

---

# CoinEM: Tuning-Free Particle-Based Variational Inference for Latent Variable Models

---

**Louis Sharrock\***

Department of Mathematics and Statistics  
Lancaster University, UK  
l.sharrock@lancaster.ac.uk

**Daniel Dodd\***

Department of Mathematics and Statistics  
Lancaster University, UK  
d.dodd@lancaster.ac.uk

**Christopher Nemeth**

Department of Mathematics and Statistics  
Lancaster University, UK  
c.nemeth@lancaster.ac.uk

## Abstract

We introduce two new particle-based algorithms for learning latent variable models via marginal maximum likelihood estimation, including one which is entirely tuning-free. Our methods are based on the perspective of marginal maximum likelihood estimation as an optimization problem: namely, as the minimization of a free energy functional. One way to solve this problem is to consider the discretization of a gradient flow associated with the free energy. We study one such approach, which resembles an extension of the popular Stein variational gradient descent algorithm. In particular, we establish a descent lemma for this algorithm, which guarantees that the free energy decreases at each iteration. This method, and any other obtained as the discretization of the gradient flow, will necessarily depend on a learning rate which must be carefully tuned by the practitioner in order to ensure convergence at a suitable rate. With this in mind, we also propose another algorithm for optimizing the free energy which is entirely learning rate free, based on coin betting techniques from convex optimization. We validate the performance of our algorithms across a broad range of numerical experiments, including several high-dimensional settings. Our results are competitive with existing particle-based methods, without the need for any hyperparameter tuning.

## 1 Introduction

In statistics and machine learning, probabilistic latent variable models  $p_\theta(z, x)$  comprising model parameters  $\theta \in \Theta \subseteq \mathbb{R}^{d_\theta}$ , unobserved *latent variables*  $z \in \mathcal{Z} \subseteq \mathbb{R}^{d_z}$ , and *observations*  $x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ , are widely used to capture the hidden structure of complex data such as images [7], audio [65], text [8], and graphs [31]. In this paper, we consider the task of estimating the parameters in such models by maximizing the marginal likelihood of the observed data,

$$\theta_* = \arg \max_{\theta \in \Theta} p_\theta(x) := \arg \max_{\theta \in \Theta} \int_{\mathcal{Z}} p_\theta(z, x) dz, \quad (1)$$

and quantifying the uncertainty in the latent variables through the corresponding posterior  $p_{\theta_*}(z|x) = p_{\theta_*}(z, x)/p_{\theta_*}(x)$ . This framework, which represents a pragmatic compromise between frequentist and Bayesian approaches, is known as the empirical Bayes paradigm [13, 59].

---

\*Equal contribution.

A classical approach for solving the marginal maximum likelihood estimation problem in (1) is the Expectation Maximization (EM) algorithm [22]. This iterative method consists of two steps: an expectation step (E-step) and a maximization step (M-step). In the  $t^{\text{th}}$  iteration, the E-step involves computing the expectation of the log-likelihood with respect to the current posterior distribution  $\mu_t := p_{\theta_t}(\cdot|x)$  of the latent variables, viz,

$$Q_t(\theta) = \int_{\mathcal{Z}} \log \pi_{\theta}(z) \mu_t(z) dz, \quad (\text{E})$$

where  $\pi_{\theta}(z) := p_{\theta}(z, x)$  denotes the joint density of  $z$  and  $x$ , given fixed  $x \in \mathbb{R}^{d_x}$ . Meanwhile, the M-step involves optimizing this quantity with respect to the parameters, namely

$$\theta_{t+1} := \arg \max_{\theta \in \Theta} Q_t(\theta). \quad (\text{M})$$

Under fairly general conditions, this procedure guarantees convergence of the parameters  $\theta_t$  to a stationary point  $\theta_*$  of the marginal likelihood, and convergence of the corresponding posterior approximations  $p_{\theta_k}(\cdot|x)$  to  $p_{\theta_*}(\cdot|x)$  [5, 48, 49, 58, 73]. In many applications of interest, neither the E-step nor the M-step is analytically tractable, in which case it is necessary to use approximations. In particular, the M-step can be approximated using numerical optimization routines [40, 44, 49]. Meanwhile, the E-step can be approximated via Monte-Carlo methods, leading to the so-called Monte Carlo EM (MCEM) algorithm [9, 12, 14, 63, 66, 71], or otherwise a Robbins-Monro type stochastic approximation procedure [39, 60], resulting in the stochastic approximation EM (SAEM) algorithm [21]. In practice, it is often not possible to sample exactly from the current posterior distribution  $p_{\theta_t}(\cdot|x)$ . In this setting, it is standard to use a Markov chain Monte Carlo (MCMC) approximation for the E-step [e.g., 1, 4, 11, 20, 28, 38, 42, 47, 51, 57]. In this paper, we follow a different approach, based on an observation first made in [50], and recently revisited in [38], that the EM algorithm can be viewed as coordinate-descent on a free-energy functional  $\mathcal{F}$  (see Sec. 2.2). Leveraging this perspective, [38] obtain a set of easy-to-implement, particle-based algorithms for optimizing the free energy, and thus for solving (1). A similar approach has since also been studied in [1]. We provide a more detailed discussion of these two works in Section 3.

Both in theory and in practice, the performance of these methods relies on a careful choice of hyperparameters such as the learning rate. In particular, the learning rate must be small enough to ensure stability of the parameter updates, whilst also large enough to guarantee convergence at a reasonable rate [1, Theorem 1]. The situation is additionally complicated by the interdependence between the parameters and the latent variables, as well as the sensitivity of the learning rate to other hyperparameters such as the number of particles. As a result, it is typically necessary to make use of heuristics such as Adagrad [25], Adam [35], or RMSProp [67], or otherwise to resort to computationally expensive quasi-Newton updates [38, App. C] which exploit a second order approximation of the log-likelihood to better capture its local geometry. In this context, it is natural to ask whether we can obtain alternative algorithms which are more robust to different specifications of the learning rate, or even remove the dependence on the learning rate entirely.

**Our contributions.** In this paper, we propose two new particle-based algorithms for marginal maximum likelihood estimation in latent variable models, including one which is completely tuning free. Our methods can be applied to a very broad class of latent variable models, namely, any for which the density  $p_{\theta}(z, x)$  is differentiable in  $\theta$  and  $z$ . Inspired by [38], both of our algorithms are rooted in the viewpoint of marginal maximum likelihood estimation as the minimization of the free energy functional. Our first approach, SVGD EM, is motivated by recent developments in the theory and application of gradient flows on the space of probability measures [e.g., 2, 26, 36, 56]. In particular, SVGD EM corresponds to the discretization of a particular gradient flow of the free energy  $\mathcal{F}$  on  $\Theta \times \mathcal{P}_2(\mathcal{X})$ , and resembles a generalization of the popular Stein variational gradient descent (SVGD) algorithm. Meanwhile, our second approach, Coin EM, is inspired by the parameter-free stochastic optimization methods developed by Orabona and coworkers [19, 53, 55], and their recent extension to the space of probability measures in [62]. In particular, Coin EM leverages a reduction of the minimization of the free energy to two coin betting games. Unlike our first algorithm, or the recent schemes proposed in [1, 38], Coin EM does not correspond to the time-discretization of any gradient flow, and has no learning rates. It thus bears little resemblance to existing particle-based methods for training latent variable models.

After introducing our algorithms, we study the convergence properties of SVGD EM, establishing a descent lemma which guarantees that this algorithm decreases the free energy at each iteration. We

then illustrate the performance of Coin EM and SVGD EM on a wide range of examples, including a toy hierarchical model, two Bayesian logistic regression models, two Bayesian neural networks, and a latent space network model. Our results indicate that SVGD EM and Coin EM achieve comparable or superior performance to existing particle-based EM algorithms, with Coin EM removing entirely the need to tune any hyperparameters.

## 2 Maximum Likelihood Training of Latent Variable Models

### 2.1 Notation

We will make use of the following notation. Let  $\mathcal{P}_2(\mathcal{Z})$  denote the set of probability measures with finite second moment. Given  $\mu \in \mathcal{P}_2(\mathcal{Z})$ , let  $L^2(\mu)$  denote the space of functions  $f : \mathcal{Z} \rightarrow \mathbb{R}$  such that  $\int \|f\|^2 d\mu < \infty$ . We write  $\|\cdot\|_{L^2(\mu)}$  and  $\langle \cdot, \cdot \rangle_{L^2(\mu)}$  for the norm and inner product of this space. Given a probability measure  $\mu \in \mathcal{P}_2(\mathcal{Z})$  and a measurable function  $T : \mathbb{R}^d \rightarrow \mathbb{R}^d$ , we write  $T_{\#}\mu$  for the pushforward measure of  $\mu$  under  $T$ . For  $\mu, \nu \in \mathcal{P}_2(\mathcal{Z})$ , the quadratic Wasserstein distance between  $\mu$  and  $\nu$  is defined by  $W_2^2(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{\mathcal{Z} \times \mathcal{Z}} \|z_1 - z_2\|^2 \gamma(dz_1, dz_2)$ , where  $\Gamma(\mu, \nu)$  denotes the set of couplings between  $\mu$  and  $\nu$ . We refer to the metric space  $(\mathcal{P}_2(\mathcal{Z}), W_2)$  as the Wasserstein space.

We will later also write  $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  for a positive semi-definite kernel,  $\mathcal{H}_k$  for the reproducing kernel Hilbert space (RKHS) associated with this kernel, and  $\mathcal{H} := \mathcal{H}_k^{d_z}$  for the product RKHS consisting of elements  $f = (f_1, \dots, f_{d_z})$ , with  $f_i \in \mathcal{H}_k$ . We also write  $S_\mu : L^2(\mu) \rightarrow \mathcal{H}$  for the integral operator associated with  $k$  and the measure  $\mu$ , defined according to  $S_\mu f(z) = \int_{\mathbb{R}^d} k(z, w) f(w) \mu(dw)$ . Finally, we write  $P_\mu : L^2(\mu) \rightarrow L^2(\mu)$  for the operator  $P_\mu = \iota S_\mu$ , where  $\iota : \mathcal{H} \rightarrow L^2(\mu)$  is the inclusion map, with adjoint  $\iota^* = S_\mu$ . This map differs from  $S_\mu$  only in its range.

### 2.2 The Free Energy

As outlined in Section 1, our approach will leverage the connection between marginal maximum likelihood estimation and the optimization of the free-energy functional  $\mathcal{F} : \Theta \times \mathcal{P}(\mathcal{Z}) \rightarrow \mathbb{R}$ , defined according to

$$\mathcal{F}(\theta, \mu) := \int \log(\mu(z)) \mu(z) dz - \int \log(\pi_\theta(z)) \mu(z) dz. \quad (2)$$

To be precise, we build on the observation made in [50], and recently revisited in [38], that finding  $\theta_* = \arg \max_{\theta \in \Theta} p_\theta(x)$  and computing the corresponding posterior  $\mu_* = p_{\theta_*}(\cdot|x)$  is equivalent to solving the joint minimization problem

$$(\theta_*, \mu_*) = \arg \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu). \quad (3)$$

In particular, as noted in [50], the EM algorithm corresponds precisely to a coordinate descent scheme applied to  $\mathcal{F}$ : given some initial  $\theta_0 \in \Theta$ , solve

$$\mu_t := \arg \min_{\mu \in \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta_t, \mu) \quad (\text{E})$$

$$\theta_{t+1} := \arg \min_{\theta \in \Theta} \mathcal{F}(\theta, \mu_t), \quad (\text{M})$$

until convergence. Despite the simplicity of this approach, its practical use is limited for more complex models. In particular, it can only be applied when it is possible to solve the two optimization subroutines (i.e., compute the E-step and the M-step) exactly.

### 2.3 SVGD EM: Minimizing the Free Energy using a Stein-Variational Gradient Flow

As pointed out in [38], instead of using (E) and (M) in order to solve (2), a natural alternative is to use a discretization of a gradient flow associated with (2). We are then faced with several questions. First, what is an appropriate notion of the gradient flow of the functional  $\mathcal{F}(\theta, \mu)$ ? Second, how should we discretize this flow? Regarding the first question, a natural way in which to construct a gradient flow for  $\mathcal{F}(\theta, \mu)$  is to consider a Euclidean gradient flow w.r.t. the first argument, and a Wasserstein gradient flow w.r.t. the second argument [e.g., 2, Chapter 11]. In particular, we will say that  $(\theta, \mu) : [0, \infty) \rightarrow \Theta \times \mathcal{P}_2(\mathcal{Z})$  is a solution of a ‘Euclidean-Wasserstein’ gradient flow of  $\mathcal{F}$  if,

$$\partial_t \theta_t = -\nabla_\theta \mathcal{F}(\theta_t, \mu_t), \quad \partial_t \mu_t = -\nabla_\mu \mathcal{F}(\theta_t, \mu_t), \quad (4)$$

---

**Algorithm 1** SVGD EM

---

**Input:** number of iterations  $T$ , number of particles  $N$ , initial particles  $\{z_0^i\}_{i=1}^N \sim \mu_0$ , initial  $\theta_0$ , target density  $\pi$ , kernel  $k$ , learning rate  $\gamma$ .  
**for**  $t = 0, 1, \dots, T - 1$  **do**

$$\begin{aligned}\theta_{t+1} &= \theta_t + \frac{\gamma}{N} \sum_{j=1}^N \nabla_{\theta} \log \pi_{\theta_t}(z_t^j) \\ z_{t+1}^i &= z_t^i + \frac{\gamma}{N} \sum_{j=1}^N \left[ k(z_t^j, z_t^i) \nabla_z \log \pi_{\theta_{t+1}}(z_t^j) + \nabla_{z_t^j} k(z_t^j, z_t^i) \right], \quad i \in [N]\end{aligned}$$

**end for**  
**return**  $\theta_T$  and  $\{z_T^i\}_{i=1}^N$ .

---

where  $\nabla_{\theta} \mathcal{F}(\theta, \mu)$  is the standard Euclidean gradient of  $\mathcal{F}(\cdot, \mu)$  at  $\theta$ , and where  $\nabla_{\mu} \mathcal{F}(\theta, \mu) = -\nabla \cdot (\mu \nabla_{W_2} \mathcal{F}(\theta, \mu))$ , with  $\nabla_{W_2} \mathcal{F}(\mu, \theta)$  denoting the Wasserstein gradient of  $\mathcal{F}(\theta, \cdot)$  at  $\mu$ , which exists and is given by  $\nabla_{W_2} \mathcal{F}(\mu, \theta) = \nabla_z \log(\frac{\mu}{\pi_{\theta}})$  under mild regularity conditions on  $\mu \in \mathcal{P}_2(\mathcal{Z})$  [e.g. 2, Lemma 10.4.13]. Explicitly, the gradients in (4) are thus given by

$$\nabla_{\theta} \mathcal{F}(\theta, \mu) = -\int \nabla_{\theta} \log \pi_{\theta}(z) \mu(z) dz, \quad \nabla_{\mu} \mathcal{F}(\theta, \mu) = -\nabla \cdot \left( \mu \nabla_z \log \left( \frac{\mu}{\pi_{\theta}} \right) \right). \quad (5)$$

To obtain an implementable discrete-time algorithm, an obvious choice is to consider an explicit Euler discretization of (4) which, for  $t \in \mathbb{N}_0$ , corresponds to<sup>1</sup>

$$\theta_{t+1} = \theta_t + \gamma \int \nabla_{\theta} \log \pi_{\theta_t}(z) \mu_t(z) dz, \quad \mu_{t+1} = \left( \text{id} - \gamma \nabla_z \log \left( \frac{\mu_t}{\pi_{\theta_{t+1}}} \right) \right)_{\#} \mu_t, \quad (6)$$

where  $\gamma > 0$  is a step size or learning rate, and  $\text{id}$  is the identity map. Unfortunately, implementing this scheme would require estimating the density of  $\mu_t$  based on samples, which is rather challenging. Inspired by Stein variational gradient descent (SVGD) [45], suppose that we replace the Wasserstein gradient  $\nabla_{W_2} \mathcal{F}(\theta_t, \mu_t)$  by its image  $P_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t)$ , under the integral operator  $P_{\mu_t}$ . This essentially plays the role of the Wasserstein gradient in the RKHS  $\mathcal{H}_k$ . Then, crucially, under the assumption that  $\lim_{\|z\| \rightarrow \infty} k(z, \cdot) \pi(z) = 0$ , one can show using integration by parts [e.g., 45] that

$$P_{\mu} \nabla_z \log \left( \frac{\mu}{\pi_{\theta}} \right) (\cdot) = -\int [\nabla_z \log \pi_{\theta}(z) k(z, \cdot) + \nabla_z k(z, \cdot)] \mu(z) dz, \quad (7)$$

which can easily be approximated using samples from  $\mu$ . Using this result, our algorithm now reads

$$\theta_{t+1} = \theta_t + \gamma \int \nabla_{\theta} \log \pi_{\theta_t}(z) d\mu_t(z) \quad (8)$$

$$\mu_{t+1} = \left( \text{id} + \gamma \int [\nabla_z \log \pi_{\theta_{t+1}}(z) k(z, \cdot) + \nabla_z k(z, \cdot)] \mu_t(z) dz \right)_{\#} \mu_t. \quad (9)$$

Finally, we can approximate the two intractable integrals using a set of  $N$  interacting particles,  $\{z_t^i\}_{i=1}^N$ . Based on this approximation, we arrive at Alg. 1.

Naturally, we would like to establish the convergence of this algorithm to the minimizer of  $\mathcal{F}(\theta, \mu)$ . In App. A.1, we establish that  $\mathcal{F}(\theta_t, \mu_t)$  converges exponentially fast along the continuous-time SVGD EM dynamics, under a fairly natural gradient dominance condition. Here, we focus on the discrete-time case. We will require the following assumptions, which extend those introduced in [36].

**Assumption 1.** *There exists  $B > 0$  such that, for all  $z \in \mathcal{Z}$ ,  $\|k(z, \cdot)\|_{\mathcal{H}_k}$  and  $\|\nabla_z k(z, \cdot)\|_{\mathcal{H}} \leq B$ .*

**Assumption 2.** *For all  $z \in \mathcal{P}(\mathcal{Z})$ , the Hessian  $H_{V_z}$  of  $V_z = -\log \pi_{(\cdot)}(z)$  is well defined, and there exists  $M_1 > 0$  such that  $\|H_{V_z}\|_{\text{op}} \leq M_1$ . In addition, for all  $\theta \in \Theta$ , the Hessian  $H_{V_{\theta}}$  of  $V_{\theta} = -\log \pi_{\theta}(\cdot)$  is well-defined, and there exists  $M_2 > 0$  such that  $\|H_{V_{\theta}}\|_{\text{op}} \leq M_2$ .*

**Assumption 3.** *For all  $\theta \in \Theta$ , there exists  $C > 0$  such that  $\|S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta, \mu_t)\|_{\mathcal{H}}^2 \leq C$  for all  $t \in \mathbb{N}_0$ .*

**Theorem 1.** *Assume that Assumptions 1 - 3 hold. Let  $\alpha > 1$ , and suppose that  $\gamma < \frac{\alpha-1}{\alpha B C^2}$ . Then, for all  $t \geq 0$ , the updates in (8) - (9) guarantee*

$$\begin{aligned}\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_t, \mu_t) &\leq -\gamma \left[ \left( 1 - \frac{M_1 \gamma}{2} \right) \|\nabla_{\theta} \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_{\theta}}}^2 \right. \\ &\quad \left. + \left( 1 - \frac{(M_2 + \alpha^2) B^2 \gamma}{2} \right) \|S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}}^2 \right]. \quad (10)\end{aligned}$$

---

<sup>1</sup>In a slight abuse of notation, we use  $t$  to index both time in the continuous-time dynamics, and the iteration in the discrete-time algorithm. The appropriate meaning should always be clear from context.

We prove this Theorem in App. B.2. An immediate consequence of Theorem 1 is the convergence of  $\|\nabla_{\theta}\mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_{\theta}}}^2$  and  $\|S_{\mu_t}\nabla_{W_2}\mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}}^2$  to zero, under appropriate conditions on the learning rate  $\gamma$ . We provide a precise statement of this result in App. A.

