Scalable Training of Inference Networks for Gaussian-Process Models

Jiaxin Shi [1] **Mohammad Emtiyaz Khan** [2] **Jun Zhu** [1]

Abstract

Inference in Gaussian process (GP) models is computationally challenging for large data, and often difficult to approximate with a small number of inducing points. We explore an alternative approximation that employs stochastic inference networks for a flexible inference. Unfortunately, for such networks, minibatch training is difficult to be able to learn meaningful correlations over function outputs for a large dataset. We propose an algorithm that enables such training by tracking a stochastic, functional mirror-descent algorithm. At each iteration, this only requires considering a finite number of input locations, resulting in a scalable and easy-to-implement algorithm. Empirical results show comparable and, sometimes, superior performance to existing sparse variational GP methods.

1. Introduction

Gaussian processes (GP) (Rasmussen & Williams, 2006) and their deep variants (Damianou & Lawrence, 2013) are powerful nonparametric distributions for both supervised (Williams & Rasmussen, 1996; Bernardo et al., 1998; Williams & Barber, 1998) and unsupervised machine-learning (Lawrence, 2005; Damianou et al., 2016). Such processes can generate smooth functions to model complex data and provide principled approaches for uncertainty quantification. Despite this, their application has been limited because Bayesian inference for such modeling requires inversion of a matrix which is computationally challenging (typically $\mathcal{O}(N^3)$ for N data examples).

Many methods have been proposed to tackle this issue, and they mostly rely on finding a small number of *inducing points* in the input[1] space to reduce the cost of matrix inversion (Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009). Methods that employ variational inference to estimate inducing points (Titsias, 2009) can scale well to large data by minibatch stochastic-gradient methods (Hensman et al., 2013), but the quality of posterior approximations obtained could be limited since the number of inducing points needs to be small, even when the data are extremely large (Cheng & Boots, 2017). Optimization of inducing points is another challenging problem when the objective is nonconvex and stochastic-gradient methods can get stuck (Bauer et al., 2016).

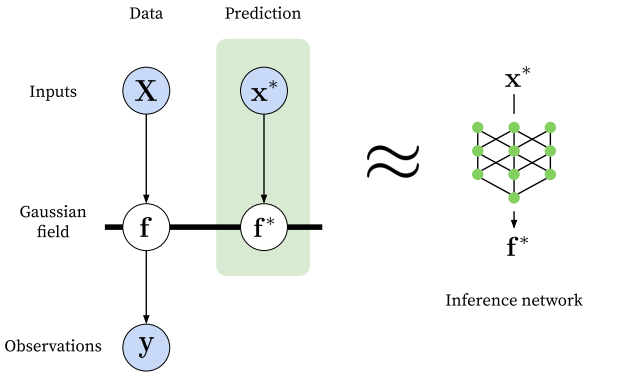


Figure 1. Inference networks for GPs are an alternative way to predict output $\mathbf{f}^*$ given test inputs $\mathbf{x}^*$.

A relatively less-explored approach is to approximate the GP posterior distribution by using function approximators, such as deep neural networks (Sun et al., 2019). By introducing randomness in the parameters of such *inference networks*, it is possible to generate functions similar to those generated by the posterior distribution, as illustrated in Fig. 1. By directly approximating the functions, we can obtain more flexible alternatives to sparse variational approaches. Unfortunately, unlike sparse methods, training with minibatches is challenging in the function space. The difficulty is to be able to maintain meaningful correlations over function outputs while only looking at a handful of examples in each iteration. Scalable training of such inference networks is extremely important for them to be useful for flexible posterior inference in GP models.

[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, THBI Lab, Tsinghua University, Beijing, China [2]RIKEN Center for Advanced Intelligence project, Tokyo, Japan. Correspondence to: Jiaxin Shi <shijx15@mails.tsinghua.edu.cn>, Jun Zhu <dcszj@tsinghua.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]For supervised learning, these can be features, while for unsupervised learning, these could be latent vectors.

In this paper, we propose an algorithm to scalably train the network by tracking an adaptive Bayesian filter defined in the function space. The filter is obtained by using a stochastic, functional mirror-descent algorithm which is guaranteed to converge to the exact posterior process but is computationally intractable. By bootstrapping from approximations given by inference networks, we can use stochastic gradients to train the network with minibatches of data. We demonstrate training of various types of networks, such as those based on random feature expansions and deep neural networks. Our results show that the problem caused by incorrectly minibatching in previous works is fixed by our method. Unlike exist works that use neural networks to parameterize deep kernel in the GP prior (Wilson et al., 2016), our variational treatment prevents overfitting and allows arbitrarily complex networks, which often achieve better performance than inducing-point approaches in regression tasks. Finally, we show that our method is a more flexible alternative for GP inference than existing sparse methods, which allows us to train deep ConvNets for the inference of GPs induced from infinite-width Bayesian ConvNets.

1.1. Related work

For a scalable inference in GP models, plenty of work has been done on developing sparse methods (Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009; Hensman et al., 2013). These works have made GPs a viable choice for practical problems by making minibatch training possible. Our work takes a different approach than these methods by using inference networks for posterior approximation. The computation complexity of our method is similar to sparse methods, but our posterior approximations are much more flexible. Currently, it is challenging to train such inference networks in the function space while achieving a good performance. Our work fills this gap and shows that it is indeed possible to scalably train them and get a similar, and sometimes better, performance than sparse GP methods.

A few recent approaches have used inference networks for posterior approximations in the function space, although they have not directly applied it to inference in GP models. For example, Neural Processes (NP) (Garnelo et al., 2018a;b) uses inference networks on a model with a learned prior, while Variational Implicit Processes (VIP) (Ma et al., 2018), in a similar setup, uses a GP for approximations. Hafner et al. (2018) design a function-space prior called the noise contrastive prior. The work of Sun et al. (2019) is perhaps the closest to our approach, but they use an heuristic procedure for minibatch training. Our work proposes a scalable minibatch training method which is useful for all of these existing works.

The idea of tracking a learning process to *distill* information has been used in many previous approaches, e.g., Bui et al. (2017) use streaming variational Bayes for inference in sparse GPs, while Balan et al. (2015) track stochastic gradient Langevin dynamics (SGLD) (the *teacher*) to distill information into a neural network (the *student*). Our method can also be interpreted in a teacher-student framework, where the teacher can be obtain from the student network by taking a stochastic mirror descent step, which is much cheaper than simultaneously running another inference algorithm like SGLD.

