Draw Samples with Amortized Stein Variational Gradient Descent. In *Proceedings of the Conference on Uncertainty In Artificial Intelligence (UAI 2017)*, Sydney, Australia, 2017. 1

Futami, F., Cui, Z., Sato, I., and Sugiyama, M. Frank-Wolfe Stein Sampling. *arXiv preprint*, 2019a. doi: 10.48550/arXiv.1805.07912. 1

Futami, F., Cui, Z., Sato, I., and Sugiyama, M. Bayesian Posterior Approximation via Greedy Particle Optimization. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, HI, 2019b. 1

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. *Bayesian Data Analysis*. Chapman and Hall/CRC, New York, 3rd edition, 2013. doi: 10.1201/b16018. 1

Gershman, S. J., Hoffman, M. D., and Blei, D. M. Nonparametric Variational Inference. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, UK, 2012. 8

Gigli, N. On the inverse implication of Brenier-Mccann theorems and the structure of (P2(M),W2). *Methods and Applications of Analysis*, 18(2), 2011. doi: 10.4310/MAA.2011.v18.n2.a1. 15

Gong, W., Li, Y., and Hernández-Lobato, J. M. Sliced Kernelized Stein Discrepancy. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*, Online, 2021. 9

Gorham, J. and Mackey, L. Measuring Sample Quality with Kernels. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Sydney, Australia, 2017. 24, 27, 28

Güler, O. On the Convergence of the Proximal Point Algorithm for Convex Minimization. *SIAM Journal on Control and Optimization*, 29(2):403–419, mar 1991. ISSN 0363-0129. doi: 10.1137/0329022. 3

Guo, W., Hur, Y., Liang, T., and Ryan, C. T. Online Learning to Transport via the Minimal Selection Principle. In *Proceedings of the 35th Annual Conference on Learning Theory (COLT 2022)*, London, UK, 2022. 5, 6

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Sydney, Australia, 2017. 1

Han, J. and Liu, Q. Stein Variational Gradient Descent Without Gradient. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, Stockholm, Sweden, 2018. 1

Harper, F. M. and Konstan, J. A. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4), 2015. ISSN 2160-6455. doi: 10.1145/2827872. 9

Hartmann, M., Girolami, M., and Klami, A. Lagrangian Manifold Monte Carlo on Monge Patches. In *Proceedings of the 25th International Conference on Aritificial Intelligence and Statistics (AISTATS 2022)*, Online, 2022. 26

Hernandez-Lobato, J. M. and Adams, R. P. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, Lille, France, 2015. 8

Jordan, R., Kinderlehrer, D., and Otto, F. The Variational Formulation of the Fokker–Planck Equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998. doi: 10.1137/S0036141096303359. 1, 2, 5

Jun, K.-S. and Orabona, F. Parameter-Free Online Convex Optimization with Sub-Exponential Noise. In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT 2019)*, Phoenix, AZ, 2019. 2

Kingma, D. P. and Ba, J. Adam: a method for stochastic optimisation. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR '15)*, pp. 1–13, San Diego, CA, 2015. 26

Korba, A., Salim, A., Arbel, M., Luise, G., and Gretton, A. A Non-Asymptotic Analysis for Stein Variational Gradient Descent. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020. 2, 6

Korba, A., Pierre-Cyril, Aubin-Frankowski, Majewski, S., and Ablin, P. Kernel Stein Discrepancy Descent. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, Online, 2021. 2, 4, 5, 7, 24, 26, 30, 31

Krauth, W. *Statistical Mechanics: Algorithms and Computations*. Oxford University Press, 2006. ISBN 9780198515364. 1

Krichevsky, R. E. and Trofimov, V. K. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–207, 1981. doi: 10.1109/TIT.1981.1056331. 4

Lelièvre, T. and Stoltz, G. Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25:681–880, 2016. ISSN 0962-4929. doi: 10.1017/S0962492916000039. 1

Li, L., Liu, Q., Korba, A., Yurochkin, M., and Solomon, J. Sampling with Mollified Interaction Energy Descent. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023. 2

Liu, C. and Zhu, J. Riemannian Stein Variational Gradient Descent for Bayesian Inference. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, LA, 2018. 1

Liu, C., Zhuo, J., Cheng, P., Zhang, R., Zhu, J., and Carin, L. Understanding and Accelerating Particle-Based Variational Inference. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, 2019a. 1, 2

Liu, C., Zhuo, J., and Zhu, J. Understanding MCMC Dynamics as Flows on the Wasserstein Space. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, Long Beach, CA, 2019b. 2

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. ISSN 1436-4646. doi: 10.1007/BF01589116. 24

Liu, J. S. *Monte Carlo Strategies in Scientific Computing*. Springer-Verlag, New York, 2009. ISBN 9780387763699. 1

Liu, Q. Stein variational gradient descent as gradient flow. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, pp. 3118–3126, Red Hook, NY, 2017. ISBN 9781510860964. 1, 2

Liu, Q. and Wang, D. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Proceedings of the 30th Conference on Neural Information Processings Systems (NIPS 2016)*, Barcelona, Spain, 2016. 1, 2, 5, 6, 7, 8, 24, 26, 32

Liu, Q., Lee, J. D., and Jordan, M. A Kernelized Stein Discrepancy for Goodness-of-fit Tests. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, New York, NY, 2016. 24

Liu, X., Zhu, H., Ton, J.-F., Wynne, G., and Duncan, A. Grassmann Stein Variational Gradient Descent. In *Proceedings of the 25th International Conference on Aritificial Intelligence and Statistics (AISTATS 2022)*, Online, 2022. 1

Liu, Y., Ramachandran, P., Liu, Q., and Peng, J. Stein Variational Policy Gradient. In *Proceedings of the Conference on Uncertainty In Artificial Intelligence (UAI 2017)*, Sydney, Australia, 2017. 1, 6

MacKay, D. J. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. ISBN 9780521642989. 1

Neal, R. M. *Bayesian Learning for Neural Networks*. Springer, New York, 1996. doi: 10.1007/978-1-4612-0745-0. 1

Neal, R. M. Slice sampling. *The Annals of Statistics*, 31(3): 705–767, jun 2003. doi: 10.1214/aos/1056562461. 27

Orabona, F. Simultaneous Model Selection and Optimization through Parameter-free Stochastic Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS 2014)*, Montreal, Canada, 2014. 2

Orabona, F. A Modern Introduction to Online Learning. *arXiv preprint*, 2022. doi: 10.48550/arXiv.1912.13213. 16

Orabona, F. and Cutkosky, A. Tutorial on Parameter-Free Online Learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, Online, 2020. 16

Orabona, F. and Pal, D. Coin Betting and Parameter-Free Online Learning. In *Proceedings of the 30th Conference on Neural Information Processings Systems (NIPS 2016)*, Barcelona, Spain, 2016. 2, 3, 4, 5, 6, 16, 17, 19, 22

Orabona, F. and Tommasi, T. Training Deep Networks without Learning Rates Through Coin Betting. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, 2017. 2, 4, 6, 16, 25

Otto, F. The Geometry of Dissipative Evolution Equations: The Porous Medium Equation. *Communications in Partial Differential Equations*, 26(1-2):101–174, 2001. ISSN 0360-5302. doi: 10.1081/PDE-100002243. 2

Otto, F. and Westdickenberg, M. Eulerian Calculus for the Contraction in the Wasserstein Distance. *SIAM Journal on Mathematical Analysis*, 37(4):1227–1255, 2005. doi: 10.1137/050622420. 2

Pagani, F., Wiegand, M., and Nadarajah, S. An n-dimensional Rosenbrock distribution for Markov chain Monte Carlo testing. *Scandinavian Journal of Statistics*, 49(2):657–680, jun 2022. ISSN 0303-6898. doi: 10.1111/sjos.12532. 26

Peypouquet, J. and Sorin, S. Evolution Equations for Maximal Monotone Operators: Asymptotic Analysis in Continuous and Discrete Time. *Journal of Convex Analysis*, 17(3-4):1113–1163, 2010. 3

Pu, Y., Gan, Z., Henao, R., Li, C., Han, S., and Carin, L. VAE Learning via Stein Variational Gradient Descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, 2017. 1

Robert, C. P. and Casella, G. *Monte Carlo Statistical Methods*. Springer-Verlag, New York, 2 edition, 2004. doi: 10.1007/978-1-4757-4145-22. 1

Salakhutdinov, R. and Mnih, A. Bayesian Probabilistic Matrix Factorization Using Markov Chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pp. 880–887, New York, NY, 2008. doi: 10.1145/1390156.1390267. 9

Salim, A., Korba, A., and Luise, G. The Wasserstein Proximal Gradient Algorithm. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020. 2

Salim, A., Sun, L., and Peter Richtárik. A Convergence Theory for SVGD in the Population Limit under Talagrand's Inequality T1. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, Online, 2022. 2, 6

Santambrogio, F. {Euclidean, metric, and Wasserstein} gradient flows: an overview. *Bulletin of Mathematical Sciences*, 7(1):87–154, 2017. ISSN 1664-3615. doi: 10.1007/s13373-017-0101-1. 3, 5

Shi, J. and Mackey, L. A Finite-Particle Convergence Rate for Stein Variational Gradient Descent. *arXiv preprint*, 2022. doi: 10.48550/arXiv.2211.09721. 6

Shor, N. Z. *Minimization Methods for Non-Differentiable Functions*. Springer, Berlin, Heidelberg, 1985. doi: 10.1007/978-3-642-82118-9. 3

Simon-Gabriel, C.-J. *Distribution-Dissimilarities in Machine Learning*. Phd, Eberhard Karls Universität Tübingen, Germany, 2018. 4

Sun, L. and Richtárik, P. Improved Stein Variational Gradient Descent with Importance Weights. *arXiv preprint*, 2022. doi: 10.48550/arXiv.2210.00462. 1, 2

Székely, G. J. and Rizzo, M. L. Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272, 2013. doi: 10.1016/j.jspi.2013.03.018. 7

Tieleman, T. and Hinton, G. E. Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4 (2):26–31, 2012. 26

Villani, C. *Optimal Transport: Old and New*. Springer-Verlag, Berlin, 2008. ISBN 978-3-540-71049-3. doi: 10.1007/978-3-540-71050-9. 4

Wang, D. and Liu, Q. Learning to Draw Samples: With Application to Amortized MLE for Generative Adversarial Learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France, 2017. 1

Wang, D., Tang, Z., Bajaj, C., and Liu, Q. Stein Variational Gradient Descent with Matrix-Valued Kernels. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019. 1, 9

Welling, M. and Teh, Y. W. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, Bellevue, WA, 2011. 9

Wenliang, L. K. and Kanagawa, H. Blindness of score-based methods to isolated components and mixing proportions. *arXiv preprint*, 2021. doi: arXiv.2008.10087. 31

Wibisono, A. Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem. In *Proceedings of the 31st Annual Conference on Learning Theory (COLT 2018)*, Stockholm, Sweden, 2018. 1, 2, 6

Wilson, A. G. and Izmailov, P. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020. 1

Ye, M., Ren, T., and Liu, Q. Stein Self-Repulsive Dynamics: Benefits from Past Samples. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020. 1

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Bayesian Model-Agnostic Meta-Learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal, Canada, 2018. 1

Zhang, R., Chen, C., Li, C., and Carin, L. Policy Optimization as Wasserstein Gradient Flows. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, Stockholm, Sweden, 2018. 1

Zhu, Y. and Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.04.018. 1

Zhuo, J., Liu, C., Shi, J., Zhu, J., Chen, N., and Zhang, B. Message Passing Stein Variational Gradient Descent. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, Stockholm, Sweden, 2018. 1

Zinkevich, M. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pp. 928–935, Washington DC, 2003. 3

A. Background

A.1. Additional Properties of The Wasserstein Space

One important property of the Wasserstein space $(\mathcal{P}_2(\mathbb{R}^d), W_2)$ is that, under appropriate regularity conditions, there exists a unique optimal coupling $\gamma_* \in \Gamma(\mu, \nu)$ which minimises the transport cost $\int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||^p \gamma_*(\mathrm{d}x, \mathrm{d}y)$. This optimal coupling is of the form

$$\gamma = (\mathbf{id} \times \boldsymbol{t}_\mu^\nu)_{\#}\mu, \tag{29}$$

where $\mathbf{id} : \mathbb{R}^d \to \mathbb{R}^d$ is the identity map, and $\boldsymbol{t}_\mu^\nu$ is known as the optimal transport map (Brenier, 1991; Gigli, 2011). It follows that $(\boldsymbol{t}_\mu^\nu)_{\#}\mu = \nu$ and

$$W_2^2(\mu, \nu) = \int_{\mathbb{R}^d} ||x - y||^2 \gamma_*(\mathrm{d}x, \mathrm{d}y) = \int_{\mathbb{R}^d} ||x - \boldsymbol{t}_\mu^\nu(x)||^2 \mathrm{d}x. \tag{30}$$

A.2. Geodesic Convexity

Let $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^d)$. We define a constant speed geodesic between μ and ν as a curve $(\lambda_\eta^{\mu\to\nu})_{\eta\in[0,1]}$ such that $\lambda_0 = \mu$, $\lambda_1 = \nu$, and $W_2(\lambda_\iota, \lambda_\eta) = (\eta - \iota)W_2(\mu, \nu)$ for all $\iota, \eta \in [0, 1]$. If $\boldsymbol{t}_\mu^\nu$ is the optimal transport map between μ and ν, then a constant speed geodesic is given by (e.g., Ambrosio et al., 2008, Sec. 7.2)

$$\lambda_\eta^{\mu\to\nu} = \left((1 - \eta)\mathbf{id} + \eta \boldsymbol{t}_\mu^\nu\right)_{\#} \mu. \tag{31}$$

Let $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to (-\infty, \infty]$. The functional $\mathcal{F}$ is said to be lower semi-continuous if, for all $M \in \mathbb{R}$, $\{\mathcal{F} \leq M\}$ is a closed subset of $\mathcal{P}_2(\mathbb{R}^d)$. For $m \geq 0$, we say that $\mathcal{F}$ is m-geodesically convex if, for any $\mu, \nu \in \mathcal{P}_2(\mathbb{R}^d)$, there exists a constant speed geodesic $(\lambda_\eta^{\mu\to\nu})_{\eta\in[0,1]}$ between μ and ν such that, for all $\eta \in [0, 1]$,

$$\mathcal{F}(\lambda_\eta^{\mu\to\nu}) \leq (1 - \eta)\mathcal{F}(\mu) + \eta\mathcal{F}(\nu) - \frac{m}{2}\eta(1 - \eta)W_2^2(\mu, \nu). \tag{32}$$

In the case that this inequality holds for $m = 0$, we will simply say that $\mathcal{F}$ is geodesically convex.