Naturally, the convergence of this algorithm depends on the choice of learning rate  $\gamma > 0$ . Indeed, this is necessarily the case for any algorithm obtained as the discretization of a gradient flow associated with the free energy functional, including those recently studied in [1, 38]. In the case that  $\gamma$  is too large, the algorithm is unstable and the parameter and particles will diverge. Meanwhile, if  $\gamma$  is too small, the algorithm will converge slowly. Empirically, [38] noted the challenges of appropriately setting the learning rate parameter simultaneously for the parameters and the latent variables. This is also observed in our empirical results (see Sec. 4). We now address this issue by proposing a learning rate free algorithm for minimizing the free energy functional, which removes entirely the need for user-chosen learning rates, and leads empirically to significantly faster convergence rates.

## 2.4 Coin EM: Minimizing the Free Energy using Coin Sampling

Our approach is based on the learning rate free optimization techniques introduced in [53, 54, 55], and their recent extension to optimization problems on spaces of probability measures [62]. Roughly speaking, our method can be viewed as a hybrid between the coin betting techniques from [53], which we use for the optimization over  $\Theta$ , and one of the coin sampling algorithms (Coin SVGD) from [62], which we use for the optimization over  $\mathcal{P}_2(\mathcal{Z})$ . The resulting algorithm, which we term Coin EM, represents a automatic, general-purpose EM algorithm, which does not require a user-defined learning rate, and outperforms existing algorithms across a range of applications (see Sec. 4).

The coin betting framework [53] involves a gambler making repeated bets on the outcomes of a series of adversarial coin flips. The gambler’s goal is to maximize their wealth, starting from some initial wealth,  $w_0 = 1$ . In each round of the game,  $t \in \mathbb{N}$ , the gambler bets on the outcome of a coin flip, either heads or tails, without borrowing any additional money. The gambler’s bet is encoded by  $z_t \in \mathbb{R}$ , where the sign indicates whether the bet is a heads or tails and the absolute value indicates the size of the bet. The gambler’s wealth thus accumulates according to  $w_t = w_0 + \sum_{s=1}^t c_s z_s$ , where  $c_t \in \{-1, 1\}$  denotes the outcome of the coin flip. The gambler’s bets are restricted to satisfy  $z_t = \beta_t w_{t-1}$ , where  $\beta_t \in [-1, 1]$  denotes a betting fraction, meaning the gambler can only bet a fraction  $\beta_t$  of their accumulated wealth up to time  $t$ , and cannot borrow any additional money.

In [53], the authors demonstrated how this coin betting game could be used to solve convex optimization problems of the form  $z_* = \arg \min_{z \in \mathbb{R}^d} f(z)$ , for some convex function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$ . In particular, by considering a game with outcomes  $c_t = -\nabla f(z_t)$ , and replacing scalar multiplications with scalar products in the framework described above, [53] proved that, under certain assumptions on the betting strategy,  $f(\frac{1}{t} \sum_{s=1}^t z_s) \rightarrow f(z_*)$  at a rate determined by this strategy. In the case that  $|c_t| \leq 1$ ,<sup>2</sup> a standard choice for the betting fraction is  $\beta_t = \frac{1}{t} \sum_{s=1}^{t-1} c_s$ , known as the Krichevsky-Trofimov (KT) betting strategy after [37]. This choice results in the sequence of bets

$$z_t = \beta_t w_{t-1} = \frac{\sum_{s=1}^{t-1} c_s}{t} \left(1 + \sum_{s=1}^{t-1} \langle c_s, z_s \rangle\right). \quad (11)$$

Recently, [62] extended this approach to optimization problems on the space of probability measures, thus obtaining learning-rate free algorithms which could be used for sampling problems. In this setting, several modifications are required. First, in round  $t$ , one now bets  $z_t - z_0$ , rather than  $z_t$ , where  $z_0$  is distributed according to some initial betting distribution  $\mu_0 \in \mathcal{P}_2(\mathcal{Z})$ . In this case, viewing  $z_t : \mathcal{Z} \rightarrow \mathcal{Z}$  as a function that maps  $z_0 \mapsto z_t(z_0)$ , one can define the betting distribution  $\mu_t \in \mathcal{P}_2(\mathcal{Z})$  as the push-forward of  $\mu_0$  under the function  $z_t$ . This definition implies, in particular, that if  $z_0 \sim \mu_0$ , then  $z_t := z_t(z_0)$  is distributed according to the betting distribution  $\mu_t$ .

Using this approach, [62] showed that one can solve optimization problems of the form  $\mu_* = \arg \min_{\mu \in \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\mu)$ . In particular, by setting  $\mathcal{F}(\mu) = \text{KL}(\mu|\pi)$ , constructing a betting game in which the outcomes are given by the kernelized Wasserstein gradients  $c_t = -\mathcal{P}_{\mu_t} \nabla_{W_2} \mathcal{F}(\mu_t)(z_t)$ , and approximating  $(c_t)_{t \in \mathbb{N}}$  using a set of  $N$  particles in the corresponding sequence of bets, [62] obtained a learning-rate free analogue of the SVGD, known as Coin SVGD. Empirically, this approach has demonstrated competitive performance with SVGD, with no need to tune a learning rate.

<sup>2</sup>If, instead,  $|c_t| \leq L$ , for some constant  $L > 0$ , then one can replace  $c_t$  by its normalized version. If, more commonly, such a constant is unknown, then one can replace it by an empirical estimate, which is updated as the betting game progresses. We provide details on this approach, and its application to our setting, in App. D.

---

**Algorithm 2** Coin EM

---

**Input:** number of iterations  $T$ , number of particles  $N$ , initial particles  $\{z_0^i\}_{i=1}^N \sim \mu_0$ , initial  $\theta_0$ , target density  $\pi$ , kernel  $k$ .

**for**  $t = 1, \dots, T$  **do**

$$\theta_t = \theta_0 + \frac{\sum_{s=1}^{t-1} \frac{1}{N} \sum_{j=1}^N \nabla_{\theta} \log \pi_{\theta_s}(z_s^j)}{t} \left( 1 + \sum_{s=1}^{t-1} \left\langle \frac{1}{N} \sum_{j=1}^N \nabla_{\theta} \log \pi_{\theta_s}(z_s^j), \theta_s - \theta_0 \right\rangle \right)$$

$$z_n^i = z_0^i + \frac{\sum_{s=1}^{t-1} \frac{1}{N} \sum_{j=1}^N k(z_s^j, z_s^i) \nabla_z \log \pi_{\theta_s}(z_s^j) + \nabla_{z_s^j} k(z_s^j, z_s^i)}{t} \\ \times \left( 1 + \sum_{s=1}^{t-1} \left\langle \frac{1}{N} \sum_{j=1}^N k(z_s^j, z_s^i) \nabla_z \log \pi_{\theta_s}(z_s^j) + \nabla_{z_s^j} k(z_s^j, z_s^i), z_s^i - z_0^i \right\rangle \right), \quad i \in [N]$$

**end for**

**return**  $\theta_T$  and  $\{z_T^i\}_{i=1}^N$ .

---

By combining the original coin betting algorithm in [53, 54] to optimize  $\mathcal{F}(\theta, \mu)$  over  $\theta \in \Theta$ , and the Coin SVGD algorithm in [62] to optimize over  $\mu \in \mathcal{P}_2(\mathcal{Z})$ , we now have a learning rate free method for solving (2). In particular, initialized at some  $\theta_0 \in \Theta$  and  $z_0^i \stackrel{i.i.d.}{\sim} \mu_0$ , we update, for  $t \in \mathbb{N}_0$ ,

$$\theta_t = \theta_0 - \frac{\sum_{s=1}^{t-1} \nabla_{\theta} \mathcal{F}(\theta_s, \mu_s^N)}{t} \left( 1 - \sum_{s=1}^{t-1} \langle \nabla_{\theta} \mathcal{F}(\theta_s, \mu_s^N), \theta_s - \theta_0 \rangle \right), \quad (12)$$

$$z_t^i = z_0^i - \frac{\sum_{s=1}^{t-1} \mathcal{P}_{\mu_s^N} \nabla_{W_2} \mathcal{F}(\mu_s^N)(z_s^i)}{t} \left( 1 - \sum_{s=1}^{t-1} \langle \mathcal{P}_{\mu_s^N} \nabla_{W_2} \mathcal{F}(\theta_s, \mu_s^N)(z_s^i), z_s^i - z_0^i \rangle \right), \quad (13)$$

where  $\mu_t^N = N^{-1} \sum_{j=1}^N \delta_{z_t^j}$  denotes the empirical measure of the interacting particles. We will refer this approach, which is summarized in full in Alg. 2, as Coin EM.

### 3 Related Work

**Comparison with Kuntz et al. [38] and Akyildiz et al. [1].** In a recent paper, [38] revisited the perspective of marginal maximum likelihood estimation as the minimization of the free energy functional, and proposed the gradient flow in (4) to minimize this functional. In contrast to us, their resulting algorithms are based on the observation that (4) is a mean-field Fokker-Planck equation satisfied by the law of the McKean-Vlasov SDE

$$d\theta_t = \left[ \int \nabla_{\theta} \log \pi_{\theta_t}(z_t) d\mu_t(z) \right] dt, \quad dz_t = \nabla_z \log \pi_{\theta_t}(z_t) dt + \sqrt{2} dw_t, \quad (14)$$

where  $w = (w_t)_{t \geq 0}$  is a standard  $d_z$ -dimensional Brownian motion. By approximating this SDE using a system of interacting particles  $\{z_t^j\}_{j=1}^N$ , and discretizing in time, [38] obtain the particle gradient descent (PGD) algorithm. [1] have since analyzed an extension of this algorithm, which includes a carefully chosen noise term in the  $\theta$  dynamics in (14), allowing them to obtain a non-asymptotic concentration bound for  $\theta_t$ . [38] also develop two variants of PGD. The first is the particle quasi-Newton (PQN) algorithm, which includes a preconditioning term in the  $\theta$  dynamics [38, App. C]. In principle, one could obtain a similar version of SVGD EM based on the techniques in [23]. The second is the particle marginal gradient descent (PMGD) algorithm, in which the  $\theta$  update in (14) is replaced by  $\theta_t = \theta_*(\mu_t^N)$ , where  $\theta_*(\mu) = \arg \min_{\theta \in \Theta} \mathcal{F}(\theta, \mu)$ , which can be applied whenever it is possible to compute  $\mu \mapsto \theta_*(\mu)$  in closed form [38, App. D]. In a similar vein, we can obtain marginal variants SVGD EM (Alg. 1) and Coin EM (Alg. 2). These are given in App. C.

**Learning-rate free methods for optimization and sampling.** The idea of using coin betting for parameter free online learning was first introduced in [53, 54] and has since been extensively developed by Orabona and coworkers [3, 16, 17, 19, 32, 33, 55]. Meanwhile, other than the recent work of [61, 62], the literature on learning-rate free methods for (gradient-based) Bayesian inference is non-existent. In practice, the standard technique when using methods such as stochastic gradient Langevin dynamics (SGLD) is to use a grid search, running the algorithm of choice for multiple learning rates, and selecting the value which minimizes an appropriately chosen metric. While there have been some efforts to semi-automate the design of effective learning rate schedules [15, 18, 34, 43], typically these approaches still rely on an appropriate choice of certain hyperparameters.



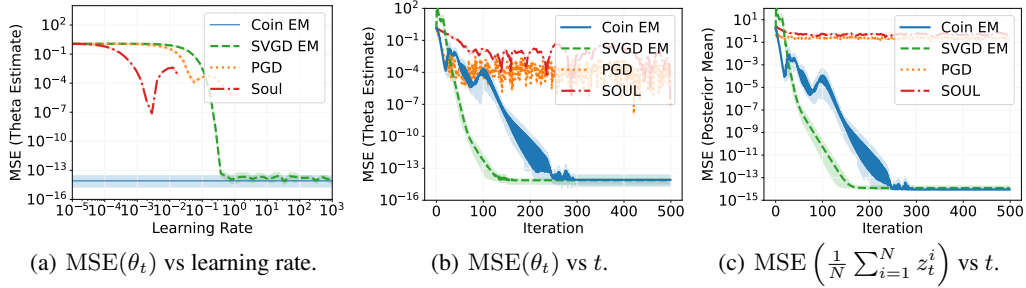


Figure 1: **Results for the toy hierarchical model.** MSE of the parameter estimate  $\theta_t$  as a function of the learning rate after  $T = 500$  iterations (a); and MSE of the parameter estimate (b) and the posterior mean (c) as a function of the number of iterations, using the optimal learning rate from (a).

## 4 Numerical Experiments

We now evaluate the performance of SVGD EM (Alg. 1) and Coin EM (Alg. 2) against other recent approaches in the literature.

### 4.1 Toy hierarchical model

We begin by considering a toy hierarchical model introduced in [38]. In particular, suppose that we observe data  $x = (x_1, \dots, x_{d_z})^\top$  generated according to  $x_i | z_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(x_i | z_i, 1)$ , where the latent variables  $z_i \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(\theta, 1)$ , for some real valued parameter  $\theta \in \mathbb{R}$ . Our model is thus given by  $p_\theta(z, x) = \prod_{i=1}^{d_z} \frac{1}{2\pi} \exp[-(z_i - \theta)^2/2 - (x_i - z_i)^2/2]$ . In this case, the marginal likelihood  $\theta \mapsto p_\theta(x)$  has a unique maximum given by  $\theta_* = d_z^{-1} \sum_{i=1}^{d_z} x_i$ , and one can obtain an explicit expression for the corresponding posterior  $p_{\theta_*}(\cdot | x)$  [38, App. E.1].

In Fig. 1, we evaluate the performance of SVGD EM and Coin EM on this model, setting  $d_z = 100$  and  $\theta = 1$ . We also include results for PGD [38] and the stochastic optimization via unadjusted Langevin (SOUL) algorithm [20]. In this case, both of our methods generate parameters  $\theta_t$  which converge rapidly to  $\theta_*$ , and particles  $(z_t^i)_{i=1}^N$  whose mean converges to the corresponding posterior mean. Even in this toy example, it is clear that PGD, SOUL, and to a lesser extent SVGD EM, are very sensitive to the learning rate (Fig. 1(a)). If the learning rate is too small, then convergence is slow; if it is too large, then the parameter estimates are unstable and may fail to converge (Fig. 8 in App. F.1). Coin EM circumvents this problem entirely, obtaining comparable or superior performance to the competing methods, with no need to tune a learning rate. In Fig. 2, we further investigate the performance of our methods, plotting the posterior variance estimates from Coin EM and SVGD EM in the case  $d_z = 1$ . Here, we see there is a significant bias when using a small number of particles. This should not be a surprise: even if the parameters were fixed, the SVGD updates in Alg. 1 only converge to the true posterior in the joint continuous time and mean-field limit [e.g. 26]. Nonetheless, as is evident in Fig. 2, this bias can be all but eliminated by taking a sufficiently large number of particles. Additional results, further illustrating the robustness of our method to changes in the number of particles (Fig. 6, Fig. 7) and the initialization (Fig. 9), can be found in App. F.1.

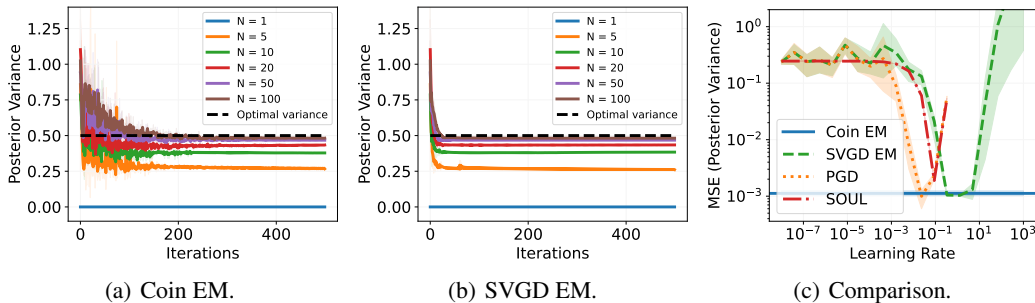


Figure 2: **Additional results for the toy hierarchical model.** Estimates for the posterior variance in the case  $d_z = 1$  obtained using (a) Coin EM and (b) SVGD EM, as a function of the number of iterations. In (c), we plot the MSE of the posterior variance estimate as a function of the learning rate, for Coin EM, SVGD EM, PGD, and SOUL, after  $T = 250$  iterations and with  $N = 50$  particles.

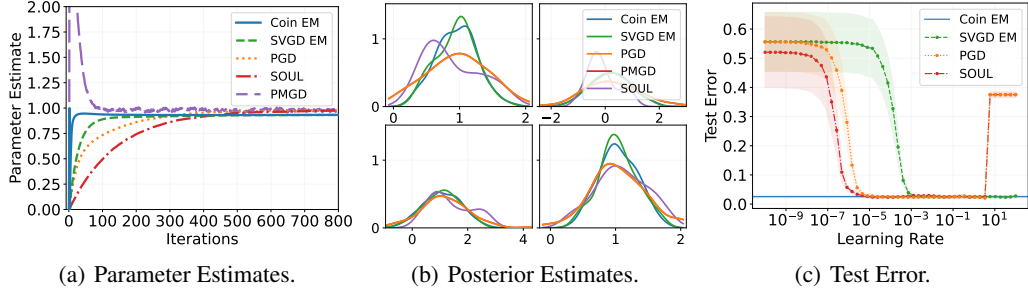


Figure 3: **Results for the Bayesian logistic regression.** Plots of (a) the sequence of parameter estimates  $\theta_t$  initialized at zero, (b) the kernel density estimate of four components of the posterior approximation  $\hat{\mu}_{800}^n = \frac{1}{n} \sum_{j=1}^n \delta_{z_{800}^j}$ , (c) the test error as a function of the learning rate.

## 4.2 Bayesian logistic regression

We next consider a standard Bayesian logistic regression with Gaussian priors, fit using the Wisconsin dataset [72]; see also [20, Sec. 4.1]. In this case, the latent variables are the regression weights. We place an isotropic Gaussian prior  $\mathcal{N}(\theta \mathbf{1}_{d_z}, 51d_z)$  on the weights, and aim to estimate the unique maximizer  $\theta_*$  of the marginal likelihood. We provide full details on this model in App. E.2, and additional results for an alternative Bayesian logistic regression model in App. F.3.

In Fig. 3, we compare the performance of our algorithms with PGD [38], PMGD [38], and SOUL [20]. We first plot an illustrative sequence of parameter estimates (Fig. 3(a)) for each method, initialized at zero, using  $N = 100$  particles and  $T = 800$  iterations. All methods converge to approximately the same parameter  $\theta_*$ . In this case, Coin EM converges noticeably faster than its competitors. The same is true when the parameter is initialized further from  $\theta_*$ , in which case the shorter transient period exhibited by Coin EM is even more pronounced (Fig. 10 in App. F.2). By increasing the learning rates of PGD, PMGD, and SOUL, one can improve their convergence rates, but this comes at the cost of a (significant) bias in the resulting parameter estimate (Fig. 11 in App. F.2). Meanwhile, the posterior estimates obtained by each method are generally rather similar, as is their predictive performance. In fact, in this case the predictive power of the posterior approximations obtained by all of the methods are robust both to the choice of learning rate (Fig. 3(c)) and to the number of particles (see Fig. 12 in App. F.2). This can be largely attributed to the relatively simple nature of the posterior, which is both peaked and unimodal; see also [38, Sec. 3.1] and [20, Sec. 4.1].

## 4.3 Bayesian neural network

We now consider an example with a notably more complex posterior, namely, a Bayesian neural network. We consider a similar setting to the one described in [30, 45], and apply a single layer neural network to perform regression on several UCI benchmark datasets. We assume a Gaussian likelihood with precision  $\gamma$ , and assign a Gaussian prior with precision  $\lambda$  on each of the network weights. We then place a Gamma prior on  $\gamma$ , and a Gamma hyperprior on  $\lambda$ . The latent variables are the weights  $w$ , the precisions  $\lambda$ , and  $\gamma$ . Meanwhile, the parameters are the hyperparameters of the Gamma prior on  $\gamma$  and the Gamma hyperprior on  $\lambda$ . We provide full details of this model in App. E.4.

