Developments in Gaussian processes with applications to climate science and network problems.

Thomas Pinder, BSc (Hons), MSc

School of Mathematics and Statistics

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

March, 2023

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words including appendices and footnotes, but excluding the bibliography. The word count of this thesis is 35562 .

Thomas Pinder

Developments in Gaussian processes with applications to climate science and network problems.

Thomas Pinder, BSc (Hons), MSc.

School of Mathematics and Statistics, Lancaster University

A thesis submitted for the degree of *Doctor of Philosophy.* March, 2023.

Abstract

The ability to efficiently model complex datasets using probabilistic models is a key component of many machine learning workflows as it offers the ability to extract accurate predictions and well-characterised uncertainty estimates. Consequently, it becomes possible to develop models that can be deployed as decision-making tools. However, evaluating such models is often computationally expensive, particularly when assumptions of independence and identically distributed data can no longer be made. This thesis explores how Gaussian process models can be used to model climate data, and how the kernel function of a Gaussian process can be adapted to operate on data observed on a network. Methodological developments are proposed to enable faster inference for Gaussian processes, to use Gaussian processes as tools for embedding hypergraphs, and to extract latent functions from vector-valued datasets. In application, this thesis explores the effect of Covid-19 on air pollution in the United Kingdom, how air pollution varies at a street-level, the latent structure of political networks, and what future warmings can be expected on planet Earth. The consideration of how such models can be computationally developed is carefully considered throughout, with a specific chapter dedicated to the development of a new Gaussian process software package that allows for new computational methods to be developed and tested.

Publications

The work contained in this thesis is based on the following articles:

T. Pinder, C. Nemeth, and D. Leslie. Stein variational Gaussian processes. *arXiv preprint arXiv:2009.12141*, 2020

T. Pinder, K. Turnbull, C. Nemeth, and D. Leslie. Gaussian processes on hypergraphs. *arXiv preprint arXiv:2106.01982*, 2021

T. Pinder and D. Dodd. GPJax: a Gaussian process framework in jax. *Journal of Open Source Software*, 7(75):4455, 2022. DOI: 10.21105/JOSS.04455. URL: HTTPS://DOI.ORG/10.21105/JOSS.04455

T. Pinder, K. Turnbull, C. Nemeth, and D. Leslie. Street-level air pollution modelling with graph Gaussian processes. *ICLR: AI for Earth and Space Science*, Feb. 2022

T. Pinder, M. Amos, D. Booker, R. Duncan, L. Gouldsbrough, and P. Young. LancasterAQ: a mobile air quality dataset measuring ultrafine levels in Lancaster. *Under review*, Sept. 2022

T. Pinder, M. Amos, and P. Young. Identifying latent climate signals using sparse hierarchical Gaussian processes. *NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*, Sept. 2022

Acknowledgements

My acknowledgements must begin with my supervisors: David Leslie, Christopher Nemeth, and Paul Young. Your excellent guidance and support have allowed my PhD to evolve into a process which I have enjoyed endlessly. I am particularly thankful to David and Chris for taking a chance on me when times were tough. You always encouraged me to be curious and explore ideas that excite me; a skill which I will forever appreciate. I am also grateful to Theodoros Damoulas and Christopher Sherlock for examining and improving my thesis. Any errors that remain are, of course, my own.

I would like to thank Alessandra Tosi for effortlessly helping me navigate the world of machine learning in industry, and course-correcting me whenever I was going down the wrong path. To James Hensman I owe a huge debt of gratitude for his enthusiasm and unrelenting kindness. I think it is unlikely that I will ever meet someone so capable of bringing science to life and instilling a sense of excitement in even the most technical topics.

Lancaster has been a wonderful place to complete a PhD. The tight-knit statistics and machine learning community here has been an invaluable sounding board for ideas, and many of the people I have met have become close friends: Matt Amos, for your patience and empathy answering all of my witless questions on climate science, Daniel Dodd for helping me feel less alone whilst navigating the world of open-source software development, and Mike O'Malley for always indulging in my problems and questions, provided there was a regular supply of beer. It would be remiss of me not to acknowledge Alex Ahern, Callum Barltrop, Jez Carter, Dan Clarkson, Diarmuid Corr, Rachael Duncan, Henry Moss, Martin Paley, Pola Schwöbel, Harry Spearing, Kathryn Turnbull, and everyone else for whom I can say nothing but thank you for your friendship and support.

And back to where it all began; my family. Mum, Dad, and Sam — the loving environment that you provided me with growing up has allowed me to grow into the person I am today. From taxiing me from rugby training, to athletics practice, and back to rugby, your generosity knew no bounds. You pushed when I needed encouragement, and you were there to catch me when I fell. You told me to never get onto the treadmill of life unless I was ready and to always follow my heart. I would like to also acknowledge my Arra. From showing me how to build computers to teaching me to weld a bird feeder, your love for science has always been an inspiration to me. To Rosemary, Eryl, Rhian-Mai, Tasha, Heulwen, Duane, and Ceri, thank you for welcoming me into your family and allowing me to experience the infectious energy that you radiate. Spending time with you all has often been the perfect distraction from the PhD.

Finally, I would like to acknowledge my love and partner, Ffion. The effortless way in which you exude benevolence, empathy, and compassion is inspirational. You have provided endless enjoyment and laughter outside of the PhD, and you never let me waste a nice weekend inside. You are my greatest support, and none of this work would have been possible without you.

Contents

List of Figures

List of Tables

Glossary

ARD Automatic Relevance Determination. 56

BO Bayesian optimisation. 31

CMIP6 Coupled Model Intercomparison Project Phase 6. 148, 151

CO_2 carbon dioxide. 145, 146

DPP Determinantal point process. 71

DTC Deterministic Training Conditional. 28–30

ECE expected calibration error. 122

ELBO evidence lower bound. 6, 7, 9, 32, 34, 37, 38, 86, 112, 118, 119, 122, 137, 138, 154, 156, 157

FITC Fully Independent Training Conditional. 29, 30, 35

GP Gaussian process. 1, 3, 4, 11, 12, 16–19, 21–23, 25–28, 31, 33, 38–41, 43–46, 55–58, 62–66, 68–72, 74, 75, 77, 80, 84, 85, 89, 90, 99, 102, 104, 105, 110, 111, 113, 115–117, 121, 122, 129–136, 139, 141, 142, 148, 153–155, 162, 167–170, 173–179

GPLVM Gaussian process latent variable model. 36, 39, 40, 102, 108, 110–112, 116–118, 123, 124, 127, 128

GPS global positioning system. 94

HMC Hamiltonian Monte-Carlo. 66–69, 72, 74

IPCC Intergovernmental Panel on Climate Change. 165, 170

KLD Kullback-Leibler divergence. 5–7, 16, 38, 46, 51–54, 119, 157

KSD kernel Stein discrepancy. 51, 53, 158–161, 166

MAP maximum a posteriori. 118

MCMC Markov chain Monte-Carlo. 3, 4, 24, 39, 44–46, 56, 70, 130, 139

MMM multi-model mean. 165, 166

MP Members of Parliament. 103, 104

NO_2 nitrogen dioxide. 70, 71, 77–80, 83–85, 87, 89, 90, 176

PCA principal component analysis. 108, 110, 112, 123

PLV projected latent variables. 28

PM particulate matter. 91

PSD positive semi-definite. 19, 20

RBF radial basis function. 20, 22, 53, 68

RCP representative concentration pathway. 145, 146, 163

RKHS reproducing kernel Hilbert space. 46–48, 51, 52

RMSE root-mean-square error. 68, 69, 71

SOR Subset of Regressors. 29

SPDE stochastic partial differential equation. 104, 115

SPGP Sparse Pseudo-input GP. 29

SSP shared socio-economic pathway. 144–146, 151, 163–166

SVGD Stein variational gradient descent. 3, 4, 24, 39, 40, 43–46, 49, 51, 52, 54–58, 63, 64, 66, 72, 74, 75, 175

UFP ultrafine particle. 78, 91–95, 97

UK United Kingdom. 23, 70, 78, 79, 91, 93, 103, 175

USA United States of America. 103

VAE variational autoencoder. 55

VFE variational free energy. 28, 31, 35–39, 43

VI variational inference. 3, 5, 6, 8, 10, 11, 31, 39, 44–46, 51, 52, 54–56, 65–69, 71–74, 86, 154

Glossary

ARD Automatic Relevance Determination. 56

BO Bayesian optimisation. 31

CMIP6 Coupled Model Intercomparison Project Phase 6. 148, 151

CO_2 carbon dioxide. 145, 146

DPP Determinantal point process. 71

DTC Deterministic Training Conditional. 28–30

ECE expected calibration error. 122

ELBO evidence lower bound. 6, 7, 9, 32, 34, 37, 38, 86, 112, 118, 119, 122, 137, 138, 154, 156, 157

FITC Fully Independent Training Conditional. 29, 30, 35

GP Gaussian process. 1, 3, 4, 11, 12, 16–19, 21–23, 25–28, 31, 33, 38–41, 43–46, 55–58, 62–66, 68–72, 74, 75, 77, 80, 84, 85, 89, 90, 99, 102, 104, 105, 110, 111, 113, 115–117, 121, 122, 129–136, 139, 141, 142, 148, 153–155, 162, 167–170, 173–179

GPLVM Gaussian process latent variable model. 36, 39, 40, 102, 108, 110–112, 116–118, 123, 124, 127, 128

GPS global positioning system. 94

HMC Hamiltonian Monte-Carlo. 66–69, 72, 74

IPCC Intergovernmental Panel on Climate Change. 165, 170

KLD Kullback-Leibler divergence. 5–7, 16, 38, 46, 51–54, 119, 157

KSD kernel Stein discrepancy. 51, 53, 158–161, 166

MAP maximum a posteriori. 118

MCMC Markov chain Monte-Carlo. 3, 4, 24, 39, 44–46, 56, 70, 130, 139

MMM multi-model mean. 165, 166

MP Members of Parliament. 103, 104

NO_2 nitrogen dioxide. 70, 71, 77–80, 83–85, 87, 89, 90, 176

PCA principal component analysis. 108, 110, 112, 123

PLV projected latent variables. 28

PM particulate matter. 91

PSD positive semi-definite. 19, 20

RBF radial basis function. 20, 22, 53, 68

RCP representative concentration pathway. 145, 146, 163

RKHS reproducing kernel Hilbert space. 46–48, 51, 52

RMSE root-mean-square error. 68, 69, 71

SOR Subset of Regressors. 29

SPDE stochastic partial differential equation. 104, 115

SPGP Sparse Pseudo-input GP. 29

SSP shared socio-economic pathway. 144–146, 151, 163–166

SVGD Stein variational gradient descent. 3, 4, 24, 39, 40, 43–46, 49, 51, 52, 54–58, 63, 64, 66, 72, 74, 75, 175

UFP ultrafine particle. 78, 91–95, 97

UK United Kingdom. 23, 70, 78, 79, 91, 93, 103, 175

USA United States of America. 103

VAE variational autoencoder. 55

VFE variational free energy. 28, 31, 35–39, 43

VI variational inference. 3, 5, 6, 8, 10, 11, 31, 39, 44–46, 51, 52, 54–56, 65–69, 71–74, 86, 154

List of source codes

Chapter 1

Introduction

This thesis contributes to the methodological intersection of probabilistic modelling and environmental sciences by developing Gaussian process (GP) models to better understand and model problems concerning air quality and climate science. Consequently, this work seeks to affect both statisticians and environmental scientists by presenting the complex challenges that accompany modelling environmental data to the statistics and machine learning communities, whilst simultaneously presenting a suite of new methodologies to the environmental sciences community to allow them to better model the complex phenomena that they are frequently presented with.

Within this thesis, we adopt a probabilistic stance when approaching either a methodological or an applied problem. This is appealing, as it allows us to build models that attain accurate predictions and precisely characterise the uncertainty in the prediction. When working in an applied field, such as environmental sciences, this becomes an indispensable tool as it allows the end users of our work to reason and make decisions under the uncertainty of the model. Practically speaking, this may involve placing a new air quality sensor in an area of high

predictive uncertainty to improve our understanding of where data is currently sparse, or providing faithful estimates of the range of possible global surface temperatures in 2100.

1.1 Probabilistic modelling

A probabilistic modelling task is comprised of an observed dataset $\mathbf{y}$ for which we construct a model. The parameters θ of our model are unknown, and our goal is to conduct inference to determine their range of likely values. To achieve this, we apply Bayes' theorem (Bayes, 1763; Laplace, 1774)

$$p(\theta \,|\, \mathbf{y}) = \frac{p(\theta)p(\mathbf{y} \,|\, \theta)}{p(\mathbf{y})} = \frac{p(\theta)p(\mathbf{y} \,|\, \theta)}{\int_\theta p(\mathbf{y}, \theta) \mathrm{d}\theta}\,, \tag{1.1}$$

where $p(\mathbf{y} \,|\, \theta)$ denotes the *likelihood*, or model, and quantifies how likely the observed dataset $\mathbf{y}$ is, given the parameter estimate θ. The *prior* distribution $p(\theta)$ reflects our initial beliefs about the value of θ before observing data, whilst the *posterior* $p(\theta \,|\, \mathbf{y})$ gives an updated estimate of the parameters' value, after observing $\mathbf{y}$. The *marginal likelihood*, or Bayesian model evidence, $p(\mathbf{y})$ is the probability of the observed data under all possible hypotheses that our prior model can generate (Mackay, 1992). Within Bayesian model selection, this property makes the marginal log-likelihood an indispensable tool. Selecting models under this criterion, an approach known as Bayes' factors, places a higher emphasis on models that can generalise better to new data points.

When the posterior distribution belongs to the same family of probability distributions as the prior, we describe the prior and the likelihood as *conjugate* to each other. Such a scenario is convenient in Bayesian inference as it allows us to derive closed-form expressions for the posterior distribution. When the likelihood

function is a member of the exponential family, then there exists a conjugate prior. However, the conjugate prior may not have a form that precisely reflects the practitioner's belief surrounding the parameter. For this reason, conjugate models seldom appear; one exception to this is Gaussian process (GP) regression that we will present in Section 1.3.3.

As a motivating example of the conjugate prior, consider the following model:

$$\mathbf{y} \,|\, \mu \sim \mathcal{N}(\mu, \sigma^2) \tag{1.2}$$

$$\mu \sim \mathcal{N}(m, \tau^2)\,. \tag{1.3}$$

We have a Gaussian likelihood function with unknown mean parameter μ and known variance σ^2. By placing a Gaussian prior on μ, we can analytically derive the corresponding posterior distribution as

$$p(\mu \,|\, \mathbf{y}) = \mathcal{N}\left(m\frac{\sigma^2}{n\tau^2+\sigma^2} + \bar{y}\frac{n\tau^2}{n\tau^2+\sigma^2}, \frac{\tau^2\sigma^2}{n\tau^2+\sigma^2}\right), \tag{1.4}$$

where $\bar{y} = n^{-1}\sum_{i=1}^{n} y_i$. The prior and posterior distributions are both Gaussian distributions, thus verifying the model's conjugacy. We visualise this update in Figure 1.1.

When our model does not contain a conjugate prior, we must compute the marginal log-likelihood to normalise the posterior distribution and ensure it integrates to 1. For models with a single, 1-dimensional parameter, it may be possible to compute this integral analytically or through a quadrature scheme, such as Gauss-Hermite (Abramowitz et al., 1972). However, in machine learning, the dimensionality of θ is often large and the corresponding integral required to compute $p(\mathbf{y})$ quickly becomes intractable as the dimension grows. Techniques such as Markov chain Monte-Carlo (MCMC), variational inference (VI), or Stein variational gradient

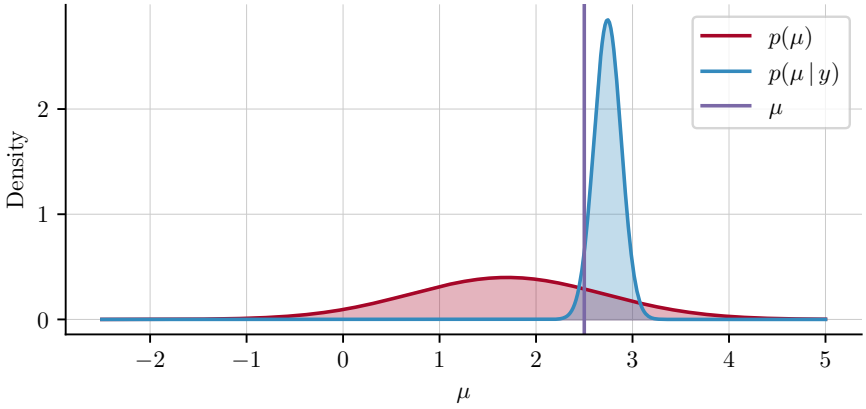


Figure 1.1: The closed form Gaussian posterior $p(\mu \mid \mathbf{y})$ distribution of the mean parameter μ whose prior $p(\mu)$ is also a Gaussian distribution.

descent (SVGD) allow us to approximate[1] integrals such as the one seen in $p(\mathbf{y})$. We devote Chapter 2 to considering how SVGD can be applied to a range of GP models.

Once a posterior distribution has been obtained, we can make predictions at new points $\mathbf{y}^\star$ through the *posterior predictive distribution*. This is achieved by integrating out the parameter set θ from our posterior distribution through

$$p(\mathbf{y}^\star \mid \mathbf{y}) = \int_\theta p(\mathbf{y}^\star, \theta \mid \mathbf{y}) \mathrm{d}\theta \tag{1.5}$$

$$= \int_\theta p(\mathbf{y}^\star \mid \theta, \mathbf{y}) p(\theta \mid \mathbf{y}) \mathrm{d}\theta \,. \tag{1.6}$$

As with the marginal log-likelihood, evaluating this quantity requires computing an integral which may not be tractable, particularly when θ is high-dimensional.

It is difficult to communicate statistics directly through a posterior distribution as it is often not possible to assigned contextual information to the distribution's shape within the scope of the modelling task at hand. To resolve this, we often compute and report moments of the posterior distribution which serve

[1] In the asymptotic limit, MCMC will be exact. However, this is a theoretical result that we cannot rely on computationally.

as informative summary statistics. When the margnial posterior distribution is Gaussian (Section 1.3.1), we often report the first moment and the centred second moment

$$\mu = \mathbb{E}[\theta \,|\, \mathbf{y}] = \int \theta p(\theta \,|\, \mathbf{y}) \mathrm{d}\theta \tag{1.7}$$

$$\sigma^2 = \mathbb{V}[\theta \,|\, \mathbf{y}] = \int (\theta - \mathbb{E}[\theta \,|\, \mathbf{y}])^2 \, p(\theta \,|\, \mathbf{y}) \mathrm{d}\theta \,. \tag{1.8}$$

Through this pair of statistics, we may communicate our beliefs about the most likely value of θ i.e., μ, and the uncertainty σ around the expected value. However, as with the marginal log-likelihood and predictive posterior distribution, computing these statistics again requires a potentially intractable integral.

1.2 Variational inference

When the marginal log-likelihood of our Bayesian model is intractable, we are unable to evaluate the true posterior distribution as it is only known in its unnormalised form i.e., $p(\theta \,|\, \mathbf{y}) \propto p(\mathbf{y} \,|\, \theta) p(\theta)$. VI introduces a λ-parameterised approximating distribution $q_\lambda(\theta)$ that belongs to a family of probability distributions, $\mathcal{Q}$, such as the family of Gaussian distributions. VI minimises the distance between $q_\lambda(\theta)$ and the true posterior $p(\theta \,|\, \mathbf{y})$ by optimising with respect to λ (Jordan et al., 1998). We refer to q as the *variational distribution*, λ as the *variational parameters*, and hereon drop the dependence of q on λ for notational clarity.

The Kullback-Leibler divergence (KLD) (Kullback et al., 1951) is a function commonly used within VI to measure the discrepancy from[2] one density to another.

[2]The KLD is an asymmetric function, so we say it is evaluated from one density to another, not between.

Definition 1.2.1 (Kullback-Leibler Divergence) *Let p and q be continuous probability distributions whose support set is $\mathcal{X}$. The Kullback-Leibler divergence from q to p is defined as*

$$\mathrm{KL}(q(\cdot)\,||\,p(\cdot)) = -\int_{\mathcal{X}} q(\cdot)\log\frac{p(\cdot)}{q(\cdot)}\mathrm{d}\theta \tag{1.9}$$

$$= -\mathbb{E}_q\left[\log\frac{p(\cdot)}{q(\cdot)}\right]. \tag{1.10}$$

Within VI, it is common to let the distribution p for which we are trying to minimise against be the true posterior distribution $p(\theta\,|\,\mathbf{y})$. However, directly minimising Equation (1.9) is not possible as it requires explicit evaluation of $p(\theta\,|\,\mathbf{y})$ whose unknown form is the reason we have to use VI. We can reveal the dependence on the marginal log-likelihood in Equation (1.9) by expanding the posterior distribution to give

$$\mathrm{KL}\left(q(\theta)\,||\,p(\theta\,|\,\mathbf{y})\right) = \int q(\theta)\log\frac{q(\theta)}{p(\theta\,|\,\mathbf{y})} \tag{1.11}$$

$$= \mathbb{E}_q\left[\log q(\theta)\right] - \mathbb{E}_q\left[\log\frac{p(\theta,\mathbf{y})}{p(\mathbf{y})}\right] \tag{1.12}$$

$$= \mathbb{E}_q[\log q(\theta)] - \mathbb{E}_q[\log p(\theta,\mathbf{y})] + \log p(\mathbf{y})\,. \tag{1.13}$$

We can see that the intractable marginal log-likelihood $p(\mathbf{y})$ is present in Equation (1.13) and it is therefore not possible to evaluate the corresponding KLD term. Instead, by rearranging Equation (1.13) and switching the signs, we can define the tractable evidence lower bound (ELBO) $\mathcal{L}$ as

$$\log p(\mathbf{y}) - \mathrm{KL}\left(q(\theta)\,||\,p(\theta\,|\,\mathbf{y})\right) = \underbrace{\mathbb{E}_q[\log p(\theta,\mathbf{y})] - \mathbb{E}_q[\log q(\theta)]}_{=\mathcal{L}(q)}\,. \tag{1.14}$$

The marginal log-likelihood is constant with respect to the variational posterior, and the evidence lower bound (ELBO) is, therefore, equal to the negative KLD from the variational posterior to the true posterior, up to this constant term. Crucially, since the KLD is non-negative, the left-hand side of Equation (1.14) is a lower bound on the marginal log-likelihood. Therefore, maximisation of the ELBO gives a computationally tractable way to minimise the KLD from the variational approximation to the true posterior distribution. In Figure 1.2 we visualise the variational approximation to the Gaussian-Gaussian model that was presented in Equations (1.2)–(1.4). After initialising the variational mean of q to be 1 and the variance as 0.5^2, we can see that over 50 iterates of the Adam (Ba et al., 2015) optimiser with a learning rate of 0.01, the variational approximation perfectly recovers the true posterior distribution. This is to be expected, given that the true posterior distribution is a Gaussian distribution and therefore belongs to our family of variational distributions i.e., $p(\theta \,|\, \mathbf{y}) \in \mathcal{Q}$.

We can further examine the ELBO term in Equation (1.14) and reduce it into quantities that are simple to compute in most computational settings. To do this, we first expand the joint distribution in Equation (1.14) before collecting the terms up into an expected log-likelihood function and a KLD term from the variational distribution to prior

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(\mathbf{y} \,|\, \theta)] + \mathbb{E}_q[\log p(\theta)] - \mathbb{E}_q[\log q(\theta)] \tag{1.15}$$

$$= \mathbb{E}_q\left[\log \frac{p(\theta)}{q(\theta)}\right] + \mathbb{E}_q\left[\log p(\mathbf{y} \,|\, \theta)\right] \tag{1.16}$$

$$= \mathbb{E}_q[\log p(\mathbf{y} \,|\, \theta)] - \mathrm{KL}(q(\theta) \,||\, p(\theta))\,. \tag{1.17}$$

When optimising Equation (1.15) with respect to the variational parameters, we can see that the KLD term acts as a prior regularisation term that will

penalise a variational approximation that diverges too far from the model's prior distribution, whilst the expected log-likelihood measure will encourage variational approximations that best explain the observed dataset.

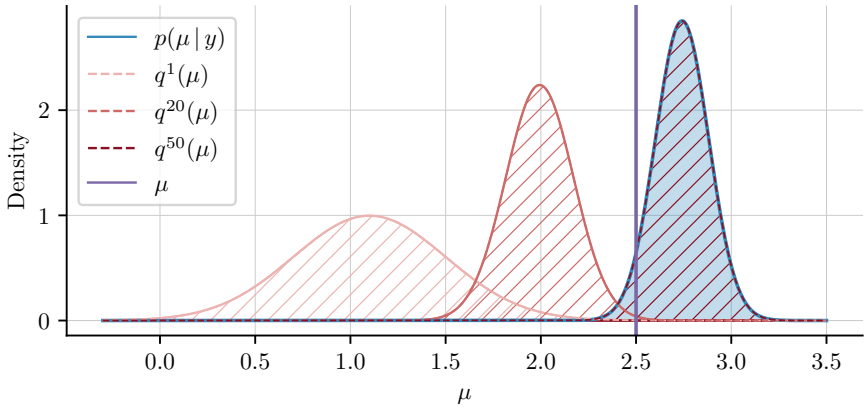


Figure 1.2: Comparison of the variational approximation $q(\mu)$ to the true posterior distribution $p(\mu \mid \mathbf{y})$ under the model given in Equations (1.2)–(1.4). We visualise the variational approximation after 1, 20 and 50 steps of the optimisation routine, as denoted by the superscript on q.

We will now introduce an alternative way to bound the marginal log-likelihood in VI that relies on Jensen's inequality. This construction will be a helpful framework to keep in mind for the remainder of this thesis. We start by establishing the notion of a *convex* function.

Definition 1.2.2 (Convex function) *A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if for any $x_1, x_2 \in \mathbb{R}^d$ and every $c \in [0, 1]$ we have*

$$cf(x_1) + (1 - c)f(x_2) \geq f(cx_1 + (1 - c)x_2)\,. \tag{1.18}$$

Informally, a convex function is a U-shaped function such that if we pick any two values x_1 and x_2 and draw a straight line between $f(x_1)$ and $f(x_2)$, then the line will be above the function's curve. We visualise this in Figure 1.3.

Using the notion of a convex function, we can define Jensen's inequality.

Theorem 1.2.3 (Jensen's Inequality) *Let* $\mathbf{y}$ *be a random variable and* f *be a convex function. Jensen's inequality states*

$$f\left(\mathbb{E}\left[\mathbf{y}\right]\right) \leq \mathbb{E}\left[f\left(\mathbf{y}\right)\right] . \tag{1.19}$$

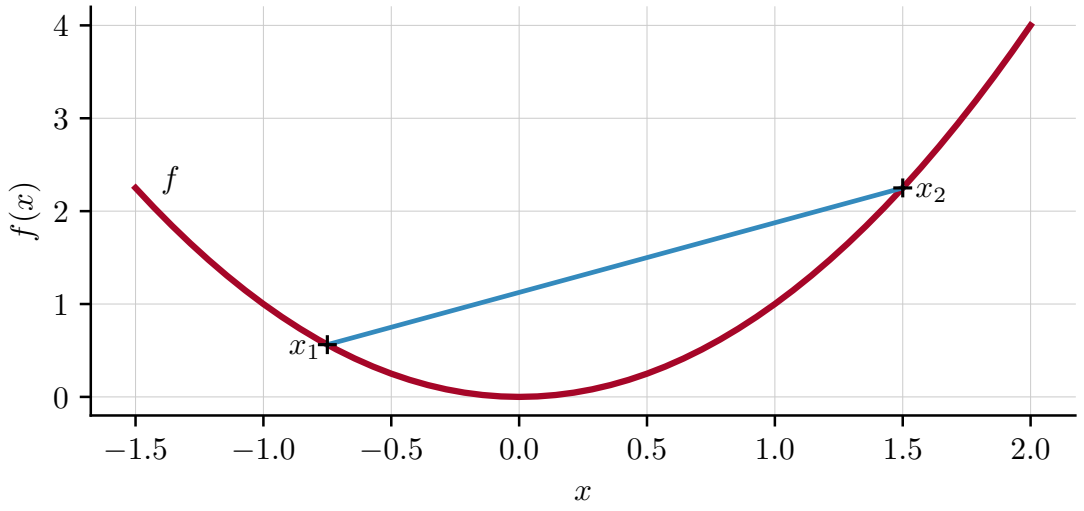


Figure 1.3: The convex function $f(x) = x^2$ evaluated at $x_1 = -0.75$ and $x_2 = 1.5$.

Starting from the definition of the marginal log-likelihood, we use Jensen's inequality (Jensen, 1906) to derive an ELBO term as follows

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \int p(\mathbf{y}, \theta) \mathrm{d}\theta && (1.20)\\
&= \log \int p(\mathbf{y}, \theta) \frac{q(\theta)}{q(\theta)} \mathrm{d}\theta && (1.21)\\
&= \log \mathbb{E}_q \left[\frac{p(\mathbf{y}, \theta)}{q(\theta)} \right] && (1.22)\\
&\geq \mathbb{E}_q \left[\log \frac{p(\mathbf{y}, \theta)}{q(\theta)} \right] . && (1.23)
\end{aligned}$$

Having applied Jensen's inequality to get from the penultimate line to the final line, we can rearrange Equation (1.20) to obtain a computationally tractable bound

that is equivalent to that given in Equation (1.15)

$$\mathbb{E}_q\left[\log \frac{p(\mathbf{y}, \theta)}{q(\theta)}\right] = \mathbb{E}_q[\log p(\mathbf{y}, \theta) - \log q(\theta)] \tag{1.24}$$

$$= \mathbb{E}_q[\log p(\mathbf{y} \,|\, \theta)] + \log p(\theta) - \log q(\theta)] \tag{1.25}$$

$$= \mathbb{E}_q[\log p(\mathbf{y} \,|\, \theta)] - \mathrm{KL}(q(\theta) \,||\, p(\theta))\,. \tag{1.26}$$

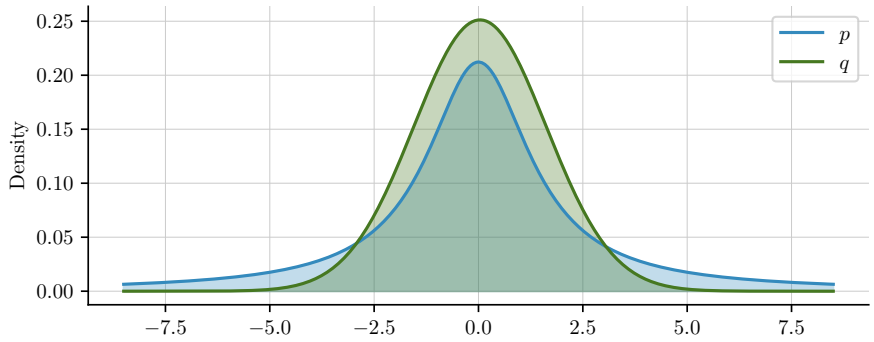


Figure 1.4: Demonstration of the shortcomings experienced in variational inference when the target distribution is non-Gaussian.

We close this section with a word of caution that, whilst elegant in its formulation and fast to compute, VI solutions can be inaccurate when the posterior distribution does not belong to the variational family of distributions $\mathcal{Q}$ specified by the practitioner. This error between the true distribution and the variational approximation is referred to as the *approximation gap* (Cremer et al., 2018). As an example of this, we consider the synthetic example where the distribution we are trying to approximate p is a Cauchy distribution with a location of 0 and scale of 1.5 (Figure 1.4). With a variational Gaussian approximation, the variational distribution q recovers sensible parameters with a location of 0.04 and a scale of 1.59. However, the variational approximation is unable to precisely model the underlying density's heavy tails as it is constrained by the choice of $\mathcal{Q}$. Whilst a more accurate variational approximation could be obtained by letting $\mathcal{Q}$ be the

family of Cauchy distributions, it is often hard in practice to accurately specify the correct form of $\mathcal{Q}$ apriori. Further, many of the computational enhancements that VI provides depend on $\mathcal{Q}$ being the family of Gaussian distributions. We address this issue in the case of GPs in Chapter 2.

1.3 An introduction to Gaussian processes

For a set $\mathcal{X}$, a stochastic process $f : \mathcal{X} \to \mathbb{R}$ is a GP if for any finite collection $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathcal{X}$ the random variable $\mathbf{f} := f(X)$ is distributed according to a joint Gaussian distribution (Rasmussen et al., 2006). For this reason, we begin this section by briefly reviewing Gaussian random variables and some basic results from probability theory that describe how we can marginalise and condition one Gaussian random variable on another Gaussian random variable. Notationally, we will let x and y be univariate random variables and $\mathbf{x}$ and $\mathbf{y}$ be multivariate random variables.

1.3.1 Gaussian random variables

We begin our review with the simplest case; a univariate Gaussian random variable.

Definition 1.3.1 (Univariate Gaussian random variable) *Let y be a random variable, $\mu \in \mathbb{R}$ be a mean scalar and $\sigma^2 \in \mathbb{R}_{>0}$ a variance scalar. If y is a Gaussian random variable, then the density of y is*

$$\mathcal{N}(y \,|\, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right), \tag{1.27}$$

We visualise three different parameterisations of Equation (1.27) in Figure 1.5.

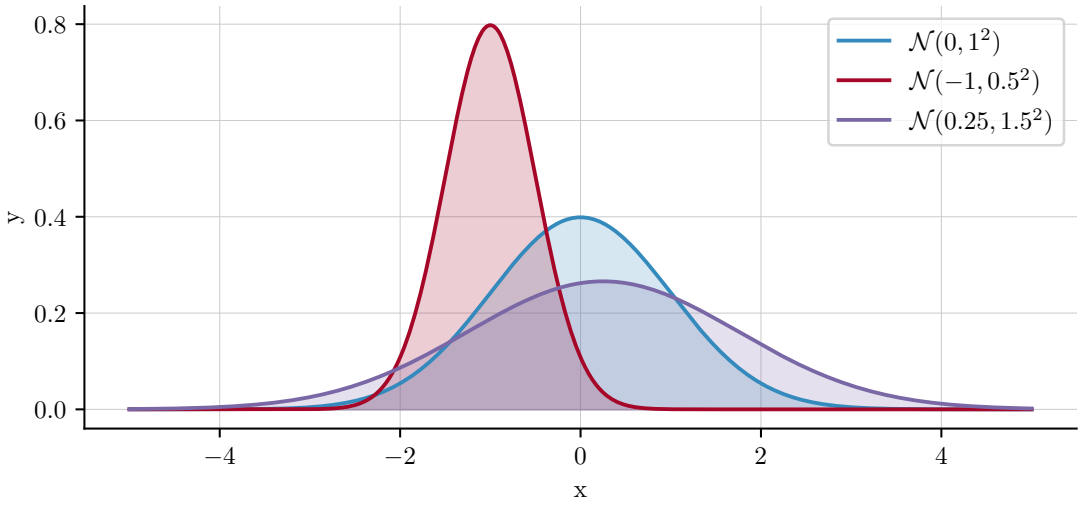


Figure 1.5: Probability density functions of three univariate Gaussian random variables with different mean and variance parameters.

A Gaussian random variable is uniquely defined in distribution by its mean μ and variance σ^2 and we therefore write $y \sim \mathcal{N}(\mu, \sigma^2)$ when describing a Gaussian random variable. Further, we can compute these two quantities by

$$\mathbb{E}[y] = \mu\,, \qquad \mathbb{E}\left[(y-\mu)^2\right] = \sigma^2\,. \tag{1.28}$$

Extending this concept to vector-valued random variables reveals the multivariate Gaussian random variables which brings us closer to the full definition of a GP.

Definition 1.3.2 (Multivariate Gaussian random variable) *Let* $\mathbf{y}$ *be a* D*-dimensional random variable,* $\boldsymbol{\mu}$ *be a* D*-dimensional mean vector and* $\boldsymbol{\Sigma}$ *be a* $D \times D$ *covariance matrix. If* $\mathbf{y}$ *is a Gaussian random variable, then the density of* $\mathbf{y}$ *is*

$$\mathcal{N}(\mathbf{y} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{2\pi}^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}\left(\mathbf{y}-\boldsymbol{\mu}\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{y}-\boldsymbol{\mu}\right)\right)\,. \tag{1.29}$$

We visualise three different 2-dimensional multivariate Gaussian random variables in Figure 1.6.

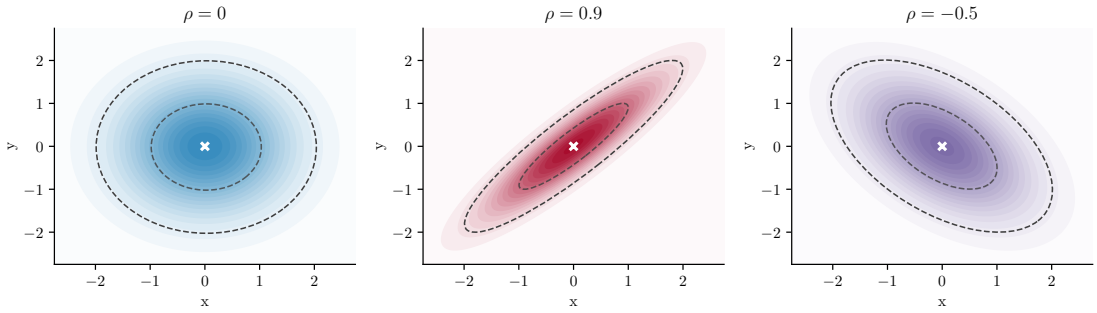


Figure 1.6: Probability density functions of three bivariate Gaussian random variables. Each random variable has an expectation of $(0, 0)$, marginal variance of 1, and correlation of ρ.