A.3. Subdifferential Calculus in the Wasserstein Space

Let $\mu \in \mathcal{P}_2(\mathbb{R}^d)$, and let $\xi \in L^2(\mu)$. Let $\mathcal{F}$ be a proper and lower semi-continuous functional on $\mathcal{P}_2(\mathbb{R}^d)$. We say that $\xi \in L^2(\mu)$ belongs to the Fréchet subdifferential of $\mathcal{F}$ at μ, and write $\xi \in \partial\mathcal{F}(\mu)$ if, for any $\nu \in \mathcal{P}_2(\mathbb{R}^d)$,

$$\liminf_{\nu\to\mu} \frac{\mathcal{F}(\nu) - \mathcal{F}(\mu) - \int_{\mathbb{R}^d} \langle \xi(x), \boldsymbol{t}_\mu^\nu(x) - x\rangle \mu(\mathrm{d}x)}{W_2(\nu, \mu)} \geq 0. \tag{33}$$

Suppose, in addition, that $\mathcal{F}$ is m-geodesically convex. Then $\xi \in L^2(\mu)$ belongs to the Fréchet subdifferential $\partial\mathcal{F}(\mu)$ if and only if, for all $\nu \in \mathcal{P}_2(\mathbb{R}^d)$,

$$\mathcal{F}(\nu) - \mathcal{F}(\mu) \geq \int_{\mathbb{R}^d} \langle \xi(x), \boldsymbol{t}_\mu^\nu(x) - x\rangle \mu(\mathrm{d}x) + \frac{m}{2}W_2^2(\mu, \nu). \tag{34}$$

For certain functionals $\mathcal{F}$, and under mild regularity conditions, (see Lemma 10.4.13 in Ambrosio et al., 2008), one has that $\partial\mathcal{F}(\mu) = \{\nabla_{W_2}\mathcal{F}(\mu)\}$, where $\nabla_{W_2}\mathcal{F}(\mu) \in L^2(\mu)$ is given by

$$\nabla_{W_2}\mathcal{F}(\mu) = \nabla\frac{\partial\mathcal{F}(\mu)}{\partial\mu}(x) \quad \text{for } \mu\text{-a.e. } x \in \mathbb{R}^d, \tag{35}$$

and $\frac{\partial\mathcal{F}(\mu)}{\partial\mu} : \mathbb{R}^d \to \mathbb{R}$ denotes the first variation of $\mathcal{F}$ at μ, that is, the unique function such that

$$\lim_{\varepsilon\to 0} \frac{1}{\varepsilon}\left(\mathcal{F}(\mu + \varepsilon\zeta) - \mathcal{F}(\mu)\right) = \int_{\mathbb{R}^d} \frac{\partial\mathcal{F}(\mu)}{\partial\mu}(x)\zeta(\mathrm{d}x), \tag{36}$$

where $\zeta = \nu - \mu$, and $\nu \in \mathcal{P}_2(\mathbb{R}^d)$. We will refer to $\nabla_{W_2}\mathcal{F}(\mu)$ as the Wasserstein gradient of $\mathcal{F}$ at μ.

B. Theoretical Results

B.1. Lemma 2.1

Proof of Lemma 2.1. This result is well known; see, e.g., Lemma 1 in Orabona & Pal (2016); Theorem 9.6 in Orabona (2022); Sec. 4 in Orabona & Tommasi (2017); Part 2 in Orabona & Cutkosky (2020). In particular, we have

$$
\begin{aligned}
f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^*) &\leq \frac{1}{T}\sum_{t=1}^{T}\Big(f(x_t) - f(x^*)\Big) && \text{(Jensen's inequality)}\\
&\leq \frac{1}{T}\left(\sum_{t=1}^{T} c_t x^* - \sum_{t=1}^{T} c_t x_t\right) && \text{(convexity)}\\
&\leq \frac{1}{T}\left(\left(\sum_{t=1}^{T} c_t\right) x^* - h\left(\sum_{t=1}^{T} c_t\right) + \varepsilon\right) && \text{(definition of } h(\cdot)\text{)}\\
&\leq \frac{1}{T}\Big(\max_{v}\left[v x^* - h\left(v\right)\right] + \varepsilon\Big) && \text{(maximum over } v = \textstyle\sum_{t=1}^{T} c_t\text{)}\\
&= \frac{h^*(x^*) + \varepsilon}{T}. && \text{(definition of } h^*(\cdot)\text{)}
\end{aligned}
$$

□

B.2. Proposition 3.3

In this section, we outline how to prove Proposition 3.3. Our proof of this result will rely on a rather strong sufficient condition, which we provide below (Assumption B.1). Before we state this assumption, we will first require some additional notation.

First, throughout this section, we will use $(\mu_t)_{t\in[T]}$ to denote the sequence of betting measures defined by Alg. 1, and $(\varphi_t)_{t\in[T]}$ to denote the transport maps from μ_0 to $(\mu_t)_{t\in[T]}$ defined by Alg. 1. In addition, we will write $(t_{\mu_t}^{\pi})_{t\in[T]}$ for the optimal transport maps from $(\mu_t)_{t\in[T]} \mapsto \pi$, and $(t_{\pi}^{\mu_t})_{t\in[T]}$ for the optimal transport maps from $\pi \mapsto (\mu_t)_{t\in[T]}$. Finally, we let $(\tilde{t}_{\pi,t}^{\mu_t})_{t\in[T]}$ denote the transport maps from π to $(\mu_t)_{t\in[T]}$ defined according to $\tilde{t}_{\pi,t}^{\mu_t} := \varphi_t \circ t_{\pi}^{\mu_0}$, and $(\tilde{t}_{\mu_0,t}^{\pi})_{t\in[T]}$ the transport maps from μ_0 to π defined according to $\tilde{t}_{\mu_0,t}^{\pi} = t_{\mu_t}^{\pi} \circ \varphi_t$. With this notation at hand, we are now ready to introduce the sufficient condition required for our convergence result.

Assumption B.1. Define the functions $v : \mathbb{R}^d \to \mathbb{R}^d$ and $\tilde{v} : \mathbb{R}^d \to \mathbb{R}^d$ according to

$$
v(x) = \sum_{t=1}^{T} -\nabla_{W_2}\mathcal{F}(\mu_t)(t_{\pi}^{\mu_t}(x))\,, \quad \tilde{v}(x) = \sum_{t=1}^{T} -\nabla_{W_2}\mathcal{F}(\mu_t)(\tilde{t}_{\pi,t}^{\mu_t}(x)). \tag{37}
$$

Then there exists a constant $K > 0$ such that, for all $x \in \mathbb{R}^d$,

$$
\frac{1}{2L^2T}\left[||v(x)||^2 - ||\tilde{v}(x)||^2\right] \leq \ln K. \tag{38}
$$

We can now proceed to the proof Proposition 3.3. For convenience, we first recall the original statement of this result.

Proposition 3.3. *Let Assumptions 3.1 - 3.2 and Assumption B.1 hold. Then*

$$
\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\Bigg[w_0 + \int_{\mathbb{R}^d} ||x|| \sqrt{T\ln\left(1 + \frac{96K^2T^2\,||x||^2}{w_0^2}\right)}\pi(\mathrm{d}x) \tag{39}
$$

$$
+ \int_{\mathbb{R}^d} ||x|| \sqrt{T\ln\left(1 + \frac{96T^2\,||x||^2}{w_0^2}\right)}\mu_0(\mathrm{d}x)\Bigg], \tag{40}
$$

where $K > 0$ is the constant defined in Assumption B.1.

Proof of Proposition 3.3. Our proof begins in much the same fashion as the proof of Lemma 2.1. On this occasion, we consider

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{1}{T}\sum_{t=1}^{T}\mathcal{F}\left(\mu_t\right) - \mathcal{F}(\pi) \tag{41}$$

$$\leq \frac{1}{T}\sum_{t=1}^{T}\int_{\mathbb{R}^d}\langle -\nabla_{W_2}\mathcal{F}(\mu_t)(x), t_{\mu_t}^{\pi}(x) - x\rangle\mu_t(\mathrm{d}x) \tag{42}$$

$$\leq \frac{L}{T}\sum_{t=1}^{T}\left[\int_{\mathbb{R}^d}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(t_{\pi}^{\mu_t}(x)), x\right\rangle\pi(\mathrm{d}x) - \int_{\mathbb{R}^d}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), \varphi_t(x)\right\rangle\mu_0(\mathrm{d}x)\right] \tag{43}$$

$$= \frac{L}{T}\Bigg[\int_{\mathbb{R}^d}\left\langle\sum_{t=1}^{T}-\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(t_{\pi}^{\mu_t}(x)), x\right\rangle\pi(\mathrm{d}x) - \int_{\mathbb{R}^d}\sum_{t=1}^{T}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), x\right\rangle\mu_0(\mathrm{d}x)$$
$$-\int_{\mathbb{R}^d}\sum_{t=1}^{T}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), \varphi_t(x)\mu_0(\mathrm{d}x) + \int_{\mathbb{R}^d}\sum_{t=1}^{T}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), x\right\rangle\mu_0(\mathrm{d}x)\right] \tag{44}$$

$$= \frac{L}{T}\Bigg[\int_{\mathbb{R}^d}\left\langle\sum_{t=1}^{T}-\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(t_{\pi}^{\mu_t}(x)), x\right\rangle - \left\langle\sum_{t=1}^{T}-\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\tilde{t}_{\pi,t}^{\mu_t}(x)), t_{\pi}^{\mu_0}(x)\right\rangle\pi(\mathrm{d}x)$$
$$-\int_{\mathbb{R}^d}\sum_{t=1}^{T}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), \varphi_t(x) - x\right\rangle\mu_0(\mathrm{d}x)\Bigg], \tag{45}$$

where in (41) we have used Jensen's inequality, in (42) we have used the definition of geodesic convexity (see App. A), in (43) we have substituted $x \mapsto t_{\pi}^{\mu_t}(x)$ and $x \mapsto \varphi_t(x)$ in the first and second integrals, respectively, used the fact that, by definition, $(t_{\pi}^{\mu_t})_{\#}\pi = \mu_t$ and $(\varphi_t)_{\#}\mu_0 = \mu_t$ (see Alg. 1), and introduced the notation $\hat{\mathcal{F}} = \frac{1}{L}\mathcal{F}$; in (44) we have added and subtracted the same integral; and in (45) we have substituted $x \mapsto t_{\pi}^{\mu_0}(x)$, used the fact that $\tilde{t}_{\pi,t}^{\mu_t}(x) = \varphi_t(t_{\pi}^{\mu_0}(x))$, and the fact that $(t_{\pi}^{\mu_0})_{\#}\pi = \mu_0$ in the second integral; and combined the third and fourth integrals.

