πVAE: Encoding stochastic process priors with variational autoencoders

Swapnil Mishra [*1] **Seth Flaxman** [*2] **Samir Bhatt** [*1]

Abstract

Stochastic processes provide a mathematically elegant way model complex data. In theory, they provide flexible priors over function classes that can encode a wide range of interesting assumptions. In practice, however, efficient inference by optimisation or marginalisation is difficult, a problem further exacerbated with big data and high dimensional input spaces. We propose a novel variational autoencoder (VAE) called the prior encoding variational autoencoder (πVAE). The πVAE is finitely exchangeable and Kolmogorov consistent, and thus is a continuous stochastic process. We use πVAE to learn low dimensional embeddings of function classes. We show that our framework can accurately learn expressive function classes such as Gaussian processes, but also properties of functions to enable statistical inference (such as the integral of a log Gaussian process). For popular tasks, such as spatial interpolation, πVAE achieves state-of-the-art performance both in terms of accuracy and computational efficiency. Perhaps most usefully, we demonstrate that the low dimensional independently distributed latent space representation learnt provides an elegant and scalable means of performing Bayesian inference for stochastic processes within probabilistic programming languages such as Stan.

1. Introduction

A central task in machine learning is to specify a function or set of functions that best generalises to new data. Stochastic processes (Ross, 1996) provide a mathematically elegant way to define a class of functions, where each element from a stochastic process is a (usually infinite) collection of random variables. Popular examples of stochastic processes in machine learning are Gaussian processes (Rasmussen & Williams, 2006), Dirichlet processes (Antoniak, 1974), log Gaussian-Cox processes (Mller et al., 1998), Hawkes processes (Hawkes, 1971), Mondrian processes (Roy & Teh, 2009) and Gauss-Markov processes (Lindgren et al., 2011). Many of these processes are intimately connected with popular techniques in deep learning, for example, the infinite width limit of a single layer neural network and the evolution of a deep neural network by gradient descent are Gaussian processes (Jacot et al., 2018; Neal, 1996). However, while stochastic processes have many favourable properties, they are often cumbersome to work with in practice. For example, inference and prediction using a Gaussian process requires matrix inversions that scale cubicly with data size, log Gaussian Cox processes require the evaluation of of an intractable integral and Markov processes are often highly correlated. Bayesian inference can be even more challenging due complex high dimensional posterior topologies. Gold standard evaluation of posterior expectations is done by Markov Chain Monte Carlo (MCMC) sampling, but high autocorrelation, narrow typical sets (Betancourt et al., 2017) and poor scalability have prevented use in big data/model settings. A plethora of approximation algorithms exist (Minka, 2001; Ritter et al., 2018; Lakshminarayanan et al., 2017; Welling & Teh, 2011; Blundell et al., 2015), but few have asymptotic guarantees and, more troubling, even fewer actually yield accurate posterior estimates (Yao et al., 2018; Huggins et al., 2019; Hoffman et al., 2013; Yao et al., 2019). In this paper, rather than relying on approximate Bayesian inference to solve complex models, we extend variational autoencoders (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) to develop portable models that can work with state-of-the-art Bayesian MCMC software such as Stan (Carpenter et al., 2017). Inference on the resulting models is tractable and yields accurate posterior expectations and uncertainty.

An autoencoder (Hinton & Salakhutdinov, 2006) is a model comprised of two component networks η: The encoder $\eta_e : \mathcal{X} \to \mathcal{Z}$ and a decoder $\eta_d : \mathcal{Z} \to \mathcal{X}$ where the latent space $\mathcal{Z}$ is typically of much lower dimension than $\mathcal{X}$. The autoencoder is then learnt through the minimisation of a loss function $\mathcal{L}$. A VAE extends the autoencoder into a generative model (Kingma & Welling, 2014). In a VAE, the

[*]Equal contribution [1]School of Public Health Imperial College, London [2]Department of Mathematics Imperial College London. Correspondence to: Swapnil Mishra <s.mishra@imperial.ac.uk>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML).

latent variables $\mathcal{Z}$ are independent normally distributed and a variational approximation of the posterior is estimated. In a variety of applications, VAEs fit well to the trained data and enable the generation of new data by sampling from the latent space; a mechanism that makes it a popular tool for probabilistic modelling (Kingma & Welling, 2019). In this paper we propose a novel use of VAEs: we learn low dimensional representations of samples from a given function class (e.g. sample paths from a Gaussian process prior). We then use the resulting low dimensional representation and decoder to perform Bayesian inference.

One key benefit of this approach is that we decouple the prior from inference to encode arbitrarily complex prior function classes, without needing to calculate any data likelihoods. A second key benefit is that when inference is performed, our sampler operates in a low dimensional, uncorrelated latent space which greatly aids efficiency and computation. One limitation of this approach is that we are restricted to encoding finite-dimensional priors, because VAEs are not stochastic processes. To overcome this limitation we introduce a new VAE called the prior encoding VAE (πVAE) that satisfies exchangeability and Kolmogorov consistency and thus is thus a valid stochastic process by the Kolmogorov extension theorem.

The most closely related approach to ours is the neural process (Gamelo et al., 2018) which, similar to Gaussian processes, tries to learn a distribution over functions.The neural process is a stochastic process that fits to data while simultaneously learning the function class. In functional form, it is similar to a VAE except, in neural processes, the KL divergence penalty term is the divergence from training and testing latent variables (as opposed to a standard normal in a VAE) and an aggregation step is included to allow for exchangeability and consistency. In practice, neural processes often learn functions that substantially underfit and therefore tend to predict both point estimates and uncertainty inaccurately (Kim et al., 2019). In contrast to the neural process, we employ a two step approach of disassociating prior encoding from the inference step.