Extending the intuition given for univariate Gaussian random variables in Equation (1.28), we can obtain the mean and covariance by

$$\mathbb{E}[\mathbf{y}] = \boldsymbol{\mu}\,, \quad \operatorname{Cov}(\mathbf{y}) = \mathbb{E}\left[(\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^\top\right] \tag{1.30}$$

$$= \mathbb{E}[\mathbf{y}\mathbf{y}^\top] - \mathbb{E}[\mathbf{y}]\mathbb{E}[\mathbf{y}]^\top \tag{1.31}$$

$$= \boldsymbol{\Sigma}\,. \tag{1.32}$$

The covariance matrix is a symmetric positive definite matrix that generalises the notion of variance to multiple dimensions. The matrix's diagonal entries contain the variance of each element, whilst the off-diagonal entries quantify the degree to which the respective pair of random variables are linearly related; this quantity is called the *covariance*.

Assuming a Gaussian likelihood function in a Bayesian model is attractive as the mean and variance parameters are highly interpretable. This makes prior elicitation straightforward as the parameters' values can be intuitively contextualised within the scope of the problem at hand. Further, in models where the posterior distribution is Gaussian, we again use the distribution's mean and variance to describe our prediction and corresponding uncertainty around a given

event occurring.

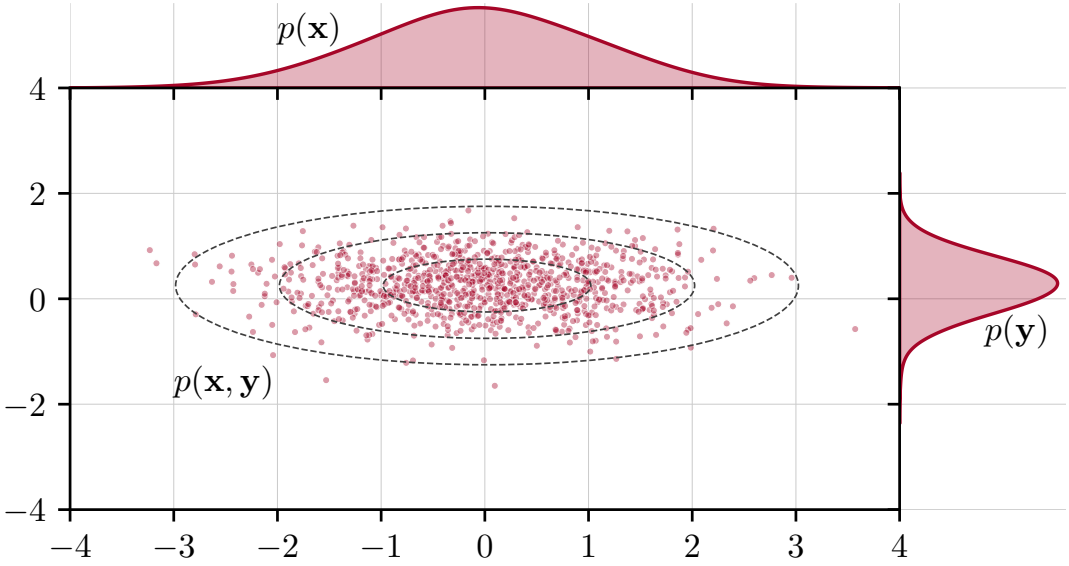


Figure 1.7: A joint probability distribution $p(\mathbf{x}, \mathbf{y})$ with marginal distributions $p(\mathbf{x}) = \mathcal{N}(0, 1^2)$ and $p(\mathbf{y}) = \mathcal{N}(0.25, 0.5^2)$. The inner panel visualises 1000 draws from the joint distribution (red points) and contours of constant density (grey lines). In the side panels, we observe the probability distribution of each marginal.

Not only are Gaussian random variables highly interpretable, but linear operations involving them lead to analytical solutions. We have already seen this in the conjugate prior example given in Section 1.1, but a second example that will be useful in the sequel is the marginalisation and conditioning property of sets of Gaussian random variables. We will present these two results now for a pair of Gaussian random variables, but it should be stressed that these results hold for any finite set of Gaussian random variables.

For a pair of random variables $\mathbf{x}$ and $\mathbf{y}$ defined on the same support, the distribution over them both is known as the *joint distribution*. The joint distribution $p(\mathbf{x}, \mathbf{y})$ quantifies the probability of two events, one from $p(\mathbf{x})$ and another from $p(\mathbf{y})$, occurring at the same time. We visualise this idea in Figure 1.7.

Definition 1.3.3 (Joint Gaussian distribution) *Let $p(\mathbf{x}, \mathbf{y})$ be the joint probability*

distribution defined over $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_{\mathbf{xx}})$ *and* $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{y}, \boldsymbol{\Sigma}_{\mathbf{yy}})$. *We define the joint distribution as*

$$p\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_\mathbf{x}\\ \boldsymbol{\mu}_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{\mathbf{xx}}, \boldsymbol{\Sigma}_{\mathbf{xy}}\\ \boldsymbol{\Sigma}_{\mathbf{yx}}, \boldsymbol{\Sigma}_{\mathbf{yy}}\end{bmatrix}\right), \tag{1.33}$$

where $\boldsymbol{\Sigma}_{\mathbf{xy}}$ *is the cross-covariance matrix of* $\mathbf{x}$ *and* $\mathbf{y}$.

When presented with a joint distribution, two tasks that we may wish to perform are *marginalisation* and *conditioning*. For a joint distribution $p(\mathbf{x}, \mathbf{y})$ where we are interested only in $p(\mathbf{x})$, we must integrate over all possible values of $\mathbf{y}$ to obtain $p(\mathbf{x})$. This process is marginalisation. Conditioning allows us to evaluate the probability of one random variable, given that the other random variable is fixed. For a joint Gaussian distribution, marginalisation and conditioning have analytical expressions where the resulting distribution is also Gaussian.

Definition 1.3.4 (Gaussian marginalisation and conditioning) *For a joint Gaussian random variable of the form given in Equation (1.33), the marginalisation of* $\mathbf{x}$ *or* $\mathbf{y}$ *is given by*

$$\int p(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{y} = p(\mathbf{x}) = \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_{\mathbf{xx}}) \tag{1.34}$$

$$\int p(\mathbf{x}, \mathbf{y}) \mathrm{d}\mathbf{x} = p(\mathbf{y}) = \mathcal{N}(\mathbf{y} \,|\, \boldsymbol{\mu}_\mathbf{y}, \boldsymbol{\Sigma}_{\mathbf{yy}})\,. \tag{1.35}$$

The conditional distributions are given by

$$p(\mathbf{x} \,|\, \mathbf{y}) = \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{x} + \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}(\mathbf{y} - \boldsymbol{\mu}_\mathbf{y}), \boldsymbol{\Sigma}_{\mathbf{xx}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}}\right) \tag{1.36}$$

$$p(\mathbf{y} \,|\, \mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}_\mathbf{y} + \boldsymbol{\Sigma}_{\mathbf{yx}}\boldsymbol{\Sigma}_{\mathbf{xx}}^{-1}(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}), \boldsymbol{\Sigma}_{\mathbf{yy}} - \boldsymbol{\Sigma}_{\mathbf{yx}}\boldsymbol{\Sigma}_{\mathbf{xx}}^{-1}\boldsymbol{\Sigma}_{\mathbf{xy}}\right)\,. \tag{1.37}$$

Returning to the KLD that was presented in Definition 1.2.1, when evaluating the KLD from one Gaussian random variable to another, the divergence can be computed in closed form. When computing a variational approximation to the posterior distribution of a GP, something that we will do in the sequel, this analytical form will be indispensable.

Definition 1.3.5 (Kullback-Leibler between multivariate Gaussians) *Letting* $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{xx}})$ *and* $q(\mathbf{y}) = \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{yy}}\right)$*, the KLD from* $p(\mathbf{x})$ *to* $q(\mathbf{y})$ *is given by*

$$\begin{aligned}\mathrm{KL}\left(p(\mathbf{x}) \,||\, q(\mathbf{y})\right) = \frac{1}{2}\Big(&\operatorname{trace}\left(\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\boldsymbol{\Sigma}_{\mathbf{xx}}\right) + \left(\boldsymbol{\mu}_{\mathbf{y}} - \boldsymbol{\mu}_{\mathbf{x}}\right)^{\top}\boldsymbol{\Sigma}_{\mathbf{yy}}^{-1}\left(\boldsymbol{\mu}_{\mathbf{y}} - \boldsymbol{\mu}_{\mathbf{x}}\right) \\ &- D + \log\det\boldsymbol{\Sigma}_{\mathbf{yy}} - \log\det\boldsymbol{\Sigma}_{\mathbf{xx}}\Big), \end{aligned} \tag{1.38}$$

where $\mathbf{x}$ *and* $\mathbf{y}$ *are random variables with equal dimensions* D*.*

Within this section, we have introduced the idea of multivariate Gaussian random variables and presented some key results concerning their properties. In the following section, we will lift our presentation of Gaussian random variables to GPs.

1.3.2 Gaussian processes

When transitioning from Gaussian random variables to GPs there is a shift in thought required to parse the forthcoming material. Firstly, to be consistent with the general literature, we hereon use $\mathbf{x}$ to denote an observed vector of data points, not a random variable as has been true up until now. To distinguish between matrices and vectors, we use bold upper case characters e.g., $\mathbf{X}$ for matrices, and bold lower case characters for vectors e.g., $\mathbf{x}$.

We are interested in modelling supervised learning problems, where we have n ob-

servations $\mathbf{y} = \{y_1, y_2, \ldots, y_n\} \subset \mathcal{Y}$ at corresponding inputs $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \subset \mathcal{X}$. We aim to capture the relationship between $\mathbf{X}$ and $\mathbf{y}$ using a model f with which we may make predictions at an unseen set of test points $\mathbf{X}^\star \subset \mathcal{X}$. We formalise this by

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon\,, \tag{1.39}$$

where ε is an observational noise term. We collectively refer to $(\mathbf{X}, \mathbf{y})$ as the *training data* and $\mathbf{X}^\star$ as the set of *test points*. This process is visualised in Figure 1.8. As we shall go on to see, GPs offer an appealing workflow for scenarios such as this, all under the Bayesian framework of Section 1.1.

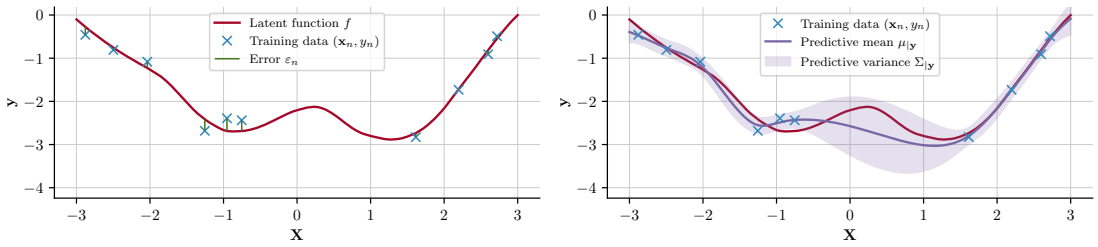


Figure 1.8: Left: Data generating process where 10 data points $\{\mathbf{x}_n, y_n\}_{n=1}^{1}0$ (blue crosses) are realisations of a latent function (red line), subject to observational noise (vertical green lines). Right: The predictive posterior distribution of a Gaussian process conditioning on the 10 observed data points.

The covariance matrix that parameterises a multivariate Gaussian distribution is finite-dimensional. Consequently, it is not possible to evaluate a multivariate random variable at a location not present in our observed dataset. A kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ resolves this by allowing us to compute the (co)variance between any two points in $\mathcal{X}$. Using a kernel function k, we write a GP $f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot, \cdot))$ with mean function $\mu : \mathcal{X} \to \mathbb{R}$ and $\boldsymbol{\theta}$-parameterised kernel k (see Definition 1.3.6). When evaluating the GP on a finite set of points $\mathbf{X} \subset \mathcal{X}$, k gives rise to the Gram matrix $\mathbf{K}_{\mathbf{ff}}$ such that the $(i, j)^{\text{th}}$ entry of the matrix is given by $[\mathbf{K}_{\mathbf{ff}}]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

As is conventional within the literature, we centre our training data and assume $\mu(\mathbf{x}) := 0$ for all $\mathbf{x} \in \mathbf{X}$. We further drop dependency on $\boldsymbol{\theta}$ and $\mathbf{X}$ for notational convenience in the remainder of this thesis.

We define a joint GP prior over the latent function

$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K_{ff}} & \mathbf{K_{f*}} \\ \mathbf{K_{*f}} & \mathbf{K_{\star\star}} \end{bmatrix}\right), \tag{1.40}$$

where $\mathbf{f}^\star = f(\mathbf{X}^\star)$. Conditional on the GP's latent function f, we assume a factorising likelihood generates our observations

$$p(\mathbf{y} \mid \mathbf{f}) = \prod_{i=1}^{n} p(y_i \mid f_i)\,. \tag{1.41}$$

To be more precise, we can re-define the likelihood function as $p(\mathbf{y} \mid \phi(\mathbf{f}))$ where ϕ is the likelihood function's associated link function. Example link functions include the probit or logistic functions for a Bernoulli likelihood and the identity function for a Gaussian likelihood. We eschew this notation for now as this section primarily considers Gaussian likelihood functions where the role of ϕ is superfluous. However, this intuition will be helpful for the work presented in Chapter 4.

Applying Bayes' theorem yields the joint posterior distribution over the latent function

$$p(\mathbf{f}, \mathbf{f}^\star \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}, \mathbf{f}^\star)}{p(\mathbf{y})}\,. \tag{1.42}$$

The choice of kernel function that we use to parameterise our GP is an important modelling decision as the choice of kernel dictates properties such as differentiability, variance and characteristic lengthscale of the functions that are

admissible under the GP prior (Figure 1.9). A kernel is a positive-definite function with parameters $\boldsymbol{\theta}$ that maps pairs of inputs $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ onto the real line.

Definition 1.3.6 (Kernel function) *For a non-empty set $\mathcal{X}$, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel on $\mathcal{X}$ if there exists a Hilbert space $\mathcal{H}$ and map $\Phi : \mathcal{X} \to \mathcal{H}$ such that*

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \tag{1.43}$$

for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

Within Equation (1.43), we refer to Φ as the *feature map* and $\mathcal{H}$ as the *feature space* of k. We revisit this concept in greater detail in Section 2.2.1.

Definition 1.3.7 (Positive semi-definite matrix) *A symmetric matrix $\mathbf{A}$ is positive semi-definite if and only if*

$$\boldsymbol{\nu}^\top \mathbf{A} \boldsymbol{\nu} \geq 0 \tag{1.44}$$

for all $\boldsymbol{\nu} \in \mathbb{R}^n$.

Proposition 1.3.8 (All Gram matrices are positive semi-definite) *For a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with feature map Φ, the corresponding Gram matrix $\mathbf{K}$ that arises from computing the kernel function over a finite set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ satisfies the positive semi-definite (PSD) condition given in Definition 1.3.7.*

Proof. For any vector $\boldsymbol{\nu} \in \mathbb{R}^n$ we have

$$\begin{aligned}
\boldsymbol{\nu}^\top \mathbf{K} \boldsymbol{\nu} &= \sum_{i=1}^{n} \sum_{j=1}^{n} \nu_i \nu_j \mathbf{K}_{ij} && (1.45)\\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \nu_i \nu_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle && (1.46)\\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \nu_i \Phi(\mathbf{x}_i), \nu_j \Phi(\mathbf{x}_j) \rangle && (1.47)\\
&= \| \sum_{i=1}^{n} \nu_i \Phi(\mathbf{x}_i) \|^2 && (1.48)\\
&\geq 0 && (1.49)
\end{aligned}$$

□

The implication of this is that the eigenvalues of the Gram matrix are non-negative; a result that seeds the work presented in Section 1.3.4. Further, by Proposition 1.3.8, we can say that the kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semi-definite function if the matrices formed by evaluating k over any finite set $\mathbf{X}$ are PSD.

A subclass of the kernel functions introduced in Definition 1.3.6 whose value depends only on the difference between its inputs $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ is the family of *stationary* kernels. To see this with an example, we can define the stationary radial basis function (RBF) kernel and the non-stationary linear kernel

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell^2} \right) \tag{1.50}$$

$$k_{\text{lin}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \mathbf{x} \mathbf{x}'^\top . \tag{1.51}$$

We can see here that the RBF kernel is parameterised by a variance parameter σ^2 and a lengthscale parameter ℓ. Meanwhile, the linear kernel is parameterised by a

variance parameter σ^2.

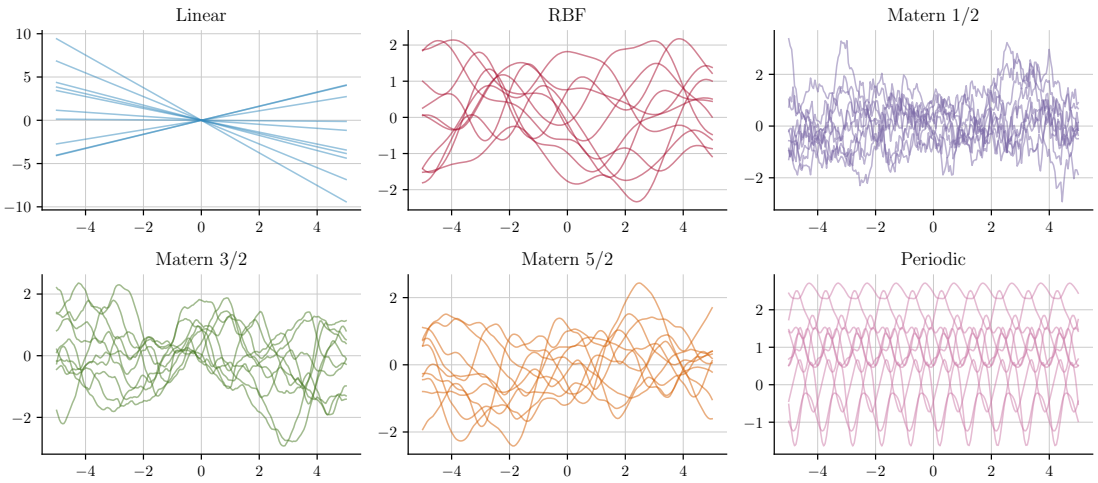


Figure 1.9: The functional priors induced by varying the kernel function. 10 samples are drawn independently from each prior model.

Another important kernel that will be pivotal to the work presented in Chapter 3 and Chapter 4 is the Matérn kernel (Matérn, 1960). We can write the Matérn kernel as

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right)^{\nu} B_\nu \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right) , \tag{1.52}$$

where Γ is the gamma function, B_ν is a modified Bessel function of the second kind, and $\nu \in \mathbb{R}_{>0}$ is a smoothness parameter that controls the GP's mean-square differentiability. When ν is half-integer, the Matérn kernel's form simplifies down to the following

$$k_{1/2}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp \left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right) \tag{1.53}$$

$$k_{3/2}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \frac{\sqrt{3}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right) \exp \left(\frac{-\sqrt{3}\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right) \tag{1.54}$$

$$k_{5/2}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \frac{\sqrt{5}\|\mathbf{x} - \mathbf{x}'\|}{\ell} + \frac{5\|\mathbf{x} - \mathbf{x}'\|^2}{3\ell^2} \right) \exp \left(-\sqrt{5} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\ell} \right) . \tag{1.55}$$

As the smoothness parameter ν increases, the GP's functions become increasingly smooth, and, in the limit that $\nu \to \infty$, we recover the RBF kernel from Equation (1.50). We visualise the GP prior samples induced by some commonly used kernels in Figure 1.9.

Finally, a convenient feature of kernels is that they are closed under addition and multiplication. This allows us to construct intricate, non-trivial kernels from a set of simpler kernels.

Proposition 1.3.9 (Closure of kernels) *For the kernel function k_1 defined on $\mathcal{X}_1$ and k_2 defined on $\mathcal{X}_2$ the following operations yield kernel functions*

1. $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$ *for* $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_1 \times \mathcal{X}_2$.
2. $k(\mathbf{x}, \mathbf{x}') = \alpha k_1(\mathbf{x}, \mathbf{x}')$ *for* $\alpha \in \mathbb{R}_{>0}$.
3. $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$ *for* $\mathbf{x}, \mathbf{x}' \in \mathcal{X}_1 \times \mathcal{X}_2$.

Proof. Let $\mathbf{K}_1$ and $\mathbf{K}_2$ be the Gram matrices formed by evaluating k_1 and k_1 on a set of points $\mathbf{X}_1 \in \mathcal{X}_1$ and $\mathbf{X}_2 \in \mathcal{X}_2$, respectively. We then have the following:

1. $\boldsymbol{\nu}^\top(\mathbf{K}_1 + \mathbf{K}_2)\boldsymbol{\nu} = \boldsymbol{\nu}^\top\mathbf{K}_1\boldsymbol{\nu} + \boldsymbol{\nu}^\top\mathbf{K}_2\boldsymbol{\nu} \geq 0$ which proves that $\mathbf{K}_1 + \mathbf{K}_2$ is a positive-definite matrix and, therefore, the kernel $k_1 + k_2$ is a valid kernel function.
2. The validity of αk_1 can be proven using the same approach: $\boldsymbol{\nu}^\top\alpha\mathbf{K}_1\boldsymbol{\nu} = \alpha\boldsymbol{\nu}^\top\mathbf{K}_1\boldsymbol{\nu} \geq 0$.
3. Let $\mathbf{K} = \mathbf{K}_1 \otimes \mathbf{K}_2$ be the Kronecker product of $\mathbf{K}_1$ and $\mathbf{K}_2$. The eigenvalues of $\mathbf{K}$ are given by multiplying the eigenvalues of $\mathbf{K}_1$ and $\mathbf{K}_2$ together. Therefore, $\mathbf{K}$ is positive-definite and the kernel k_1k_2 is a valid kernel function.

by Proposition 1.3.8. □

In practice, the implication of Proposition 1.3.9 is that one can combine kernels to construct models that accurately capture the phenomena being modelled. We make extensive use of this result in Section 2.4.3 where we combine a series of kernels to build a principled model of air pollution in the United Kingdom (UK).

1.3.3 Gaussian process regression

When the likelihood function from Equation (1.41) is a Gaussian distribution $p(y_i \,|\, f_i) = \mathcal{N}(y_i \,|\, f_i, \sigma_n^2)$, marginalising $\mathbf{f}$ from the joint posterior to obtain the posterior predictive distribution is exact

$$p(\mathbf{f}^\star \,|\, \mathbf{y}) = \int p(\mathbf{f}, \mathbf{f}^\star \,|\, \mathbf{y}) \mathrm{d}\mathbf{f} \tag{1.56}$$

$$= \mathcal{N}(\mathbf{f}^\star \,|\, \boldsymbol{\mu}_{\,|\,\mathbf{y}}, \boldsymbol{\Sigma}_{\,|\,\mathbf{y}}) \,, \tag{1.57}$$

where

$$\boldsymbol{\mu}_{\,|\,\mathbf{y}} = \mathbf{K}_{*\mathbf{f}} \left(\mathbf{K}_{\mathbf{ff}} + \sigma_n^2 \mathbf{I}_n \right)^{-1} \mathbf{y} \tag{1.58}$$

$$\boldsymbol{\Sigma}_{\,|\,\mathbf{y}} = \mathbf{K}_{\star\star} - \mathbf{K}_{*\mathbf{f}} \left(\mathbf{K}_{\mathbf{ff}} + \sigma_n^2 \mathbf{I}_n \right)^{-1} \mathbf{K}_{\mathbf{f}*} \,. \tag{1.59}$$

Further, the log of the marginal likelihood in Equation (1.42) can be analytically expressed as

$$\log p(\mathbf{y}) = \log \int p(\mathbf{y} \,|\, \mathbf{f}) p(\mathbf{f}, \mathbf{f}^\star) \mathrm{d}\mathbf{f}^\star \tag{1.60}$$

$$= 0.5 \left(- \underbrace{\mathbf{y}^\top \left(\mathbf{K}_{\mathbf{ff}} + \sigma_n^2 \mathbf{I}_n \right)^{-1} \mathbf{y}}_{\text{Data fit}} - \underbrace{\log |\mathbf{K}_{\mathbf{ff}} + \sigma_n^2 \mathbf{I}_n|}_{\text{Complexity}} - \underbrace{n \log 2\pi}_{\text{Constant}} \right) . \tag{1.61}$$

In the absence of any prior distributions on either the kernel parameters $\boldsymbol{\theta}$ or the observational noise σ_n^2, model selection can be performed for a GP through

gradient-based optimisation of $\log p(\mathbf{y})$. Collectively, we call these terms the model hyperparameters $\boldsymbol{\xi} = \{\boldsymbol{\theta}, \sigma_n^2\}$ from which the maximum likelihood estimate is given by

$$\boldsymbol{\xi}^\star = \arg\max_{\boldsymbol{\xi} \in \Xi} \log p(\mathbf{y}) \,,$$

where Ξ is the set of all possible hyperparameter values.

When a prior distribution has been placed on one or more of the model hyperparameters, Bayesian inference schemes such as MCMC or SVGD must be used to learn the full posterior distribution of the hyperparameters.

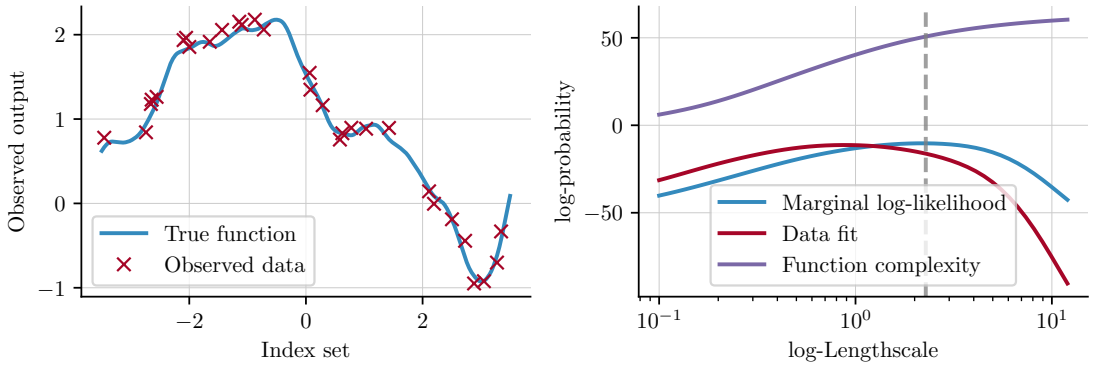


Figure 1.10: The decomposed marginal log-likelihood from Equation (1.60) as a function of the kernel's lengthscale parameter. The observed data (left panel) is drawn from a third-order Matérn process with lengthscale and variance parameters of 1. The observed data points are perturbed by a zero-mean Gaussian noise vector with a variance of 0.1. In the right panel, we set the observation noise and kernel variance terms to the true values and plot the decomposed marginal log-likelihood as a function of the kernel's lengthscale.

Observing the individual terms in Equation (1.60) can help understand exactly why optimising the marginal log-likelihood gives reasonable solutions. The *data fit* term is the only component of Equation (1.60) that includes the observed response $\mathbf{y}$ and will therefore encourage solutions that model the data well. Conversely, the *complexity* term contains a determinant operator and therefore measures the

volume of the function space covered by the GP. Whilst a more complex function has a better chance of modelling the observed data well, this is only true to a point and functions that are overly complex will overfit the data. Optimising with respect to Equation (1.60) balances these two objectives when identifying the optimal solution, as visualised in Figure 1.10.

1.3.4 Background on sparse Gaussian processes

The requirement to invert the $N \times N$ covariance matrix $\mathbf{K_{ff}}$ in Equation (1.60) means that the term's evaluation scales cubically in the number of data points. Historically, this made the deployment of GPs to real-world problems challenging as the corresponding datasets often surpassed tens of thousands of data points; a size that makes evaluating Equation (1.60) intractable. In this section, we'll review some of the early methods that were used to overcome this difficulty.

Because any positive-define kernel function yields a positive definite matrix, we can write the eigendecomposition of the covariance matrix as

$$\mathbf{K_{ff}} = \mathbf{U}\Lambda\mathbf{U}^\top , \tag{1.62}$$

where $\mathbf{U}$ an $N \times N$ orthonormal matrix whose columns are the eigenvectors and Λ is a diagonal matrix with eigenvalues of decreasing magnitude i.e., $\text{diag}(\Lambda) = [\lambda_1, \lambda_2, \ldots, \lambda_N]$ where $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$. Truncating the approximation in Equation (1.62) to include only the first M columns of $\mathbf{U}$ allows us to approximate $\mathbf{K_{ff}}$ by

$$\mathbf{K_{ff}} \approx \mathbf{U}_{:,M}\Lambda_{M,M}\mathbf{U}_{:,M}^\top . \tag{1.63}$$

Approximations of this kind are widely used for techniques such as principal

component analysis (Pearson, 1901). However, an eigendecomposition approach is ill-advised for GP models as computing the eigendecomposition is itself a cubic operation. Therefore, each time the kernel's parameters are updated the covariance matrix and its corresponding eigendecomposition will need to be re-computed. Despite this, it motivates the core idea behind some of the earliest sparse GP techniques; can we learn a low-rank approximation to the full covariance matrix in a manner that exudes better-than-cubic scaling in the number of data points?

To generalise the approximation given in Equation (1.63), we can write

$$\mathbf{K_{ff}} \approx \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^{\top}. \tag{1.64}$$

where $\mathbf{A} \in \mathbb{R}^{N\times M}$ and $\mathbf{B} \in \mathbb{R}^{M\times M}$. A natural question to then ask is how to find the matrices $\mathbf{A}$ and $\mathbf{B}$ that minimise $\|\mathbf{K_{ff}} - \mathbf{A}\mathbf{B}^{-1}\mathbf{A}^{\top}\|_F$ where $\|\cdot\|_F$ is the Frobenius norm? Approximations of this form are desirable as, assuming $N \gg M$, the dominant cost is now the matrix multiplications in Equation (1.64) that scale linearly in the number of data points and quadratically in M. However, it should be stressed that the value of M should be set pragmatically as the cost of inverting the $M \times M$ matrix $\mathbf{B}$ will scale cubically in M.

Within the GP literature, a popular framework that we make extensive use of in this thesis is that of *inducing points*. The idea is to summarise the N input data points $\mathbf{X}$ with a set of M inducing points $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M\}$ where $M \ll N$. For a kernel k, we can then build approximations of $\mathbf{K_{ff}}$ in the form of Equation (1.64) by

$$\mathbf{K_{ff}} \approx \mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{K_{fu}}^{\top}. \tag{1.65}$$

where $\mathbf{K_{fu}} = k(\mathbf{X}, \mathbf{Z})$ and $\mathbf{K_{uu}} = k(\mathbf{Z}, \mathbf{Z})$. For a well-designed algorithm, the

quality of the approximation in Equation (1.65) should improve as the number of inducing points increases (Figure 1.11).

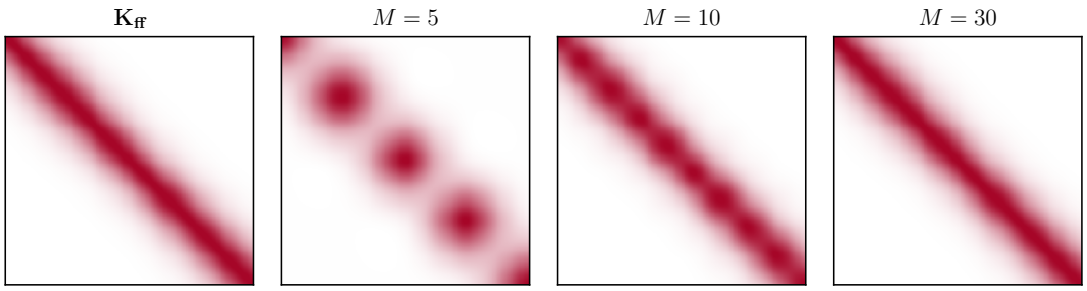


Figure 1.11: A series of low-rank approximation to the covariance matrix $\mathbf{K_{ff}}$ formed by evaluating the RBF kernel with unit lengthscale and variance parameter on 500 uniformly sampled points. The inducing points are chosen to be a random subset of the 500-point dataset. In panels 2-4, we see the improved approximation quality that is given as the number of inducing points increases from 5 to 10, to 30.

We call the evaluation of our GP at the inducing points the *inducing variable* and denote its value by $\mathbf{u} = f(\mathbf{Z})$. We can then augment the GP joint prior from Equation (1.40) with the inducing variable to give

$$p(\mathbf{f}^\star, \mathbf{f}) = \int p(\mathbf{f}^\star, \mathbf{f}, \mathbf{u}) \mathrm{d}\mathbf{u} \tag{1.66}$$

$$= \int p(\mathbf{f}^\star, \mathbf{f} \,|\, \mathbf{u}) p(\mathbf{u}) \mathrm{d}\mathbf{u}\,, \tag{1.67}$$

where $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K_{uu}})$. By the GP's consistency property (Definition 1.3.4), we can be sure that augmenting the joint prior distribution with the inducing variable and marginalising it out leaves no imprint on the joint prior's distribution. However, the augmented prior distribution in Equation (1.66) is exact, meaning that inference will still scale cubically in the number of data points. To improve this scaling, we assume that $\mathbf{f}$ and $\mathbf{f}^\star$ are conditionally independent, given $\mathbf{u}$, to

give the approximate joint prior

$$p(\mathbf{f}^\star, \mathbf{f}) \approx q(\mathbf{f}^\star, \mathbf{f}) \tag{1.68}$$

$$= \int q(\mathbf{f}^\star \,|\, \mathbf{u}) q(\mathbf{f} \,|\, \mathbf{u}) p(\mathbf{u}) \mathrm{d}\mathbf{u}\,. \tag{1.69}$$

We refer to $q(\mathbf{f}^\star \,|\, \mathbf{u})$ as the *approximate test conditional* and $q(\mathbf{f} \,|\, \mathbf{u})$ as the *approximate training conditional*. For reference, the exact forms of these two conditional distributions are given by

$$p(\mathbf{f} \,|\, \mathbf{u}) = \mathcal{N}(\mathbf{f} \,|\, \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}) \tag{1.70}$$

$$p(\mathbf{f}^\star \,|\, \mathbf{u}) = \mathcal{N}(\mathbf{f}^\star \,|\, \mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{\star\star} - \mathbf{Q}_{\star\star})\,, \tag{1.71}$$

where $\mathbf{Q}_{\mathbf{ab}} = \mathbf{K}_{\mathbf{au}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{ub}}$. We visualise the predictive posterior of a sparse GP, namely the variational free energy (VFE) model covered in Section 1.3.5, along with its inducing points in Figure 1.12.

Deterministic Training Conditional The Deterministic Training Conditional (DTC) approach, originally named the projected latent variables (PLV) method (Seeger et al., 2003), induces scalability by representing M latent function values which are projected into an N-dimensional space when evaluating the likelihood

$$p(\mathbf{y} \,|\, \mathbf{f}) \approx q(\mathbf{y} \,|\, \mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \sigma^2\mathbf{I}_n)\,. \tag{1.72}$$

In the exposition of Quinonero-Candela et al. (2005), a more unified presentation of this method is given where the exact likelihood function is retained and the training conditional is approximated by

$$q_{\mathrm{DTC}}(\mathbf{f} \,|\, \mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{0})\,. \tag{1.73}$$

A valid Gaussian density cannot be parameterised by a $\mathbf{0}$ covariance, but we use this notation to illustrate the deterministic relationship that exists between $\mathbf{f}$ and $\mathbf{u}$.

The exact test conditional is retained i.e., $q(\mathbf{f}^\star \mid \mathbf{u}) = p(\mathbf{f}^\star \mid \mathbf{u})$ and the approximate marginal log-likelihood takes the form

$$\log p(\mathbf{y}) = \log \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{Q}_{\mathbf{ff}} + \sigma^2 \mathbf{I}_n) \,. \tag{1.74}$$

The DTC approximation provided a large improvement in scalable methods by correcting the nonsensical posterior variances that were experienced in previous methods, such as the Subset of Regressors (SOR) (Silverman, 1985; Wahba et al., 1998). Further, the $\mathcal{O}(NM^2)$ scaling of the DTC enabled datasets containing tens of thousands of data points to be efficiently modelled.

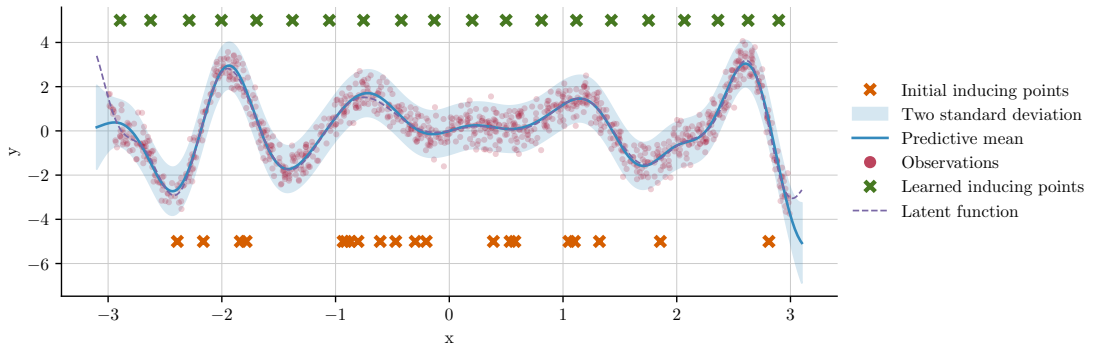


Figure 1.12: Illustration of a sparse Gaussian process being fit to data simulated from the function $f(x) = \sin(2x^2) + x\cos(5x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.5^2)$. The inducing points are randomly initialised and then optimised with respect to evidence lower bound.