By construction, the betting strategy in Alg. 1 guarantees that, for any $x \in \mathbb{R}^d$, and any arbitrary sequence $c_1(x), \ldots, c_T(x) \in \mathbb{R}^d$, such that $||c_t(x)|| \leq 1$, there exists an even, logarithmically convex function $h : \mathbb{R} \to \mathbb{R}_+$ such that the wealth is lower bounded as (Orabona & Pal, 2016, Proof of Theorem 3, App. C)

$$w_T(x) = w_0 + \sum_{t=1}^{T}\langle c_t(x), \varphi_t(x) - x\rangle \geq h\left(\left|\left|\sum_{t=1}^{T}c_t(x)\right|\right|\right). \tag{46}$$

In particular, the betting strategy in Alg. 1 guarantees that this inequality holds with (Orabona & Pal, 2016, App. F.1, Proof of Corollary 5)

$$h\left(u\right) = w_0\frac{2^T\Gamma(1)\Gamma(\frac{T+1}{2} + \frac{u}{2})\cdot\Gamma(\frac{T+1}{2} - \frac{u}{2})}{\Gamma^2(\frac{1}{2})\Gamma(T+1)}. \tag{47}$$

Due to Lemma 16 in Orabona & Pal (2016), we also have that

$$h(u) \geq i(u) := \frac{w_0}{K_1\sqrt{T}}\exp\left(\frac{u^2}{2T}\right), \tag{48}$$

where $K_1 = e\sqrt{\pi}$ is a universal constant. We will apply the inequality in (46) for the sequence $c_t(x) = -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))$. In particular, substituting this sequence into (46), and using also the inequality in (48), we have that

$$w_0 + \sum_{t=1}^{T}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), \varphi_t(x) - x\right\rangle \geq i\left(\left|\left|\sum_{t=1}^{T}-\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))\right|\right|\right). \tag{49}$$

Suppose now that we define the function $I : \mathbb{R}^d \to (-\infty, \infty]$ according to $I(u) = i(||u||)$. Thus, in particular,

$$I(u) = \frac{w_0}{K_1\sqrt{T}} \exp\left(\frac{||u||^2}{2T}\right). \tag{50}$$

By using this definition in (49), and then substituting (49) into (41) - (45), we then have

$$\begin{aligned}\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\Bigg[\sum_{t=1}^{T}\int_{\mathbb{R}^d}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(t_\pi^{\mu_t}(x)), x\right\rangle - \left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\tilde{t}_{\pi,t}^{\mu_t}(x)), t_\pi^{\mu_0}(x)\right\rangle \pi(\mathrm{d}x) \\ - \int_{\mathbb{R}^d} I\left(\sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))\right)\mu_0(\mathrm{d}x) + w_0\Bigg] \qquad (51)\\ = \frac{L}{T}\Bigg[\int_{\mathbb{R}^d}\left\langle \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(t_\pi^{\mu_t}(x)), x\right\rangle - \left\langle \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\tilde{t}_{\pi,t}^{\mu_t}(x)), t_\pi^{\mu_0}(x)\right\rangle \pi(\mathrm{d}x) \\ - \int_{\mathbb{R}^d} I\left(\sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\tilde{t}_{\pi,t}^{\mu_t}(x))\right)\pi(\mathrm{d}x) + w_0\Bigg], \qquad (52)\end{aligned}$$

where, in (52), we have substituted $x \mapsto t_\pi^{\mu_0}(x)$ in the second integral, used the fact that $\tilde{t}_{\pi,t}^{\mu_t}(x) = \varphi_t(t_\pi^{\mu_0}(x))$, and finally the fact that $(t_\pi^{\mu_0})_{\#}\pi = \mu_0$. Suppose we also now define

$$u(x) = \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(t_\pi^{\mu_t}(x)) \quad , \quad \tilde{u}(x) = \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\tilde{t}_{\pi,t}^{\mu_t}(x)) \quad , \quad A = \frac{\int_{\mathbb{R}^d} I\left(\tilde{u}(x)\right)\pi(\mathrm{d}x)}{\int_{\mathbb{R}^d} I\left(u(x)\right))\pi(\mathrm{d}x)}. \tag{53}$$

Using this notation, and re-ordering terms, we can now rewrite the previous inequality as

$$\begin{aligned}\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) &\leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}\langle u(x), x\rangle\,\pi(\mathrm{d}x) + \int_{\mathbb{R}^d}\langle \tilde{u}(x), -t_\pi^{\mu_0}(x)\rangle\,\pi(\mathrm{d}x) - \int_{\mathbb{R}^d} I\left(\tilde{u}(x)\right)\pi(\mathrm{d}x)\right] \qquad (54)\\ &\leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}\langle u(x), x\rangle - \frac{1}{2}I(\tilde{u}(x))\pi(\mathrm{d}x) + \int_{\mathbb{R}^d}\langle \tilde{u}(x), -t_\pi^{\mu_0}(x)\rangle - \frac{1}{2}I(\tilde{u}(x))\pi(\mathrm{d}x)\right] \qquad (55)\\ &= \frac{L}{T}\Bigg[w_0 + \int_{\mathbb{R}^d} A\left(\langle u(x), \frac{x}{A}\rangle - \frac{1}{2}I(u(x))\right)\pi(\mathrm{d}x) \\ &\quad + \int_{\mathbb{R}^d}\left(\langle \tilde{u}(x), -t_\pi^{\mu_0}(x)\rangle - \frac{1}{2}I(\tilde{u}(x))\right)\pi(\mathrm{d}x)\Bigg]. \qquad (56)\end{aligned}$$

Suppose that we now fix $x \in \mathbb{R}^d$, and write $\theta = u(x)$, $\eta = \tilde{u}(x)$, $F = \frac{1}{2}I$, $x^* = \frac{x}{A}$, and $x^\dagger = -t_\pi^{\mu_0}(x)$. Using this notation, for each $x \in \mathbb{R}^d$, we can now rewrite the first and second integrands in (56) as

$$\begin{aligned}\langle u(x), \frac{x}{A}\rangle - \frac{1}{2}I(u(x)) &:= \langle \theta, x^*\rangle - F(\theta),\\ \langle \tilde{u}(x), -t_\pi^{\mu_0}(x)\rangle - \frac{1}{2}I(\tilde{u}(x)) &:= \langle \eta, x^\dagger\rangle - F(\eta).\end{aligned}$$

Taking the supremum over $\theta \in \mathbb{R}^d$ and $\eta \in \mathbb{R}^d$, respectively, and using the definition of the convex conjugate, we can easily upper bound these expressions by

$$\langle \theta, x^*\rangle - F(\theta) \leq \sup_{\theta\in\mathbb{R}^d}\left(\langle \theta, x^*\rangle - F(\theta)\right) \leq F^*(x^*), \tag{57}$$

$$\langle \eta, x^\dagger\rangle - F(\eta) \leq \sup_{\eta\in\mathbb{R}^d}\left(\langle \eta, x^\dagger\rangle - F(\eta)\right) \leq F^*(x^\dagger), \tag{58}$$

where, as elsewhere, F^* denotes the Fenchel conjugate of F. Returning to our previous notation, and using the fact that $x \in \mathbb{R}^d$ was chosen arbitrarily, we thus have that

$$\langle u(x), \frac{x}{A}\rangle - \frac{1}{2}I(u(x)) \leq (\frac{1}{2}I)^*\left(\frac{x}{A}\right) = \frac{1}{2}I^*\left(\frac{2x}{A}\right), \tag{59}$$

$$\langle \tilde{u}(x), -t_\pi^{\mu_0}(x)\rangle - \frac{1}{2}I(\tilde{u}(x)) \leq (\frac{1}{2}I)^*\left(-t_\pi^{\mu_0}(x)\right) = \frac{1}{2}I^*\left(-2t_\pi^{\mu_0}(x)\right), \tag{60}$$

for all $x \in \mathbb{R}^d$, where in both lines we have used the fact that $(af)^*(x^*) = af^*(\frac{x^*}{a})$, with $a = \frac{1}{2}$. Substituting (59) and (60) into (56), it now follows straightforwardly that

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}\frac{1}{2}AI^*\left(\frac{2x}{A}\right)\pi(\mathrm{d}x) + \int_{\mathbb{R}^d}\frac{1}{2}I^*\left(-2t_\pi^{\mu_0}(x)\right)\pi(\mathrm{d}x)\right] \tag{61}$$

$$= \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}\frac{1}{2}AI^*\left(\frac{2t_{\mu_0}^{\pi}(x)}{A}\right)\mu_0(\mathrm{d}x) + \int_{\mathbb{R}^d}\frac{1}{2}I^*\left(-\frac{1}{2}x\right)\mu_0(\mathrm{d}x)\right] \tag{62}$$

$$= \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}\frac{1}{2}Ai^*\left(\frac{2\left|\left|t_{\mu_0}^{\pi}(x)\right|\right|}{A}\right)\mu_0(\mathrm{d}x) + \int_{\mathbb{R}^d}\frac{1}{2}i^*\left(2\left|\left|x\right|\right|\right)\mu_0(\mathrm{d}x)\right], \tag{63}$$

where in the second line we have substituted $x \mapsto t_{\mu_0}^{\pi}(x)$ into both integrals, and used the fact that $(t_{\mu_0}^{\pi})_{\#}\mu_0 = \pi$; and, in the final line, we have used the fact that the Fenchel conjugate of $i(||\cdot||)$ is $i^*(||\cdot||)$ since i^* is an even function (Bauschke & Combettes, 2011, Example 13.7). Now, by Lemma 18 in Orabona & Pal (2016), we have the upper bound

$$i^*(u) \leq |u|\sqrt{T\ln\left(1 + \frac{24T^2u^2}{w_0^2}\right)} - \frac{w_0}{K_1\sqrt{T}}. \tag{64}$$

Substituting this into our previous bound (63), we finally arrive at

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0\left(1 - \frac{A}{2K_1\sqrt{T}}\right) + \int_{\mathbb{R}^d}\frac{1}{2}A\frac{2||t_{\mu_0}^{\pi}(x)||}{A}\sqrt{T\ln\left(1 + \frac{24T^2||2t_{\mu_0}^{\pi}(x)||^2}{A^2w_0^2}\right)}\mu_0(\mathrm{d}x)\right.$$
$$\left. + \int_{\mathbb{R}^d}\frac{1}{2}2||x||\sqrt{T\ln\left(1 + \frac{24T^2||2x||^2}{A^2w_0^2}\right)}\mu_0(\mathrm{d}x)\right] \tag{65}$$

$$\leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}||x||\sqrt{T\ln\left(1 + \frac{96T^2||x||^2}{A^2w_0^2}\right)}\pi(\mathrm{d}x)\right.$$
$$\left. + \int_{\mathbb{R}^d}||x||\sqrt{T\ln\left(1 + \frac{96T^2||x||^2}{w_0^2}\right)}\mu_0(\mathrm{d}x)\right], \tag{66}$$

where, in the second line, we have substituted $x \mapsto t_{\pi}^{\mu_0}(x)$ in the first integral, and used the fact that $(t_{\pi}^{\mu_0})_{\#}\pi = \mu_0$. It remains to bound the constant A from below, or, equivalently, the constant A^{-1} from above. The required bound will follow directly from our sufficient condition (Assumption B.1). Indeed, from Assumption B.1, we have that

$$\frac{1}{2L^2T}\left[||v(x)||^2 - ||\tilde{v}(x)||^2\right] \leq \ln K \implies \exp\left[\frac{||v(x)||^2}{2L^2T}\right] \leq K\exp\left[\frac{||\tilde{v}(x)||^2}{2L^2T}\right]. \tag{67}$$

Using this bound, and the definition of A in (53), it then follows straightforwardly that

$$A^{-1} = \frac{\displaystyle\int_{\mathbb{R}^d}\frac{w_0}{K_1\sqrt{T}}\exp\left[\frac{||u(x)||^2}{2T}\right]\pi(\mathrm{d}x)}{\displaystyle\int_{\mathbb{R}^d}\frac{w_0}{K_1\sqrt{T}}\exp\left[\frac{||\tilde{u}(x)||^2}{2T}\right]\pi(\mathrm{d}x)} = \frac{\displaystyle\int_{\mathbb{R}^d}\exp\left[\frac{||v(x)||^2}{2L^2T}\right]\pi(\mathrm{d}x)}{\displaystyle\int_{\mathbb{R}^d}\exp\left[\frac{||\tilde{v}(x)||^2}{2L^2T}\right]\pi(\mathrm{d}x)} \leq \frac{\displaystyle\int_{\mathbb{R}^d}K\exp\left[\frac{||\tilde{v}(x)||^2}{2L^2T}\right]\pi(\mathrm{d}x)}{\displaystyle\int_{\mathbb{R}^d}\exp\left[\frac{||\tilde{v}(x)||^2}{2L^2T}\right]\pi(\mathrm{d}x)} = K. \tag{68}$$

Finally, substituting the bound in (68) into (66), we arrive at

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}||x||\sqrt{T\ln\left(1 + \frac{96K^2T^2\,||x||^2}{w_0^2}\right)}\pi(\mathrm{d}x)\right.$$
$$\left. + \int_{\mathbb{R}^d}||x||\sqrt{T\ln\left(1 + \frac{96T^2\,||x||^2}{w_0^2}\right)}\mu_0(\mathrm{d}x)\right]. \tag{69}$$

□

B.3. An Alternative to Proposition 3.3

In this section, we outline how to obtain an alternative bound to the one given in Proposition 3.3, under a slightly different sufficient condition.

Assumption B.1'. *There exists a transport map $T^{\pi}_{\mu_0} : \mathbb{R}^d \to \mathbb{R}^d$ from μ_0 to π, which does not depend on $t \in [T]$, such that, for some $K > 0$,*

$$\sum_{t=1}^{T} \left|\left| t^{\pi}_{\mu_t} \circ \varphi_t - T^{\pi}_{\mu_0} \right|\right|_{L^2(\mu_0)} \leq K\sqrt{T} \ln\left[\text{poly}(T)\right]. \tag{70}$$

Remark. It is, at present, unclear how to establish the existence of a transport map $T^{\pi}_{\mu_0}$ which satisfies (70), aside from in one very simple case (see Sec. B.4). In the general case, one possible candidate for $T^{\pi}_{\mu_0}$ is the optimal transport map $t^{\pi}_{\mu_0}$ from μ_0 to π, so that $T^{\pi}_{\mu_0}(x) = t^{\pi}_{\mu_0}(x)$. In this case, (70) reads

$$\sum_{t=1}^{T} ||t^{\pi}_{\mu_t} \circ \varphi_t - t^{\pi}_{\mu_0}||_{L^2(\mu_0)} \leq K\sqrt{T} \ln\left[\text{poly}(T)\right], \tag{71}$$

which can be interpreted as a bound on the sum of the distances between the optimal transport map $t^{\pi}_{\mu_0}$ from μ_0 to π, and the maps $(t^{\pi}_{\mu_t} \circ \varphi_t)_{t\in[T]}$ which first transport μ_0 to $(\mu_t)_{t\in[T]}$ according to the transport maps $(\varphi_t)_{t\in[T]}$ defined by Alg. 1, and then map $(\mu_t)_{t\in[T]}$ to π via the optimal transport maps $(t^{\pi}_{\mu_t})_{t\in[T]}$.