Some of our inference networks are based on random feature expansion and are related to existing work on spectrum approximations for GPs (Lázaro-Gredilla et al., 2010; Gal & Turner, 2015; Hensman et al., 2017). Among them Lázaro-Gredilla et al. (2010) does function-space inference and is similar to our work, but is a full-batch algorithm.

2. Bayesian Inference in GP Models

We start by discussing the challenges associated with existing methods for inference in GP models, and then discuss the difficulty in scalable training of inference networks.

2.1. Gaussian Processes

A Gaussian Process (GP) is a stochastic process defined using a mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right). \tag{1}$$

A remarkable property of GPs is that, for any finite number of inputs $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, the marginal distribution of the function values $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$ follow a multivariate Gaussian distribution: $\mathbf{f} \sim \mathcal{N}\left(\mathbf{m}_\mathcal{D}, \mathbf{K}_{\mathcal{D},\mathcal{D}}\right)$, where $\mathcal{D} := [1, 2, \ldots, N]$ are the indices of the data examples, $\mathbf{m}_\mathcal{D}$ is a vector with entries $m(\mathbf{x}_i)$ and $\mathbf{K}_{\mathcal{D},\mathcal{D}}$ is a matrix with (i, j)'th entry as $k(\mathbf{x}_i, \mathbf{x}_j)$. This property can be utilized to obtain function approximations for the modeling of complex data. For example, for regression analysis given input features $\mathbf{X}$, the function values $\mathbf{f}$ can be used to model the output mean: $\mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I})$ where σ^2 is the noise variance. Correlations in the data vector $\mathbf{y}$ can now be explained through the correlation defined using the GP prior in the function space. This approach is widely used to design nonlinear methods for supervised and unsupervised learning.

Given such a prior distribution, the goal of Bayesian inference is to compute the posterior distribution over the function $f(\mathbf{x})$ evaluated at arbitrary test inputs $\mathbf{x}$. For Gaussian likelihoods such as in regression, due to the conjugacy, the posterior distribution takes a convenient closed-form solution, thus the predictive distribution at a test location $\mathbf{x}^*$ is available in closed-form:

$$f(\mathbf{x}^*)|\mathbf{y} \sim \mathcal{N}\left(\mathbf{k}_{*,\mathcal{D}}^T(\mathbf{K}_{\mathcal{D},\mathcal{D}} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}_\mathcal{D}),\right.$$
$$\left. k_{**} - \mathbf{k}_{*\mathcal{D}}^T(\mathbf{K}_{\mathcal{D},\mathcal{D}} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_{\mathcal{D},*}\right), \tag{2}$$

where $[\mathbf{k}_{*\mathcal{D}}]_i = k(\mathbf{x}^*, \mathbf{x}_i)$ and $k_{**} = k(\mathbf{x}^*, \mathbf{x}^*)$. Equation (2) is difficult to compute in practice when N is large. The computational cost of matrix inversion is in $\mathcal{O}(N^3)$. This inversion is only possible when N is of moderate size, usually only a few thousands.

2.2. Sparse Approximations for GPs

Sparse-approximation methods reduce the computation cost by choosing a small number of M points, where $M \ll N$, to perform the matrix inversion. These points are known as the *inducing points*. A variety of such methods have been proposed (Quiñonero-Candela & Rasmussen, 2005) and they mostly differ in the manner of selection of these points and the kind of approximations used for the matrix inverse. Among these, methods based on variational inference are perhaps one of the most popular (Titsias, 2009). Given M inducing points $\mathbf{Z} := [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_M]^\top$ and their function values $\mathbf{u} := [f(\mathbf{z}_1), f(\mathbf{z}_2), \dots, f(\mathbf{z}_M)]^\top$, these methods approximate the posterior distribution $p(\mathbf{f}, \mathbf{u}|\mathbf{y}, \mathbf{X}, \mathbf{Z})$ with a variational distribution $q(\mathbf{f}, \mathbf{u})$. By restricting the variational approximation to be $q(\mathbf{f}, \mathbf{u}) := q(\mathbf{u})p(\mathbf{f}|\mathbf{u})$, the variational lower bound is greatly simplified,

$$\mathcal{L}(q, \mathbf{Z}) := \mathbb{E}_{q(\mathbf{u})p(\mathbf{f}|\mathbf{u})} [\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[q(\mathbf{u})\|p(\mathbf{u})]. \quad (3)$$

For GP regression, the optimal $q(\mathbf{u})$ is a Gaussian whose parameters can be obtained in closed-form (Titsias, 2009). Even though the computation of $q(\mathbf{u})$ scales linearly with N, its covariance matrix can be inverted in $\mathcal{O}(M^3)$ which reduces the prediction cost drastically. The linear dependence on N can be further reduced by using a minibatch training proposed in Hensman et al. (2013), which makes the method scale well to large data. Sparse variational methods can scale well, and also perform reasonably with enough number of properly chosen inducing-points.

In practice, however, the number of inducing points is limited to make the matrix inversion feasible. This limits the flexibility of the posterior distribution whose complexity might grow with the complexity of the problem. Finding good inducing points is another challenging issue. The landscape of the lower bound with respect to $\mathbf{Z}$ presents a difficult optimization problem, and, so far, there are no good methods for this problem.

2.3. Function Space View and Inference Networks

The sparse variational method discussed above can be derived by using a function-space view of GPs. As discussed in several recent works (Cheng & Boots, 2016; 2017; Mallasto & Feragen, 2017), a GP has a dual representation in a separable Banach space, which contains the RKHS $\mathcal{H}$ induced by the covariance kernel. This view motivates to directly apply variational inference in the function space. Following Cheng & Boots (2016), if we denote the dual representation of the posterior and variational distributions by $p(f|\mathbf{y})$ and $q(f)$ respectively, then the variational objective in the function space can be written as following:

$$\mathcal{L}(q(f)) = \mathbb{E}_{q(f)} [\log p(\mathbf{y}|f)] - \mathrm{KL}\,[q(f)\|p(f)]\,. \quad (4)$$

We recover the sparse variational GP problem in (3) when we restrict $q(f) \propto p(f_*|\mathbf{u})q(\mathbf{u})|\mathbf{K}_{\mathbf{u}}|^{1/2}|\mathbf{K}_* - \mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u}*}|^{1/2}$, where f_* represents the function outputs not covered by $\mathbf{u}$. The two determinants arise from the change of measure from function f to its outputs: $f(\mathbf{x}) = \langle f, k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}}$.