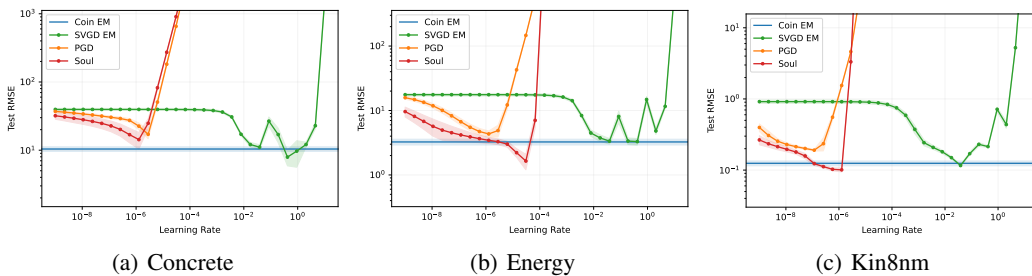


Figure 4: **Results for the Bayesian neural network.** Root mean-squared-error (RMSE) as a function of the learning rate, for several UCI datasets, averaged over ten random test-train splits.



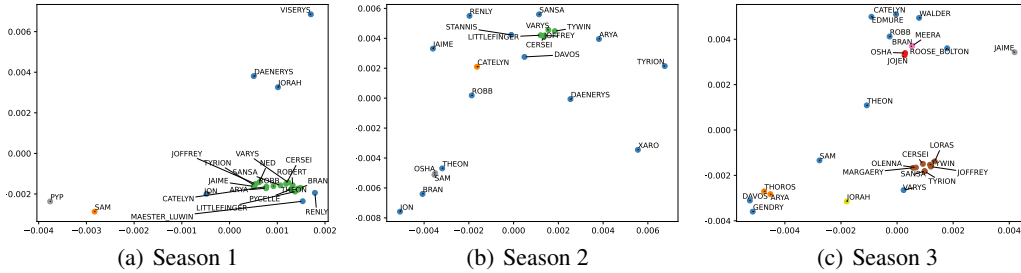


Figure 5: **Results for the latent space network model.** Mean of the particles  $\{z_T^i\}_{i=1}^N$  output by Coin EM after  $T = 500$  iterations. Each node of the network represents a Game of Thrones character.

In Fig. 4 and Fig. 14 (App. F.4), we plot the test error as a function of the learning rate for several UCI benchmark datasets. Here, the optimal predictive performance of Coin EM is comparable with the optimal predictive performance of SVGD EM and SOUL. Meanwhile, Coin EM, SVGD EM, and SOUL all tend to outperform PGD. In this case, the predictive performance is much more sensitive to the choice of learning rate (see Fig. 4), and the other methods only achieve comparable (or superior) performance to Coin EM for a narrow range of learning rates. In App. F.5, we provide results for another Bayesian neural network model used on an MNIST classification task. Here, Coin EM offers competitive predictive performance with PGD and SOUL, while also providing increased robustness to changes in the number of particles (Fig. 16 in App. F.5), the need for additional heuristics to deal with ill-conditioning (Fig. 17 in App. F.5), and, of course, the learning rate (Fig. 15 in App. F.5).

#### 4.4 Latent space network model

Finally, we consider a latent space network model [31, 46, 68]. Such models assume that each node in a network has an unobserved latent position in a low-dimensional Euclidean space, and that the probability of a link between two nodes  $i$  and  $j$  depends on the distance between their latent variables  $\|z_{(i)} - z_{(j)}\|$ . Latent space network models can account for transitivity, homophily, and other network properties, and provide a natural way to analyze social network data. In particular, the estimated latent positions can be used to visualize the network, and also for downstream machine learning tasks such as node classification or community detection through clustering.

Here, we consider fitting a latent space network model to a dataset curated by [6], which consists of a sequence of binary undirected networks indicating whether or not an interaction occurred between two characters in the TV series *Game of Thrones*, with one network for each of the eight seasons (see App. E.6 for further details). In Fig. 5, we plot the latent representation of the character interactions obtained using Coin EM. Plots for the other algorithms are given in App. F.6. In this case, we find that the latent representations obtained via Coin EM successfully capture the natural groupings of the characters, and how these groupings evolve through the first three seasons. Meanwhile, this is not the case for PGD (Fig. 20 in App. F.6) or SOUL (Fig. 21 in App. F.6).

## 5 Discussion

**Summary.** We introduce two new particle-based algorithms for marginal maximum likelihood estimation and empirical Bayesian inference in latent variable models, including one which is entirely tuning-free. Our first algorithm, SVGD EM (Sec. 2.3), can be viewed as a particular form of gradient descent on the free-energy functional  $\mathcal{F}$  over the product space  $\Theta \times \mathcal{P}_2(\mathcal{Z})$ . Our second algorithm, Coin EM (Sec. 2.4), is entirely different, and leverages coin betting ideas introduced in [53] and recently extended in [62], to remove any dependence on learning rates.

**Limitations and Future Work.** We highlight several limitations of our methods. First, similar to SVGD, our algorithms have a cost  $O(N^2)$  per update, which prohibits their use with very large numbers of particles. Second, our convergence results for SVGD EM were derived in the population limit. We leave to future work the extension of these results to the finite particle case; here, the results in [36, Sec. 6] and in particular [64] will likely prove a good starting point. Finally, we leave open the problem of establishing convergence and convergence rates for Coin EM, both in the population limit and the finite particle setting.

## Acknowledgments

We are grateful to Juan Kuntz for many insightful discussions. LS and CN were supported by the Engineering and Physical Sciences Research Council (EPSRC), grant number EP/V022636/1. CN acknowledges further support from the EPSRC, grant number EP/R01860X/1. DD was supported by the EPSRC funded STOR-i Centre for Doctoral Training, grant number EP/L015692/1.

## References

- [1] Ö. D. Akyıldız, F. R. Crucinio, M. Girolami, T. Johnston, and S. Sabanis. Interacting Particle Langevin Algorithm for Maximum Marginal Likelihood Estimation. *arXiv preprint*, 2023. 2, 5, 6
- [2] L. Ambrosio, N. Gigli, and Giuseppe Savaré. *Gradient Flows: In Metric Spaces and in the Space of Probability Measures*. Birkhäuser, Basel, 2008. 2, 3, 4
- [3] H. Asi and J. C. Duchi. Stochastic (Approximate) Proximal Point Methods: Convergence, Optimality, and Adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290, 2019. 6
- [4] Y. F. Atchadé, G. Fort, and E. Moulines. On Perturbed Proximal Gradient Algorithms. *Journal of Machine Learning Research*, 18:1–33, 2017. 2
- [5] S. Balakrishnan, M. J. Wainwright, and B. Yu. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77–120, 2017. 2
- [6] A. Beveridge and M. Chemers. The Game of Game of Thrones: Networked Concordances and Fractal Dramaturgy. In *Reading Contemporary Serial Television Universes*, pages 201–225. Routledge, 2018. 9, 26
- [7] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, 2006. 1
- [8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. 1
- [9] J. G. Booth and J. P. Hobert. Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(1):265–285, 1999. 2
- [10] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. 23
- [11] B. S. Caffo, W. Jank, and G. L. Jones. Ascent-based Monte Carlo expectation– maximization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):235–251, 2005. 2
- [12] O. Cappé, A. Doucet, M. Lavielle, and E. Moulines. Simulation-based methods for blind maximum-likelihood filter identification. *Signal Processing*, 73(1):3–25, 1999. 2
- [13] G. Casella. An Introduction to Empirical Bayes Data Analysis. *The American Statistician*, 39(2):83–87, 1985. 1
- [14] K. S. Chan and J. Ledolter. Monte Carlo EM Estimation for Time Series Models Involving Counts. *Journal of the American Statistical Association*, 90(429):242–252, 1995. 2
- [15] C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin. Bridging the Gap between Stochastic Gradient MCMC and Stochastic Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, Cadiz, Spain, 2016. 6
- [16] K. Chen, A. Cutkosky, and F. Orabona. Implicit Parameter-free Online Learning with Truncated Linear Models. In *Proceedings of the 33rd International Conference on Algorithmic Learning Theory (ALT 2022)*, Paris, France, 2022. 6

- [17] K. Chen, J. Langford, and F. Orabona. 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, 2022. 6
- [18] J. Coullon, L. South, and C. Nemeth. Efficient and Generalizable Tuning Strategies for Stochastic Gradient MCMC. *arXiv preprint*, 2021. 6
- [19] A. Cutkosky and F. Orabona. 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, 6
- [20] V. De Bortoli, A. Durmus, M. Pereyra, and A. F. Vidal. Efficient stochastic optimisation by unadjusted Langevin Monte Carlo. *Statistics and Computing*, 31(3):29, 2021. 2, 7, 8, 24, 28, 29
- [21] B. Delyon, M. Lavielle, and E. Moulines. Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94–128, 1999. 2
- [22] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Stat. Soc. B*, 39(1):1–38, 1977. 2
- [23] G. Detommaso, T. Cui, A. Spantini, Y. Marzouk, and R. Scheichl. A Stein variational Newton method. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS 2018)*, Montreal, Canada, 2018. 6
- [24] D. Dua and C. Graff. UCI Machine Learning Repository. Technical report, University of California, Irvine, School of Information and Computer Sciences, 2019. 24, 25
- [25] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. 2, 23, 25, 27
- [26] A. Duncan, N. Nüsken, and L. Szpruch. On the geometry of Stein variational gradient descent. *Journal of Machine Learning Research*, 24:1–40, 2023. 2, 7, 14, 15
- [27] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996. 33
- [28] G. Fort, E. Moulines, and P. Priouret. Convergence of adaptive and interacting Markov chain Monte Carlo algorithms. *The Annals of Statistics*, 39(6):3262–3289, 2011. 2
- [29] S. J. Gershman, M. D. Hoffman, and D. M. Blei. Nonparametric Variational Inference. In *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, UK, 2012. 24
- [30] J. M. Hernandez-Lobato and R. P. Adams. 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, 25
- [31] P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002. 1, 9, 26
- [32] K.-S. Jun and F. Orabona. Parameter-Free Online Convex Optimization with Sub-Exponential Noise. In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT 2019)*, Phoenix, AZ, 2019. 6
- [33] K.-S. Jun, F. Orabona, S. Wright, and R. Willett. Online Learning for Changing Environments using Coin Betting. *Electronic Journal of Statistics*, 11, 2017. 6
- [34] S. Kim, Q. Song, and F. Liang. Stochastic gradient Langevin dynamics with adaptive drifts. *Journal of Statistical Computation and Simulation*, 92(2):318–336, 2022. 6
- [35] D. P. Kingma and J. Ba. Adam: a method for stochastic optimisation. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR '15)*, pages 1–13, San Diego, CA, 2015. 2

- [36] A. Korba, A. Salim, M. Arbel, G. Luise, and A. Gretton. 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, 4, 9, 14, 15, 16, 18, 19, 20, 21
- [37] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–207, 1981. 5
- [38] J. Kuntz, J. N. Lim, and A. M. Johansen. Particle algorithms for maximum likelihood training of latent variable models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS 2023)*, Valencia, Spain, 2023. 2, 3, 5, 6, 7, 8, 14, 16, 19, 24, 25, 26, 29, 30, 32
- [39] H. J. Kushner and D. S. Clark. *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, New York, 1978. 2
- [40] K. Lange. A Gradient Algorithm Locally Equivalent to the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(2):425–437, 1995. 2
- [41] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 25, 26
- [42] R. A. Levine and G. Casella. Implementations of the Monte Carlo EM Algorithm. *Journal of Computational and Graphical Statistics*, 10(3):422–439, 2001. 2
- [43] C. Li, C. Chen, D. Carlson, and L. Carin. Preconditioned Stochastic Gradient Langevin Dynamics for Deep Neural Networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, Phoenix, AZ, 2016. 6
- [44] C. Liu and D. B. Rubin. The ECME Algorithm: A Simple Extension of EM and ECM with Faster Monotone Convergence. *Biometrika*, 81(4):633–648, 1994. 2
- [45] Q. Liu and D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016. 4, 8, 25
- [46] J. D. Loyal and Y. Chen. A Bayesian Nonparametric Latent Space Approach to Modeling Evolving Communities in Dynamic Networks. *Bayesian Analysis*, 18(1):49–77, 2023. 9
- [47] C. E. McCulloch. Maximum Likelihood Algorithms for Generalized Linear Mixed Models. *Journal of the American Statistical Association*, 92(437):162–170, 1997. 2
- [48] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, 2nd edition, 2007. 2
- [49] X.-L. Meng and D. B. Rubin. Maximum Likelihood Estimation via the ECM Algorithm: A General Framework. *Biometrika*, 80(2):267–278, 1993. 2
- [50] R. M. Neal and G. E. Hinton. A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Springer Netherlands, Dordrecht, 1998. 2, 3
- [51] E. Nijkamp, B. Pang, T. Han, S.-C. Zhu, and Y. N. Wu. Learning Multi-layer Latent Variable Model via Variational Optimization of Short Run MCMC for Approximate Inference. In *European Conference on Computer Vision*, pages 361–378, Online, 2020. 2
- [52] F. Orabona and A. Cutkosky. Tutorial on Parameter-Free Online Learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, Online, 2020. 26
- [53] F. Orabona and D. Pal. Coin Betting and Parameter-Free Online Learning. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016. 2, 5, 6, 9

- [54] F. Orabona and D. Pal. Parameter-Free Convex Learning Through Coin Betting. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016): AutoML Workshop*, New York, NY, 2016. 5, 6
- [55] F. Orabona and T. Tommasi. 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, 5, 6, 22
- [56] F. Otto. The Geometry of Dissipative Evolution Equations: The Porous Medium Equation. *Communications in Partial Differential Equations*, 26(1-2):101–174, 2001. 2
- [57] Y. Qiu and X. Wang. Stochastic Approximate Gradient Descent via the Langevin Algorithm. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20)*, New York, NY, 2020. 2
- [58] R. A. Redner and H. F. Walker. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review*, 26(2):195–239, 1984. 2
- [59] H. Robbins. An empirical Bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*, pages 157–164, 1956. 1
- [60] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. 2
- [61] L. Sharrock, L. Mackey, and C. Nemeth. Learning Rate Free Bayesian Inference in Constrained Domains. *arXiv preprint*, 2023. 6
- [62] L. Sharrock and C. Nemeth. Coin Sampling: Gradient-Based Bayesian Inference without Learning Rates. *To appear in Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, 2023. 2, 5, 6, 9, 22
- [63] R. P. Sherman, Y.-Y. K. Ho, and S. R. Dalal. Conditions for convergence of Monte Carlo EM sequences with an application to product diffusion modeling. *The Econometrics Journal*, 2(2):248–267, 1999. 2
- [64] J. Shi and L. Mackey. A Finite-Particle Convergence Rate for Stein Variational Gradient Descent. *arXiv preprint*, 2022. 9
- [65] P. Smaragdis, B. Raj, and M. Shashanka. A Probabilistic Latent Variable Model for Acoustic Modeling. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems: Workshop on Advances in Models for Acoustic Processing (NIPS 2006)*, Vancouver, Canada, 2006. 1
- [66] M. A. Tanner. *Tools for Statistical Inference*. Springer-Verlag, New York, NY, 2nd edition, 1993. 2
- [67] T. Tieleman and G. E. Hinton. 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. 2
- [68] K. Turnbull, S. Lunagómez, C. Nemeth, and E. Airoldi. Latent space modelling of hypergraph data. *arXiv preprint*, 2019. 9, 26
- [69] T. van Erven and P. Harremoës. Rényi Divergence and Kullback-Leibler Divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014. 19
- [70] C. Villani. *Topics in Optimal Transportation*. American Mathematical Society, Providence, Rhode Island, 2003. 20
- [71] G. C. G. Wei and M. A. Tanner. A Monte Carlo Implementation of the EM Algorithm and the Poor Man’s Data Augmentation Algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990. 2
- [72] W. H. Wolberg and O. L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the National Academy of Sciences*, 87(23):9193–9196, 1990. 8, 24



- [73] C. Wu. On the convergence properties of the EM algorithm. *Annals of Statistics*, 11(1):95–103, 1983. 2
- [74] Y. Yao, A. Vehtari, and A. Gelman. Stacking for nonmixing Bayesian computations: The curse and blessing of multimodal posteriors. *Journal of Machine Learning Research*, 23(79):1–45, 2022. 25, 26

## A Additional Theoretical Results

We will require the following additional notation. We will write  $\mathcal{U}$  to denote the Cartesian product of the Euclidean space  $(\Theta, \|\cdot\|_{\mathbb{R}^{d_\theta}})$  and the Hilbert product space  $(\mathcal{H}_k^{d_z}, \|\cdot\|_{\mathcal{H}_k^{d_z}})$ , with norm  $\|\cdot\|_{\mathcal{U}}$  defined according to  $\|(\theta, f)\|_{\mathcal{U}}^2 = \|\theta\|_{\mathbb{R}^{d_\theta}}^2 + \|f\|_{\mathcal{H}_k^{d_z}}^2$  for  $\theta \in \Theta$  and  $f \in \mathcal{H}_k^{d_z}$ . In addition, we will write  $\nabla_{\mathcal{U}}\mathcal{F} = (\nabla_{\theta}\mathcal{F}, S_{\mu}\nabla_{W_2}\mathcal{F}) \in \mathcal{U}$ .

### A.1 Continuous Time Results

In this section, we study the properties of the SVGD EM gradient flow, which we recall is given by

$$\frac{\partial\theta_t}{\partial t} = -\nabla_{\theta}\mathcal{F}(\theta_t, \mu_t), \quad \nabla_{\theta}\mathcal{F}(\theta_t, \mu_t) = -\int \nabla_{\theta} \log \pi_{\theta_t}(z) \mu_t(z) dz, \quad (15)$$

$$\frac{\partial\mu_t}{\partial t} = -\nabla_{\mu}\mathcal{F}(\theta_t, \mu_t), \quad \nabla_{\mu}\mathcal{F}(\theta_t, \mu_t) := -\nabla \cdot \left( \mu_t \left[ P_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t) \right] \right), \quad (16)$$

where  $\nabla_{W_2}\mathcal{F}(\theta_t, \mu_t) = \nabla_z \log \left( \frac{\mu_t}{\pi_{\theta_t}} \right)$  denotes the Wasserstein gradient of  $\mathcal{F}(\theta_t, \cdot)$  at  $\mu_t$ . This gradient flow was obtained by replacing the Wasserstein gradient appearing in (4), (5) with its kernelized version,  $P_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t)$ .

We first provide a result which quantifies the dissipation of the free energy along the trajectory of the continuous-time SVGD EM dynamics.

**Proposition 1.** *The dissipation of the free energy along the SVGD EM gradient flow (15) - (16) is given by*

$$\frac{d\mathcal{F}(\theta_t, \mu_t)}{dt} = -\int \|\nabla_{\theta} \log \pi_{\theta_t}(z)\|_{\mathbb{R}^{d_\theta}}^2 \mu_t(z) dz - \left\| S_{\mu_t} \nabla_z \log \left( \frac{\mu_t}{p_{\theta_t}(\cdot|x)} \right) \right\|_{\mathcal{H}_k^d}^2. \quad (17)$$

*Proof.* See App. B.1. □

**Remark 1.** *We can identify the second term in (17) as the Stein Fisher information of  $\mu_t$  relative to the posterior  $p_{\theta_t}(\cdot|x)$  [26, 36], written*

$$I_{\text{Stein}}(\mu_t | p_{\theta_t}(\cdot|x)) := \left\| S_{\mu_t} \nabla_z \log \left( \frac{\mu_t}{p_{\theta_t}(\cdot|x)} \right) \right\|_{\mathcal{H}_k^d}^2. \quad (18)$$

*This quantity is sometimes also referred to as the squared kernel Stein discrepancy (KSD).*

Since both of the terms on the RHS in (17) are negative, Proposition 1 shows that the free energy decreases along the SVGD EM gradient flow. Under some additional assumption, we can actually establish convergence of the two terms on the RHS to zero. First, we will require the following rather mild regularity condition on the marginal likelihood; see also [38, Assumption 1].

