An Overview of Sparse Gaussian Processes

Harrison Zhu
Last updated on November 10, 2021

1 Introduction

Gaussian processes (GPs) are a collections of random variables or functions that have nice properties that enable them to model or emulate complicated systems. By complicated systems, these can be aerodynamics, disease transmission, climate or taxi scheduling, to name a few. Many of the systems that we would like to model are large in nature, rendering classical inference methods for GPs obsolete. One particular way to fit GP models in "big data" regimes is using sparse variational GPs, in which the underlying objective becomes a minimisation of a valid divergence measure (such as the KL-divergence). In contrary to exact or asymptotically exact inference methods (such as MCMC), these methods suffer from a lack of a principled way to assess approximation quality, although the numerous possibilities that these approximations, due to the computational tractability, have been able to unlock in the realm of GP modelling should not be understated.

The focus of these notes is on sparse GPs and so we will omit many interesting aspects of GPs. However, if you are interested in GPs, I have included a list of interesting references here: [Wilkinson et al., 2021, Kanagawa et al., 2018, Salimbeni et al., 2018, Dutordoir et al., 2018, De G. Matthews et al., 2017, Hensman et al., 2015a,b, Alvarez et al., 2012, Duvenaud et al., 2011, Rasmussen and Williams, 2006].

Most of the contents from this document is taken from [Leibfried et al., 2021, van der Wilk et al., 2020, Dutordoir et al., 2018, Salimbeni et al., 2018, De G. Matthews et al., 2017, Titsias, 2009, Hensman et al., 2015b,a, 2013].

2 Background

In this section, we will give a rigorous introduction to GPs. $\mathcal{X} \subseteq \mathbb{R}^d$ will denote a feature space with an associated probability distribution Π that admits a density function π. Let $(\Omega, \mathcal{F}, \mathbb{P})$ denote a (abstract) probability space with a distribution $\mathbb{P}$, which we will use to measure GP samples that live in the space Ω. $\mathcal{N}(a, b)$ and $N(\cdot; a, b)$ will denote a Gaussian distribution (possibly multivariate) and density function respectively, with mean a and covariance b. A stochastic process is $g : \mathcal{X} \times \Omega \to \mathbb{R}$, where for all $x \in \mathcal{X}$, $g(x, \cdot)$ is a random variable, and for all $\omega \in \Omega$, $g(\cdot, \omega)$ is a function. We usually omit the second argument for conciseness. See Figure 1 for a graphical illustration.

With a suitable $\mathbb{P}$, $g(\cdot, \omega)$ can help flexibly approximate or learn functions $f : \mathcal{X} \to \mathbb{R}$.

The probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where the randomness can be "measured". Let $\mathbb{E}_{\mathbb{P}}[g]$ denote the expectation with respect to the distribution $\mathbb{P}$:

$$\mathbb{E}_{\mathbb{P}}[g] := \int_{\Omega} g(\cdot, \omega) \mathrm{d}\mathbb{P}(\omega).$$

In general, given any distribution P, we will integrate with respect to where P is defined. For instance, $\mathbb{E}_{\Pi}$ will be equivalent to

$$\mathbb{E}_{\Pi}[g] := \int_{\mathcal{X}} g(x, \cdot) \mathrm{d}\Pi(x).$$

We can also construct a credible interval/set $[t_{\delta/2}(x), t_{1-\delta/2}(x)] \subseteq \mathbb{R}$ such that

$$\mathbb{P}(\omega \in A : g(x, \omega) \in [t_{\delta/2}(x), t_{1-\delta/2}(x)]) = 1 - \delta$$

for each $x \in \mathcal{X}$ and $\delta \in [0, 1]$.

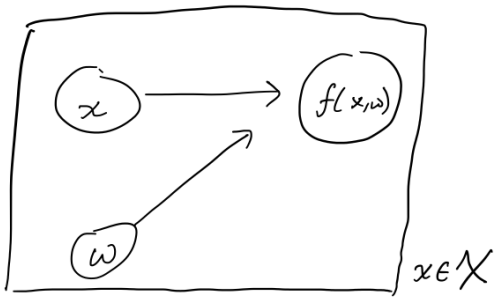

Figure 1: Graphical model of stochastic processes.

2.1 Gaussian Processes

There are many ways of defining GPs that are all equally valid and can be suitable for different applications. We will define GPs using a stochastic process viewpoint as it enables more careful treatments of randomness later on.