We can improve the flexibility of the approximations by employing better choices of $q(f)$. Function approximators with random parameters, such as stochastic neural networks, are such alternatives, where we can learn to generate functions that mimic the samples from the posterior. This is illustrated in Fig. 1. Drawing an analogy to the networks used for inference in deep generative models (Kingma & Welling, 2013), we call them *inference networks*. When trained well, such networks can yield much more flexible posterior approximations than sparse variational approximation which is restricted by the choice of inducing points.

Unfortunately, training inference networks is much more challenging than sparse methods. The sparse approach simplifies the problem to a parametric form where we only need to deal with a small number inducing points and data examples. It is challenging to design a similar procedure without any sparse assumption on $q(f)$.

Existing approaches have mostly relied on heuristic procedures to solve this problem. For example, a recent approach of Sun et al. (2019) proposes to match marginal distributions of $q(f)$ and $p(f|\mathbf{y})$ at a finite number of *measurement points* $\mathbf{X}_{\mathcal{M}} := [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M]^\top$ sampled from a distribution $c(\mathbf{x})$, by minimizing[2] $\mathrm{KL}\,[q(\mathbf{f}_{\mathcal{M}})\|p(\mathbf{f}_{\mathcal{M}}|\mathbf{y})]$ where $\mathbf{f}_{\mathcal{M}}$ are the function values at $\mathbf{X}_{\mathcal{M}}$. This is a reasonable criterion to match two GPs, since they are completely determined by their first two moments and the solution is unique when $M > 2$ (Sun et al., 2019). However, this is challenging to optimize due to the dependence of the term $p(\mathbf{f}_{\mathcal{M}}|\mathbf{y})$ on the whole dataset $\mathbf{y}$. Sun et al. (2019) propose to pick subsets of data at each iteration but then the minimization problem no longer corresponds to matching $q(f)$ and $p(f|\mathbf{y})$ faithfully. Many other existing works on function-space inference with neural networks face similar challenges when it comes to minibatch training (Wang et al., 2019; Ma et al., 2018; Hafner et al., 2018). Such scalable training of inference network, while maintaining meaningful correlations in function outputs for a large dataset, remains a challenging problem.

[2] We drop the dependence of p and q on training data $\mathbf{X}$ and sampled locations $\mathbf{X}_{\mathcal{M}}$.

3. Scalable Training of Inference Networks

We present a new algorithm to scalably train inference networks by using minibatches. Our main idea is to track a stochastic, functional mirror descent algorithm to enable efficient minibatch training. We start with a brief description of the mirror descent algorithm.

3.1. Stochastic Functional Mirror Descent Algorithm

We follow the functional mirror-descent method proposed in Dai et al. (2016); Cheng & Boots (2016) to optimize (4). The functional mirror descent algorithm is an extension of gradient descent where gradient steps are taken in a function space and the length of the steps is measured by using a Bregman divergence (e.g., KL) instead of a Euclidean distance. A stochastic version of this algorithm is analogous to stochastic gradient descent where a minibatch of data could be used to build a stochastic approximation of the functional gradient. The method takes the following form:

$$q_{t+1} = \arg\max_q \int \hat{\partial}\mathcal{L}(q_t)q(f)df - \frac{1}{\beta_t}\mathrm{KL}\left[q\|q_t\right]. \quad (5)$$

where t is the iteration, $\beta_t > 0$ is the learning rate, $q_t := q_t(f)$ is the previous approximation, and $\hat{\partial}\mathcal{L}(q_t) = N\log p(y_n|f) + \log p(f) - \log q_t(f)$ is an unbiased stochastic approximation of the functional gradient of $\mathcal{L}(q)$ at $q = q_t$ obtained by randomly sampling a data example n. An attractive property is that there is a closed-form solution given as follows,

$$q_{t+1}(f) \propto p(y_n|f)^{N\beta_t}p(f)^{\beta_t}q_t(f)^{1-\beta_t}. \quad (6)$$

This update can be seen as an *adaptive Bayesian filter* where the previous posterior approximation $q_t(f)$ is used to modify the prior distribution $p(f)$ and a random likelihood is used to update the posterior approximation.

Each step of this algorithm only requires subsampling a single data point, which makes it attractive for our purposes. The algorithm is also guaranteed to converge to the true posterior $p(f|\mathbf{y})$ as discussed in Dai et al. (2016), where a particle-based approach is proposed for the update in (6). Khan & Lin (2017) used a parametric version where the update can be performed analytically. Different from our case, in these works the random variable is a vector, whose density can be represented in compact form by using particles or exponential families. Unfortunately, this is intractable for $q(f)$. Our approach is to instead use an inference network to implement this filter. We will use a version of (6) to compute stochastic gradients and update the parameters of the inference network. This is explained next.

3.2. Minibatch Training of Inference Networks

We propose a tractable approximation to (6) by bootstrapping from the inference network at each iteration. We denote the inference network by $q_\gamma(f)$ where γ is the set of parameters that we wish to estimate. We assume that we can evaluate the network at a finite set of M points $\mathbf{X}_\mathcal{M} := [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M]^\top$ and that, just like a GP, the evaluated points follow a Gaussian distribution, i.e., $q_\gamma(\mathbf{f}_\mathcal{M}) = \mathcal{N}(\boldsymbol{\mu}_\mathcal{M}, \boldsymbol{\Sigma}_\mathcal{M})$, where $\boldsymbol{\mu}_\mathcal{M}$ and $\boldsymbol{\Sigma}_\mathcal{M}$ are the mean and covariance that depend on the parameter γ. In section 4, we will give many examples of inference networks that have this property.