**Assumption 4.** *The super-level sets of the marginal likelihood  $p_{\theta}(x)$  are compact. That is, the set  $\{\theta \in \Theta : p_{\theta}(x) \geq c\}$  is bounded for all  $c \geq 0$ .*

We will also require an additional control on the derivatives of the function  $V_{\theta} : \mathcal{Z} \rightarrow \mathbb{R}$  which maps  $z \mapsto -\log \pi_{\theta}(z)$ , for fixed  $\theta \in \Theta$ , and on the mixed partial derivatives of the function  $V : \Theta \times \mathcal{Z} \rightarrow \mathbb{R}$  which maps  $(\theta, z) \mapsto -\log \pi_{\theta}(z)$

**Assumption 5.** *For all  $\theta \in \Theta$ , the gradient of the function  $V_{\theta}$  grows at most linearly. In particular, there exists  $L > 0$  such that, for all  $\theta \in \Theta$ ,  $\|\nabla_z V_{\theta}(z)\|_{\mathbb{R}^{d_z}} \leq K(1 + \|z\|)$ .*

**Assumption 6.** *The matrix  $H_{V_{\theta, z}} : \mathbb{R}^{d_\theta} \rightarrow \mathbb{R}^{d_\theta}$  containing the mixed partial derivatives of  $V$ , namely  $[H_{V_{\theta, z}}]_{ij} = \partial_{\theta_i} \partial_{z_j} V$ , is well defined. In addition, there exists  $M_3$  such that  $\|H_{\theta, z}\|_{\text{op}} \leq M_3$ .*

**Proposition 2.** Let  $(\theta_t)_{t \geq 0}$  and  $(\mu_t)_{t \geq 0}$  be solutions of the SVGD EM gradient flow (15) - (16). Suppose that Assumptions 1, 2, 4, 5, and 6 hold. In addition, suppose that there exists a positive constant  $C > 0$  such that  $\int \|z\| \mu_t(z) dz$  for all  $t \geq 0$ . Then

$$\lim_{t \rightarrow \infty} \left[ \|\nabla_{\theta} \mathcal{F}(\theta_t, \mu_t)\|^2 + \|\mathcal{S}_{\mu_t} \nabla_z \log \left( \frac{\mu_t}{\pi_{\theta_t}} \right) \|_{\mathcal{H}_k^d}^2 \right] = 0. \quad (19)$$

*Proof.* See App. B.1. □

**Remark 2.** We note that, as an alternative to Assumptions 2 and 6, one could instead assume directly that the Hessian  $H_V$  of  $V$  is well defined, and that there exists a constant  $M$  such that  $\|H_V\|_{\text{op}} \leq M$ . In this case, Assumptions 2 and 6 would certainly hold, taking  $M_1 = M_2 = M_3 = M$ . We separate these two assumptions since for certain results, e.g., Theorem 1, we only require boundedness of the ‘diagonal blocks’ of this Hessian. That is, the Hessian of  $V_z : \Theta \rightarrow \mathbb{R}$  which maps  $\theta \mapsto -\log \pi_{\theta}(z)$  for fixed  $z \in \mathcal{Z}$ , and the Hessian of  $V_{\theta} : \mathcal{Z} \rightarrow \mathbb{R}$  which maps  $z \mapsto -\log \pi_{\theta}(z)$  for fixed  $\theta \in \Theta$ .

In order to establish exponential convergence along the SVGD EM gradient flow, we will require an additional condition, which characterizes the properties of  $\mathcal{F}$  around its equilibria. We assume a particular ‘gradient dominance’ condition, which represents a natural extension of the corresponding conditions used for Euclidean gradients flows - the Polyak-Łojasiewicz inequality - and for the SVGD gradient flow - the Stein log-Sobolev inequality [26, 36].

**Assumption 7.** There exists  $\lambda > 0$  such that  $\mathcal{F}$  satisfies the following gradient dominance

$$\mathcal{F}(\theta, \mu) - \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu) \leq \frac{1}{2\lambda} \|\nabla_{\mathcal{U}} \mathcal{F}\|_{\mathcal{U}}^2. \quad (20)$$

**Proposition 3.** Assume that Assumption 7 holds. Then the free energy  $\mathcal{F}(\theta, \mu)$  decreases exponentially fast along the SVGD EM gradient flow. In particular,

$$\mathcal{F}(\theta_t, \mu_t) - \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu) \leq e^{-2\lambda t} \left[ \mathcal{F}(\theta_0, \mu_0) - \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu) \right]. \quad (21)$$

*Proof.* See App. B.1. □

## A.2 Additional Discrete Time Results

We now consider the forward Euler discretisation of the dynamics in (15) - (16), as given in (8) - (9). For convenience, we recall these update equations again now:

$$\theta_{t+1} = \theta_t + \gamma \int \nabla_{\theta} \log \pi_{\theta_t}(z) d\mu_t(z) \quad (22)$$

$$\mu_{t+1} = \left( \text{id} + \gamma \int [\nabla_z \log \pi_{\theta_{t+1}}(z) k(z, \cdot) + \nabla_z k(z, \cdot)] \mu_t(z) dz \right)_{\#} \mu_t. \quad (23)$$

The updates in (22) - (23) represent the population limit of SVGD EM (Alg. 1). In Theorem 1, we established a descent lemma, guaranteeing that, given a suitable choice of the learning rate  $\gamma$ , the free energy decreases at each iteration of the SVGD EM algorithm. As a corollary to this result, we now obtain a discrete time convergence rate for the average of  $\|\nabla_{\theta} \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_{\theta}}}^2$  and  $\|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2$ . In particular, we have the following result.

**Corollary 1.** Let  $\alpha > 1$ , and  $\gamma \leq \min \left( \frac{\alpha-1}{\alpha B C^{1/2}}, \frac{2}{M_1}, \frac{2}{(M_2 + \alpha^2) B^2} \right)$ . Then, defining  $c_{\gamma} = \gamma \left( 1 - \frac{[M_1 + (M_2 + \alpha^2) B^2] \gamma}{2} \right)$ , the discrete-time, population-limit, SVGD EM updates in (8) - (9) satisfy

$$\min_{t=1, \dots, T} \left( \|\nabla_{\theta} \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_{\theta}}}^2 + \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \right) \quad (24)$$

$$\leq \frac{1}{T} \sum_{t=1}^T \left( \|\nabla_{\theta} \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_{\theta}}}^2 + \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \right) \quad (25)$$

$$\leq \frac{\mathcal{F}(\theta_0, \mu_0) - \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu)}{c_{\gamma} T}. \quad (26)$$

*Proof.* See App. B.3. □

## B Proofs of Theoretical Results

### B.1 Proof of Propositions 1, 2, and 3.

*Proof of Proposition 1.* Using differential calculus in the product space  $\Theta \times \mathcal{P}_2(\mathcal{Z})$ , and the chain rule, we have that

$$\begin{aligned} \frac{d\mathcal{F}(\theta_t, \mu_t)}{dt} &= \langle -\nabla_\theta \mathcal{F}(\theta_t, \mu_t), \nabla_\theta \mathcal{F}(\theta_t, \mu_t) \rangle_{\mathbb{R}^{d_\theta}} + \langle P_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t), \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t) \rangle_{L^2(\mu_t)} \\ &= -\|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 - \|S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \end{aligned} \quad (27)$$

where in the second line we have used the fact that, given  $\mu \in \mathcal{P}_2(\mathcal{Z})$ , and functions  $f, g \in L^2(\mu)$ ,  $\mathcal{H}_k^{d_z}$ , it holds that  $\langle f, \iota g \rangle_{L^2(\mu)} = \langle \iota^* f, g \rangle_{\mathcal{H}_k^{d_z}} = \langle S_\mu f, g \rangle_{\mathcal{H}_k^{d_z}}$ , since the adjoint of the inclusion  $\iota : \mathcal{H} \rightarrow L^2(\mu)$  is  $\iota^* = S_\mu$ . To obtain the first term on the RHS of (17), we can now just substitute the expression for  $\nabla_\theta \mathcal{F}(\theta_t, \mu_t)$  from (15) into the first term on the RHS of (27). For the remaining term, recalling the definition of  $\pi_\theta(z)$ , namely,  $\pi_\theta(z) := p_\theta(z, x) = p_\theta(z|x)p_\theta(x)$ , we have

$$\nabla_z \log \left( \frac{\mu_t}{\pi_{\theta_t}} \right) = \nabla_z \log \left( \frac{\mu_t}{p_{\theta_t}(\cdot|x)p_{\theta_t}(x)} \right) = \nabla_z \log \left( \frac{\mu_t}{p_{\theta_t}(\cdot|x)} \right) \quad (28)$$

where the final equality follows from the fact that  $p_{\theta_t}(x)$  is independent of  $z$ . Substituting this into (27) completes the proof.  $\square$

*Proof of Proposition 2.* We use similar arguments to those in the proofs of [38, Theorem 3] and [36, Proposition 8], adapted appropriately to our setting. For notational convenience, let us define

$$I(\theta_t, \mu_t) := \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 + \|S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2. \quad (29)$$

We start by establishing that, under the stated assumptions, there exists a positive constant  $\beta > 0$  which guarantees that

$$\left| \frac{dI(\theta_t, \mu_t)}{dt} \right| \leq \beta I(\theta_t, \mu_t). \quad (30)$$

We will then show that this implies convergence of  $I(\theta_t, \mu_t)$  to zero. To prove (30), we first compute

$$\frac{dI(\theta_t, \mu_t)}{dt} = \underbrace{\frac{d}{dt} \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2}_{I_t^{(1)}(\theta_t, \mu_t)} + \underbrace{\frac{d}{dt} \|S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2}_{I_t^{(2)}(\theta_t, \mu_t)} \quad (31)$$

We begin with  $I_t^{(1)}(\theta_t, \mu_t)$ . Let us define  $v_t = \nabla_\theta \mathcal{F}(\theta_t, \mu_t) = -\int \nabla_\theta \log \pi_{\theta_t}(z) \mu_t(z) dz$  and  $w_t = S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t) = S_{\mu_t} \nabla \log \left( \frac{\mu_t}{\pi_{\theta_t}} \right)$ . We then have

$$I_t^{(1)}(\theta_t, \mu_t) = \frac{d}{dt} \|v_t\|_{\mathbb{R}^{d_\theta}}^2 = 2 \left\langle v_t, \frac{d}{dt} v_t \right\rangle_{\mathbb{R}^{d_\theta}}. \quad (32)$$

We now need to obtain  $\frac{d}{dt} v_t$ . Using the chain rule, and then integration by parts, we have

$$\frac{d}{dt} v_t = - \int \frac{d}{dt} [\nabla_\theta \log \pi_{\theta_t}(z) \mu_t(z)] dz \quad (33)$$

$$= - \int \frac{d}{dt} [\nabla_\theta \log \pi_{\theta_t}(z)] \mu_t(z) dz - \int \nabla_\theta \log \pi_{\theta_t}(z) \frac{\partial \mu_t}{\partial t}(z) dz \quad (34)$$

$$= - \int \nabla_\theta^2 \log \pi_{\theta_t}(z) \mu_t(z) \frac{\partial \theta_t}{\partial t} dz - \int \nabla_\theta \log \pi_{\theta_t}(z) \nabla_z \cdot (w_t(z) \mu_t(z)) dz \quad (35)$$

$$= \int \nabla_\theta^2 \log \pi_{\theta_t}(z) v_t \mu_t(z) dz + \int \nabla_\theta \nabla_z \log \pi_{\theta_t}(z) w_t(z) \mu_t(z) dz. \quad (36)$$

It follows, in particular, that

$$\begin{aligned} I_t^{(1)}(\theta_t, \mu_t) &= -2 \sum_{i=1}^{d_\theta} \sum_{j=1}^{d_\theta} \int v_t^i [\nabla_\theta^2 \log \pi_{\theta_t}(z)]_{ij} v_t^j \mu_t(z) dz \\ &\quad + 2 \sum_{i=1}^{d_\theta} \sum_{j=1}^{d_z} \int v_t^i [\nabla_\theta \nabla_z \log \pi_{\theta_t}(z)]_{ij} w_t^j(z) \mu_t(z) dz. \end{aligned} \quad (37)$$

For the second term, we will work component-wise. First, using the notation  $w_t = (w_t^1, \dots, w_t^{d_z})$ , we have  $\|w_t\|_{\mathcal{H}_k}^2 = \sum_{j=1}^{d_z} \|w_t^j\|_{\mathcal{H}_k}^2$ . Thus, in particular,

$$I_t^{(2)}(\theta_t, \mu_t) = \frac{d}{dt} \sum_{j=1}^{d_z} \|w_t^j\|_{\mathcal{H}_k}^2 = 2 \sum_{j=1}^{d_z} \langle w_t^j, \frac{d}{dt} w_t^j \rangle_{\mathcal{H}_k}. \quad (38)$$

It remains to compute the components  $\frac{d}{dt} w_t^j$ , for  $j \in [d_z]$ . Starting from the definition, using integration by parts, the chain rule, and then integration by parts again, we have

$$\frac{d}{dt} w_t^j(z) = \frac{d}{dt} \int k(z', z) \partial_{z'_i} \log \left( \frac{\mu_t}{\pi_{\theta_t}} \right) (z') \mu_t(z') dz' \quad (39)$$

$$= - \int \frac{d}{dt} \left[ \left[ \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z) + \partial_{z'_j} k(z', z) \right] \mu_t(z') \right] dz' \quad (40)$$

$$= - \int \langle \nabla_{\theta} \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z), \frac{\partial \theta_t}{\partial t} \rangle_{\mathbb{R}^{d_{\theta}}} \mu_t(z') dz' \\ - \int \left[ \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z) + \partial_{z'_j} k(z', z) \right] \frac{\partial}{\partial t} \mu_t(z') dz' \quad (41)$$

$$= \int \langle \nabla_{\theta} \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z), v_t \rangle_{\mathbb{R}^{d_{\theta}}} \mu_t(z') dz' \\ + \int \langle \nabla_{z'} \left[ \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z) + \partial_{z'_j} k(z', z) \right], w_t(z') \rangle_{\mathbb{R}^{d_z}} \mu_t(z') dz' \quad (42)$$

$$= \int \langle \nabla_{\theta} \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z), v_t \rangle_{\mathbb{R}^{d_{\theta}}} \mu_t(z') dz' \\ + \int \sum_{i=1}^{d_z} \left[ \partial_{z'_i} \partial_{z'_j} \log \pi_{\theta_t}(z') k(z', z) + \partial_{z'_j} \log \pi_{\theta_t}(z') \partial_{z'_i} k(z', z) \right. \\ \left. + \partial_{z'_i} \partial_{z'_j} k(z', z) \right] w_t^i(z') \mu_t(z') dz'. \quad (43)$$

Now, using the fact that each component of  $w_t^j$  is an elements of the RKHS  $\mathcal{H}_k$ , and thus satisfies the reproducing property  $w_t^j(z) = \langle w_t^j, k(z, \cdot) \rangle_{\mathcal{H}_k}$ , we can rewrite the previous display as

$$I_t^{(2)}(\theta_t, \mu_t) = 2 \sum_{i=1}^{d_{\theta}} \sum_{j=1}^{d_z} \int v_t^i [\nabla_{\theta} \nabla_z \log \pi_{\theta_t}(z)]_{ij} w_t^j(z) \mu_t(z) dz \\ + 2 \sum_{i=1}^d \sum_{j=1}^{d_z} \int \left[ \partial_{z'_i} \partial_{z'_j} \log \pi_{\theta_t}(z) w_t^j(z) + \partial_{z'_i} \log \pi_{\theta_t}(z) \partial_{z'_j} w_t^j(z) \right. \\ \left. + \partial_{z'_j} \partial_{z'_i} w_t^j(z) \right] w_t^i(z) \mu_t(z) dz. \quad (44)$$

Substituting (37) and (44) into (31), before once more making use of the reproducing property, we thus have

$$\frac{dI(\theta_t, \mu_t)}{dt} = 2 \sum_{i=1}^{d_{\theta}} \sum_{j=1}^{d_{\theta}} \int v_t^i [-\nabla_{\theta}^2 \log \pi_{\theta_t}(z)]_{ij} v_t^j \mu_t(z) dz \\ + 2 \sum_{i=1}^{d_{\theta}} \sum_{j=1}^{d_z} \int v_t^i [\nabla_{\theta} \nabla_z \log \pi_{\theta_t}(z)]_{ij} w_t^j(z) \mu_t(z) dz \\ + 2 \sum_{i=1}^{d_z} \sum_{j=1}^{d_z} \int \left[ \partial_{z'_i} \partial_{z'_j} \log \pi_{\theta_t}(z) w_t^j(z) + \partial_{z'_i} \log \pi_{\theta_t}(z) \partial_{z'_j} w_t^j(z) \right. \\ \left. + \partial_{z'_j} \partial_{z'_i} w_t^j(z) \right] w_t^i(z) \mu_t(z) dz \quad (45)$$

$$= \sum_{i=1}^{d_{\theta}} \sum_{j=1}^{d_{\theta}} v_t^i A_t^{ij} v_t^j + \sum_{i=1}^{d_{\theta}} \sum_{j=1}^{d_z} v_t^i \langle B_t^{ij}, w_t^j \rangle_{\mathcal{H}_k} + \sum_{i=1}^{d_z} \sum_{j=1}^{d_z} \langle w_t^i, C_t^{ij} w_t^j \rangle_{\mathcal{H}_k}, \quad (46)$$

where in the final line we have defined

$$A_t^{ij} = 2 \int [-\partial_{\theta_i} \partial_{\theta_j} \log \pi_{\theta_t}(z)] \mu_t(dz) \quad (47)$$

$$B_t^{ij} = 2 \int k(z, \cdot) \partial_{\theta_i} \partial_{z_i} \log \pi_{\theta_t}(z) \mu_t(z) dz \quad (48)$$

$$\begin{aligned} C_t^{ij} &= 2 \int k(z, \cdot) \otimes k(z', \cdot) \partial_{z_i} \partial_{z_j} \log \pi(z) \mu_t(z) \mu_t(z') dz dz' \\ &\quad + 2 \int \partial_{z_i} k(z, \cdot) \otimes k(z', \cdot) \partial_{z_i} \log \pi_{\theta_t}(z) \mu_t(z) dz \mu_t(z') dz' \\ &\quad + 2 \int \partial_{z_i} k(z, \cdot) \otimes \partial_{z_j} k(z', \cdot) \mu_t(z) \mu_t(z') dz dz'. \end{aligned} \quad (49)$$

We now need to bound each of these terms in an appropriate sense. In particular, if we can show that  $\|A_t^{ij}\|_{\text{op}} \leq D_1$ ,  $\|B_t^{ij}\|_{\mathcal{H}_k} \leq D_2$ , and  $\|C_t^{ij}\|_{\text{HS}} \leq D_3$ , for all  $t \geq 0$ , where  $\|\cdot\|_{\text{HS}}$  denotes the Hilbert-Schmidt norm, then it follows immediately that

$$\begin{aligned} \left| \frac{dI(\theta_t, \mu_t)}{dt} \right| &\leq D_1 \sum_{i=1}^{d_\theta} \|v_t^i\|^2 + D_2 d_\theta^{\frac{1}{2}} d_z^{\frac{1}{2}} \left( \sum_{i=1}^{d_\theta} \|v_t^i\|^2 \right)^{\frac{1}{2}} \left( \sum_{i=1}^{d_z} \|w_t^i\|_{\mathcal{H}_k}^2 \right)^{\frac{1}{2}} + D_3 d_z \sum_{i=1}^{d_z} \|w_t^i\|_{\mathcal{H}_k}^2 \\ &\leq D \left( \|v_t\|_{\mathbb{R}^{d_\theta}}^2 + 2 \|v_t\|_{\mathbb{R}^{d_\theta}} \|w_t\|_{\mathcal{H}_k^{d_z}} + \|w_t\|_{\mathcal{H}_k^{d_z}}^2 \right) \end{aligned} \quad (50)$$

$$\leq 2D \left( \|v_t\|_{\mathbb{R}^{d_\theta}}^2 + \|w_t\|_{\mathcal{H}_k^{d_z}}^2 \right) := \beta I(\theta_t, \mu_t), \quad (51)$$

where in the first line we have used the Cauchy-Schwarz inequality, in the second line we have defined the constant  $D = \max(D_1, \frac{1}{2} D_2 d_\theta^{\frac{1}{2}} d_z^{\frac{1}{2}}, D_3 d_z)$ , and in the final line we have used  $a^2 + 2ab + b^2 = (a+b)^2 \leq 2(a^2 + b^2)$ , and set  $\beta = 2D$ . It remains to obtain the bounds on  $A_t^{ij}$ ,  $B_t^{ij}$ , and  $C_t^{ij}$ .