Another possible candidate for $T^{\pi}_{\mu_0}$ is the average, over all iterations, of the composition of the optimal transport maps $(t^{\pi}_{\mu_t})_{t\in[T]}$ from $(\mu_t)_{t\in[T]}$ to π, and the maps $(\varphi_t)_{t\in[T]}$ from μ_0 to $(\mu_t)_{t\in[T]}$ defined by Alg. 1, viz $T^{\pi}_{\mu_0}(x) = \frac{1}{T}\sum_{t=1}^{T} t^{\pi}_{\mu_t}(\varphi_t(x))$. In this case, (70) becomes

$$\sum_{t=1}^{T} ||t^{\pi}_{\mu_t} \circ \varphi_t - \frac{1}{T}\sum_{s=1}^{T} t^{\pi}_{\mu_s} \circ \varphi_s||_{L^2(\mu_0)} \leq K\sqrt{T} \ln\left[\text{poly}(T)\right]. \tag{72}$$

Proposition 3.3'. *Let Assumptions 3.1 - 3.2 and Assumption B.1' hold. Then*

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \tag{73}$$
$$\leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d} \left|\left|T^{\pi}_{\mu_0}(x) - x\right|\right| \sqrt{T \ln\left(1 + \frac{24T^2 \left|\left|T^{\pi}_{\mu_0}(x) - x\right|\right|^2}{w_0^2}\right)} \mu_0(\mathrm{d}x) + K\sqrt{T}\ln\left[\text{poly}(T)\right]\right].$$

where $K > 0$ and $T^{\pi}_{\mu_0} : \mathbb{R}^d \to \mathbb{R}^d$ are defined in Assumption B.1'.

Proof of Proposition 3.3'. Once again, our proof begins in a similar fashion to the proof of Lemma 2.1. In this case, we now have

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{1}{T}\sum_{t=1}^{T}\mathcal{F}\left(\mu_t\right) - \mathcal{F}(\pi) \tag{74}$$
$$\leq \frac{1}{T}\sum_{t=1}^{T}\int_{\mathbb{R}^d} \langle -\nabla_{W_2}\mathcal{F}(\mu_t)(x), t^{\pi}_{\mu_t}(x) - x\rangle \mu_t(\mathrm{d}x) \tag{75}$$
$$= \frac{L}{T}\sum_{t=1}^{T}\int_{\mathbb{R}^d} \langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), t^{\pi}_{\mu_t}(\varphi_t(x)) - \varphi_t(x)\rangle \mu_0(\mathrm{d}x) \tag{76}$$
$$= \frac{L}{T}\Bigg[\sum_{t=1}^{T}\int_{\mathbb{R}^d} \left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), t^{\pi}_{\mu_t}(\varphi_t(x)) - x\right\rangle \mu_0(\mathrm{d}x)$$
$$- \int_{\mathbb{R}^d}\sum_{t=1}^{T}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), \varphi_t(x) - x\right\rangle \mu_0(\mathrm{d}x)\Bigg], \tag{77}$$

where, in (74) we have used Jensen's inequality, in (75) we have used the definition of geodesic convexity (see App. A), in (76) we have substituted $x \mapsto \varphi_t(x)$, and used the fact that, by definition, $(\varphi_t)_{\#}\mu_0 = \mu_t$, and in (77) we have added and subtracted the same integral. Following the same argument as in (46) - (51), it then follows that

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[\sum_{t=1}^{T}\int_{\mathbb{R}^d}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), t^{\pi}_{\mu_t}(\varphi_t(x)) - x\right\rangle \mu_0(\mathrm{d}x) \right.$$
$$\left. - \int_{\mathbb{R}^d} I\left(\sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))\right)\mu_0(\mathrm{d}x) + w_0\right], \tag{78}$$

where $I : \mathbb{R}^d \to (-\infty, \infty]$ is the function defined in the proof of Proposition 3.3, c.f. (50). To proceed, we will now write

$$t^{\pi}_{\mu_t}(\varphi_t(x)) = T^{\pi}_{\mu_0}(x) + \underbrace{t^{\pi}_{\mu_t}(\varphi_t(x)) - T^{\pi}_{\mu_0}(x)}_{S_t(x)}. \tag{79}$$

Based on this decomposition, we can rewrite the first term in (78) as

$$\frac{L}{T}\sum_{t=1}^{T}\int_{\mathbb{R}^d}\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), t^{\pi}_{\mu_t}(\varphi_t(x)) - x\right\rangle \mu_0(\mathrm{d}x)$$
$$= \frac{L}{T}\left[\int_{\mathbb{R}^d}\left\langle \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), T^{\pi}_{\mu_0}(x) - x\right\rangle \mu_0(\mathrm{d}x) + \sum_{t=1}^{T}\int_{\mathbb{R}^d}\underbrace{\left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), S_t(x)\right\rangle}_{R_t(x)}\mu_0(\mathrm{d}x)\right]. \tag{80}$$

By substituting (80), the previous inequality (78) can now be written as

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[\int_{\mathbb{R}^d}\left\langle \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), T^{\pi}_{\mu_0}(x) - x\right\rangle \mu_0(\mathrm{d}x) + \sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x)\right.$$
$$\left. - \int_{\mathbb{R}^d} I\left(\sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))\right)\mu_0(\mathrm{d}x) + w_0\right] \tag{81}$$

$$= \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d}\left[\langle z(x), T^{\pi}_{\mu_0}(x) - x\rangle - I(z(x))\right]\mu_0(\mathrm{d}x) + \sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x)\right], \tag{82}$$

where in the final line we have introduced the notation $z(x) = \sum_{t=1}^{T} -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))$. Suppose we now fix $x \in \mathbb{R}^d$, and write $\theta = z(x)$ and $x^* = T^{\pi}_{\mu_0}(x) - x$. Using this notation, we can now rewrite the first integrand in the previous expression as

$$\langle z(x), T^{\pi}_{\mu_0}(x) - x\rangle - I(z(x)) := \langle \theta, x^*\rangle - I(\theta), \tag{83}$$

for each fixed $x \in \mathbb{R}^d$. Taking the supremum over $\theta \in \mathbb{R}^d$ and using the definition of the convex conjugate, we can easily upper bound this expression by

$$\langle \theta, x^*\rangle - I(\theta) \leq \sup_{\theta \in \mathbb{R}^d}\left(\langle \theta, x^*\rangle - I(\theta)\right) \leq I^*(x^*). \tag{84}$$

Returning to our previous notation, and using the fact $x \in \mathbb{R}^d$ was chosen arbitrarily, we thus have

$$\langle z(x), T^{\pi}_{\mu_0}(x) - x\rangle - I(z(x)) \leq I^*(T^{\pi}_{\mu_0}(x) - x) \tag{85}$$

for all $x \in \mathbb{R}^d$. Substituting this upper bound into (81) - (82), it then follows straightforwardly that

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d} I^*(T^{\pi}_{\mu_0}(x) - x)\mu_0(\mathrm{d}x) + \sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x)\right] \tag{86}$$

$$= \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d} i^*(\left|\left|T^{\pi}_{\mu_0}(x) - x\right|\right|)\mu_0(\mathrm{d}x) + \sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x)\right], \tag{87}$$

where, in the second line, we have used the fact that the Fenchel conjugate of $i(||\cdot||)$ is $i^*(||\cdot||)$ since i^* is an even function (Bauschke & Combettes, 2011, Example 13.7). Similar to before, Lemma 18 of Orabona & Pal (2016) allows to bound this Fenchel conjugate as

$$i^*(u) \leq |u| \sqrt{T \ln\left(1 + \frac{24T^2u^2}{w_0^2}\right)} - \frac{w_0}{K_1\sqrt{T}}. \tag{88}$$

Substituting this into the previous bound in (87), we then have that

$$\begin{aligned}
\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) &\leq \frac{L}{T}\Bigg[w_0\left(1 - \frac{1}{K_1\sqrt{T}}\right) + \int_{\mathbb{R}^d} \left|\left|T^\pi_{\mu_0}(x) - x\right|\right| \sqrt{T \ln\left(1 + \frac{24T^2\left|\left|T^\pi_{\mu_0}(x) - x\right|\right|^2}{w_0^2}\right)}\mu_0(\mathrm{d}x) \\
&\qquad + \sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x)\Bigg] \\
&\leq \frac{L}{T}\Bigg[w_0 + \int_{\mathbb{R}^d} \left|\left|T^\pi_{\mu_0}(x) - x\right|\right| \sqrt{T \ln\left(1 + \frac{24T^2\left|\left|T^\pi_{\mu_0}(x) - x\right|\right|^2}{w_0^2}\right)}\mu_0(\mathrm{d}x) \\
&\qquad + \sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x)\Bigg].
\end{aligned} \tag{89}$$

It remains to deal with the final term. Unsurprisingly, the bound on this term will follow directly from our alternative sufficient condition (Assumption B.1'). In particular, recalling the definition of R_t from (80), we have

$$\begin{aligned}
\sum_{t=1}^{T}\int_{\mathbb{R}^d} R_t(x)\mu_0(\mathrm{d}x) &= \sum_{t=1}^{T}\int_{\mathbb{R}^d} \left\langle -\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x)), S_t(x)\right\rangle \mu_0(\mathrm{d}x) && (90)\\
&\leq \sum_{t=1}^{T}\left[\int_{\mathbb{R}^d} ||\nabla_{W_2}\hat{\mathcal{F}}(\mu_t)(\varphi_t(x))||^2\mu_0(\mathrm{d}x)\right]^{\frac{1}{2}}\left[\int_{\mathbb{R}^d} ||S_t(x)||^2\mu_0(\mathrm{d}x)\right]^{\frac{1}{2}} && (91)\\
&\leq \sum_{t=1}^{T}\left[\int_{\mathbb{R}^d} ||S_t(x)||^2\mu_0(\mathrm{d}x)\right]^{\frac{1}{2}} && (92)\\
&= \sum_{t=1}^{T}\left[\int_{\mathbb{R}^d} ||t^\pi_{\mu_t}(\varphi_t(x)) - T^\pi_{\mu_0}(x)||^2\mu_0(\mathrm{d}x)\right]^{\frac{1}{2}} \leq K\sqrt{T}\ln\left[\mathrm{poly}(T)\right], && (93)
\end{aligned}$$

where in (91) we have used the Cauchy-Schwarz inequality, in (92) we have used the assumed bound on $||\nabla_{W_2}\mathcal{F}(\mu_t)(\varphi_t(x))||$ (Assumption 3.2), and in (93) we have substituted the definition of S_t from (79), and used Assumption B.1'. Finally, substituting (93) into (89), we arrive at

$$\begin{aligned}
&\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) && (94)\\
&\leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d} \left|\left|T^\pi_{\mu_0}(x) - x\right|\right| \sqrt{T \ln\left(1 + \frac{24T^2\left|\left|T^\pi_{\mu_0}(x) - x\right|\right|^2}{w_0^2}\right)}\mu_0(\mathrm{d}x) + K\sqrt{T}\ln\left[\mathrm{poly}(T)\right]\right].
\end{aligned}$$

□

B.4. A Simple Gaussian Case

In this section, we establish a stronger version of Proposition 3.3' in a very simple Gaussian setting. In particular, we consider the case in which the initial distribution is Gaussian, the target distribution is Gaussian, and these two distributions have the same covariance. In this case, not only can we tighten the bound in Proposition 3.3', but we no longer require any additional technical assumptions (Assumption B.1 or B.1').

Proposition 3.4. *Let $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$. Let $\mu_0 \sim \mathcal{N}(m_0, \Sigma)$ and $\pi \sim \mathcal{N}(m_\pi, \Sigma)$. Suppose that Assumption 3.2 holds. Then*

$$\mathcal{F}\left(\frac{1}{T}\sum_{t=1}^{T}\mu_t\right) - \mathcal{F}(\pi) \leq \frac{L}{T}\left[w_0 + \int_{\mathbb{R}^d} \left|\left|t^\pi_{\mu_0}(x) - x\right|\right| \sqrt{T \ln\left(1 + \frac{24T^2\left|\left|t^\pi_{\mu_0}(x) - x\right|\right|^2}{w_0^2}\right)}\mu_0(\mathrm{d}x)\right]. \tag{95}$$

Proof of Proposition 3.4. We will establish this result as a special case of Proposition 3.3'. We must therefore begin by verifying the additional assumptions required by Proposition 3.3', namely, Assumption 3.1 and Assumption B.1'. We begin with Assumption 3.1. Given $\pi \sim \mathcal{N}(m_\pi, \sigma^2)$, it follows from standard results that the functional $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$ is proper, lower semi-continuous, and geodesically convex (e.g., Ambrosio et al., 2008, Sec. 9.4). Thus, this assumption is indeed satisfied.

We now turn our attention to Assumption B.1'. We will show that Assumption B.1' is satisfied with $T^\pi_{\mu_0} = t^\pi_{\mu_0}$, and $K = 0$. To prove this, it is clearly sufficient to show that for all $t \in [T]$,

$$t^\pi_{\mu_t}(\varphi_t(x)) = t^\pi_{\mu_0}(x), \tag{96}$$

since in this case the LHS of the bound in Assumption B.1' is identically zero. Our proof of (96) will consist of two steps. We will first establish that $\varphi_t(x) = t^{\mu_t}_{\mu_0}(x)$, i.e., the transport map defined by Alg. 1 coincides with the optimal transport map. We will then show that $t^\pi_{\mu_t}(t^{\mu_t}_{\mu_0}(x)) = t^\pi_{\mu_0}(x)$, i.e., the composition of the optimal transport maps $t^{\mu_t}_{\mu_0}$ and $t^\pi_{\mu_t}$ coincides with the optimal transport map $t^\pi_{\mu_0}$. By substituting the first of these identities into the second, we obtain the required result in (96).