With such a $q_\gamma(f)$, we hope to track (6), so that $q_{\gamma_{t+1}}(f)$ moves closer to the true posterior process than $q_{\gamma_t}(f)$. For this purpose, an obvious solution is to replace $q_t(f)$ by $q_{\gamma_t}(f)$, i.e., we can bootstrap from the current posterior approximation given by the inference network:

$$\hat{q}_{t+1}(f) \propto p(y_n|f)^{N\beta_t}p(f)^{\beta_t}q_{\gamma_t}(f)^{1-\beta_t}. \quad (7)$$

The idea of bootstrap has long been used in particle filtering (Doucet et al., 2001) to obtain better posterior approximations. An attractive property of (7) for GP regression is that, given inputs $\mathbf{X}_\mathcal{M}$, all the quantities in the right hand side follow a Gaussian distribution, therefore $\hat{q}_{t+1}$ has a Gaussian distribution whose mean and covariance are available in closed-form. The last two terms in (7) are Gaussian and can be multiplied to get the new GP prior:

$$\begin{aligned} &p(\mathbf{f}_\mathcal{M}, f_n)^{\beta_t}q_{\gamma_t}(\mathbf{f}_\mathcal{M}, f_n)^{1-\beta_t} \\ &:= \mathcal{N}\left(\begin{bmatrix}\widetilde{\mathbf{m}}_\mathcal{M}\\ \widetilde{\mathbf{m}}_n\end{bmatrix}, \begin{bmatrix}\widetilde{\mathbf{K}}_{\mathcal{M},\mathcal{M}} & \widetilde{\mathbf{K}}_{\mathcal{M},n}\\ \widetilde{\mathbf{K}}_{n,\mathcal{M}} & \widetilde{\mathbf{K}}_{n,n}\end{bmatrix}\right), \quad (8)\\ &\propto \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathcal{M},\mathcal{M}} & \mathbf{K}_{\mathcal{M},n}\\ \mathbf{K}_{n,\mathcal{M}} & \mathbf{K}_{n,n}\end{bmatrix}\right)^{\beta_t} \\ &\quad\times \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_\mathcal{M}\\ \boldsymbol{\mu}_n\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\mathcal{M},\mathcal{M}} & \boldsymbol{\Sigma}_{\mathcal{M},n}\\ \boldsymbol{\Sigma}_{n,\mathcal{M}} & \boldsymbol{\Sigma}_{n,n}\end{bmatrix}\right)^{(1-\beta_t)}, \quad (9)\end{aligned}$$

where $\widetilde{\mathbf{m}}$ and $\widetilde{\mathbf{K}}$ denotes the new mean and covariance. Using this in (7) and multiplying by the Gaussian likelihood, we get the following new GP regression problem, which has a closed-form solution similar to (2):

$$\begin{aligned}\hat{q}_{t+1}(\mathbf{f}_\mathcal{M}, f_n) &\propto \mathcal{N}(y_n|f_n, \sigma^2/(N\beta_t)) \\ &\times \mathcal{N}\left(\begin{bmatrix}\widetilde{\mathbf{m}}_\mathcal{M}\\ \widetilde{\mathbf{m}}_n\end{bmatrix}, \begin{bmatrix}\widetilde{\mathbf{K}}_{\mathcal{M},\mathcal{M}} & \widetilde{\mathbf{K}}_{\mathcal{M},n}\\ \widetilde{\mathbf{K}}_{n,\mathcal{M}} & \widetilde{\mathbf{K}}_{n,n}\end{bmatrix}\right). \quad (10)\end{aligned}$$

The marginal $\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})$ can be read from this directly. Though we have access to any finite marginal distribution of $\hat{q}_{t+1}(f)$, mapping this to the inference network parameters is difficult. We can use the approach of Sun et al. (2019) to match the marginals of the $q_\gamma(f)$ and $\hat{q}_{t+1}(f)$ at finite number of *measurement points* $\mathbf{X}_\mathcal{M}$ sampled from a distribution $c(\mathbf{x})$, i.e., we update γ using the gradient of the KL divergence as shown below, where η is the learning rate:

$$\gamma_{t+1} = \gamma_t - \eta\nabla_\gamma \mathrm{KL}\left[q_\gamma(\mathbf{f}_\mathcal{M})\|\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})\right]|_{\gamma=\gamma_t}. \quad (11)$$

When the likelihood is non-Gaussian, $\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})$ does not have a closed-form expression. We propose to upper bound $\mathrm{KL}\left[q_\gamma(\mathbf{f}_\mathcal{M})\|\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})\right]$ with the KL divergence between the two joint distributions $\mathrm{KL}\left[q_\gamma(\mathbf{f}_\mathcal{M}, f_n)\|\hat{q}_{t+1}(\mathbf{f}_\mathcal{M}, f_n)\right]$. Minimizing this is equivalent to maximizing:

$$\mathcal{L}_t(q_\gamma; q_{\gamma_t}, \mathbf{X}_\mathcal{M}) = \mathbb{E}_{q_\gamma(\mathbf{f}_\mathcal{M}, f_n)}\left[N\beta_t \log p(y_n|f_n) + \beta_t \log p(\mathbf{f}_\mathcal{M}, f_n) + (1-\beta_t)\log q_{\gamma_t}(\mathbf{f}_\mathcal{M}, f_n) - \log q_\gamma(\mathbf{f}_\mathcal{M}, f_n)\right].$$

Our method is similar in spirit to RL methods such as temporal difference learning with function approximators (Sutton & Barto, 1998), which also employs stochastic gradients to bootstrap from existing value function approximation. Their success indicates that taking a gradient step with a small step size here might ensure good performance in practice.

3.3. Algorithm

We name our algorithm *Gaussian Process Inference Networks* (GPNet), which is summarized in Algorithm 1. For each iteration of our algorithm, the stochastic mirror descent update is computed by subsampling a datapoint from the training set, then the inference network is trained to track the update at a set of measurement locations sampled from $c(\mathbf{x})$. Though we have described the algorithm using a single data example, it is straightforward to extend it to minibatches.

The computation cost in case of GP regression is the cost of matrix inversion which is $\mathcal{O}(M^3)$. Note that, unlike sparse methods, M does not have to be large for a flexible inference. The inference network can be a complex model containing neural networks which can be very flexible. Our procedure essentially uses M locations to be able to compute stochastic gradients to update the parameters of the network.

Choice of $c(\mathbf{x})$: Previous works on function-space inference (Sun et al., 2019; Wang et al., 2019; Hafner et al., 2018) have studied ways to sample the measurement points. The general approach is to apply uniform sampling in the input domain for low-dimensional problems; while for high-dimensional input space, we can sample "near" the training data by adding noise to them. A useful trick for RBF kernels is to set $c(\mathbf{x})$ as the training distribution convolved with the kernel. In applications where the input region of test points is known, we can set the $c(\mathbf{x})$ to include it.