For the first term, we just require the boundedness of the Hessian  $H_{V_z} : \mathbb{R}^{d_\theta} \rightarrow \mathbb{R}^{d_\theta}$  of the function  $V_z : \theta \mapsto -\log \pi_\theta(z)$ , for each  $z \in \mathcal{Z}$ , which is guaranteed by Assumption 2. In particular, we recall that  $\|H_{V_z}\|_{\text{op}} \leq M_1$ , from which it immediately follows that  $\|A_t^{ij}\|_{\text{op}} \leq 2M_1$ . For the second term, using the boundedness of  $H_{V_{z,\theta}}$  (Assumption 6), and the boundedness of the kernel (Assumption 1), we have

$$\|B_t^{ij}\|_{\mathcal{H}_k} \leq 2 \int \|k(z, \cdot)\|_{\mathcal{H}_k} |\partial_{\theta_i} \partial_{z_i} \log \pi_{\theta_t}(z)| \mu_t(z) dz \leq BM_3. \quad (52)$$

For the final term, we argue similarly to in [36, Proposition 8]. In particular, using the boundedness of the Hessian  $H_{V_\theta} : \mathbb{R}^{d_z} \rightarrow \mathbb{R}^{d_z}$  (Assumption 2) of the function  $V_\theta : z \mapsto -\log \pi_\theta(z)$  for each  $\theta \in \Theta$ , and the boundedness of the kernel and its gradient (Assumption 1), we have

$$\begin{aligned} \|C_t^{ij}\|_{\text{HS}} &\leq 2 \int \|k(z, \cdot)\|_{\mathcal{H}_k} |\partial_{z_i} \partial_{z_j} \log \pi_{\theta_t}(z)| \mu_t(z) dz \|k(z', \cdot)\|_{\mathcal{H}_k} \mu_t(z') dz' \\ &\quad + 2 \int \|\partial_{z_i} k(z, \cdot)\|_{\mathcal{H}_k} |\partial_{z_i} \log \pi_{\theta_t}(z)| \mu_t(z) dz \int \|k(z', \cdot)\|_{\mathcal{H}_k} \mu_t(z') dz' \\ &\quad + 2 \left( \int \|k(z, \cdot)\|_{\mathcal{H}_k} \mu_t(z) dz \right)^2 \leq 2B^2 \left( M_2 + \int |\partial_{z_i} \log \pi_{\theta_t}(z)| \mu_t(z) dz + 1 \right). \end{aligned} \quad (53)$$

To bound the remaining integral, we use the linear growth of the function  $z \mapsto \nabla_z \log \pi_\theta(z)$  (Assumption 5), and the bounded moment condition stated in the proposition. From these assumptions, it follows that

$$\int |\partial_{z_i} \log \pi_{\theta_t}(z)| \mu_t(dz) \leq \int L [1 + \|z\|] \mu_t(z) dz \leq L(1 + C). \quad (54)$$

Substituting (54) into (53), we thus have that  $\|C_t^{ij}\|_{\text{HS}} \leq 2B^2(1 + CL + L + M_2)$ . Thus, setting  $D_1 = 2M_1$ ,  $D_2 = BM_3$ ,  $D_3 = 2B^2(1 + CL + L + M_2)$ , and using the argument in (50) - (51), we have established the result in (30).



It remains to establish that  $I(\theta_t, \mu_t) \rightarrow 0$ . Once more, we will utilize some of the ideas from the proofs of [38, Theorem 3] and [36, Proposition 8]. We begin by observing that the free energy can be written as

$$\mathcal{F}(\theta, \mu) = \int \nabla_z \log \left( \frac{\mu}{\pi_\theta} \right) \mu(z) dz \quad (55)$$

$$= \int \log \left( \frac{\mu}{p_\theta(\cdot|x)p_\theta(x)} \right) \mu(z) dz \quad (56)$$

$$= \underbrace{\int \log \left( \frac{\mu_t}{p_{\theta_t}(\cdot|x)} \right) \mu(z) dz}_{\text{KL}(\mu_t|p_{\theta_t}(\cdot|x))} - \log p_\theta(x). \quad (57)$$

Rearranging (57), using the non-negativity of the KL divergence, and finally the fact that the free energy dissipates along the flow of the SVGD EM dynamics (Proposition 1), it then follows that

$$\log p_{\theta_t}(x) = -\mathcal{F}(\theta_t, \mu_t) + \text{KL}(\mu_t|p_{\theta_t}(\cdot|x)) \geq \mathcal{F}(\theta_t, \mu_t) \geq \mathcal{F}(\theta_0, \mu_0). \quad (58)$$

Under Assumption 4, the set  $\Theta_{\mathcal{F}_0} = \{\theta \in \Theta : \log p_\theta(x) \geq \mathcal{F}(\theta_0, \mu_0)\}$  is compact. This, along with (58), immediately implies that  $(\theta_t)_{t \geq 0}$  is relatively compact. We now return to (57). In particular, once more rearranging this equation, we have that, for each  $t \geq 0$ ,

$$\inf_{\theta \in \Theta_{\mathcal{F}_0}} \text{KL}(\mu_t|p_\theta(\cdot|x)) \leq \text{KL}(\mu_t|p_{\theta_t}(\cdot|x)) = \mathcal{F}(\theta_t, \mu_t) + \log p_{\theta_t}(x) \quad (59)$$

$$\leq \mathcal{F}(\theta_0, \mu_0) + \log p_{\theta_t}(x) \leq \mathcal{F}(\theta_0, \mu_0) + \sup_{\theta \in \Theta_{\mathcal{F}_0}} \log p_\theta(x). \quad (60)$$

It thus follows, using also the fact that the KL divergence has compact sub-level sets in the weak topology [69, Theorem 20], that the family  $(\mu_t)_{t \geq 0}$  is weakly relatively compact. Based on these two observations, and also the (weak) continuity of  $I(\theta, \mu)$ , we can conclude that  $\sup_{t \geq 0} I(\theta_t, \mu_t) < \infty$ . Thus, using the bound (30) that we established earlier in the proof, we have that  $|\frac{d}{dt} I(\theta_t, \mu_t)| \leq K$  for some  $K > 0$ .

Finally, we are ready to show that  $I(\theta_t, \mu_t) \rightarrow 0$ . We follow closely the final part of the proof of [36, Proposition 8], which we can now apply almost verbatim. For the interested reader, we also provide the details here. We will argue by contradiction. In particular, suppose that  $I(\theta_t, \mu_t)$ , so that there exists a sequence  $t_m \rightarrow \infty$  such that  $I(\theta_{t_m}, \mu_{t_m}) > \varepsilon > 0$ . In addition, since  $|\frac{d}{dt} I(\theta_t, \mu_t)|$  is bounded, it is uniformly  $K$  Lipschitz in time. Thus, there exists a sequence of intervals  $J_m$  of length  $\frac{m}{K}$ , and centered at  $t_m$ , such that  $I(\theta_t, \mu_t) \geq \frac{\varepsilon}{2}$  for all  $t \in J_k$ . Finally, integrating the dissipation over  $s \in [0, t]$ , we have

$$\mathcal{F}(\theta_0, \mu_0) - \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu) \geq \mathcal{F}(\theta_0, \mu_0) - \mathcal{F}(\mu_t, \theta_t) \quad (61)$$

$$= \int_0^t I(\theta_s, \mu_s) ds \geq \sum_{m: t_m \leq t} \frac{\varepsilon^2}{2K}, \quad (62)$$

which diverges as  $t \rightarrow \infty$  since the subsequence  $t_m \rightarrow \infty$ . But this is contradiction, since  $\mathcal{F}(\theta_0, \mu_0) < \infty$ . This completes the proof.  $\square$

*Proof of Proposition 3.* Suppose we write  $\mathcal{F}^* = \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu)$ . Proceeding almost identically to the start of the proof of Proposition 1, we have

$$\frac{d}{dt} (\mathcal{F}(\theta_t, \mu_t) - \mathcal{F}^*) \quad (63)$$

$$= \langle -\nabla_\theta \mathcal{F}(\theta_t, \mu_t), \nabla_\theta \mathcal{F}(\theta_t, \mu_t) \rangle_{\mathbb{R}^{d_\theta}} + \langle -P_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t), \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t) \rangle_{L^2(\mu_t)} \quad (64)$$

$$= -\|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 - \|S_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \quad (65)$$

$$\leq -2\lambda (\mathcal{F}(\theta_t, \mu_t) - \mathcal{F}^*). \quad (66)$$

where in the final line we have used the gradient dominance condition (Assumption 7). The conclusion now follows straightforwardly via Grönwall's inequality.  $\square$

## B.2 Proof of Theorem 1.

*Proof.* For fixed  $t \in \mathbb{N}_0$ , consider the following decomposition

$$\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_t, \mu_t) = \underbrace{\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_{t+1}, \mu_t)}_{\text{I}} + \underbrace{\mathcal{F}(\theta_{t+1}, \mu_t) - \mathcal{F}(\theta_t, \mu_t)}_{\text{II}}. \quad (67)$$

We will deal with (I) and (II) in turn, beginning with (II). First, for each  $\tau \geq 0$ , and for fixed  $t \in \mathbb{N}_0$ , let  $\theta_\tau = \theta_t - \tau \nabla_\theta \mathcal{F}(\theta_t, \mu_t)$ . We then have  $\theta_0 = \theta_t$  and  $\theta_\gamma = \theta_t - \gamma \nabla_\theta \mathcal{F}(\theta_t, \mu_t) = \theta_{t+1}$ . We also have that  $\dot{\theta}_\tau = -\nabla_\theta \mathcal{F}(\theta_t, \mu_{t+1})$  and  $\ddot{\theta}_\tau = 0$ , where here we emphasize that  $\cdot$  and  $\ddot{\cdot}$  denote derivatives with respect to the continuous  $\tau \in \mathbb{R}_+$ , rather than the discrete  $t \in \mathbb{N}_0$ .

In addition, suppose we let  $g(\tau) = \mathcal{F}(\theta_\tau, \mu_t)$ . We then have  $g(0) = \mathcal{F}(\theta_t, \mu_t)$  and  $g(\gamma) = \mathcal{F}(\theta_{t+1}, \mu_t)$ . In addition, using a Taylor expansion, we have that

$$g(\gamma) = g(0) + \gamma g'(0) + \int_0^\gamma (\gamma - \tau) g''(\tau) d\tau. \quad (68)$$

Let us identify the remaining terms in this expansion. Using the chain rule, and in the second line also the fact that  $\ddot{\theta}_\tau = 0$ , we have that

$$g'(\tau) = \langle \dot{\theta}_\tau, \nabla_\theta \mathcal{F}(\theta_\tau, \mu_t) \rangle_{\mathbb{R}^{d_\theta}} = -\langle \nabla_\theta \mathcal{F}(\theta_t, \mu_t), \nabla_\theta \mathcal{F}(\theta_\tau, \mu_t) \rangle_{\mathbb{R}^{d_\theta}} \quad (69)$$

$$g''(\tau) = \langle \dot{\theta}_\tau, \nabla_\theta^2 \mathcal{F}(\theta_\tau, \mu_t) \dot{\theta}_\tau \rangle_{\mathbb{R}^{d_\theta}} = \langle \nabla_\theta \mathcal{F}(\theta_t, \mu_t), \text{Hess}_\theta(\mathcal{F})(\theta_\tau, \mu_t) \nabla_\theta \mathcal{F}(\theta_t, \mu_t) \rangle_{\mathbb{R}^{d_\theta}} \quad (70)$$

where  $\text{Hess}_\theta(\mathcal{F})$  denotes the Hessian matrix of  $\mathcal{F}(\cdot, \mu_t)$ . Thus,  $g'(0) = -\|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2$ , and  $g''(\tau) \leq M_1 \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2$ , the latter by Assumption 2. Putting everything together, we thus have that

$$\mathcal{F}(\theta_{t+1}, \mu_t) \leq \mathcal{F}(\theta_t, \mu_t) - \gamma \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 + M_1 \int_0^\gamma (\gamma - \tau) \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 d\tau \quad (71)$$

$$\leq \mathcal{F}(\theta_t, \mu_t) - \gamma \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 + \frac{M_1 \gamma^2}{2} \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 \quad (72)$$

which implies, in particular, that

$$\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_t, \mu_t) \leq -\gamma \left(1 - \frac{M_1 \gamma}{2}\right) \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2. \quad (73)$$

We will use a very similar argument to obtain an upper bound on (I), adapted appropriately to the Wasserstein space. Here, we follow very closely the proof of [36, Proposition 5]. In the interest of completeness, we provide this argument in full. Similar to above, for  $\tau \geq 0$ , let  $\psi_\tau = \text{id} - \tau \xi$ , where  $\xi = \mathcal{P}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)$ , and where  $\text{id}$  denotes the identity operator. In addition, let  $\rho_\tau = (\psi_\tau)_\# \mu_t$ . By definition, we then have  $\rho_0 = \mu_t$  and  $\rho_\gamma = \mu_{t+1}$ .

In addition, let us define  $h(\tau) = \mathcal{F}(\theta_{t+1}, \rho_\tau)$ . Using the definition above, we have that  $h(0) = \mathcal{F}(\theta_{t+1}, \mu_t)$  and  $h(\gamma) = \mathcal{F}(\theta_{t+1}, \mu_{t+1})$ . Now, via a Taylor expansion,

$$h(\gamma) = h(0) + \gamma h'(0) + \int_0^\gamma (\gamma - \tau) h''(\tau) d\tau. \quad (74)$$

Similar to before, it remains to identify the final two terms. In this case, use an appropriate chain rule [e.g. 70, Section 8.2], we have

$$h'(\tau) = \langle \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \rho_\tau), w_\tau \rangle_{L^2(\rho_\tau)} \quad (75)$$

$$h''(\tau) = \langle w_\tau, \text{Hess}_{W_2}(\mathcal{F})(\theta_{t+1}, \rho_\tau) w_\tau \rangle_{L^2(\rho_\tau)} \quad (76)$$

where  $w_\tau \in L^2(\rho_\tau)$ , defined according to  $w_\tau(z) = -\xi(\psi_\tau^{-1}(z))$ , is the velocity field which rules the time evolution of  $\rho_\tau$  [70, Theorem 5.34], and  $\text{Hess}_{W_2}(\mathcal{F})$  is the Hessian of  $\mathcal{F}(\theta_{t+1}, \cdot)$ . It follows, in particular, that

$$h'(0) = -\langle \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t), \xi \rangle_{L^2(\mu_t)} = -\|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2. \quad (77)$$

In addition, we have that

$$h''(\tau) = \underbrace{\mathbb{E}_{z \sim \rho_\tau} [\langle w_\tau(z), \text{Hess}_z(-\log \pi_{\theta_{t+1}}(z)) w_\tau(z) \rangle]}_{h_1(\tau)} + \underbrace{\mathbb{E}_{z \sim \rho_\tau} [\|J(w_\tau(z))\|_{\text{HS}}^2]}_{h_2(\tau)} \quad (78)$$

where  $J$  denotes the Jacobian matrix, and  $\|\cdot\|_{\text{HS}}$  denotes the Hilbert-Schmidt norm. It remains to bound these two terms. Recalling that  $\rho_\tau = (\psi_\tau)_\# \mu_t$  and  $w_\tau(z) = -\xi(\psi_\tau^{-1}(z))$ , and using Assumption 2, we have

$$h_1(\tau) = \mathbb{E}_{z \sim \rho_\tau} [\langle w_\tau(z), \text{Hess}(-\log \pi_{\theta_{t+1}}(z)) w_\tau(z) \rangle] \quad (79)$$

$$= \mathbb{E}_{z \sim \rho_\tau} [\langle \xi(\psi_\tau^{-1}(z)), \text{Hess}(-\log \pi_{\theta_{t+1}}(z)) \xi(\psi_\tau^{-1}(z)) \rangle] \quad (80)$$

$$= \mathbb{E}_{z \sim \mu_t} [\langle \xi(z), \text{Hess}(-\log \pi_{\theta_{t+1}}(z)) \xi(z) \rangle] \quad (81)$$

$$\leq M_2 \mathbb{E}_{z \sim \mu_t} [\|\xi(z)\|^2] \leq M_2 B^2 \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2, \quad (82)$$

where in the final line we have used [36, Lemma 9], which holds under Assumption 1. For the remaining term, first note that, by the chain rule, we have  $Jw_\tau(z) = J\xi(\psi_\tau^{-1}(z))(J\psi_\tau)^{-1}(\psi_\tau^{-1}(z))$ ; see the proof of [36, Lemma 11]. It follows that

$$h_2(\tau) = \mathbb{E}_{z \sim \rho_\tau} [\|J(w_\tau(z))\|_{\text{HS}}^2] = \mathbb{E}_{z \sim \rho_\tau} [\|J\xi(\psi_\tau^{-1}(z))(J\psi_\tau)^{-1}(\psi_\tau^{-1}(z))\|_{\text{HS}}^2] \quad (83)$$

$$= \mathbb{E}_{z \sim \mu_t} [\|J\xi(z)(J\psi_\tau)^{-1}(z)\|_{\text{HS}}^2] \leq \alpha^2 B^2 \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2, \quad (84)$$

where the final display follows from [36, Lemma 9,10], which hold under Assumption 1 and Assumptions 1 and 3, respectively, as well as our assumption on the step size. Finally, substituting everything back into (74), we have

$$\mathcal{F}(\theta_{t+1}, \mu_{t+1}) \leq \mathcal{F}(\theta_{t+1}, \mu_t) - \gamma \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \quad (85)$$

$$+ (M_2 + \alpha^2) B^2 \int_0^\gamma (\gamma - \tau) \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 d\tau \quad (86)$$

$$\leq \mathcal{F}(\theta_{t+1}, \mu_t) - \gamma \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \quad (87)$$

$$+ \left( \frac{M_2 + \alpha^2}{2} \right) B^2 \gamma^2 \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2, \quad (88)$$

and thus, in particular, that

$$\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_{t+1}, \mu_t) \leq -\gamma \left( 1 - \frac{(M_2 + \alpha^2) B^2 \gamma}{2} \right) \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2. \quad (89)$$

Finally, combining (73) and (89), we have the desired bound,

$$\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_t, \mu_t) \leq -\gamma \left[ \left( 1 - \frac{M_1 \gamma}{2} \right) \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 + \left( 1 - \frac{(M_2 + \alpha^2) B^2 \gamma}{2} \right) \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \right]. \quad (90)$$

□

### B.3 Proof of Corollary 1.

*Proof.* Using Theorem 1, and the definition of  $c_\gamma$ , we have that

$$\mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_t, \mu_t) \leq -c_\gamma \left( \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 + \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \right).$$

Let  $\mathcal{F}^* = \min_{(\theta, \mu) \in \Theta \times \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu)$ . Using the previous inequality, and the definition of  $\mathcal{F}^*$ , we have

$$-\mathcal{F}(\theta_0, \mu_0) \leq -\mathcal{F}(\theta_1, \mu_1) \leq (\mathcal{F}(\theta_{T+1}, \mu_{T+1}) - \mathcal{F}^*) - \mathcal{F}(\theta_1, \mu_1), \quad (91)$$

Rearranging, and using a telescoping sum, it follows straightforwardly that

$$-(\mathcal{F}(\theta_0, \mu_0) - \mathcal{F}^*) \leq \sum_{t=1}^T \mathcal{F}(\theta_{t+1}, \mu_{t+1}) - \mathcal{F}(\theta_t, \mu_t) \quad (92)$$

$$\leq -c_\gamma \sum_{t=0}^{T-1} \left( \|\nabla_\theta \mathcal{F}(\theta_t, \mu_t)\|_{\mathbb{R}^{d_\theta}}^2 + \|\mathcal{S}_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_{t+1}, \mu_t)\|_{\mathcal{H}_k^{d_z}}^2 \right). \quad (93)$$

Finally, and dividing both sides by  $-c_\gamma T$ , we have the required conclusion. □

## C Marginal SVGD EM and Marginal Coin EM

In certain models, the M step is tractable. In other words, it is possible to derive a tractable expression for  $\theta_*(\mu) = \arg \max_{\mu \in \mathcal{P}_2(\mathcal{Z})} \mathcal{F}(\theta, \mu)$ . Thus, in particular, given the empirical measure  $\mu^N = \frac{1}{N} \sum_{j=1}^N \delta_{z^j}$ , we can compute  $\theta_*(z^{1:N}) := \theta_*(\mu^N)$ , where  $z^{1:N} = (z^1, \dots, z^N) \in \mathcal{Z}^N$ . In such cases, instead of SVGD EM (Alg. 1) or Coin EM (Alg. 2), we can use marginal variants of these algorithms, in which the  $\theta$  updates are now exact.