We begin by showing that $\varphi_t(x) = t^{\mu_t}_{\mu_0}(x)$. To prove this, we will first establish that $\mu_t \sim \mathcal{N}(m_t, \Sigma)$ and $\varphi_t(x) = m_t + (x - m_0)$. We proceed by induction. In the base case ($t = 1$), it follows from the update equation (23) in Alg. 1 that $\varphi_1(x) = \mathrm{id}(x) = x$. We thus have, by definition, that $\mu_1 = (\varphi_1)_\# \mu_0 = \mu_0$. Thus, in particular, $\mu_1 \sim \mathcal{N}(m_1, \Sigma)$ and $\varphi_1(x) = m_1 + (x - m_0)$, in this case with $m_1 = m_0$. This proves the base case.

We now proceed to the inductive step. Assume that, for $s = 1, \ldots, t-1$, it is indeed the case that $\mu_s \sim \mathcal{N}(m_s, \Sigma)$ and $\varphi_s(x) = m_s + (x - m_0)$. We then have

$$\varphi_t(x) = x - \frac{\sum_{s=1}^{t-1} \nabla_{W_2}\mathcal{F}(\mu_s)(\varphi_s(x))}{Lt}\left(w_0 - \sum_{s=1}^{t-1}\langle \frac{1}{L}\nabla_{W_2}\mathcal{F}(\mu_s)(\varphi_s(x)), \varphi_s(x) - x\rangle\right) \tag{97}$$

$$= x + \underbrace{-\frac{\sum_{s=1}^{t-1}\Sigma^{-1}(m_s - m_\pi)}{Lt}\left(w_0 - \sum_{s=1}^{t-1}\langle \frac{1}{L}\Sigma^{-1}(m_s - m_\pi), m_s - m_0\rangle\right)}_{:= m_t - m_0} = m_t + (x - m_0), \tag{98}$$

where in the second line we have used the fact that $\nabla_{W_2}\mathcal{F}(\mu) = \nabla_{W_2}\mathrm{KL}(\mu|\pi) = \nabla \log \frac{\mu}{\pi} = \nabla \log \mu - \nabla \log \pi$, which implies in particular that

$$\nabla_{W_2}\mathcal{F}(\mu_s)(\varphi_s(x)) = \nabla \log \mu_s(x_s) - \nabla \log \pi(x_s) \tag{99}$$

$$= -\Sigma^{-1}(\varphi_s(x) - m_s) + \Sigma^{-1}(\varphi_s(x) - m_\pi) = \Sigma^{-1}(m_s - m_\pi). \tag{100}$$

Thus, we do indeed have $\varphi_t(x) = m_t + (x - m_0)$. Moreover, using standard properties regarding affine transformations of normal random variables, it follows that $\mu_t = (\varphi_t)_\# \mu_0 \overset{d}{=} \mathcal{N}(m_t, \Sigma)$. This completes the inductive step. We have thus shown that $\mu_t \sim \mathcal{N}(m_t, \Sigma)$ and $\varphi_t(x) = m_t + (x - m_0)$ for all $t \in \mathbb{N}$. The required identity, namely $t^{\mu_t}_{\mu_0}(x) = \varphi_t(x)$, now follows using the fact that the optimal transport map between $\mu_0 \sim \mathcal{N}(m_0, \Sigma)$ and $\mu_t \sim \mathcal{N}(m_t, \Sigma)$ is precisely given by $t^{\mu_t}_{\mu_0}(x) = m_t + (x - m_0)$ (e.g., Chen et al., 2019).

It remains to show that $t^\pi_{\mu_t}(t^{\mu_t}_{\mu_0}(x)) = t^\pi_{\mu_0}(x)$. Based on the previous result, and standard results on the optimal transport map between Gaussians (e.g., Chen et al., 2019), we have $t^\pi_{\mu_t}(x) = m_\pi + (x - m_t)$ and $t^{\mu_t}_{\mu_0}(x) = m_t + (x - m_0)$. It follows straightforwardly that

$$t^\pi_{\mu_t}(t^{\mu_t}_{\mu_0}(x)) = m_\pi + ([m_t + (x - m_0)] - m_t) = m_\pi + (x - m_0) = t^\pi_{\mu_0}(x). \tag{101}$$

We have thus proved both of the identities required to establish (96). This verifies Assumption B.1' with $T^\pi_{\mu_0} = t^\pi_{\mu_0}$, and $K = 0$. Finally, substituting $T^\pi_{\mu_0} = t^\pi_{\mu_0}$ and $K = 0$ into the bound in Proposition 3.3', we arrive at the required bound in Proposition 3.4. □

C. Other Coin ParVI Algorithms

In Sec. 3.4, we presented Coin SVGD, an algorithm which can be viewed as the coin sampling analogue of SVGD (Liu & Wang, 2016). In this section, we provide details of two other algorithms - Coin LAWGD and Coin KSDD - which represent coin sampling analogues of two other ParVI algorithms.

C.1. Coin Laplacian Adjusted Wasserstein Gradient Descent

Let $\mathcal{F}(\mu) = \mathrm{KL}(\mu|\pi)$, with $\nabla_{W_2}\mathcal{F}(\mu) = \nabla \ln \frac{\mathrm{d}\mu}{\mathrm{d}\pi}$. Following Chewi et al. (2020), suppose that we replace $\nabla_{W_2}\mathcal{F}(\mu)$ in Alg. 1 by $\nabla P_{\pi,k_\mathcal{L}} \frac{\mathrm{d}\mu}{\mathrm{d}\pi}$, the gradient of the image of $\frac{\mathrm{d}\mu}{\mathrm{d}\pi}$ under the integral operator $P_{\mu,k_\mathcal{L}}$, where $k_\mathcal{L}$ is the kernel such that $P_{\pi,k_\mathcal{L}} = -\mathcal{L}_\pi^{-1}$. Here, $\mathcal{L}_\pi$ denotes the infinitesimal generator of the overdamped Langevin diffusion with stationary distribution π. In this case, one can show that $\nabla P_{\pi,k_\mathcal{L}} \frac{\mathrm{d}\mu}{\mathrm{d}\pi} = \mathbb{E}_{x\sim\mu}[\nabla_1 k_\mathcal{L}(\cdot, x)]$ (Chewi et al., 2020, Sec. 4), and thus

$$\nabla P_{\pi,k_\mathcal{L}} \frac{\mathrm{d}\mu^N}{\mathrm{d}\pi}(x_t^i) = \frac{1}{N}\sum_{j=1}^N \nabla_1 k_\mathcal{L}(x_t^i, x_t^j). \tag{102}$$

By using these gradients in Alg. 1, we obtain a learning-rate free analogue of the LAWGD algorithm (Chewi et al., 2020, Alg. 1). This algorithm is summarised in Alg. 3.

Algorithm 3 Coin Laplacian Adjusted Wasserstein Gradient Descent (Coin LAWGD)

Input: initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}_d)$, initial particles $(x_0^i)_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mu_0$, initial wealth of particles $(w_0^i)_{i=1}^N \in \mathbb{R}_+$, gradient upper bound L.
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do**
    Compute

$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1}\sum_{j=1}^N \nabla_1 k_\mathcal{L}(x_s^i, x_s^j)}{LNt}\left(w_0^i - \sum_{s=1}^{t-1}\left\langle \frac{1}{LN}\sum_{j=1}^N \nabla_1 k_\mathcal{L}(x_s^i, x_s^j), x_s^i - x_0^i\right\rangle\right). \tag{103}$$

    Define $\mu_t^N = \frac{1}{N}\sum_{i=1}^N \delta_{x_t^i}$.
Output: μ_T^N or $\frac{1}{T}\sum_{t=1}^T \mu_t^N$.

C.2. Coin Kernel Stein Discrepancy Descent

Let $\mathcal{F}(\mu) = \frac{1}{2}\mathrm{KSD}^2(\mu|\pi)$, where $\mathrm{KSD}(\mu|\pi)$ is the kernel Stein discrepancy, defined according to (Liu et al., 2016; Chwialkowski et al., 2016; Gorham & Mackey, 2017)

$$\mathrm{KSD}(\mu|\pi) = \sqrt{\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} k_\pi(x,y)\mu(\mathrm{d}x)\mu(\mathrm{d}y)}, \tag{104}$$

and where k_π is the Stein kernel, defined in terms of the score $s = \nabla \log \pi$, and a positive semi-definite kernel k, as

$$k_\pi(x,y) = s^T(x)s(y)k(x,y) + s^T(x)\nabla_2 k(x,y) + \nabla_1 k^T(x,y)s(y) + \nabla_{\cdot 1}\nabla_2 k(x,y). \tag{105}$$

In this case, given a discrete measure $\mu^N = \frac{1}{N}\sum_{j=1}^N \delta_{x^j}$, the loss function and its gradient are given by

$$\mathcal{F}(\mu^N) = \frac{1}{N^2}\sum_{i,j=1}^N k_\pi(x^i, x^j) \quad , \quad \nabla_{x_i}\mathcal{F}(\mu_t^N) = \frac{1}{N^2}\sum_{j=1}^N \nabla_2 k_\pi(x_t^j, x_t^i). \tag{106}$$

By substituting these gradients into Alg. 1, we obtain a learning-rate free analogue of KSDD (Korba et al., 2021).[3] This algorithm is summarised in Alg. 4.

[3] In fact, Korba et al. (2021) also propose a learning-rate free version of KSDD based on the quasi-Newton L-BFGS algorithm (Liu & Nocedal, 1989). Our method provides an alternative approach based on the 'coin-betting' paradigm.

Algorithm 4 Coin Kernel Stein Discrepancy Descent (Coin KSDD)

Input: initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}_d)$, initial particles $(x_0^i)_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mu_0$, initial wealth of particles $(w_0^i)_{i=1}^N \in \mathbb{R}_+$, kernel k, gradient upper bound L.
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do**
    Compute
$$x_t^i = x_0^i - \frac{\sum_{s=1}^{t-1}\sum_{j=1}^{N}\nabla_2 k_\pi(x_s^j, x_s^i)}{LN^2 t}\left(w_0^i - \sum_{s=1}^{t-1}\left\langle \frac{1}{LN^2}\sum_{j=1}^{N}\nabla_2 k_\pi(x_s^j, x_s^i), x_s^i - x_0^i\right\rangle\right). \tag{107}$$
    Define $\mu_t^N = \frac{1}{N}\sum_{i=1}^{x,N}\delta_{x_t^i}$.
Output: μ_T^N or $\frac{1}{T}\sum_{t=1}^{T}\mu_t^N$.

D. Coin Sampling with Adaptive Gradient Bounds

D.1. Adaptive Coin Wasserstein Gradient Descent

In principle, Alg. 1 (and Alg. 2) depend on knowledge of a constant $L > 0$ such that, for all $t \in [T]$, $||\nabla_{W_2}\mathcal{F}(\mu_t)(x_t)|| \leq L$. In practice, however, such a constant may not be not known in advance. In this case, following Orabona & Tommasi (2017), we can use a modified version of our algorithm in which the gradient bounds are adaptively estimated. This algorithm is summarised in Alg. 5.

Algorithm 5 Adaptive Coin Wasserstein Gradient Descent

Input: initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$, initial parameter $x_0 \sim \mu_0$, dissimilarity functional $\mathcal{F} : \mathcal{P}_2(\mathbb{R}^d) \to (-\infty, \infty]$.
Initialise: for $j = 1, \ldots, d$, $L_{0,j} = 0$, $G_{0,j} = 0$, $R_{0,j} = 0$.
for $t = 1$ **to** T **do**
  Compute the negative Wasserstein gradient: $c_{t-1} = -\nabla_{W_2}\mathcal{F}(\mu_{t-1})(x_{t-1})$.
  for $j = 1$ **to** d **do**
    Update the maximum observed scale $L_{t,j} = \max(L_{t-1,j}, |c_{t-1,j}|)$.
    Update the sum of the absolute value of the gradients: $G_{t,j} = G_{t-1,j} + |c_{t-1,j}|$.
    Update the reward: $R_{t,j} = \max(R_{t-1,j} + c_{t-1,j}(x_{t-1,j} - x_{0,j}), 0)$.
    Update the parameter
$$x_{t,j} = x_{0,j} + \frac{\sum_{s=1}^{t-1} c_{s,j}}{G_{t,j} + L_{t,j}}(1 + \frac{R_{t,j}}{L_{t,j}}). \tag{108}$$
  Define $\mu_t = (\varphi_t)_{\#}\mu_0$, where $\varphi_t : x_0 \mapsto x_t$.
Output: μ_T.

D.2. Adaptive Coin Stein Variational Gradient Descent

In the same way, one can also obtain an adaptive version of Coin SVGD (Alg. 2) and, indeed, Coin LAWGD (Alg. 3) and Coin KSDD (Alg. 4).[4] The adaptive Coin SVGD algorithm is summarised in Alg. 6. Following Orabona & Tommasi (2017), we make one further alteration when we use the adaptive version of Coin SVGD to perform inference in Bayesian neural networks (see Sec. 4.4). In particular, we now modify the denominator of the betting fraction in Alg. 6 such that it is at least $\alpha L_{t,j}^i$, for some positive constant $\alpha > 0$, which as a default we set equal to 100. Thus, the update in (110) now becomes

$$x_{t,j}^i = x_{0,j}^i + \frac{\sum_{s=1}^{t-1} c_{s,j}^i}{\max(G_{t,j}^i + L_{t,j}^i, \alpha L_{t,j}^i)}(1 + \frac{R_{t,j}^i}{L_{t,j}^i}). \tag{109}$$

In practice, is is these adaptive algorithms that we use in all of our numerical experiments (Sec. 4), and that we recommend for use in future work.