Fully Independent Training Conditional The Fully Independent Training Conditional (FITC) approach was introduced by Snelson et al. (2005) under the name Sparse Pseudo-input GP (SPGP). The first improvement given in the FITC approach when compared to DTC is the richer covariance structure that the

training conditional is equipped with

$$q(\mathbf{f} \mid \mathbf{u}) = \prod_{i=1}^{N} p(f_i \mid \mathbf{u}) \tag{1.75}$$

$$= \mathcal{N}(\mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{u}, \operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}))\,. \tag{1.76}$$

As with the DTC approach, an exact test conditional is used in FITC. However, unlike DTC, a non-deterministic relationship exists between $\mathbf{f}$ and $\mathbf{u}$ which allows for more complex relationships to be encoded within the coupling. The marginal log-likelihood in FITC is given by

$$\log p(\mathbf{y}) = \frac{N}{2}\log(2\pi) + \frac{1}{2}\log|\mathbf{Q_{ff}} + \mathbf{G}| + \frac{1}{2}\mathbf{y}^\top(\mathbf{Q_{ff}} + \mathbf{G})^{-1}\mathbf{y}\,, \tag{1.77}$$

where $\mathbf{G} = \operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2\mathbf{I}_n$. As with the DTC model, FITC offers $\mathcal{O}(NM^2)$ scaling making it amenable to modelling large datasets.

Contrary to all previous sparse methods, a type 2 maximum likelihood estimate, sometimes termed the evidence approximation (Bishop, 2006), of the model's hyperparameters can be jointly learned with the inducing points' position by optimising against Equation (1.77). This transcends previous works by relaxing the assumption that the inducing points are a subset of the dataset, and instead allowing the inducing points to exist anywhere in the dataset's support which, consequently, leads to better mean predictions from the model. However, fitting models with a FITC approximation severely underestimates the noise variance parameter, meaning that the posterior distribution *pinches* at locations where an inducing point is present (M. Bauer et al., 2016). This pathology stems from the $|\mathbf{Q_{ff}} + \mathbf{G}|$ term that can be interpreted as an input-dependent (heteroscedastic) noise term. Therefore, when we optimise the inducing points' position that give rise to $\mathbf{Q_{ff}}$, our objective function is implicitly minimising the noise variance. This

is disastrous for applications, such as Bayesian optimisation (BO), where we would like to reason and make decisions under the model's uncertainty.

1.3.5 Variational free energy

A solution to the issue of an underestimated noise variance term was proposed in the variational free energy (VFE) approximation (Titsias, 2009). Unlike previous sparse methods, the VFE approach works with the exact GP prior and builds a variational approximation to the posterior in which inference is tractable. This concept can be elegantly summarised through the following quote: "*Approximate the posterior, not the model*" (Hensman, 2020). In essence, once the model has been approximated, we can never hope to recover the exact posterior. However, if the model is kept intact and the posterior approximation is optimal, then we may be able to recover the exact posterior.

Despite its wide adoption, a full derivation of the VFE model is often omitted from technical reports. Due to its popularity and the fact it plays a fundamental role in all subsequent chapters of this thesis, we will now provide a full derivation of the VFE model.

Letting $\mathbf{u}$ be the evaluation of our GP at the inducing points i.e., $\mathbf{u} = f(\mathbf{Z})$, the VFE approach begins by augmenting the posterior distribution with the set of inducing variables

$$p(\mathbf{f}, \mathbf{u} \,|\, \mathbf{y}) = p(\mathbf{f} \,|\, \mathbf{u}, \mathbf{y}) p(\mathbf{u} \,|\, \mathbf{y}) \,. \tag{1.78}$$

Following the VI workflow of Section 1.2, a variational approximation to the true

posterior is then introduced with the form

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u}) \,, \tag{1.79}$$

such that q is a free-form variational distribution, meaning that we make no assumption surrounding the family of probability distributions $\mathcal{Q}$ that q belongs to. Starting from the definition of the ELBO given in Equation (1.14), we can substitute in the definition of $q(\mathbf{f}, \mathbf{u})$ to yield

$$\begin{aligned}
\mathcal{L}(q) &= \mathbb{E}_{q(\mathbf{f},\mathbf{u})} \left[\log p(\mathbf{f}, \mathbf{u}, \mathbf{y})\right] - \mathbb{E}_{q(\mathbf{f},\mathbf{u})} \left[\log q(\mathbf{f}, \mathbf{u})\right] && (1.80)\\
&= \mathbb{E}_{q(\mathbf{f},\mathbf{u})} \left[\log \frac{p(\mathbf{f}, \mathbf{u}, \mathbf{y})}{q(\mathbf{f}, \mathbf{u})}\right] && (1.81)\\
&= \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{f}, \mathbf{u}, \mathbf{y})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} && (1.82)\\
&= \int p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) \cancel{p(\mathbf{f} \mid \mathbf{u})} p(\mathbf{u})}{\cancel{p(\mathbf{f} \mid \mathbf{u})} q(\mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} && (1.83)\\
&= \int q(\mathbf{u}) \left(\int p(\mathbf{f} \mid \mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{u})}{q(\mathbf{u})} \mathrm{d}\mathbf{f} \right) \mathrm{d}\mathbf{u} && (1.84)\\
&= \int q(\mathbf{u}) \left(\underbrace{\int p(\mathbf{f} \mid \mathbf{u}) \log p(\mathbf{y} \mid \mathbf{f}) \mathrm{d}\mathbf{f}}_{\psi(\mathbf{u}, \mathbf{y})} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right) \mathrm{d}\mathbf{u} \,. && (1.85)
\end{aligned}$$

This formulation allows us, in what follows, to fully eliminate the expensive to compute training conditional $p(\mathbf{f} \mid \mathbf{u})$ from the ELBO in Equations (1.80)–(1.84). By assuming a Gaussian likelihood distribution in the model, we can rearrange the ELBO such that integration over $\mathbf{f}$ need only be done with respect to a simpler

quantity; a task that can be done analytically

$$\psi(\mathbf{u}, \mathbf{y}) = \int p(\mathbf{f} \mid \mathbf{u}) \log p(\mathbf{y} \mid \mathbf{f}) \mathrm{d}\mathbf{f} \tag{1.86}$$

$$= \int p(\mathbf{f} \mid \mathbf{u}) \left(-\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \operatorname{trace}\left(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{f}^\top + \mathbf{f}\mathbf{f}^\top\right)\right) \mathrm{d}\mathbf{f} \tag{1.87}$$

$$= \mathbb{E}_{p(\mathbf{f} \mid \mathbf{u})}\left[-\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \operatorname{trace}\left(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\mathbf{f}^\top + \mathbf{f}\mathbf{f}^\top\right)\right] \tag{1.88}$$

$$= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \operatorname{trace}\left(\mathbf{y}\mathbf{y}^\top - 2\mathbb{E}_{p(\mathbf{f} \mid \mathbf{u})}\left[\mathbf{y}\mathbf{f}^\top\right] + \mathbb{E}_{p(\mathbf{f} \mid \mathbf{u})}\left[\mathbf{f}\mathbf{f}^\top\right]\right) . \tag{1.89}$$

To arrive at Equation (1.89), we have first expressed $\log p(\mathbf{y} \mid \mathbf{f})$ in terms of its density. Within the expectation, we can see that $-\frac{n}{2}\log(2\pi\sigma^2)$ and $\frac{1}{2\sigma^2}$ are independent of $\mathbf{f}$, and can therefore be moved outside of the expectation. Finally, by the linearity of the trace operator, we can rewrite the expectation of the trace term as the trace of the expectation applied to each of the three constituent terms, noting that $\mathbf{y}\mathbf{y}^\top$ is independent of $p(\mathbf{f} \mid \mathbf{u})$. To compute the two expectation terms left in Equation (1.89), we can use the relationship between the auto-covariance and autocorrelation matrices

$$\operatorname{Cov}[\mathbf{f}, \mathbf{f}] = \mathbb{E}[\mathbf{f}\mathbf{f}^\top] - \mathbb{E}[\mathbf{f}]\mathbb{E}[\mathbf{f}]^\top \iff \mathbb{E}[\mathbf{f}\mathbf{f}^\top] = \operatorname{Cov}[\mathbf{f}, \mathbf{f}] + \mathbb{E}[\mathbf{f}]\mathbb{E}[\mathbf{f}]^\top . \tag{1.90}$$

By the definition of $p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}(\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}})$ from Equation (1.70), the conditional expectation of the GP prior can be written as

$$\mathbb{E}_{p(\mathbf{f} \mid \mathbf{u})}[\mathbf{f}] = \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u} \tag{1.91}$$

which we denote as $\boldsymbol{\alpha}$.

Through Equations (1.90)–(1.91), the pair of expectations in Equation (1.89) can

be expressed as

$$= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\operatorname{trace}\left(\mathbf{y}\mathbf{y}^\top - 2\mathbf{y}\boldsymbol{\alpha}^\top + \boldsymbol{\alpha}\boldsymbol{\alpha}^\top + \mathbf{K_{ff}} - \mathbf{Q_{ff}}\right) \tag{1.92}$$

$$= \log\mathcal{N}(\mathbf{y} \mid \boldsymbol{\alpha}, \sigma^2\mathbf{I}_n) - \frac{1}{2\sigma^2}\operatorname{trace}\left(\mathbf{K_{ff}} - \mathbf{Q_{ff}}\right) . \tag{1.93}$$

Observing that $\mathbf{K_{ff}}$ and $\mathbf{Q_{ff}}$ are independent of $\mathbf{u}$, we now substitute Equation (1.93) back into the ELBO of Equation (1.85) and simplify

$$\mathcal{L}(q) = \int q(\mathbf{u})\left(\log\mathcal{N}(\mathbf{y} \mid \boldsymbol{\alpha}, \sigma^2\mathbf{I}_n) - \frac{1}{2\sigma^2}\operatorname{trace}\left(\mathbf{K_{ff}} - \mathbf{Q_{ff}}\right) + \log\frac{p(\mathbf{u})}{q(\mathbf{u})}\right)\mathrm{d}\mathbf{u} \tag{1.94}$$

$$= \int q(\mathbf{u})\log\frac{\mathcal{N}(\mathbf{y} \mid \boldsymbol{\alpha}, \sigma^2\mathbf{I}_n)p(\mathbf{u})}{q(\mathbf{u})}\mathrm{d}\mathbf{u} - \frac{1}{2\sigma^2}\operatorname{trace}\left(\mathbf{K_{ff}} - \mathbf{Q_{ff}}\right) . \tag{1.95}$$

The ELBO that is now under consideration is void of the expensive to compute $p(\mathbf{f} \mid \mathbf{u})$ term. Instead, the cost in evaluating the ELBO is dominated by $\boldsymbol{\alpha}$, a quantity that scales linearly with the number of data points and quadratically in the number of inducing points i.e., $\mathcal{O}(NM^2)$. This scaling stems from the need to compute the matrix multiplication $\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}$. In practice, evaluating such terms is computationally tractable on modern computers for tens of thousands of data points with hundreds of inducing points. Beyond this, the memory requirements for computing $\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}$ surpass the capacity of most modern computers, although there are techniques to improve this scaling that we discuss in Section 1.3.6.

The starting point of this derivation was the ELBO term that was formed by lower bounding the marginal log-likelihood using Jensen's inequality. To identify the *optimal* distribution we would usually be required to compute a derivative of the ELBO with respect to the variational distribution, equate to zero and solve for q. However, it was explicitly specified in Equation (1.79) that q is a free-form distribution and therefore did not belong to a specific family of probability

distributions. As shown in Titsias (2009), we can reverse Jensen's inequality by moving the log outside of the integral. Consequently, the lower bound's inequality will become an equality that identifies the *optimal* variational distribution $q^\star(\mathbf{u})$ through the following

$$\mathcal{L}^\star(q) = \log \int \cancel{q(\mathbf{u})} \frac{\mathcal{N}(\mathbf{y} \,|\, \boldsymbol{\alpha}, \sigma^2 \mathbf{I}_n) p(\mathbf{u})}{\cancel{q(\mathbf{u})}} \mathrm{d}\mathbf{u} - \frac{1}{2\sigma^2} \operatorname{trace}\left(\mathbf{K_{ff}} - \mathbf{Q_{ff}}\right) \tag{1.96}$$

$$= \log \mathcal{N}(\mathbf{y} \,|\, \boldsymbol{\alpha}, \mathbf{Q_{ff}} + \sigma^2 \mathbf{I}_n) - \frac{1}{2\sigma^2} \operatorname{trace}\left(\mathbf{K_{ff}} - \mathbf{Q_{ff}}\right) , \tag{1.97}$$

noting that $\mathcal{L}^\star(q) \geq \mathcal{L}(q)$. From Equation (1.97), we can see that the optimal distribution $q^\star$ is a Gaussian distribution with the form

$$q^\star(\mathbf{u}) = \mathcal{N}(\mathbf{y} \,|\, \boldsymbol{\alpha}, \sigma^2 \mathbf{I}_n) p(\mathbf{u}) \tag{1.98}$$

$$= c \exp\left(\frac{1}{2} \mathbf{u}^\top \left(\mathbf{K_{uu}^{-1}} + \frac{1}{2\sigma^2} \mathbf{K_{uu}^{-1}} \mathbf{K_{uf}} \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}}\right) \mathbf{u} + \frac{1}{2\sigma^2} \mathbf{y}^\top \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{u}\right) , \tag{1.99}$$

where c is the constant that is used to scale the distribution. By completing the square, the quadratic form of Equation (1.99) is the Gaussian

$$q^\star(\mathbf{u}) = \mathcal{N}(\mathbf{u} \,|\, \sigma^{-2} \mathbf{K_{uu}} \boldsymbol{\Sigma}^{-1} \mathbf{K_{uf}} \mathbf{y}, \mathbf{K_{uu}} \boldsymbol{\Sigma}^{-1} \mathbf{K_{uu}}) , \tag{1.100}$$

where $\boldsymbol{\Sigma} = \mathbf{K_{uu}} + \sigma^{-2} \mathbf{K_{uf}} \mathbf{K_{fu}}$.

As with FITC, we can compute derivatives of Equation (1.97) with respect to the model's hyperparameters and the inducing points $\mathbf{Z}$ to learn a tighter bound. Unlike FITC though, the resulting predictions that come from this model do not underestimate the predictive variance (M. Bauer et al., 2016).

To make predictions using the variational posterior learned using the VFE model, we must compute $p(\mathbf{f}^\star \,|\, \mathbf{y})$. By again appealing to the conditional independence

assumptions made in Equation (1.69), we can write the predictive distribution as

$$p(\mathbf{f}^\star \,|\, \mathbf{y}) = \int p(\mathbf{f}^\star \,|\, \mathbf{u}) p(\mathbf{f} \,|\, \mathbf{u}) p(\mathbf{u} \,|\, \mathbf{y}) \mathrm{d}\mathbf{f} \mathrm{d}\mathbf{u} \tag{1.101}$$

$$= \int p(\mathbf{f}^\star \,|\, \mathbf{u}) p(\mathbf{u} \,|\, \mathbf{y}) \mathrm{d}\mathbf{u} \int p(\mathbf{f} \,|\, \mathbf{u}) \mathrm{d}\mathbf{f} \tag{1.102}$$

$$= \int p(\mathbf{f}^\star \,|\, \mathbf{u}) p(\mathbf{u} \,|\, \mathbf{y}) \mathrm{d}\mathbf{u} \tag{1.103}$$

$$\approx \int p(\mathbf{f}^\star \,|\, \mathbf{u}) q(\mathbf{u}) \mathrm{d}\mathbf{u} \tag{1.104}$$

$$= \mathcal{N}(\mathbf{f}^\star \,|\, m_q(\cdot), k_q(\cdot, \cdot)) \,. \tag{1.105}$$

where

$$m_q(\mathbf{x}^\star) = \mathbf{K}_{*\mathbf{u}} \mathbf{K}_{\mathbf{uu}}^{-1} \boldsymbol{\mu}_{\boldsymbol{q}} \tag{1.106}$$

$$k_q(\mathbf{x}^\star, \mathbf{x}^{\star\prime}) = k(\mathbf{x}^\star, \mathbf{x}^{\star\prime}) - \mathbf{K}_{*\mathbf{u}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{K}_{\mathbf{u}*\prime} + \mathbf{K}_{*\mathbf{u}} \mathbf{K}_{\mathbf{uu}}^{-1} \boldsymbol{\Sigma}_{\boldsymbol{q}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{K}_{\mathbf{u}*\prime} \tag{1.107}$$

where $\boldsymbol{\mu}_{\boldsymbol{q}}$ is the variational distribution's mean and $\boldsymbol{\Sigma}_{\boldsymbol{q}}$ is the variational distribution's covariance, each computed with respect to the training data.

The computational complexity of the VFE model is $\mathcal{O}(NM^2)$ and, despite being over 13 years old now, it is still widely used in the literature due to its efficiency and well-understood theoretical foundations. In Burt et al. (2020), rates of convergence were derived for the VFE model with respect to the number of inducing points. A scheme for sequentially updating the variational posterior distribution as new data is received in an online setting was proposed in Bui et al. (2017). Finally, the VFE model has been used in the Bayesian Gaussian process latent variable model (GPLVM) to perform unsupervised learning of latent spaces (Titsias et al., 2010), a model that we extend in Chapter 4.

1.3.6 Accelerating sparse Gaussian processes

The elegance of the VFE approach is the unrestricted form of q that allows its optimal form to be identified analytically. However, the method's complexity is still dependent on the full dataset's size, meaning that for datasets containing hundreds of thousands of observations, the method will become intractable. To resolve this, Hensman et al. (2013a) developed an alternative ELBO that is amenable to mini-batching, thus replacing the method's dependence on the full dataset's size with a mini-batch size term.

We assume that the variational family $\mathcal{Q}$ is the set of multivariate Gaussian distributions and any variational distribution q will, therefore, be a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \,. \tag{1.108}$$

By assuming a Gaussian likelihood function, letting y_n and f_n be the n^{th} entries of $\mathbf{y}$ and $\mathbf{f}$ respectively, and applying the variational framework of Section 1.2, we can obtain the ELBO

$$\begin{aligned}\mathcal{L}(q) = \sum_{n=1}^{N} \left[\log \mathcal{N}(y_n \,|\, \mathbf{K}_{f_n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\boldsymbol{\mu}, \sigma^2) - \frac{1}{2}\left(\operatorname{trace}(\boldsymbol{\Sigma}\mathbf{A}) - \sigma^{-2}\mathbf{B}_{n,n}\right)\right] \\ - \operatorname{KL}(q(\mathbf{u}) \,||\, p(\mathbf{u}))\,,\end{aligned} \tag{1.109}$$

where

$$\mathbf{A} = \sigma^{-2}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{f_n\mathbf{u}}\mathbf{K}_{\mathbf{u}f_n}\mathbf{K}_{\mathbf{uu}}^{-1} \tag{1.110}$$

$$\mathbf{B} = \mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}\,. \tag{1.111}$$

A fundamental observation that enables tractable computations of Equation (1.109)

is that the $q(\mathbf{u})$ and $p(\mathbf{u})$ are both multivariate Gaussian distributions and the KLD can, therefore, be computed analytically using Definition 1.3.5.

Unlike the bound given in the VFE model, we have no closed form expression for the optimal variational distribution. When performing inference in the model, we must therefore learn the variational distribution's M-dimensional mean vector and $M \times M$ covariance matrix alongside the model's hyperparameters. To accelerate this, natural gradients (Amari, 1998; Martens, 2020) have been widely employed with great success for optimising the variational parameters of a sparse GP, whilst first-order methods such as Adam (Ba et al., 2015) are used for the model's hyperparameters (Salimbeni et al., 2018).

The bound given in Equation (1.109) assumes a Gaussian likelihood function. However, a further advantage of this bound in comparison to the VFE model is that non-Gaussian likelihoods can be used. To achieve this, we define

$$q(\mathbf{f}) = \int q(\mathbf{f}, \mathbf{u}) \mathrm{d}\mathbf{u} \tag{1.112}$$

$$= \int p(\mathbf{f} \,|\, \mathbf{u}) q(\mathbf{u}) \mathrm{d}\mathbf{u} \,. \tag{1.113}$$

Under the assumption now that our likelihood factorises over the N data points, we can redefine the original ELBO from Equation (1.15) as

$$\mathcal{L}(q) = \sum_{n=1}^{N} \mathbb{E}_{q(f_n)} \left[\log p(y_n \,|\, \phi(f_n)] - \mathrm{KL}(q(\mathbf{u}) \,||\, p(\mathbf{u})) \right., \tag{1.114}$$

where $\phi(\cdot)$ is the likelihood distribution's link function. The expectation in Equation (1.114) requires solving a 1-dimensional integral that is unavailable analytically, but can be solved using quadrature (Hensman et al., 2015b). Despite its flexibility to handle non-Gaussian likelihoods, the bound in Equation (1.114) is a slacker bound on the marginal log-likelihood when compared to Equation (1.109).

It is therefore advisable to use Equation (1.109) when the likelihood is Gaussian, but the number of data points is insurmountable for the VFE approach.

Modern extensions to the bounds given in Equation (1.109) and Equation (1.114) have considered *inter-domain* inducing variables (Lázaro-Gredilla et al., 2009). An inter-domain approach relaxes the assumption that the inducing points $\mathbf{Z}$ at which we evaluate our GP f should exist within the domain of our observed data inputs $\mathbf{X}$. Instead, we can define the m^{th} inducing variable $\mathbf{u}_m = \int f(\mathbf{X})\xi_m(\mathbf{X})\mathrm{d}\mathbf{X}$ where ξ_m is a real-valued function that we refer to as an inducing feature. The downstream inference then works in an identical manner to the approaches given in Sections 1.3.5–1.3.6. This can be seen as a generalisation of the above sparse approaches as letting $\xi_m = \delta_{\mathbf{z}_m}(\cdot)$ where $\delta_{\mathbf{z}_m}(\cdot)$ is the Dirac delta function that assigns all the probability mass to $\mathbf{z}_m$ yields a regular sparse GP. The works of Lázaro-Gredilla et al. (2010) and Hensman et al. (2018) consider transformations of the form $\xi_m(\cdot) = \exp(-i\omega_m^\top \cdot)$ where i is the complex unit and ω_m is a harmonic frequency vector. Meanwhile, for certain kernel functions, applying the transformation $\xi_m(\cdot) = v_m(\cdot)$ where $v_m(\cdot)$ corresponds to the kernel function's m^{th} eigenfunction can lead to sparse GPs that scale linearly in the number of inducing points due to the resulting diagonal form of $\mathbf{K}_{\mathbf{uu}}$ (Dutordoir et al., 2020).

1.4 Thesis structure

This thesis has focussed on developing GP methodology for the analysis of climate and network data. Contributions include: a demonstration of how SVGD can be used as an alternative to VI or MCMC for inference in a range of GP models; pollution modelling on a network using GP; the development of a GPLVM to embed hypergraphs into a latent space; a probabilistic ensemble of GP models using Wasserstein barycentres; and the development of an efficient GP software

package. A brief outline of the material presented in this thesis is given below.

- Chapter 2 introduces Stein's method and SVGD. We demonstrate how SVGD can be efficiently used to perform inference in a range of GP models. Theoretical results are then derived to demonstrate the asymptotic convergence guarantees that can be provided by SVGD in a latent Gaussian model. The efficacy of SVGD for GP models is demonstrated on a range of benchmark regression and classification datasets, a challenging multi-modal dataset, and a spatiotemporal air quality modelling problem that explores the change in pollution across the time window that the United Kingdom entered a Covid-19 lockdown. This chapter is based on Pinder et al. (2020).

- Chapter 3 introduces graph kernels and how they can be used to parameterise a GP whose support is the vertex set of a graph. We extend this model to accommodate a heteroscedastic noise component. The GP model is then used to model air pollution in London where fine-scale predictions of air pollution along an individual's journey can be made. The chapter concludes by describing the collection of a new air quality dataset where a series of pollutants were measured in a mobile manner every second for a week in Lancaster. This chapter is based on Pinder et al. (2022d) and Pinder et al. (2022a).

- Chapter 4 introduces hypergraphs and the range of datasets that they can be used to model. We construct a new GPLVM that embeds the vertices of a hypergraph into a latent space. The latent space is then used for downstream modelling or easier visualisation of the hypergraph's relational structure. We demonstrate this model on a series of political networks. This chapter is based on Pinder et al. (2021).

- Chapter 5 introduces `GPJax`; a GP package written in the JAX framework that offers performant computational abstractions of common GP operations. We empirically benchmark `GPJax` against `GPFlow` and `GPyTorch`. This chapter is based on Pinder et al. (2022c).

- Chapters 6–7 conclude the thesis by first outlining some ongoing work in Chapter 6 that presents a probabilistic ensemble model with applications to global surface temperature projections. The framework first emulates the output of climate models using a novel sparse hierarchical GP. Proper scoring rules are then used to rank each climate model's emulator based on how well it represents *real-world* observations. The rankings are then used to assign weights to each emulator within the final ensemble; a quantity that is identified using ideas from the optimal transport literature. The thesis is concluded in Chapter 7 by consolidating the work presented in this thesis and outlining some potential next steps.

Chapter 2

Stein Variational Gaussian Processes

Whilst elegant in its formulation, the VFE approach of (Titsias, 2009) is only compatible with models where the likelihood function is a Gaussian distribution. The stochastic extension of (Hensman et al., 2013a) is amenable to any factorising likelihood, however, unlike the VFE model, there is no guarantee that the variational family is the set of multivariate Gaussian distributions. Further, due to the variational scheme used in both approaches, specifying prior distributions and learning a posterior distribution over the model's hyperparameters is not possible. In this chapter, we seek to resolve this by connecting SVGD with GPs to propose a SteinGP model. With this model, we enable a fully Bayesian treatment of the model's hyperparameters whilst retaining the computational efficiency of variational approaches. The contents of this chapter are adapted from (Pinder et al., 2020) where background details on general GP models are omitted in favour of the exposition given in Section 1.3.

Table 2.1: Features of key inference methods for Gaussian process models.

Reference	$p(\mathbf{y}\|\mathbf{f})$	Sparse	Approx. posterior	Hyperparams	Inference
Opper et al. (2008)	Binary	✗	Gaussian	Point estimate	Variational
Titsias (2009)	Gaussian	✓	Gaussian	Point estimate	Variational
Nguyen et al. (2014)	Any	✗	Gaussian mixture	Point estimate	Variational
Hensman et al. (2015a)	Any	✓	True posterior	Marginalised	MCMC
This work	Any	✓	True posterior	Marginalised	SVGD

2.1 Introduction

GPs are highly expressive, non-parametric distributions over continuous functions and are frequently employed in both regression and classification tasks (Rasmussen et al., 2006). In recent years, GPs have received significant attention in the machine learning community due to their successes in domains such as reinforcement learning (M. P. Deisenroth et al., 2015), variance reduction (Oates et al., 2017), and optimisation (Mockus, 2012). This recent blossoming has been facilitated by advances in inference methods, and especially by VI which provides a tractable approach to fitting GP models to large and/or non-Gaussian data sets (e.g. Hensman et al., 2013a; Cheng et al., 2017).

Whilst computationally efficient, VI typically relies on the practitioner placing a parametric constraint upon the approximating posterior distribution. Unfortunately, this assumption can often severely inhibit the quality of the approximate posterior should the true posterior not belong to the chosen family of probability distributions, as often happens with GPs (Havasi et al., 2018). The most common (asymptotically) exact inference method for GPs is MCMC. However, sampling may be problematic if the posterior distribution is non-convex as the sampler can become trapped in local modes (Rudoy et al., 2006). Additionally, MCMC does not enjoy the same computational scalability as VI, and for this reason, it is impractical for modelling problems with a large number of observations.

In this work we propose the use of SVGD (Liu et al., 2016b), a non-parametric VI approach, as an effective inference method for GP models. SVGD can be thought of as a particle-based approach, whereby particles are sequentially transformed until they become samples from an arbitrary variational distribution that closely approximates the posterior of interest. SVGD can be considered a hybrid of VI and Monte Carlo approaches, yielding benefits over both. The first benefit is removing the parametric assumption used in VI. The result of this is that inference through SVGD allows a richer variational distribution to be learned. A second benefit is that we do not need to compute the acceptance step required in MCMC. This leads to greater efficiency in SVGD as we are only required to compute the score function, an operation that can be accelerated for big datasets using the subsampling trick in Section 2.3. Finally, through the use of a kernel function acting over the set of particles, SVGD encourages full exploration over the posterior space, meaning that we can better represent the uncertainty in multimodal posteriors. This is a critical difference in the quality of inference that is possible through SVGD in comparison to alternative methods and we provide compelling empirical evidence to support this in Section 2.4.2. Table 2.1 shows the position of this work within the current literature. It should be acknowledged that SVGD is not without its shortcomings. Unlike MCMC, there is no metric to assess the quality of the posterior approximation. Further, the performance of SVGD is dependent on the number of particles being sufficient to represent the posterior. As we shall go on to see in Section 2.4.2, for posterior distributions where the geometry is non-convex, selecting an appropriate number of particles is a non-trivial task.

Our article demonstrates how to use SVGD to fit GPs to both Gaussian and non-Gaussian data, including when computational scalability is addressed through an inducing point representation of the original data. We prove that the

SVGD scheme reduces the KLD to the target distribution on each iteration. We empirically demonstrate the performance of SVGD in a range of both classification and regression datasets, comparing against traditional VI and modern implementations of MCMC for GPs, including in a large-scale spatiotemporal model for air quality in the UK. We acknowledge that since the release of this work several orthogonal lines of work have approached this problem from different perspectives (Hamelijnck et al., 2021; Tebbutt et al., 2021). We release, at HTTPS://GITHUB.COM/THOMASPINDER/STEINGP, code for reproducing the experiments in Section 2.4, and a general library for fitting GPs using SVGD based entirely upon GPFlow (Matthews et al., 2017) and TensorFlow (Abadi et al., 2016). For a demonstration of the package see Section 2.4.4.

2.2 Stein's method in machine learning

The core of this chapter is centred around Stein's identity (Stein, 1972). As we shall go on to see, in its original form, Stein's identity is intractable for most machine learning models. However, by building on recent developments in the statistics and machine learning literature, reproducing kernel Hilbert space (RKHS) theory can be used to derive an analytical solution to Stein's identity for a wide range of machine learning models. RKHS is intimately linked to kernel methods, and we, therefore, commence this chapter by building upon the definition of a kernel function that was given in Definition 1.3.6 to establish some key results from the RKHS literature. This provides a natural foundation for understanding the subsequent work that considers how Stein's identity (Stein, 1972) can be used to perform inference with a GP. For a detailed review of RKHS theory, see the seminal work of Aronszajn (1950) or the textbooks of Berlinet et al. (2004) and Steinwart et al. (2008).

2.2.1 Reproducing kernel Hilbert spaces

A Hilbert space $\mathcal{H}$ is a complete inner product space that allows us to generalise from Euclidean spaces to infinite dimensional spaces. Examples of a Hilbert space include

1. The space $\mathbb{R}^d$ is a Hilbert space over $\mathbb{R}$ with inner product $\langle x, y\rangle = \sum_{i=1}^{d} = x_i y_i$.
2. The space $\mathbb{C}^d$ is a Hilbert space over $\mathbb{C}$ with inner product $\langle x, y\rangle = \sum_{i=1}^{d} x_i \overline{y_i}$.
3. The space $L_2[a, b] = \{f : [a, b] \to \mathbb{R}, \text{ where } \int_a^b |f(t)|^2 \mathrm{d}t < \infty\}$ is a Hilbert space over $\mathbb{R}$ with inner product $\langle f, g\rangle = \int_a^b f(t)\overline{g(t)}\mathrm{d}t$.

We now proceed to lift this concept to a type of a Hilbert space known as a reproducing kernel Hilbert space (RKHS). To achieve this, we first define a *reproducing kernel function* before showing that a Hilbert space endowed with a reproducing kernel yields a unique RKHS.

Definition 2.2.1 (Reproducing kernels) *For a non-empty set $\mathcal{X}$, let $\mathcal{H}$ be a Hilbert space defined on $\mathcal{X}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. A symmetric positive-definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a reproducing kernel of $\mathcal{H}$ if and only if the following two properties are satisfied:*

1. *For all $x \in \mathcal{X}$, $k(\cdot, x) \in \mathcal{H}$.*
2. *For all $x \in \mathcal{X}$, and for all $h \in \mathcal{H}$*

$$\langle h(\cdot), k(\cdot, x)\rangle_{\mathcal{H}} = h(x)\,. \tag{2.1}$$

Equation (2.1) is known as the *reproducing property* and it tells us that the result of $h(x)$ is *reproduced* by computing the inner product of h with the kernel function

$k(x, \cdot)$. It is important here to acknowledge that any reproducing kernel is a kernel function, as presented in Definition 1.3.6. The feature map $\Phi : \mathcal{X} \to \mathcal{H}$ of a reproducing kernel k that endows the Hilbert space $\mathcal{H}$ defined over $\mathcal{X}$ is given by

$$\Phi(x) = k(\cdot, x) \tag{2.2}$$

for $x \in \mathcal{X}$ (Steinwart et al., 2008, Lemma 4.19). In such circumstances, Φ is called the *canonical feature map*.

Definition 2.2.2 (Evaluation functional) *Let $\mathcal{X}$ be a non-empty set and $\mathcal{H}_k$ a Hilbert space of functions over $\mathcal{X}$. A linear evaluation functional δ_x evaluates each h at x:*

$$\delta_x[h] = h(x)\,, \tag{2.3}$$

for all $h \in \mathcal{H}$.

Equipped with the concept of an evaluation functional and a reproducing kernel, we can now define a RKHS.

Proposition 2.2.3 (Reproducing kernel Hilbert space) *For a non-empty set $\mathcal{X}$, the Hilbert space $\mathcal{H}$ of functions over $\mathcal{X}$ is an RKHS if the evaluation functional δ_x is continuous for every $x \in \mathcal{X}$ (Steinwart et al., 2008).*

The correspondence between a kernel function k and the RKHS $\mathcal{H}_k$ that is associated to k is a unique one-to-one correspondence (Berlinet et al., 2004, Theorem 3). This is often referred to as the Moore-Aronszajn theorem. We notationally link the kernel and its associated RKHS by subscribing the RKHS with the kernel that endows the space.

2.2.2 Stein's discrepancy

The foundation for SVGD is the *Stein operator* (Stein, 1972) that we now introduce in $\mathbb{R}^d$ following Oates et al. (2017); Chwialkowski et al. (2016). For a continuously differentiable density p with support on $\mathbb{R}^d$ that vanishes at infinity, let $\mathcal{X} \subset \mathbb{R}^d$ and $\phi_i : \mathcal{X} \to \mathbb{R}^d$ for $i = 1, 2, \ldots, d$ be scalar functions on $\mathcal{X}$. Further, let $\boldsymbol{\phi}(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_d(x)]^\top \in \mathbb{R}^d$ define a bounded smooth vector-valued function $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^d$. The Stein operator is then given by

$$\mathcal{A}_p \boldsymbol{\phi}(x) = \sum_{i=1}^{d} \left(\phi_i(x) \nabla_{x_i} \log p(x) + \nabla_{x_i} \phi_i(x) \right) , \tag{2.4}$$

where ∇_{x_i} denotes the partial derivative operator with respect to x_i. We call $\mathcal{A}_p$ a Stein operator if *Stein's identity* holds:

$$\mathbb{E}_p \left[\mathcal{A}_p \boldsymbol{\phi} \right] = \mathbf{0} , \tag{2.5}$$

for all ϕ in an appropriate family of functions $\mathcal{F}$, a set sometimes referred to as the *Stein set* (Gorham et al., 2020). We can validate that the operator in Equation (2.5) is a valid Stein operator through integration by parts

$$\begin{aligned} \mathbb{E}_p \left[\mathcal{A}_p \boldsymbol{\phi} \right] &= \int_{\mathbb{R}^d} \mathcal{A}_p \boldsymbol{\phi}(x)\, p(x) \mathrm{d}x & (2.6) \\ &= \sum_{i=1}^{d} \int_{\mathbb{R}^d} \nabla_{x_i} \left(\phi_i(x) p(x) \right) \mathrm{d}x & (2.7) \\ &= 0 , & (2.8) \end{aligned}$$

by virtue of the fact that p vanishes at infinity, giving

$$\lim_{x \to \pm\infty} \phi_i(x) p(x) = 0 , \tag{2.9}$$

for $i = 1, 2, \ldots, d$. The function $\boldsymbol{\phi}$ is referred to as a *test function*.

Equation (2.5) was originally designed for distributional comparisons against a Gaussian distribution. Subsequently, the result was extended for Poisson distributions (L. H. Chen, 1975). More recently, Stein's identity has found uses in a range of modern machine learning problems including variance reduction (Oates et al., 2017), model selection (Kanagawa et al., 2019) and generative modelling (Pu et al., 2017). In our application, p will be the posterior density of interest.