Hyperparameter selection: We set $\beta_t = \beta_0(1+\xi\sqrt{t})^{-1}$ to ensure that the original stochastic mirror descent converges. Typical values are $\{1, 0.1, 0.01\}$ for β_0, and $\{1, 0.1\}$ for ξ. We can update GP hyperparameters when needed in an online fashion using the lower bound of minibatch log marginal likelihood: $\sum_{i\in B}\mathbb{E}_{q_t}\sum_{i\in B}\log p(y_i|f_i) - \mathrm{KL}\left[q_t(f_i)\|p(f_i)\right]$, which is similar to sparse variational methods. In our experiments the learning rate for GP hyperparameters is the same as η.

Algorithm 1 GPNet for supervised learning

Input: $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, $c(\mathbf{x})$, M, T, β, η.
Initialize the inference network q_γ.
for $t = 1, \ldots, T$ **do**
  Randomly sample a training data $(\mathbf{x}_n, y_n)$.
  Sample $\mathbf{X}_\mathcal{M} = (\mathbf{x}_1, \ldots, \mathbf{x}_M)$ from $c(\mathbf{x})$.
  if Gaussian likelihood **then**
    Compute $\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})$ using (10).
    $\gamma_{t+1} \leftarrow \gamma_t - \eta\nabla_\gamma \mathrm{KL}\left[q_\gamma(\mathbf{f}_\mathcal{M})\|\hat{q}_{t+1}(\mathbf{f}_\mathcal{M})\right]$.
  else
    $\gamma_{t+1} \leftarrow \gamma_t + \eta\nabla_\gamma \mathcal{L}_t(q_\gamma; q_{\gamma_t}, \mathbf{X}_\mathcal{M})$.
  end if
end for
return q_{γ_t}.

4. Examples of Inference Networks for GPs

So far we have used $q_\gamma(f)$ to denote the inference networks, without discussing how to construct them. Below we explore several types of networks that can be used in GPNet.

4.1. Bayesian Neural Networks (BNN)

A well-known fact in the community is: The function defined by a single-layer fully-connected neural network (NN) with infinitely many hidden units and independent weight randomness is equivalent to a GP (Neal, 1995). Recently, the result is extended to deep NNs (Lee et al., 2018; Matthews et al., 2018; Garriga-Alonso et al., 2019; Novak et al., 2019). This equivalence has motivated the use of BNNs as inference networks to model the distribution of functions (Sun et al., 2019; Wang et al., 2019). Given a neural network $g(\mathbf{x}; \Omega)$, where Ω denotes the network weights, a BNN is constructed by introducing weight randomness: $\Omega \sim \mathcal{N}(\Omega_0, \mathbf{V})$. Typically $\mathbf{V}$ is a factorized or matrix-variate Gaussian that factorizes across layers. In this case the inference network parameters should be $\gamma = \{\Omega_0, \mathbf{V}\}$.

However, this approach has several problems. First, the output density of BNN is intractable. In Flam-Shepherd et al. (2017); Wang et al. (2019) this difficulty is addressed by approximating the output distribution as a Gaussian and estimating the moments from samples, but drawing samples is costly due to many forward passes through the NN. The covariance estimate will present large variance for the typical sample size we can afford. In Sun et al. (2019) the situation is improved by directly estimating the gradients $\nabla_{\mathbf{f}_\mathcal{M}} \log q(\mathbf{f}_\mathcal{M})$ instead of the moments, with a low-variance but biased gradient estimator (Shi et al., 2018). However, the sample size still needs to be hundreds. Moreover, because the variational process defined by BNNs with finite-width is not guaranteed to be a GP, the typical solution by matching the finite marginal distributions may fail due to violations of the condition.

4.2. Tractable Variants

Given the problems faced with BNNs, we explore two more types of inference networks that are equivalent to GPs. Both approaches naturally arise from the feature-space representation of GPs. It is known that for a Bayesian linear regression with input feature $\boldsymbol{\phi}(\mathbf{x})$ and Gaussian weights $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, the output distribution is equivalent to a GP with the covariance $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma} \boldsymbol{\phi}(\mathbf{x}')$ (Rasmussen & Williams, 2006). In general, any positive definite kernel $k(\mathbf{x}, \mathbf{x}')$ can be written as the inner product of two feature maps $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{x}')$. As long as we know the $\boldsymbol{\phi}(\mathbf{x})$ that corresponds to the kernel, we can interpret our GP latent function $f \sim \mathcal{GP}(0, k)$ as a parametric model:

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{12}$$

We could define the variational process in a similar form:

$$q(f) : f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}), \tag{13}$$

where $\{\mathbf{m}, \mathbf{V}\}$ are the parameters of the inference network.

Random Feature Expansion (RFE) For GPs with stationary kernels (i.e., kernels that only depend on the difference between inputs, $k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x} - \mathbf{x}')$), Bochner's theorem guarantees that the covariance function can be written as a Fourier transform:

$$k(\mathbf{x}, \mathbf{x}') = \int e^{i\mathbf{s}^\top (\mathbf{x} - \mathbf{x}')} p(\mathbf{s}) d\mathbf{s}, \tag{14}$$

where $p(\mathbf{s})$ is a spectral density in one-to-one correspondence with ψ. Random Fourier features (Rahimi & Recht, 2008) is an approximation to kernel methods which gives explicit feature maps. The key observation is that eq. (14) can be approximated by Monte-Carlo:

$$k(\mathbf{x}, \mathbf{x}') \approx \frac{1}{M} \sum_{m=1}^{M} \cos(\mathbf{s}_m^\top (\mathbf{x} - \mathbf{x}')), \quad \mathbf{s}_{1:M} \sim p(\mathbf{s}),$$

where the imaginary part is zero. Defining $\boldsymbol{\phi}_r(\mathbf{x}) = \frac{1}{\sqrt{M}} [\cos(\mathbf{s}_1^\top \mathbf{x}), \ldots, \cos(\mathbf{s}_M^\top \mathbf{x}), \sin(\mathbf{s}_1^\top \mathbf{x}), \ldots, \sin(\mathbf{s}_M^\top \mathbf{x})]^\top$, we can use it as the approximate feature map: $k(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\phi}_r(\mathbf{x})^\top \boldsymbol{\phi}_r(\mathbf{x}')$. When using random Fourier features, the inference network $\mathbf{w}^\top \boldsymbol{\phi}_r(\mathbf{x})$ is a neural network with one hidden layer. The activation functions for the hidden layer are cos and sin. $\mathbf{s}_1, \ldots, \mathbf{s}_M$ and $\mathbf{w}$ serve as the input-to-hidden and the hidden-to-output weights, respectively. This architecture is called *Random Feature Expansion* in Cutajar et al. (2016), where they use a multi-layer stack to mimic a deep GP prior, though inference is still in the weight space. As done in their work, we relax $\mathbf{s}_1, \ldots, \mathbf{s}_M$ to be trainable so that the inference network parameters are $\gamma = \{\mathbf{m}, \mathbf{V}, \mathbf{s}_{1:M}\}$.