Once a πVAE is trained and defined the complexity of the decoder scales linearly in the size of the largest hidden layer. Additionally, because the latent variables are penalised via the KL term from deviating from standard Gaussians, the latent space is approximately uncorrelated, leading to high effective sample sizes in MCMC sampling. The main contributions of this paper are:

- We apply the current generative framework of VAEs to perform full Bayesian inference by first encoding priors in training and then, given new data, running inference on the latent representation while keeping the trained decoder fixed.
- We propose a new generative model, πVAE, that further generalizes VAEs to be able to learn both priors over a functions and properties of functions. We show that πVAE satisfies the Kolmogorov extension theorem and is therefore a stochastic process.
- Finally, we show the performance of πVAE on a range of simulated and real data, and show that πVAE achieves state-of-the-art performance in a spatial interpolation task.

The rest of this paper is structured as follows. Section 2 details the proposed framework and the generative model along with toy fitting examples. The experiments on large real world datasets are outlined in Section 3. We discuss our findings and conclude in Section 4.

2. Methods

In this section we first introduce the basics of VAE and propose a way to extend them for learning priors over random vectors in Section 2.1. Next in Section 2.2 we introduce a novel VAE framework, πVAE, which is a stochastic process and encodes priors over a function class.

2.1. Variational Autoencoders (VAEs)

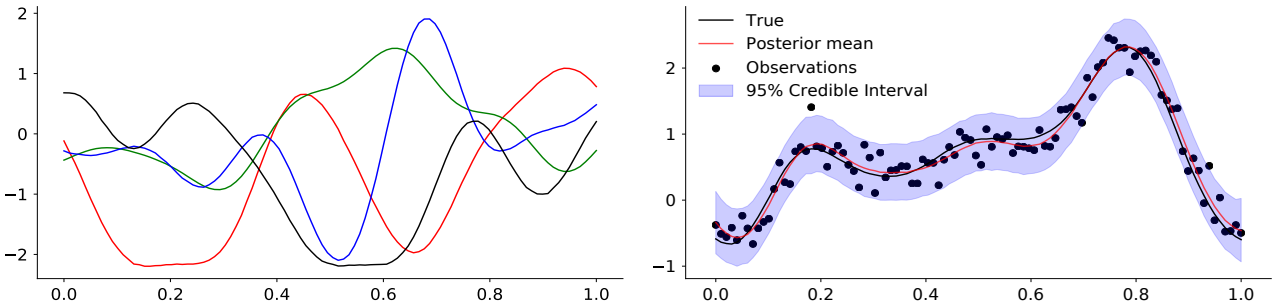
A standard VAE has three components, an encoder network ($e(\cdot)$) with parameters η_e, a latent space $\mathcal{Z}$, and a decoder network ($d(\cdot)$) with parameters η_d. In many settings, given an input, $x \in \mathbb{R}$ (e.g. a flattened image or discrete time series), $e(\eta_e, x)$ and $d(\eta_d, \mathcal{Z})$ are simply multilayer perceptrons, but can be more complicated and include convolution or recurrent layers. The output of the encoder network $\mathcal{Z}$ is a latent representation of x in a lower-dimensional Gaussian probabilistic space. The decoder network takes this latent space and tries to reconstruct the input by producing $\hat{x}$. Putting both the encoder and decoder together results in the following model

$$\begin{aligned} [z_\mu, z_{sd}]^\top &= e(\eta_e, x) \\ \mathcal{Z} &\sim \mathcal{N}(z_\mu, z_{sd}^2 \mathbb{I}) \\ \hat{x} &= d(\eta_d, \mathcal{Z}) \end{aligned} \tag{1}$$

End to end, equation 1 can be compactly expressed as $\hat{x} = d(\eta_d, e(\eta_e, x))$. To train a VAE, a variational approximation is used to estimate the posterior distribution $p(\mathcal{Z}|x, \eta_d, \eta_e) \propto p(x|\mathcal{Z}, \eta_d, \eta_e)p(\mathcal{Z})$. The variational approximation greatly simplifies inference by turning a marginalisation problem into an optimisation problem. The optimal parameters for the encoder and decoder are found by maximising the evidence lower bound:

$$\arg\max_{\eta_e, \eta_d} p(x|\mathcal{Z}, \eta_d, \eta_e) - \mathrm{KL}\left(\mathcal{Z} \| \mathcal{N}(0, \mathbb{I})\right) \tag{2}$$

The first term in equation 2 is the likelihood quantifying how well $\hat{x}$ matches x. The second term is a Kullback-



(a) (b)

Figure 1: Learning functions with VAE: (a) Prior samples from a VAE trained on Gaussian process samples (b) we fit our VAE model to data drawn from a GP (blue) plus noise (black points). The posterior mean of our model is in red with the 95% epistemic credible intervals shown in purple.

Leibler divergence penalty to ensure that $\mathcal{Z}$ is as similar as possible to the prior distribution, a standard normal.

Once fully trained on a set of samples, we fix η_d, and use the decoder as a generative model. To simplify subsequent notation we refer to a fully trained decoder as d and when evaluated as $d(\cdot)$. Generating a new sample vector v is simple: First draw $z \sim \mathcal{N}(z_\mu, z_{sd}^2\mathbb{I})$ and then perform a deterministic transformation via the decoder $v = d(z)$.

