Amortised Learning by Wake-Sleep

Li K. Wenliang [1] **Theodore Moskovitz** [1] **Heishiro Kanagawa** [1] **Maneesh Sahani** [1]

Abstract

Models that employ latent variables to capture structure in observed data lie at the heart of many current unsupervised learning algorithms, but exact maximum-likelihood learning for powerful and flexible latent-variable models is almost always intractable. Thus, state-of-the-art approaches either abandon the maximum-likelihood framework entirely, or else rely on a variety of variational approximations to the posterior distribution over the latents. Here, we propose an alternative approach that we call amortised learning. Rather than computing an approximation to the posterior over latents, we use a wake-sleep Monte-Carlo strategy to learn a function that directly estimates the maximum-likelihood parameter updates. Amortised learning is possible whenever samples of latents and observations can be simulated from the generative model, treating the model as a "black box". We demonstrate its effectiveness on a wide range of complex models, including those with latents that are discrete or supported on non-Euclidean spaces.

1. Introduction

Many problems in machine learning, particularly unsupervised learning, can be approached by fitting flexible parametric probabilistic models to data, often based on "local" latent variables whose number scales with the number of observations. Once the optimal parameters are found, the resulting model may be used to synthesise samples, detect outliers, or relate observations to a latent "representation". The quality of all of these operations depends on the appropriateness of the model class chosen and the optimality of the identified parameters.

Although many fitting objectives have been explored in the literature, maximum-likelihood (ML) estimation remains prominent and comes with attractive theoretical properties, including consistency and asymptotic efficiency (Newey & McFadden, 1994). A challenge, however, is that analytic evaluation of the likelihoods of rich, flexible latent variable models is usually intractable. The Expectation-Maximisation (EM) algorithm (Dempster et al., 1977) offers one route to ML estimation in such circumstances, but it in turn requires an explicit calculation of (expected values under) the posterior distribution over latent variables, which also proves to be intractable in most cases of interest. Consequently, state-of-the-art ML-related methods almost always rely on approximations, particularly in large-data settings.

Denote the joint distribution of a generative model as $p_{\boldsymbol{\theta}}(\boldsymbol{z},\boldsymbol{x})$ where $\boldsymbol{z}$ is latent and $\boldsymbol{x}$ is observed, and $\boldsymbol{\theta}$ is the vector of parameters. EM breaks the ML problem into an iteration of two sub-problems. Given parameters $\boldsymbol{\theta}_t$ on the tth iteration, first find the posterior $p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})$; then maximise a lower bound to the likelihood that depends on this posterior to obtain $\boldsymbol{\theta}_{t+1}$. This bound is tight when computed using the correct posterior, ensuring convergence to a local mode of the likelihood.

The intractability of $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ forces some combination of Monte-Carlo estimation and the use of a tractable parametric approximating family which we call $q(\boldsymbol{z}|\boldsymbol{x})$ (Bishop, 2006). To avoid repeating the expensive optimisation in finding $q(\boldsymbol{z}|\boldsymbol{x})$ for each $\boldsymbol{x}$, amortised inference trains an encoding or recognition model, with parameters $\boldsymbol{\phi}$, to map from any $\boldsymbol{x}$ directly to an approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. Examples of amortised inference models include the Helmholtz machine (Dayan et al., 1995; Hinton et al., 1995) trained by the wake-sleep algorithm; and the variational auto-encoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) trained using reparamerisation gradient methods. With considerable effort on improving variational inference (reviewed in (Zhang et al., 2018)), complex and flexible generative models have been trained on large, high-dimensional datasets.

However, approximate variational inference poses at least three challenges. First, the parametric form of the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$, and particularly any factorisations assumed, must be crafted for each model. Second, methods such as reparameterisation require specific transformations tailored to the type of latent variables, whether they are continuous or discrete, and whether or not the support is

[1]Gatsby Computational Neuroscience Unit. Correspondence to: Li K. Wenliang <kevinli@gatsby.ucl.ac.uk>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

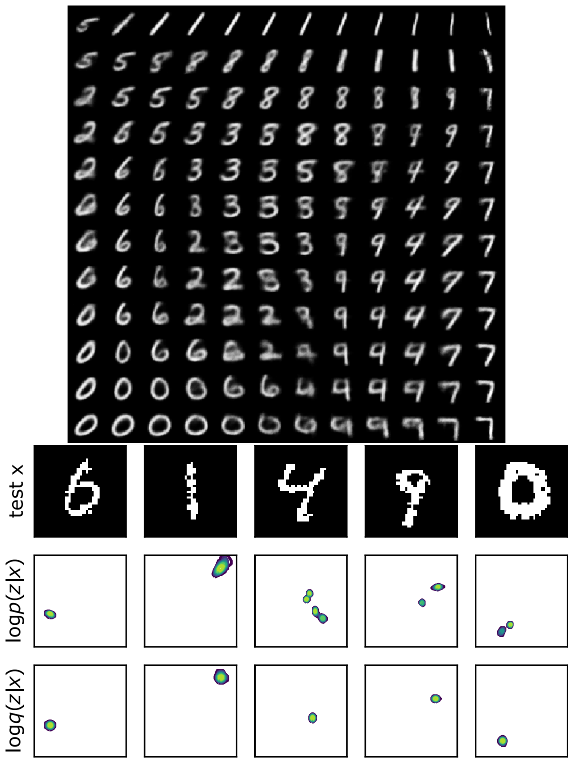

Figure 1. VAE trained on binarised MNIST digits. Top: mean images generated by decoding points on a grid of 2-D latent variables. Bottom three rows show five samples of real MNSIT digit (top), the corresponding true posteriors (middle) found by histogram and the approximate posteriors computed by the encoder.

Euclidean. Third, given a flexible generative model, such as one with conditional dependence modelled using neural networks, the true posteriors may be irregular in ways that are difficult to approximate. We illustrate this latter effect using a standard VAE with two-dimensional $\boldsymbol{z}$ trained on binarised MNIST digits (Figure 1). The exact posterior may be distorted or multi-modal, even though only Gaussian posteriors are ever produced by the encoder.

When inference is only approximate, the M-step of EM may not increase the likelihood, and so approximate methods usually converge away from the ML parameter values. The dependence of learnt parameters on the quality of the posterior approximation is not straightforward, and the error may not be reduced by (say) approximations with lower Kullback-Leibler (KL) divergence (Turner & Sahani, 2011); indeed errors in posterior statistics that enter the objective function may be unbounded (Huggins et al., 2019).

Here, we propose a novel approach to ML learning in flexible latent variable models that avoids the complications of posterior estimation, instead learning to predict the gradient of the likelihood directly—an approach we call *amortised learning*. The particular realisation we develop here, amortised learning by wake sleep (ALWS), requires only that sampling from the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ be possible, and that the gradient $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ be available (possibly by automated methods), but otherwise does not make assumptions about the latent variable form or distribution. We test the performance of ALWS on a wide range of tasks and models, including hierarchical models with heterogeneous priors, nonlinear dynamical systems, and deep models of images. All experiments use the same form of gradient model trained by simple least-squares regression. For image generation, we find that models trained with ALWS can produce samples of considerably better quality than those trained using algorithms based on variational inference.