Neural Tangent Kernel (NTK) It is not always easy to find an inner-product form of a given kernel except the stationary case discussed. We may need a black-box approach, e.g., to parameterize $\boldsymbol{\phi}(\mathbf{x})$ with a function approximator such as neural networks with general nonlinearities (e.g., ReLU and $\tanh$) and fit it during inference. However, in practice we found that a plain NN can hardly approximate the feature map of even a simple kernel such as RBF kernels. The reason is that for many kernels, different dimensions of the feature map $\boldsymbol{\phi}(\mathbf{x})$ often represent components with different frequencies, which can hardly be captured by different output units in $\tanh$ or ReLU networks. Instead of using features given by NN outputs, we consider another kind of features: the vector of how much information stored in each weight parameter. Inspired by the Fisher kernel (Jaakkola & Haussler, 1999), we can measure this information by the gradients of network outputs with respect to weights $\nabla_{\boldsymbol{\Omega}} g(\mathbf{x}; \boldsymbol{\Omega})$. Plugging into $\boldsymbol{\phi}(\mathbf{x})$, we get the *Neural Tangent Kernel* (Jacot-Guillarmod et al., 2018):

$$k_{\text{NTK}}(\mathbf{x}, \mathbf{x}') = \nabla_{\boldsymbol{\Omega}} g(\mathbf{x}; \boldsymbol{\Omega}_0)^\top \mathbf{V} \nabla_{\boldsymbol{\Omega}} g(\mathbf{x}; \boldsymbol{\Omega}_0).$$

The performance of an inference network parameterized in this way is similar to a BNN because the NTK can be interpreted as introducing weight randomness on a first-order expansion of neural networks:

$$g(\mathbf{x}; \boldsymbol{\Omega}) = g(\mathbf{x}; \boldsymbol{\Omega}_0) + \nabla_{\boldsymbol{\Omega}} g(\mathbf{x}; \boldsymbol{\Omega}_0)(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0),$$

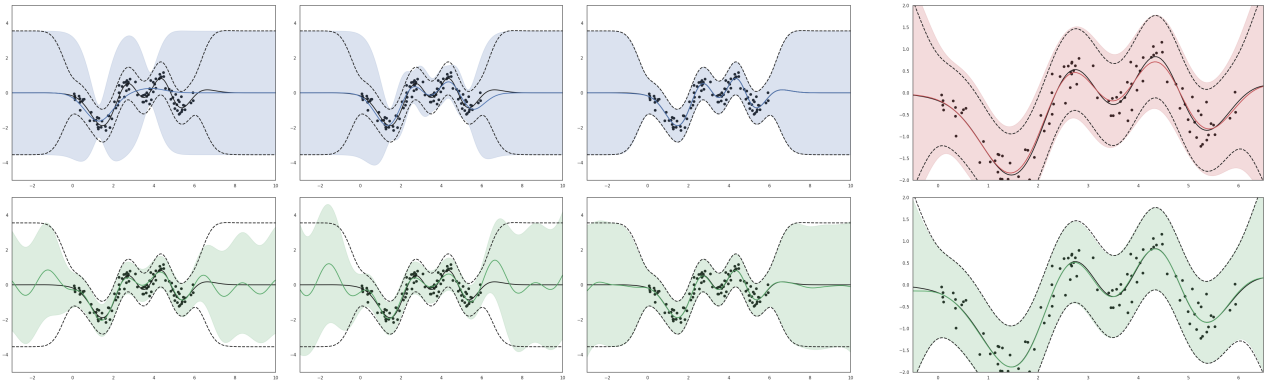
where if $\boldsymbol{\Omega} \sim \mathcal{N}(\boldsymbol{\Omega}_0, \mathbf{V})$ then it is equivalent to the GP: $f \sim \mathcal{GP}(g(\cdot; \boldsymbol{\Omega}_0), k_{\text{NTK}})$. As seen, the mean function is as flexible as a deep NN, while the kernel utilizes the gradients as features, which can easily represent different frequencies.

5. Experiments

Throughout all experiments, we use M to both refer to the number of inducing points in SVGP and the number of measurement points in GPNet and FBNN (Sun et al., 2019). Implementations are based on a customized version of GPflow (Matthews et al., 2017; Sun et al., 2018) and ZhuSuan (Shi et al., 2017). Code is available at `https://github.com/thjashin/gp-infer-net`.

5.1. Synthetic Data

We consider the inference of a GP with RBF kernel on the synthetic dataset introduced in Snelson & Ghahramani (2006). We analyze the properties of our method and compare with SVGP and FBNN (Sun et al., 2019). We fit these algorithms with minibatch size 20 on 100 data points. We ran for 40K iterations and used learning rate 0.003 for all methods. For fair comparison, for all three methods we pretrain the prior hyperparameters for 100 iterations using the GP marginal likelihood and keep them fixed thereafter. We vary M in $\{2, 5, 20\}$ for all methods. The networks used in GPNet and FBNN are the same RFE with 20 hidden units.



(a) (b)

Figure 2. Posterior process on the Snelson dataset. (a) **Top row**: SVGP with $M \in \{2, 5, 20\}$ (left to right) inducing points; **Bottom row**: GPNet with $M \in \{2, 5, 20\}$ measurement points. (b) **Top**: FBNN, $M = 20$; **Bottom**: GPNet, $M = 20$.

Table 1. Large-scale regression on the airline dataset.

METRIC	M=100			M=500		
	SVGP	GPNET	FBNN	SVGP	GPNET	FBNN
RMSE	24.261	24.055	24.257	23.698	23.675	24.431
Test LL	-4.618	-4.616	-4.611	-4.594	-4.601	-4.598

Results are plotted in Fig. 2. We can see that the performance of SVGP grows with more inducing points. When $M = 20$, both SVGP and GPNet can recover the exact GP prediction. GPNet fits the data better when $M = 2, 5$. This is because M does not constrain the capacity of the inference network, though it does affect the convergence speed, i.e., smaller M causes larger variance in the training. In Fig. 2b we take a closer look at the predictions by GPNet and FBNN near the training data. We can see that FBNN consistently overestimates the uncertainty. This effect can be well explained by their heuristic way of doing minibatch, that in each iteration they fit a different objective to match the local effect of 20 training points in a minibatch, while our stochastic mirror descent maintains a shared global objective that takes all observations into consideration.