Gaussian process regression (GPR): We first look at GP regression (GPR), which is the "hello world" equivalent of GPs. A GP f is a stochastic process, denoted by the short-hand $f \sim \mathcal{GP}(\mu, k)$, with mean function μ and covariance/kernel function k such that

1. $\mu(x) := \mathbb{E}_{\mathbb{P}}[g(x, \cdot)] = \int_\Omega g(x, \omega) \mathrm{d}\mathbb{P}(\omega)$ for all $x \in \mathcal{X}$,
2. $k(x, x') := \mathrm{Cov}_{\mathbb{P}}(g(x, \cdot), g(x', \cdot)) = \mathbb{E}_{\mathbb{P}}[(g(x, \omega) - \mu(x))(g(x', \omega) - \mu(x'))]$ for all $x, x' \in \mathcal{X}$.
3. Given a finite subset $X := (x_1, \ldots, x_n)^\intercal \in \mathbb{R}^{n \times d}$, $\mathbb{P}$ is such that $(g(x_1), \ldots, g(x_n))^\intercal \sim \mathcal{N}(\mu_X, k_{XX})$, where $\mu_X \equiv \mu(X) := (\mu(x_1), \ldots, \mu(x_n))^\intercal$ and $k_{XX} = k(X, X) \in \mathbb{R}^{n \times n}$ such that $(k_{XX})_{ij} = k(x_i, x_j)$. We will also use the shorthand $k_{x,X} := k_{X,x}^\intercal \in \mathbb{R}^{n \times 1}$ with $(k_{X,x})_i = k(x_i, x)$.

See Figure 2 for a graphical illustration of GPs as a collection of random variables.

Intuitively, the kernel measures the similarity between points x and x', and so following this intuition one could design suitable kernels for different applications. For instance, $k(x, x') = c^2 x x'$ for $c \in \mathbb{R}$ and $\mu(x) = 0$ will give a GP that is equivalent to the function $f(x) = dx$ for $d \in \mathbb{R}$, which are linear functions and is equivalent to performing Bayesian linear regression. Another commonly used kernel is the RBF kernel $k(x, x') = \sigma_k^2 \exp\left(-\frac{(x-x')^2}{2\ell_k^2}\right)$ for some parameters $\sigma_k, \ell_k > 0$. See the kernel cookbook[1] for more details on kernels.

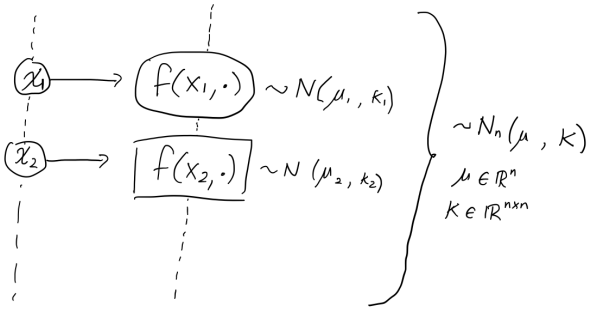

Figure 2: Illustration of Gaussian processes as a collection of random variables.

[1] https://www.cs.toronto.edu/ duvenaud/cookbook/

Noise-Free Regression: We now discuss how Gaussian processes can be updated using the Bayesian update rule when we observe data.

Given finite observations $X = (x_1, \ldots, x_n)^\intercal$ and labels $y := (f(x_1), \ldots, f(x_n)) = (y_1, \ldots, y_n)^\intercal \in \mathbb{R}^n$, f conditioned on this gives us a posterior Gaussian process with posterior distribution over the samples $\tilde{\mathbb{P}}$ such that $f|X, y \sim \mathcal{GP}(\tilde{\mu}, \tilde{k})$ with

$$\begin{aligned}\tilde{\mu}(x) :=& \mu(x) + k_{xX} k_{XX}^{-1}(y - \mu_X),\\ \tilde{k}(x, x') :=& k(x, x') - k_{xX} k_{XX}^{-1} k_{Xx'},\end{aligned}$$

for any $x, x' \in \mathcal{X}$. The GP distribution over the functions are updated so that the prior mean and kernel are perturbed by correction terms that depend on the observed data. Intuitively, in areas of $\mathcal{X}$ where the observation data lives, k_{xX} tell that us x is "similar" to the observations and so pushes $\tilde{\mu}$ to be similar to y. A similar intuition exists for $\tilde{k}$.