When the density p under which we evaluate the expectation in Equation (2.5) is replaced with a second density q, Equation (2.5) holds if and only if $p = q$. For a test function $\boldsymbol{\phi}$, we can write this as

$$\mathbb{E}_q \left[\mathcal{A}_p \boldsymbol{\phi}(x)\right] = \mathbb{E}_q \left[\mathcal{A}_p \boldsymbol{\phi}(x)\right] - \mathbb{E}_q \left[\mathcal{A}_q \boldsymbol{\phi}(x)\right] \tag{2.10}$$

$$= \mathbb{E}_q \left[\boldsymbol{\phi}(x)^\top \left(\nabla_x \log p(x) - \nabla_x \log q(x)\right)\right] , \tag{2.11}$$

assuming that Equation (2.5) is satisfied for q. Observing Equation (2.11), it can be seen that the result may only be 0 if $\nabla_x \log p(x) = \nabla_x \log q(x)$, that is, the score functions of p and q equate to one another. This observation also unveils another key property of Stein's identity: we only require access to the score function of a density. In the context of Bayesian inference where p may be our unnormalised posterior distribution, this property is immensely useful as evaluating Equation (2.11) bypasses the need to evaluate the often intractable marginal log-likelihood.

Given the dependence of Equation (2.11) on the test function $\boldsymbol{\phi}$, it is more informative to consider the test function $\boldsymbol{\phi} \in \mathcal{F}$ that leads to the *largest violation of Stein's identity*. This is Stein's discrepancy. Formally, we posit Stein's discrepancy

as

$$\mathbb{D}(p, q) = \sup_{\phi \in \mathcal{F}} \mathbb{E}_q \left[\mathcal{A}_p \phi \right] . \tag{2.12}$$

where $\mathcal{F}$ is the family of functions which we will optimise over. Choosing $\mathcal{F}$ is challenging as we are faced with the following dichotomy: 1) $\mathcal{F}$ must be rich enough to contain the test function that truly leads to the largest violation of Stein's identity and 2) $\mathcal{F}$ must be small enough to enable tractable optimisation within Equation (2.12). To resolve this, we can appeal to the RKHS theory that was presented in Section 2.2.1 to obtain a closed form solution to Equation (2.12).

As shown in Liu et al. (2016a), letting $\mathcal{H}_k$ be the RKHS endowed with the kernel function k, we can define $\mathcal{F}$ to be the unit ball of $\mathcal{H}_k$, i.e., $\mathcal{F} = \{\phi \in \mathcal{H}_k : \|\phi\|_{\mathcal{H}_k} \leq 1\}$. Consequently, the space $\mathcal{F}$ contains an infinite number of basis functions. The test function $\phi \in \mathcal{H}_k$ that solves the optimisation problem in Equation (2.12) takes the closed form solution

$$\hat{\phi}(\cdot) = \frac{\beta_{p,q}(\cdot)}{\|\beta_{p,q}(\cdot)\|_{\mathcal{H}_k}} , \tag{2.13}$$

where $\beta_{p,q} = \mathbb{E}_{x \sim q} \left[\mathcal{A}_p k(x, \cdot) \right]$ and $\hat{\phi}$ denotes the maximising test function.

Unlike the KLD function given in Definition 1.2.1, kernel Stein discrepancy (KSD) is a symmetric divergence measure (Chwialkowski et al., 2016, Theorem 2.2). We shall now proceed to see how KSD can be used within a Bayesian inference framework.

2.2.3 Stein Variational Gradient Descent

In SVGD, as in classical VI, we approximate the true posterior distribution p with a variational distribution q that minimises the KLD between p and q. The innovation

in SVGD is that we assume no parametric form for q. From an arbitrary initial distribution q_0, SVGD iterates through a series of pushforward transformations that reduce the KLD between the target distribution and the distribution q_t after t iterations.

In particular, we consider the transformation that is defined by the mapping[1] $\mathcal{T}(\boldsymbol{\lambda}) = \boldsymbol{\lambda} + \epsilon\boldsymbol{\phi}(\boldsymbol{\lambda})$ for an arbitrary function ϕ and perturbation magnitude ϵ, where $\boldsymbol{\lambda} \sim q_t$. The role of ϕ in this update is to define the velocity field over which the distribution's shape will be transformed. The transformed distribution q_{t+1} is the distribution of $\mathcal{T}(\boldsymbol{\lambda})$. Following Gorham et al. (2015) and Liu et al. (2016a), we assume ϕ lives in the RKHS $\mathcal{H}^d$. Under this assumption, it can be shown that

$$\nabla_\epsilon \operatorname{KL}\left(q_{t+1} \| p\right)\big|_{\epsilon=0} = -\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\operatorname{trace}\left(\mathcal{A}_p\boldsymbol{\phi}(\boldsymbol{\lambda})\right)\right]. \tag{2.14}$$

Comparing Equation (2.14) and Equation (2.12) we see that letting $\boldsymbol{\phi}$ be equals to $\hat{\boldsymbol{\phi}}$ from Equation (2.13) maximally decreases the KLD. If we consider the task of Bayesian inference from the perspective of a traditional VI framework, the significance of this should not be understated, as our fundamental goal is to minimise $\operatorname{KL}(q \,||\, p)$ with respect to q[2]. By the result given in Equation (2.14), we can see that moving our approximating distribution q in the direction given by Equation (2.13), we can guarantee that we are taking steps in the optimal direction.

To implement the recursion, we maintain a finite set of J samples that empirically represent q, referred to as *particles*. These particles $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}^j\}_{j=1}^J$ are initially sampled independently from q_0, which is typically taken to be the prior distribution.[3]

[1] Note, other transformations such as the preconditioned update given in Detommaso et al. (2018) are also possible.

[2] An illuminating orthogonal line of work considers using metrics other than the KLD (Y. Li et al., 2016; Knoblauch et al., 2022)

[3] In the asymptotic limit, the final particle values are invariant to the initial distribution that

The transformation $\mathcal{T}$ is then applied repeatedly to the set of particles, where at each stage the optimal ϕ from Equation (2.13) is estimated empirically using the particles $\boldsymbol{\Lambda}_t = \{\boldsymbol{\lambda}_t^j\}_{j=1}^J$ at the t^{th} iteration:

$$\hat{\phi}_{\Lambda_t}(\boldsymbol{\lambda}) = \frac{1}{J}\sum_{j=1}^{J}\Big[\underbrace{\kappa(\boldsymbol{\lambda}_t^j, \boldsymbol{\lambda})\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{\lambda}_t^j)}_{\text{Attraction}} + \underbrace{\nabla_{\boldsymbol{\lambda}}\kappa(\boldsymbol{\lambda}_t^j, \boldsymbol{\lambda})}_{\text{Repulsion}}\Big]. \tag{2.15}$$

When the process terminates after T iterations, each of the particles is a sample from a distribution q_T with low KLD from the target p, and we can use the particles in the same way as a standard Monte Carlo sample.

Examining the update step in Equation (2.15), it can be seen that the first term transports particles towards areas in the posterior distribution that represent high probability mass. Conversely, the second term is the derivative of the kernel function; a term that will inflate when the particles are close to one another. To demonstrate this through an example, we take $\kappa(\cdot,\cdot)$ to be an isotropic[4] RBF kernel (a valid kernel when computing KSD (Gorham et al., 2015)), then we have

$$\kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda}') = \exp\left(\frac{\|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|^2}{-\ell^2}\right) \tag{2.16}$$

whereby

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}}\kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda}') &= -\frac{2\boldsymbol{\lambda} - 2\boldsymbol{\lambda}'}{\ell^2}\exp\left(\frac{\|\boldsymbol{\lambda} - \boldsymbol{\lambda}'\|^2}{-\ell^2}\right) \\ &= -\frac{2(\boldsymbol{\lambda} - \boldsymbol{\lambda}')}{\ell^2}\kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda}').\end{aligned} \tag{2.17}$$

Should particles be densely clustered, then the resultant Gram matrix will be dense. This will lead to a larger quantity being computed upon evaluation of

particles are initialised from (Papamakarios et al., 2019)

[4] An anisotropic form is valid and, whilst more computationally demanding, will allow high-dimensional particles to move independent in each dimension.

Equation (2.17), compared to when particles are sufficiently far from one another.

In the case that $J = 1$, the summation in Equation (2.15) disappears, and the entire scheme reduces to regular gradient-based optimisation. Additionally, there is no danger of running SVGD with J too large as, by the *propagation of chaos* (Kac, 1976), the final distribution of the i^{th} particle is invariant to J as the number of iteration steps $T \to \infty$ (Liu et al., 2016b).

2.2.4 Connection to variational inference

Typically, in VI (Section 1.2) we minimise the KLD between a ξ-parameterised variational distribution $q_\xi(\boldsymbol{\lambda})$ and the target density:

$$\xi^\star = \arg\min_\xi \mathrm{KL}(q_\xi(\boldsymbol{\lambda}) || p(\boldsymbol{\lambda})). \tag{2.18}$$

ξ often parameterises a family $\{q_\xi\}$ of Gaussian distributions. The resultant parameters $\xi^\star$ are then used to form the optimal variational distribution $q^\star_\xi(\boldsymbol{\lambda})$, used in place of the intractable $p(\boldsymbol{\lambda})$.

In a regular VI framework, the explicit form placed on q can be highly restrictive, particularly if the true posterior density is not well approximated by the variational family selected. Conversely, SVGD allows for a non-parametric representation of the posterior distribution to be learned that that enriches the choice of variational family beyond the Gaussian distribution that is commonly used in regular VI. An additional advantage is that SVGD only requires evaluation of the posterior's score function, a quantity that is invariant to the normalisation constant and can be unbiasedly approximated in large data settings.

2.2.5 Related works

SVGD has been used in the context of fitting Bayesian logistic regression and Bayesian neural networks (Liu, 2017). Further, SVGD was used in the context of a variational autoencoder (VAE) to model the latent space (Pu et al., 2017). By relaxing the Gaussian assumption that is typically made of the latent space, it was possible to learn a more complex distribution over the latent space of the VAE. Further examples of the applications of SVGD can be found in reinforcement learning (Liu, 2017), Bayesian optimisation (Gong et al., 2019), and in conjunction with deep learning (Grathwohl et al., 2020). Theoretical analysis has also established connections between SVGD and the overdamped Langevin diffusion (Duncan et al., 2019), and black-box VI (Chu et al., 2020).

This article leverages the effective and efficient SVGD optimisation framework to address the computational and multi-modality challenges endemic in GP inference. We show that compared to standard inference approaches for GPs, the SVGD framework offers the best trade-off between accuracy, computational efficiency and model flexibility.

2.3 Stein variational Gaussian processes

2.3.1 Conjugate models

Our goal is to establish an inference scheme that allows us to use SVGD to infer the posterior distribution $p(\mathbf{f} \,|\, \mathbf{y}, \boldsymbol{\theta})$ of a GP in a manner that is robust to model mispecification. We deviate slightly from the notion that has been established in Section 1.3 by conditioning on $\boldsymbol{\theta}$. This is because our goal in the forthcoming sections of this chapter is to place prior distributions on the components of $\boldsymbol{\theta}$ and, consequently, their notation becomes non-trivial. When the data likelihood

Algorithm 1 Pseudocode for fitting a Gaussian process using T iterations of SVGD.

Require: Base distribution q_0. Target distribution $p(\boldsymbol{\lambda} \,|\, \mathbf{y})$ where $\boldsymbol{\lambda} = \{\boldsymbol{\theta}, \boldsymbol{\nu}\}$.
  Create $\boldsymbol{\Lambda}_0 = \{\boldsymbol{\lambda}_0^j\}_{j=1}^J$ where $\boldsymbol{\lambda}_0^j \overset{\text{iid}}{\sim} q_0$.
  for t in 1:T **do**
    for j in 1:J **do**
      $\boldsymbol{\lambda}_t^j \leftarrow \boldsymbol{\lambda}_{t-1}^j + \epsilon \hat{\phi}_{\boldsymbol{\Lambda}_{t-1}}(\boldsymbol{\lambda}_{t-1}^j)$ (see Equation (2.15))
    end for
    $\boldsymbol{\Lambda}_t = \{\boldsymbol{\lambda}_t^j\}_{j=1}^J$
  end for
  return $\boldsymbol{\Lambda}_T$

Algorithm 2 Pseudocode for predictive inference over test inputs $X^\star$.

Require: Set of particles $\{\boldsymbol{\lambda}_j\}_{j=1}^J$
  Initialise `sample`= {}
  for j in 1:J **do**
    Set $(\boldsymbol{\theta}, \boldsymbol{\nu}) = \boldsymbol{\lambda}^j$
    for k in 1:K **do**
      Sample $\mathbf{y}^* \sim p(\cdot \,|\, \boldsymbol{\theta}, \boldsymbol{\nu})$
      Append $\mathbf{y}^*$ to `sample`
    end for
  end for
  return `mean(sample), var(sample)`

is Gaussian, we can analytically integrate out f and use SVGD to learn the kernel hyperparameters and observation noise σ^2, which comprise $\boldsymbol{\theta}$. This means that each particle $\boldsymbol{\lambda}^j$ represents a sampled $\boldsymbol{\theta}$ value. SVGD is useful even in these cases, as estimating kernel parameters such as the Automatic Relevance Determination (ARD)lengthscale vector can be particularly challenging for VI and MCMC. This is due to their unidentifiable nature that often manifests itself through a multimodal marginal posterior distribution. See Section 2.4.2 for an example of this. Through a set of interacting particles, SVGD is able to efficiently capture these modes. Accounting for posterior mass beyond the dominant mode is of utmost importance when trying to give realistic posterior predictions (Palacios et al., 2006; Gelfand et al., 2010).

2.3.2 Non-conjugate models

When the likelihood function of the data $p(y_i \,|\, f_i, \boldsymbol{\theta})$ in a GP model does not admit a Gaussian distribution, we are required to learn the latent values $\mathbf{f}$ of the GP in

addition to $\boldsymbol{\theta}$. To decouple the strong dependency that exists between $\mathbf{f}$ and $\boldsymbol{\theta}$, we centre (or 'whiten') the GP's covariance matrix such that $\mathbf{f} = L_{\boldsymbol{\theta}}\boldsymbol{\nu}$, where $L_{\boldsymbol{\theta}}$ is the lower Cholesky decomposition of the Gram matrix $\mathbf{K}_{\star\star}$, and $\nu_i \sim \mathcal{N}(0, 1)$. Applying such a transformation has been shown to enhance the performance of inferential schemes in the GP setting (I. Murray et al., 2010; Hensman et al., 2015a; Salimbeni et al., 2017). Once whitened, we can use the joint posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\nu} \,|\, X, \mathbf{y})$ as the target distribution for SVGD and *post-hoc* deterministically transform the posterior samples to give $p(\boldsymbol{\theta}, \mathbf{f} \,|\, X, \mathbf{y})$.

SVGD requires evaluation of the score function of the density at the current particle values to evaluate Equation (2.15). Using automatic differentiation, this is a trivial task. However, it is accompanied by a computational cost that scales quadratically with N since we can no longer marginalise $\mathbf{f}$. For this reason, in datasets surpassing several thousand data points, we estimate the score function using subsampled mini-batches $\Psi \subset \{1, \ldots, N\}$. The result of this is a score function approximation that can be written as

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\nu}} \log p(\boldsymbol{\theta}, \boldsymbol{\nu} \,|\, \mathbf{y}) \approx \nabla_{\boldsymbol{\theta},\boldsymbol{\nu}} \log p_0(\boldsymbol{\theta}, \boldsymbol{\nu}) \frac{N}{|\Psi|} \sum_{i \in \Psi} \nabla_{\boldsymbol{\theta},\boldsymbol{\nu}} \log p(y_i \,|\, \boldsymbol{\theta}, \nu_j) \,, \tag{2.19}$$

where i indexes the respective elements in $\mathbf{y}$ j indexes ν from 1 to i. The quality of this approximation is empirically demonstrated in Section 2.4.3 where we observe comparable performance to the current state-of-the-art GP inference scheme.

2.3.3 Posterior predictions

Once a set of particles $\boldsymbol{\Lambda}$ has been learned, we can use the value of each particle to make predictive inference. Recalling that the primary motivation of SVGD is to enable accurate predictive inference in posterior distributions with complex and multimodal geometries, naively taking the mean particle value for each parameter

as the final estimate of each parameter in the GP could become problematic when the posterior is complex. For this reason, to obtain a predictive mean and variance, we sample K times from the predictive posterior of the GP for each of the J particles and compute the mean and variance of the predictive samples. Such an approach is equivalent to Monte-Carlo sampling and, consequently, admits has an error rate of $1/\sqrt{K}$. To elucidate this notion, the procedure is summarised in Algorithm 2 .

We would like to show that the SteinGP in Algorithm Algorithm 1 iteratively improves the posterior approximation. Consider the variational distributions q_t of the particles at time t. We can show that the Kullback-Leibler divergence between these distributions and the target p decreases monotonically as t increases. The following result is similar to that of Liu (2017) and Korba et al. (2020); however, our proof focusses on the asymptotic case, an assumption that has since been relaxed to give non-asymptotic results. This serves as a useful starting point for more rigorous analysis of GPs (e.g., (Burt et al., 2020)).

Theorem 2.3.1 *Consider SVGD in a general model with* $\log p(\boldsymbol{\lambda})$ *at least twice continuously differentiable and* $\nabla \log p(\boldsymbol{\lambda})$ *is smooth with Lipschitz constant* L*. The Kullback-Leibler divergence between the target distribution* p *and its SVGD approximation* q_t *at iteration* t *is monotonically decreasing, satisfying*

$$\begin{aligned} \mathrm{KL}(q_{t+1}||p) - \mathrm{KL}(q_t||p) \leq -\epsilon \mathbb{D}(q_t, p)^2 (1 - \\ \epsilon \mathbb{E}_{\boldsymbol{\lambda} \sim q_t} \left[L\kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda})/2 + \nabla_{\boldsymbol{\lambda}, \boldsymbol{\lambda}} \kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda}) \right]) \end{aligned} \tag{2.20}$$

for a step-size parameter ϵ *that is chosen to satisfy* $0 < \epsilon \leq \rho(\nabla_{\boldsymbol{\lambda}} \hat{\phi}(\boldsymbol{\lambda}))^{-1}$*, where* $\rho(\cdot)$ *is the matrix spectral norm.*

Proof. SVGD maps particles using $\boldsymbol{\lambda}_{t+1} = \mathcal{T}(\boldsymbol{\lambda}_t) = \boldsymbol{\lambda}_t + \epsilon \hat{\phi}(\boldsymbol{\lambda}_t)$. Denote the

corresponding mapping of densities by $q_{t+1} = T(q_t)$. We have that

$$\begin{aligned}
\mathrm{KL}(q_{t+1}||p) - \mathrm{KL}(q_t||p) =& \mathrm{KL}(T(q_t)||p) - \mathrm{KL}(q_t||p) \\
=& \mathrm{KL}(q_t||T^{-1}(p)) - \mathrm{KL}(q_t||p) \\
=& \mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\log q_t(\boldsymbol{\lambda}) - \log T^{-1}(p)(\boldsymbol{\lambda})\right] \\
& - \mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\log q_t(\boldsymbol{\lambda}) - \log p(\boldsymbol{\lambda})\right] \\
=& \mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\log p(\boldsymbol{\lambda}) - \log T^{-1}(p)(\boldsymbol{\lambda})\right]. \qquad (2.21)
\end{aligned}$$

Under the change of variable formula for densities, we have

$$T^{-1}(p)(\boldsymbol{\lambda}) = p(\mathcal{T}(\boldsymbol{\lambda})) \cdot |\det(\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda}))|$$

which allows us to rewrite Equation (2.21) as

$$\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\log p(\boldsymbol{\lambda}) - \log p(\boldsymbol{\lambda} + \epsilon\hat{\phi}(\boldsymbol{\lambda})) - \log|\det(\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda}))|\right]. \qquad (2.22)$$

Assuming that $\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{\lambda})$ is Lipschitz smooth with constant L and $\log p(\boldsymbol{\lambda}) \in C^2$, a second order Taylor series approximation of $\log p(\boldsymbol{\lambda} + \epsilon\hat{\phi}(\boldsymbol{\lambda}))$ about $\boldsymbol{\lambda}$ (assuming $\epsilon \ll 1$) lets us bound the first two terms in Equation (2.22) by

$$\log p(\boldsymbol{\lambda}) - \log p(\boldsymbol{\lambda} + \epsilon\hat{\phi}(\boldsymbol{\lambda})) \leq -\epsilon\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{\lambda})^\intercal\hat{\phi}(\boldsymbol{\lambda}) + \frac{L\epsilon^2}{2}\hat{\phi}(\boldsymbol{\lambda})^\intercal\hat{\phi}(\boldsymbol{\lambda}). \qquad (2.23)$$

Noting the definition of the Stein operator from Equation (2.4), we have that $-\epsilon\nabla_{\boldsymbol{\lambda}} \log p(\boldsymbol{\lambda})^\intercal\hat{\phi}(\boldsymbol{\lambda}) = \mathrm{trace}(-\epsilon\mathcal{A}_p\hat{\phi}(\boldsymbol{\lambda}) + \epsilon\nabla_\lambda\hat{\phi}(\boldsymbol{\lambda}))$. Note also that $\mathcal{T}(\boldsymbol{\lambda}) = \boldsymbol{\lambda} + \epsilon\hat{\phi}(\boldsymbol{\lambda})$, and therefore $\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda}) = I + \epsilon\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda})$. We can lower bound the final term in Equation (2.22) by first noting that by the approximate Neumann expansion of

the inverse matrix $\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda})^{-1}$,

$$\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda})^{-1} = (I + \epsilon\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}))^{-1} \approx I - \epsilon\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}) + (\epsilon\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}))^2 \tag{2.24}$$

which holds for $0 < \epsilon \leq \rho(\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}))^{-1}$, where $\rho(\cdot)$ is the matrix spectral norm. We can bound the final term in Equation (2.22) using the following lower bound,

$$\log|\det(\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda}))| \geq \sum_{i=1}^{d}(1 - e_i^{-1}) = \text{trace}(I - \nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda})^{-1}) \tag{2.25}$$

where $e_1, \ldots, e_d$ are the eigenvalues of $\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda})$. Replacing the Neumann expansion for $\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda})^{-1}$ in Equation (2.25), gives the following lower bound,

$$\log|\det(\nabla_{\boldsymbol{\lambda}}\mathcal{T}(\boldsymbol{\lambda}))| \geq \epsilon\nabla_{\boldsymbol{\lambda}} \cdot \hat{\phi}(\boldsymbol{\lambda}) - \epsilon^2||\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}))||_F^2, \tag{2.26}$$

where $||\cdot||_F$ is the Frobenius norm.

Combining Equation (2.23) and Equation (2.26) gives the following upper bound for Equation (2.22),

$$\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[-\epsilon\nabla_{\boldsymbol{\lambda}}\log p(\boldsymbol{\lambda})^\intercal\hat{\phi}(\boldsymbol{\lambda}) + \frac{L\epsilon^2}{2}\hat{\phi}(\boldsymbol{\lambda})^\intercal\hat{\phi}(\boldsymbol{\lambda}) + \epsilon\nabla_{\boldsymbol{\lambda}}\cdot\hat{\phi}(\boldsymbol{\lambda}) + \epsilon^2||\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}))||_F^2\right].$$

$$= \underbrace{-\epsilon\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\mathcal{A}_p\hat{\phi}(\boldsymbol{\lambda})\right]}_{B} + \underbrace{\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\underbrace{\epsilon^2||\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda}))||_F^2}_{C1} + \underbrace{\frac{L\epsilon^2}{2}\hat{\phi}(\boldsymbol{\lambda})^\intercal\hat{\phi}(\boldsymbol{\lambda})}_{C2}\right]}_{C}.$$

By definition of the Stein discrepancy Equation (2.12), $B = -\epsilon\mathbb{D}(q_t, p)^2$, and

Equation (2.21) becomes

$$\mathrm{KL}(q_{t+1}||p) - \mathrm{KL}(q_t||p) \leq -\epsilon \mathbb{D}(q_t, p)^2 + C. \tag{2.27}$$

Based on this, we must now show that C is bounded, which we can do by considering each term individually.

C2: We can bound this term using the properties of the RKHS (Berlinet et al., 2004). As $\hat{\phi} = (\hat{\phi}_1, \ldots, \hat{\phi}_d)'$ and $\hat{\phi}_i \in \mathcal{H}_0 \implies \hat{\phi} \in \mathcal{H}^d$ then

$$\begin{aligned}
\left\|\hat{\phi}(\boldsymbol{\lambda})\right\|_2^2 &= \sum_{i=1}^{d} \hat{\phi}_i(\boldsymbol{\lambda})^2 \\
&= \sum_{i=1}^{d} \left(\langle \hat{\phi}_i(\cdot), \kappa(\boldsymbol{\lambda}, \cdot)\rangle_{\mathcal{H}_0}\right)^2 \text{ which follows from the RKHS properties} \\
&\leq \sum_{i=1}^{d} \left\|\hat{\phi}\right\|_{\mathcal{H}_0}^2 \|\kappa(\boldsymbol{\lambda}, \cdot)\|_{\mathcal{H}_0}^2 \text{ by Cauchy-Schwarz} \\
&= \left\|\hat{\phi}\right\|_{\mathcal{H}^d}^2 \kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda}') \\
&= \mathbb{D}(q_t, p)^2 \kappa(\boldsymbol{\lambda}, \boldsymbol{\lambda}') \text{ which follows by Equation (2.13)}
\end{aligned} \tag{2.28}$$

C1: We upper bound the matrix norm $\epsilon||\nabla_{\boldsymbol{\lambda}} \hat{\phi}(\boldsymbol{\lambda})||_F^2$ using the same RKHS property used in C2.

$$
\begin{aligned}
\left\|\nabla_{\boldsymbol{\lambda}}\hat{\phi}(\boldsymbol{\lambda})\right\|_F^2 &= \sum_{i=1}^{d}\sum_{j=1}^{d}\left(\frac{\partial\hat{\phi}_i(\boldsymbol{\lambda})}{\partial\boldsymbol{\lambda}_j}\right)^2 \quad \text{from definition above and the Frobenius norm} \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d}\left(\langle\hat{\phi}_i(\cdot),\partial\kappa(\boldsymbol{\lambda},\cdot)/\partial\boldsymbol{\lambda}_j\rangle_{\mathcal{H}_0}\right)^2 \quad \text{by Theorem 1 of D. X. Zhou (2008)} \\
&\leq \sum_{i=1}^{d}\sum_{j=1}^{d}\left\|\hat{\phi}_i\right\|_{\mathcal{H}_0}^2\left\|\partial\kappa(\boldsymbol{\lambda},\cdot)/\partial\boldsymbol{\lambda}_j\right\|_{\mathcal{H}_0}^2 \quad \text{by Cauchy-Schwarz} \\
&= \left\|\hat{\phi}\right\|_{\mathcal{H}^d}^2\nabla_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}\kappa(\boldsymbol{\lambda},\boldsymbol{\lambda}') \\
&= \mathbb{D}(q_t,p)^2\nabla_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}\kappa(\boldsymbol{\lambda},\boldsymbol{\lambda}').
\end{aligned}
$$

Finally, putting terms C1 and C2 together, Equation (2.27) now becomes

$$
\begin{aligned}
\mathrm{KL}(q_{t+1}||p) - \mathrm{KL}(q_t||p) \leq &-\epsilon\mathbb{D}(q_t,p)^2 + \epsilon^2\mathbb{D}(q_t,p)^2\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\nabla_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}\kappa(\boldsymbol{\lambda},\boldsymbol{\lambda})\right] + \\
&\frac{\epsilon^2 L}{2}\mathbb{D}(q_t,p)^2\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[\kappa(\boldsymbol{\lambda},\boldsymbol{\lambda})\right] \\
= &-\epsilon\mathbb{D}(q_t,p)^2\left(1-\epsilon\mathbb{E}_{\boldsymbol{\lambda}\sim q_t}\left[L\kappa(\boldsymbol{\lambda},\boldsymbol{\lambda})/2+\nabla_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}\kappa(\boldsymbol{\lambda},\boldsymbol{\lambda})\right]\right).
\end{aligned}
$$

□

It can be shown that for the GP models considered in this paper, the gradients are Lipschitz smooth and therefore Theorem 2.3.1 holds (Solak et al., 2002; Lederer et al., 2019). Additionally, Theorem 8 of Gorham et al. (2015) establishes weak convergence for a sequence of probability measures $(q_t)_{t\geq 1}$, where $q_t \implies p$ if $\mathbb{D}(q_t,p)\to 0$, and so it follows from Theorem 2.3.1, that $q_t \implies p$ as $t\to\infty$.

2.4 Experiments

For the datasets used in Section 2.4.1, models are fit using 70% of the full dataset, whilst the remaining 30% is used for model assessment. To obtain standard errors,

Table 2.2: Full list of the UCI datasets used in Section 2.4.1. N corresponds the number of observations, whilst the dimension column quantifies the dimensionality of the inputs. For datasets with a binary target, we also report the dataset's imbalance through the proportion of observations whereby the target is positive i.e. 1.

Dataset	N	Dimension	Positive-proportion
airfoil	1503	5	-
autompg	392	7	-
blood	748	5	23.8%
boston	506	13	-
breast-cancer	286	10	29.72%
challenger	23	4	-
concrete	1030	8	-
concreteslump	103	7	-
fertility	100	10	12.0%
gas	2565	128	-
hepatitis	155	20	79.35%
machine	209	7	-
mammographic	961	6	46.31%
parkinsons	5875	20	-
servo	167	4	-
skillcraft	3338	19	-
spectf	267	45	79.4%
winered	1599	11	-
winewhite	4898	11	-

the procedure is repeated five times using a different training/test split each time i.e., 5-fold cross validation. The partitioning of data, hyperparameter initialisation and computing environment used are the same for all models. For each dataset, we standardise the inputs and outputs to zero mean and unit standard deviation.

Throughout our experiments, we use an RBF kernel κ to specify the SVGD step Equation (2.15) (which is an independent choice from the kernel within the GP model). The RBF kernel's variance is set to 1, and we select the lengthscale at

each iteration of SVGD using the median rule (Garreau et al., 2017) as in Liu et al. (2016b). We run experiments with $J = 2$, 5, 10 and 20 particles (labelled SteinGP2 through to SteinGP20).

2.4.1 UCI datasets

We analyse 19 UC Irvine (UCI) datasets where the target is 1-dimensional. Datasets range in size from 23 to 5875 data points. As highlighted in Table 2.2, the targets in 6 of the datasets are binary, whilst the remaining 14 datasets are continuous values. To accommodate this, we use a GP with a Bernoulli likelihood for the classification tasks, and a Gaussian likelihood for regression. For regression, we infer the model's hyperparameters and for classification we additionally infer the latent values of the GP.

Table 2.3: Mean test log-likelihoods ± one standard deviation (larger is better) over 5 independent data splits of the UCI benchmark experiment presented in Section 2.4.1. Bold values indicate the best performing method. Our SteinGP with 2, 5, 10 and 20 particles is compared against a Gaussian process fitted using variational inference, maximum likelihood (ML) and Hamiltonian Monte-Carlo (HMC). For datasets where there is a significant difference between the best performing and one, or more, alternative methods, the dataset's name is listed in purple.

Dataset	SteinGP2	SteinGP5	SteinGP10	SteinGP20	VI	ML/HMC
Airfoil	$\mathbf{0.06 \pm 0.04}$	0.06 ± 0.04	0.05 ± 0.06	0.05 ± 0.05	0.03 ± 0.03	0.03 ± 0.03
Autompg	-0.39 ± 0.09	-0.39 ± 0.09	-0.39 ± 0.09	-0.4 ± 0.09	$\mathbf{-0.39 \pm 0.07}$	-0.39 ± 0.07
Blood	-0.6 ± 0.05	-0.6 ± 0.04	-0.6 ± 0.05	-0.61 ± 0.04	$\mathbf{-0.51 \pm 0.05}$	-0.52 ± 0.06
Boston	-0.3 ± 0.12	$\mathbf{-0.28 \pm 0.11}$	-0.3 ± 0.12	-0.3 ± 0.13	-0.31 ± 0.13	-0.31 ± 0.13
Breast Cancer	-0.08 ± 0.04	-0.08 ± 0.02	-0.08 ± 0.02	$\mathbf{-0.08 \pm 0.01}$	-0.65 ± 0.09	$\mathbf{-0.08 \pm 0.04}$
Challenger	-1.53 ± 0.45	-1.52 ± 0.43	$\mathbf{-1.46 \pm 0.32}$	-1.53 ± 0.41	-1.51 ± 0.3	-1.51 ± 0.3
Concrete	-0.25 ± 0.07	-0.25 ± 0.07	-0.25 ± 0.07	-0.25 ± 0.07	$\mathbf{-0.24 \pm 0.05}$	-0.24 ± 0.05
Concreteslump	1.08 ± 0.39	1.07 ± 0.41	$\mathbf{1.06 \pm 0.4}$	1.08 ± 0.39	0.13 ± 1.14	0.13 ± 1.14
Fertility	-0.44 ± 0.03	-0.44 ± 0.03	-0.43 ± 0.02	$\mathbf{-0.42 \pm 0.02}$	-0.70 ± 0.08	-0.54 ± 0.02
Gas	0.88 ± 0.11	0.88 ± 0.11	$\mathbf{0.89 \pm 0.1}$	0.88 ± 0.11	0.79 ± 0.11	0.79 ± 0.11
Hepatitis	-0.41 ± 0.07	-0.41 ± 0.07	-0.42 ± 0.07	$\mathbf{-0.4 \pm 0.07}$	-0.69 ± 0	-0.44 ± 0.04
Machine	$\mathbf{-0.51 \pm 0.09}$	-0.52 ± 0.08	-0.52 ± 0.08	-0.52 ± 0.08	-0.52 ± 0.07	-0.52 ± 0.07
Mammographic	-0.37 ± 0.03	-0.37 ± 0.03	-0.37 ± 0.03	-0.37 ± 0.03	-0.39 ± 0.03	-0.38 ± 0.04
Parkinsons	4.12 ± 0.05	4.12 ± 0.05	$\mathbf{4.14 \pm 0.03}$	4.13 ± 0.06	3.95 ± 0.04	3.95 ± 0.04
Servo	-0.48 ± 0.04	-0.43 ± 0.05	-0.41 ± 0.11	-0.41 ± 0.21	-0.39 ± 0.1	$\mathbf{-0.39 \pm 0.1}$
Skillcraft	$\mathbf{-0.99 \pm 0.02}$	-0.99 ± 0.02	-0.99 ± 0.02	-0.99 ± 0.02	-1.01 ± 0.02	-1.01 ± 0.02
Spectf	-0.26 ± 0.01	-0.26 ± 0.01	-0.26 ± 0.01	$\mathbf{-0.26 \pm 0.01}$	-0.69 ± 0	-0.68 ± 0.03
Winered	-1.17 ± 0.03	-1.17 ± 0.03	-1.17 ± 0.03	-1.17 ± 0.03	$\mathbf{-1.16 \pm 0.03}$	-1.16 ± 0.03
Winewhite	0.56 ± 0.05	0.57 ± 0.05	$\mathbf{0.57 \pm 0.05}$	0.57 ± 0.05	0.49 ± 0.05	0.55 ± 0.05

When the likelihood is Gaussian, we compare our SteinGP against a GP that is fit using maximum likelihood (ML) and another GP fit using VI (VI). For Bernoulli likelihoods, maximum likelihood estimation is not feasible so we instead

use Hamiltonian Monte-Carlo (HMC) for inference along with VI. We report the test log-likelihoods for the three model classes in Table 2.3 where the performance given by inferring the GP's parameters using SVGD is significant in comparison to comparative approaches. Furthermore, we can see that increasing the number of particles for SVGD often leads to larger improved log-likelihood values.

For some datasets, we see a weak improvement in the test log-likelihood as the number of particles increases. This is due to the simple posterior distribution that is being targetted and we visualise this in Figure 2.1 where we compare the marginal posterior distributions learned through the SteinGP against the samples drawn using HMC; the gold standard for Bayesian inference. In this figure, we see that even for 2 particles, the SteinGP model can recover a reasonable representation of the posterior distribution. For 20 particles, the SteinGP has perfectly recovered the marginal posteriors over the kernel hyperparameters given using HMC. For the observation noise variance, the SteinGP with 10 or 20 particles does a reasonable job of recovering the posterior distribution, but it does struggle to capture the posterior with 2 or 5 particles; an observation that aligns with the test log-likelihood results in Table 2.3.

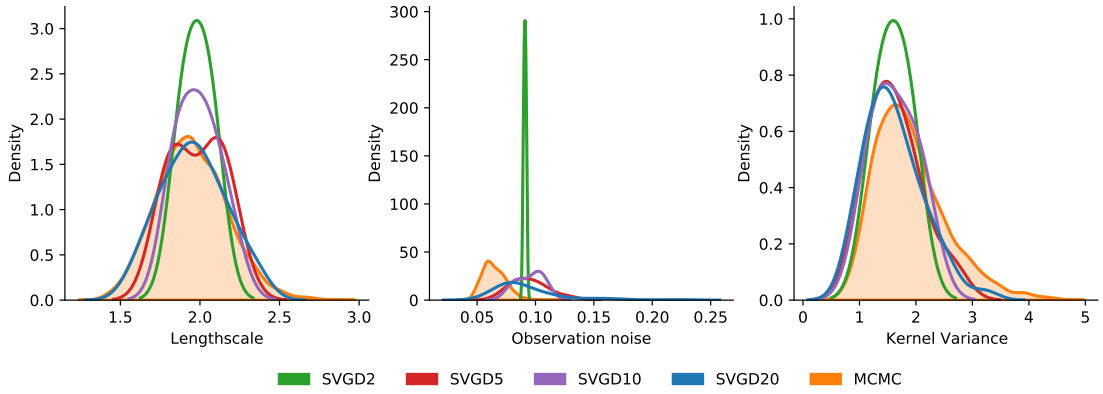


Figure 2.1: Marginal posterior distributions for the `servo` dataset used in Section 2.4.2.