5.2. Regression

Benchmarks We evaluate our method on seven standard regression benchmark datasets. We use RFE networks for this task. Following the settings of Salimbeni & Deisenroth (2017), we compare to the strong baselines: SVGP with 100 and 500 inducing points. We also compare to FBNN using the same inference network as in GPNet[3]. To put the comparison into a wider context, we include the results by probabilistic backpropagation (PBP) (Hernández-Lobato & Adams, 2015), which is an effective weight-space inference method for BNNs. Details of datasets and experiment settings can be found in Appendix A. Results are summarized in Fig. 3. We can see that GPNet has comparable or smaller RMSE than SVGP on most datasets, and the performance gap is often large when comparing them given $M = 100$. This demonstrates the effectiveness of inference networks than inducing points given similar computational complexity. The regression results on small datasets including Boston, Concrete and Energy show that overfitting is not observed with our powerful networks. Note that these three datasets only contain 1-2 minibatches of data, thus the performance of FBNN and GPNet are comparable because minibatch training is not an issue; while on larger datasets such as Kin8nm, Power and Wine, GPNet consistently outperforms FBNN. We find on Protein GPNet is slower to converge than other methods, and increasing the training iterations will give far better performance.

Airline Delay To demonstrate the advantage of our minibatch algorithm on large-scale datasets, we conducted experiments on the airline delay dataset, which includes 5.9 million flight records in the USA from Jan to Apr in 2018. Following the protocol in Hensman et al. (2013), we randomly take 700K points for training and 100K for testing. The results are shown in Table 1. Experiment details can be found in Appendix A. We can see that GPNet achieves best RMSE among three methods and has comparable test log likelihoods with SVGP. The RMSE gap between $M = 100$ and $M = 500$ for SVGP is larger than that of GPNet, which again demonstrates that the power of our inference network is not limited by M. Interestingly, larger M seems to cause underfitting of FBNN and leads to worse RMSE, which may also be due to the minibatch issue.

5.3. Classification

Finally, we demonstrate the flexibility of GPNet by fitting a deep convolutional inference network for a CNN-GP (Garriga-Alonso et al., 2019), whose covariance kernel

[3]Though the algorithm of Sun et al. (2019) is designed for BNNs, it also applies to other types of inference networks.

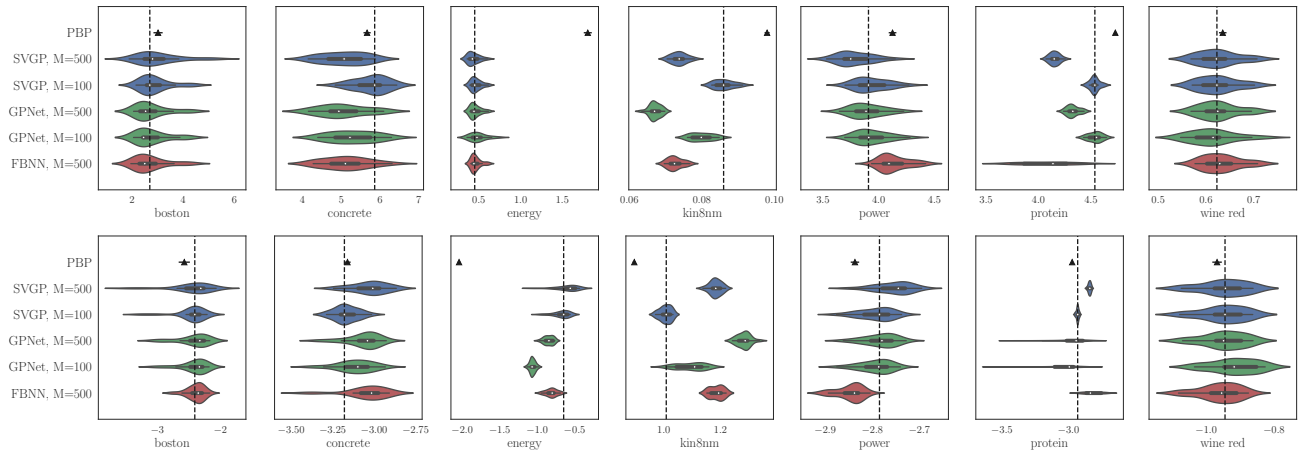


Figure 3. Regression results on benchmark datasets. **Top row**: Test RMSE; **Bottom row**: Test log likelihood. All results are drawn as violin plots which shows a kernel density of results on different splits, except for PBP we only have the mean and standard error, so an error bar is drawn instead.

Table 2. Image classification: Test error rates.

METHODS	MNIST	CIFAR10
SVGP, RBF-ARD (Krauth et al., 2016)	1.55%	-
Conv GP (van der Wilk et al., 2017)	1.22%	35.4%
SVGP, CNN-GP (Garriga-Alonso et al., 2019)	2.4%	-
GPNet, CNN-GP	1.12%	**24.63%**
NN-GP (Lee et al., 2018)	1.21%	44.34%
CNN-GP (Garriga-Alonso et al., 2019)	0.96%	-
ResNet-GP (Garriga-Alonso et al., 2019)	**0.84%**	-
CNN-GP (Novak et al., 2019)	0.88%	32.86%

are derived from an infinite-width Bayesian ConvNet (see Appendix A.3 for detailed derivation of the GP prior).

Previously, inference for GPs with such a kernel has only been investigated through exact prediction (Garriga-Alonso et al., 2019; Novak et al., 2019), where the classification problem is treated as regression so that eq. (2) applies. Though this is done for datasets like MNIST and CIFAR10 in recent works, $O(N^3)$ complexity is impractical for the method to be widely adopted. One scalable option would be sparse approximations. We tried SVGP for this prior. However, we found it difficult to stabilize the training if we update the inducing point locations. Initializing them with data or with K-means centers both result in numerical errors that prevent the method from learning. There are no other results reported using SVGP for such kernels except in Garriga-Alonso et al. (2019), where they also have to fix the inducing points (they used 1000 training data). we believe it is due to the difficulty of finding good inducing point locations in such a high-dimensional input space of images.