---

### Algorithm 3 Marginal SVGD EM

---

**input:** number of iterations  $T$ , number of particles  $N$ , initial particles  $\{z_0^i\}_{i=1}^N \sim \mu_0$ , initial  $\theta_0$ , target density  $\pi$ , kernel  $k$ , function  $\mu \mapsto \theta_*(\mu)$ , learning rate  $\gamma$ .

**for**  $t = 1, \dots, T - 1$  **do**

For  $i \in [N]$

$$z_{n+1}^i = z_n^i + \frac{\gamma}{N} \sum_{j=1}^N \left[ k(z_t^j, z_t^i) \nabla_z \log \pi_{\theta_*(z^{1:N})}(z_t^j) + \nabla_{z_t^i} k(z_t^j, z_t^i) \right]$$

**end for**

**return**  $\theta_T$  and  $\{z_T^i\}_{i=1}^N$ .

---



---

### Algorithm 4 Marginal Coin EM

---

**input:** number of iterations  $T$ , number of particles  $N$ , initial particles  $\{z_0^i\}_{i=1}^N \sim \mu_0$ , initial  $\theta_0$ , target density  $\pi$ , kernel  $k$ , function  $\mu \mapsto \theta_*(\mu)$ .

**for**  $t = 1, \dots, T$  **do**

For  $i \in [N]$

$$z_t^i = z_0^i + \frac{\sum_{s=1}^{t-1} \frac{1}{N} \sum_{j=1}^N k(z_s^j, z_s^i) \nabla_z \log \pi_{\theta_*(z^{1:N})}(z_s^j) + \nabla_{z^j} k(z_s^j, z_s^i)}{t} \\ \times \left( 1 + \sum_{s=1}^{t-1} \left\langle \frac{1}{N} \sum_{j=1}^N k(z_s^i, z_s^j) \nabla_z \log \pi_{\theta_*(z^{1:N})}(z_s^j) + \nabla_{z^j} k(z_s^j, z_s^i), z_s^i - z_0^i \right\rangle \right)$$

**end for**

**return**  $\theta_T$  and  $\{z_T^i\}_{i=1}^N$ .

---

## D Adaptive Coin EM

In Sec. 2.4, the update equation given in (11), and thus the update equations (12) - (13), and the update equations in Alg. 2, were given under the assumption that the sequence of outcomes, in this case  $(c_t^\theta)_{t \in [T]} = (\nabla_\theta \mathcal{F}(\theta_t, \mu_t^N))_{t \in [T]}$  and  $(c_t^\mu)_{t \in [T]} = (P_{\mu_t} \nabla_{W_2} \mathcal{F}(\theta_t, \mu_t^N))_{t \in [T]}$ , were bounded above by 1 (see the remark in Footnote 2). In practice, of course, this may not be the case. If, instead, there exists a known constant which upper bounds  $(c_t^\theta)_{t \in [T]}$  and  $(c_t^\mu)_{t \in [T]}$ , then one can simply replace these quantities by their normalized versions in Alg. 2. In the more unlikely scenario that such a constant is unknown, we can instead use an adaptive version of Coin EM, in which an empirical estimate of this constant is updated as the algorithm progresses; see [55, Sec. 6] and [62, App. D] for precedents. This algorithm, which we recommend and use in all experiments, is summarized in Alg. 5.

In addition, once more following [55, Sec. 6], whenever we apply CoinEM to a Bayesian neural network (see Sec. 4.3 and App. E.5), we further alter the parameter update equation in Alg. 5 to read as

$$\theta_{t,j} = \theta_{0,j} + \frac{\sum_{s=1}^{t-1} c_{s,j}^\theta}{\max(G_{t,j}^\theta + L_{t,j}^\theta, 100L_{t,j}^\theta)} \left( 1 + \frac{R_{t,j}^\theta}{L_{t,j}^\theta} \right), \quad (94)$$

with an analogous modification for the particle update equation. It is worth noting that both of these adaptive versions of the CoinEM algorithm remain entirely tuning free.

---

**Algorithm 5** Adaptive Coin EM

---

**input:** number of iterations  $T$ , number of particles  $N$ , initial particles  $\{z_0^i\}_{i=1}^N \sim \mu_0$ , initial  $\theta_0$ , target density  $\pi$ , kernel  $k$

**initialize:** for  $i \in [N]$ :  $L_{0,j}^\theta = 0$ ,  $G_{0,j}^\theta = 0$ ,  $R_{0,j}^\theta = 0$ ; for  $i \in [N]$  and  $j \in [d]$ ,  $L_{0,j}^{z,i} = 0$ ,  $G_{0,j}^{z,i} = 0$ ,  $R_{0,j}^{z,i} = 0$ .

**for**  $t = 1, \dots, T$  **do**

  Compute

$$c_{t-1}^\theta = \frac{1}{N} \sum_{j=1}^N \nabla_\theta \log \pi_{\theta_{t-1}}(z_{t-1}^j) \quad (\text{parameter gradient})$$

**for**  $j = 1, \dots, d_\theta$  **do**

    Compute

$$L_{t,j}^\theta = \max(L_{t-1,j}^\theta, |c_{t-1,j}^\theta|) \quad (\text{max. observed scale})$$

$$G_{t,j}^\theta = G_{t-1,j}^\theta + |c_{t-1,j}^\theta| \quad (\text{sum of absolute value of gradients})$$

$$R_{t,j}^\theta = \max(R_{t-1,j}^\theta + \langle c_{t-1,j}^\theta, \theta_{t-1,j} - \theta_{0,j} \rangle, 0) \quad (\text{total reward})$$

$$\theta_{t,j} = \theta_{0,j} + \frac{\sum_{s=1}^{t-1} c_{s,j}^\theta}{G_{t,j}^\theta + L_{t,j}^\theta} \left(1 + \frac{R_{t,j}^\theta}{L_{t,j}^\theta}\right). \quad (\text{parameter update})$$

**end for**

**for**  $i = 1, \dots, N$  **do**

    Compute

$$c_{t-1}^{z,i} = \frac{1}{N} \sum_{j=1}^N k(z_{t-1}^j, z_{t-1}^i) \nabla_{z^j} \log \pi_{\theta_t}(z_{t-1}^j) + \nabla_{z^j} k(z_{t-1}^j, z_{t-1}^i) \quad (\text{particles gradient})$$

**for**  $j = 1, \dots, d_z$  **do**

      Compute

$$L_{t,j}^{z,i} = \max(L_{t-1,j}^{z,i}, |c_{t-1,j}^{z,i}|) \quad (\text{max. observed scale})$$

$$G_{t,j}^{z,i} = G_{t-1,j}^{z,i} + |c_{t-1,j}^{z,i}| \quad (\text{sum of absolute value of gradients})$$

$$R_{t,j}^{z,i} = \max(R_{t-1,j}^{z,i} + \langle c_{t-1,j}^{z,i}, z_{t-1,j}^i - z_{0,j}^i \rangle, 0) \quad (\text{total reward})$$

$$z_{t,j}^i = z_{0,j}^i + \frac{\sum_{s=1}^{t-1} c_{s,j}^{z,i}}{G_{t,j}^{z,i} + L_{t,j}^{z,i}} \left(1 + \frac{R_{t,j}^{z,i}}{L_{t,j}^{z,i}}\right). \quad (\text{particles update})$$

**end for**

**end for**

**end for**

**output:**  $\theta_T$  and  $(z_T^i)_{i=1}^N$ .

---

## E Additional Experimental Details

We implement all of the algorithms using Python 3 and JAX [10]. We perform all experiments using a MacBook Pro 16" (2021) laptop with Apple M1 Pro chip and 16GB of RAM.

### E.1 Toy hierarchical model

**Implementation Details.** For the results in Fig. 1, we initialize all methods with  $\theta_0 \sim \mathcal{N}(0, 0.1^2)$  and  $z_0^i \sim \mathcal{N}(0, 1)$ . We use  $N = 10$  particles and run each algorithm for  $T = 500$  iterations. Additional results for larger numbers of particles are given in App. F.1. In Fig. 1(a), we run each algorithm over a grid of 50 logarithmically spaced learning rates  $\gamma \in [10^{-5}, 10^3]$ . We use Adagrad [25] to automatically adjust the learning rates of SVGD EM and PGD. In Fig. 1(b) and Fig. 1(c), we run each algorithm using the learning rate which obtained the lowest final iterate MSE in Fig. 1(a).



For Fig. 2(a) and Fig. 2(b), we run the algorithms for  $T = 5000$  iterations, and report the empirical variance of the latent particles  $(z_t^i)_{i=1}^N$  for each  $t \in [0, T]$ . In Fig. 2(c), we run each algorithm for  $T = 250$  iterations with  $N = 50$  particles. For Coin EM and SVGD EM, we compute the MSE between the empirical variance of the final particles  $(z_T^i)_{i=1}^N$ , and the true posterior variance, which in this case is just given by 0.5 [38, App. E.1]. For PGD and SOUL, we use the empirical variance of time-averaged particles, due to the noise used in the particle updates. All results are averaged over 10 random seeds (5 random seeds for Fig. 2(c)), and we report 95% bootstrap CIs as well as the mean.

## E.2 Bayesian logistic regression

**Model.** We consider the setup in [20, Sec. 4.1]. The observed data is  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ , where  $x_i \in \mathbb{R}^{d_x}$  are  $d_x$ -dimensional real-valued covariates, and  $y_i \in \{0, 1\}$  are binary class labels. We assume the labels  $y_i$  are conditionally independent given the features  $x_i$  and the regression weights  $z \in \mathbb{R}^{d_z}$ , with  $p(\{y_i, x_i\}|z) = \sigma(x_i^T z)^{y_i} [1 - \sigma(x_i^T z)]^{1-y_i}$  for  $i \in [N]$ , where  $\sigma(u) := e^u / (1 + e^u)$  denotes the standard logistic function. We place a Gaussian prior on the latent weights,  $p(z) = \mathcal{N}(z|\theta \mathbf{1}_{d_z}, 5 \mathbf{1}_{d_z})$ , for some real parameter  $\theta \in \mathbb{R}$  which we would like to estimate. The model is thus

$$p_\theta(z, \mathcal{D}) = \mathcal{N}(z|\theta \mathbf{1}_{d_z}, 5 \mathbf{1}_{d_z}) \prod_{i=1}^N \sigma(x_i^T z)^{y_i} [1 - \sigma(x_i^T z)]^{1-y_i}. \quad (95)$$

**Data.** We fit this model using the Wisconsin Breast Cancer dataset [72]. This dataset contains 683 datapoints, each with nine features  $x_i \in \mathbb{R}^9$  extracted from a digitized image of a fine needle aspirate of a breast mass, and a label  $y_i \in \{0, 1\}$  indicating whether the mass is benign ( $y_i = 0$ ) or malign ( $y_i = 1$ ). We normalize the features so that each has mean zero and unit standard deviation across the dataset, and split the data using a random 80-20 train-test split.

**Implementation Details.** For the results in Fig. 3, following [20, 38], we initialize  $\theta_0 = 0$  and  $z_0^i \sim \mathcal{N}(0, 1)$ . In Fig. 3(a) and Fig. 3(b), we use a learning rate of  $\gamma = 0.02$  for PGD, PMGD, and SOUL, and a learning rate of  $\gamma = 0.2$  for SVGD EM, which typically obtains its best performance for a higher learning rate than the other methods (see, e.g., Fig. 1(a) or Fig. 2(c)). We run each algorithm with  $N = 100$  particles for  $T = 800$  iterations. In Fig. 3(c), we run each algorithm using a grid of 50 logarithmically spaced learning rates  $\gamma \in [10^{-10}, 10^2]$ , using  $N = 20$  particles and  $T = 400$  iterations. We repeat each experiment over 10 random test-train splits of the data.

## E.3 Bayesian logistic regression (alternative model)

In App. F.3, we present results for an alternative Bayesian logistic regression model, which were omitted from the main text due to space constraints. Here, we provide full details of this model.

**Model.** We now follow the setup in [29]. Similar to before, the observed data is given by  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ , where  $x_i \in \mathbb{R}^{d_x}$  are  $d_x$ -dimensional real-valued covariates, and  $y_i \in \{0, 1\}$  are binary class labels. Meanwhile, the latent variables are now given by  $z = \{w, \log \alpha\}$ , consisting of a  $d_x$ -dimensional real-valued regression coefficients  $w_k \in \mathbb{R}^{d_x}$ , and a positive precision parameter  $\alpha \in \mathbb{R}_+$ . Finally, the parameters are given by  $\theta = (\log a, \log b)$ , for positive  $a, b \in \mathbb{R}_+$ . As before, we assume that the labels  $y_i$  are conditionally independent given the features  $x_i$  and the regression weights  $z \in \mathbb{R}^{d_z}$ , with  $p(\{y_i, x_i\}|z) = \sigma(x_i^T z)^{y_i} [1 - \sigma(x_i^T z)]^{1-y_i}$  for  $i \in [N]$ , where  $\sigma(u) := e^u / (1 + e^u)$  denotes the standard logistic function. We now also assume that  $p(w_k|a) = N(w_k; 0, \alpha^{-1})$  for  $k \in [d_x]$ , and place a Gamma prior on  $\alpha$ , so  $p(\alpha) = \Gamma(\alpha|a, b)$ . In this case, the model is given by

$$p_\theta(z, x) = \text{Gamma}(\alpha|a, b) \prod_{k=1}^{d_x} N(w_k; 0, \alpha^{-1}) \prod_{i=1}^N \sigma(x_i^T z)^{y_i} [1 - \sigma(x_i^T z)]^{1-y_i}. \quad (96)$$

**Data.** We fit this model to several datasets from the UCI Machine Learning repository [24]: the Covtype dataset, which contains 581012 datapoints and 54 attributes, the Banknote dataset, which contains 1372 datapoints and 4 features, and the Cleveland heart disease dataset, which contains 303 datapoints and 13 features. We split the dataset into train-test sets using a 70-30 train-test split.

**Implementation Details.** For the results in Fig. 13, we initialize  $a_0 = 1$ ,  $b_0 = 0.01$ ,  $\alpha_0^i \sim \text{Gamma}(a_0, b_0)$ , and  $w_0^i \sim \mathcal{N}(0, 1/\alpha_0^i)$ . We run each algorithm using  $N = 10$  particles for  $T = 1000$  iterations, over a grid of logarithmically spaced learning rates  $\gamma \in [10^{-9}, 10^1]$ . We repeat each experiment over 10 random train-test splits of the data.

## E.4 Bayesian neural network

**Model.** We consider the setup described in [30, 45]. The observed data is of the form  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ , where  $x_i \in \mathbb{R}^{d_x}$  are  $d_x$ -dimensional real-valued covariates, and  $y_i \in \mathbb{R}$  are real-valued responses. We assume that the responses  $y_i$  are conditionally independent given the covariates  $x_i$  and the network weights  $w \in \mathbb{R}^{d_w}$ , and that  $p(y_i|x_i, w) = \mathcal{N}(y_i|f(x_i, w), \gamma^{-1})$  for  $i \in [n]$ , where  $f(x_i, w)$  denotes the output of the neural network. We place a Gaussian prior on each of the neural network weights, namely  $p(w_j) = \mathcal{N}(w_j|0, \lambda^{-1})$ , for  $j \in [d_w]$ . To complete our model, we assume a Gamma prior for the inverse covariance  $\gamma \in \mathbb{R}_+$ , and a Gamma hyperprior on the inverse covariance  $\lambda \in \mathbb{R}_+$ , so  $p(\gamma) = \text{Gamma}(\gamma|a_\gamma, b_\gamma)$   $p(\lambda) = \text{Gamma}(\lambda|a_\lambda, b_\lambda)$ . Our model thus takes the form

$$p_\theta(z, \mathcal{D}) = \text{Gamma}(\gamma|a_\gamma, b_\gamma) \text{Gamma}(\lambda|a_\lambda, b_\lambda) \prod_{i=1}^N \prod_{j=1}^{d_x} \mathcal{N}(y_i|f(x_i, w), \gamma^{-1}) \mathcal{N}(w_j|0, \lambda^{-1}) \quad (97)$$

where  $z = (w, \log \lambda, \log \gamma) \in \mathbb{R}^{d_w+1+1}$ , and  $\theta = (\log a_\gamma, \log b_\gamma, \log a_\lambda, \log b_\lambda)$ . Rather than fixing the parameters as in [30, 45], we allow these parameters to be learned from the data.

**Data.** We fit the Bayesian neural network using several UCI datasets [24]. The number of datapoints varies from 506 in the Boston Housing (*Boston*) dataset, to 11934 in the Naval Propulsion (*Naval*) dataset. The number of features is between 4 in the Combined Cycle Power Plant (*Power*) to 16 in the Naval Propulsion dataset (*Naval*). In each case, we normalize the features so that each has mean zero and unit standard deviation across the dataset, and split the data using a random 90-10 train-test split.

**Implementation Details.** For the results in Fig. 4 and Fig. 14, we use a neural network with a single hidden layer and with 50 hidden units, and a Relu activation function. We use Adagrad [25] to automatically adjust the learning rates of SVGD EM and PGD. We initialize the parameters by setting  $a_{\gamma_0} = 1$ ,  $b_{\gamma_0} = 0.1$ ,  $a_{\lambda_0} = 1$ ,  $b_{\lambda_0} = 0.1$ . Meanwhile, we initialize the particles for the precisions as  $\lambda \sim \text{Gamma}(a_{\lambda_0}, b_{\lambda_0})$  and  $\gamma \sim \text{Gamma}(a_{\gamma_0}, b_{\gamma_0})$ , and for the weights  $w$  according to zero-mean Gaussians with variance given by the reciprocal of the input dimension (the latent dimension for the first layer, and the number of hidden units for the second). We run each algorithm using  $N = 20$  particles for  $T = 1000$  iterations, over a logarithmically spaced grid of 30 learning rates  $\gamma \in [10^{-9}, 10^1]$ . We repeat each experiment over 10 random train-test splits of the data.

## E.5 Bayesian neural network (alternative model)

In App. F.5, we present results for an alternative Bayesian neural network model. Here, we provide full details of this model.

**Model.** We consider the setting described in [74, Section 6.5] and [38, Sec. 3.2], applying a neural network to classify MNIST images [41]. In particular, we use a Bayesian two-layer neural network with tanh activation functions, a softmax output layer, and Gaussian priors on the weights.

In this case, the observed data is of the form  $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ , where  $x_i \in \mathbb{R}^{784}$  are  $28 \times 28$  grayscale images of handwritten digits, and  $y_i \in \{0, 1\}$  are labels denoting whether the image is a 4 or 9. We normalise the 784 features so that each has mean zero and unit standard deviation across the dataset. We assume that the labels  $y_i$  are conditionally independent given the features  $x_i$  and the network weights  $z = (w, v)$ , where  $w \in \mathbb{R}^{d_w=40 \times 784}$  and  $v \in \mathbb{R}^{d_v=2 \times 40}$ , and that

$$p(y_i|x_i, w) \exp \left( \sum_{j=1}^{40} v_{ij} \tanh \left( \sum_{k=1}^{784} w_{jk} x_{ik} \right) \right) \quad (98)$$

for  $i \in [n]$ . We then place Gaussian priors on the weights of input layer and the output layer, viz  $p(w_k) = \mathcal{N}(w_k|0, e^{2\alpha})$ , for  $k \in [784]$  and  $p(v_j) = \mathcal{N}(v_j|0, e^{2\beta})$  for  $j \in [40]$ . In this case, rather than assigning a hyperprior to  $\alpha$  and  $\beta$ , we will learn them from data. Thus, our model takes the form

$$p_\theta(z, \mathcal{D}) = \prod_{j=1}^{40} \mathcal{N}(w_k|0, e^{2\alpha}) \prod_{k=1}^{784} \mathcal{N}(v_k|0, e^{2\beta}) \prod_{i=1}^N p(y_i|x_i, w) \quad (99)$$

where  $z = (w, v) \in \mathbb{R}^{40 \times 784 + 2 \times 40}$ , and  $\theta = (\alpha, \beta)$ .