Noisy Regression: In reality, y is observed noisily, such as from the additive noise process $y_i = f(x_i) + \epsilon_i$. Suppose $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ (which is rarely sufficient for real world data) for which $\sigma > 0$ and the noise is independently distributed, then we can write down the likelihood:

$$p(y_i|f, x_i) = N(y_i; f(x_i), \sigma^2)$$

if we take a sample of f i.e. take $\omega \in \Omega$ and obtain a function $f(\cdot, \omega)$. Since the labels are noisy, we observe them as $y_i = f(x_i) + \epsilon_i$, where ϵ_i is a sample from $\mathcal{N}(0, \sigma^2)$.

The posterior update here yields

$$\begin{aligned}\tilde{\mu}(x) :=& \mu(x) + k_{xX}(k_{XX}^{-1} + \sigma^2 I_n)(y - \mu_X),\\ \tilde{k}(x, x') :=& k(x, x') - k_{xX}(k_{XX}^{-1} + \sigma^2 I_n) k_{Xx'},\end{aligned}$$

for any $x, x' \in \mathcal{X}$. From a numerical analysis point of view, $\sigma^2 I_n$ can be thought of an a jitter term to ensure numerical stability when inverting k_{XX}. This gives $p(f(x)|X, y)$, but to link up the posterior GP to the noisy observations y we must consider posterior predictive distribution

$$\begin{aligned}p(y_*|X, y) =& \textstyle\int_\Omega p(y_*|f, x_*) p(f(x_*, \omega)|X, y) \mathrm{d}\mathbb{P}(\omega),\\ =& \textstyle\int_\Omega N(y_* - f(x_*, \omega); 0, \sigma^2) N(f(x_*, \omega); \tilde{\mu}(x_*), \tilde{k}(x_*, x_*)) \mathrm{d}\mathbb{P}(\omega),\\ =& N(y_*; \tilde{\mu}(x_*), \tilde{k}(x_*, x_*) + \sigma^2),\end{aligned}$$

using the formula for the convolution between 2 Gaussians. Intuitively, we average f over all possible samples Ω to get the posterior predictive. The randomness for the predictive now only comes from the noise variable ϵ.

Heterogeneous Likelihoods: GPs can also be plugged into existing generalised linear models (GLMs) by replacing the linear component with the GP. An example is for classification, we can model such responses using a Bernoulli likelihood $\text{Bernoulli}(p(f(x)))$ with the probability being modelled with $p(f(x)) = \text{logit}^{-1}(f(x))$, where $\text{logit}(x) := \log(x/(1-x))$. You may be more familiar with the equivalent mean function, or $\text{softmax}(x) := \text{logit}^{-1}(x) = 1/(1 + \exp(-x))$. The general idea is that $f(x)$ will be mapped onto the support of the distribution of the response via the inverse link function.

It should be noted that no analytic posterior exists when using non-Gaussian likelihoods for the response, in which case we would need to rely on posterior approximations such as the Laplace approximation, expectation propagation (EP), Markov chain Monte Carlo (MCMC) or variational inference (VI).

2.2 Classical Inference

There are many approximation methods for GPs. Here, we only name a few popular ones.

Maximum Likelihood: Recall that the hyperparameters of a GP observation model are the GP hyperparameters (from the mean and kernel functions) and the model hyperparameters (e.g. standard deviation). It is therefore possible to simply write down the unnormalised log-posterior and perform a non-convex optimisation over these hyperparameters. However, the limitations for these methods are that (i) the non-convex problem can be difficult to solve; (ii) the assumption that the true hyperparameters are fixed and (iii) can be expensive due to $\mathcal{O}(n^3)$ complexity without minibatches.

Markov Chain Monte Carlo: MCMC methods are often the gold standard for evaluating posterior sample quality for many algorithms. Given a target distribution, it can be theoretically shown that the samples from MCMC can asymptotically converge ($n \to \infty$) to the target distribution under mild condition. Convergence rates can also be derived by studying the geometric ergodicity of a given problem. However, the limitations of MCMC are (i) Whilst asymptotically exact with $\mathcal{O}(nd)$ complexity to the number of data points n and dimensionality d, the mixing time may increase as d increases; (ii) it is difficult to parallelise due to the single-threaded, sequential nature of the algorithm; (iii) in practice, Hamiltonian Monte Carlo (HMC) is often the workhorse, and it is often required to have a good initial distribution and fine-tuned hyperparameters to achieve good results.