VAEs have been typically used in literature to create or learn a generative model of observed data (Kingma & Ba, 2014). However, here we introduce a novel application of VAEs to use them for Bayesian inference on new data after learning a prior over random vectors. In this role we first train a VAE on random vectors from an accessible space such as a reproducing kernel Hilbert space (samples from a Gaussian process). Training a VAE embeds these vectors in a lower dimensional probabilistic space. Once fully trained, sampling from this lower dimensional space and transforming via the decoder would then be able to generate new random vectors. This decoder end of the VAE can therefore be used to perform inference on a new "data" vector, y, where the parameters of the decoder are fixed, and the latent space is trainable. In this inferential scheme the unnormalised posterior distribution of $\mathcal{Z}$ is:

$$p(\mathcal{Z}|y, d) \propto p(y|d, \mathcal{Z})p(\mathcal{Z}) \tag{3}$$

In the standard VAE approach introduced above, we have:

$$p(z_\mu, z_{sd}|y, d) \propto p(y|d, z_\mu, z_{sd})\mathcal{N}(z_\mu, z_{sd}^2\mathbb{I})$$

Note that the observation model $p(y|d, \mathcal{Z})$ in equation (3) could be generalised to include an additional parameter to account for observation error. It is useful to contrast the inference task from equation (3) to a Bayesian neural network (BNN) (Neal, 1996) or Gaussian process in primal form (Rahimi & Recht, 2008). In a BNN, with weights and biases ω, with prior hyperparameters λ, the unnormalised posterior would be

$$p(\omega, \lambda|s, y) \propto p(y|s, \omega, \lambda)p(\omega|\lambda)p(\lambda) \tag{4}$$

The key difference between equations (4) and (3) is the term $p(\omega|\lambda)$. ω is typically huge, sometimes in the millions, and is conditional on λ, where as in equation (3) the latent dimension of $\mathcal{Z}$ is typically small (< 50), uncorrelated and unconditioned. Given the sizes of ω full batch MCMC training is difficult and approximation algorithms tend to poorly capture the complex posterior (Yao et al., 2019; 2018). Additionally, ω tends to be highly correlated, making efficient MCMC nearly impossible. Finally, as the dimension and depth increases the posterior distribution suffers from complex multi-modality, and concentration to a narrow typical set (Betancourt et al., 2017). By contrast, off-the-shelf MCMC methods like Stan (Carpenter et al., 2017) are very effective for equation (3), since they simply need to sample from a multidimensional Gaussian distribution, while the complexity is accounted for by the deterministic decoder.

An example of using VAEs to perform inference is shown in Figure 1 where we train a VAE ($\mathcal{Z} = 10$) on samples drawn from a zero mean Gaussian process with RBF kernel ($K(\delta) = e^{-\delta^2/8^2}$). We drew 10^5 samples of size $n = 200$ over an equally spaced grid and trained a VAE for 100 epochs with batch size 500. The trained weights from the resulting decoder were then extracted and embedded within the probabilistic programming language Stan (Carpenter et al., 2017). We then sampled a new function from our GP with noise $y = GP(0, K) + \mathcal{N}(0, 0.2)$ and performed inference on only the low dimensional latent space $\mathcal{Z}$ while keeping η_d fixed. As Figure 1 shows, we closely recover the true function, but also correctly estimate the posterior data noise parameter. Most crucially, our MCMC samples showed virtually no autocorrelation, and all diagnostic checks such as $\hat{R}$ and simulation-Based Calibration

(Gabry et al., 2019) were excellent, detailed results in Appendix (Anonymous, 2020). Solving the equivalent problem using a Gaussian process prior would not only be considerably more expensive ($\mathcal{O}(n^3)$) but correlation in the parameter space would complicate MCMC sampling and necessitate long chains with large amounts of thinning.

While this simple example might seem useful, inference and prediction using a VAEs will not be possible if new input locations are required, or if the input locations are permuted. These limitations occur because a VAE is not a valid stochastic process in the sense of the Kolmogorov extension theorem. The Kolmogorov extension theorem guarantees that a suitably consistent and exchangeable collection of finite-dimensional distributions will define a stochastic process. Specifically, exchangeability means, for a permutation function $\pi(\cdot)$, $p(y_{1:n}|x_{1:n}) = p(y_{\pi(1:n)}|x_{\pi(1:n)})$ and Kolmogorov consistency means that that probability defined on an observed set of points is the same as the probability defined on an extended sequence of points when marginalised i.e. $p(y_{1:n}|x_{1:n}) = \int p(y_{1:n+m}|x_{1:n+m})\, dp(y_{n+1:n+m})$. A VAE does not satisfy either of these conditions, as can be seen by the the lack of conditioning on input locations in equation (3). In short, in a VAE the choice of input locations needs to be fixed ahead of time so no extrapolation is possible.

To overcome these problems we introduce a new extension to VAE called πVAE that is capable of learning a stochastic process class of functions.

2.2. The stochastic process prior encoding Variational Autoencoder - πVAE

To create a VAE with the ability perform inference on a wide range of problems we have to ensure exchangeability and Kolmogorov consistency. Previous attempts to do this have relied on introducing an aggregation (typically an average) to create an order invariate global distribution (Gamelo et al., 2018). However, as noted by (Kim et al., 2019), this can lead to underfitting. For πVAE we learn a feature mapping Φ over the input space. This ensures that πVAE is a valid stochastic process (see theorem 1 and theorem 2).

Consider a probability space $(\Omega, \mathcal{F}, P)$ where Ω is a sample space, $\mathcal{F}$ is a σ-algebra and P is a probability measure. We can define a set of locations S, e.g. for time series this is a set of time labels, and for spatial data $S = \mathbb{R}^2 = \{(\text{latitude}, \text{longitude}\}$. Consider a real-valued stochastic process denoted as: $\{Z(s, \omega) : s \in S, \omega \in \Omega\}$

To understand this process, consider fixing each argument in turn. For a fixed $s_i \in S$, $Z(s_i, \cdot) : \Omega \to \mathbb{R}$ is a real-valued random variable, mapping from the sample space Ω to $\mathbb{R}$. If we instead fix a particular event from the sample space $\omega_i \in \Omega$ then $Z(\cdot, \omega_i) : S \to \mathcal{R}$ is a realisation (also known as a sample path) of the stochastic process, assigning a real value to every location. Here we use (s, x) to denote any arbitrary pair of (input,output) locations.