[4] In the interest of brevity, we do not provide the adaptive versions of Coin LAWGD and Coin KSDD in full. However, these are easily obtained by substituting the relevant gradients into the adaptive version of Coin SVGD.

Algorithm 6 Adaptive Coin Stein Variational Gradient Descent

Input: initial measure $\mu_0 \in \mathcal{P}_2(\mathbb{R}^d)$; initial particles $(x_0^i)_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mu_0$.
Initialise: for $i = 1, \dots, N$, $j = 1, \dots, d$, $L_{0,j}^i = 0$, $G_{0,j}^i = 0$, $R_{0,j}^i = 0$.
for $t = 1$ **to** T **do**
  for $i = 1$ **to** N **do**
    Compute the negative gradient $c_{t-1}^i = -\frac{1}{N}\sum_{j=1}^N [k(x_{t-1}^j, x_{t-1}^i)\nabla U(x_{t-1}^j) - \nabla_1 k(x_{t-1}^j, x_{t-1}^i)]$.
    for $j = 1$ **to** d **do**
      Update the maximum observed scale: $L_{t,j}^i = \max(L_{t-1,j}^i, |c_{t-1,j}^i|)$.
      Update the sum of the absolute value of the gradients: $G_{t,j}^i = G_{t-1,j}^i + |c_{t-1,j}^i|$.
      Update the reward $R_{t,j}^i = \max(R_{t-1,j}^i + \langle c_{t-1,j}^i, x_{t-1,j}^i - x_{0,j}^i\rangle, 0)$.
      Update the parameter

$$x_{t,j}^i = x_{0,j}^i + \frac{\sum_{s=1}^{t-1} c_{s,j}^i}{G_{t,j}^i + L_{t,j}^i}(1 + \frac{R_{t,j}^i}{L_{t,j}^i}). \tag{110}$$

  Define $\mu_t^N = \frac{1}{N}\sum_{i=1}^N \delta_{x_t^i}$.
Output: μ_T^N.

Computational Complexity. In terms of computational cost and memory requirements, the adaptive variant of Coin SVGD is similar to SVGD when the latter is paired, as is common, with a method such as Adagrad (Duchi et al., 2011), RMSProp (Tieleman & Hinton, 2012), or Adam (Kingma & Ba, 2015). The computational cost per iteration is $O(N^2)$ in the number of particles N, due to the kernelised gradient. In terms of memory requirements, it is necessary to keep track of the sum of the previous gradients, the sum of the absolute value of the previous gradients, the maximum observed absolute value of the previous gradients, and the reward, for each of the particles. Thus, in particular, the memory requirement is $O(Nd)$ in the number of particles N and the dimension d. This is identical to, e.g., Adagrad, which must keep track of the sum of the squares of the previous gradients, for each of the particles (Duchi et al., 2011).

E. Additional Experimental Details and Numerical Results

We implement our methods using Python 3, PyTorch, Theano, and Jax. For our comparisons, we use existing implementations of `SVGD` by Liu & Wang (2016), `LAWGD` by Chewi et al. (2020), and `KSDD` by Korba et al. (2021). We perform all experiments using a MacBook Pro 16" (2021) laptop with Apple M1 Pro chip and 16GB of RAM.

E.1. Toy Examples

E.1.1. Coin SVGD

Experimental Details. In Sec. 1, we compare the performance of SVGD (Liu & Wang, 2016) and Coin SVGD (Alg. 2) on the following two-dimensional distributions.

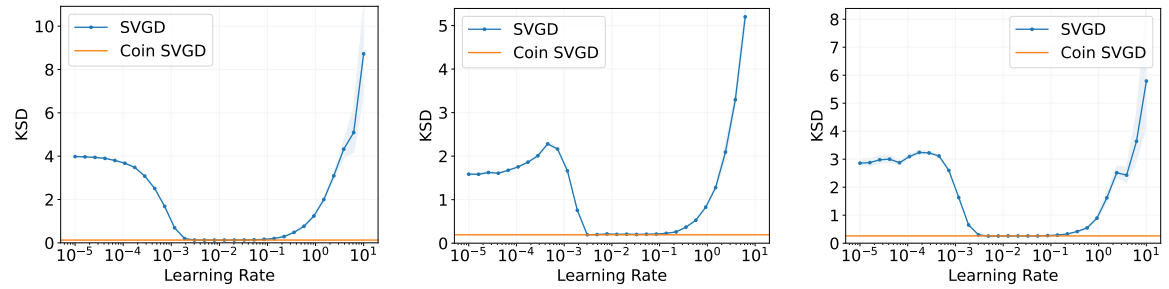
Two-Dimensional Gaussian. We first consider an anisotropic bivariate Gaussian distribution, $p(x) = \mathcal{N}(x|\mu, \Sigma)$, where we set $\mu = (-1, 1)^\top$ and $\Sigma^{-1} = \left(\begin{smallmatrix} 3 & -0.5 \\ -0.5 & 1 \end{smallmatrix}\right)$.

Mixture of Two Two-Dimensional Gaussians. For the second example, we consider a mixture of two bivariate Gaussian distributions, $p(x) = \alpha_1\mathcal{N}(x; \mu_1, \Sigma_1) + \alpha_2\mathcal{N}(x; \mu_2, \Sigma_2)$, with $\alpha_1 = 0.5$, $\mu_1 = (-2, 2)^\top$, and $\Sigma_1 = \frac{1}{2}\mathbb{1}$; $\alpha_2 = 0.5$, $\mu_2 = (2, -2)^\top$, and $\Sigma_2 = \frac{1}{2}\mathbb{1}$.

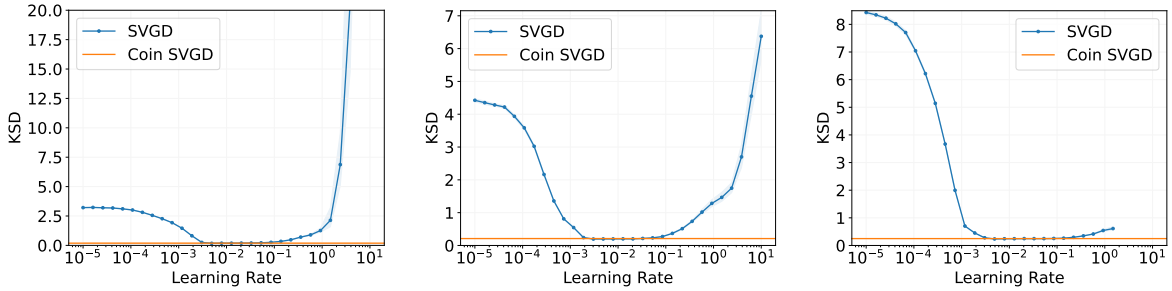
Donut Distribution. We next consider an annulus or 'donut' distribution, with density $p(x) \propto \exp(-\frac{(|x|-r_0)^2}{2\sigma^2})$, where we set $r_0 = 2.5$ and $\sigma^2 = 0.5$.

Rosenbrock Distribution. The next example is a variant on the so-called Rosenbrock or 'banana' distribution (Pagani et al., 2022). The target density is given by $p(x) \propto \exp[-\frac{1}{2}(x'-\mu)^\top\Sigma^{-1}(x'-\mu)]$, where $x_1' = x_1/a$, and $x_2' = ax_1 + ab(x_1^2 + a^2)$. In our experiments, we set $a = -1$, $b = 1$, $\mu = (0, 1)^\top$, and $\Sigma = \left(\begin{smallmatrix} 1 & 0.5 \\ 0.5 & 1 \end{smallmatrix}\right)$.

Squiggle Distribution. Our penultimate example is a two-dimensional 'squiggle' distribution; see, e.g., App. E in Hartmann et al. (2022). In this case, the target density again takes the form $p(x) \propto \exp\left[-\frac{1}{2}(x' - \mu)^\top\Sigma^{-1}(x' - \mu)\right]$, where now



(a) Gaussian. (b) Mixture of Gaussians. (c) Donut.



(d) Rosenbrock Banana. (e) Squiggle. (f) Funnel.

Figure 6. **Additional results for the toy examples in Sec. 4.1.** Plots of the kernel Stein discrepancy (KSD) between each of the target distributions in Fig. 1, and the corresponding approximations generated by Coin SVGD and SVGD. We run SVGD over a grid of 30 logarithmically spaced learning rates $\gamma \in [1 \times 10^{-5}, 1 \times 10^{1}]$. We run both algorithms for $T = 1000$ iterations, and using $N = 20$ particles. The results are averaged over 50 random trials.

$x_1' = x_1$ and $x_2' = x_2 + \sin(\omega x_1)$. In our experiments, we set $\mu = (1, 1)^\top$, $\Sigma = \begin{pmatrix} 2 & 0.25 \\ 0.25 & 0.5 \end{pmatrix}$, and the frequency $\omega = 2$.

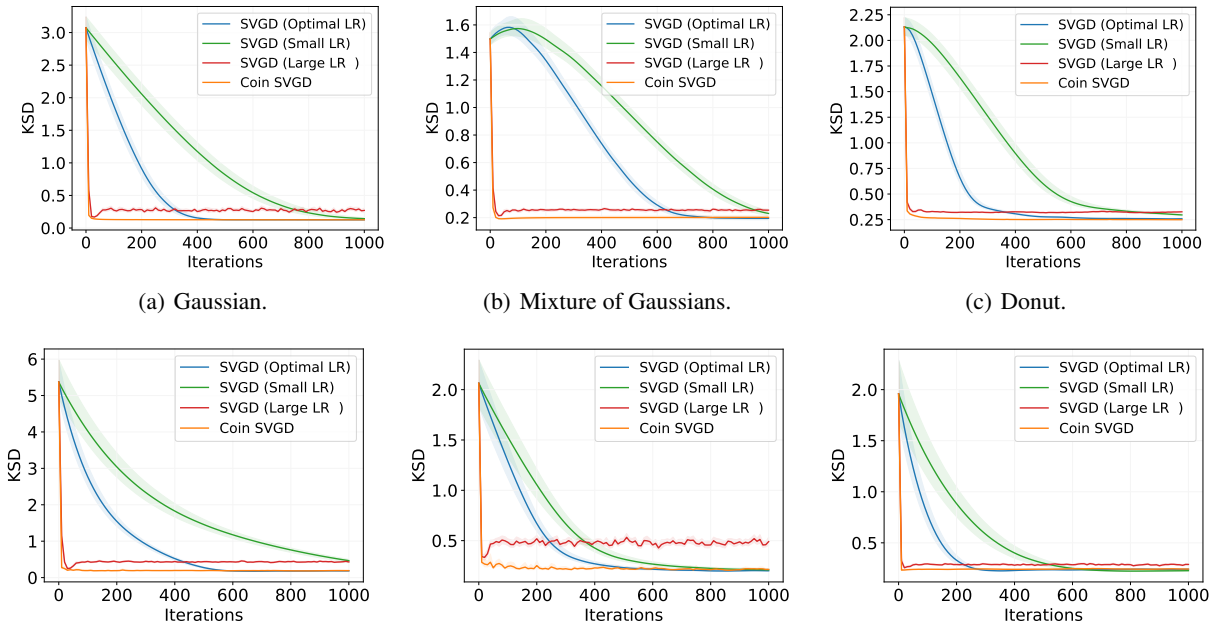
Funnel Distribution. Our final example is a two-dimensional 'funnel' distribution, with density $p(x_1, x_2) \propto \mathcal{N}(x_1; \mu_1, \exp(x_2))\mathcal{N}(x_2; \mu_2, \sigma_2^2)$. In our experiments, we set $\mu_1 = 1$, $\mu_2 = 4$, and $\sigma_2 = 3$. This example, in ten-dimensions, was first introduced in Neal (2003) to illustrate the difficulty of sampling from some hierarchical models.

In all cases, we run both algorithms using $N = 20$ particles, and for $T = 1000$ iterations. We initialise the particles according to $(\theta_0^i)_{i=1}^N \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.1^2)$. Finally, we use Adagrad (Duchi et al., 2011) to adapt the learning rate for SVGD.

Numerical Results. In Fig. 6 and Fig. 7, we provide a more detailed comparison of SVGD and Coin SVGD. In particular, in Fig. 6, we plot the the KSD for both algorithms after $T = 1000$ iterations as a function of the learning rate. Meanwhile, in Fig. 7, we plot the KSD as a function of the iterations, using the optimal learning rate as determined by the results in Fig. 6, and two other learning rates. In both cases, following Gorham & Mackey (2017), we use the inverse multi-quadratic (IMQ) kernel $k(x, x') = (c^2 + ||x - x'||_2^2)^\beta$ to compute the KSD, where $c > 0$ and $\beta \in (-1, 0)$. In the interest of a fair comparison, we also use Adagrad (Duchi et al., 2011) to adapt the learning rate in SVGD.

In these examples, the performance of Coin SVGD is competitive with the best performance of SVGD, using the optimal but a prior unknown learning rate. Moreover, Coin SVGD clearly outperforms SVGD for sub-optimal choices of the learning rate, attaining significantly lower values of the KSD. In particular, using a step size which is too small is insufficient to guarantee convergence within 1000 iterations (Fig. 7, green lines), while using a step size which is too large leads to non-convergence (Fig. 7, red lines) and, ultimately, numerical instability. It is worth emphasising that it is difficult to determine a good step size, or to implement a line-search method, since SVGD does not minimise a simple function. On the other hand, Coin SVGD achieves performance close to, or even better than, the performance of optimally-tuned SVGD, without any need to tune a step size.