2. Background

2.1. Model Definition

Consider a probabilistic generative model with parameter vector $\boldsymbol{\theta}$ that defines a prior on latents $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and a conditional on observations $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. In ML learning, we seek parameters that maximise the log (marginal) likelihood

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \log \int p_{\boldsymbol{\theta}}(\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \tag{1}$$

averaged over a set of i.i.d. data $\mathcal{D} = \{\boldsymbol{x}_m^*\}_{m=1}^M$. One approach is to iteratively update $\boldsymbol{\theta}$ by following the gradient

$$\Delta_{\boldsymbol{\theta}}(\boldsymbol{x}) := \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{2}$$

at each iteration[1]

2.2. Variational Inference for Learning

For many models of interest, the integral in (1) cannot be evaluated analytically, and so direct computation of the gradient is intractable. A popular alternative is to maximise a variational lower bound on the marginal likelihood defined by a distribution $q(\boldsymbol{z})$:

$$\mathcal{F}(q, \boldsymbol{\theta}) := \mathbb{E}_{q(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] + \mathbb{H}[q] \le \log p_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{3}$$

where $\mathbb{H}[q]$ is the entropy of q. Thus, the parameter $\boldsymbol{\theta}$ can be updated by following the gradient of $\mathcal{F}(q, \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{F}(q, \boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{q(\boldsymbol{z})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})] \\ &= \mathbb{E}_{q(\boldsymbol{z})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})]. \end{aligned} \tag{4}$$

When $q(\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, the lower bound in (3) is tight, and the gradient in (4) is equal to that of the likelihood (see Appendix A.3). Variational approximations attempt to bring q close to $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$, usually by seeking to minimise $D_{\mathrm{KL}}[q(\boldsymbol{z})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})]$ (which corresponds to maximising the bound $\mathcal{F}$ w.r.t. q). However, although minimising

[1]We define the likelihood gradient for a single data point here and throughout; an actual update will typically follow the gradient averaged over i.i.d data.

$D_{\mathrm{KL}}[q(\boldsymbol{z})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})]$ over q ensures consistent optimisation of a single objective, the resulting gradient in (4) will often be a poor approximation to the likelihood gradient (2).

2.3. Conditional Expectation and LSR

Our approach is to avoid the difficulties introduced by approximating $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ with $q(\boldsymbol{z})$ in (4), and instead estimate the conditional expectation directly using least-squares regression (LSR). Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be random vectors with a joint distribution $\rho(\boldsymbol{x}, \boldsymbol{y})$ on $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. In LSR, we seek a (vector-valued) function $\boldsymbol{f}$ that achieves the lowest mean squared error (MSE) $\mathbb{E}_{\rho(\boldsymbol{x},\boldsymbol{y})}\left[\|\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})\|_2^2\right]$. The ideal solution is given by $\boldsymbol{f}_\rho(\boldsymbol{x}) := \mathbb{E}_{\rho(\boldsymbol{y}|\boldsymbol{x})}[\boldsymbol{y}]$, as the problem can be cast as the minimisation of $\mathbb{E}_{\rho(\boldsymbol{x})}\left[\|\boldsymbol{f}_\rho(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x})\|_2^2\right]$, where $\rho(\boldsymbol{x})$ is the marginal distribution of $\boldsymbol{x}$ (see Appendix A.1). Note that $\boldsymbol{f}_\rho(\boldsymbol{x})$ takes a similar form as the desired (4). In practice, the distribution $\rho(\boldsymbol{x}, \boldsymbol{y})$ is known only through a sample $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N \overset{\text{i.i.d.}}{\sim} \rho(\boldsymbol{x}, \boldsymbol{y})$; thus, LSR can be understood to seek a good approximation of $\boldsymbol{f}_\rho$ based on the sample.

2.4. Kernel Ridge Regression

In LSR, as the target $\boldsymbol{f}_\rho$ is unknown, it is desirable to construct an estimate without imposing restrictions on its form. Kernel ridge regression (KRR) is a nonlinear regression method that draws the estimated regression function from a flexible class of functions called a reproducing-kernel Hilbert space (RKHS) (Hofmann et al., 2008). The KRR estimator is found by minimising the regularised empirical risk

$$\min_{\boldsymbol{f}\in\mathcal{H}} \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{y}_n - \boldsymbol{f}(\boldsymbol{x}_n)\|_2^2 + \lambda\|\boldsymbol{f}\|_{\mathcal{H}}^2, \tag{5}$$

where $\lambda > 0$ is a regularisation parameter, and $\mathcal{H}$ is the RKHS corresponding to a matrix-valued kernel $\kappa : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}^{d_y \times d_y}$ (Carmeli et al., 2006). The solution can be found conveniently in closed-form, which allows a further simplification detailed in Section 3.2. In this paper, we use a kernel of the form $\kappa(x, x') = k(x, x')\boldsymbol{I}_y$, where $\boldsymbol{I}_y$ is the identity matrix, and k is a scalar-valued positive definite kernel; therefore, the matrix-valued kernel κ can be identified with its scalar counterpart k. In particular, in the scalar output case $d_y = 1$, this choice of κ coincides with KRR with the scalar kernel k. Importantly, the closed-form solution $\hat{\boldsymbol{f}}_\lambda$ of KRR in (5) can be expressed as

$$\hat{\boldsymbol{f}}_\lambda(\boldsymbol{x}^*) = \mathbf{Y}(\boldsymbol{K} + N\lambda\boldsymbol{I}_N)^{-1}\boldsymbol{k}^*, \tag{6}$$

where $\mathbf{Y}$ is the concatenation of the training targets $[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{R}^{d_y \times N}$, $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is the gram matrix whose element is $(\boldsymbol{K})_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{I}_N$ is the identity matrix and $\boldsymbol{k}^* = (k(\boldsymbol{x}_i, \boldsymbol{x}^*))_{i=1}^N \in \mathbb{R}^N$ for a test point $\boldsymbol{x}^*$.

In the limit of $N \to \infty$ and $\lambda \to 0$, the solution $\hat{\boldsymbol{f}}_\lambda$ will achieve the minimum MSE in the RKHS (Caponnetto & De Vito, 2007). In general, the target $\boldsymbol{f}_{\boldsymbol{\rho}}$ may not be in the RKHS[2]; nonetheless, if the RKHS is sufficiently rich (or C_0 universal (Carmeli et al., 2010)), the error made by the estimator $\mathbb{E}_{\rho(\boldsymbol{x})}\left[\|\hat{\boldsymbol{f}}_\lambda(\boldsymbol{x}) - \boldsymbol{f}_\rho(\boldsymbol{x})\|_2^2\right]$ will converge to zero (Szabó et al., 2016, Theorem 7).

3. Amortised Learning by Wake-Sleep

3.1. Gradient of Log-Likelihood

As stated above and derived in Appendix A.3, the log-likelihood gradient function evaluated on observation $\boldsymbol{x}$ at iteration t (with current parameters $\boldsymbol{\theta}_t$) can be written

$$\begin{aligned}\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x}) &= \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}_t}(\boldsymbol{x})\big|_{\boldsymbol{\theta}_t} \\ &= \nabla_{\boldsymbol{\theta}} \mathcal{F}(p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x}), \boldsymbol{\theta})\big|_{\boldsymbol{\theta}_t},\end{aligned} \tag{7}$$

where the gradient in the second line is taken w.r.t. the second argument of $\mathcal{F}$; the posterior distribution is for a fixed $\boldsymbol{\theta}$ at the current $\boldsymbol{\theta}_t$.

We want to directly estimate of this gradient without explicit computation of the posterior. Inserting the definition from (4) into (7) we have,

$$\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x}) = \mathbb{E}_{p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})\big|_{\boldsymbol{\theta}_t}\right] \tag{8}$$

$$\begin{aligned}&= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})]\big|_{\boldsymbol{\theta}_t} \\ &= \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}(\boldsymbol{x})\big|_{\boldsymbol{\theta}_t}.\end{aligned} \tag{9}$$

where $J_{\boldsymbol{\theta}}(\boldsymbol{x}) := \mathbb{E}_{p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})]$. Note that the function $J_{\boldsymbol{\theta}}(\boldsymbol{x})$ changes with iteration due to the dependence on $p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})$. It can be regarded as an instantaneous objective for ML learning starting from $\boldsymbol{\theta}_t$. Neither (8) nor (9) can be computed in closed form, and therefore need to be estimated. We refer to ML learning via the estimation of $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ either through $J_{\boldsymbol{\theta}}$ by (9) or directly by (8) as amortised learning. The difference between the two equations lies purely in implementation: The former estimates the high-dimensional $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ directly, whereas the latter implements the same computation by differentiating $J_{\boldsymbol{\theta}}(\boldsymbol{x})$. We term an estimator of $J_{\boldsymbol{\theta}}$ a *gradient model*, as it retains information about $\boldsymbol{\theta}$ and is used to estimate the gradient $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$. In the next section, we develop a concrete instantiation of amortised learning.