To convert a VAE into a stochastic process, instead of directly learning mappings of function values at a fixed set of locations we first learn a feature map and then a linear mapping from this feature map to the observed values of the function. The autoencoder learns a latent probabilistic representation over the linear mappings. These two ingredients correspond to the inputs to the stochastic process $Z(s, \omega)$: the feature map means that we can evaluate our stochastic process at any $s \in S$ and the latent space corresponds to the sample space Ω.

Let us assume, without loss of generality, we have N function draws where each function is evaluated at K different locations[1]. In the encoder we transform each observed location for a function s_i^k, where $i \in N$ and $k \in K$, to a fixed feature space through a transformation function $\Phi(s_i^k)$ that is shared for all $s \in S$ locations across all N functions. $\Phi(s)$ could be an explicit feature representation for an RKHS (e.g. an RBF network or a random Fourier feature basis (Rahimi & Recht, 2008)), a neural network of arbitrary construction or, as we use in the examples in this paper, a combination of both. Following this transformation, a linear basis β is used to predict function evaluations at an arbitrary set of locations s. The intuition behind these two transformations is to learn the association between locations and observations. In contrast to a standard VAE encoder that takes as input function evaluations, $[z_\mu, z_{sd}]^\top = e(\eta_e, x_i)$ and encodes these to a latent space, πVAE encoder first transforms the locations to a higher dimensional feature space via Φ, and then connects this feature space to outputs, x, through a linear mapping, β. The πVAE decoder takes $\mathcal{Z}$ from the encoder, and attempts to recreate β from a lower dimensional probabilistic embedding. This recreation, $\hat{\beta}$, is then used as a linear mapping with the *same* Φ to get a reconstruction of the outputs $\hat{x}$. It is crucial to note, that a single β vector is *not* learnt. Instead for each function in $i \in N$ a β_i is learnt. End-to-end πVAE is:

$$\begin{aligned} \hat{x}_{e,i}^k &= \beta_i^\top \Phi(s_i^k) \\ [z_\mu, z_{sd}]^\top &= e(\eta_e, \beta_i) \\ \mathcal{Z} &\sim \mathcal{N}(z_\mu, z_{sd}^2 \mathbb{I}) \\ \hat{\beta}_i &= d(\eta_d, \mathcal{Z}) \\ \hat{x}_{d,i}^k &= \hat{\beta}_i^\top \Phi(s_i^k) \end{aligned} \tag{5}$$

It can be seen from equation 5 that the central task is not,

[1]The number of evaluation across each function can change in our model, all equations would still hold if K is conditioned on $n \in N$ to be K_n.

as in the VAE, to recreate the input (note $\hat{x}$ in both ends of πVAE) but to reconstruct the linear parameters β that can map a shared Φ onto the inputs. End to end learning is performed in πVAE to estimate parameters for η_e, η_d, Φ and β. The evidence lower bound to be maximised is similar to that for the VAE, except the likelihood term now has an additional term for the reconstructing via feature transformation and linear mapping.

$$\underset{\eta_e,\eta_d,\Phi,\beta_i}{\arg\max}\ p(x_i^k|\beta_i, s_i^k, \phi, \eta_e) + p(x_i^k|\mathcal{Z}, s_i^k, \phi, \eta_d) - \mathrm{KL}\left(\mathcal{Z}\|\mathcal{N}(0,\mathbb{I})\right) \tag{6}$$

When assuming a Gaussian likelihood, equation 6 can be simplified to the following loss function:

$$\underset{\eta_e,\eta_d,\Phi,\beta}{\arg\min}\ (x_i^k - \beta_i^T\Phi(s_i^k))^2 + (x_i^k - \hat{\beta}_i{}^T\Phi(s_i^k))^2 + \mathrm{KL}\left(\mathcal{Z}\|\mathcal{N}(0,\mathbb{I})\right) \tag{7}$$

In equation 7, the first term ensures a minimum square error of the linear mapping (β) of Φ and the input. The second term ensures a minimum square error of the reconstructed ($\hat{\beta}$) linear mapping of Φ and the input. The third term is a KL loss to ensure the latent dimension is as close to $\mathcal{N}(0,\mathbb{I})$ as possible. Once again it is important to note the indices i. A different β_i is learnt for every function draw (x_i, s_i) but the same Φ is used. This means we need to learn η_e, η_d, Φ but also N βs, one for each sample function. While this may seem like a huge computational task, K is typically quite small (< 200) and so learning can be relatively quick (dominated by matrix multiplication of hidden layers).

The πVAE can be used as a generative model. Generating a function f is done by first, selecting input locations, s, and then drawing $z \sim \mathcal{N}(z_\mu, z_{sd}^2\mathbb{I})$. This sample z is then transformed into a sample through a deterministic transformation via the decoder and Φ, $f(\cdot) := d(z)^\top\Phi(\cdot)$. The πVAE can be used for inference on new data pairs (s_j, y_j), where the unnormalised posterior distribution is

$$p(\mathcal{Z}|d, y_j, s_j, \Phi) \propto p(y_j|d, s_j, \mathcal{Z}, \Phi)p(\mathcal{Z}) \tag{8}$$
$$p(z_\mu, z_{sd}|d, y_j, s_j, \Phi) \propto p(y_j|d, s_j, z_\mu, z_{sd}, \Phi)\mathcal{N}(z_\mu, z_{sd}^2\mathbb{I})$$