In Fig 8 and 9, we further investigate the performance of our algorithm as a function of the number of particles N. In particular, we now compute the energy distance (Fig. 8) and the KSD (Fig. 9) for SVGD and Coin SVGD as a function of N. In terms of tuning the learning rate used by SVGD, we consider two possibilities. The first is that the learning rate is only tuned once, using a fixed and pre-determined number of particles. This is the case in Fig. 8. More generally, this



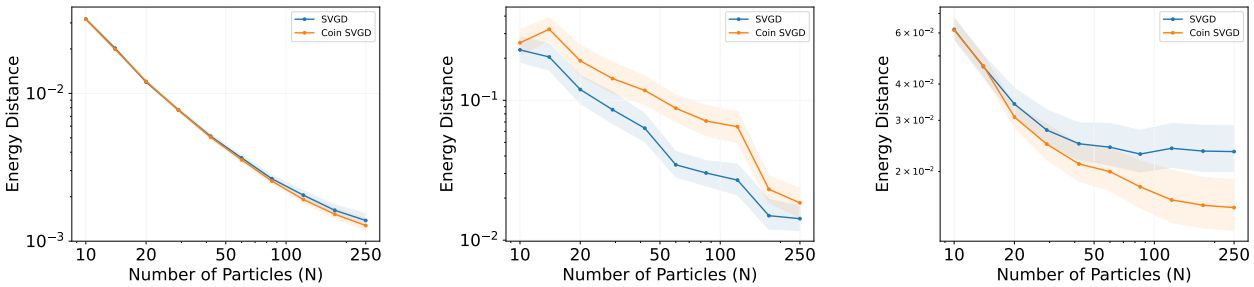
(a) Gaussian. (b) Mixture of Gaussians. (c) Donut.

(d) Rosenbrock Banana. (e) Squiggle. (f) Funnel.

Figure 7. **Additional results for the toy examples in Sec. 4.1.** Plots of the KSD between each of the target distributions in Fig. 1, and the corresponding approximations generated by Coin SVGD and SVGD, as a function of the number of iterations. We run SVGD use three learning rates: the optimal learning rate as determined by Fig. 6 (blue), a smaller learning rate of $\gamma = 2 \times 10^{-3}$ (green), and a larger learning rate of $\gamma = 2 \times 10^{-1}$ (red). We run both algorithms for $T = 1000$ iterations, and using $N = 20$ particles. The results are averaged over 50 random trials.

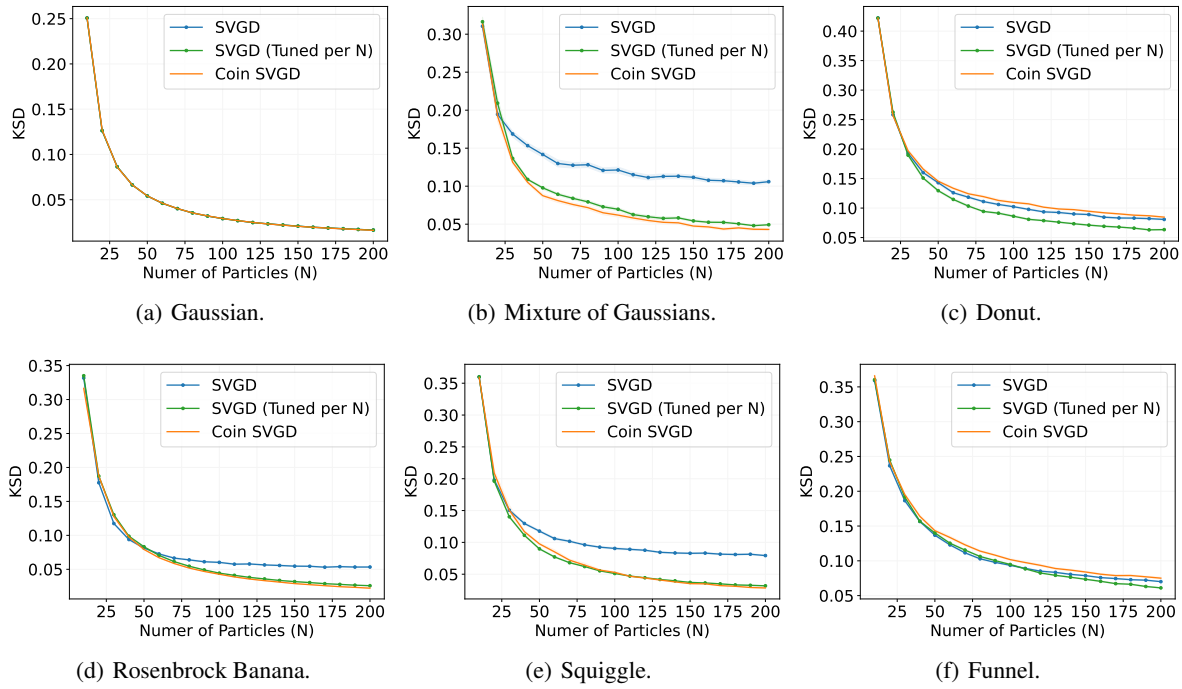
may be a realistic scenario in settings where one would like to use a large number of particles (e.g., for complex targets), but it is prohibitively expensive to tune the learning rate using this number (e.g., for high-dimensional targets, or big data settings). In the second case, the learning rate is re-tuned every time we change the number of particles. In both cases, we determined the optimal learning rate by running SVGD over a grid of 30 learning rates $\gamma \in [1 \times 10^{-5}, 1 \times 10^{1}]$, and selecting the learning rate which achieves the lowest KSD after 1000 iterations (as in Fig. 11).

For both algorithms, the energy distance and the KSD converge towards zero as the number of particles increases, suggesting that the approximate posterior samples generated provide increasingly accurate approximations of the true posterior (e.g. Gorham & Mackey, 2017, Theorem 8). Interestingly, our results also suggest that the best choice of learning rate in SVGD



(a) Gaussian. (b) Mixture of Gaussians. (c) Squiggle.

Figure 8. **Additional results for the toy examples in Sec. 4.1.** Plots of the energy distance between a subset of the target distributions in Fig. 1, and the corresponding approximations generated by Coin SVGD and SVGD, as a function of the number of particles. In each case, we run SVGD using the best learning rate as determined by the results in Fig. 6. We run both algorithms for $T = 1000$ iterations, and using between $N = 10$ and $N = 250$ particles. The results are averaged over 50 random trials.



(a) Gaussian. (b) Mixture of Gaussians. (c) Donut.

(d) Rosenbrock Banana. (e) Squiggle. (f) Funnel.

Figure 9. **Additional results for the toy examples in Sec. 4.1.** Plots of the KSD between each of the target distributions in Fig. 1, and the corresponding approximations generated by Coin SVGD and SVGD, as a function of the number of particles N. We run both algorithms for $T = 1000$ iterations. For SVGD, we either tune the learning rate once, using a fixed and pre-determined $N = 20$ particles (blue); or re-tune the learning rate every time we change the number of particles (green). The results are averaged over 50 random trials.

can be somewhat sensitive to the number of particles. In particular, if one tunes the SVGD learning rate using a small number of particles (in Fig. 6, we use $N = 20$), and then runs the full algorithm using a large value of N with the same learning rate, this can lead to sub-optimal performance (e.g., Fig. 9(b), Fig. 9(d), or Fig. 9(e)). Coin SVGD suffers no such problems, and provides us with an approach which is robust to the specification of the number of particles.

E.1.2. Coin LAWGD

Experimental Details. We next compare the performance of LAWGD (Chewi et al., 2020) and Coin LAWGD (Alg. 3) on the following examples.

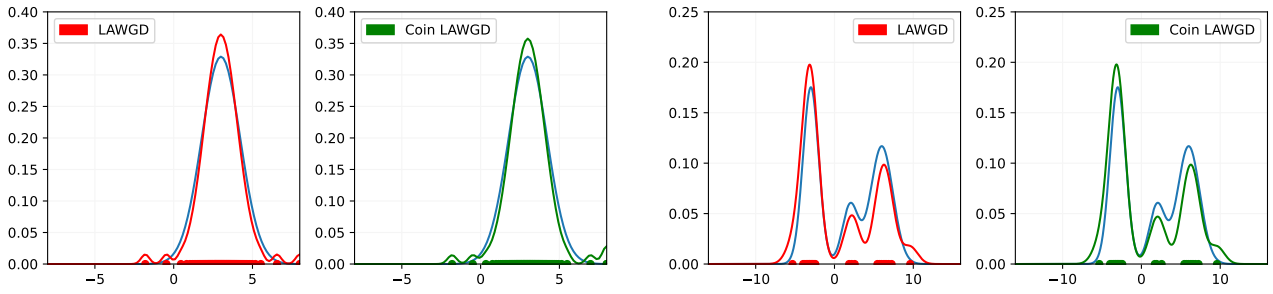
One-Dimensional Gaussian. We begin by considering a simple one-dimensional Gaussian, with density $p(x) = \mathcal{N}(x; \mu, \sigma^2)$, where $\mu = 3$ and $\sigma^2 = 1.5$.

Mixture of Three One-Dimensional Gaussians. We also consider a mixture of three one-dimensional Gaussians, with $p(x) = \sum_{i=1}^{3} \alpha_i \mathcal{N}(x|\mu_i, \sigma_i^2)$, where we set $\alpha_1 = \frac{1}{3}$, $\mu_1 = 6$, $\sigma_1^2 = 2$; $\alpha_2 = \frac{1}{2}$, $\mu_2 = -3$, and $\sigma_2^2 = 1$; and $\alpha_3 = \frac{1}{6}$, $\mu_3 = 2$, and $\sigma_3^2 = 1$.

We use either $N = 100$ particles (Fig. 10(a)) or $N = 25$ particles (Fig. 10(b)). The initial particles are i.i.d. $\mathcal{U}(-1, 1)$, or i.i.d. $\mathcal{U}(-2, 2)$. In both cases, we run the algorithms for $T = 2500$ iterations.

To compute the kernel $k_{\mathcal{L}}$ required to implement LAWGD and Coin LAWGD, it is necessary to approximate the eigenfunctions and eigenvalues of the operator $\mathcal{L}$ (Chewi et al., 2020, Sec. 5). Following Chewi et al. (2020), we approximate these quantities using a basic finite difference scheme. For the target in Fig. 10(a), we use a finite difference scheme based on 1000 equally spaced grid points between -8 and 8, and use the first 150 eigenvalues and eigenfunctions. For the target in Fig. 10(b), we use 500 equally spaced grid points between -16 and 16, and again use the first 150 eigenvalues and eigenfunctions.

Numerical Results. In Fig. 10, we plot an illustrative set of samples obtained using Coin LAWGD and LAWGD for these two examples. Similar to our other coin sampling algorithms, we see that Coin LAWGD converges to the target distribution for both of our test cases, and enjoys a similar performance to the standard LAWGD algorithm.



(a) Gaussian. (b) Mixture of Three Gaussians.

Figure 10. **A comparison between LAWGD (Chewi et al., 2020) and its learning-rate free analogue, Coin LAWGD (Alg. 3).** We plot the samples generated by both methods for the two target distributions detailed in App. E.1.2.

E.1.3. COIN KSDD

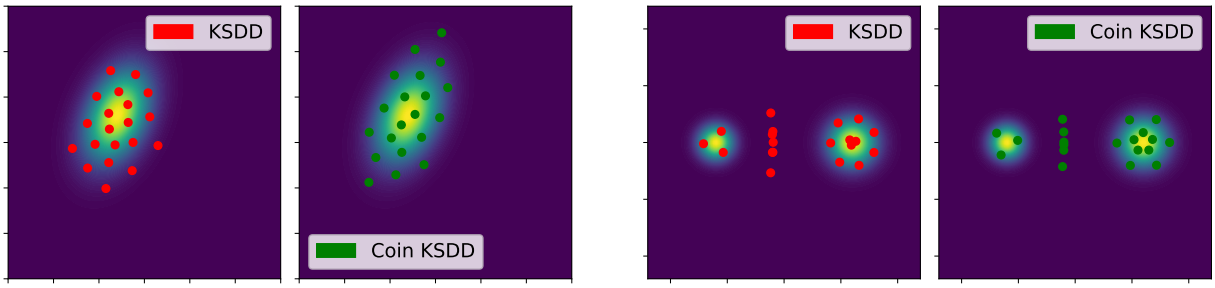
Experimental Details We next compare the performance of KSDD (Korba et al., 2021) and Coin KSDD (Alg. 4). We consider the following examples.

Anistropic Two-Dimensional Gaussian. We first consider a single bivariate Gaussian, $p(x) = \mathcal{N}(x; \mu, \Sigma)$, where $\mu = (-3, 3)^\top$ and $\Sigma^{-1} = \left(\begin{smallmatrix} 0.2 & -0.05 \\ -0.05 & 0.1 \end{smallmatrix}\right)$.

Symmetric Mixture of Two Two-Dimensional Gaussians. For our second and third examples, we consider a symmetric mixture of two, two-dimensional, isotropic Gaussians with different covariances. In particular, $p(x) = \frac{1}{2}\mathcal{N}(x; \mu, \sigma_1^2 \mathbb{1}) + \frac{1}{2}\mathcal{N}(x; -\mu, \sigma_2^2 \mathbb{1})$, where $\mu = (6, 0)^\top$, $\sigma_1^2 = 2$, $\sigma_2^2 = 1$ in Fig. 11(b); and $\mu = (5, 5)^\top$, $\sigma_1^2 = 2$, $\sigma_2^2 = 2$ in Fig. 12.