3.2. Training KRR Gradient Model by Wake-Sleep

As discussed in Section 2.3, LSR allows us to estimate the conditional expectation of an output variable given an input. Thus, although the gradient in (8) (or in (9)) involves an intractable conditional expectation, we can obtain an estimate of the gradient $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ by regressing from $\boldsymbol{x}$ to

[2]In this case, $\boldsymbol{f}_{\boldsymbol{\rho}}$ is only assumed to be square-integrable with respect to $\boldsymbol{\rho}$

$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ (or $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$). Any reasonable regression model, e.g., a neural network, could serve this purpose, but here we choose to use KRR introduced in Section 2.4. Other possible forms of gradient model are discussed in Appendix B.1.

The expression in (8) leads to the following LSR problem

$$\min_{\boldsymbol{f} \in \tilde{\mathcal{H}}} \frac{1}{N} \sum_{n=1}^{N} \|\nabla_{\boldsymbol{\theta}}(y_{\boldsymbol{\theta},n})\big|_{\boldsymbol{\theta}_t} - \boldsymbol{f}(\boldsymbol{x}_n)\|_2^2 + \lambda \|\boldsymbol{f}\|_{\tilde{\mathcal{H}}}^2, \quad (10)$$

where $y_{\boldsymbol{\theta},n} = \log p_{\boldsymbol{\theta}}(\boldsymbol{z}_n, \boldsymbol{x}_n)$, $\tilde{\mathcal{H}}$ is an RKHS and $\{(\boldsymbol{z}_n, \boldsymbol{x}_n)\}_{n=1}^N \sim p_{\boldsymbol{\theta}_t}$. Brehmer et al. (2020) also noticed that log-likelihood gradient can be obtained by LSR. However, regressing to a vector-valued $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}$ can be expensive, and evaluating the target $y_{\boldsymbol{\theta},n}$ on all $(\boldsymbol{z}_n, \boldsymbol{x}_n)$ is slow. Alternatively, we can use (9) and find an estimator for the scalar-valued $J_{\boldsymbol{\theta}}$ that keeps the dependence on $\boldsymbol{\theta}$ and then evaluate its gradient by automatic differentiation. Thus, we construct an estimator by

$$\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}} = \arg\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} |y_{\boldsymbol{\theta},n} - f(\boldsymbol{x}_n)|^2 + \lambda \|f\|_{\mathcal{H}}^2, \quad (11)$$

where $\mathcal{H}$ is the RKHS induced by a kernel $k_{\boldsymbol{\omega}}(\cdot,\cdot)$ with hyperparameters $\boldsymbol{\omega}$, and $\boldsymbol{\gamma} = \{\boldsymbol{\omega}, \lambda\}$. For each data point $\boldsymbol{x}^* \in \mathcal{D}$, the estimate of $J_{\boldsymbol{\theta}}(\boldsymbol{x}^*)$ is

$$\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\boldsymbol{x}^*) = \boldsymbol{\alpha}_{\boldsymbol{\theta},\boldsymbol{\gamma}} \cdot \boldsymbol{k}_{\boldsymbol{\omega}}^*, \quad (12)$$

$$\boldsymbol{\alpha}_{\boldsymbol{\theta},\boldsymbol{\gamma}} = \boldsymbol{y}_{\boldsymbol{\theta}} \left(\boldsymbol{K}_{\boldsymbol{\omega}} + \lambda N \boldsymbol{I}_N\right)^{-1}, \quad (\boldsymbol{y}_{\boldsymbol{\theta}})_n = \log p_{\boldsymbol{\theta}}(\boldsymbol{z}_n, \boldsymbol{x}_n)$$
$$K_{\boldsymbol{\omega},i,j} = k_{\boldsymbol{\omega}}(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad k_{\boldsymbol{\omega},j}^* = k_{\boldsymbol{\omega}}(\boldsymbol{x}_j, \boldsymbol{x}^*)$$

where $\boldsymbol{I}_N$ is the identity matrix of size $N \times N$. Note that the dependence of $\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}$ on $\boldsymbol{\theta}$ is only through evaluations of $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ on samples drawn from $p_{\boldsymbol{\theta}_t}$ for fixed $\boldsymbol{\theta} = \boldsymbol{\theta}_t$. The gradient $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ is then estimated as

$$\hat{\Delta}_{\boldsymbol{\theta}_t,\boldsymbol{\gamma}}(\boldsymbol{x}) := \nabla_{\boldsymbol{\theta}} \hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\boldsymbol{x})\big|_{\boldsymbol{\theta}_t}.$$

In general, a good estimator of $J_{\boldsymbol{\theta}}$ may not yield a reliable estimate of its gradient $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$; however, for the KRR estimate, taking the derivative of $\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}$ w.r.t. $\boldsymbol{\theta}$ is equivalent to replacing $\boldsymbol{y}_{\boldsymbol{\theta}}$ in (12) with $\nabla_{\boldsymbol{\theta}}(\boldsymbol{y}_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}_t}$, which is the solution for the optimisation in (10), with $\tilde{\mathcal{H}}$ being a vector-valued RKHS given by a kernel $\kappa_{\boldsymbol{\omega}} = k_{\boldsymbol{\omega}} \boldsymbol{I}$ (see Section 2.4). We show in Appendix A.2 that, under mild conditions, the target of the regression $\mathbb{E}_{p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})}\left[\nabla_{\boldsymbol{\theta}} y_{\boldsymbol{\theta},n}\big|_{\boldsymbol{\theta}_t}\right]$ is square-integrable under $p_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ for common generative models.

In summary, learning proceeds according to the following wake-sleep procedure: at the tth step when $\boldsymbol{\theta} = \boldsymbol{\theta}_t$, the gradient model is first trained using "sleep samples" $(\boldsymbol{z}_n, \boldsymbol{x}_n) \sim p_{\boldsymbol{\theta}_t}$ and evaluations $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_n, \boldsymbol{x}_n)$, keeping the dependence on $\boldsymbol{\theta}$; then the gradient model is applied to real data ("wake" samples) $\boldsymbol{x}^* \in \mathcal{D}$ to produce $\hat{\Delta}_{\boldsymbol{\theta}_t,\boldsymbol{\gamma}}(\boldsymbol{x}^*)$ by differentiating $\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}$ and evaluating at $\boldsymbol{\theta}_t$. See Algorithm 1. Two points are worth emphasis: (a) The algorithm does not require explicit computation or approximation of the posterior, and (b) We only need samples from the model $p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ and differentiable evaluations of $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$.

3.3. Exponential Family Conditionals

In many common models, the conditional $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ lies in the exponential family (e.g. Gaussian, Bernoulli), and we can exploit this structure to simplify the estimation of $J_{\boldsymbol{\theta}}$. In this case, the log joint can be written as

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x}) &= \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ &= \boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{z}) \cdot \boldsymbol{s}(\boldsymbol{x}) - \log Z_{\boldsymbol{\theta}}(\boldsymbol{z}) + \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ &= \boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{z}) \cdot \boldsymbol{s}(\boldsymbol{x}) - \Psi_{\boldsymbol{\theta}}(\boldsymbol{z})\end{aligned}$$

where $\boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{z})$, $\boldsymbol{s}(\boldsymbol{x})$ and $Z_{\boldsymbol{\theta}}(\boldsymbol{z})$ are, respectively, the natural parameter, sufficient statistics and normaliser of the likelihood, and $\Psi_{\boldsymbol{\theta}} := \log Z_{\boldsymbol{\theta}}(\boldsymbol{z}) - \log p_{\boldsymbol{\theta}}(\boldsymbol{z})$. By taking the posterior expectation, $J_{\boldsymbol{\theta}}(\boldsymbol{x})$ in (9) becomes

$$J_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}_t}}[\boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{z})]}_{\boldsymbol{h}_{\boldsymbol{\theta}}^{\boldsymbol{\eta}}(\boldsymbol{x})} \cdot \boldsymbol{s}(\boldsymbol{x}) - \underbrace{\mathbb{E}_{p_{\boldsymbol{\theta}_t}}[\Psi_{\boldsymbol{\theta}}(\boldsymbol{z})]}_{h_{\boldsymbol{\theta}}^{\Psi}(\boldsymbol{x})} \quad (13)$$