We note, (s_j, y_j) is a *set* of observed data, i.e., it is a collection of (input, output) locations from a new unknown function that is to be inferred. Further, the distinguishing difference between equation (3) and equation (8) is conditioning on input locations and Φ. It is Φ that ensures πVAE is a valid stochastic process. We formally prove this below:

Theorem 1 (Exchangeability) *Consider a permutation function π, locations $s_1, \ldots, s_n$, fixed and bounded functions Φ and d, and a multivariate Gaussian random variable $\mathcal{Z}$. Together, these define the random function $f(\cdot) = d(\mathcal{Z})^\top\Phi(\cdot)$. We claim:*

$$p(f(s_1), \ldots, f(s_n)) = p(f(s_{\pi(1)}), \ldots, f(s_{\pi(n)}))$$

Proof:

$$\begin{aligned} p(f(s_1), \ldots, f(s_n)) &= p(f(s_1), \ldots, f(s_n)|\mathcal{Z})p(\mathcal{Z}) \\ &= \prod_{i=1}^n p(d(\mathcal{Z})^\top\Phi(s_i)|\mathcal{Z})p(\mathcal{Z}) \\ &= p(f(s_{\pi(1)}, \ldots, f(s_{\pi(n)})|\mathcal{Z})p(\mathcal{Z}) \\ &= p(f(s_{\pi(1)}), \ldots, f(s_{\pi(n)})) \end{aligned}$$

Where the second line follows by the definition of f. □

Theorem 2 (Consistency) *Under the same conditions as Theorem 1, and given an extra location s_{n+1} we claim:*

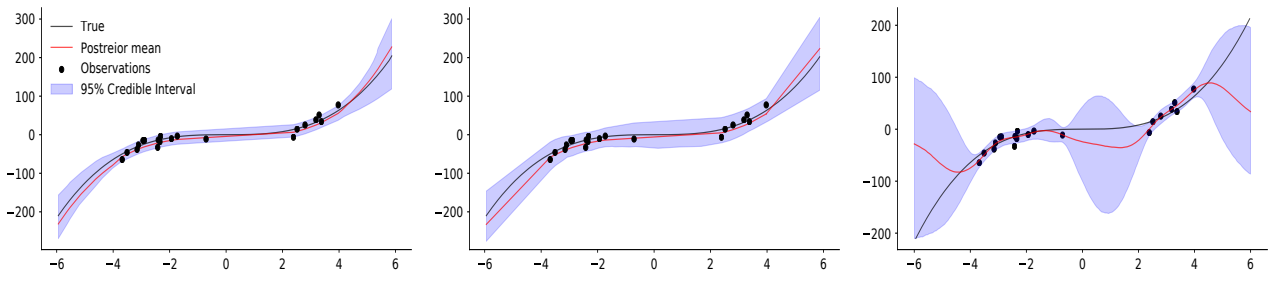
$$p(f(s_1), \ldots, f(s_n)) = \int p(f(s_1), \ldots, f(s_{n+1}))df(s_{n+1})$$

Proof:

$$\begin{aligned} &\int p(f(s_1), \ldots, f(s_{n+1}))df(s_{n+1}) \\ &= \int p(f(s_1), \ldots, f(s_{n+1})|\mathcal{Z})p(\mathcal{Z})df(s_{n+1}) \\ &= \int \prod_{i=1}^n p(d(\mathcal{Z})^\top\Phi(s_i)|\mathcal{Z})p(d(\mathcal{Z})^\top\Phi(s_{i+1})|\mathcal{Z})p(\mathcal{Z})df(s_{n+1}) \\ &= \prod_{i=1}^n p(d(\mathcal{Z})^\top\Phi(s_i)|\mathcal{Z})p(\mathcal{Z}) \int p(d(\mathcal{Z})^\top\Phi(s_{i+1})|\mathcal{Z})df(s_{n+1}) \\ &= \prod_{i=1}^n p(d(\mathcal{Z})^\top\Phi(s_i)|\mathcal{Z})p(\mathcal{Z}) = p(f(s_1), \ldots, f(s_n)) \quad \square \end{aligned}$$

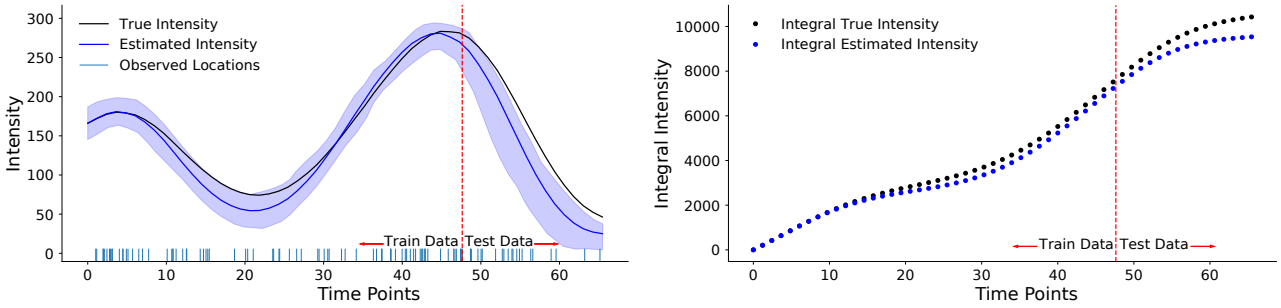
2.3. Examples

We first demonstrate the utility of our proposed πVAE model by fitting the simulated 1-D regression problem introduced in (Hernández-Lobato & Adams, 2015). The training points for the dataset are created by uniform sampling of 20 inputs, x, between $(-4, 4)$. The corresponding output is set as $y \sim \mathcal{N}(x^3, 9)$. We fit two different variants of πVAE, representing two different prior classes of functions. The first prior produces cubic monotonic functions and the second prior is a GP with an RBF kernel and a two layer neural network. We generated 10^4 different function draws from both priors to train the respective πVAE. One important consideration in πVAE is to chose a sufficiently expressive Φ, we used a RBF layer with trainable centres coupled with two layer neural network with 20 hidden units each. We compare our results against 20,000 Hamiltonian Monte Carlo (HMC) samples (Neal, 1993) implemented