3 Sparse Gaussian Processes

Sparse Gaussian processes is a class of GP posteriors that are approximated by minimising a divergence measure, D, with the posterior distribution of the GP. Here, we discuss the main classes, sparse Gaussian process regression (SGPR; [Titsias, 2009]), sparse variational Gaussian processes (SVGP; [Hensman et al., 2013]) and sparse variational Gaussian processes using MCMC (SGPMC; [Hensman et al., 2015b]). In this discussion, we will only consider $D = \text{KL}$, the Kullback-Leibler divergence. It is also possible to perform inference over GPs using other techniques such as conjugate gradient (in `gpytorch`).

Variational Inference (VI): Given a target distribution p, VI seeks to construct an approximate distribution q_* such that

$$q_* := \text{argmin}_{q \in \mathcal{Q}} \text{KL}(q||p) = \text{argmin}_{q \in \mathcal{Q}} \int_{\mathcal{X}} \log \frac{q(x)}{p(x)} q(x) \text{d}x,$$

where $\mathcal{Q}$ is a family of variational distributions that is user-defined. Note that $\mathcal{Q}$ may not necessarily contain p, the target distribution, but will find the closest q_* to p within the defined region $\mathcal{Q}$. See Figure 3 for an illustration.

The KL-divergence has the property that (1) it is always positive and (2) $\text{KL}(q||p) = 0 \Leftrightarrow q = p$. As to how it is minimised, most methods are gradient-based and utilises modern software architecture for gradient-based optimisation (such as L-BFGS or Adam). One caveat is that a low KL-divergence may not necessarily guarantee good posterior approximations (see counterexamples in [Huggins et al., 2020]).

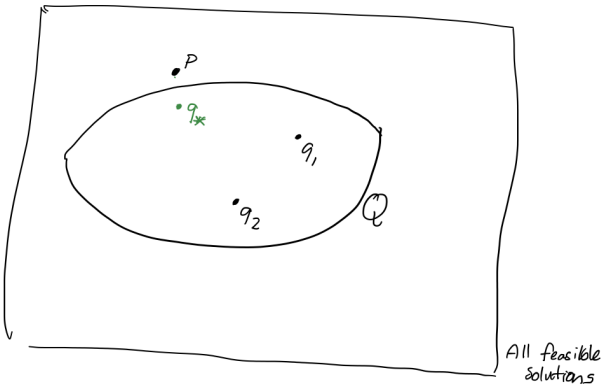

Figure 3: Illustration of the variational approximation from $\mathcal{Q}$.

3.1 Sparse Gaussian Processes

We will denote n as the number of training points, $n_b << n$ the minibatch size and m the number of inducing points. For now, the inducing points are a set of function values $u = (u_1, \ldots, u_m)^\intercal$ such that $u_i = f(z_i)$ for landmark points $z_i \in \mathcal{X}$ for $i = 1, \ldots, m$. Therefore to define inducing points, we simply need to pick landmark points $z = (z_1, \ldots, z_m)^\intercal \subset \mathcal{X}$ appropriately, such as using K-Means cluster centres [Oglic and Gärtner, 2017]. Sparse GPs approximate the posterior $p(f, u|y)$

using a variational approximation $q(f,u) := p(f|u)q(u)$, where only $q(u)$ needs to be specified and optimised (other than the other model hyperparameters). We construct $q(u)$ such that

$$\begin{aligned} q(u) &= \text{argmin}_{q\in\mathcal{Q}}\text{KL}(p(f,u|y)||q(f,u)) \\ &= \text{argmin}_{q\in\mathcal{Q}}\text{KL}(q(u)||p(u)) - \textstyle\sum_{i=1}^{n}\mathbb{E}_{p(f_i|u)q(u)}[\log p(y_i|f_i)], \\ &=: \text{argmin}_{q\in\mathcal{Q}} - \text{ELBO}(q) \\ &= \text{argmax}_{q\in\mathcal{Q}}\text{ELBO}(q) \end{aligned}$$

in which $f_i := f(x_i)$. Intuitively, all the information about the stochastic process is being compressed into u and we simply need to learn the best $q(u)$ such that the KL divergence is smallest to get a close approximation $q(f,u)$ to the true posterior.

We assume that we are in the iid Gaussian noise regression case. All of the upcoming approximations are efficiently implemented in `gpflow` [De G. Matthews et al., 2017].