where $p_{\boldsymbol{\theta}_t}$ stands for $p_{\boldsymbol{\theta}_t}(\boldsymbol{z}|\boldsymbol{x})$. Therefore, for exponential family likelihoods, the regression to $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ in (11) can be replaced by two separate regressions to $\boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $\Psi_{\boldsymbol{\theta}}(\boldsymbol{z})$, which are functions of $\boldsymbol{z}$ alone. The resulting estimators $\hat{\boldsymbol{h}}_{\boldsymbol{\theta},\boldsymbol{\gamma}}^{\boldsymbol{\eta}}$ and $\hat{\boldsymbol{h}}_{\boldsymbol{\theta},\boldsymbol{\gamma}}^{\Psi}$ are combined to yield

$$\hat{\Delta}_{\boldsymbol{\theta}_t,\boldsymbol{\gamma}}(\boldsymbol{x}) = \nabla_{\boldsymbol{\theta}} \left[\hat{\boldsymbol{h}}_{\boldsymbol{\theta},\boldsymbol{\gamma}}^{\boldsymbol{\eta}}(\boldsymbol{x}) \cdot \boldsymbol{s}(\boldsymbol{x})\right]\Big|_{\boldsymbol{\theta}_t} - \nabla_{\boldsymbol{\theta}} \hat{\boldsymbol{h}}_{\boldsymbol{\theta},\boldsymbol{\gamma}}^{\Psi}(\boldsymbol{x})|_{\boldsymbol{\theta}_t},$$

where the Jacobian vector product applies to the first term.

3.4. Kernel Structure and Learning

The kernel $k_{\boldsymbol{\omega}}$ used in the gradient model affects how well $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ is estimated. It can be made more flexible by augmenting with a neural network as in (Wilson et al., 2016; Wenliang et al., 2019)

$$k_{\boldsymbol{\omega}}(\boldsymbol{x}, \boldsymbol{x}') = \varkappa_{\boldsymbol{\sigma}}(\boldsymbol{\psi}_{\boldsymbol{v}}(\boldsymbol{x}), \boldsymbol{\psi}_{\boldsymbol{v}}(\boldsymbol{x}'))$$

where $\varkappa_{\boldsymbol{\sigma}}$ is a standard kernel (e.g. exponentiated-quadratic) with parameter $\boldsymbol{\sigma}$ (e.g. bandwidth), and $\boldsymbol{\psi}_{\boldsymbol{v}}$ is a neural network with parameter $\boldsymbol{v}$, so $\boldsymbol{\omega} = \{\boldsymbol{\sigma}, \boldsymbol{v}\}$. Other details of the kernel structure are described in Appendix B.2.

The gradient model parameter $\boldsymbol{\gamma} = \{\boldsymbol{\omega}, \lambda\}$ can be learned to further minimise the MSE in (11) using a scheme of cross-validation by gradient descent (Wenliang et al., 2019). Specifically, we generate two sets of sleep samples from $p_{\boldsymbol{\theta}}$; we use one set to compute $\boldsymbol{\alpha}_{\boldsymbol{\theta},\boldsymbol{\gamma}}$ in closed form; then, on the other set $\{(\boldsymbol{z}'_l, \boldsymbol{x}'_l)\}_{l=1}^L$, we compute the MSE between the estimator $\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\boldsymbol{x}'_l)$ and the ground truth value $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}'_l, \boldsymbol{x}'_l)$, and minimise this by gradient descent on $\boldsymbol{\gamma}$. The full ALWS procedure is presented in Algorithm 1.

Algorithm 1: Amortised learning by wake sleep

```
input :Dataset $\mathcal{D}$, gradient model parameters $\boldsymbol{\gamma}$,
       generative model $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$, or $\boldsymbol{\eta}_{\boldsymbol{\theta}}$ and $\Psi_{\boldsymbol{\theta}}$
       with parameters $\boldsymbol{\theta}$ initialised s.t. $p_{\boldsymbol{\theta}}(\boldsymbol{x})$
       covers/dominates the data distribution, max
       epoch and any convergence criteria.
while $\boldsymbol{\theta}$ not converged within max epoch do
    Sleep phase: train gradient model
     Sample $\{\boldsymbol{z}_n, \boldsymbol{x}_n\}_{n=1}^N \sim p_{\boldsymbol{\theta}}$
    if $p(\boldsymbol{x}|\boldsymbol{z})$ is not in exponential family then
        Find $\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\cdot)$ by computing $\boldsymbol{\alpha}_{\boldsymbol{\theta},\boldsymbol{\gamma}}$ in (12)
    else
        Find $\hat{\boldsymbol{h}}^{\boldsymbol{\eta}}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\cdot)$ and $\hat{\boldsymbol{h}}^{\Psi}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\cdot)$ similar to (12)
        $\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\cdot) = \hat{\boldsymbol{h}}^{\boldsymbol{\eta}}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\cdot) \cdot \boldsymbol{s}(\boldsymbol{x}) - \hat{\boldsymbol{h}}^{\Psi}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\cdot)$ in (13)
    Sleep phase: update $\boldsymbol{\gamma}$
     Sample $\{\boldsymbol{z}'_l, \boldsymbol{x}'_l\}_{l=1}^L \sim p_{\boldsymbol{\theta}}$
     Compute $d_l := \log p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$
     Compute $\mathcal{E}_{\boldsymbol{\gamma}} = \frac{1}{L}\sum_{l=1}^L(\hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\boldsymbol{x}'_l) - d_l)^2$
     Update $\boldsymbol{\gamma} \propto \nabla_{\boldsymbol{\gamma}}\mathcal{E}_{\boldsymbol{\gamma}}$
    Wake phase: update $\boldsymbol{\theta}$
     Sample $\{\boldsymbol{x}^*_m\}_{m=1}^M \in \mathcal{D}$
     $\bar{J}_{\boldsymbol{\theta}} = \frac{1}{M}\sum_i^M \hat{J}_{\boldsymbol{\theta},\boldsymbol{\gamma}}(\boldsymbol{x}^*_m)$
     Update $\boldsymbol{\theta} \propto \nabla_{\boldsymbol{\theta}}\bar{J}_{\boldsymbol{\theta}}$
end
return : $\boldsymbol{\theta}$
```

3.5. Dealing with Covariate Shift

The gradient model is to be used to estimate $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ on $\boldsymbol{x}^*$ drawn from an underlying data distribution p^*, but it is trained using sleep samples from $p_{\boldsymbol{\theta}_t}$. This mismatch in input data distribution for training and evaluation is known as covariate shift (Shimodaira, 2000).

Here, to ensure that the gradient model performs reasonably well on p^*, we initialise $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ to be overdispersed relative to p^* by setting a large noise in $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$. Since ML estimation minimises $D_{\mathrm{KL}}[p^* \| p_{\boldsymbol{\theta}}]$, which penalises a distribution $p_{\boldsymbol{\theta}}$ that is narrower than p^*, we expect the noise to continue to cover the data before the model is well trained. For image data only, we also apply batch normalisation in $\boldsymbol{\psi}_{\boldsymbol{w}}$ of the kernel. We find these simple remedies to be effective, though other more principled methods, such as kernel mean matching (Gretton et al., 2009) and binary classification (Gutmann & Hyvärinen, 2010; Goodfellow et al., 2014), may further improve the results.

4. Experiments

We evaluate ALWS on a wide range of generative models. Details for each experiment can be found in Appendix C. [3]

[3]Code is at github.com/kevin-w-li/al-ws

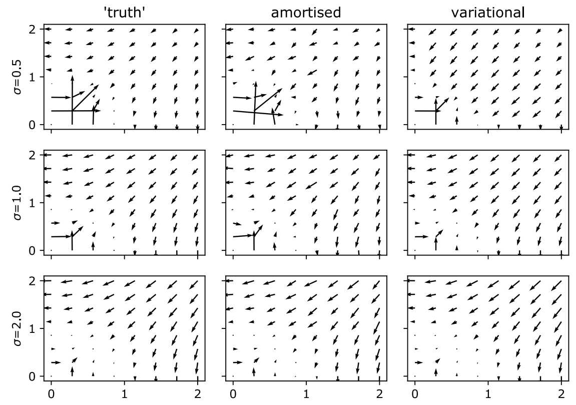

Figure 2. Gradient estimated using amortised learning and variational inference. The true gradients are approximated by importance sampling.