For two particles, a SteinGP is nearly always faster than comparative methods.

Table 2.4: Relative average computational runtimes of comparative methods. Results are reported relative to a SteinGP with 2 particles i.e., a value of 2 would indicate that method was two times slower than SteinGP2.

	ML	VI	HMC
Relative wall time	1.8	1.3	5.6

The two exceptions to this are when 1) the maximum likelihood approach converges quickly or 2) VI is used for non-conjugate models. In the case of 1), the difference is in the order of seconds. Across all UCI datasets, we report the average runtime of each comparative method and report timings relative to a SteinGP with 2 particles Table 2.4. It is clear that a SteinGP is a computationally efficient model, particularly compared to a HMC approach, and a SteinGP consistently produces optimal predictive results.

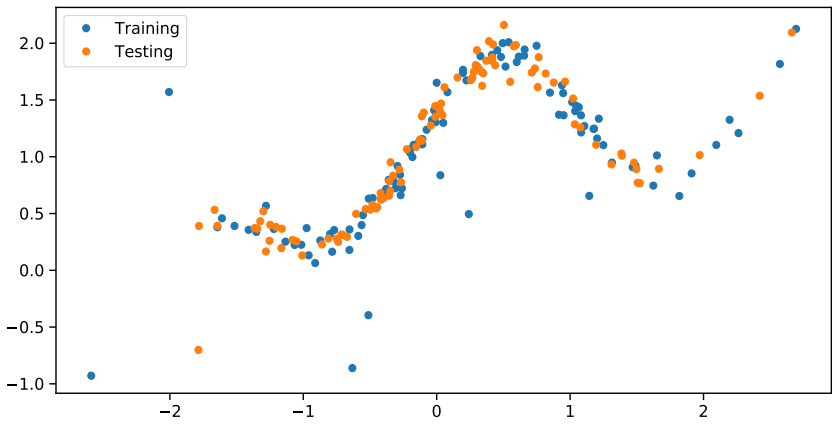


Figure 2.2: The multimodal dataset from Neal (1997) that is used in Section 2.4.2.

2.4.2 Multimodal posteriors

One way in which a multimodal posterior can be observed is in the case of a misspecified model. However, posterior multimodality can also occur due to corrupted data. To see this, we use the 1-dimensional example from Neal (1997)

whereby data is generated according to

$$y_i = 0.3 + 0.3x_i + 0.5\sin(2.7x_i) + \frac{1.1}{1+x_i^2} + \epsilon_i$$

where $x_i \sim \mathcal{N}(0,1)$ and $\epsilon_i \sim \mathcal{N}(0,\sigma^2)$. We set $\sigma = 1$ with probability 0.05, and $\sigma = 0.1$ otherwise, thus inflating the variance of some data points and creating outliers. We simulate 200 points from this model, with the first 100 used for the fitting of the GP and the remaining 100 used for evaluation Figure 2.2.

Table 2.5: Predictive metrics on 100 held out data points from the multimodal example in Section 2.4.2. Coverage statistics are computed by the number of test points that fall within a 90% credible interval of the predictive posterior distribution. A perfect score is, therefore, 90%.

	SteinGP2	SteinGP5	SteinGP10	SteinGP20	HMC	VI
RMSE	0.045	0.040	**0.039**	**0.039**	0.043	0.044
Log posterior density	-29.40	-25.94	-24.81	**-24.33**	-24.38	-27.31
Coverage 90%	64%	65%	68%	**70%**	68%	60%
Runtime (seconds)	**6.9**	9.1	14.9	24.8	50.4	14.7

The data are modelled using a GP that is equipped with an RBF kernel $k(x,x') = \alpha^2 \exp\left(-(x-x')^2/2\ell^2\right)$. We assume the observation noise follows a zero-mean Gaussian distribution with variance σ^2. We, therefore, wish to learn the posterior distribution $p(\boldsymbol{\theta})$ where $\boldsymbol{\theta} = \{\alpha, \ell, \sigma\}$.

We compare the posteriors of a SteinGP with 20 particles against the posterior samples from HMC in Figure 2.3. The point estimates inferred using VI are included for a full and faithful comparison. Accurately quantifying the posterior uncertainty is not only useful for parameter interpretation but also leads to higher-quality predictive inference (see Table 2.5). From Table 2.5 it is clear that at the cost of slightly longer computational runtime than VI, the performance in predictive inference given by a SteinGP is significant. This can be seen through the order of magnitude improvement in root-mean-square error (RMSE) compared

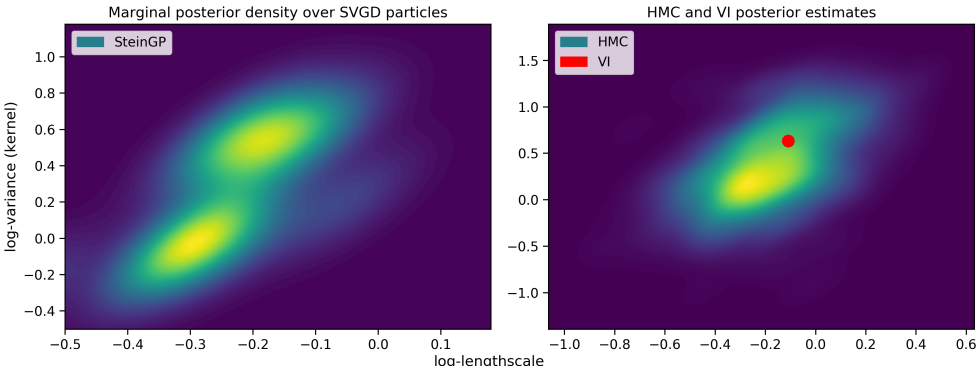


Figure 2.3: The marginal posterior distributions of the GP model's kernel parameters in the multimodal example of Section 2.4.2. The left panel shows the bimodal posterior learned through SVGD, whereas the Hamiltonian Monte-Carlo (right panel) inferred posterior has only identified a single mode. The scalar parameters learned with the variational GP are overlayed onto the HMC-inferred posterior distribution through a red point.

to VI and HMC.

In contrast to the posteriors of Section 2.4.1, the bimodality of the marginal posterior in this experiment demands a larger number of particles to be used within the SteinGP. This is to ensure that there are enough particles to capture additional modalities within the posterior distribution. In Table 2.5, we observe that optimal performance is achieved when 20 particles are used. This is evidenced through the reduced RMSE which indicates a more precise predictive mean, and improved coverage and posterior density metrics that quantify the quality of the posterior uncertainty. Whilst a SteinGP with 20 particles performs optimally, we note that 10 particles can provide an accurate representation of the posterior distribution, and with this number of particles inference in the SteinGP is only 0.2 seconds slower than the variational GP and 35.5 seconds faster than HMC.

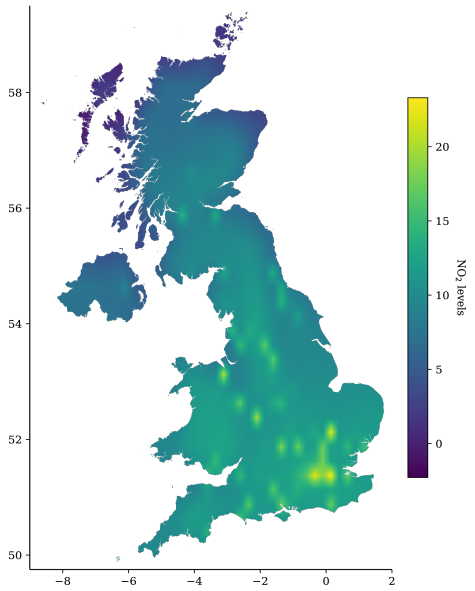


Figure 2.4: Inferred NO_2 spatial surface (μgm^{-3}) in the UK at 9AM on March 23rd; the day that initial lockdown measures were announced.

2.4.3 Spatiotemporal modelling

As a final experiment, we consider a spatiotemporal air quality dataset comprised of 550,134 data points. Data is recorded hourly from 237 sensors across the UK. The period of interest is February 1st to April 30th 2020; the period in which the UK entered a Covid-19 lockdown. Consequently, the spatiotemporal dynamics of air quality levels are chaotic and challenging to model as many individuals adopt lifestyle changes with ramifications for their contribution to air pollution. Historically, fully Bayesian inference with GP models would be infeasible on such datasets due to the challenging scaling of MCMC samplers. However, using the batched approach in Equation (2.19) we can make full Bayesian inference tractable.

To further demonstrate the efficacy of a SteinGP, we develop, in conjunction with climate scientists, a complex separable kernel that principally captures the complex spatiotemporal dynamics of atmospheric nitrogen dioxide (NO_2). Across the spatial dimensions we use a third-order Matérn kernel and in the temporal dimension we use a product of a first-order Matérn and a third-degree polynomial

kernel to capture the temporal nonstationarity. A third-degree polynomial was selected on the advice of climate scientists who advised that, as a result of the lockdown, there NO_2 levels would rise as time approaches the lockdown date before immediately dropping. From the lockdown date, it is believed that, slowly, activity will resume and a rise in NO_2 levels will be obsered. We further include a white noise process. The kernel function that parameterises our GP is, therefore, given by

$$k(\mathbf{x}, \mathbf{x}') = k_{\text{mat}_3}(\mathbf{s}, \mathbf{s}) + k_{\text{mat}_1}(t, t')k_{\text{poly}_3}(t, t') + \sigma^2\delta'_{x,x'}\,, \tag{2.29}$$

where $\mathbf{s}$ is 2-dimensional spatial coordinate and t the temporal indices. k_{mat_i} is the Matérn kernel of i^{th} order, k_{poly_3} is the third-order polynomial kernel, and δ is the Dirac-delta function.

Table 2.6: Comparison of our SteinGP with 30 particles against a Gaussian process fitted using stochastic variational inference on the air quality data of Section 2.4.3. Standard errors are computed by fitting each model on 5 random splits of the data, with 30% of the data being used for prediction.

	SteinGP	SVI
RMSE	0.82 ± 0.09	0.79 ± 0.07
Log-likelihood	-1.358 ± 0.15	-1.342 ± 0.11

Predictive performance We assess the performance of our model by computing the predictive log-likelihood and the RMSE of our model and compare against the state-of-art sparse GP fitted using stochastic VI (Hensman et al., 2013a). To induce sparsity, the same set 600 inducing points are used in both models. We initialise the inducing points through a Determinantal point process (DPP), as per Burt et al. (2020).

We optimise both models by running the SteinGP for 1000 iterations and the

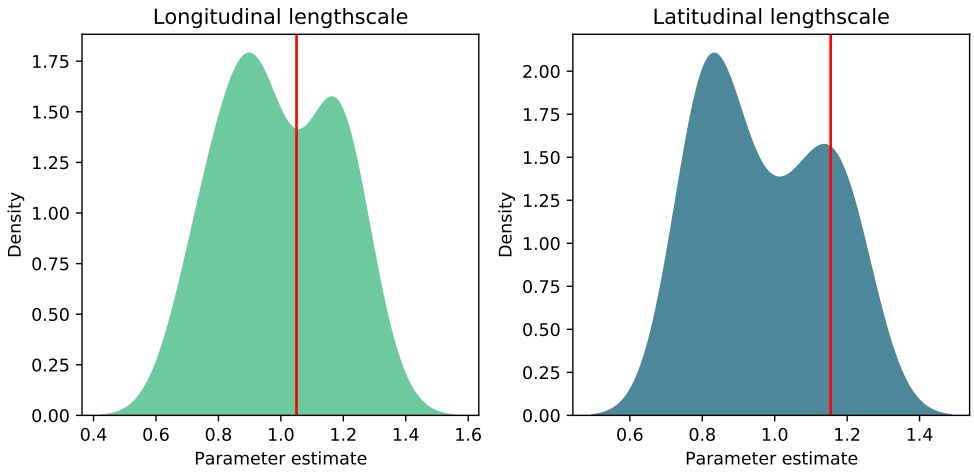


Figure 2.5: Marginal posterior distribution of the 2-dimensional lengthscale parameter used to model the normalised longitude and latitude coordinates in degrees.. The density for each element of the lengthscale is estimated using the optimised particles in SVGD and the red line corresponds to the scalar estimates produced through a stochastic VI scheme.

stochastic VI model for 10000. For both models, we use a batch size of 256 and a learning rate of 0.001. 30 particles are used for the SVGD routine. Due to the size of the dataset, a single step of a HMC sampler took over 10 minutes. Given that several thousand steps of the HMC sampler would be required, a comparison against HMC is infeasible.

Table 2.6 shows there is no significant difference between the GP fit using SVGD and stochastic VI. Unsurprisingly, VI is faster than SVGD with average computational wall times of 141 seconds and 332 seconds, respectively. However, this is to be expected due to the kernel computations that are required at each iteration of SVGD.

Uncertainty quantification Although the headline predictive performance of the two methods is comparable, the fully Bayesian treatment of the GP model that SVGD enables leads to full posterior inference and improved uncertainty quantification. To see this, we hold out all stations in the midlands of the UK (see

Table 2.7: Spatial interpolation of a SteinGP with 30 particles and a stochastic variational inference Gaussian process. Here, stations within (-2.2^o, 52^o), (-1^o, 54^o) are held back for testing.

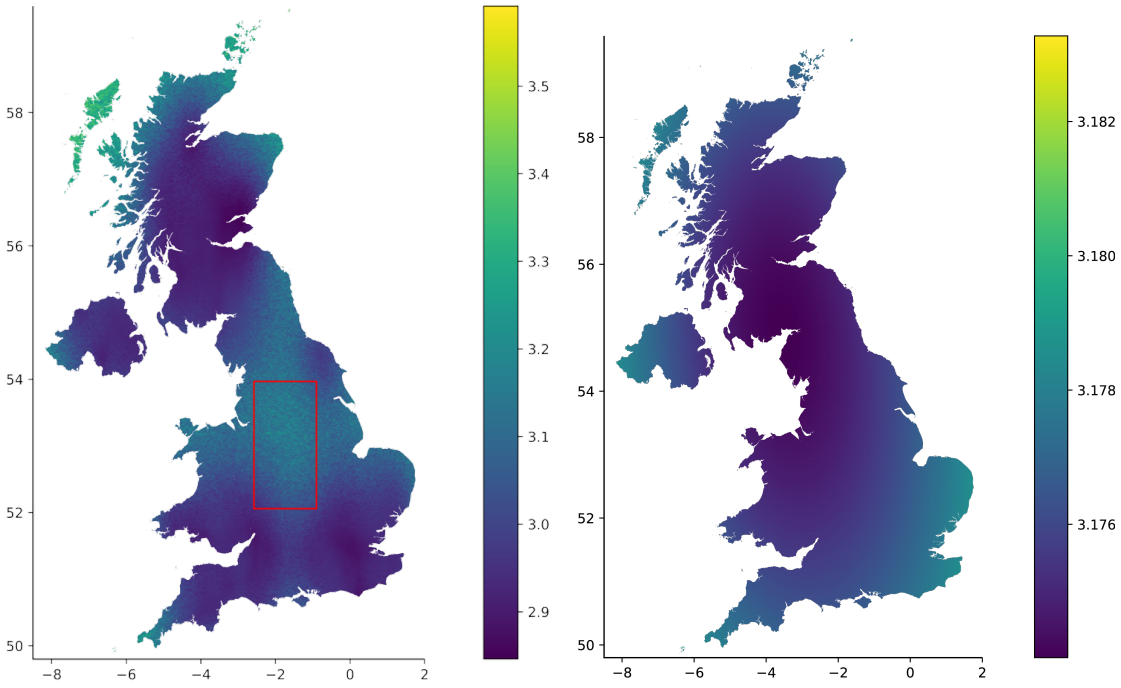
	SteinGP	SVI
RMSE	0.97	1.00
Log-likelihood	-1.467	-1.462

the red box in Figure 2.6(a)) and re-fit both the SteinGP and the stochastic VI counterpart. We then perform spatial interpolation over the midlands of the UK from February 1$^{\text{st}}$ through to April 30$^{\text{th}}$. In a model that is capable of generating effective predictive uncertainties, the predictive variance should be much larger over the midlands due to the lack of observations there.

Both models achieve comparable predictive metrics (Table 2.7). However, the purpose of this study is to assess the predictive uncertainties that each model yields. It can be seen in Figure 2.5 that estimating the spatial lengthscale parameters is challenging due to multimodalities. The SteinGP appears to have captured a secondary mode for each parameter. For completeness, we also give the scalar estimates produced by the stochastic VI procedure.

From Figure 2.6(a) it can be seen that the richer posterior inference provided from a SteinGP results in more reasonable posterior uncertainty estimates than those displayed in Figure 2.6(b). This can be seen by observing the increased uncertainty over the midlands of the UK. Noting the different legend colour scaling, it can be seen in comparison that the GP inferred using stochastic VI is incapable of capturing this behaviour and yields an almost uniform predictive variance across the UK.

2.4.4 Demo



(a) Figure A (b) Figure B

Figure 2.6: Posterior variances for a SteinGP (Figure 2.6(a)) and stochastic variational inference Gaussian process (Figure 2.6(b)). Data within the red square in Figure 2.6(a) was held back and predictions were then made across the entire UK. The lighter colours indicate a higher predictive variance; something that is expected when there are no observation present. We note the differing colour scales used in Figure 2.6(a) and Figure 2.6(b).

In Listing 1 we demonstrate the code released alongside this work. Code has been purposefully designed to integrate with `GPFlow` (Matthews et al., 2017), meaning that all operations are GPU-compatible and can be integrated with the extensive functionality provided in `GPFlow`.

2.5 Conclusions

We have shown that SVGD can be used to provide comparable inferential quality to the gold standard HMC sampler in GP inference, whilst only requiring a computational cost comparable to VI. The ability to carry out joint inference over

```python
from steingp import SteinGPR, RBF, Median, SVGD
import numpy as np
import gpflow as gpf

# Build data
X = np.random.uniform(-5, 5, 100).reshape(-1,1)
y = np.sin(x)

# Define model
kernel = gpf.kernels.SquaredExponential()
model = SteinGPR((X, y), kernel)

# Fit
opt = SVGD(model, RBF(bandwidth=Median()), n_particles=5)
opt.run(iterations = 1000)

# Predict
Xte = np.linspace(-5, 5, 500).reshape(-1, 1)
theta = opt.get_particles()
posterior_samples = model.predict(Xte, theta, n_samples=5)
```

Listing 1: Demonstration of the supplementary code package that implements a Stein variational Gaussian process.

latent function values and kernel hyperparameters allows for a full and proper consideration of uncertainty in the inference.

For simple problems where the true posterior is Gaussian, two or five SVGD particles are required to achieve strong inference, as can be seen in the experiments carried out in Section 2.4.1. However, for modelling tasks where posterior geometry is non-convex, such as those in Section 2.4.2, more particles can help capture subtle posterior geometries.

Well quantified predictive uncertainties are critical when GP models are being used in real-world scenarios. As shown in Section 2.4.3, when compared to variational GP, a SteinGP can provide well-quantified uncertainty estimates, even in big data scenarios. This, in conjunction with strong predictive performance, makes SteinGPs a useful tool for practitioners fitting GP models.

Chapter 3

Street-Level Modelling of Air Pollution Using Graph Gaussian Processes

In this chapter, an alternative approach to modelling air pollution is proposed by allowing the space on which the data is observed to be the vertex set of a graph, not a coordinate system, such as the one used in Section 2.4.3. As will be shown in Section 3.3, constructing GP models in this way enables questions such as "*What levels of NO_2 will I be exposed to if I wish to walk from one location to another?*" or "*Which road has the highest levels of NO_2 in my town?*" to be answered.

Extending regular GP models such that their support is the vertex set of a graph requires the derivation of kernel functions that can quantify the similarity between any pair of vertices within a graph. In this chapter, we build upon the work Borovitskiy et al. (2021) by developing a graph GP that is capable of modelling heteroscedastic noise processes on the vertices of a graph.

We demonstrate the use of our model by investigating NO_2 levels in the Mitcham and Tooting regions of London, using a dataset collected by the Breathe London project that gives mobile measurements collected in a moving vehicle. Through our model, we can infer NO_2 levels at a high resolution and begin to answer pertinent questions that link air pollution to matters such as public health and policy decision-making. We conclude this chapter by presenting the `LancasterAQ` dataset - a mobile dataset of ultrafine particle (UFP) that we have collected in Lancaster, UK. This new dataset provides a useful augmentation to the existing set of air pollution datasets as it measures UFP, an under-regulated pollutant, in a small city. This contrasts with existing datasets which typically measure larger particulate matters and NO_2 in major cities such as London and Barcelona.

3.1 Introduction

Exposure to poor ambient air pollution is a major public health issue and producing accurate prediction maps of pollution levels is a critical task in the earth science and machine learning communities. In this work, we focus our attention on NO_2 - a pollution compound that is primarily produced by the burning of fuels, such as by combustion engines in cars. Disentangling the direct effect of NO_2 on one's health is challenging, however, it is estimated that 3.6 million deaths per year result from exposure to pollution created by combustion engines (Lelieveld et al., 2019). There is no single illness that causes these deaths, but short-term exposure to NO_2 has been linked to reduced lung function and increased cardiovascular risk (Strak et al., 2012; Strak et al., 2013).

Constructing regression models capable of interpolating and forecasting air quality measurements in the absence of monitoring stations is challenging due to data scarcity and the complex underlying spatial topology that exists in urban areas.

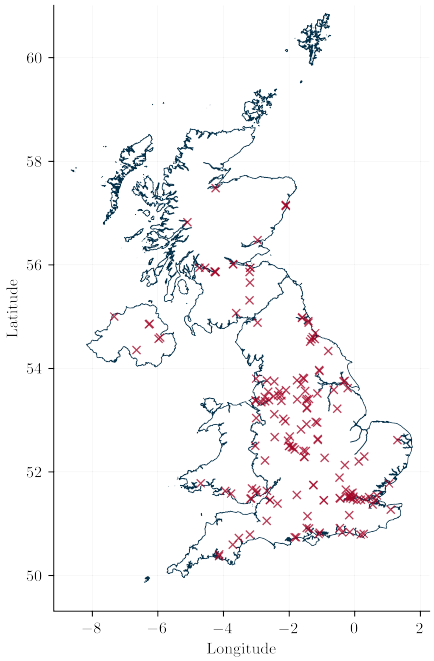


Figure 3.1: Location of the 145 automatic urban and rural network (AURN) sites within the UK that measure NO_2 levels.

Previous works have combined datasets of different resolution to try and resolve this issue (Hamelijnck et al., 2019), however, it cannot always be guaranteed that a range of suitable datasets will be available. In the UK, the largest network of NO_2 sensors contains 145 stations with a disproportionately large number of these residing in major cities (Figure 3.1). With so few sensors, it is often infeasible to build fine-scale models that are capable of inferring NO_2 levels at a street level. There exists a parallel line-of-work that considers *fusing* multiple datasets at varying resolutions together to obtain fine-scale predictions (Law et al., 2018; Hamelijnck et al., 2019; H. Zhu et al., 2021); however, we do not consider this here as it relies on multiple datasets of the same quantity being available, a requirement that often cannot be satisfied. Furthermore, assuming the model's domain is Euclidean is a common assumption, yet unrealistic in cities and urban areas. Factors such as varying traffic speeds and congestion levels are more accurately explained using a road network (i.e., the graph approach) rather than assuming

the underlying space is unconstrained and continuous (Euclidean approach). A pertinent example of this can be visualised in Figure 3.2(a) where we see the problem encountered when computing Euclidean distances on network spaces.

We wish to design a statistical model for air quality that is well-calibrated, accounts for the underlying road network, and allows us to make predictions at locations for which we do not have observations. Firstly, a well-calibrated model is essential for informing policy decisions since this highlights points at which the model can be trusted and those where further investigation is required. Secondly, our model should reflect the topology of the road network so that outputs from the model appropriately reflect constraints implied by the real world. Finally, the cost and time-intensive nature of the data collection process motivates the development of models which offer accurate predictions away from the observations. This allows practitioners to make statements based on limited data and outputs from the model can be viewed as a cheap alternative to data collection.

In this work, we construct a GP regression model that is defined on a graph representation of London's road network where a junction is represented by a vertex and a road by an edge. GPs are a well-motivated choice of model as they offer calibrated uncertainty estimates and their flexible non-parametric form allows us to model the complex dynamics exhibited by NO_2. We parameterise our GP using a graph kernel, which allows us to capture the correlation structure present within London's road network. Such a representation allows us to use our model to assess the levels of pollution exposure experienced when taking common journeys, such as a commute or bike ride (Figure 3.5). Identifying roads with dangerous levels of pollution (Figure 3.7) enables us to recognise and proactively assist individuals who are more at risk of contracting pollution-related illnesses.

3.2 Graph theory primer

A graph $\mathcal{G} = \{V, E, W\}$ is a triad comprised of a set of vertices $V = \{v_1, v_2, \dots v_{n_v}\}$, a set of edges $E = \{(v_i, v_j) : v_i, v_j \in V\} \subseteq V \times V$ for $|E| = n_e$, and a weight function $w : V \times V \to \mathbb{R}$ that assigns a value to each edge in the graph. For three examples of real-world graphs, consider the following

1. A molecule, where a vertex is an atom and an edge represents a bond between two atoms. The bonds' strength is given by the weight function.

2. Social networks, where individual users constitute a vertex, and an edge is present between two users if they're connected on the social network. The edge's weight is given by the amount the two users interact with one another.

3. Road networks, where each vertex is a junction and an edge connects two junctions if they are connected by a road. Assuming bidirectional traffic he edge's weight is given by the inverse length of the road (see Section 3.3.1 for further intuition on this point.).

Within graph signal processing, a graph signal is a function $g : V \to \mathbb{R}$ that is defined on the vertices of a graph. A realisation of this signal is denoted $\mathbf{g} = [g(v_1), g(v_2), \dots, g(v_{n_v})] \in \mathbb{R}^{n_v}$. Through a model, the task is to learn the graph signal's functional form so that predictions can be made at vertices where the signal was not observed.

Three matrices that can be used to represent a graph are the adjacency matrix $\mathbf{A}$, the degree matrix $\mathbf{D}$, and the Laplacian matrix $\mathbf{\Delta}$. The graph *adjacency matrix* $\mathbf{A}$ is a square matrix such that $[\mathbf{A}]_{i,j}$ is equal to $w(v_i, v_j)$ if and only if $(v_i, v_j) \in E$, and 0 otherwise. To denote adjacency between two vertices v_i, v_j, we can also write $v_i \sim v_j$. We only consider symmetric graphs in this work i.e., $(v_i, v_j) \in \mathrm{E} \iff (v_j, v_i) \in \mathrm{E}$ and, for graphs of this form, the adjacency matrix

is symmetric. The *degree matrix* $\mathbf{D}$ of a graph is a diagonal matrix such that $[\mathbf{D}]_{i,i} = \sum_{j \neq i} A_{i,j}$.

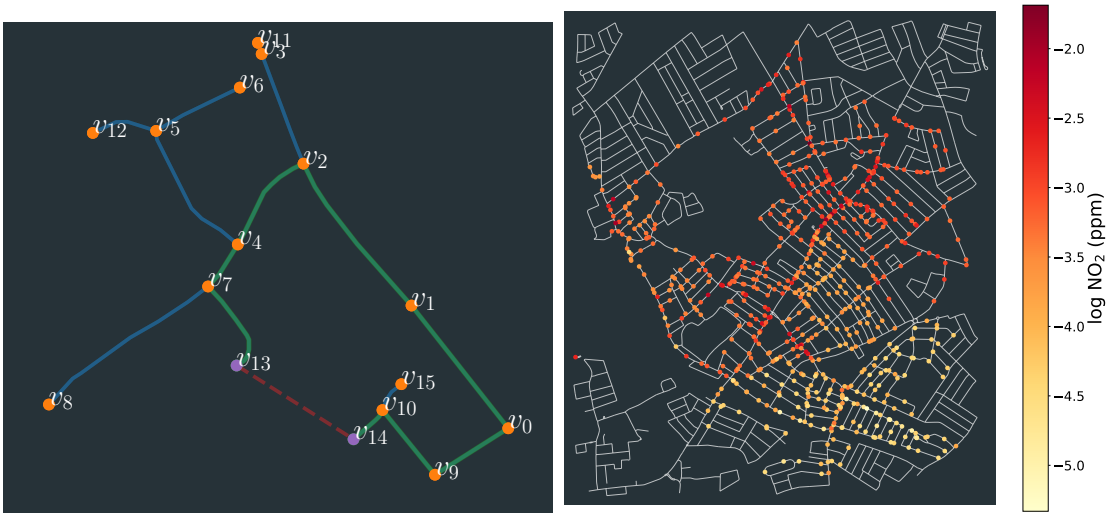


Figure 3.2: Left panel: A comparison of graph (green) and Euclidean (red) distances when the underlying space is a network. Right panel: NO2 observations overlayed onto Mitcham's road network. Observations have been log-scaled to aid visualisation.

The *graph Laplacian* combines the adjacency and degree matrices through $\mathbf{\Delta} = \mathbf{D} - \mathbf{A}$. The graph Laplacian is a diagonally dominant matrix and it is a fundamental concept in spectral graph theory. For a graph signal $\mathbf{g}$, the graph Laplacian can be used to quantify the smoothness of the signal over the graph's vertices (Smola et al., 2003). To see this, we can consider the following

$$\langle \mathbf{g}, \mathbf{\Delta g} \rangle = \mathbf{g}^\top \mathbf{\Delta g} = \frac{1}{2} \sum_{v_i \sim v_j} w_{i,j} (g_i - g_j)^2 \,, \tag{3.1}$$

for all $\mathbf{g} \in \mathbb{R}^{n_v}$. Functions that are smooth over the vertices will result in small $(g_i - g_j)^2$ values, whereas rough functions will change rapidly across vertices, meaning the quadratic difference will be larger. The graph Laplacian is symmetric positive-

definite and therefore admits the eigendecomposition

$$\boldsymbol{\Delta} = \mathbf{U}\boldsymbol{\lambda}\mathbf{U}^{\top}, \tag{3.2}$$

where $\boldsymbol{\lambda}$ is the diagonal matrix of non-negative eigenvalues and $\mathbf{U}$ is the orthonormal matrix whose columns are the eigenvectors of $\boldsymbol{\Delta}$.

3.3 Case study

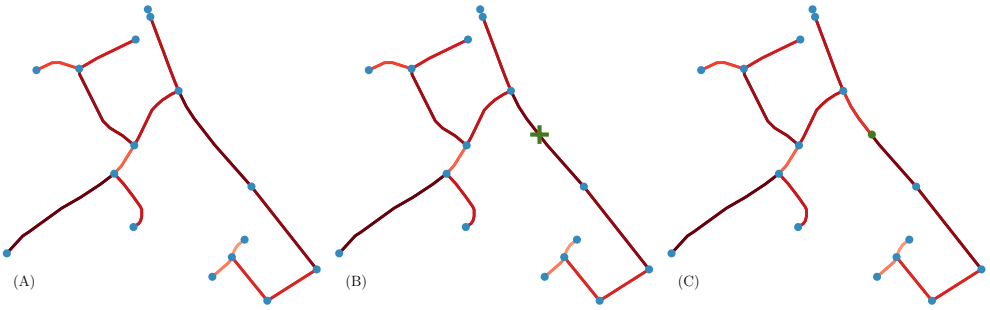


Figure 3.3: Schematic describing the inclusion of NO2 measurements into a road network. With an underlying road network (A), vertices are junctions and edges are roads that connect two junctions. The colour of an edge denotes the edge's length. Measurements are made at points on the road (green cross, (B)) and the nearest edge is spliced at this point. We then reconstruct the edge as a pair of edges with length proportional (C).

3.3.1 Data

We use NO_2 measurements from the Breathe London Mobile study where pollution measurement sensors were fixed to two Google street view cars (Hasenkopf et al., 2015). NO_2 levels were measured every 1-10 seconds as the cars drove around London. 600 journeys were made from Autumn 2018 to Autumn 2019 when the study ended. We use data from 6-7.30 AM on December 18th, 2018 in the suburb of Mitcham, which provides us with 695 measurements (Figure 3.2(b)).

To represent the roads of Mitcham as a graph, we use data from OpenStreetMap

(Bennett, 2010) and let each junction be a vertex v. Two vertices are connected by an edge e in the graph if a road connects the corresponding junction pair. Intuitively, an adjacent pair of vertices are *more similar* the closer they are to one another. For this reason, we weight the edges in our graph by the inverse distance between the adjacent vertex pair. We next augment the graph with the NO_2 measurements. For each measurement, we identify the nearest edge, splice the edge at the measurement's coordinate, and reconstruct the graph with the measurement becoming a new vertex and two new edges being created to connect the measurement vertex to its adjacent junctions. We calculate length using the measurement's distance from its adjacent junction pair. We visualise this process in Figure 3.3.

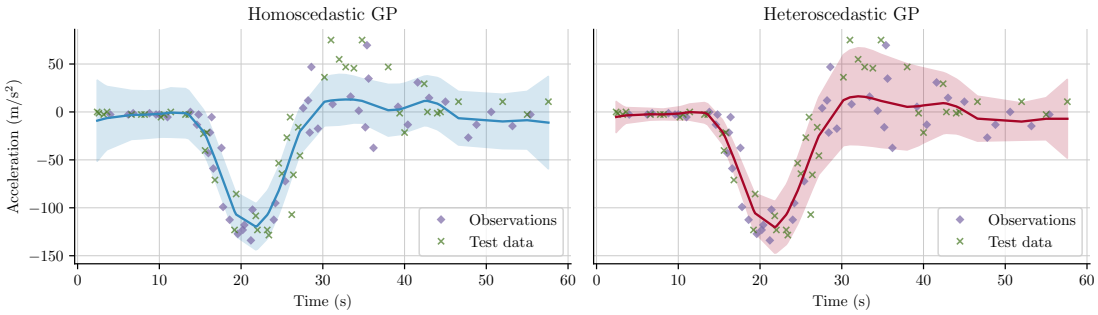


Figure 3.4: A comparison of a homoscedastic GP (left) whose noise term is constant and a heteroscedastic GP (right) whose noise term is a vector. Both models are fitted using the same train/test split of the motorcycle dataset given in (Silverman, 1985). In each panel, the shaded regions represent the 90% credible interval around the predicted mean, given by the solid line.

3.3.2 Model specification

Equipped with a graph representation, we now seek to define a GP on the graph's vertices. Apriori, it is challenging to infer the smoothness of the air pollution data being modelled. For this reason, we use the Matérn kernel $k : V \times V \to \mathbb{R}$ given in Borovitskiy et al. (2021) as its flexible parameterisation will allow us to capture a

range of functions with differing smoothness. For a smoothness parameter $\nu \in \mathbb{R}_{>0}$ and lengthscale $\ell \in \mathbb{R}_{>0}$, the GP prior can be written as

$$p(\mathbf{f}) = \mathcal{N}\left(\mathbf{0}, \left(\frac{2\nu}{\ell^2}\mathbf{I}_{n_v} + \boldsymbol{\Delta}\right)^{-\nu}\right) \tag{3.3}$$

$$= \mathcal{N}\left(\mathbf{0}, \mathbf{U}\left(\frac{2\nu}{\ell^2}\mathbf{I}_{n_v} + \operatorname{diag}(\boldsymbol{\lambda})\right)^{-\nu}\mathbf{U}^\top\right), \tag{3.4}$$

using the eigendecomposition of the Laplacian matrix given in Equation (3.2) where n_ν denotes the number of vertices within the graph.

Up until this point, a constant, homoscedastic noise term has been used. However, in this work, the NO_2 values' variance increases proportionally to the magnitude of the value. To account for this, we will seek to learn a vector of noise values $\boldsymbol{\sigma^2} = \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_{n_v}^2\}$, where each observation is mapped to a unique element in $\boldsymbol{\sigma}$. Such a model is known as a heteroscedastic model and we visualise it for a simple 1-dimensional regression problem in Figure 3.4.