**Data.** We use the MNIST dataset [41], which contains 70'000  $28 \times 28$  grayscale images  $x_i \in \mathbb{R}^{784}$  of handwritten digits between 0 and 9, each with an accompanying label  $y_i \in \{0, 1, \dots, 9\}$ . Following [38, 74], we subsample 1000 points with labels 4 and 9. We normalize the features so that each has mean zero and unit standard deviation across the dataset, and split the data using a random 80-20 train-test split.

**Implementation Details.** For the results in Fig. 15 - 18, we initialize  $\alpha_0 = 0$ ,  $\beta_0 = 0$ , and  $w_k^i \sim \mathcal{N}(0, e^{2\alpha_0})$ ,  $v_j \sim \mathcal{N}(0, e^{2\beta_0})$ , and run each algorithm for  $T = 500$  iterations. In addition, in Fig. 18, we use  $N = 10$  particles, and run each algorithm for 50 logarithmically spaced learning rates  $\gamma \in [10^{-9}, 10^1]$ . We repeat each experiment over 5 (Fig. 15 - 17) or 10 (Fig. 18) random train-test splits of the data.

## E.6 Latent space network model

**Model.** We consider the latent space network model first introduced in [31], where, in the context of social networks, each entity is represented by a node of the network. Edges between nodes indicate connections between entities. We restricted ourselves to binary undirected graphs, where an edge in the network between nodes  $i$  and  $j$  is represented by a binary indicator,  $y_{i,j} = \{0, 1\}$ .

The latent space network model [31, 68] is popular network model to simplify complex network data by embedding the network's nodes in a lower-dimensional space  $z$ . A general form of this model is,

$$Y_{i,j} \sim \text{Bernoulli}(p_{ij}) \quad i \neq j; \quad i, j = 1, \dots, n \quad (100)$$

$$\text{logit}(p_{ij}) = \theta + d(z_i, z_j),$$

where  $d(\cdot, \cdot)$  is a function of the similarity between the latent variables. In this paper, we let  $d(z_i, z_j) = \|z_i - z_j\|$  be the Euclidean distance between  $z_i$  and  $z_j$ , although other similarity metrics which preserve the triangle inequality are also possible.

**Data.** We use a dataset curated by [6] from fan scripts on the website Genius. The dataset contains the frequency of interactions among characters in the first four seasons of the TV series Game of Thrones. Each season is represented by a binary undirected network, where an edge indicates that two characters interacted at least 10 times in that season. For the purpose of visualization, we filter out minor characters with fewer than 10 interactions per season. The dataset covers four seasons of the TV series, one network per season, with  $n = 165$  nodes each season.

**Implementation Details.** We follow a similar inference scheme as [31]. Firstly, we find the maximum likelihood estimates (MLEs) for  $(\theta, z)$  by maximizing the likelihood function (100). Using the MLEs  $(\hat{\theta}, \hat{z})$ , we initialize PGD, SOUL and Coin EM with  $\theta_0 = \hat{\theta}$  and for  $i = 1, \dots, N$ , set  $z_0^i \sim \mathcal{N}(\hat{z}, 0.1)$ . One of the well-known challenges of using latent space network models is that a set of points in Euclidean space will have the same distances regardless of how they are rotated, reflected, or translated. This means that the likelihood (100) is invariant to transformation of the latent positions  $z$ . This issue can be resolved by using a Procrustes transformation of the latent variables, relative to a fixed point, which we choose to be the MLE,  $\hat{z}$ . Therefore, for all  $z_t^i$  in all algorithms we store and plot  $\tilde{z}_t^i = \text{argmin}_{Tz} \text{tr}(\hat{z} - Tz_t^i)^\top (\hat{z} - Tz_t^i)$ . For the SOUL and PGD algorithms we use learning rate parameter  $\gamma = 0.001$ . For all algorithms we use  $N = 10$  particles at  $T = 500$  iterations.

## F Additional Numerical Results

### F.1 Toy hierarchical model

In this section, we provide additional results for the toy hierarchical model studied in Sec. 4.1.

**Additional results for a different number of particles.** We first consider the impact of varying the number of particles. In Fig. 6 and Fig. 7, we replicate the results in Fig. 1, now using  $N = 20$  and  $N = 50$  particles, rather than  $N = 10$ . There is little to distinguish between the two sets of results. In particular, all methods still converge to  $\theta_*$ ; and Coin EM and SVGD EM still tend to achieve a lower MSE than PGD and SOUL. We note that the appearance of wide blue bands in Fig. 6 and Fig. 7 (and Fig. 1) is an artifact of the plotting resolution, and the logarithmic scale. In truth, the CoinEM parameter estimates oscillate rapidly as they convergence towards the marginal maximum likelihood estimate, giving the appearance of these bands at this resolution. This type of oscillatory convergence is common for coin betting algorithms; see, e.g., [52, pg. 36] for a very simple example.

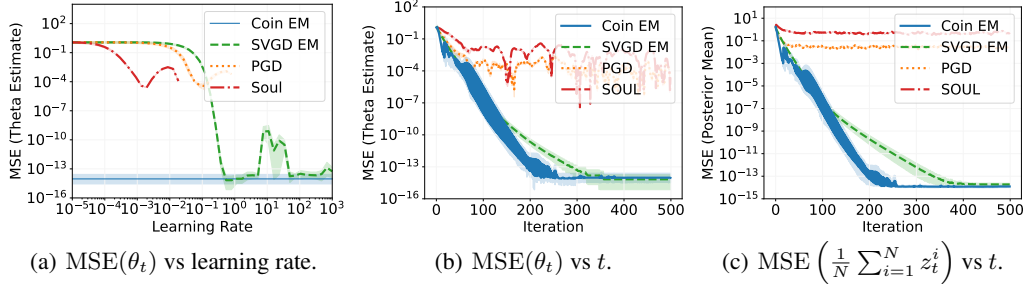


Figure 6: **Additional results for the toy hierarchical model with  $N = 20$  particles.** MSE of the parameter estimate  $\theta_t$  as a function of the learning rate after  $T = 500$  iterations (a); and MSE of the parameter estimate (b) and the posterior mean (c) as a function of the number of iterations, using the optimal learning rate from (a).

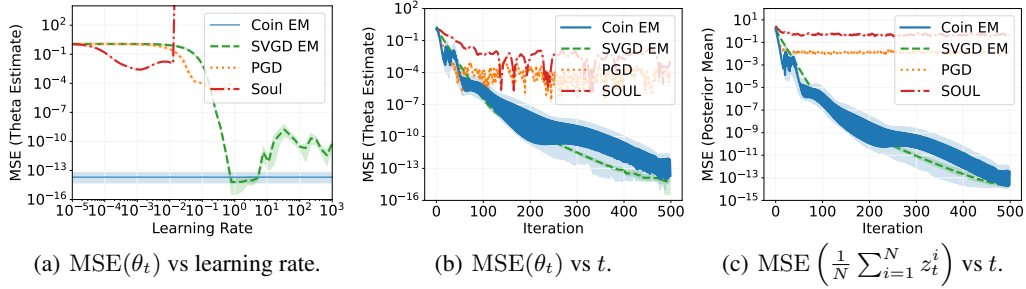


Figure 7: **Additional results for the toy hierarchical model with  $N = 50$  particles.** MSE of the parameter estimate  $\theta_t$  as a function of the learning rate after  $T = 500$  iterations (a); and MSE of the parameter estimate (b) and the posterior mean (c) as a function of the number of iterations, using the optimal learning rate from (a).

**Additional results for different learning rates.** Next, we provide an additional demonstration of how the choice of learning rate can affect the parameter estimates generated by PGD, SOUL, and SVGD EM. In particular, in Fig. 8, we plot the parameter estimates generated by SVGD EM, PGD, and SOUL for three different learning rates: the optimal learning rate as determined by Fig. 7(a), a smaller learning rate, and a larger learning rate ‘at the edge of stability’. The specific learning rates  $\{\gamma_{\text{opt}}, \gamma_{\text{small}}, \gamma_{\text{large}}\}$  used in this figure  $\{0.79, 0.039, 100.00\}$  for SVGD EM,<sup>3</sup>  $\{0.26, 0.013, 1.15\}$  for PGD, and  $\{0.0013, 0.000066, 0.018\}$  for SOUL.

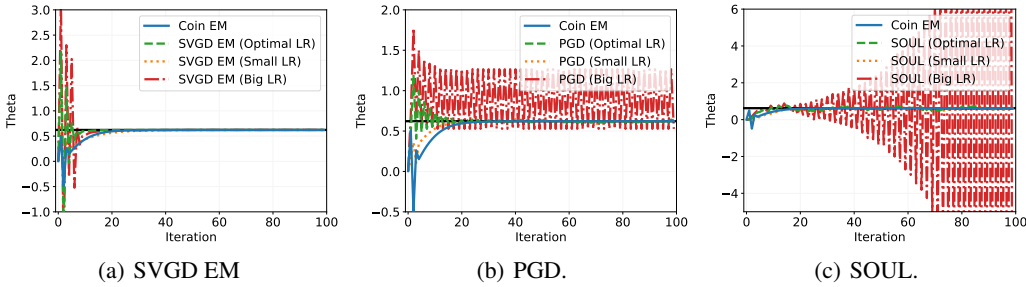


Figure 8: **Additional results for the toy hierarchical model with different learning rates.** The sequence of parameter estimates generated by SVGD EM, PGD, and SOUL, for three different learning rates: the ‘optimal’ learning rate from Fig. 7(a), a smaller learning rate, and a larger learning rate at the edge of stability.

**Additional results for a different initialization.** We now consider the impact of varying the initialization. In Fig. 9, we repeat the experiment in Sec. 4.1, but now using an initialization far away

<sup>3</sup>We note that, when SVGD EM is implemented with an adaptive method such as Adagrad [25], as is the case here, it remains stable even for very large values of the learning rate.

from the true parameter  $\theta = 1$ . In particular, we now initialize the parameters using  $\theta_0 \sim \mathcal{N}(10, 0.1^2)$ . The results are rather similar to before. In particular, Coin EM and SVGD EM both converge rapidly to the true parameter, and tend to outperform the competitors.

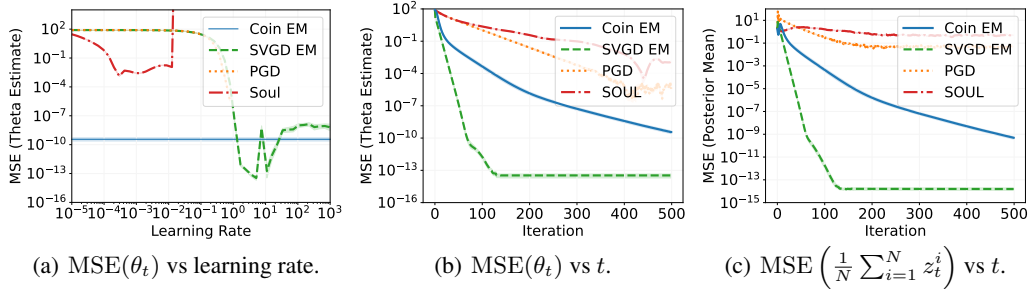


Figure 9: **Additional results for the toy hierarchical model with initialization  $\theta_0 \sim \mathcal{N}(10, 0.1)$ .** MSE of the parameter estimate  $\theta_t$  as a function of the learning rate after  $T = 500$  iterations (a); and MSE of the parameter estimate (b) and the posterior mean (c) as a function of the number of iterations, using the optimal learning rate from (a).

## F.2 Bayesian logistic regression

In this section, we present additional numerical results for the Bayesian logistic regression model in Sec. 4.2.

**Additional results for different initializations.** In Fig. 10, we plot the sequence of parameter estimates output by Coin EM, SVGD EM, PGD, PMGD, and SOUL, for different parameter initializations. In particular, we now initialize the parameter at  $\theta_0 = 10$  or  $\theta_0 = -10$ , compared to  $\theta_0 = 0$  in Fig. 3(a). In both cases, all of methods converge to a similar value. SOUL is known to obtain accurate estimates of  $\theta_*$  in this example [20], provided the learning rate is suitably small, and thus we use this as a benchmark. In these examples, where the parameter estimate is initialized far from the true parameter, the shorter transient exhibited by Coin EM relative to the other methods is even more evident.

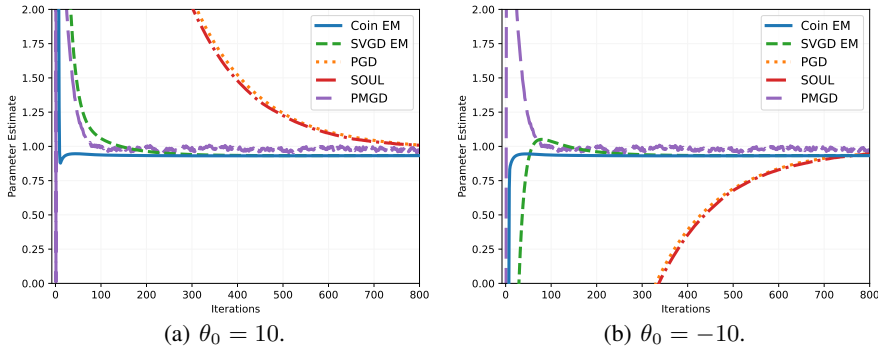


Figure 10: **Additional results for the Bayesian logistic regression in Sec. 4.2.** Parameter estimates  $\theta_t$  from Coin EM, SVGD EM, PGD, SOUL, and PMGD, initialized at (a)  $\theta_0 = 10$  and (b)  $\theta_0 = -10$ .

**Additional results for different learning rates.** In Fig. 11, we plot the sequence of parameter estimates output by Coin EM, SVGD EM, PGD, PMGD, and SOUL, for different learning rates. In particular, we now use smaller learning rates of  $\gamma = 0.001$  (PGD, PMGD, SOUL) or  $\gamma = 0.005$  (SVGD EM), or larger learning rates of  $\gamma = 0.1$  (PGD, PMGD, SOUL) or  $\gamma = 0.5$  (SVGD EM). For reference, we also include the results for the original learning rates of  $\gamma = 0.02$  (PGD, PMGD, SOUL) or  $\gamma = 0.2$  (SVGD EM) in Fig. 3(a) (see Sec. E.2).

These figures illustrate the difficulties associated with tuning the learning rate for PGD, PMGD, SOUL and, to a lesser extent, SVGD EM. In particular, if the learning rate is chosen too small (Fig.



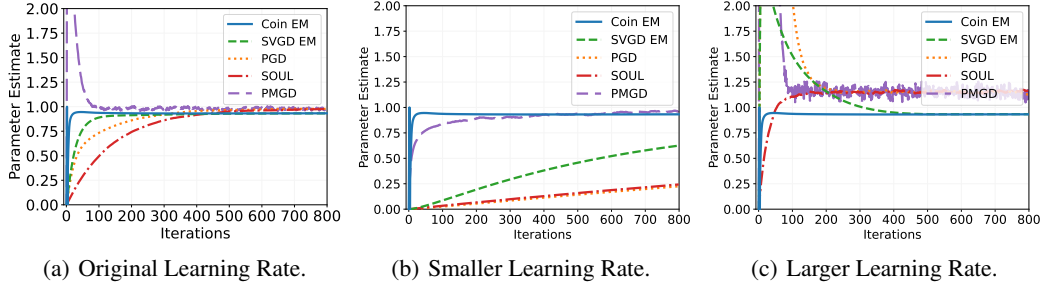


Figure 11: **Additional results for the Bayesian logistic regression in Sec. 4.2.** Parameter estimates  $\theta_t$  from Coin EM, SVGD EM, PGD, SOUL, and PMGD, using (a) smaller learning rates  $\gamma = 0.001$  (PGD, SOUL, PMGD) and  $\gamma = 0.005$  (SVGD EM); and (b) larger learning rates  $\gamma = 0.1$  (PGD, SOUL, PMGD) and  $\gamma = 0.5$  (SVGD EM).

11(b)), then convergence to the true parameter  $\theta_*$  is painfully slow. On the other hand, if one selects a larger learning rate (Fig. 11(c)), convergence is more rapid, but one incurs a (significant) bias in the resulting parameter estimates. This is evident if one compares the asymptotic parameter estimates obtained in Fig. 11(a) and in Fig. 11(c). For PGD, PMGD, and SOUL, this bias originates in the bias associated with an Euler-Maruyama discretization of the Langevin dynamics; see, e.g., the discussion in [38, Sec. 2]. Interestingly, SVGD EM does not seem to incur this bias to the same extent, despite the error associated with the discretization of the continuous-time SVGD dynamics. Coin EM, of course, has no dependence on the learning rate, and is consistent across these experiments.

**Additional results for different numbers of particles.** Finally, we consider the impact of changing the number of particles used in the latent variable updates on the predictive performance. In Fig. 12, we repeat the experiment used to generate Fig. 3(c), but now using  $N = 5, 20, 100$  particles. In this example, we see that there is little to be gained from increasing the number of particles in terms of the predictive performance, other than a minor increase in the performance of the learning-rate dependent methods (SVGD EM, PGD, SOUL) for sub-optimal choices of the learning rate. In this example, the posteriors are peaked and unimodal (see [20, Fig. 2]), and can be approximated well using even a single particle in the vicinity of the modes.

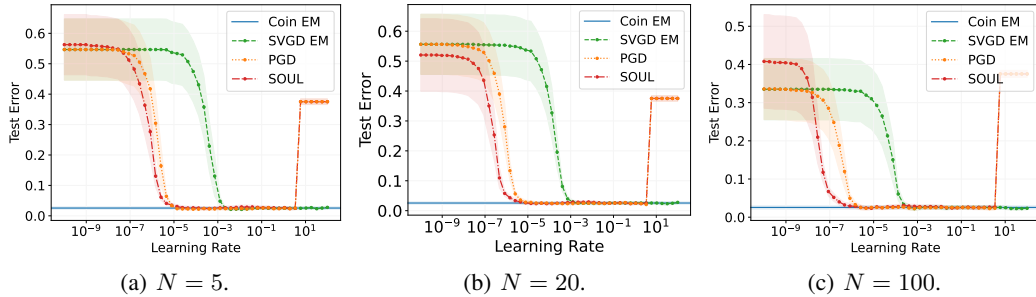


Figure 12: **Additional results for the Bayesian logistic regression in Sec. 4.2.** Test error as a function of the learning rate, for different numbers of particles.

### E.3 Bayesian logistic regression (alternative model)

We now present numerical results for the alternative Bayesian logistic regression model described in App. E.3. In particular, in Fig. 13, we compare SVGD EM and Coin EM against PGD and SOUL, plotting the area under the receiver operator characteristic curve (AUC) as a function of the learning rate, after running each algorithm with 10 particles for 1000 iterations. For each of the datasets considered, the predictive performance of Coin EM is similar to the performance of SVGD EM, PGD and SOUL with well tuned learning rates. In comparison to the Bayesian logistic regression studied in Section 4.2, here the predictive performance of the SVGD EM, PGD, and SOUL is rather more sensitive to the learning rate, particularly for the Covertypes dataset. In particular, in this case there is a much smaller range of values for which these methods exhibit performance on-par with, or superior to, Coin EM.

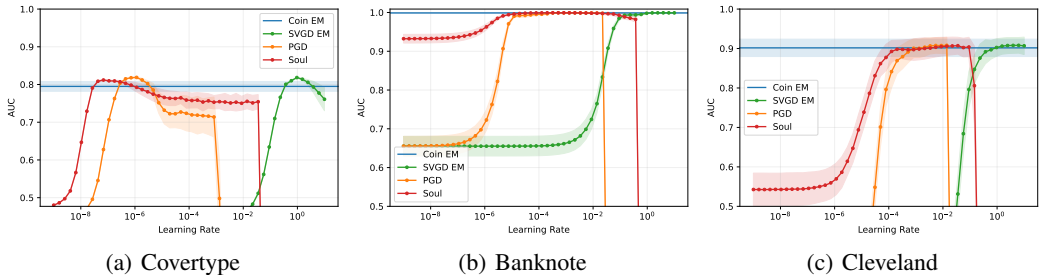


Figure 13: **Results for the alternative Bayesian logistic regression in App. E.3.** AUC as a function of the learning rate, averaged over 10 random test-train splits.