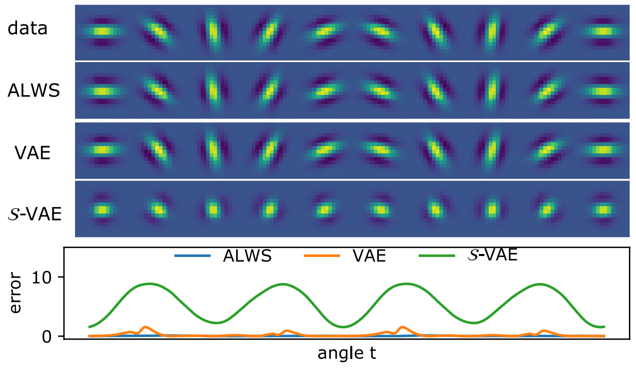

Figure 3. Learning to generate Gabor filters given a 1-D circular uniform prior. Top images show samples generated by latents separated by fixed rotation on the circle. For VAE, a 2-D Gaussian prior was used, and the images are generated by latents on the unit circle. $\mathcal{S}$-VAEs cannot reliably learn the filters. The errors below show the squared distance between generated images and data at each orientation. For each method, an angle offset and direction are chosen to minimise the total error.

4.1. Parameter Gradient Estimation

First, we demonstrate that KRR can estimate $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ well on a simple toy generative model described by

$$z_1, z_2 \sim \mathcal{N}(0,1),\ x|\boldsymbol{z} \sim \mathcal{N}(\text{softplus}(\boldsymbol{b}\cdot\boldsymbol{z}) - \|\boldsymbol{b}\|_2^2, \sigma_x^2).$$

The training data are 100 data points from the model given $\boldsymbol{b} = [1,1], \sigma_x = 0.1$. we estimate the gradients of the log-likelihood w.r.t. $\boldsymbol{b}$ evaluated at a grid of $\boldsymbol{b}$ by ALWS, and compare them to estimates using importance sampling ("truth") and a factorised Gaussian posterior that minimises the forward KL for each $\boldsymbol{x}$. For ALWS, we used a Gaussian kernel with a bandwidth equal to the median distance between samples generated for each $\boldsymbol{b}$, and set $\lambda = 0.01$. For variational inference, we assumed a factorised Gaussian posterior for each sample of $\boldsymbol{x}$, and optimise posterior pa-

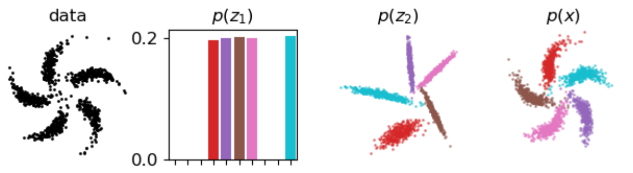

Figure 4. Learning hierarchical model with discrete and continuous latents. From left to right: data sample, component probabilities, samples of the first latent distribution and samples of generated data. Colours correspond to different components

rameters until convergence. ALWS tends to estimate better, especially for small $\boldsymbol{b}$ (Figure 2). For the smallest σ_x, the KRR estimates are noisier, whereas variational inference introduces greater bias.

4.2. Non-Euclidean Priors

The prior $p(\boldsymbol{z})$ may capture special topological structure in the data. For instance, a prior over the hypersphere can be used to describe circular features (Davidson et al., 2018; Xu & Durrett, 2018). Training models with such a prior is straightforward using ALWS, while learning by amortised inference requires special reparameterisation for a posterior on the hypersphere, such as the von-Mises Fisher (vMF) used in the $\mathcal{S}$-VAE (Davidson et al., 2018; Xu & Durrett, 2018). We fit a model with uniform circular latent and neural-network output:

$$\boldsymbol{z} = [\cos(a), \sin(a)], \quad p(a) = \mathcal{U}(a; (-\pi, \pi)),$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \mathrm{NN}_{\boldsymbol{w}}(\boldsymbol{z}), \sigma_x^2 \boldsymbol{I}),$$

(where $\mathcal{U}$ is a uniform distribution) on a data set of Gabor wavelets with uniformly distributed orientations. As shown in Figure 3, ALWS learns to generate images that closely resemble the training data. A fixed rotation around the latent circle corresponds to almost a fixed rotation of the Gabor wavelet in the image. The VAE with a 2-D Gaussian latent also generates good filters given latents on the circle, but the length of the filter varies with rotation. Surprisingly, $\mathcal{S}$-VAE is not able to learn on this dataset, the vMF posterior is almost flat for any input image. This hints at potential optimisation issues with the complicated reparameterisation. This advantage also extends to priors over the hyperbolic space, which are used to capture tree-like hierarchical structures (Nagano et al., 2019; Mathieu et al., 2019).

4.3. Hierarchical Models

Rich hierarchical structures in the data can be captured with multiple layers of latents. Provided that samples can be drawn from the hierarchical model and the joint log-likelihood evaluated, ALWS extends straightforwardly to hierarchies, even with mixed discrete and continuous latents. The pinwheel distribution (Johnson et al., 2016; Lin et al., 2018) has five clusters of distorted Gaussian distributions (Figure 4), and can be described by the following model:

$$p(\boldsymbol{z}_1) = \mathrm{Cat}(\boldsymbol{z}_1; \boldsymbol{m}), \;\; p(\boldsymbol{z}_2|\boldsymbol{z}_1 = k) = \mathcal{N}(\boldsymbol{z}_2; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$
$$p(\boldsymbol{x}|\boldsymbol{z}_2) = \mathcal{N}(\boldsymbol{x}; \mathrm{NN}_{\boldsymbol{w}}(\boldsymbol{z}_2), \boldsymbol{\Sigma}_x),$$

where Cat is the categorical distribution. The parameters are the logits $\boldsymbol{m}$ in 10 dimensions, the means and covariance matrices of the component distributions $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{10}$, the weights $\boldsymbol{w}$ in NN, and the diagonal covariance $\boldsymbol{\Sigma}_x$. The logits $\boldsymbol{m}$ are penalised according to a Dirichlet prior, and $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{10}$ by a normal-Wishart prior. After training with ALWS, the categorical distribution correctly identifies the five components, and the generated samples match the training data. We compare these samples with those reconstructed from a Bayesian version of the model trained by structured inference network (SIN) (Lin et al., 2018)[4]. A three-way maximum mean discrepancy (MMD) test (Bounliphone et al., 2016) finds that samples from the two models are equally close to the training data ($p = 0.514$, $N = 1,000$ samples). Details are in Appendix C.3.

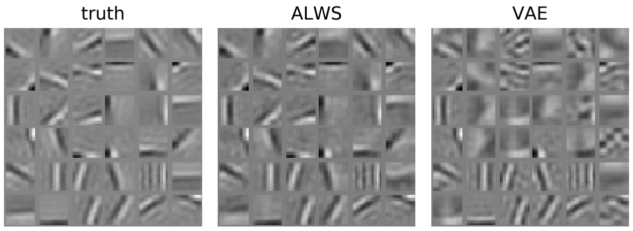

Figure 5. Feature identification. Left, true basis used to generate images. Middle, basis recovered by ALWS. Right, basis recovered by VAE. The filters are arranged according to correlations with the true basis.

4.4. Feature Identification

Independent Components. Learning informative features from complex data can benefit downstream tasks. We use ALWS to identify features from data generated by

$$p(z_i) = \mathrm{Lap}(z_i; 0, 1), \quad p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{W}\boldsymbol{z}, \sigma^2 I),$$

where Lap is the Laplace distribution, $\sigma = 0.1$ and basis $\boldsymbol{W}$ contains independent components of natural images (Hateren & Schaaf, 1998) found by the FastICA algorithm (Hyvärinen & Oja, 2000). Since this model is identifiable, we perform model recovery from a random initialisation of $\boldsymbol{W}$ using ALWS and compare with a VAE. ALWS clearly finds better features, as shown in Figure 5. On generated samples, a three-way MMD test favours ALWS over the Laplace-VAE ($p < 10^{-5}$) based on $10,000$ samples. Details are in Appendix C.4.