To effectively learn a heteroscedastic noise term, we introduce a second GP $p(r(\cdot)) = \mathcal{GP}(\mu_r, k_r(\cdot,\cdot))$ where $\mathbf{r} = [r(v_1), r(v_2), \ldots, r(v_{n_v})]$ into our model. The role of this GP is to functionally map the graph's vertices to a vector of noise terms. Unlike the GPs that we have seen so far in this thesis, we explicitly seek to model the mean function of this GP as a function of the graph, as this term allows us to control the scale of the variances' values. To ensure the noise terms are positive, we use the exponential link function, and the likelihood function in our model is now given by

$$p(\mathbf{y} \mid \mathbf{f}, \mathbf{r}) = \mathcal{N}(\mathbf{f}, \exp(\mathbf{r})^2\mathbf{I}_n)\,.$$

Introducing $\exp(\mathbf{r})$ makes inference for our model intractable. To resolve this, we

introduce a pair of variational distributions: $q(\mathbf{f})$ and $q(\mathbf{r})$. Applying the standard VI procedure outlined in Section 1.2 then yields the following ELBO

$$\mathcal{L}(q(\mathbf{f}), q(\mathbf{r})) = \log p(\mathbf{y}) - \mathrm{KL}(q(\mathbf{f}), q(\mathbf{r}) \,||\, p(\mathbf{f}, \mathbf{r} \,|\, \mathbf{y}))\,. \tag{3.5}$$

Following Lázaro-Gredilla et al. (2011), the optimal form of $q(\mathbf{f})$ is given by

$$\begin{aligned} q^\star(\mathbf{f}) &= \arg\max_{q(\mathbf{f})} \mathcal{L}(q(\mathbf{f}), q(\mathbf{r})) && (3.6)\\ &= \mathbf{C}^{-1} p(\mathbf{f}) \exp\left(\int q(\mathbf{r}) \log p(\mathbf{y} \,|\, \mathbf{f}, \mathbf{r}) \mathrm{d}\mathbf{r} \right)\,, && (3.7) \end{aligned}$$

where $\mathbf{C}$ is the normalisation constant that ensures $q^\star(\mathbf{f})$ integrates to 1. Letting $q(\mathbf{r})$ be a multivariate Gaussian with mean $\mathbf{m}_r$ and covariance $\mathbf{\Sigma}_r$, we may analytically compute the integral inside the exponent by

$$\begin{aligned} \int q(\mathbf{r}) \log p(\mathbf{y} \,|\, \mathbf{f}, \mathbf{r}) \mathrm{d}\mathbf{r} &= \int \mathcal{N}(\mathbf{r} \,|\, \mathbf{m}_r, \mathbf{\Sigma}_r) \log p(\mathbf{y} \,|\, \mathbf{f}, \mathbf{r}) \mathrm{d}\mathbf{r} && (3.8)\\ &= \log \mathcal{N}(\mathbf{r} \,|\, \mathbf{f}, \mathbf{R}) - \frac{1}{4} \mathrm{trace}(\mathbf{\Sigma}_r)\,, && (3.9) \end{aligned}$$

where $\mathbf{R}$ is a diagonal matrix whose entries are $\mathrm{diag}(\mathbf{R}) = \exp\left(\mathbf{m}_r + \mathrm{diag}(\mathbf{\Sigma}_r)\right)$. With this result, the normalisation constant $\mathbf{C}$ can be analytically computed and the corresponding ELBO is then given by

$$\mathcal{L}(q(\mathbf{r})) = \log \mathcal{N}(\mathbf{y} \,|\, \mathbf{K}_\mathbf{f} + \mathbf{R}) - \frac{1}{4} \mathrm{trace}(\mathbf{\Sigma}_r) - \mathrm{KL}(q(\mathbf{r}) \,||\, p(\mathbf{r}))\,. \tag{3.10}$$

The bound on the true marginal log-likelihood that is given by the ELBO is tightened by computing the derivative of Equation (3.10) with respect to the variational parameters and the kernel hyperparameters that are associated with f and r. A gradient-based optimiser can then be used to find the optimal values for

these parameters.

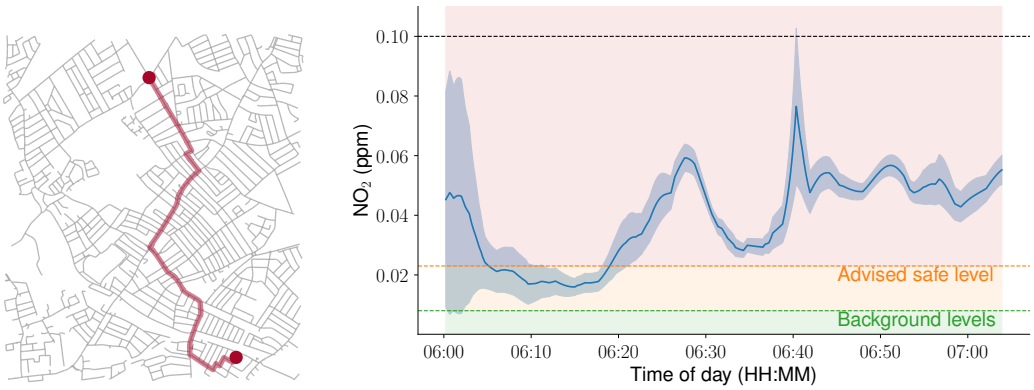


Figure 3.5: Left: Example 4.2km route that we query our Gaussian process for. Right: The NO_2 exposure that would be experienced whilst walking the journey. The shading indicates the risk of a given level. Plotted in blue is the GP's predictive mean and one standard deviation.

3.3.3 Model validation

With each observation pair (v_i, y_i) corresponding to an individual vertex and associated NO_2 value, we assess our model's performance by partitioning the dataset into a training and testing set. We hold back 137 vertices for testing, and fit our model to the remaining 548 vertices. Note that a spatial cross-validation approach would be inappropriate as we have assumed observations to be observed on a network, not a Euclidean space. To validate our model, we compare it against its Euclidean and homoscedastic analogues. We construct a Euclidean representation of the data by computing the coordinate location of each vertex, making the input space $\mathbb{R}^2$. A Euclidean Matérn kernel (Equation (1.52)) is then fit to the data and we compare these models both with and without the heteroscedastic noise term.

R^2 coefficients measure the amount of variation in the data that is explained by the model. A perfect model will attain an R^2 score of 1, whilst predicting the data's

mean $\bar{y} = n^{-1} \sum_{i=1}^{n} y_i$ everywhere will yield a score of 0. The predictive posterior density evaluated on test data quantifies how well the model explains unseen data points. We compute these metrics for each model (Table 3.1) and find that the optimal model is one with a heteroscedastic noise term and a graph kernel.

Table 3.1: R^2 coefficients and predictive posterior density scores on a held-back test set for the four models considered in Section 3.3.2. For both metrics, a higher score is better.

Kernel Domain	Noise Model	R^2	Predictive likelihood
Euclidean ($\mathbb{R}^2$)	Homoscedastic	0.636	0.511
	Heteroscedastic	0.716	0.672
Graph (V)	Homoscedastic	0.688	0.521
	Heteroscedastic	**0.722**	**0.724**

We further validate our model through an analysis of residuals in Figure 3.6. The left panel plots the predicted values $\hat{y}$ and the corresponding ground-truth value y, the middle panel shows the predicted values against the residual $y - \hat{y}$. In the right panel, we plot a histogram of the residuals. In each of the three plots, we observe that the residuals are dispersed with no obvious bias or structural error. Further, the residuals' behaviours are consistent across the training and test data, indicating that we have not overfit to the training data.

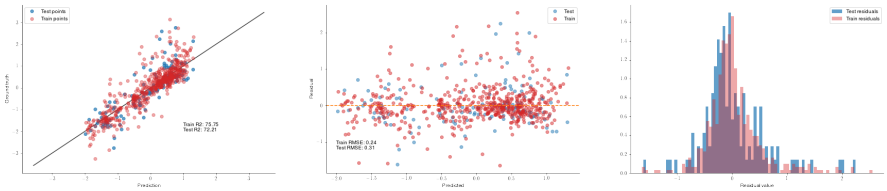


Figure 3.6: Analysis of residual plots for the heteroscedastic graph GP presented in Section 3.3.2.

The improvement in model diagnostic measures (Table 3.1) alone is enough to justify using a graph kernel. However, we would like to explicitly acknowledge

that the computational cost incurred when fitting a graph kernel scales equally to a Euclidean kernel. Furthermore, the choice of kernel domain is a modelling decision, and in this work, a graph kernel is more appropriate as it more accurately reflects the fact that observations are collected whilst a vehicle moves along the roads of Mitcham. This is of importance as the vehicles are one of the primary sources of NO_2 and, therefore, by using a graph kernel, we can better capture the relative *closeness* of locations that a vehicle has passed through, under the assumption of bi-directionality. Selecting a Euclidean kernel would imply that the closeness of locations is simply a function of distance which is untrue if we consider parks, flanks of buildings, and areas of construction; all areas where a vehicle is unable to travel through and should, therefore, be less correlated with a position on the road.

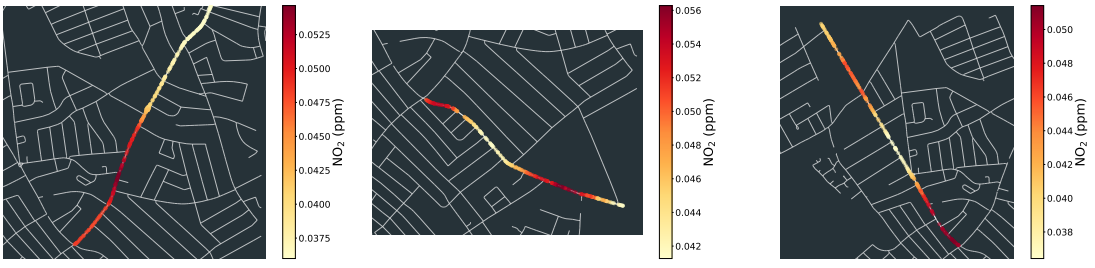
3.3.4 Nitrogen dioxide exposure levels

Here we seek to explore the NO_2 levels that an individual would be exposed to during a *typical* walking journey. To answer this question, we consider the journey plotted in orange in the left panel of Figure 3.5 and infer from our fitted GP model the NO_2 levels that would have been experienced at 6 AM. The World Health Organisation consider NO_2 levels of 0.023 ppm to be unsafe (World Health Organization, 1997). The prediction in Figure 3.5 we can see that a pedestrian would spend 51 minutes of a 64-minute walk at levels greater than this threshold. The significance of this should not be underplayed as the European commission consider more than 18 exceedances of 0.1ppm on an hourly average in a single year to be a violation of EU law.

3.3.5 Areas of greatest exposure

The fine-scale resolution our model provides allows us to precisely identify the streets whose NO_2 exposure is greatest. To do this, we evaluate the GP's predictive mean at every street for which we have at least one observation. We then compute the mean NO_2 value for each street and identify the three streets with the largest mean NO_2 value. NO_2 predictions at vertices that are adjacent to two or more streets are repeated once per street. We identify Tooting Bec Road, Trinity Road and Balham High Road as the three streets with the largest mean NO_2 values.

To further investigate the NO_2 exposure experienced along these roads, we simulate 250 additional vertices for each of the three streets mentioned above and include the vertices into our graph in an identical manner to the process described in Figure 3.3. We refit our GP to the expanded graph and plot the GP's predictive mean at the vertices associated with each of the three above streets. In Figure 3.7, we can see that despite the average exposure being large on these streets, there are areas more exposed to high NO_2 levels than others. Such analyses are invaluable from a policy-making perspective as they enable us to better understand which people are more at risk of pollution-related illnesses.



(a) Balham High Road (b) Tooting Bec Road (c) Trinity Road

Figure 3.7: The three streets in Mitcham with the largest mean NO_2. In each panel, we visualise the predictive mean along the respective street where we can see regions with larger NO_2 values.

3.4 LancasterAQ - A mobile datatset of ultrafines

We present an air quality dataset collected in Lancaster, UK that comprises of UFPs measurements, collected by a combination of vehicle-borne and bicycle-borne instrumentation. The high-resolution at which UFPs have been measured makes this a novel dataset that will enable further research into the impacts of UFPs on human health and enable more informed public policy. The aim of this paper is to provide the required knowledge to use and understand the data.

Exposure to poor air quality is a major hazard to human health with an estimated 4.2 million premature deaths globally each year resulting from outdoor air pollution (Cohen et al., 2017), including particulate matter (PM). UFPs are PM less than 0.1 μm in diameter and are linked to a range of negative health effects, including cardiovascular and pulmonary diseases (Ohlwein et al., 2019). However, UFPs are not included in current premature death estimates and are currently unregulated in air quality standards worldwide, in part, due to the lack of systematic measurement. UFPs are directly emitted from combustion sources, although they are also created by atmospheric reactions. In urban environments, the primary source of UFPs is traffic, including tailpipe, road abrasion, and non-exhaust emissions (e.g., brake and tyre wear).

UFPs concentrations vary across space and time and are influenced by physical (e.g., meteorology and atmospheric chemistry) and social (e.g., rush hour traffic and public transport use) processes. Therefore, datasets measured at a high spatiotemporal resolution are essential to understand the behaviour and impacts of UFPs. Existing air quality products range from coarse global-scale reanalysis (e.g., Inness et al., 2019; Kong et al., 2021) to small and localised low-cost

static sensor networks (e.g., Purple Air). However, these datasets seldom contain measurements of UFPs, let alone provide information at scales appropriate for their high spatiotemproal variability.

Better measurements of UFPs will improve our understanding of meteorological, climate, and social systems. Through atmospheric reactions they form important seeds for cloud creation, currently a major source of uncertainty in modelling climate change and a possible method for geoengineering climate solutions. Socially, it will be possible to quantify the effects of pollution intervention and design impactful policies, such as active traffic management, low-traffic neighbourhoods and low-emission zones.

Table 3.2: Summary of data collected from this measurement campaign including average meteorological variables temperature, humidity and surface pressure. Driving measurements are shown in blue and cycling in red.

Date	Time	Data count	Temp. (°C)	Humid. (%)	Press. (mb)
Tuesday 03/05/2022	0827-1045	9650	9.2	84.6	1010
	1353-1655	10924	12.5	77.9	1009
	0757-0953	6637	8.7	85.5	1010
Wednesday 04/05/2022	0855-1219	12256	10.5	100.0	1005
	1454-1800	11155	11.6	77.9	1006
	0854-1051	6539	10.4	100.0	1005
Thursday 05/05/2022	1013-1315	10907	12.3	83.7	1012
	1553-1858	11132	12.5	81.8	1012
	1001-1143	5391	11.5	85.7	1012
	1550-1709	4494	13.3	77.2	1013
Friday 06/05/2022	1057-1414	11823	12.3	88.8	1011
	1053-1210	4521	12.3	88.7	1011
Sunday 08/05/2022	0910-1124	8020	12.4	69.2	1017

Our new dataset of mobile UFP measurements is an important addition to existing air quality products. It provides highly resolved UFP concentrations at the street level of an under measured, but important pollutant. Our data is available through

a Python package[1] that facilitates statistical modelling from either a spatial or network perspective. Finally, as our dataset is collected in the small city of Lancaster, UK, it contrasts with existing mobile datasets, which focus on larger cities (e.g., Breathe London and C. Chen et al. (2001)).

Our mobile dataset can be used in a number of impactful ways, including the following:

- Route planning to factor in UFP exposure levels (e.g., Y. Wang et al., 2022; J. D. Smith et al., 2022).
- Decision making and experimental design for sensor placement (e.g., Diggle et al., 2010).
- Investigating causal relationships from UFPs to human behaviour and urban design (e.g., Schrotter et al., 2020).

3.4.1 Collection

UFP measurements were made by car and bicycle over 5 days during one week in May 2022 in Lancaster, UK. Measurements were made between 8am and 7pm, and were designed to maximise temporal coverage and sample the city's road and cycle networks. Measurements were made over $\sim$ 3 hours for driving and $\sim$ 90 minutes for cycling. One driving measurement campaign was also carried out on a Sunday morning to give an indicative background UFP concentration. In total, we completed 8 measurements by car and 5 by bicycle. Routes were driven with the aim of driving past 6 key waypoints: 1) the A6, which is the main road from the city centre out to the university and the M6 motorway; 2) the one-way system that goes around the city centre and is the main thoroughfare between Lancaster

[1] https://github.com/lgouldsbrough/LancasterAQ

and the adjoining urban centre of Morecambe; 3/4) Bowerham and Hala, which are both residential areas with some schools and a frequent bus route; 5) the Greyhound Bridge, which is a major bottleneck in Lancaster; and 6) the Lune Industrial Estate. The cycling routes sampled key cycling infrastructure including the canal towpath and popular cycle paths through the city centre. Where cycle paths were unavailable, cycling was done on the road.

All UFPs measurements were made using the NAQTS V2000[2]. Two devices were co-located prior to deployment for calibration. Flow and zero checks were done daily before monitoring. For the vehicle-based monitoring, a V2000 was put in the rear passenger seat with a tube out of the window facing towards the front of the vehicle, following the testing methodology outlined by Lim et al. (2022) that also used a V2000. For the bicycle-based measurements, we modified a bicycle trailer to carry the V2000. UFP loss corrections for the sample tubes were calculated before measurements. Measurements were made at 1-second time intervals and were geo-located using an onboard global positioning system (GPS).

3.4.2 Metadata

Lancaster is a city in North West England, with a population of 52234 as of the last census. All routes started and finished at Lancaster University, located approximately 5km south of the city centre. In total, we covered 397km and collected 31.5 hours (113449 data points) of on-road UFP data (113449 data points), which is summarised in Table 3.2 alongside complementary meteorological metadata collected by Lancaster University's weather station, Hazelrigg.

[2] https://www.naqts.com/our-technology/v2000/

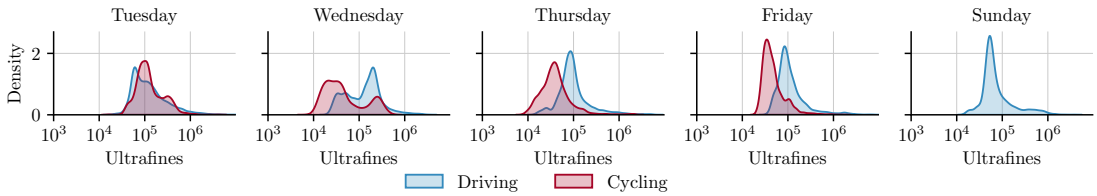


Figure 3.8: Kernel density estimate of log UFP levels stratified by the day of the week and the vehicle used to collect the readings.

3.4.3 Limitations

While the data are available at a high time resolution (1 second), we recognise that the data are temporally limited as the measurements were only taken for portions of 5 days in May 2022. This was partially mitigated by using a sliding start time each day, which varied the time windows captured. Some covariate information is available for the meteorological conditions, which are important for UFPs levels, but there are no associated traffic data available, which represents a key source for UFPs. Finally, the data are right-skewed and exhibit discontinuities which makes modelling a challenging task that we address in Section 3.4.4. Nevertheless, the data are amenable to modelling approaches on a range of supports, including spatial grids (Matheron, 1963) or networks (Pinder et al., 2022d; Nikitin et al., 2022).

3.4.4 Preliminary analysis

In this section, we will present a brief summary of the dataset. We do not aim to explore any patterns in the data, but rather highlight some of the information they contain, to enable and motivate wider community engagement.

For regression modelling, the UFP or logUFP column in Table 3.3 will be our response variable. In Figure 3.8 we explore its shape through a set of kernel density estimates: one per day of the week and vehicle type. The modelling challenge here

is immediately apparent through the measurements' right-skew.

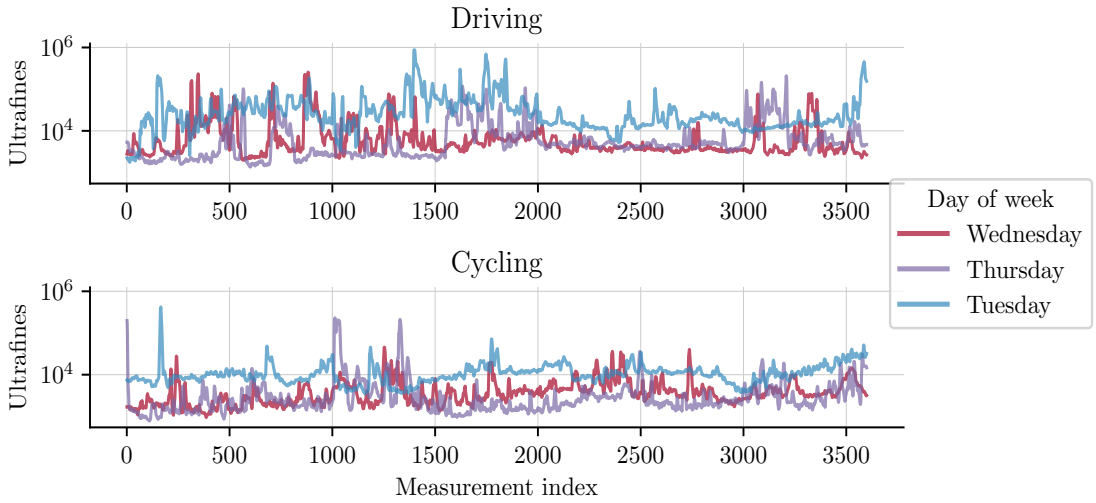


Figure 3.9: 1 hour of log-scaled UFP measurements with a 2-second moving average applied. Each line represents an individual journey.

As data collection process here is sequential, a natural task one may wish to carry out is time series modelling. In Figure 3.9 we plot the time series associated with four trips: the cycling and driving journeys that were carried out on both the Tuesday and Wednesday mornings. For easier visualisation, the data have been log-scaled and a 2-second moving average has been applied. However, even with these transformations applied, the roughness of the data is apparent.

Table 3.3: Leading 5 rows of the collected data as given by the LancasterAQ API.

Datetime	Lat	Lon	Ultrafines	Vehicle	Log ultrafines
2022-05-05 15:53:08	54.01	-2.79	6591.00	driving	8.79
2022-05-05 15:53:11	54.01	-2.79	7095.00	driving	8.87
2022-05-05 15:53:12	54.01	-2.79	7505.00	driving	8.92
2022-05-05 15:53:12	54.01	-2.79	7505.00	driving	8.92
2022-05-05 15:53:12	54.01	-2.79	7505.00	driving	8.92

3.4.5 Conclusion

To allow members of the climate science and atmospheric communities to more easily access the data, we have created and published a Python package titled `LancasterAQ` on PyPi[3]. Through this package, the UFPs data can be loaded into a Python module as either a tabular `GeoPandas` DataFrame (Jordahl et al., 2020), or a `networkx` graph object (Hagberg et al., 2008). We visualise the leading five rows of this data in tabular format in Table 3.3.

With this data, we will initially model UFP concentrations on a graph structure (Pinder et al., 2022d) to produce local exposure maps and a route planner which accounts for exposure to poor air quality. This is only possible due to the dataset's high resolution. Our motivation for publishing this data is to encourage the applied machine learning community to investigate and model questions including the influence of humans on UFPs, the links between climate, clouds and UFPs, and how public policy can mitigate impacts on our changing social and physical world.

[3]https://pypi.org/project/LancasterAQ/

Chapter 4

Probabilistic Embedding of Hypergraphs Through Gaussian Process Latent Variable Models

Chapter 3 explored how graphs and GPs can be combined to construct informative models of air pollution. In this chapter, we extend this idea by generalising the concept of a graph to a hypergraph and develop GP methodology on the hypergraph. Such a representation allows for a richer dependency structure within the data to be modelled with negligible additional computational complexity.

The methodology of this chapter contrasts with previous chapters as we depart from conditional regression modelling and instead consider unsupervised learning algorithms. More specifically, we seek to probabilistically learn a latent vector representation of a hypergraph that can be used for downstream modelling and visualisation. We demonstrate such a model on a set of political scenarios including the Congressional voting system of Peru, and the Senate committees within the United States of America's Congress.

4.1 Introduction

Modelling a complex system where there exist interactions between elements of the system requires a descriptive data representation that precisely encodes how sets of elements interact, if at all. A graph is a popular choice for modelling such datasets as it enables individual elements to be represented by a *vertex* and relations between any two elements are represented through an *edge*. However, as we further our understanding of real-world systems, the need to characterise richer and more complex relational structures within a graph becomes increasingly pertinent. The limiting behaviour of simple graphs is quickly realised here as we are constrained to only ever modelling pairwise interactions between elements within our systems, whereas in many real-world systems the interactions truly occur at a group level. To resolve this issue, a hypergraph is often used to model data where there exists higher-order interaction structures within the data. Hypergraphs are a powerful generalisation of simple graphs that enable higher-order, complex systems to be represented. Hypergraphs have been shown to be useful in the domains of video segmentation (Huang et al., 2009), collaboration networks (Patania et al., 2017), and cellular networks (Klamt et al., 2009). However, the use of hypergraphs is often impaired by the less established hypergraph modelling literature.

Despite their capacity to model complex systems, two limitations of hypergraphs are the difficulty in visualising the hypergraph's structure and how the hypergraph can be used within a machine learning modelling framework. Whilst small hypergraphs can be visualised using modern scientific computing libraries, this task becomes increasingly difficult as the hypergraph's size grows. Recent years have seen a rapid blossoming in the number of algorithms that have been adapted to operate on a graph (Bronstein et al., 2021). However, very few of these methods have been extended to the hypergraph setting.

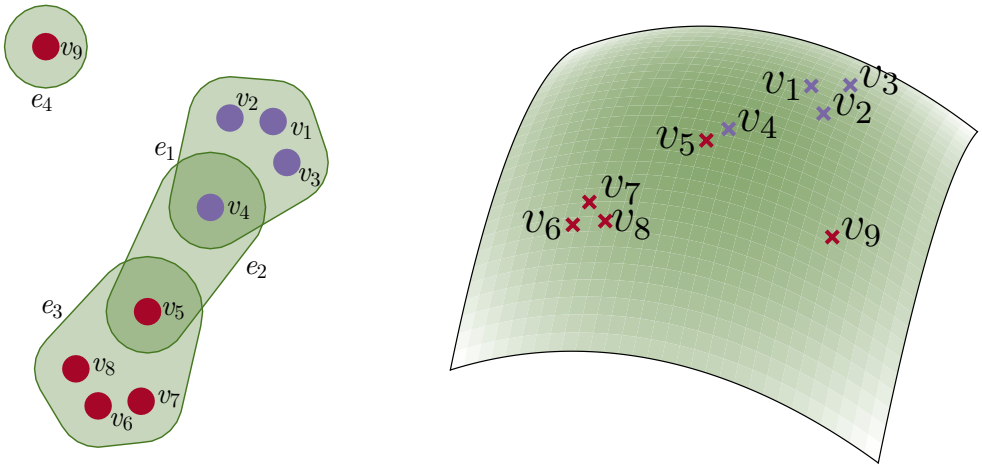


Figure 4.1: Given a hypergraph (left panel, formally introduced in Section 4.2.3), we seek to embed the vertices of the hypergraph into a low-dimensional latent space.

Within this work, we seek to learn a low-dimensional, latent vector representation of each vertex within our hypergraph (visualised in Figure 4.1). The hypergraph's latent representation may then be used to more easily visualise the relational structure present within the hypergraph and identify communities or clusters whose constituent vertices exhibit similar behaviour within the hypergraph. Further, we can use the latent vector that is associated with each vertex for downstream modelling tasks, such as vertex classification, by making use of the rich software ecosystem that exists for vector-valued inference e.g., Scikit-learn (Pedregosa et al., 2011). This is of enormous utility as it obviates the need to design complex analogues of existing techniques so that they are amenable to being applied to a hypergraph. For example, if we had a hypergraph whereby a subset of the hypergraph's vertices was labelled with a binary class, we could embed the hypergraph into a latent space and then use a logistic regression model to classify the vertices of the hypergraph that were previously unlabelled. Currently, adapting models such as a logistic classifier to operate on a hypergraph is a non-trivial task.

Our latent representation requires that the original hypergraph's dependency structure is preserved in the latent vector space. For example, two vertices with similar interactions in the hypergraph should be *close* to one another in vector space whereas two vertices that exhibit very different behaviour within the hypergraph should be far apart in the vector space. Hypergraphs can often exceed a few hundred vertices, and we therefore also require our latent representation to be tractable to compute for large hypergraphs.

Building on recent developments in the GP literature, we propose the use of a GPLVM (Lawrence et al., 2005) to learn a latent space representation of a hypergraph. Such an approach is well motivated as through the use of a GP we can learn a vector representation of each vertex through a nonparametric non-linear mapping. Furthermore, we can reflect our beliefs surrounding the smoothness of the hypergraph's latent structure through kernel selection within our GP. Within the graph theory literature, analogies of traditional dimensionality reduction algorithms, such as principal component analysis, have been adapted for graphs (Jiang et al., 2013). More recently, techniques that rely on geometric deep learning architectures have been proposed (Perozzi et al., 2014). However, neither principal component analysis or the the existing deep learning approaches are able to probabilistically infer a latent space and enable the practitioner's beliefs to be incorporated into the model, as is the case with our proposed methodology.

To illuminate the utility provided by our model, we will apply our model to a range of political datasets where the vertices of the hypergraph are political figures. Political network analysis is a common task in the domain of graph representation learning, however, as we shall go on to see in Section 4.4, modelling the data as a hypergraph reveals a number of interesting insights.

4.2 Background

4.2.1 Political latent space modelling

The task of inferring a latent space representation of a graph describing political entities is not novel and we here review some of the existing literature on this topic. This serves to not only establish our work within the literature but also provide a concrete context with which all subsequent modelling choices can be associated. To the best of our knowledge, all of this literature considers a graph, not a hypergraph, to model the data.

Edges within a graph encode a relationship between two vertices. However, it is reasonable to assume that "*the friend of my friend is also a friend of mine*" (Rapoport, 1953). Under this assumption, it is reasonable to believe that there exists a latent representation of the graph whereby similar vertices will be close together. Within the domain of politics, political figures become the vertices of a network (Ward et al., 2011). In the United States of America (USA), two politicians within Congress were assumed to interact if they voted the same way on any given roll call vote (Yu et al., 2020). The resulting network was then used to perform Bayesian factor analysis. Meanwhile, Sewell et al. (2015) propose a latent space for a dynamical graph where an interaction between two politicians is present if they co-sponsored a bill. Subsequent work extended this model to allow for weighted edges to be incorporated into the model, thus assuming that not all interactions were equal in weight (Sewell et al., 2016). In such models, an interaction is represented by an edge in the graph.

Outside of the USA, Paul et al. (2016) devised a discrete latent space model of Members of Parliament (MPs) in the UK where each vertex is assigned a discrete class, rather than a continuous vector value. Interactions in this model sought

to encode the strength of the relationship between any pair of MPs through their interactions on Twitter. Similarly, Barberá (2015) modelled the political ideologies of the people within six countries based on the people that they interacted with and followed on Twitter.

4.2.2 Whittle Matérn fields

The presentation of a GP given in this thesis has so far viewed them through a functional representation (Rasmussen et al., 2006). However, representing a GP as a stochastic partial differential equation (SPDE) provides an alternative viewpoint that will be helpful in the sequel (Särkkä et al., 2019).

A SPDE equates the differential of a function f with a stochastic process, such as a white noise process, $\mathcal{W}$. For an SPDE, this can be written as

$$\mathcal{T} f(\cdot) = \mathcal{W}(\cdot)\,, \tag{4.1}$$

The differential operator $\mathcal{T}$ given here is any operator that is defined as a function of the differentiation operator (Lototsky et al., 2017). Examples of $\mathcal{T}$ are therefore the first or second derivative operator or the Laplacian. The function f that satisfies Equation (4.1) is termed the solution to the SPDE. Considering Equation (4.1) from the Gaussian processes perspective, we can see that by letting f be a zero-mean GP with kernel operator k, then solutions of Equation (4.1) will provide us with a tool to uniquely identify GP models.

When the differential operator in Equation (4.1) is known, the analytical form of the SPDE's solution can be written as

$$f(x) = \int g(x - v)\mathcal{W}(v)\mathrm{d}v\,. \tag{4.2}$$

The function g is known as Green's function and can be found by computing the inverse Fourier transform of $\mathcal{T}f$ (Green, 1889). With this in mind, one can show that the covariance between $f(x)$, and $f(y)$ can be computed through

$$
\begin{aligned}
k(x,y) &= \mathbb{E}[f(x)f(y)] && (4.3)\\
&= \mathbb{E}\left[\int g(x-u)\mathcal{W}(u)\mathrm{d}u \int g(y-v)\mathcal{W}(v)\mathrm{d}v\right] && (4.4)\\
&= \mathbb{E}\left[\int g(x-u)g(y-u)\mathrm{d}u\right] && (4.5)\\
&= \int g(x-u)g(y-u)\mathrm{d}u && (4.6)
\end{aligned}
$$

where the penultimate line is obtained through Itô's isometry (Oksendal, 2013).

From this, it can be seen that the exact form of $\mathcal{T}$ and kernel function k of the GP f are intimately linked, and different forms of $\mathcal{T}$ will yield different kernels. For Euclidean domains $\mathcal{X} = \mathbb{R}^d$, the seminal work of Whittle (1963) showed that when $\mathcal{T}$ is the continuous Laplacian $\boldsymbol{\Delta}$, the following provides a solution to Equation (4.1)

$$
\left(\frac{2\nu}{\ell^2} - \boldsymbol{\Delta}\right)^{\nu/2+d/4} f = \mathcal{W}\,, \tag{4.7}
$$

where $\boldsymbol{\Delta}$ is the Laplacian. This can then be rearranged to give a *Matérn GP process*

$$
f \sim \mathcal{GP}\left(0, \left(\frac{2\nu}{\ell^2} - \boldsymbol{\Delta}\right)^{-\nu-d/2}\right). \tag{4.8}
$$

It is through this connection that sparse precision matrices for Gaussian Markov random fields can be efficiently constructed in the seminal work of Lindgren et al. (2011). Since then, this connection has been further developed for GP modelling on Riemannian manifolds (Borovitskiy et al., 2020), graphs (Borovitskiy et al., 2021;

Bolin et al., 2022), and spatial domains (Bolin, 2014). Further, an alternative line of work seeks to parameterise neural networks such that they *learn* the solutions to the differential equation (Salvi et al., 2021). For a contemporary review of these works and more, see Lindgren et al. (2022).

4.2.3 Hypergraphs

A *hypergraph* is a generalisation of a graph in which interactions occur among an arbitrary set of nodes (Berge, 1973). We visualise this in Figure 4.2 where we see a hypergraph comprised of four hyperedges $E = \{e_1, e_2, e_3, e_4\}$, and six vertices $V = \{v_1, v_2, \ldots, v_6\}$. Each hyperedge $e_i \in E$ corresponds to a subset of vertices from V with no repeated elements. A hyperedge e_i is said to be *weighted* when it has a positive value $w(e_i)$ assigned to it and *incident* to a vertex v_j if $v_j \in e_i$. A hypergraph comprised of N vertices and D hyperedges can be represented by an incidence matrix $\mathbf{H} \in \{0, 1\}^{N \times D}$ with $(i, j)^{\text{th}}$ entry equal to 1 if v_i is incident to e_j and 0 otherwise.

Representing network data as a hypergraph provides a flexible and descriptive framework to encode higher-order relationships within a graph-like structure. This is because, within a single hyperedge, we assume that a *group-level* interaction occurs amongst the set of constituent vertices. Modelling such interactions would not be possible in a simple graph as it would be impossible to disentangle multiple pairwise connection from a fully-connected group-level interaction. Returning to our early example of political networks, modelling the US senate as a hypergraph would allow us to represent the Congresspeople as vertices and senate committees as hyperedges. A vertex would then be incident to a hyperedge if the respective Congressperson is a member of the given committee. Similarly, a hyperedge could be an individual piece of legislation whereby vertices belong to the hyperedge

if the respective Congressperson voted in favour of the legislation. We study these scenarios in Section 4.4. Beyond politics, hypergraphs have been successfully applied to friendships within social networks (Andjelković et al., 2015), ecological species equilibria (De Oliveira et al., 2000), and protein modelling (Ruepp et al., 2010).

Every hypergraph also admits a bipartite representation (Battiston et al., 2020) in which each hyperedge is assigned a node, and an edge from a node vertex to a hyperedge vertex indicates incidence (see Figure 4.2). Since a hypergraph generalises the graph representation, it follows that these structures coincide when each hyperedge contains precisely two nodes.

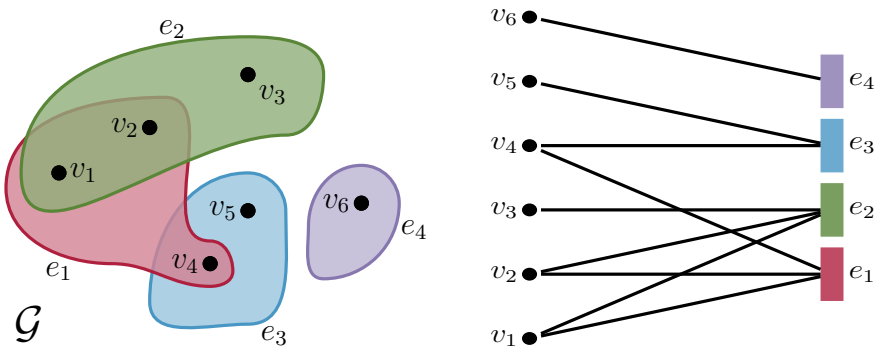


Figure 4.2: A hypergraph comprised of four hyperedges among six vertices and the corresponding bipartite representation. See Section 2.1.2 in (Bretto, 2013) for details of this relationship.

Unlike pairwise graphs, the Laplacian for hypergraphs has no unique form. However, for this work, we rely on the form given in (D. Zhou et al., 2006) which can be written as

$$\boldsymbol{\Delta} = \mathbf{I} - \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}, \tag{4.9}$$

where $\mathbf{D}_v$ and $\mathbf{D}_e$ are diagonal matrices with non-zero entries containing the node

and hyperedge degrees, respectively. We write

$$[\mathbf{D}_v]_{i,i} = \sum_{e \in E} w(e)h(v_i, e) \qquad [\mathbf{D}_e]_{i,i} = \sum_{v \in V} h(v, e_i), \tag{4.10}$$

where $h(v, e) = 1$ if v is incident to e i.e., $v \in e$. In this work, we only considered unweighted hypergraphs (i.e. $w(e_i) = 1$ for $1 \leq i \leq M$); however, extending our framework to accommodate weighted hyperedges poses no challenge as, through an appropriate weight function w, the matrices $\mathbf{D}_v$ and $\mathbf{D}_e$ can be recomputed. From here, all subsequent methodology is unchanged.