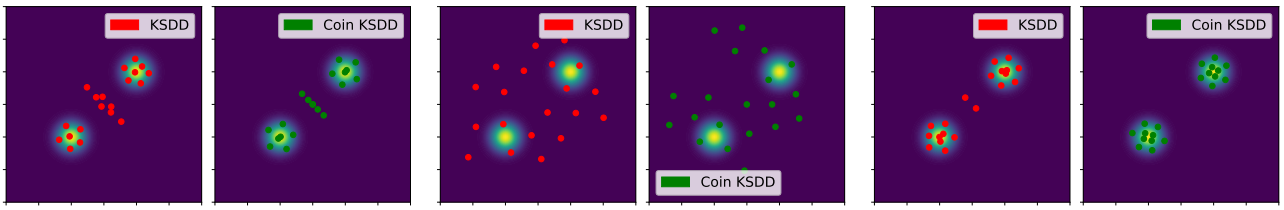
We use $N = 20$ particles, and run both methods for $T = 5000$ iterations. We initialise the particles according from $\mathcal{N}(0, 0.5^2)$ in Fig. 11(a), or $\mathcal{N}(0, 2^2)$ in Fig. 11(b).

Numerical Results. In Fig. 11, we plot the samples obtained using KSDD and Coin KSDD after 5000 iterations. Similar to before, the samples generated by our coin sampling method are very similar to those generated by the original algorithm. In fact, even the dynamics of the two algorithms share many of the same properties. For example, the Coin KSDD particles seem initially to be guided by the final repulsive term in the update, which determine their global arrangement. They are then transported towards the mode(s), driven by the remaining score-based terms. This is in contrast to the Coin SVGD particles, which are first driven by the score term, before being dispersed around the mode by the repulsive term. These dynamics were first observed in Korba et al. (2021) for the standard SVGD and KSDD algorithms, and also to be present for their step-size free analogues.



(a) Anisotropic Gaussian. (b) Symmetric Mixture of Gaussians.

Figure 11. **A comparison between KSDD (Korba et al., 2021) and its learning-rate free analogue, Coin KSDD (Alg. 4).** We plots the samples generated by both methods for the two target distributions detailed in App. E.1.3.



(a) $\beta = 1$ (b) $\beta = 0.02$ (c) $\beta = 0.02 \to 1$.

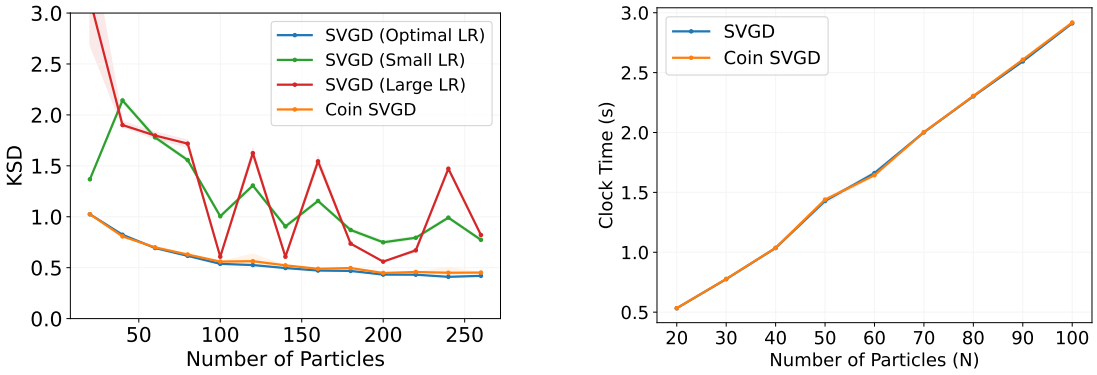
Figure 12. **A comparison between annealing KSDD (Korba et al., 2021) and annealing Coin KSDD (Alg. 4).** Samples generated by both methods using no annealing ($\beta = 1$), after the first step of the annealing method ($\beta = 0.02$), and after the full annealing method ($\beta = 0.02 \to \beta = 1$).

Unsurprisingly, Coin KSDD also inherits some of the shortcomings of KSDD. Given a symmetric target, and a radial kernel, it is known that any plane of symmetry is invariant under the KSD gradient flow (Korba et al., 2021, Lemma 11). Thus, if KSDD is initialised close to a plane of symmetry, it can become stuck there indefinitely. In practice, this also appears to holds true for Coin KSDD (see Fig. 11). Korba et al. (2021) propose an annealing strategy can be used to resolve this behaviour; see also Wenliang & Kanagawa (2021). One first runs KSDD to obtain samples from the target $\pi_\beta(x) \propto \exp(-\beta U(x))$, where the inverse temperature $\beta \sim 0$. One then runs the algorithm a second time, initialised at these samples, on the true target $\pi(x) \propto \exp(-U(x))$. A similar strategy can also be used for Coin KSDD (see Fig. 12).

E.2. Bayesian Independent Component Analysis

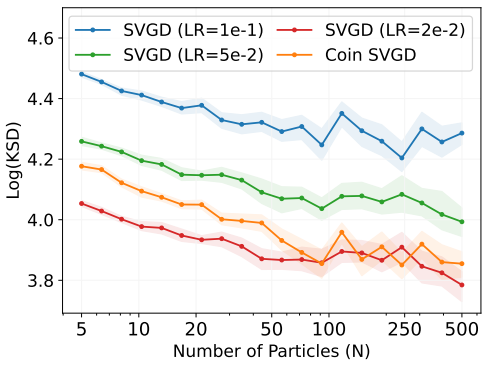
Additional Experimental Details. For the results in Fig. 2 (Sec. 4.2) and Fig. 13 (below), we tune the SVGD learning rate by running SVGD for $T = 1000$ iterations, using learning rates $\gamma \in \{1\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}, 1\times10^{-2}, 1\times10^{-1}, 1\times10^{0}\}$. We then define the optimal learning rate as the one for which SVGD outputs approximate unmixing matrices $(\bar{\boldsymbol{W}}_i)_{i=1}^{10}$ with the lowest Amari distance to the true unmixing matrix $\boldsymbol{W}$, averaged over 10 random trials. The small and large learning rates are then chosen to be one order of magnitude smaller or greater than the optimal learning rate, respectively.

Additional Numerical Results. In Fig. 13, we provide a further comparison between Coin SVGD and SVGD, plotting the KSD (Fig. 13(a)) and the clock time (Fig. 13(b)) as a function of the number of particles N, in the case $p = 4$. Similar to elsewhere, the performance of Coin SVGD is similar to the best performance of SVGD (out of the learning rates considered), and both algorithms provide increasingly accurate approximations of the posterior as the number of particles increases. The computational cost of both algorithms is essentially identical.

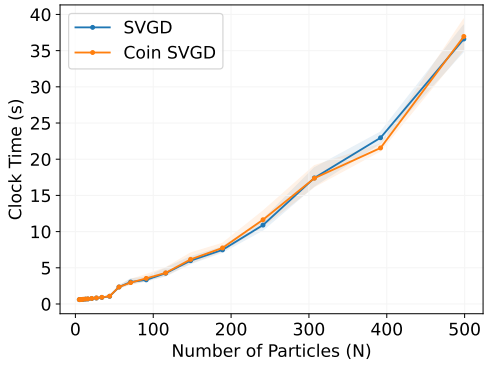


(a) KSD vs Number of Particles. (b) Time vs Number of Particles.

Figure 13. **Additional results for the Bayesian ICA model**. (a) KSD between the target posterior and the posterior approximations generated by Coin SVGD and SVGD after $T = 1000$ iterations, as a function of the number of particles N. (b) Time (s) to run Coin SVGD and SVGD for $T = 1000$ iterations as a function of the number of particles. For both sets of results, we average the results over 10 random trials.



(a) KSD vs Number of Particles.



(b) Time vs Number of Particles.

Figure 14. **Additional results for the Bayesian logistic regression model**. (a) KSD between the target posterior and the posterior approximations generated by Coin SVGD and SVGD after $T = 2500$ iterations, as a function of the number of particles N. (b) Time (s) to run Coin SVGD and SVGD for $T = 2500$ iterations as a function of the number of particles. For both sets of results, we average the results over 20 random train-test splits.

E.3. Bayesian Logistic Regression

Additional Experimental Details. For the results in Fig. 3(a) and Fig. 3(b), we run SVGD using a grid of 10 logarithmically spaced learning rates $\gamma \in [1 \times 10^{-5}, 1 \times 10^{0}]$. For the results in Fig. 3(c) and Fig. 3(d), we then define the 'optimal' learning rate to be the one which obtained the highest test accuracy after $T = 5000$ iterations. Meanwhile, the 'small' and 'big' learning rates correspond to those with indices two smaller or two larger than the index of the optimal learning rate in the original grid of learning rates. As in Liu & Wang (2016), we use Adagrad (Duchi et al., 2011) to adapt the learning rate for SVGD on the fly.

Additional Numerical Results. In Fig. 14, we provide additional numerical results for the Bayesian logistic regression considered in Sec. 4.3. In Fig. 14(a), we plot the KSD between the posterior approximations generated by SVGD and Coin SVGD, and the true target posterior, as a function of the number of particles. For SVGD, we consider several fixed learning rates, namely $\gamma \in \{2 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-1}\}$ across all values of N, which are determined based on the results in Fig. 3(a) and Fig. 3(b). Our results suggest, as we would expect, that the KSD achieved by both methods decreases as a function of the number of particles; i.e., both methods generate increasingly accurate approximations of the target posterior as the number of particles increases. Meanwhile, the performance of Coin SVGD is broadly comparable to the best performance of SVGD (among the learning rates considered), as measured by the KSD.

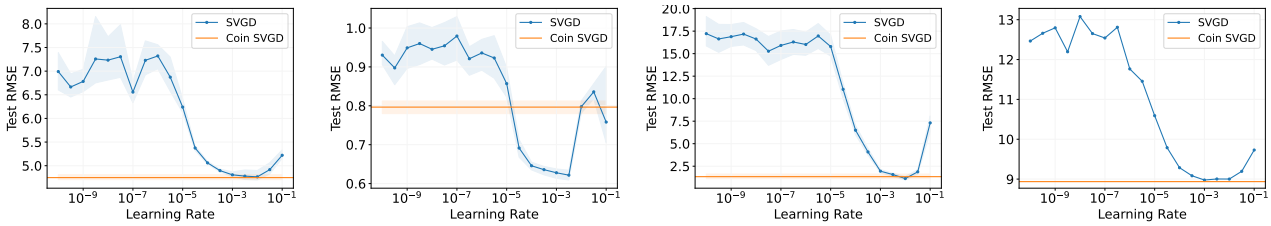
In Fig. 14(b), we plot the time taken by Coin SVGD and SVGD to complete $T = 2500$ iterations, again as a function of the number of particles. Here, we see no meaningful difference between the two algorithms. This is unsurprising on the basis of our earlier discussion on computational cost in Sec. D.2.

E.4. Bayesian Neural Network

Additional Experimental Details. For the results in Fig. 4 (Sec. 4.4) and Fig. 15 (see below), we run SVGD using a grid of 20 logarithmically spaced learning rates $\gamma \in [1 \times 10^{-10}, 1 \times 10^{-0.5}]$. Following Liu & Wang (2016), for the Protein and Year datasets in Fig. 15, we use 100 hidden units (rather than 50). In addition, we use a mini-batch size of 1000 for Year (rather than 100). We report results averaged over 20 random train-test splits for all datasets other than Year, for which we just report a single run. Once again, we use Adagrad (Duchi et al., 2011) to adapt the learning rate for SVGD.

Additional Numerical Results. In Fig. 15, we plot the average test RMSE achieved by Coin SVGD and SVGD after $T = 2000$ iterations, for several additional UCI datasets. As noted in the main text, for certain datasets there remains a considerable gap between the optimal performance of SVGD, and the performance of Coin SVGD (see, e.g., Fig. 15(b)). We expect, however, that this performance gap could be significantly reduced by appropriately extending recent advances in parameter-free stochastic optimisation to our setting (e.g. Chen et al., 2022a;b).

Finally, in Fig. 16, we plot the average time (s) for both methods to complete $T = 2000$ iterations. As in our previous experiments, there is no meaningful difference between the two methods in terms of clock time.



(a) Protein. (b) Wine. (c) Yacht. (d) Year.

Figure 15. **Additional results for the Bayesian neural network**. Average test RMSE for Coin SVGD and SVGD, as a function of the learning rate, after $T = 2000$ iterations, for several additional UCI datasets.

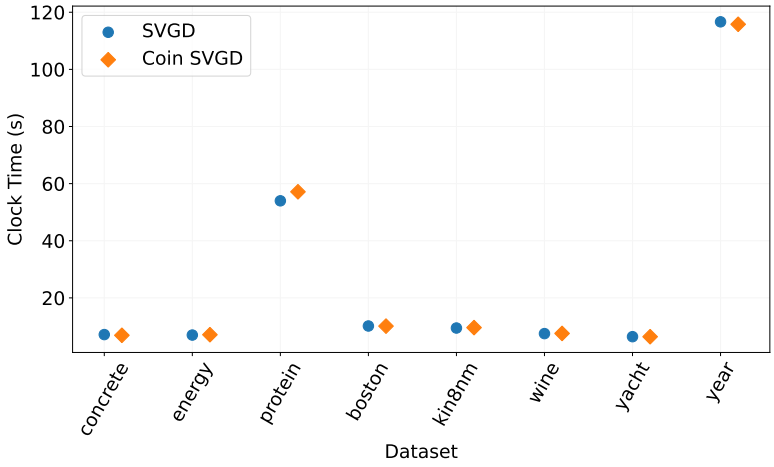


Figure 16. **Additional results for the Bayesian neural network**. Time (s) to run Coin SVGD and SVGD for $T = 2000$ iterations, for each of the UCI datasets considered in Fig. 4 and Fig. 15.