SGPR: This algorithm has per-iteration complexity $\mathcal{O}(nm^2 + m^2)$. Assuming that $q(u) = N(m_u, S_u)$, the optimal posterior could be computed analytically [Titsias, 2009], giving

$$\text{ELBO}(q) = \log N(y; 0, Q_{ff} + \sigma^2 I) - \tfrac{1}{2\sigma^2}\text{Tr}(k_{XX} - Q_{ff}),$$

where $Q_{ff} := k_{XZ}k_{ZZ}^{-1}k_{ZX}$. The optimal $q(u)$ is exactly given by

$$\begin{aligned} q(u) =& N(m_u, S_u), \\ S_u =& k_{ZZ}^{-1} + k_{ZZ}^{-1}k_{ZX}k_{XZ}k_{ZZ}^{-1}\sigma^2, \\ m_u =& \sigma^2 S_u^{-1}k_{ZZ}^{-1}k_{ZX}y. \end{aligned}$$

Therefore it only remains to optimise over the kernel hyperparameters and σ^2 using gradient-based optimisation. To perform prediction at a new point $x \in \mathcal{X}$, we can obtain a posterior distribution

$$\begin{aligned} q(f(x)) :=& \textstyle\int p(f(x)|u)q(u)\mathrm{d}u = N(f(x); \tilde{\mu}(x), \tilde{v}(x)), \\ \tilde{\mu}(x) =& k_{xZ}k_{ZZ}^{-1}m_u, \\ \tilde{v}(x) =& k_{xx} - k_{xZ}(k_{ZZ}^{-1} + k_{ZZ}^{-1}S_u k_{ZZ}^{-1})k_{xZ}, \end{aligned}$$

and then subsequently the posterior predictive distribution

$$\begin{aligned} p(y|x) &\approx \textstyle\int p(y|f(x))q(f(x))\mathrm{d}f(x) \\ &\equiv \textstyle\int_\Omega p(y|f(x,\omega)))\mathrm{d}\tilde{\mathbb{P}}(\omega) \\ &= N(y; \tilde{\mu}(x), \tilde{v}(x) + \sigma^2), \end{aligned}$$

where $\tilde{\mathbb{P}}$ represents the approximate posterior distribution over the GP samples. Although SGPR may save you trouble from optimising the variational approximation, SGPR is only possible for iid Gaussian noise regression problems and may not be computationally feasible when n (e.g. label-rich datasets) or m (e.g. spatiotemporal datasets where the number of inducing points explode) are very large.

SVGP: This algorithm has per-iteration complexity $\mathcal{O}(n_b m^2 + m^2)$. To overcome the difficulty that n can be too large, [Hensman et al., 2015a] proposes a minibatch training procedure, in which we only take a minibatch n_b of the training points and do not use the optimal posterior $q(u)$. We estimate the ELBO with the minibatch estimate

$$\begin{aligned} &\textstyle\sum_{i=1}^{n}\mathbb{E}_{p(f_i|u)q(u)}[\log p(y_i|f_i)] - \text{KL}(q(u)||p(u)) \\ &= \mathbb{E}_B[\tfrac{n}{n_b}\textstyle\sum_{B\in B}\mathbb{E}_{p(f_i|u)q(u)}[\log p(y_i|f_i)]] - \text{KL}(q(u)||p(u)), \\ &\approx \tfrac{1}{L}\textstyle\sum_{b=1}^{L}\tfrac{n}{n_b}\sum_{i\in B_b}\mathbb{E}_{p(f_i|u)q(u)}[\log p(y_i|f_i)] - \text{KL}(q(u)||p(u)), \end{aligned}$$

where $|B| = |B_b| = n_b << n$ with minibatches B or B_b. In practice, we will only take $L = 1$. Note that both terms, if we are in the iid Gaussian noise case, are analytically tractable. Otherwise, in the non-Gaussian likelihood case, it is also possible to compute the first term using quadrature or Monte Carlo integration.

Differing from SGPR, we now have a lower per-iteration complexity and also need to learn the variational parameters m_u and S_u. This could be done simply by using standard gradient descent (e.g. Adam), alongside the other hyperparameters, but it could be shown that using natural gradient descent could yield better solutions in practice [Salimbeni et al., 2018].

However..