4.2.4 Latent variable models

The literature on probabilistic generative models is vast, spanning probabilistic principal component analysis (PCA) (Tipping et al., 1999), the GPLVM (Lawrence et al., 2005), deep Boltzmann machines (Salakhutdinov et al., 2009), energy models (Ngiam et al., 2011), variational autoencoders (Kingma et al., 2013), generative adversarial networks (Goodfellow et al., 2014), normalising flows (Rezende et al., 2015), and diffusion models (J. Ho et al., 2020). Common amongst all of these methods is the goal of learning a latent representation of a complex dataset from which we can more easily visualise the underlying structure of the data, draw a sample from the latent space, and use the latent space for downstream modelling. In this work, we focus our attention on the GPLVM which we shall now proceed to introduce, starting from the perspective of probabilistic PCA.

Probabilistic principal component analysis Probabilistic PCA (Tipping et al., 1999) is a dimensionality reduction method that relies on two key assumptions:

1. The observed data is a linear function of some latent variables.

2. The latent variables are random variables that follow zero-mean and identity

covariance multivariate normal.

For an observed dataset $\mathbf{Y} \in \mathbb{R}^{N \times D}$ (so, this applies to continuous data rather than binary), we assume that they are generated given a linear function applied to a set of T-dimensional latent variables $\mathbf{X} \in \mathbb{R}^{N \times T}$ where $T \ll D$. For each latent variable, we assign the prior distribution

$$p(\mathbf{x}_{i,:}) = \mathcal{N}(\mathbf{0}, \mathbf{I}_T)\,. \tag{4.11}$$

We define the conditional distribution of an observed data point, given its respective latent variable, to also be a Gaussian

$$p(\mathbf{y}_{i,:} \,|\, \mathbf{x}_{i,:}) = \mathcal{N}(\mathbf{y}_{i,:} \,|\, \mathbf{W}\mathbf{x}_{i,:}, \sigma^2 \mathbf{I}_D)\,, \tag{4.12}$$

where $\mathbf{W} \in \mathbb{R}^{D \times T}$ is a linear map and σ^2 is a noise term. Using Equations (4.11)–(4.12), we can write the generative model as

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\epsilon}\,, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D)\,. \tag{4.13}$$

To estimate the optimal parameters $\mathbf{W}$ and σ^2, we first marginalise out the latent variable $\mathbf{X}$ from Equation (4.12) to give the distribution of each observation as

$$p(\mathbf{y}_i) = \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}_D)\,. \tag{4.14}$$

The model's marginal log-likelihood is optimised with respect to $\mathbf{W}$ and σ^2. When $\sigma^2 \to 0$, the quantity in Equation (4.14) simplifies to regular PCA.

The intuition given by Bishop (2006) is a helpful note to close this section on:

> *We can think of the distribution $p(\mathbf{y}_i)$ as being defined by taking an*

isotropic Gaussian 'spray can' and moving it across the principal subspace spraying Gaussian ink with density determined σ^2 and weighted by the prior distribution. The accumulated ink density gives rise to a 'pancake' shaped distribution representing the marginal density $p(\mathbf{y}_i)$.

Gaussian process latent variable models In probabilistic PCA we first integrate out the latent variable $\mathbf{X}$, then optimise the mapping matrix $\mathbf{W}$. As a matrix, $\mathbf{W}$ can only represent a linear relationship between our latent variable and the observed data. The GPLVM offers a solution to this problem by optimising $\mathbf{X}$ and marginalising $\mathbf{W}$. This distinction is highlighted in Table 4.1.

Table 4.1: The different treatments that probabilistic PCA and the GPLVM apply to the model's latent variable and the observed data.

	Latent variable ($\mathbf{X}$)	Mapping operator ($\mathbf{W}$)
Probabilistic PCA	Marginalise	Optimise
GPLVM	Optimise	Marginalise

Following the GP notation that has been used so far in this thesis, the GPLVM places independent GP priors $w(\mathbf{x}_{:,i}) \sim \mathcal{GP}(\mathbf{0}, \mathbf{K_{ff}})$ on each column of $\mathbf{W}$.

$$p(\mathbf{w}_{:,i} \,|\, \mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{w}_{:,i} \,|\, \mathbf{0}, \mathbf{K_{ff}})\,. \tag{4.15}$$

A prior distribution is placed on the latent variable $\mathbf{X}$ which we will specify to be $p(\mathbf{X}) = \mathcal{N}(\mathbf{0}, \mathbf{I}_N)$.

If we assume our observations have an additive Gaussian noise model, then a joint

model for the GPLVM takes the form

$$p(\mathbf{Y}, \mathbf{W}, \mathbf{X} \,|\, \boldsymbol{\theta}, \sigma^2) = p(\mathbf{Y} \,|\, \mathbf{W}, \sigma^2) p(\mathbf{W} \,|\, \mathbf{X}, \boldsymbol{\theta}) p(\mathbf{X}) \\ = \prod_{d=1}^{D} \underbrace{p(\mathbf{y}_{:,d} \,|\, \mathbf{w}_{:,d}, \sigma^2)}_{\text{Observation Likelihood}} \underbrace{p(\mathbf{w}_{:,d} \,|\, \mathbf{X}, \boldsymbol{\theta})}_{\text{GP prior}} \underbrace{p(\mathbf{X})}_{\text{Latent space prior}} . \tag{4.16}$$

Analytically marginalising the mapping matrix $\mathbf{W}$ and the latent variable $\mathbf{X}$ from Equation (4.16) is computationally intractable due to the interplay that exists between these two quantities. Unlike regular GP regression, $\mathbf{X}$ is now a random variable and we are therefore required to propagate its uncertainty through the GP. Problems arise when evaluating the marginal log-likelihood of a GP as $\mathbf{X}$ appears non-linearly in the $(\mathbf{K_{ff}} + \sigma^2 \mathbf{I}_N)^{-1}$ matrix. Instead, to allow us to analytically marginalise $\mathbf{W}$, we condition on the latent variable

$$p(\mathbf{Y} \,|\, \mathbf{X}, \boldsymbol{\theta}, \sigma^2) p(\mathbf{X}) = \left(\int p(\mathbf{Y} \,|\, \mathbf{W}, \sigma^2) p(\mathbf{W} \,|\, \mathbf{X}, \boldsymbol{\theta}) \mathrm{d}\mathbf{W} \right) p(\mathbf{X}) \tag{4.17}$$

where

$$p(\mathbf{Y} \,|\, \mathbf{X}, \boldsymbol{\theta}, \sigma^2) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_{:,d} \,|\, \mathbf{0}, \mathbf{K_{ff}} + \sigma^2 \mathbf{I}_N) . \tag{4.18}$$

If we collect our latent variables $\mathbf{X}$ and the model hyperparameters $\{\boldsymbol{\theta}, \sigma^2\}$ into the set $\boldsymbol{\psi} = \{\mathbf{X}, \boldsymbol{\theta}, \sigma^2\}$ then we can learn the optimal point estimates $\boldsymbol{\psi}^\star$ for each parameter using gradient-based optimisation of

$$\boldsymbol{\psi}^\star = \arg\max_{\boldsymbol{\psi}} p(\mathbf{Y} \,|\, \mathbf{X}, \boldsymbol{\theta}, \sigma^2) p(\mathbf{X}) . \tag{4.19}$$

The learned values of $\mathbf{X}^\star$ give the set of latent coordinates with a one-to-one mapping between $\mathbf{X}^\star$ and $\mathbf{Y}$.

Connection to probabilistic PCA If we let the kernel function that produces $\mathbf{K_{ff}}$ in Equation (4.15) be a linear kernel i.e., $k(x, x') = \alpha^2 xx'$ where α^2 is the kernel's variance, then inference in the GPLVM is equivalent to probabilistic PCA (see Figure 4.3). The advantage of the GPLVM is that for non-linear kernels, such as the Matérn family of kernels, it is possible to learn a non-linear mapping between the observed data and latent variable.

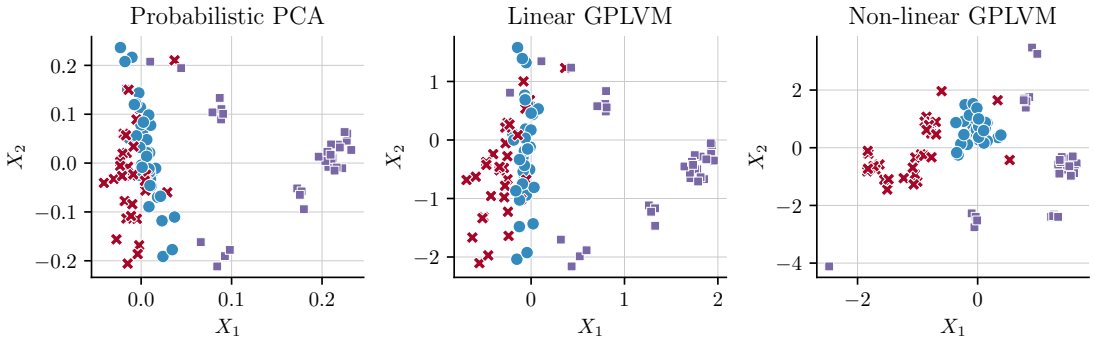


Figure 4.3: Relationship between probabilistic PCA (left) and the GPLVM with a linear kernel (centre) and a non-linear squared exponential kernel (right). Each panel shows the inferred 2-dimensional latent space computed from the original 8-dimensional Oil dataset (Bishop et al., 1993).

4.3 Embedding hypergraphs through Gaussian process latent variable models

In this section, we outline our main contribution: a GPLVM model that is capable of embedding the vertices of a hypergraph into a latent space. To achieve this, we first provide a unifying framework for defining a family of kernel functions on the vertices of a hypergraph. We then incorporate this kernel function into a GPLVM model to learn a latent space embedding of a hypergraph. To resolve the binary nature of the hypergraph, we derive an ELBO term that enables tractable inference in our model.

4.3.1 Kernels on hypergraphs through regularisation functions

We start the presentation of our embedding framework by generalising recent developments in the GP literature (Borovitskiy et al., 2021) through regularisation theory. We then extend this framework from the context of graphs to hypergraphs, the result being that we can define kernel functions $k : V \times V \to \mathbb{R}$ on the vertices of our hypergraph through the following result

Theorem 4.3.1 (Smola et al. (2003)) *Let $r : \mathbb{R} \to \mathbb{R}_{>0}$ be a monotonically increasing, strictly positive function whose evaluation on a normalised Laplacian matrix $\mathbf{\Delta}$ yields the regularisation matrix $\mathbf{R} = r(\mathbf{\Delta})$. The corresponding kernel matrix is given by $\mathbf{K} = r^{-1}(\mathbf{\Delta})$*

$$\begin{aligned}\mathbf{K} &= \mathbf{U}\mathbf{R}^{-1}\mathbf{U}^{\top} \\ &= \mathbf{U}r^{-1}(\mathbf{\Lambda})\mathbf{U}^{\top}\,,\end{aligned} \tag{4.20}$$

where $\mathbf{U}$ is an orthonormal matrix of eigenvectors, and $\mathbf{\Lambda}$ a diagonal matrix whose elements are the set of eigenvalues.

Intuitively, Theorem 4.3.1 allows us to compute a covariance matrix on the vertices of a graph, by specifying a regularisation function r which we can invert either exactly or using Moore-Penrose inversion. Applying this inverse function element-wise to the entries of a graph's Laplacian matrix will then yield a valid kernel matrix. As we will now go on to demonstrate, this is a powerful tool, as it provides us with the freedom to define a range of valid kernel functions by simply specifying a regularisation function.

We are interested in learning a function $f : V \to \mathbb{R}$ that is defined on

the hypergraph's vertex set. With this in mind, the discrete hypergraph Laplacian $\mathbf{\Delta} \in \mathbb{R}^{N \times N}$ can be used to quantify the smoothness of a function $\mathbf{f} = [f(v_1), f(v_2), \ldots, f(v_N)]$. To regularise the degree to which rough functions are penalised through the Laplacian, we can consider a regularisation function $r : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$. Regularising the Laplacian can then be done through

$$r(\mathbf{\Delta}) = \mathbf{U} r(\mathbf{\Lambda}) \mathbf{U}^\top . \tag{4.21}$$

where $\mathbf{U}$ are the eigenvectors and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ the diagonal matrix of eigenvalues corresponding to the Laplacian's eigendecomposition.

When designing a regularisation function, we must ensure that its codomain is non-negative to ensure we do not receive a negative eigenvalue. Further, eigenvectors that correspond to larger eigenvalues will be rougher than those eigenvectors corresponding to smaller eigenvalues. We require our regularisation function to be monotonically increasing in $\mathbb{R}_{>0}$ to ensure that eigenvectors corresponding to larger eigenvalues are penalised more than the eigenvectors corresponding to smaller eigenvalues. We summarise three commonly used regularisation functions in Table 4.2.

Table 4.2: Three commonly used regularisation functions r and their corresponding inverse applied to a single eigenvalue λ.

	$r(\lambda)$	$r^{-1}(\lambda)$
Diffusion process	$\exp\left(\frac{\sigma^2}{2\lambda}\right)$	$\exp\left(\frac{-\sigma^2}{2\lambda}\right)$
p-step random walk	$(\alpha\mathbf{I} - \lambda)^{-p}$	$(\alpha\mathbf{I} - \lambda)^{p}$
Inverse cosine	$\cos\left(\frac{\lambda\pi}{4}\right)^{-1}$	$\cos\left(\frac{\lambda\pi}{4}\right)$

If we are to define a regularisation function to be

$$r(\lambda) = \left(\frac{2\nu}{\ell^2} + \lambda\right)^{\nu/2}, \tag{4.22}$$

for smoothness parameter $\nu \in \mathbb{R}_{>0}$ and lengthscale parameter $\ell \in \mathbb{R}_{>0}$, then we can compute the kernel's associated Gram matrix $\mathbf{K}_{vv}$ by

$$\begin{aligned}\mathbf{K}_{vv} &= \mathbf{U} r^{-1}(\mathbf{\Lambda})\mathbf{U}^\top \\ &= \mathbf{U}\left(\frac{2\nu}{\ell^2} + \mathbf{\Lambda}\right)^{-\nu/2}\mathbf{U}^\top .\end{aligned} \tag{4.23}$$

The value of Equation (4.22) will increase as the size of the eigenvalue λ increases. However, through ν we are able to control the smoothness of functions realised on the hypergraph and ℓ allows us to control the regularisation's curvature. Rewriting Equation (4.1) in terms of Equation (4.23) where $\mathbf{W}$ is a finite-dimensional multivariate Gaussian, we have the SPDE

$$\left(\frac{2\nu}{\ell^2} + \mathbf{\Delta}\right)^{\nu/2}\mathbf{f} = \mathbf{W}, \tag{4.24}$$

which we can see is identical in form to the Matérn process defined in Equation (4.7), except the continuous Euclidean Laplacian has now been replaced by the discrete graph Laplacian. We can rewrite Equation (4.24) as a GP prior

$$p(\mathbf{f}) = \mathcal{N}\left(\mathbf{0}, \left(\frac{2\nu}{\ell^2} + \mathbf{\Delta}\right)^{-\nu}\right), \tag{4.25}$$

noting that the $d/2$ term that ensure regularity has in the Euclidean support is not required for the hypergraph (Borovitskiy et al., 2021). From this point, regular GP inference can be achieved using the outline given in Section 1.3.

4.3.2 Variational bound

We will now combine the previous sections that consider hypergraphs and kernels functions on the vertices of a hypergraph to propose a model that can embed the hypergraph's vertices into a latent space where vertex-to-vertex similarity is established using an appropriate kernel. To achieve this, we represent a hypergraph through its incidence matrix $\mathbf{H} \in \{0,1\}^{N\times D}$ and probabilistically learn a latent embedding $\mathbf{X} \in \mathbb{R}^{N\times T}$ where $T \ll D$ using the GPLVM model presented in Section 4.2.4. Therefore, the goal is to infer a projection matrix $\mathbf{W} \in \mathbb{R}^{D\times T}$ that maps the data into a lower-dimensional space.

To achieve this, we build upon the GPLVM that was presented in Section 4.2.4 where the observed data $\mathbf{Y}$ is now our hypergraph's incidence matrix $\mathbf{H}$. In a GPLVM, a maximum a posteriori estimate of $\mathbf{X}$ is obtained jointly with the model's hyperparameters using gradient-based optimisation (Lawrence et al., 2005). However, these models assume that the likelihood distribution $p(\mathbf{H} \,|\, \mathbf{X})$ is Gaussian, making the GP's marginal likelihood distribution analytical. For unweighted hypergraphs, the incidence matrix is binary, meaning that our likelihood is instead a Bernoulli distribution. To resolve this, we will now proceed to derive a variational scheme that enables tractable inference in models where the likelihood function is non-Gaussian. Existing work by S. Murray et al. (2018) considered a similar problem where observed dataset being modelled by the GPLVM had a mixture of data types. Meanwhile, Ramchandran et al. (2021) considered incomplete medical data, and Lalchand et al. (2022) derived a stochastic bound that uses a deep learning model to amortised inference in the latent space. However, none of these works have considered the problem of embedding a hypergraph into a latent space; the canonical issue being broached in this work. Further, whilst the work presented here considers Bernoulli data, we

would stress that this scheme can be used for other non-Gaussian distributions.

When the likelihood function $p(\mathbf{H} \mid \mathbf{X})$ is non-Gaussian, the optimisation routine given in Equation (4.19) is intractable. In this section, we derive a variational bound that can be evaluated for any likelihood function that belongs to the exponential family. We, therefore, redefine the likelihood function in the general form

$$p(\mathbf{H} \mid \mathbf{W}) = \prod_{n=1}^{N} p(\mathbf{h}_{n,:} \mid \phi(\mathbf{w}_{n,:})) \,, \tag{4.26}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is the link function corresponding to our choice of noise model. For the hypergraph's incidence matrix, a correctly specified model would assume a Bernoulli likelihood with a logistic or probit link function. Similarly, for a weighted incidence matrix whose weights have been normalised to $[0, 1]$, a likelihood with support on $[0, 1]$, such as the Beta likelihood with a logistic link function would be appropriate.

To resolve this, we augment our joint GPLVM using a set of $\mathbf{Z} \in \mathbb{R}^{M \times T}$ *latent inputs* with corresponding outputs $\mathbf{U} = w(\mathbf{Z})$ in a similar fashion to the sparse GPs introduced in Sections 1.3.5–1.3.6 to give

$$\begin{aligned} &p(\mathbf{H}, \mathbf{W}, \mathbf{U}, \mathbf{X} \mid \mathbf{Z}) \\ &\quad = p(\mathbf{H} \mid \mathbf{W}) p(\mathbf{W} \mid \mathbf{U}, \mathbf{X}, \mathbf{Z}) p(\mathbf{U} \mid \mathbf{Z}) p(\mathbf{X}) \qquad (4.27) \\ &\quad = \left(\prod_{d=1}^{D} \underbrace{p(\mathbf{h}_{:,d} \mid \phi(\mathbf{w}_{:,d}))}_{\text{Observation likelihood}} \underbrace{p(\mathbf{w}_{:,d} \mid \mathbf{u}_{:,d}, \mathbf{X}, \mathbf{Z})}_{\text{Conditional prior}} \underbrace{p(\mathbf{u}_{:,d} \mid \mathbf{Z})}_{\text{Inducing prior}} \right) \underbrace{p(\mathbf{X})}_{\text{Latent prior}} \,. \qquad (4.28) \end{aligned}$$

The benefits of this augmentation are twofold. Firstly, the introduction of the inducing variables enables more efficient inference. Secondly, unlike each latent variable $\mathbf{x}_i \in \mathbf{X}$ which is a random variable, each inducing input $\mathbf{z}_i \in \mathbf{Z}$ is a T-

dimensional vector. Because of this, we can optimise their value by computing derivatives of the ELBO with respect to $\mathbf{Z}$. This differs from the regular GPLVM where we are forced to conduct maximum a posteriori (MAP) inference on $\mathbf{X}$.

Our model's posterior is defined by $\mathbf{X}, \mathbf{W}, \mathbf{U}$ with corresponding marginal log-likelihood quantity

$$\log p(\mathbf{H}) = \log \int p(\mathbf{W} \,|\, \mathbf{X}, \mathbf{U}) p(\mathbf{H} \,|\, \phi(\mathbf{W})) p(\mathbf{X}) p(\mathbf{U}) \mathrm{d}\mathbf{X} \mathrm{d}\mathbf{W} \mathrm{d}\mathbf{U} \,. \tag{4.29}$$

To be able to derive a tractable lower bound on the marginal log-likelihood, we introduce the following variational approximation to the true posterior distribution

$$q(\mathbf{W}, \mathbf{U}, \mathbf{X}) = p(\mathbf{W} \,|\, \mathbf{X}, \mathbf{U}) q(\mathbf{X}) q(\mathbf{U}) \,, \tag{4.30}$$

where

$$q(\mathbf{U}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{u}_{:,d} \,|\, \boldsymbol{\mu}_d, \Sigma_d) \tag{4.31}$$

$$q(\mathbf{X}) = \prod_{n=1}^{N} \prod_{t=1}^{T} \mathcal{N}(x_{n,t} \,|\, m_{n,t}, s_{n,t}) \tag{4.32}$$

where $\boldsymbol{\mu}, m$ are variational mean parameters and Σ, s are variational variance and variance parameters, respectively. We can now bound Equation (4.29) using

Jensen's inequality

$$\log p(\mathbf{H}) = \log \int p(\mathbf{X})p(\mathbf{U})p(\mathbf{W} \mid \mathbf{X}, \mathbf{U})p(\mathbf{H} \mid \phi(\mathbf{W})) \frac{q(\mathbf{X}, \mathbf{W}, \mathbf{U})}{q(\mathbf{X}, \mathbf{W}, \mathbf{U})} \mathrm{d}\mathbf{X}\mathrm{d}\mathbf{W}\mathrm{d}\mathbf{U} \tag{4.33}$$

$$\geq \int q(\mathbf{X})q(\mathbf{U})q(\mathbf{W} \mid \mathbf{X}, \mathbf{U}) \log \frac{p(\mathbf{X})p(\mathbf{U})p(\mathbf{W} \mid \mathbf{X}, \mathbf{U})p(\mathbf{H} \mid \phi(\mathbf{W}))}{q(\mathbf{X})q(\mathbf{U})p(\mathbf{W} \mid \mathbf{X}, \mathbf{U})} \tag{4.34}$$

$$= \underbrace{\int q(\mathbf{X}) \log \frac{p(\mathbf{X})}{q(\mathbf{X})} \mathrm{d}\mathbf{X}}_{-\mathrm{KL}(q(\mathbf{X}) \,||\, p(\mathbf{X}))} + \underbrace{\int q(\mathbf{U}) \log \frac{p(\mathbf{U})}{q(\mathbf{U})} \mathrm{d}\mathbf{U}}_{-\mathrm{KL}(q(\mathbf{U}) \,||\, p(\mathbf{U}))}$$
$$+ \mathbb{E}_{q(\mathbf{x}_n)q(\mathbf{u}_d)p(w_{n,d} \mid \mathbf{u}_d, \mathbf{x}_n)} \left[\log p(h_{n,d} \mid \phi(w_{n,d}))\right] \tag{4.35}$$

$$\approx -\mathrm{KL}(q(\mathbf{X}) \,||\, p(\mathbf{X})) - \mathrm{KL}(q(\mathbf{U}) \,||\, p(\mathbf{U}))$$
$$+ \frac{1}{S} \sum_{s=1}^{S} \log p(h_{n,d} \mid \phi(w_{n,d}^{(s)})) \,. \tag{4.36}$$

The approximation given in Equation (4.36) is a Monte-Carlo approximation evaluated over S samples that can be achieved by simulating

$$u_{n,d}^{(i)} \sim q(\mathbf{U}) \,, \tag{4.37}$$

$$\mathbf{X}^{(i)} \sim q(\mathbf{X}) \,, \tag{4.38}$$

$$\mathbf{w}_{:,d}^{(i)} \sim p(\mathbf{w}_{d,:} \mid u_{n,d}^{(i)}, \mathbf{X}^{(i)}) \,, \tag{4.39}$$

from which we can then evaluate $1/S \sum_{s=1}^{S} \log p(h_{n,d} \mid \phi(w_{n,d}^{(s)}))$. Note that the KLD terms for $\mathbf{X}$ and $\mathbf{U}$ in Equation (4.36) will factor across T and D, respectively, and are analytically computable by Definition 1.3.5. A similar Monte-Carlo scheme is given in Gal et al. (2015) for the specific case of a categorical likelihood function.

Using the ELBO in Equation (4.36), we can now use first-order gradient methods to identify the model's hyperparameters and the inducing point locations Z. We

find that 10 Monte-Carlo samples are sufficient to ensure a stable optimisation routine.

4.4 Experiments

Within this section, we empirically investigate the utility of our proposed model in the context of political networks. We present two examples that concern the legislation voting in Peru's Congress, and the house committee formation in the United States of America's Congress.

Where relevant, we detail and compare our model to a competing model that exists within the hypergraph literature. However, in some cases, such an approach does not exist and we instead compare our model to a pairwise graph alternative. To project a hypergraph into a pairwise graph, we let each hyperedge be a clique and then construct a pairwise graph from the set of cliques. Where relevant, we weight the edges of the pairwise graph by the number of times the edge appeared within a clique. This process is formally known as a clique expansion (Agarwal et al., 2005).

4.4.1 Senate of Peru

Data In this first experiment, we consider data that describes the co-sponsorship of legislation within the Congress of the Republic of Peru in 2007 (Lee et al., 2017). To represent this dataset as a hypergraph, we form a hyperedge for each piece of legislation and the constituent vertices correspond to the members of Congress responsible for drafting the respective legislation. Within Peru's Congress, each congressperson has a party affiliation. Further, each party belongs to one of the three Congressional groups: the government, opposition and minority groups. Our

task here is to predict the group affiliation of a member of Congress given only the hypergraph structure.

Table 4.3: The number of Congressional groups and parties that were represented within a single hyperedge e.g., there were 80 hyperedges that contained congresspersons from three parties.

Unique count	Party	Group
1	668	718
2	44	75
3	80	28
4	16	-
5	9	-
6	4	-

The resulting hypergraph is made up of 121 vertices and 821 hyperedges. The average number of vertices per hyperedge is 8, the smallest hyperedge contains 1 vertex and the largest hyperedge contains 40 vertices. In Table 4.3 we summarise the party and group diversity within each hyperedge.

Latent function recovery The first research question we consider is *how well can a Matérn kernel defined on the vertices of a hypergraph represent the dependency structure between the hypergraph's vertices?*. We construct a multi-class classification GP on the vertices of our hypergraph that will probabilistically infer the party affiliation of a senator, given the pieces of legislation that the senator was involved in creating. To the best of our knowledge, we are the first to consider GP modelling on the vertices of a hypergraph so no directly comparative method is available. To establish suitable benchmarks, we represent our hypergraph as a pairwise graph using clique expansion. From thereon, GP modelling can be accomplished using the graph kernel provided in Borovitskiy et al. (2021).

We employ the categorical likelihood function of Hernández-Lobato et al., 2011

Table 4.4: The performance of the GP model on a graph and hypergraph structure. Results are reported for the 40 held-out vertices, split by the underlying (hyper)graph representation. Bold values denote the best performing model and standard errors are computed across 10 random partitions of the data. For every metric, excluding expected calibration error (ECE), a larger value is better.

Metric	Hypergraph	Graph Binary expansion	Graph Weighted expansion
Accuracy	$\mathbf{0.9 \pm 0.002}$	0.7 ± 0.015	0.75 ± 0.009
Recall	$\mathbf{0.89 \pm 0.002}$	0.74 ± 0.002	0.78 ± 0.006
Precision	$\mathbf{0.9 \pm 0.002}$	0.73 ± 0.033	0.79 ± 0.007
ECE	$\mathbf{0.13 \pm 0.001}$	0.29 ± 0.003	0.2 ± 0.002
Log-posterior density	$\mathbf{-0.34 \pm 0.017}$	-0.89 ± 0.001	-0.45 ± 0.005

and a multi-output GP $f : V \to \mathbb{P}^2$, where $\mathbb{P}^2$ is the probability 2-simplex and each output dimension corresponds to the probability of the respective vertex being attributed to one of the three political groups. As can be seen in Table 4.4, the hypergraph representation yields significantly fewer misclassified nodes with equally compelling precision and recall statistics. GPs are commonly used due to their ability to quantify predictive uncertainty and, as can be seen by the expected calibration error (ECE) values in Table 4.4, the hypergraph representation facilitates substantially improved posterior calibration. From these results we can conclude that, for this dataset, there is a utility to be gained from the hypergraph representation and the Matérn kernel can effectively capture the dependency structure between the vertices of the hypergraph.

Embedding Satisfied that a GP defined on the vertices of a hypergraph can model the dependency structure between the vertices, we now proceed to infer a latent representation of the hypergraph. To achieve this, we optimise the ELBO objective given in Equation (4.36) until convergence. The resulting latent representation is shown in Figure 4.4(a). We can see the separation that has

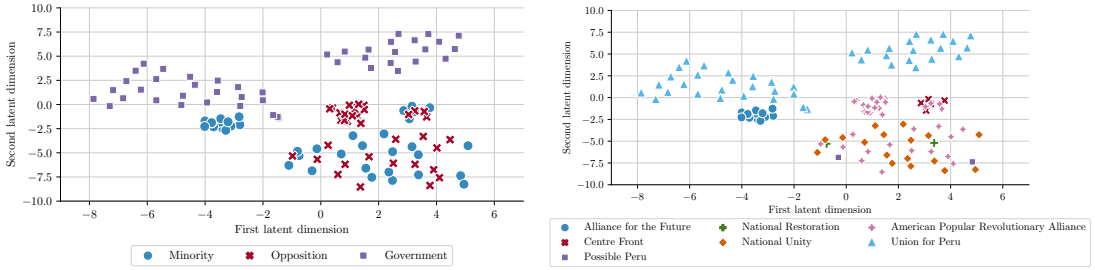


Figure 4.4: 2-dimensional latent space inferred by our GPLVM. Vertices are coloured by the respective Senator's Congressional group (left) and their party affiliation (right).

occurred between the government group and the opposition and minority groups. Further, we can observe a distinct split within the government group. In the 2006 elections, the Union for Peru party allied with the Peruvian Nationalist Party. Collectively, the alliance won 45 of the 120 seats in Congress and formed the government. However, after the election, the alliance internally dissolved and it is this dynamic that explains the separation within the government party in Figure 4.4(a). We visualise the latent representation of the hypergraph using the party affiliation of each senator in Figure 4.4(b).

In Figure 4.5(a) and Figure 4.5(b) we visualise the latent space that is inferred using the spectral embedding algorithm of D. Zhou et al. (2006). Under this approach, the leading two eigenvectors of the hypergaph's Laplacian matrix are used as the latent vectors, an approach very similar in spirit to PCA. We can see that, visually, the spectral embedding algorithm has been unable to learn a latent representation that is as dispersed as the one inferred by the GPLVM. We now proceed to empirically evaluate the quality of the latent spaces inferred by these two methods.

Latent space quality The motivation for learning a latent representation of a hypergraph was to enable easier visualisation of the vertices and to facilitate

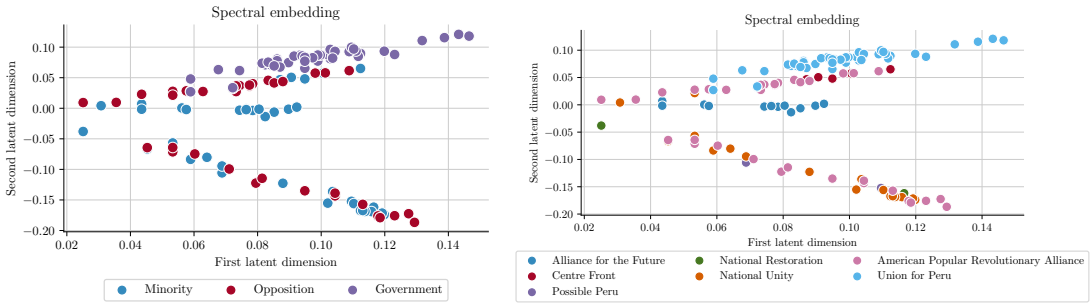


Figure 4.5: The 2-dimensional latent space inferred by the spectral method of D. Zhou et al. (2006). The colouring of a vertex is given by the representative senator's congressional group (left) and their party affiliation (right).

the construction of a machine learning model that maps from the vertices to an observed response value. Having explored the former task of visualising the vertices in the previous section, we now demonstrate the additional information that is contained within our latent space and the improvement that this can yield for downstream modelling. To achieve this, we let the 2-dimensional latent vector that is associated with each vertex be the input to a generalised linear model and support vector machine. We then use the vertices' corresponding Congressional group and political party as our observed response value. We use 80% of the vertices as training data and the remaining 20% for testing. The mean accuracy and one standard deviation of a 10-fold cross-validation procedure are reported in Table 4.5.

We can see from Table 4.5 that the latent space inferred by our GPLVM yields a substantial improvement in classification accuracy when compared to the spectral embedding method. This is evidenced by the improved accuracy when trying to predict both the Congressional group and the political party using both the logistic regression model and support vector machine. Whilst simplistic in nature, this is just one example of a downstream modelling task that one may wish to perform on the vertices of a hypergraph. However, defining such a model directly on the

Table 4.5: Predictive accuracy of a logistic regression and support vector machine model whose inputs are the 2-dimensional latent space vectors inferred using our GPLVM model and the spectral approach. Results are given as the mean ± 1 standard deviation as determined by 10-fold cross-validation. Bold values indicate the best performing approach across each classifier and the corresponding response variable.

		Logistic Regression	Support Vector Machine
Response	Method		
Party	Spectral	49.17 ± 12.6	59.17 ± 15.5
	GPLVM	$\mathbf{74.17} \pm 14.7$	$\mathbf{71.67} \pm 15$
Group	Spectral	47.50 ± 15.4	70.83 ± 9.3
	GPLVM	$\mathbf{77.50} \pm 9.9$	$\mathbf{75.00} \pm 10.5$

hypergraph itself will be challenging as off-the-shelf machine learning libraries will not be able to handle the hypergraph as input without significant preprocessing that will potentially result in a loss of information.

4.4.2 United States Senate Committees

In this experiment, we follow an identical workflow to that of Section 4.4.1. However, we now consider data from the United States of America's Congress. To construct a hypergraph, we let each congressperson be a vertex. The Congressional committees that each congressperson is a member of form the hyperedges in our hypergraph, and a vertex belongs to a hyperedge if the corresponding Congressperson belongs to that committee. Within Congress, there are several different types of committees. Some committees are permanent, such as the Standing Committee of Defense, and others are temporary, such as the Select Committee for Intelligence. For this reason, we use data from 1993 to 2017 and follow Chodrow et al. (2021) by assuming temporal stationarity across this time i.e., a senator cannot flip party affiliation.

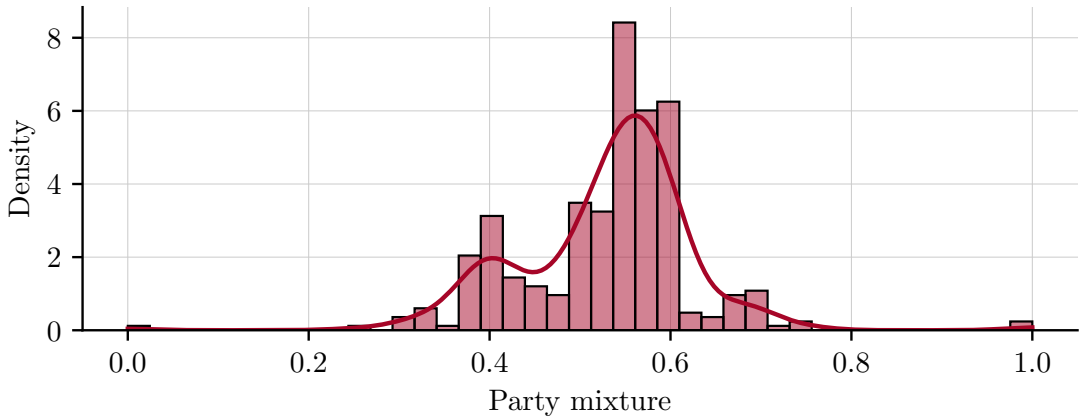


Figure 4.6: The representation of Republican members within committees. A value of 0.5 indicates that the committee is equally split between Republicans and Democrats.

The hypergraph contains 341 hyperedges and 1290 vertices. The average hyperedge contains 35 vertices and the distribution of Republican and Democratic representation within each hyperedge can be visualised in Figure 4.6. Here we see that, on average, most committees are comprised of an equal split of Republicans and Democrats. However, there does exist hyperedges comprised of entirely Republican or Democratic members. Treating the senators' party affiliation as the observed response value, we have a balanced classification task with the dataset containing 670 Republicans and 620 Democrats.

In Figure 4.7 we visualise the latent space inferred by our GPLVM and the spectral embedding method of D. Zhou et al. (2006). It is noteworthy that we do not observe the same clear separation between vertices as in Section 4.4.1. This is because, to form a committee, a member of the majority party will chair the committee and the ranking members of the committee are then selected from the minority party. The committees are, therefore, bipartisan and we should therefore expect to see a more gradual transition from one party to another as this corresponds to the changing political ideologies that are observed as we move

from a left-leaning committee to a right-leaning committee.