(a) (b) (c)

Figure 2: The true underlying function is $y \sim \mathcal{N}(x^3, 9)$. (a) πVAE trained on a class of cubic functions, (b) πVAE trained on samples from a Gaussian process with RBF kernel and (c) is a Gaussian process with RBF kernel fit with HMC



(a) (b)

Figure 3: Inferring the intensity of a log Gaussian Cox Process. (a) compares the posterior distribution of the intensity estimated by πVAE to the true intensity function on train and test data. (b) compares the posterior mean of the cumulative integral over time estimated by πVAE to the true cumulative integral on train and test data.

using Stan (Carpenter et al., 2017). Details for the implementation for all the models can be found in the Appendix (Anonymous, 2020).

Figure 2(a) presents results for πVAE with a cubic prior, Figure 2(b) with the GP+NN prior, and Figure 2(c) for a standard RBF + HMC. Both the mean estimates and the uncertainty from πVAE variants are closer, and more constrained than the ones using HMC. Importantly, πVAE with cubic prior not only produces better point estimates but is the able to capture better uncertainty bounds. This demonstrates that πVAE can be used to incorporate domain knowledge about the functions being modelled.

In many scenarios, learning just the mapping of inputs to outputs is not sufficient as other functional properties are required to perform useful (interesting) analysis. For example, using point processes requires knowing the underlying intensity function, however, to perform inference we need to calculate the integral of that intensity function too. Calculating this integral, even in known analytical form, is very expensive. Hence, in order to circumvent the issue we use πVAE to learn both function values and its integral for the observed events. Figure 3 shows πVAE prediction for both the intensity and integral of a simulated 1-D Log Gaussian Cox Process (LGCP).

In order to train πVAE to learn the function space of 1-D LGCP functions, we first create a training set by drawing 10,000 different samples of intensity function using an RBF kernel for 1-D LGCP. For each of the drawn intensity function we choose an appropriate time horizon to sample 80 observed events (locations) from the intensity function. The πVAE is trained on the sampled 80 locations with their corresponding intensity and the integral. The πVAE therefore outputs both the instantaneous intensity and the integral of the intensity. The implementation details can be seen in the Appendix (Anonymous, 2020). For testing, we first draw a new intensity function (1-D LGCP) using the same mechanism used in training and sample 100 events (locations). As seen in Figure 3 our estimated intensity is very close to true intensity and even the estimated integral is close to the true integral. This example shows that the πVAE approach can be used to learn not only function evaluations but properties of functions.

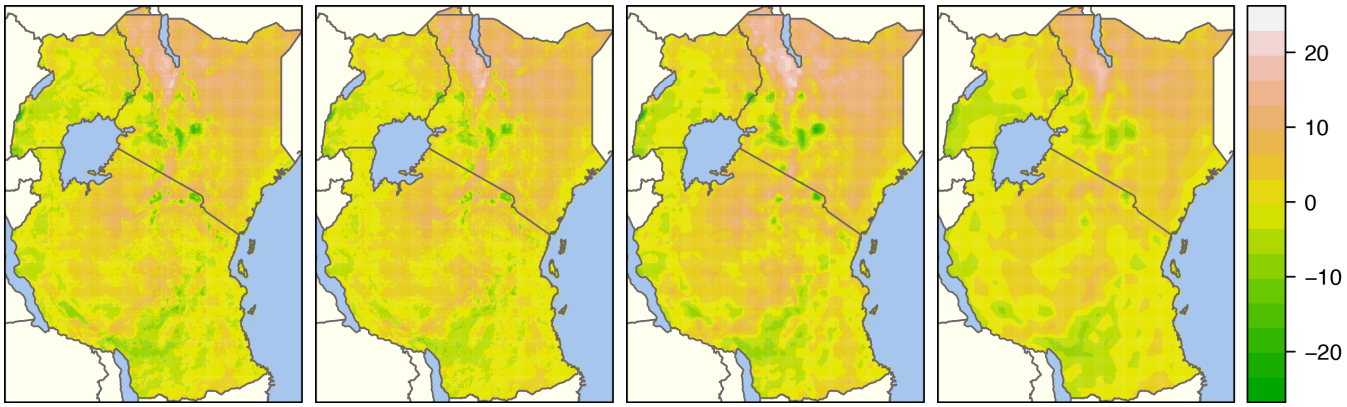


Figure 4: Deviation in land surface temperature for East Africa trained on 6000 random uniformly chosen locations (Ton et al., 2018). Plot (a) is the actual data, (b) is our πVAE approach (testing MSE: 0.38), (c) Is a full rank GP with Matérn $\frac{3}{2}$ kernel (testing MSE: 2.47) and (d) Is a low rank SPDE approximation with 1953 basis functions (Lindgren et al., 2011) and a Matérn $\frac{3}{2}$ kernel (testing MSE: 2.91). The πVAE not only has a substantially lower test error but captures fine scale features much better than a Gaussian process.