SGPMC: This algorithm has per-iteration complexity $\mathcal{O}(nm^2 + m^2)$. Setting $\mathcal{Q}$ to the multivariate Gaussian family may be too restrictive. In addition, one may also want to learn the hyperparameters as non-point estimates. [Hensman et al., 2015b] proposes to directly sample $q(u, \theta)$, where θ are the hyperparameters, via MCMC. It can be directly shown that the optimal posterior distribution is of the form

$$\log q(u, \theta) = \mathbb{E}_{p(f|u,\theta)}[\log p(y|f)] + \log p(u|\theta) + \log p(\theta) - \log C,$$

for some constants C. Using some clever tricks to compute $\log q(u, \theta)$ and initialise the MCMC sampler, one can directly obtain samples of (u, θ) using MCMC. The only "issue" is that we lose the tractability of the posterior GP.

3.2 VI Techniques

Interdomain Inducing Points: As discussed earlier, the easiest way to initialise them is to use K-Means clustering and set $u = f(z)$. Recall that given $q(u) = N(m_u, S_u)$, the posterior distribution is

$$\begin{aligned} q(f(x)) :=& \int p(f(x)|u)q(u)\mathrm{d}u = N(f(x); \tilde{\mu}(x), \tilde{v}(x)), \\ \tilde{\mu}(x) =& k_{xZ}k_{ZZ}^{-1}m_u, \\ \tilde{v}(x) =& k_{xx} - k_{xZ}(k_{ZZ}^{-1} + k_{ZZ}^{-1}S_u k_{ZZ}^{-1})k_{xZ}. \end{aligned}$$

As argued in [van der Wilk et al., 2020], the function observations $u = f(z)$ are not the only pieces of information we know about f. We could define a linear operator $\mathcal{L}$ such that $u = \mathcal{L}f(\cdot)$. For example, $\mathcal{L}f(\cdot) = f(z)$ would give the usual inducing points, and $\mathcal{L}f(\cdot) = \frac{\partial}{\partial x_d} f(z)$ would describe the inducing points using the derivative on the dth dimension of z. Another popular approach is to define $\mathcal{L}$ as an integral operator [Leibfried et al., 2021]

$$\mathcal{L}f(\cdot) = \int_{\mathcal{X}} f(x)\phi(x)\mathrm{d}x, (\mathrm{d}\Pi(x)?)$$

where $\phi(x)$ are "inducing features". To construct these inter-domain inducing points, define $\phi_1, \ldots, \phi_m$ inducing features and obtain $u_i = \int_{\mathcal{X}} f(x)\phi_m(x)\mathrm{d}x$.

Since f is random with sample distribution $\mathbb{P}$, we have the prior $u \sim \mathcal{N}(\mu_u, k_{uu})$ with

$$\begin{aligned} (\mu_u)_i &= \mathbb{E}_{\mathbb{P}}[u] = \int_{\mathcal{X}} \mathbb{E}_{\mathbb{P}}[f(x)]\phi_i(x)\mathrm{d}x = \int_{\mathcal{X}} \mu(x)\phi_i(x)\mathrm{d}x, \\ k(u_i, u_j) &= \mathbb{E}_{\mathbb{P}}[(u_i - (\mu_u)_i)(u_j - (\mu_u)_j)] = \int_{\mathcal{X}} \int_{\mathcal{X}} k(x, x')\phi_i(x)\phi_j(x')\mathrm{d}x\mathrm{d}x'. \end{aligned}$$

In addition, given x or $f(x)$, we have

$$k(f(x), u_i) = \mathbb{E}_{\mathbb{P}}[(f(x) - \mu(x))(u_j - (\mu_u)_j)] = \int k(x, x')\phi_i(x')\mathrm{d}x'.$$

Given these 3 terms, we can now fully specify the posterior distribution $q(f(x))$ by replacing Z with u for the relevant kernels.

To sum up, instead of picking the landmark points z in order to define inducing points, we now have a more general formulation of inducing points via the integral operator and inducing features ϕ_i. What remains is the choice of ϕ_i. As discussed in [Leibfried et al., 2021], there are many interesting choices of ϕ_i. If $\phi_i(x) = \delta_{z_i}$, where z_i is a landmark point, then this formulation reduces to the standard inducing points. A well-studied type of features is Fourier features: $\phi_i(x) = \exp(-\mathbf{i}\omega_i^{\mathsf{T}} x)$, where $\mathbf{i}$ is the imaginary number and ω_i is an inducing frequency vector, which could be chosen on a grid (in the frequency domain). See [Leibfried et al., 2021, Hensman et al., 2017] for further details.