[4] `github.com/emtiyaz/vmp-for-svae`

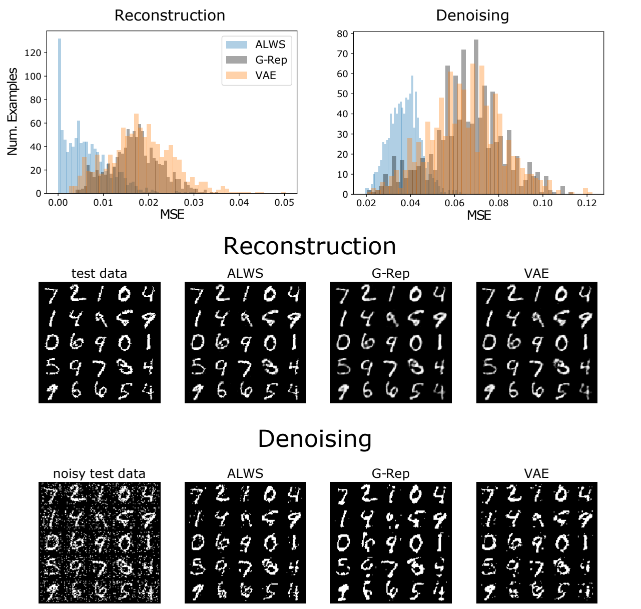

Figure 6. Beta-Gamma Matrix Factorisation. Top, mean squared error across 1,000 test inputs compared to G-Rep and VAE. Bottom, examples of real data, reconstructed and denoised samples.

Matrix Factorisation. A more accurate data model may improve performance on a downstream task that relies on inference of associated latent variables. Following (Ruiz et al., 2016), we test post-learning inference on a probabilistic non-negative matrix factorisation model:

$$p(z_i) = \mathcal{U}(z_i; 0, 1), \quad p(x_i|\boldsymbol{z}) = \text{Bernoulli}\,(x_i; \bar{x}_i)$$
$$\bar{x}_i = \text{sigmoid}\,(\boldsymbol{w}_i \cdot \text{logit}(\boldsymbol{z}) + b_i)\,.$$

For each element of each $\boldsymbol{w}_i$, we place a penalty consistent with a Gamma$(w; 0.9, 0.3)$ prior on each entry and learn $\boldsymbol{W}$ and $\boldsymbol{b}$. We include $\boldsymbol{b}$ to the model trained by ALWS as it prevents samples with opposite colour polarity to be generated, which creates a more severe covariate shift that harms the gradient model. We evaluate the models on reconstructing and denoising handwritten digits from the binarised MNIST dataset. To recover the original image given a clean or noisy $\boldsymbol{x}^*$, we generate $\boldsymbol{x}$ given the posterior mode found by maximising $\log p(\boldsymbol{z}, \boldsymbol{x}^*)$ over $\boldsymbol{z}$. We compare with a Bayesian version of the model trained by generalised reparameterisation Ruiz et al. (2016) and a VAE-like model in which the decoder has the generative structure as above and the posterior is a reparametrised Beta distribution. The results for both tasks are depicted in Figure 6. The leftmost panels show the histograms of MSE on 1 000 test images, and the other panels show examples of 25 test images and reconstructions by each method. ALWS achieved significantly lower error ($p < 10^{-10}$ for both a two-tailed t-test and a Wilcoxon signed-rank test).

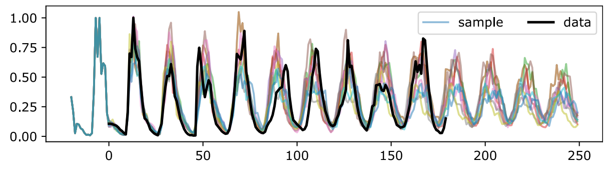

Figure 7. Modelling blowfly population time series. Black, training data. Coloured, samples for an extended time period drawn from trained model.

4.5. Neural Processes

The neural process (NP) (Garnelo et al., 2018) is a model that learns to infer over functions. Conceptually, the computational goal of NPs is similar to predictive inference in Gaussian Processes, but without defining an explicit prior over functions. We review NPs in more detail and illustrate how they can be trained by ML using ALWS in Appendix C.5. We compared ALWS with the original variational learning method on a toy problem. NP trained by ALWS produces better prediction and uncertainty estimates on test inputs. See Figure 10 in Appendix C.5.

4.6. Dynamical Models

In fields such as biology and environmental science, the behaviour of complex systems is often described by simulation-based dynamical models. Estimating parameters for these models from data is crucial for prediction and policy-making. (Lintusaari et al., 2016; Sunnåker et al., 2013; Kypraios et al., 2017)

A dynamical model can be expressed, in discrete time, as

$$\boldsymbol{z}_t = \boldsymbol{l}_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t-1}, \boldsymbol{u}_t, \boldsymbol{\epsilon}_t), \quad \boldsymbol{x}_t = \boldsymbol{o}_{\boldsymbol{\theta}}(\boldsymbol{z}_t) + \boldsymbol{e}_t$$

where $\boldsymbol{l}_{\boldsymbol{\theta}}$ describes a latent process that can depend on a control input $\boldsymbol{u}_t$, a noise source $\boldsymbol{\epsilon}_t$ and the history of latents $\boldsymbol{z}_{1:t-1}$ and measurements $\boldsymbol{x}_{1:t-1}$. The function $\boldsymbol{o}_{\boldsymbol{\theta}}$ maps the latent $\boldsymbol{z}_t$ to measurement with noise $\boldsymbol{e}_t$. For ALWS, we need that $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \boldsymbol{\epsilon}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t-1}, \boldsymbol{u}_t)$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{e}_t|\boldsymbol{z}_t)$ are tractable so that $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{z}_{1:T}, \boldsymbol{x}_{1:T})$ can be evaluated, where T is the length of the data. However, learning using approximate inference may be challenging due to complex dependencies between latent variables and across time.

Here, we fit the parameters of two dynamical models: the Hodgkin-Huxley (HH) model (Pospischil et al., 2008) on the membrane potential of a simulated neuron, and an ecological model (ECO) on blowfly data (Wood, 2010). The HH equations describe the membrane potential and three ion-channel state variables of a neuron that follow complicated nonlinear transitions. Details of the experiment are in Appendix C.6. Results in Figure 12 show that the trained model can not only reproduce the training data well but also predict the response given new inputs $\boldsymbol{u}_t$. ECO describes nonlinear and non-Gaussian dynamics and has discrete and continuous latent variables. Fitting ECO on blowfly data was used to validate approximate Bayesian computation (ABC) methods (Park et al., 2016). The model trained with ALWS can simulated sequences very close to data Figure 7, and are visibly closer than sequences from the model trained with ABC (Park et al., 2016, Figure 2b).

4.7. Sample Quality

Finally, we train deep models of images and test sample quality. We chose six benchmark datasets: the binarised and original MNIST (LeCun et al., 1998) (B-MNIST and MNIST, respectively), fashion MNIST (Fashion) (Xiao et al., 2017), natural images (Natural) (Hateren & Schaaf, 1998), CIFAR-10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015). The original un-binarised MNIST is known to be difficult for most VAE-based methods (Loaiza-Ganem & Cunningham, 2019). Natural images consist of grey-scale images from natural scenes. All images have size 32×32 with colour channels. For ALWS, we test two variants. In ALWS-F, gradient model parameters γ are fixed. In ALWS-A, γ is adapted as described in Section 3.4 except for λ which is fixed at 0.1. Fixing λ improved quality for the higher-dimensional CIFAR-10 and CelebA, but lowered quality for Natural and did not affect much on the other datasets.