### F.4 Bayesian neural network

In this section, we present additional numerical results for the Bayesian neural network model in Sec. 4.3.

**Additional results for different datasets.** In Fig. 14, we compare the performance of Coin EM and SVGD EM on several additional UCI benchmarks, using the Bayesian neural network model considered in Sec. 4.3 (see also App. E.4). Once again, we compare our algorithms with PGD and SOUL. As noted in Sec. 4.3, in this case the performance of the three learning-rate-dependent algorithms (SVGD EM, PGD, SOUL) is highly sensitive to the choice of learning rate. In particular, there is generally a very small range of learning rates for which these algorithms outperform Coin EM. Given an optimal choice of learning rate, the best predictive performance of SVGD EM is generally comparable with the best predictive performance of SOUL, and better than the best predictive performance of PGD, though there are cases in which Coin EM and SVGD EM significantly outperform the competing methods (see Fig. 14(c)).

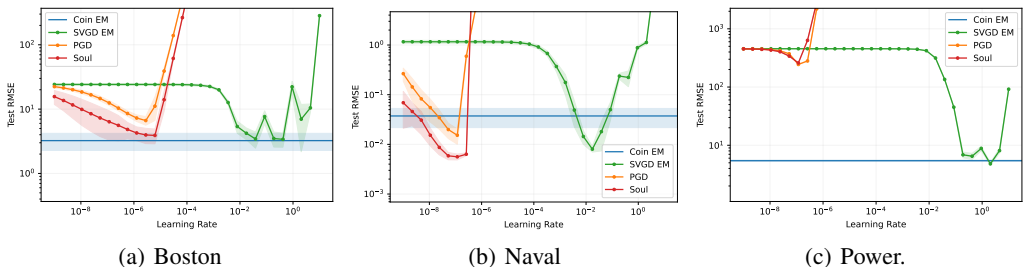


Figure 14: **Additional results for the Bayesian neural network in Sec. 4.3.** Root mean-squared-error (RMSE) as a function of the learning rate, for several UCI datasets, averaged over ten random test-train splits.

### F.5 Bayesian neural network (alternative model)

We now present numerical results for the alternative Bayesian neural network model described in App. E.5.

In Fig. 15, we plot the test error achieved by SVGD EM, PGD, and SOUL for several different choices of learning rate; and for several different choices of the number of particles. For comparison, we also plot the test error achieved by Coin EM. We also include results for SVGD EM, PGD, and SOUL when using a scaling heuristic recommended in [38, Sec. 2], which is designed to stabilize the updates and avoid ill-conditioning. We refer to these methods as SVGD EM', PGD', and SOUL'.

Unsurprisingly, all methods other than Coin EM are highly dependent on the choice of learning rate. While, in all cases, we observe that the convergence rate can be improved by increasing the learning rate, this approach can only go so far. In particular, if the learning rate is increased much beyond the largest values considered in Figure 15, then the updates are likely to become unstable (see, e.g., Fig. 15(m) - 15(o)).

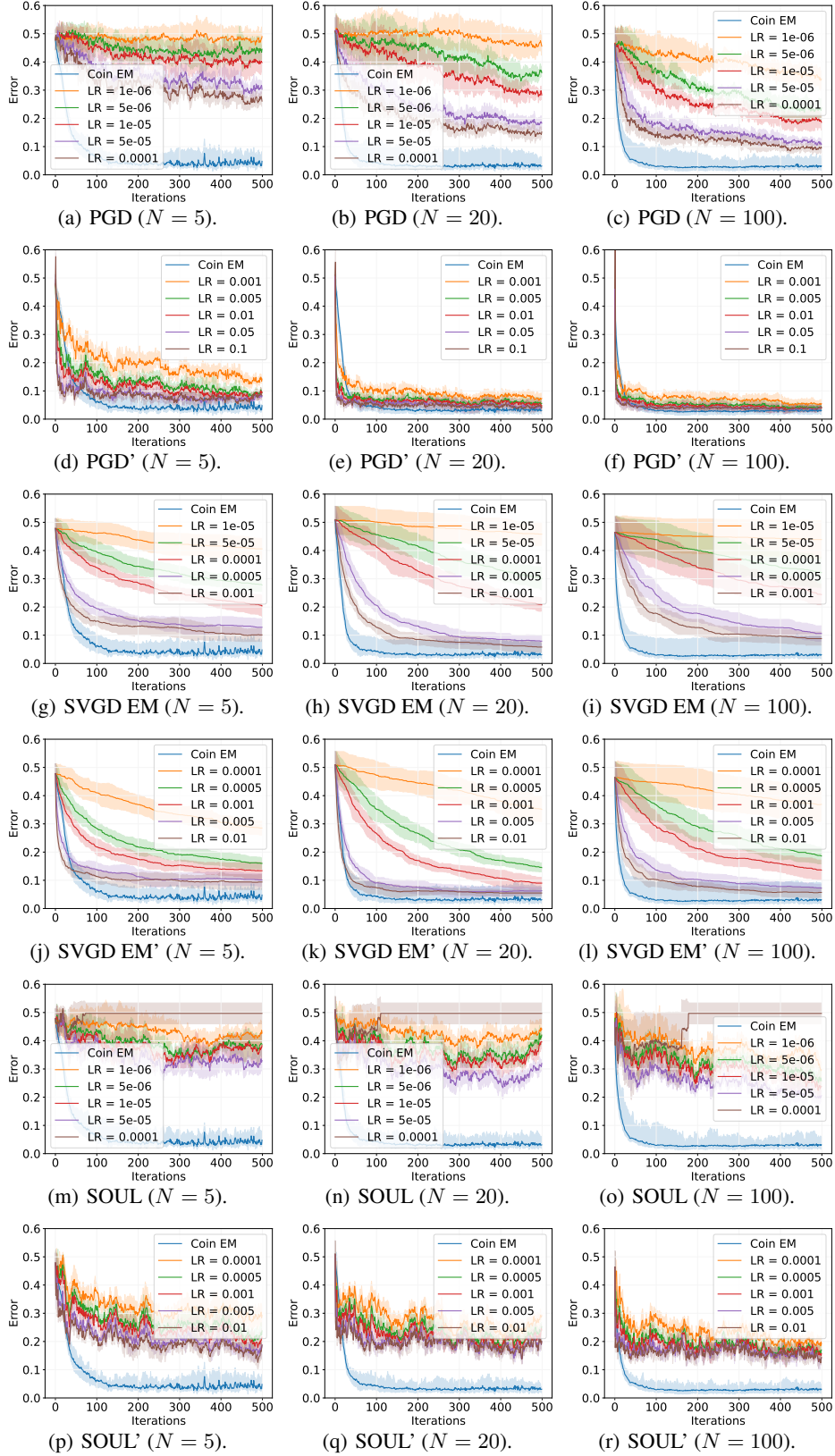


Figure 15: **Additional results for the alternative Bayesian neural network model in App. E.5: learning rate comparison.** Test error achieved by several methods over  $T = 500$  training iterations, for different learning rates. For reference, we also include the test error achieved by Coin EM.

In Fig. 16, we further investigate the performance of each algorithm as we vary the number of particles. In this case, each method is improved by increasing the number of particles. Interestingly, Coin EM is robust to the number of particles, achieving good predictive performance even when the number is small (Fig. 16(a)). The same is true, to a lesser extent, for SVGD EM (with or without the heuristic), which performs relatively well across experiments (Fig. 16(b), Fig. 16(c)). PGD observes a more significant performance increase as the number of particles is increased (Fig. 16(d)), consistent with the observations in [38, Sec. 3.2], although performs relatively well even for small particular numbers with the heuristic (Fig. 16(e)). Finally, even for large numbers of particles, and when using the heuristic, SOUL struggles to compete with the other methods (Fig. 16(f), Fig. 16(g)).

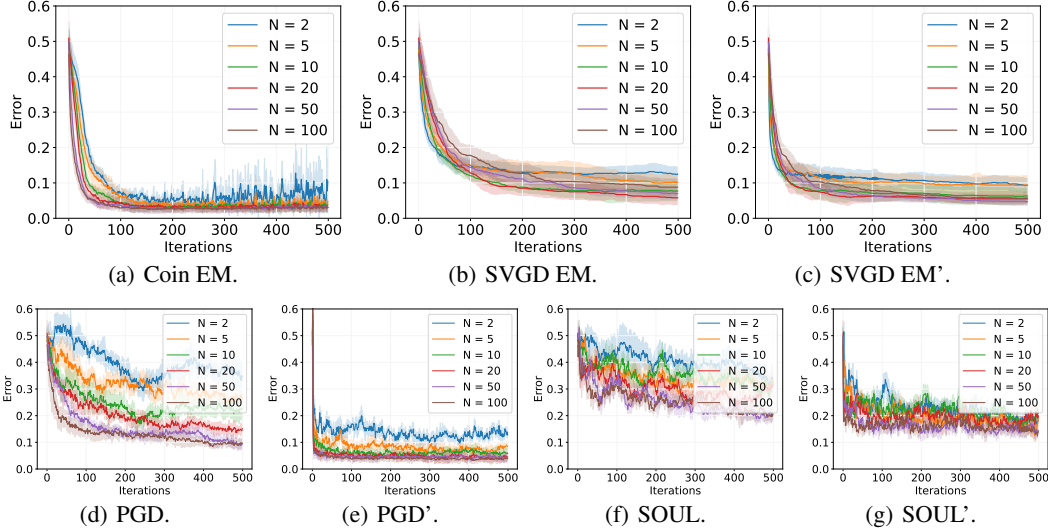


Figure 16: **Additional results for the alternative Bayesian neural network model in App. E.5: particle number comparison.** Test error over  $T = 500$  training iterations, for different numbers of particles. For all learning-rate dependent methods, we use the best learning rate as determined by Fig. 15.

In Fig. 17, we compare the predictive performance of all of the methods, now using the optimal learning rate according to the results in Fig. 15. We highlight three observations. First, Coin EM offers comparable performance to the best competing method, namely, PGD'. While the initial convergence of PGD' is typically faster, this difference tends to be relatively small. Meanwhile, the asymptotic predictive performance of Coin EM is generally slightly better, particularly for small numbers of particles. Second, SVGD EM (and SVGD EM') is bettered only by Coin EM and PGD'. Moreover, the scaling heuristic seems to have a smaller impact on SVGD EM than it does on PGD (i.e., SVGD EM and SVGD EM' are typically more similar than PGD and PGD'). Finally, the convergence of SVGD EM, and to a lesser extent Coin EM, are less 'noisy' than the convergence of PGD and SOUL. This is unsurprising, given that the latent variable updates in PGD and SOUL are stochastic, while the latent variable updates in SVGD EM and Coin EM are deterministic.

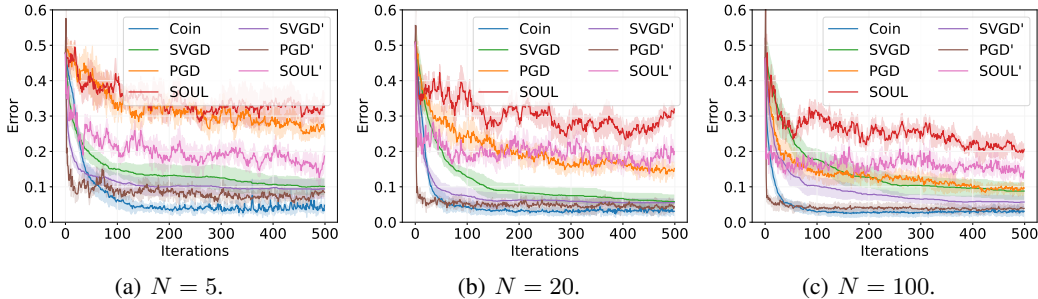


Figure 17: **Additional results for the alternative Bayesian neural network model in App. E.5: method comparison.** Test error over  $T = 500$  training iterations, for different numbers of particles. For all learning-rate dependent methods, we use the best learning rate as determined by Fig. 15.

Finally, in Fig. 18, we provide a comparison of the test accuracy achieved by Coin EM, SVGD EM, PGD, and SOUL, now over a much finer grid of learning rates.

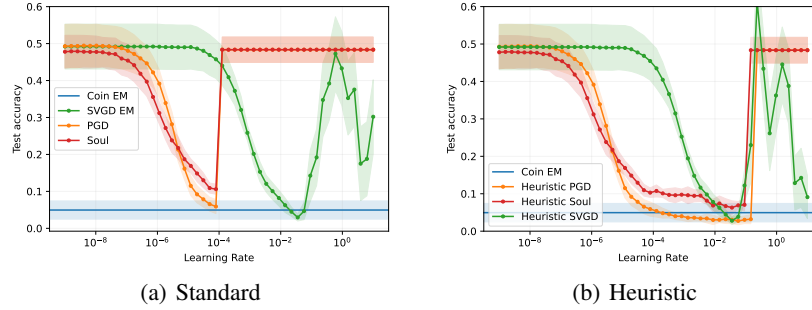


Figure 18: **Additional results for the Bayesian neural network in Sec. E.5.** Accuracy as a function of the learning rate, for the MNIST dataset, averaged over ten random test-train splits.

### F.6 Network model

In this section, we provide additional results for latent space network model studied in Sec. 4.4. In Fig. 19 - Fig. 21, we plot the mean of the particles  $\{z_T^i\}_{i=1}^N$  output by Coin EM (Fig. 19), PGD (Fig. 20), and SOUL (Fig. 21), using  $N = 10$  particles and  $T = 500$  iterations. In this case, each latent variable represents a network node, which corresponds to a Game of Thrones character. In each plot, Fig. (a) - (d) correspond to Series 1 - 4 of the TV series, respectively. The nodes are colour coded according to their cluster assignment from running DBScan [27]. In this case, only Coin EM is able to successfully capture the correct relationships between characters. We experimented with various learning rates for PGD and SOUL and were not able to improve the latent representation of the nodes to infer useful clusters among the characters. If additional covariate information were available, e.g. house labels such as Targaryen, Lannister, etc. then it is possible that the additional information could improve the latent representation for the PGD and SOUL algorithms.

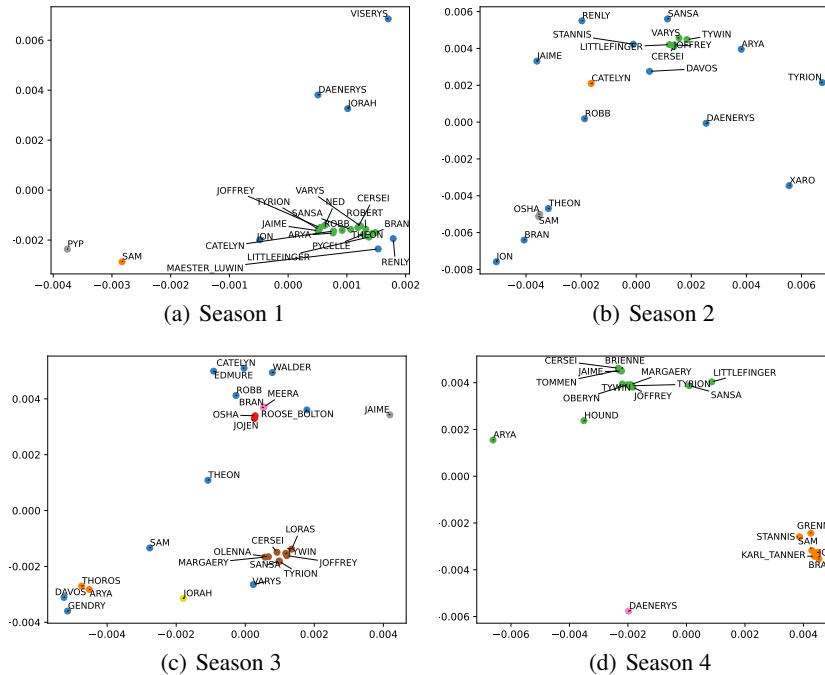


Figure 19: **Additional results for the network model in Sec. 4.4.** Posterior mean  $\frac{1}{N} \sum_{j=1}^N z_T^j$  of the particles generated by Coin EM after  $T = 500$  iterations.

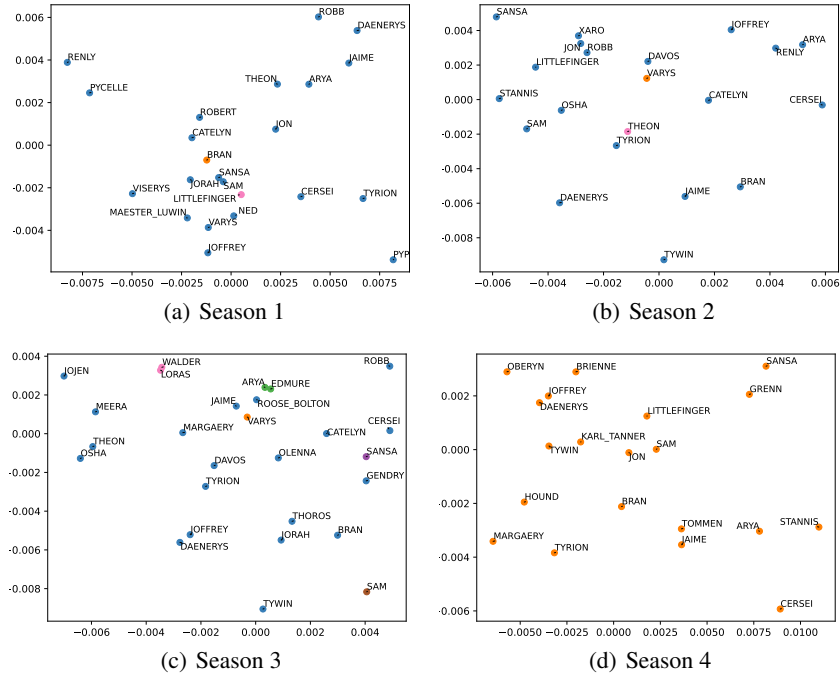


Figure 20: **Additional results for the network model in Sec. 4.4.** Posterior mean  $\frac{1}{N} \sum_{j=1}^N z_T^j$  of the particles generated by PGD after  $T = 500$  iterations.

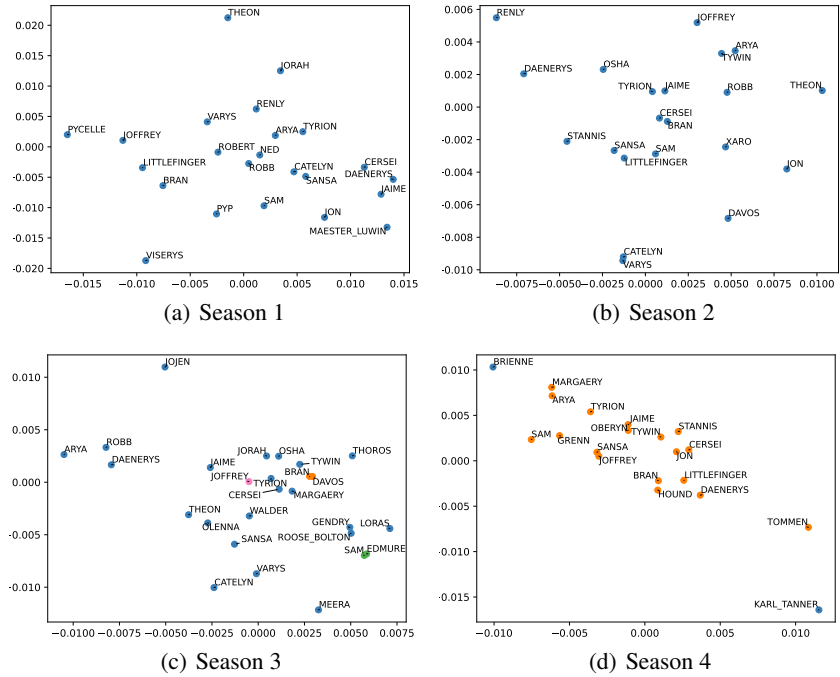


Figure 21: **Additional results for the network model in Sec. 4.4.** Posterior mean  $\frac{1}{N} \sum_{j=1}^N z_T^j$  of the particles generated by SOUL after  $T = 500$  iterations.