Natural Gradients: As mentioned earlier, it may be more "natural" to learn $q(u)$ using natural gradients [Salimbeni et al., 2018] rather than standard gradient descent over m_u and S_u. One can also perform optimisation using the geometry of $\mathcal{Q}$:

$$\eta_t^{k+1} \leftarrow \eta_t^k + \rho_k \tilde{\nabla}_\xi \text{ELBO}(\eta_t^k, \theta_t),$$

where $\tilde{\nabla}_\xi := F(\xi)^{-1}\nabla_\xi$, where $F(\xi)$ is the Fisher information matrix of $q(u;\xi)$. One can identify a statistical manifold (Riemannian) with metric tensor being $F(\xi)$ and it can be shown that $\tilde{\nabla}_\xi \text{ELBO}(\xi, \theta)$ is the Riemannian gradient if we pose ξ as lying on the statistical manifold and perform Riemannian gradient descent.

4 Frontiers of Research

4.1 Inference Techniques

4.2 Interdomain Inducing Points

4.3 Deep Gaussian Processes

4.4 Latent Variable Gaussian Processes (LVGP)

4.5 State Space Models

4.6 Software

I would say that `gpflow` is the best package if you would like to use sparse variational GPs for research. `gpflux` is also a nice package built on top of `gpflow` that one could use to implement deep GPs. If you're interested in state-space GPs, then `markovflow` is also a package under development that is built on top of `gpflow`.

References

Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for Vector-Valued Functions: a Review. *arXiv:1106.6251 [cs, math, stat]*, April 2012. URL `http://arxiv.org/abs/1106.6251`. arXiv: 1106.6251.

Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017. Publisher: JMLR. org.

Vincent Dutordoir, Hugh Salimbeni, Marc Deisenroth, and James Hensman. Gaussian Process Conditional Density Estimation. *arXiv:1810.12750 [cs, stat]*, October 2018. URL `http://arxiv.org/abs/1810.12750`. arXiv: 1810.12750.

David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive gaussian processes. In *Advances in neural information processing systems*, pages 226–234, 2011.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian Processes for Big Data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 282–290, Arlington, Virginia, USA, 2013. AUAI Press. event-place: Bellevue, WA.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015a.

James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for Variationally Sparse Gaussian Processes. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015b. URL `https://proceedings.neurips.cc/paper/2015/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf`.

James Hensman, Nicolas Durrande, Arno Solin, and others. Variational Fourier Features for Gaussian Processes. *J. Mach. Learn. Res.*, 18(1):5537–5588, 2017.

Jonathan H. Huggins, Mikołaj Kasprzak, Trevor Campbell, and Tamara Broderick. Validated Variational Inference via Practical Posterior Error Bounds. *arXiv:1910.04102 [cs, math, stat]*, February 2020. URL `http://arxiv.org/abs/1910.04102`. arXiv: 1910.04102.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K. Sriperumbudur. Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences. *arXiv:1807.02582 [cs, stat]*, July 2018. URL http://arxiv.org/abs/1807.02582. arXiv: 1807.02582.

Felix Leibfried, Vincent Dutordoir, S. T. John, and Nicolas Durrande. A Tutorial on Sparse Gaussian Processes and Variational Inference. *arXiv:2012.13962 [cs, stat]*, July 2021. URL http://arxiv.org/abs/2012.13962. arXiv: 2012.13962.

Dino Oglic and Thomas Gärtner. Nyström Method with Kernel K-means++ Samples as Landmarks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2652–2660. PMLR, August 2017. URL http://proceedings.mlr.press/v70/oglic17a.html.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9. OCLC: ocm61285753.

Hugh Salimbeni, Stefanos Eleftheriadis, and James Hensman. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pages 689–697. PMLR, 2018.

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

Mark van der Wilk, Vincent Dutordoir, S. T. John, Artem Artemev, Vincent Adam, and James Hensman. A Framework for Interdomain and Multioutput Gaussian Processes. *arXiv:2003.01115 [cs, stat]*, March 2020. URL http://arxiv.org/abs/2003.01115. arXiv: 2003.01115.

William J. Wilkinson, Arno Solin, and Vincent Adam. Sparse Algorithms for Markovian Gaussian Processes. *arXiv:2103.10710 [cs, stat]*, June 2021. URL http://arxiv.org/abs/2103.10710. arXiv: 2103.10710.