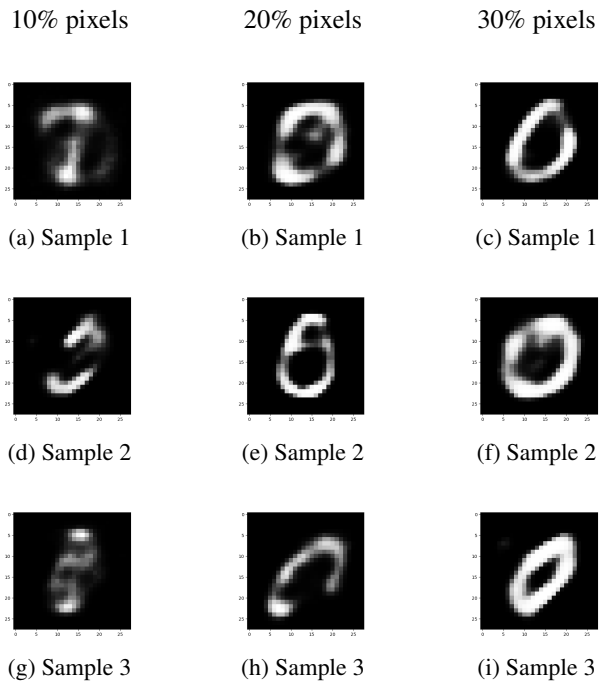
3. Results

Here we show applications of πVAE on three real world datasets. In our first example we use πVAE to predict the the deviation in land surface temperature in East Africa (Ton et al., 2018). We have the deviation in land surface temperatures for $\sim$89,000 locations across East Africa. Our training data consisted of 6,000 uniformly sampled locations. Temperature was predicted using only the spatial locations as inputs. Figure 4 shows the results of the ground truth (a), our πVAE (b), a full rank Gaussian process with Matérn kernel (Gardner et al., 2018), and low rank Gauss Markov random field (a widely used approach in the field of geostatistics) with $1,046$ ($\frac{1}{6}$th of the training size) basis functions (Lindgren et al., 2011; Rue et al., 2009). We train our πVAE model on 10^7 functions draws from 2-D GP with small lengthscales between 10^{-5} to 2. Φ was set to be a Matérn layer with 1,000 centres followed by a two layer neural network 100 hidden units. The πVAE latent dimension was set to 20. As seen in Figure 4, πVAE is able to captures small scale features and produces a far better reconstruction than the both full and low rank GP and despite having a much smaller latent dimension of 20 vs 6,000 (full) vs 1,046 (low). The testing error is substantially better than the full rank GP which begs the question, why does πVAE perform so much better than a GP, despite being trained on samples from a GP. One possible reason is that the extra hidden layers in Φ create a much richer structure that could capture elements of non-stationarity (Ton et al., 2018). Alternatively, the ability to use state-of-the-art MCMC and estimate a reliable posterior expectation might create resilience to overfitting. The training/testing error for πVAE is $0.07/0.38$, while the full rank GP is $0.002/2.47$. Therefore the training error is 37 times smaller in the GP, but the testing error is only 6 times smaller in πVAE suggesting that, despite marginalisation, the GP is still overfitting.

Method	RMSE	NLL
Full rank GP	0.099	-0.258
πVAE	0.112	0.006
SGPR ($m = 512$)	0.273	0.087
SVGP ($m = 1024$)	0.268	0.236

Table 1: Test results for πVAE, a full rank GP and two approximate algorithms SGPR and SVGP on *Kin40K*.

Table 1 compares πVAE on the *Kin40K* (Schwaighofer & Tresp, 2003) dataset to state-of-the-art full and approximate GPs, with results taken from Wang et al. (2019). The objective is to predict the distance of a robotic arm from the target given the position of all 8 links present on the robotic arm. In total we have 40,000 samples which are divided randomly into $\frac{2}{3}$ training samples and $\frac{1}{3}$ test samples. We train πVAE on 10^7 functions drawn from an 8-D GP, observed at 200 locations, where each of the 8 dimensions had values drawn uniformly from the range $(-2, 2)$ and lengthscale varied between 10^{-3} to 10. Once πVAE was trained on the prior function we use it to infer the posterior distribution for the training examples in *Kin40K*. Table 1 shows results for RMSE and negative log-likelihood (NLL) of πVAE against various GP methods on test samples. The full rank GP results reported in Wang et al. (2019) are better than those we obtain from πVAE, but we are competitive, and far better than the approximate GP methods. We also note here that the exact GP is based on maximum



(a) Sample 1 (b) Sample 1 (c) Sample 1

(d) Sample 2 (e) Sample 2 (f) Sample 2

(g) Sample 3 (h) Sample 3 (i) Sample 3

Figure 5: MNIST reconstruction after observing only 10, 20 or 30% of pixels from original data.

marginal likelihood while πVAE performs full Bayesian inference; all posterior checks were excellent showing calibration both in uncertainty and point estimates. For detailed diagnostics see the Appendix.

Finally we apply πVAE to the task of reconstructing MNIST digits, except from partial observations. Similar to the earlier temperature prediction task image completion can also be seen as a regression task in 2-D. The regression task is to predict the intensity of pixels given the pixel locations. We first train our πVAE on MNIST digits dataset (not partial) with 40 as the dimension for the latent dimension. As with previous example the decoder and encoder networks are made up of two layer neural networks. The hidden units for encoder are 256 and 128 for first and second layer respectively. For decoder the architecture of encoder is reversed.

Once we have trained πVAE with MNIST images in train set we now use images from test set for prediction. Images in testing set are sampled in such a way that only 10, 20 or 30% of pixel values are first shown to πVAE to fix the parameter for a specific test function and then πVAE is used to predict the intensity at all other pixel locations. As seen from Figure 5 the performance of πVAE increases with increase in pixel locations available during prediction but still even with 10% pixels our model is able to learn a decent approximation of the image. The uncertainty is prediction can be seen from the different samples produced by the model for the same data. As number of given location increases the variance between samples decreases with quality of image also increasing.