We test GPNet on MNIST and CIFAR10 with a CNN-GP prior. The ConvNet that defines this prior has 6 residual blocks (details in appendix A). It is natural to use the original ConvNet with trainable weight randomness as the inference network ($q(f)$) for this CNN-GP. However, as discussed in section 4, using a BNN results in intractable output distribution which requires many efforts to address. To avoid this, we use a deterministic ConvNet with an NTK on top of it defined using the fully-connected layers. This enables flexible covariance modeling while still allowing an efficient training. With the non-conjugate form of our algorithm, we are free to use a softmax likelihood, which is more suitable to classification tasks.

Results are compared to recent works in table 2. The first half of the table are approximate inference approaches with classification likelihoods, while the second half does exact prediction by GP regression. By comparing to carefully-designed inducing-point approaches such as Conv GP (van der Wilk et al., 2017), we can clearly see the advantage of our method, i.e., easily scaling up GP inference to highly-structured kernels by using flexible inference networks that match the structures, while getting superior performance than carefully-designed inducing-point methods.

6. Conclusion

We propose an algorithm to scalably train a stochastic inference network to approximate the GP posterior distribution. In the algorithm the inference network is trained by tracking a stochastic functional mirror descent update which is cheap to compute from the current approximation using a minibatch of data. Experiments show that our algorithm fixes the minibatch issue of previous works on function-space inference. Empirical comparisons to sparse variational GP methods show that our method is a more flexible alternative to GP inference.

Acknowledgements

We thank Ziyu Wang, Shengyang Sun, Ching-An Cheng for helpful discussions and Hugh Salimbeni for help with the experiments. JS was supported by a Microsoft Research Asia Fellowship. This work was supported by the National Key Research and Development Program of China (No. 2017YFA0700904), NSFC Projects (Nos. 61620106010, 61621136008, 61571261), Beijing NSF Project (No. L172037), DITD Program JCKY2017204B064, Tiangong Institute for Intelligent Computing, NVIDIA NVAIL Program, and the projects from Siemens and Intel.

References

Balan, A. K., Rathod, V., Murphy, K. P., and Welling, M. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pp. 3438–3446, 2015.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*, pp. 1533–1541, 2016.

Bernardo, J., Berger, J., Dawid, A., Smith, A., et al. Regression and classification using Gaussian process priors. *Bayesian statistics*, 6:475, 1998.

Bui, T. D., Nguyen, C., and Turner, R. E. Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*, pp. 3299–3307, 2017.

Cheng, C.-A. and Boots, B. Incremental variational sparse Gaussian process regression. In *Advances in Neural Information Processing Systems*, pp. 4410–4418, 2016.

Cheng, C.-A. and Boots, B. Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, pp. 5184–5194, 2017.

Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. Random feature expansions for deep Gaussian processes. *arXiv preprint arXiv:1610.04386*, 2016.

Dai, B., He, N., Dai, H., and Song, L. Provable Bayesian inference via particle mirror descent. In *Artificial Intelligence and Statistics*, pp. 985–994, 2016.

Damianou, A. and Lawrence, N. Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

Damianou, A. C., Titsias, M. K., and Lawrence, N. D. Variational inference for latent variables and uncertain inputs in Gaussian processes. *The Journal of Machine Learning Research*, 17(1):1425–1486, 2016.

Doucet, A., De Freitas, N., and Gordon, N. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Flam-Shepherd, D., Requeima, J., and Duvenaud, D. Mapping Gaussian process priors to Bayesian neural networks. *NIPS Bayesian Deep Learning Workshop*, 2017.

Gal, Y. and Turner, R. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning*, pp. 655–664, 2015.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1690–1699, 2018a.

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

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. Deep convolutional networks as shallow Gaussian processes. In *International Conference on Learning Representations*, 2019.

Hafner, D., Tran, D., Irpan, A., Lillicrap, T., and Davidson, J. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *arXiv preprint arXiv:1807.09289*, 2018.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

Hensman, J., Durrande, N., and Solin, A. Variational fourier features for Gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588, 2017.

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.

Jaakkola, T. and Haussler, D. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, pp. 487–493, 1999.

Jacot-Guillarmod, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, pp. 8580–8589. 2018.

Khan, M. and Lin, W. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics*, pp. 878–887, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krauth, K., Bonilla, E. V., Cutajar, K., and Filippone, M. AutoGP: Exploring the capabilities and limitations of Gaussian process models. *arXiv preprint arXiv:1610.05392*, 2016.

Lawrence, N. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research*, 6(Nov):1783–1816, 2005.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881, 2010.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*, 2018.

Ma, C., Li, Y., and Hernández-Lobato, J. M. Variational implicit processes. *arXiv preprint arXiv:1806.02390*, 2018.

MacKay, D. J. Bayesian non-linear modeling for the prediction competition. In *Maximum Entropy and Bayesian Methods*, pp. 221–234. Springer, 1996.

Mallasto, A. and Feragen, A. Learning from uncertain curves: The 2-wasserstein metric for Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 5660–5670, 2017.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Neal, R. M. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Novak, R., Xiao, L., Bahri, Y., Lee, J., Yang, G., Abolafia, D. A., Pennington, J., and Sohl-dickstein, J. Bayesian deep convolutional networks with many channels are Gaussian processes. In *International Conference on Learning Representations*, 2019.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

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. *Gaussian processes for machine learning*. MIT Press, 2006.

Salimbeni, H. and Deisenroth, M. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 4588–4599, 2017.

Shi, J., Chen, J., Zhu, J., Sun, S., Luo, Y., Gu, Y., and Zhou, Y. ZhuSuan: A library for Bayesian deep learning. *arXiv preprint arXiv:1709.05870*, 2017.

Shi, J., Sun, S., and Zhu, J. A spectral approach to gradient estimation for implicit distributions. In *International Conference on Machine Learning*, pp. 4651–4660, 2018.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2006.

Sun, S., Zhang, G., Wang, C., Zeng, W., Li, J., and Grosse, R. Differentiable compositional kernel learning for Gaussian processes. In *International Conference on Machine Learning*, pp. 4835–4844, 2018.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Sutton, R. S. and Barto, A. G. *Reinforcement learning - an introduction*. MIT Press, 1998.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 2849–2858, 2017.

Wang, Z., Ren, T., Zhu, J., and Zhang, B. Function space particle optimization for Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Williams, C. K. and Barber, D. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

Williams, C. K. and Rasmussen, C. E. Gaussian processes for regression. In *Advances in Neural Information Processing Systems*, pp. 514–520, 1996.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.