We compare these methods with four other approaches: the vanilla VAE (Kingma & Welling, 2014), VAE with a Sylvester (orthogonal) flow as an inference network (van den Berg et al., 2018) (Syl-VAE)[5], semi-implicit variational inference (Yin & Zhou, 2018) (SIVI)[6], and reweighted wake-sleep (Bornschein & Bengio, 2015). Each algorithm has the same generative network architecture as in DCGAN[7] with the last convolutional layer removed. We also run WGAN-GP (Gulrajani et al., 2017)[8] for reference, although it is not trained by ML methods. Each algorithm is run for 50 epochs 10 times with different initialisations, except for SIVI where we trained for 1000 epochs with a lower learning rate for stability. To test the generative quality, we compute both the Fréchet Inception Distance (FID) (Heusel et al., 2017) and Kernel Inception Distance (KID) (Binkowski et al., 2018) on 10,000 generated images. The results are shown in Figure 8. According to FID, ALWS-A is the best ML method for binarised MNIST, Fashion, and CIFAR-10. Notably, both ALWS-A and ALWS-F have much smaller FID and KID on MNIST and Fashion than other ML methods. WGAN-GP did not produce a good score on CIFAR-10 within 50 epochs but becomes the best model for all datasets with further training. Samples are shown from Figure 15 to Figure 20 in Appendix C.7 with additional experiments to show the effectiveness of ALWS.

[5] github.com/riannevdberg/sylvester-flows
[6] github.com/mingzhang-yin/SIVI
[7] pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
[8] github.com/caogang/wgan-gp

5. Related Work

5.1. Amortised Variational Inference

Using $\mathcal{F}(q, \boldsymbol{\theta})$ as the objective for learning $\boldsymbol{\theta}$, the gradient for $\boldsymbol{\theta}$ is given by an intractable posterior expectation. The large majority of learning algorithms based on amortised variational inference use Monte Carlo estimators for the gradient. The Variational auto-encoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) parametrises $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ by simple distributions using reparameterised samples to obtain gradients for $\boldsymbol{\psi}$. Approximate posteriors may also be incorporated into tighter bounds on $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by reweighting (Burda et al., 2016; Bornschein & Bengio, 2015; Le et al., 2019), although with some loss of gradient signal (Rainforth et al., 2018). More expressive forms of $q_{\boldsymbol{\phi}}$ can be formed by invertible transformations (normalising flows) (Rezende & Mohamed, 2015; Kingma et al., 2016; van den Berg et al., 2018)) that allow $\mathbb{H}[q_{\boldsymbol{\phi}}]$ to be computed easily, or by non-invertible mappings (implicit variational inference), which requires estimating $\mathbb{H}[q_{\boldsymbol{\phi}}]$ or its gradient w.r.t. $\boldsymbol{\phi}$ (Shi et al., 2018; Li & Turner, 2018; Yin & Zhou, 2018; Huszár, 2017). Reparametrising posterior samples may require non-trivial methods (Jang et al., 2017; Vahdat et al., 2018; Rolfe, 2017; Ruiz et al., 2016; Figurnov et al., 2018). On the other hand, amortised learning focuses exclusively on estimating the gradient for ML learning, making no assumptions on the type of latent variables.

Our approach is related to at least two other algorithms inspired by the original Helmholtz machine (HM) (Dayan et al., 1995; Hinton et al., 1995). The distributed distributional code HM (DDC-HM) (Vértes & Sahani, 2018) represents posteriors by expectations of pre-defined and finite nonlinear features, which are used to approximate $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ by linearity of expectation. ALWS differs from DDC-HM in two ways. First, our gradient model integrates the inferential model and the linear readout for $\Delta_{\boldsymbol{\theta}_t}(\boldsymbol{x})$ in DDC-HM using adaptive and more flexible KRR. Second, using (9) avoids explicit computation of $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}$ and makes ALWS easily applicable to more complex generative models. Reweighted wake-sleep (RWS) (Bornschein & Bengio, 2015) addressed covariance shift by training an inferential model to increase the likelihood of not only sleep $\boldsymbol{z}$ given sleep $\boldsymbol{x}$ as in the HM, but also weighted posterior samples given data $\boldsymbol{x}^*$. ALWS does not make assumptions about the posterior distributions, and we found that simple strategies mitigated covariate shift in practice, but this is a point that deserves further investigation.

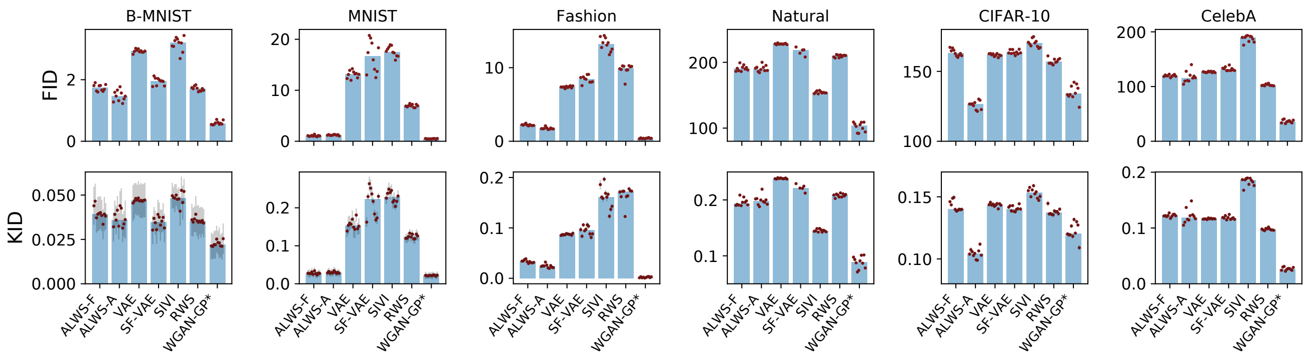

Figure 8. FID and KID scores (lower is better) for different datasets and methods. Red dot is the score for a single run. Bars are medians of the dots for each method. Short bars on KID dots shows standard error of the estimate. All models are trained for 50 epochs.

5.2. Training Implicit Generative Models

Implicit generative models, including generative adversarial networks (GANs) (Goodfellow et al., 2014) and simulation-based models considered by approximate Bayesian computation (ABC) (Tavaré et al., 1997; Marin et al., 2012), do not have an explicitly defined likelihood function, but can be trained using simulated data. Amortised learning requires an explicit joint likelihood function $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$, but can also train simulation-based generative models (Section 4.6). In GANs, the generator is improved by a discriminator that is concurrently trained to tell apart real and generated samples. The approach is able to synthesise high-quality samples in high dimensions. However, the competitive setting can be problematic for convergence, and the discriminator needs to be carefully regularised to be less effective at its own task but more informative to the generator. (Arjovsky et al., 2017; Gulrajani et al., 2017; Arbel et al., 2018; Mescheder et al., 2018). In amortised learning, a better gradient model always helps when training the generative model. Importantly, amortised learning can directly train real-world simulators for which samples of $\boldsymbol{x}$ are not differentiable w.r.t. $\boldsymbol{\theta}$, such as the Galton board, where GANs are not directly applicable.

Rather than performing maximum likelihood estimation, ABC estimates a posterior of $\boldsymbol{\theta}$ using simulated data and a chosen prior on $\boldsymbol{\theta}$. Amortised learning can be seen as maximum likelihood learning based on simulations, since the gradient model is trained using data from the generative model. In particular, ALWS is similar to Kernel-ABC (Nakagome et al., 2013) in which the posterior is found by weighting prior samples using KRR on pre-defined summary statistics. The kernel recursive ABC (Kajihara et al., 2018) iteratively updates the prior over $\boldsymbol{\theta}$ by herding from a kernel embedding (Song et al., 2009) of the posterior, converging to a maximum likelihood solution. ALWS does not maintain a distribution of $\boldsymbol{\theta}$, but iteratively updates them by gradient methods so that the model distribution approaches the data distribution. Also, ALWS performs well even when the number of parameters is large for which traditional ABC methods are likely to be expensive.