4. Discussion and Conclusion

In this paper we have proposed a novel VAE formulation of a stochastic process, with the ability to learn function classes and properties of functions. Our πVAEs typically have a small, uncorrelated latent dimension (5-50) of parameters, so Bayesian inference with MCMC is straightforward and highly effective at successfully exploring the posterior distribution. The accurate estimation of uncertainty is essential in many areas such as medical decision-making.

While approximate Bayesian inference has seen wide application in recent years, the accuracy of these posterior expectations is generally poor (Yao et al., 2018) (or unknown) when benchmarked against MCMC approaches. However, MCMC approaches are often unavailable in real-world problems due to poor scalability and difficulties arising from complex posteriors.

πVAE combines the power of deep learning to create high capacity function classes, while ensuring tractable inference using fully Bayesian MCMC approaches. Our 1-D example in Figure 2 demonstrates that an exciting use of πVAE is to incorporate domain knowledge about the problem. Monotonicity or complicated dynamics can be encoded directly into the prior (Caterini et al., 2018) on which πVAE is trained. Our log-Gaussian Cox Process example shows that not only functions can me modelled, but also properties of functions such as integrals. Perhaps the most surprising result is the performance of the πVAE on spatial interpolation. Despite being trained on samples from a Gaussian process, πVAE substantially outperforms a full rank GP. We conjecture this is due to the more complex structure of the feature representation Φ and due to a resilience to overfitting. The MNIST example highlights how πVAE can used as a generative model on a much smaller subset of training data.

There are costs to using πVAE, especially the large upfront cost in training. For complex priors, training could take days or weeks and will invariably require heuristics and parameter search inherent in applied deep learning to achieve good performance. However, once trained, a πVAE network is applicable on a wide range of problems, with the Bayesian inference MCMC step taking seconds or minutes. We will make available pretrained networks for various popular priors.

Future work should investigate the performance of πVAE on higher dimensional settings. Other stochastic processes, such as Dirichlet processes, could be considered.

References

Anonymous. πVAE: Encoding stochastic process priors with variational autoencoders (Appendix). In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, 2020.

Antoniak, C. E. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, pp. 1152–1174, 1974.

Betancourt, M., Byrne, S., Livingstone, S., and Girolami, M. The geometric foundations of Hamiltonian Monte Carlo. *Bernoulli*, 2017. ISSN 13507265. doi: 10.3150/16-BEJ810.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. In *32nd International Conference on Machine Learning, ICML 2015*, 2015. ISBN 9781510810587.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. Stan : A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1):1–32, 2017. ISSN 1548-7660. doi: 10.18637/jss.v076.i01.

Caterini, A. L., Doucet, A., and Sejdinovic, D. Hamiltonian variational auto-encoder. In *Advances in Neural Information Processing Systems*, 2018.

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. Visualization in Bayesian workflow. *Journal of the Royal Statistical Society. Series A: Statistics in Society*, 2019. ISSN 1467985X. doi: 10.1111/rssa.12378.

Gamelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. M. Conditional neural processes. In *35th International Conference on Machine Learning, ICML 2018*, 2018. ISBN 9781510867963.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Hawkes, A. G. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971. ISSN 00063444. doi: 10.1093/biomet/58.1.83.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313 (5786):504–507, 2006.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Huggins, J. H., Kasprzak, M., Campbell, T., and Broderick, T. Practical posterior error bounds from variational objectives. *arXiv preprint arXiv:1910.04102*, 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, S. M. A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive Neural Processes. *CoRR*, abs/1901.0, 2019.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 12 2014.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes (VAE, reparameterization trick). *ICLR 2014*, 2014.

Kingma, D. P. and Welling, M. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017.

Lindgren, F., Rue, H., and Lindström, J. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011. ISSN 13697412. doi: 10.1111/j.1467-9868.2011.00777.x.

Minka, T. P. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI01, pp. 362369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608001.

Mller, J., Syversveen, A. R., and Waagepetersen, R. P. Log gaussian cox processes. *Scandinavian Journal of Statistics*, 25(3):451–482, 1998. doi: 10.1111/1467-9469.00115.

Neal, R. Bayesian Learning for Neural Networks. *LECTURE NOTES IN STATISTICS -NEW YORK-SPRINGER VERLAG-*, 1996. ISSN 0930-0325.

Neal, R. M. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, Ontario, Canada, 1993.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X. URL http://www.worldcat.org/oclc/61285753.

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

Ritter, H., Botev, A., and Barber, D. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Ross, S. M. *Stochastic processes*, volume 2. John Wiley & Sons, 1996.

Roy, D. M. and Teh, Y. W. The Mondrian process. In *Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference*, 2009. ISBN 9781605609492.

Rue, H., Martino, S., and Chopin, N. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 71(2):319–392, 4 2009. ISSN 13697412. doi: 10.1111/j.1467-9868.2008.00700.x.

Schwaighofer, A. and Tresp, V. Transductive and inductive methods for approximate gaussian process regression. In *Advances in Neural Information Processing Systems*, pp. 977–984, 2003.

Ton, J.-F., Flaxman, S., Sejdinovic, D., and Bhatt, S. Spatial mapping with Gaussian processes and nonstationary Fourier features. *Spatial Statistics*, 2018. ISSN 22116753. doi: 10.1016/j.spasta.2018.02.002.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pp. 14622–14632, 2019.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011. ISBN 9781450306195.

Yao, J., Pan, W., Ghosh, S., and Doshi-Velez, F. Quality of uncertainty quantification for bayesian neural network inference. *arXiv preprint arXiv:1906.09686*, 2019.

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. Yes, but did it work?: Evaluating variational inference. In *35th International Conference on Machine Learning, ICML 2018*, 2018. ISBN 9781510867963.