6. Discussion

Direct estimation of the expected log-likelihood and its gradient in a latent variable model circumvents the challenges and issues posed by explicit approximation of posteriors. The KRR gradient model is consistent, easy to implement, and avoids the need for explicit computation of derivatives. However, we observe the following issues with the current instance of amortised learning. First, its computational complexity limits the number of sleep samples that can be used to train the gradient model and thus the quality of the approximation. Techniques such as random feature- and Nystrom-approximations could make KRR more efficient. Second, the KRR prediction is a linear combination of the set $\{\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}_n, \boldsymbol{x}_n)\}_{n=1}^N$, but the true gradient function, which can be much higher-dimensional than N, may lie outside this span—an issue that might be compounded by covariate shift. Further, hyper-parameter learning using the meta-learning method described in Section 3.4 improves the estimation of $J_{\boldsymbol{\theta}}$ rather than $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$, which might explain why adapting λ on some tasks worsens the results. Therefore, alternative amortised learning models may be worth future exploration. Nonetheless, we have found here that ALWS based on KRR provides accurate parameter estimates in many settings where approximate inference-based approaches appear to struggle.

ALWS can be extended to training generative models of other types of data, such as graphs, as long as an appropriate kernel is used. Another useful extension is to train conditional generative models, which we explored briefly in the neural processes experiment. In this case, the gradient model needs to depend on any conditioning variables (or sets). Finally, while we used LSR to approximate the gradient of the model w.r.t θ, other useful quantities could also be estimated in a similar fashion Brehmer et al. (2020).

Acknowledgements

We thank Arthur Gretton, Sebastian Nowozin, Jiaxin Shi and Eszter Vértes for helpful discussions; we thank Ferenc Huszár for discussion and comments on an earlier draft.

References

Arbel, M., Sutherland, D. J., Binkowski, M., and Gretton, A. On gradient regularizers for MMD GANs. In *NeurIPS*, pp. 6701–6711, 2018.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *ICML*, 2017.

Binkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *ICLR*, 2018.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Bornschein, J. and Bengio, Y. Reweighted wake-sleep. In *ICLR*, 2015.

Boucheron, S., Lugosi, G., and Massart, P. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, February 2013. ISBN 978-0-19-953525-5. doi: 10.1093/acprof:oso/9780199535255.001.0001.

Bounliphone, W., Belilovsky, E., Blaschko, M. B., Antonoglou, I., and Gretton, A. A test of relative similarity for model selection in generative models. In *ICLR*, 2016.

Brehmer, J., Louppe, G., Pavez, J., and Cranmer, K. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10):5242–5249, 2020.

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In *ICLR*, 2016.

Caponnetto, A. and De Vito, E. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 2007.

Carmeli, C., De Vito, E., and Toigo, A. Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem. *Analysis and Applications*, 2006.

Carmeli, C., De Vito, E., Toigo, A., and Umanitá, V. Vector valued reproducing kernel Hilbert spaces and universality. *Analysis and Applications*, 2010.

Chatterjee, S., Diaconis, P., et al. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

Davidson, T. R., Falorsi, L., Cao, N. D., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. In *UAI*, 2018.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The Helmholtz machine. *Neural computation*, 1995.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1977.

Dieng, A. B. and Paisley, J. Reweighted expectation maximization. *arXiv preprint arXiv:1906.05850*, 2019.

Figurnov, M., Mohamed, S., and Mnih, A. Implicit reparameterization gradients. In *NeurIPS*, 2018.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NeurIPS*, pp. 2672–2680, 2014.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 2009.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. In *NeurIPS*, 2017.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

Hateren, J. H. v. and Schaaf, A. v. d. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 1998.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 1995.

Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics*, 2008.

Huggins, J. H., Kasprzak, M., Campbell, T., and Broderick, T. Practical posterior error bounds from variational objectives. *CoRR*, abs/1910.04102, 2019.

Huszár, F. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Hyvärinen, A. and Oja, E. Independent component analysis: algorithms and applications. *Neural Networks*, 2000.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *NeurIPS*, pp. 2946–2954, 2016.

Kajihara, T., Kanagawa, M., Yamazaki, K., and Fukumizu, K. Kernel recursive abc: Point estimation with intractable likelihood. In *International Conference on Machine Learning*, pp. 2400–2409, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *NIPS*, pp. 4743–4751, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, 2009.

Kypraios, T., Neal, P., and Prangle, D. A tutorial introduction to Bayesian inference for stochastic epidemic models using Approximate Bayesian Computation. *Mathematical biosciences*, 2017.

Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. Revisiting reweighted wake-sleep for models with stochastic control flow. In *UAI*, 2019.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, Y. and Turner, R. E. Gradient estimators for implicit models. In *ICLR*, 2018.

Lin, W., Hubacher, N., and Khan, M. E. Variational message passing with structured inference networks. In *ICLR*, 2018.

Lintusaari, J., Gutmann, M. U., Dutta, R., Kaski, S., and Corander, J. Fundamentals and Recent Developments in Approximate Bayesian Computation. *Systematic Biology*, 2016.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *ICCV*, 2015.

Loaiza-Ganem, G. and Cunningham, J. P. The continuous Bernoulli: fixing a pervasive error in variational autoencoders. In *NeurIPS*, 2019.

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. Approximate Bayesian computational methods. *Statistics and Computing*, 2012.

Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., and Teh, Y. W. Continuous hierarchical representations with Poincaré variational auto-encoders. In *NeurIPS*, 2019.

Mescheder, L. M., Geiger, A., and Nowozin, S. Which training methods for GANs do actually converge? In *ICML*, 2018.

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *ICML*, 2019.

Nakagome, S., Fukumizu, K., and Mano, S. Kernel approximate bayesian computation in population genetic inferences. *Statistical applications in genetics and molecular biology*, 2013.

Newey, K. and McFadden, D. Large sample estimation and hypothesis. *Handbook of Econometrics, IV, Edited by RF Engle and DL McFadden*, 1994.

Park, M., Jitkrittum, W., and Sejdinovic, D. K2-ABC: Approximate Bayesian Computation with kernel embeddings. In *AISTATS*, 2016.

Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., and Destexhe, A. Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological cybernetics*, 2008.

Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. W. Tighter variational bounds are not necessarily better. In *ICML*, 2018.

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

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pp. 1278–1286, 2014.

Rolfe, J. T. Discrete variational autoencoders. In *ICLR*, 2017.

Ruiz, F. J. R., Titsias, M. K., and Blei, D. M. The generalized reparameterization gradient. In *NeurIPS*, 2016.

Shi, J., Sun, S., and Zhu, J. Kernel implicit variational inference. In *ICLR*, 2018.

Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 2000.

Song, L., Huang, J., Smola, A., and Fukumizu, K. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *ICML*, 2009.

Sunnåker, M., Busetto, A. G., Numminen, E., Corander, J., Foll, M., and Dessimoz, C. Approximate Bayesian Computation. *PLoS CB*, 2013.

Szabó, Z., Sriperumbudur, B. K., Póczos, B., and Gretton, A. Learning theory for distribution regression. *Journal of Machine Learning Research*, 2016.

Tavaré, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. Inferring coalescence times from DNA sequence data. *Genetics*, 1997.

Turner, R. and Sahani, M. Two problems with variational expectation maximisation for time-series models. *Bayesian Time Series Models*, 2011.

Vahdat, A., Macready, W. G., Bian, Z., Khoshaman, A., and Andriyash, E. DVAE++: Discrete variational autoencoders with overlapping transformations. In *ICML*, 2018.

van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 393–402, 2018.

Vértes, E. and Sahani, M. Flexible and accurate inference and learning for deep generative models. In *NeurIPS*, pp. 4166–4175, 2018.

Wenliang, L., Sutherland, D. J., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities. In *ICML*, 2019.

Wenliang, L. K. and Sahani, M. A neurally plausible model for online recognition and postdiction in a dynamical environment. In *NeurIPS*, 2019.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *AISTATS*, 2016.

Wood, S. N. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 2010.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xu, J. and Durrett, G. Spherical latent spaces for stable variational autoencoders. *EMNLP*, 2018.

Yin, M. and Zhou, M. Semi-implicit variational inference. In *ICML*, 2018.

Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. Advances in variational inference. *Pattern analysis and machine intelligence*, 2018.