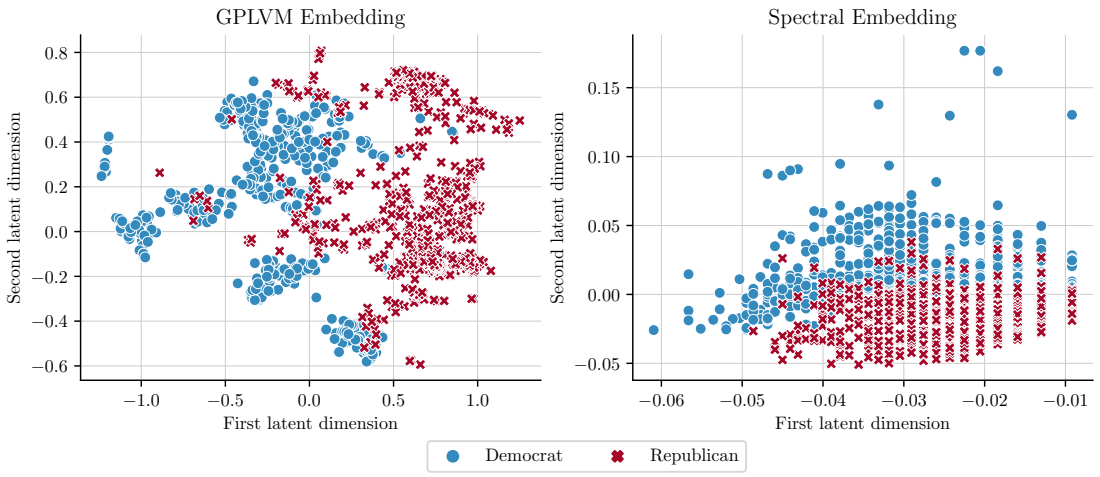


Figure 4.7: 2-dimensional latent spaces inferred by our GPLVM (left) and the spectral method (right) of D. Zhou et al. (2006). Senators are coloured blue if they identify as a democrat and red if republican.

Despite the lack of clear separation between vertices, we still see that using the latent space for downstream modelling is a worthy endeavour. Replicating the classification experiment that was described in Section 4.4.1 where the response variable is now the party affiliation of each Congressperson, we can see in Table 4.6 that using the latent vectors produced by our GPLVM yields an improvement over the latent vectors given by a spectral embedding. Using the latent vectors inferred by our GPLVM in conjunction with a logistic regression model, we can attain a 90.8% accuracy on the test set. This is a remarkably reassuring result as we have relied on no covariate information to produce such high quality predictions, and instead use only the hypergraph's structure to infer a latent space from which we can fit our logistic regressor.

Table 4.6: Predictive accuracy of a logistic regression and support vector machine model whose inputs are the 2-dimensional latent space vectors inferred using our GPLVM model and the spectral approach. Results are given as the mean ± 1 standard deviation as determined by 10-fold cross-validation. Bold indicates the best performing model, subject to variation across folds.

	Logistic Regression	Support Vector Machine
Spectral	83.7 ± 3.4	84.2 ± 2.7
GPLVM	$\mathbf{90.0 \pm 1.8}$	$\mathbf{90.5 \pm 2.6}$

4.5 Conclusions

Within this chapter, we have introduced the concept of a hypergraph. With this concept, we proceeded to construct a GPLVM that could embed the vertices of a hypergraph into a low-dimensional latent space. To achieve this, we extended the regular GPLVM model of Lawrence et al. (2005) to handle the hypergraph's structure. We then demonstrated the utility of our model on two real-world political examples.

An exciting avenue for future work is to investigate the geometric properties of the latent space using the tools developed in Tosi et al. (2014). By equipping the latent space with a Riemannian metric, the distance between any two latent coordinates can be precisely computed using the manifold's corresponding geodesic.

Chapter 5

GPJax - A Didactic Gaussian Process Library in JAX

This chapter provides an overview of GPJax, a GP package that I have developed in the JAX ecosystem. At the time of writing this, GPJax has been downloaded 75321 times with an average of 1012 downloads per week[1], and has amassed 210 stars on Github. Further, the open-source development of GPJax has attracted a community of eleven contributors.

GPJax is designed to be a library that can be easily used by researchers and individuals looking to develop their own custom GP algorithms. As a consequence of this, a heavy emphasis has been placed on providing computational abstractions that closely resemble the underlying maths. As we shall go on to see, this does not come at the cost of efficiency, as GPJax is comparable, and often faster, that the current state-of-the-art GP libraries.

[1] As reported in PyPi.

5.1 Introduction

GPs (Rasmussen et al., 2006) are Bayesian nonparametric models that have been successfully used in applications such as geostatistics (Matheron, 1963), Bayesian optimisation (Mockus et al., 1978), and reinforcement learning (M. Deisenroth et al., 2011). `GPJax` is a didactic GP library that provides users with a set of composable objects from which a GP model can be constructed. This low-level computational interaction with a GP means that it is simple to construct new GP methodologies and, for this reason, `GPJax` is perfectly suited for researchers wishing to design new GP models. However, for users wishing to simply fit a GP to a dataset, `GPJax` may not be the most appropriate library as it will require a certain level of understanding of the underlying maths.

By the virtue of being written in `JAX` (Bradbury et al., 2018), `GPJax` natively supports CPUs, GPUs and TPUs through efficient compilation to XLA, automatic differentiation and vectorised operations. Consequently, `GPJax` provides a modern GP package that can effortlessly be tailored, extended and interleaved with other libraries to meet the individual needs of researchers and scientists.

From both an applied and methodological perspective, GPs are widely employed in the statistics and machine learning communities. High-quality software packages that promote GP modelling are accountable for much of their success. However, there currently exists a gap within the JAX ecosystem for a Gaussian process package to be developed that incorporates scalable inference techniques. GPJax seeks to resolve this.

`GPJax` has been carefully tailored to amalgamate with the `JAX` ecosystem. For efficient MCMC inference, `GPJax` can utilise samplers from `BlackJax` (BlackJax, 2021) and `TensorFlow Probability` (Abadi et al., 2016). For gradient-based

optimisation, GPJax integrates seamlessly with Optax (Babuschkin et al., 2020), providing a vast suite of optimisers and learning rate schedules. To efficiently represent probability distributions, GPJax leverages Distrax (Babuschkin et al., 2020) and TensorFlow Probability (Abadi et al., 2016). To combine GPs with deep learning methods, GPJax can incorporate the functionality provided within Haiku (Babuschkin et al., 2020). The GPJax documentation includes examples of each of these integrations.

Several computational details are implemented in GPJax that enable fast and efficient computations, as we'll go on to see in Section 5.3.

- **Matrix inversion** GPJax computes the matrix's Cholesky factorisation and solves a linear system using the Cholesky factorisation. This is a computationally efficient approach to matrix inversion that is also numerically stable.
- **Parameter spaces** During optimisation, all parameters are transformed such that they have support on the entire real line to ensure numerically stable optimisation. All transformations are bijectors, meaning that the original parameter can be recovered by simply computing the inverse of the transformation.
- **Cached quantities** The posterior predictive function given in GPJax takes a set of test locations as inputs and returns the posterior predictive distribution supported at these points. To compute this quantity, the covariance matrix over the conditioned training set must be computed. However, its value is invariant to the test location set. We therefore cache quantities such as this, meaning that when the posterior is evaluated sequentially, as in Bayesian optimisation or active learning, the covariance matrix over the training set is only computed once.

- **Stateful operations** Parameters are the only quantity to have a state in `GPJax`. This is in keeping with the JAX workflow, but also provides the practitioner with complete control over the model's parameters and avoids any unintended side-effects happening.

The foundation of each abstraction given in `GPJax` is a `Chex` (Babuschkin et al., 2020) dataclass object. These require significantly less boilerplate code than regular Python classes, leading to a more readable codebase. Moreover, `Chex` dataclasses are registered as PyTree nodes, facilitating the applications of `JAX` operations such as just-in-time compilation and automatic differentiation to any `GPJax` object.

The intimacy between `GPJax` and the underlying maths also makes `GPJax` an excellent package for people new to GP modelling. Having the ability to easily cross-reference the contents of a textbook with the code that one is writing is invaluable when trying to build an intuition for a new statistical method. We further support this effort in `GPJax` through documentation that provides detailed explanations of the operations conducted within each notebook.

5.2 Existing Gaussian process libraries

Within the Python community, the three most popular packages for GP modelling are `GPFlow` (Matthews et al., 2017), `GPyTorch` (Gardner et al., 2018), and `GPy` (GPy, 2012). Despite these packages being indispensable tools for the community, none support integration with a `JAX` workflow. On the other hand, the BayesNewton (Wilkinson et al., 2021) and `TinyGP` (Foreman-Mackey, 2021) packages utilise a JAX backend. However, `BayesNewton` is designed on top of `ObJax` (Objax Developers, 2020), making integration with the broader `JAX`

ecosystem challenging. Meanwhile, TinyGP offers excellent integration with inference frameworks such as NumPyro (Phan et al., 2019) but does not yet support inducing points frameworks (Hensman et al., 2013a, e.g.,). GPJax exists to resolve these issues. Furthermore, modern research from the GP literature, such as graph kernels (Borovitskiy et al., 2021) and Wasserstein barycentres for GPs (Mallasto et al., 2017), are supported within GPJax but absent from these packages. Finally, the Stheno package (Bruinsma, 2022) supports a JAX backend along with TensorFlow, PyTorch and Numpy. Whilst this integrates GPs into an extensive JAX workflow, GPJax has the advantage of being a pure JAX codebase, whereas Stheno requires using a custom linear algebra framework.

For completeness, packages written for languages other than Python include GPML (Rasmussen et al., 2010) and GPStuff (Vanhatalo et al., 2013) in MATLAB. An R port also exists for GPStuff. Within Julia, there exists GaussianProcesses.jl (Fairbrother et al., 2022), AugmentedGaussianProcesses.jl (Galy-Fajou et al., 2020), and Stheno.jl (Tebbutt W, 2022). GP implementations are available in numerous modern probabilistic programming languages such as NumPyro (Phan et al., 2019), Stan (Carpenter et al., 2017), and PyMC (Salvatier et al., 2016).

5.3 Functionality of GPJax

As acknowledged in Section 5.1, the purpose of GPJax is to allow researchers to computationally define GP in a manner that is similar, if not identical, to how a GP is mathematically formulated on paper. A compelling example of this can be seen in Listing 2 where a non-conjugate GP posterior is defined. For completeness,

the hierarchical model can be written as

$$k(\cdot, \cdot') = \sigma^2 \exp\left(-\frac{\| \cdot - \cdot' \|^2}{2\ell^2}\right) \tag{5.1}$$

$$p(f(\cdot)) = \mathcal{N}(0, k(\cdot, \cdot')) \tag{5.2}$$

$$p(y \mid f(\cdot)) = \text{Bern}(y \mid f(\cdot)) \tag{5.3}$$

$$p(f(\cdot) \mid y) \propto p(y \mid f(\cdot))\, p(f(\cdot)) \tag{5.4}$$

In Listing 2 lines 8-9, we start by defining the kernel function (Equation (5.1)) that will be used to parameterise our GP prior (Equation (5.2)). Independent of the prior, in line 11, we then define our likelihood function that reflects our beliefs surrounding the observed data. In this example, the observed data is binary and we, therefore, adopt a Bernoulli likelihood function (Equation (5.3)) that factorises over all 100 data points.

```
1  import gpjax as gpx
2  import jax.numpy as jnp
3
4  x = jnp.linspace(-5.0, 5.0, num=100).reshape(-1, 1)
5  y = jnp.sign(x)
6  D = gpx.Dataset(X=x, y=y)
7
8  kernel = gpx.kernels.RBF()
9  prior = gpx.gps.Prior(kernel=kernel)
10
11 likelihood = gpx.likelihoods.Bernoulli(num_datapoints=100)
12
13 posterior = prior * likelihood
14 params, _, _, _ = gpx.initialise(posterior)
15
16 log_posterior_density = posterior.log_posterior_density(training=D)
17 log_posterior_density(params)
```

Listing 2: Evaluating the log-posterior density of a non-conjugate Gaussian process in GPJax.

The posterior distribution (Equation (5.4)) can then be computed up to an unknown normalising constant by taking the product of our already defined prior

and likelihood function (line 13). It is at this point that we explicitly initialise the state of our posterior's parameters through the `initialise` function (line 14). This is a significant point of difference in `GPJax` as parameters are given an explicit state in the program. Handling parameters in this way not only prevents undesirable side effects from happening but also places more control in the hands of the practitioner. In this example, the parameters are a dictionary containing the kernel's lengthscale and variance parameter, and the latent function's value.

We finally define a log-posterior density function in line 16. The returned object from this line is itself a function that will use to evaluate the model's log-posterior density as a function of the parameter set. This function, as with all functions in `GPJax`, is a *pure* function, meaning all the input data is passed through the function's parameters, and all the results are output through the function's return statements. Further, all pure functions are entirely deterministic, meaning that invoking the function multiple times will always return the same result. The benefit of structuring `GPJax` around this principle means that gradients can be taken of any function, such as the log-posterior density, by simply calling `grad(log_posterior_density)`. Similarly, the functions in `GPJax` can be compiled into XLA for significantly faster computations. Combined, these two operations make gradient-based optimisation of model parameters an incredibly efficient process in GPJax, a process that we explore further in Section 5.4.

5.4 Experiments

In this section we benchmark the performance of `GPJax` against the two most established GP packages available in Python: `GPFlow` and `GPyTorch`. We note that despite the relative immaturity of `GPJax` when compared against `GPFlow` and `GPyTorch`, the performance of `GPJax` is at least comparable, if not better than

these two established packages. All experiments were run on a 16-core, 3.00GHz Intel Xeon Gold 6248R CPU with 64Gb of RAM.

5.4.1 Marginal log-likelihood

Efficient computation of the marginal log-likelihood (Equation (1.60)) is crucial for any performant GP package. This is because in conjugate GP regression, the marginal log-likelihood is often maximised with respect to the GP's hyperparameters to determine the maximum likelihood estimate. As such, we can often evaluate the marginal log-likelihood upwards of 500 times during a typical optimisation run.

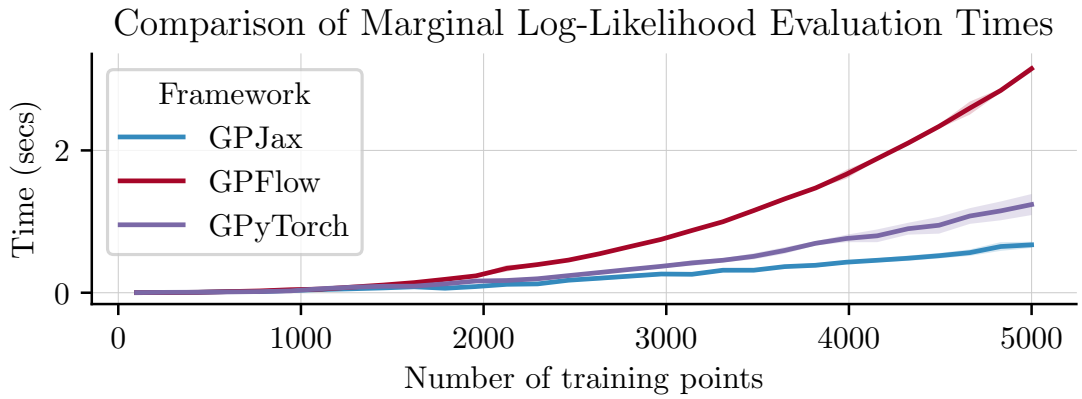


Figure 5.1: Comparison of the marginal log-likelihood computation times for a conjugate Gaussian process regression model.

To compare the performance of `GPJax` against `GPFlow` and `GPyTorch`, we first compile each package's marginal log-likelihood functionality into XLA. We then evaluate it for a range of dataset sizes, the results of which can be seen in Figure 5.1. We see that for smaller dataset sizes, there is little difference between the three packages. However, as the dataset size surpasses 2000 data points, the more efficient scaling of `GPJax` begins to emerge. For datasets of 5000 data points, `GPJax` is 1.8 times faster than `GPyTorch` and 4.7 times faster than `GPFlow`.

5.4.2 Collapsed evidence lower bound

For moderately large datasets where the marginal log-likelihood calculation would be intractable, the collapsed ELBO Equation (1.97) is a sparse approximation that provides a tractable lower bound to the marginal log-likelihood. Optimisation can then be performed with respect to this objective. In addition to the hyperparameters optimised in Section 5.4.1, the collapsed ELBO is also a function of the inducing variable that parameterises the variational approximation.

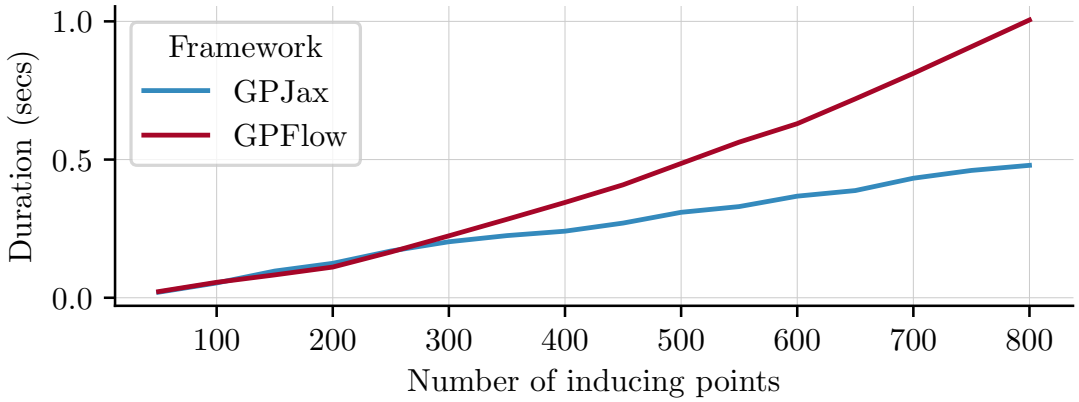


Figure 5.2: Comparison of the collapsed ELBO computation times for a sparse Gaussian process regression model.

We compare the performance of `GPJax` to `GPFlow` by simulating a dataset of 20000 points and then evaluating the collapsed ELBO for a range of inducing variable sizes. The results of this comparison can be seen in Figure 5.2. As with the marginal log-likelihood, the performance of `GPJax` is comparable to `GPFlow` for smaller sets of inducing points, but as the number of inducing points increases, `GPJax` offers significantly better scaling. For 800 inducing points, `GPJax` is 2.1 times faster than `GPFlow`.

5.4.3 Uncollapsed evidence lower bound

As a final demonstration of the computational efficiency of GPJax, we evaluate the computational cost of evaluating the uncollapsed ELBO presented in Equation (1.109). Unlike the collapsed ELBO demonstrated in Section 5.4.2, the uncollapsed ELBO can be evaluated on mini-batches of data. We, therefore, evaluate the uncollapsed ELBO for a range of inducing variable sizes and a range of minibatch sizes.

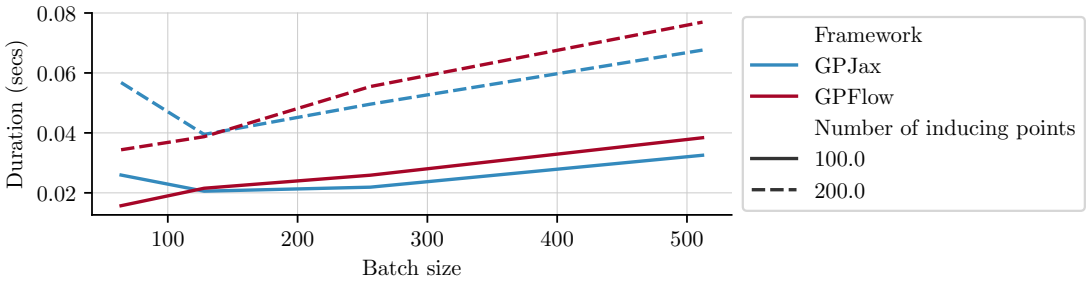


Figure 5.3: Comparison of the uncollapsed ELBO computation times for a sparse Gaussian process model as a function of the minibacth size. The number of inducing points is fixed to 100 and 200.

In Figure 5.3, we observe the scaling of the uncollapsed ELBO as the minibatch size increases. For inducing point sets of 100 and 200 data points, we see that the scaling of GPJax is marginally faster than the highly optimised implementation that is given in GPFlow. For a minibatch size of 512 data points GPJax is 1.2 times faster than GPFlow. Similarly, if we keep the minibatch size fixed at 512 data points and model the computational runtime of GPJax and GPFlow as a function of the number of inducing variables, we see in Figure 5.4 that the scaling of GPJax is almost identical to GPFlow.

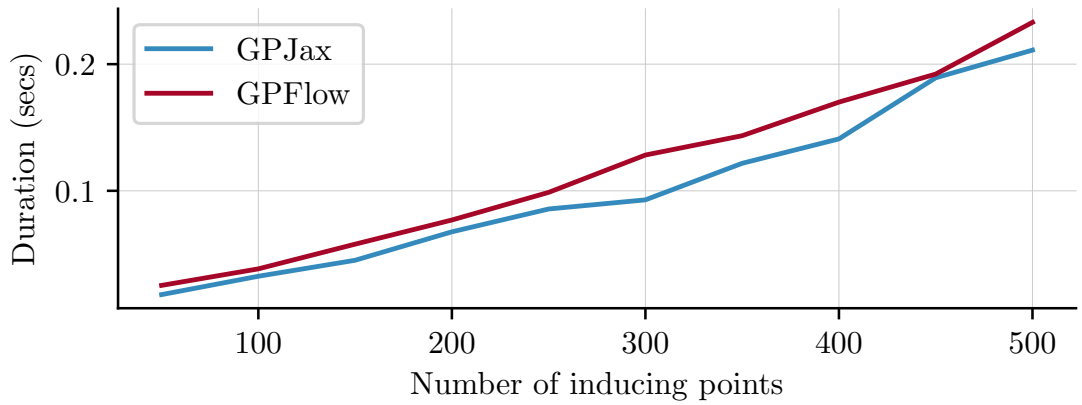


Figure 5.4: Comparison of the uncollapsed ELBO computation times for a sparse Gaussian process model as a function of the number of inducing points. The batch size is fixed to 512 data points.

5.5 Conclusions

In this chapter, we presented `GPJax`; a JAX-based GP software package that enables fast inference in GP models using both exact and approximate inference schemes. `GPJax` enables researchers wishing to develop novel GP models to rapidly implement their models. This is achieved by the low level of abstraction given in `GPJax`, meaning that the user has full control over the underlying GP model and its constituent components.

Whilst this chapter has focussed on connecting `GPJax` to the wider community of GP packages, the online documentation given in `GPJax` provides a wealth of information on the GP model and how it can be applied to a wider range of problems. Examples of this include demonstrating how a set of GP posterior distributions can be combined using a Wasserstein barycentre, and how inference can be done in a GP model with a non-Gaussian likelihood function through MCMC or Laplace approximations. The documentation is available at HTTPS://GPJAX.READTHEDOCS.IO.

Chapter 6

Probabilistic Climate Model Ensembles

The conclusion of this thesis is comprised of two sections: the first covers some ongoing work that incorporates many of the topics discussed in previous chapters and the second is a summary of the work presented in this thesis and possible future directions. Whilst the work presented in this chapter is incomplete, the framework and application are sufficiently developed to contribute to this thesis.

Chapter 6 addresses the problem of combining multiple climate models in a probabilistic manner. To construct the ensemble, we first emulate each climate model using a GP that is defined over a set of outputs from the climate model. To effectively model this, we design a sparse hierarchical GP that allows us to infer the latent function of the climate model. Using proper scoring rules, we can then rank each model based on how well its emulator's posterior distribution reflects the true observations that we observed. Finally, we learn an ensemble using a weighted Wasserstein barycentre that we can compute in closed form.

6.1 Introduction

Ensemble models An ensemble model is the formation of a "*committee*" of models for which each constituent model supplies a, possibly weighted, contribution to the final ensemble prediction. For classification problems, the ensemble prediction could be a majority vote of the constituent models, whilst in regression problems, the ensemble prediction could be a mean statistic. The paradigm motivating the construction of ensemble models is that even though a single model in isolation may not perform well, the combination of multiple models, the ensemble, will perform better.

The origin of ensemble models can be attributed to the political science literature through Condorcet's Jury Theorem (De Condorcet, 1785). This work approaches the question of how many people must form a committee to ensure that the correct decision is made for a binary task. Assuming that the probability of each person on the committee making the correct decision is greater than 0.5, then the probability of the committee making the correct decision will increase monotonically as the committee's size grows.

In recent years, ensemble methods have been widely adopted in the machine learning community through boosting and AdaBoost (Schapire, 1990; Auer et al., 1995), mixture-of-experts (Jacobs et al., 1991), aggregated bootstraps (Breiman, 1996), random forests (T. K. Ho, 1995), and gradient boosting machines (Mason et al., 1999; Friedman, 2001). Within the GP literature, ensemble methods have been borrowed to construct scalable approximations of the GP posterior distribution by subsetting the data, fitting a single GP to each subset, and then combining the predictions of each GP (M. Deisenroth et al., 2015; Srivastava et al., 2015). This work contrasts with the existing GP ensemble methods in that we are interested in creating an ensemble of GPs to combine the information contained

with multiple climate models and not to improve the scalability of our model.

Climate modelling From 1850-1900 to 2001-2020, the global surface temperature increased by 0.99°C (Pörtner et al., 2022). Until 1981, the rate at which Earth's temperature increased was 0.08°C per decade. However, from 1981 to the present day, the rate has more than doubled to 0.18°C per decade. Further, the nine years from 2013 to 2021, inclusively, rank among the 10 warmest years on record (NOAA, 2021). The ramifications of such warming cannot be understated, as the impacts of an increasing surface temperature will reduce snow cover and sea ice, cause intensifying droughts, and irreversibly alter the habitats of plants and animals. To enable us to plan and adapt to these changes, any model we construct must be able to accurately predict the range of likely future scenarios. To this end, we construct a probabilistic ensemble of climate models that allows us to propagate uncertainty from the individual models' realisations through to the final ensemble projection.

Climate models are numerical process models that are fundamental in investigating how climatic systems develop under natural and anthropogenic influences over a range of temporal and spatial scales (Taylor et al., 2012; Randall et al., 2007). Functionally, a climate model is comprised of a set of differential equations that characterise how energy and matter interact within the atmosphere. The output of climate models is resolved on a spatial grid whose resolution controls the level of detail in the model's predictions, and the computational power required to run the model. A finer grid will yield more detailed results, but will also be more computationally demanding to run.

Based on their output, climate models can be split into two categories: a *hindcast* where the model is provided with a set of initial conditions at 1800 and then

run from that point forward to the present day, and *forecasts* where the model is run from the present day forwards in time to a future year that is usually 2100 but can be as far as 2300. The uncertainty around extrapolations as far as 2100, let alone 2300, will render any output meaningless. Therefore, to control for the effects of our future socio-economic behaviours, climate models' forecasts are constrained by a set of shared socio-economic pathways (SSPs) (Riahi et al., 2017). An individual SSP describes a possible pathway that the world could take and, therefore, controls for factors such as population, economic growth, education, urbanisation and technological development. There are five main SSPs that can be briefly described as follows:

- **SSP1: Sustainability** - The most positive future scenario that sees a world of sustainable development and equality.
- **SSP2: Middle of the Road** - A world whose socioeconomic trends mimic those of the past. Progress under this scenario is slow and discriminative, but less so than in SSP3 and SSP4.
- **SSP3: Regional Rivalry** - A divided world that sees a resurgence of nationalistic politics. At the cost of global development, individual countries prioritise their energy and food security. Economic growth is slow and dependent on fossil fuels.
- **SSP4: Inequality** - Unequal investments and increasingly disjoint economic opportunities lead to a widening stratification both within and between countries over time.
- **SSP5: Fossil-fuel intensive development** - Rapid and unconstrained growth in economic activity and energy consumption. This scenario sees similar economic growth to SSP1, but with a much higher dependency on

fossil fuel usage.

In Figure 6.1, we visualise atmospheric projections of the greenhouse gas carbon dioxide (CO_2) for each of the five SSPs according to the models run in Riahi et al. (2017). As we can see, following SSP5 leads to the largest atmospheric CO_2 increase, whilst SSP1 sees the least significant increase. There is little difference between SSP2, SSP3, and SSP4, however, this could vary from one climate model to another. The SSPs in Figure 6.1 are a set of baseline scenarios, which inform

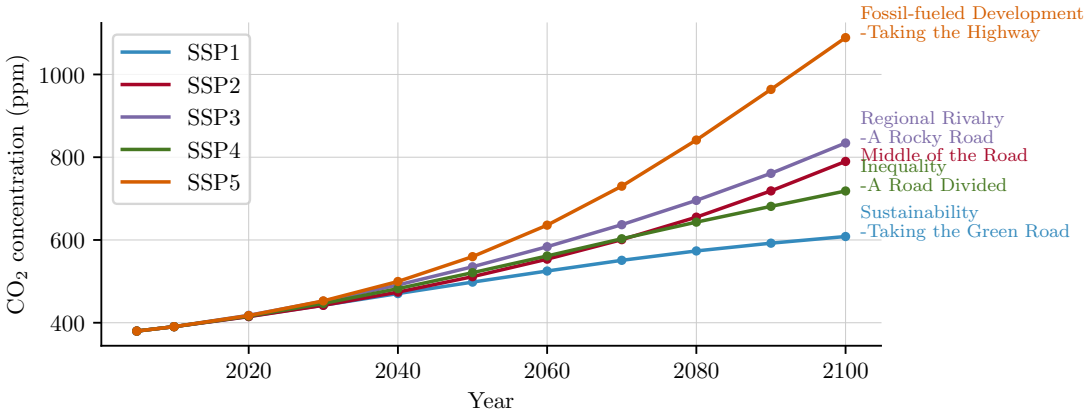


Figure 6.1: The five major shared socioeconomic pathways that describe the range of possible future atmospheric concentrations of carbon dioxide.

projections under the assumption that no new climate policies will be introduced, beyond those in existence today. These help understand the purpose of SSPs. However, when constraining the output of a climate model by a specific SSP, we impose a further constraint on the levels of greenhouse gasses and radiative forcings[1] that might be expected in the future. Such a constraint is known as a representative concentration pathway (RCP) and is characterised by a number whose value is the Watts per squared metre energy change e.g., 2.6Wm^{-2} denotes the RCP where every squared metre on earth receives an additional 2.6 Watts

[1]Radiative forcings are the difference between incoming and outgoing energy in Earth's climate system. If the levels of greenhouse gasses increase, then the amount of incoming will increase relative to the amount of outgoing energy. Consequently, the Earth will become a warmer planet.

of energy. For each climate model, the output is, in practice, constrained by a combination of a SSP and RCP, although the combinations may differ from one model to another. A combination is denoted by the concatenation of an SSP scenario with an RCP value e.g., SSP126. In Figure 6.2, we visualise atmospheric CO_2 projections for each combination of SSP and RCP.

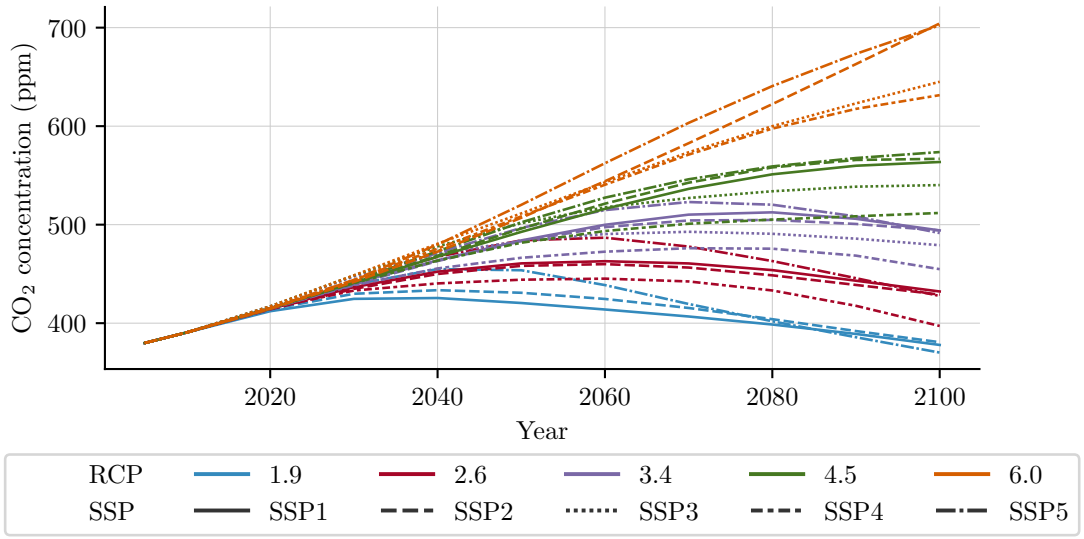


Figure 6.2: Global CO_2 emissions in parts per million for all of the SSP and RCP combinations. The colour of each line represents the RCP, and the style of each line represents the SSP.

Climate model ensembles Ensembles of climate models, where modelling groups produce simulations to an agreed experimental design, are routinely used to more accurately estimate the evolution of the climate (e.g. Hoegh-Guldberg et al., 2018; Deser et al., 2020; Vautard et al., 2021). Additionally, climate model ensembles provide a way to investigate model uncertainty and quantify the confidence in future projections, typically by considering the inter-model variance (Knutti et al., 2013; Murphy et al., 2004; Tebaldi et al., 2007; Lehner et al., 2020). However, commonly used ensemble analysis methods, such as the multi-model mean (e.g. Lamarque et al., 2013; Tebaldi et al., 2007) and weighted-mean

(e.g. Sanderson et al., 2017; Brunner et al., 2019; Amos et al., 2020), often fail to propagate uncertainties from model realisation to ensemble projection in a principled manner (IPCC, 2010).

Ensemble approaches are widely used throughout environmental and statistical sciences because an ensemble prediction is typically more accurate than a prediction from any single model (Reichler et al., 2008). Ensembles are now the standard way to perform broad model investigations (Eyring et al., 2016; WMO, 2018) including evaluating structural and epistemic model uncertainties (Deser et al., 2020; Tebaldi et al., 2007; Hawkins et al., 2009). When considering the uncertainty in an ensemble projection, it is very common to use the standard deviation across weighted and/or unweighted model simulations (e.g. Raftery et al., 2005; Dhomse et al., 2018; IPCC, 2010). However, the standard deviation is not necessarily a good estimator of the ensemble variance as it can be biased by outlier models (IPCC, 2010). Furthermore, the standard deviation across an ensemble of models does not capture a likely individual model's inter-annual variability.

Methods to produce ensemble projections from multiple models fall broadly into two categories; methods which use only the outputs from a set of models and then compute metrics such as the multi-model mean (Palmer et al., 2005; Hoegh-Guldberg et al., 2018) to derive a prediction, and estimate uncertainty by considering how far models are away from the mean model value (e.g. Giorgi et al., 2002). There are also methods which use observations to score and subsequently weight the individual models that form an ensemble (Sanderson et al., 2017; Amos et al., 2020; Liang et al., 2020). Model weighting addresses limitations of unweighted ensembling methods including their inability to account for model similarity or variable model performance (Knutti, 2010). The majority of these methods also attribute uncertainty to forward-looking projections using

the models' spread. However, quantifying the projection's uncertainty based on the spread of the models' outputs only tells us about the range of plausible outcomes, and does not provide a measure of the uncertainty in the ensemble model's prediction. Further, such an approach is accompanied by the assumption that each model is equally performant, and the models' predictions are all normally distributed around the mean.

Here we present a probabilistic ensembling method which quantifies uncertainty through the entire ensembling process, from individual simulations to ensemble output. We apply our approach to projections of surface temperature change from the Coupled Model Intercomparison Project Phase 6 (CMIP6) (Eyring et al., 2016) to demonstrate the method's ability to produce better-constrained projections with principled uncertainty estimates.

Despite drawing probabilistic conclusions from ensembles of climate models, the tools with which we generate ensemble output themselves are rarely probabilistic. We seek to address this deficit within this paper by designing an end-to-end statistical framework to probabilistically ensemble models to produce fully probabilistic climate projections.

6.2 Background

The ensemble method presented here relies on ideas from the fields of optimal transport, GPs, and scoring rules. We, therefore, review these concepts in this section where an explanation of GPs has been omitted in favour of Section 1.3.

6.2.1 Wasserstein barycentres

Optimal transport is a rich area of mathematics with origins dating back to 1781 when Gaspard Monge considered the problem that, given a set of locations from which soil can be dug up from the ground, how can it *optimally* be transported to a set of construction sites (Monge, 1781). The idea has been generalised in modern statistics to consider the case where we have two probability distributions ν_1 and ν_2 and we wish to compute the optimal way to transport probability mass from ν_1 to ν_2 such that ν_2 is equal to ν_1. The literature concerning the assignment problem is rich, and goes beyond the scope of this work. We, therefore, refer the curious reader to the detailed works of Villani (2009) and Peyré et al. (2019).

The squared 2-Wasserstein distance measure $W_2^2(\nu_1, \nu_2)$ is commonly used within machine learning and statistics for computing the distance between two probability distributions ν_1 and ν_2. More precisely, the metric can be given by

$$W_2^2(\nu_1, \nu_2) = \inf_{T_{\#\nu_1=\nu_2}} \int_{\mathcal{X}} \|x - T(x)\|^2 \mathrm{d}\nu_1(x)\,, \tag{6.1}$$

where $T_{\#\nu_1=\nu_2}$ denotes the push forward of ν_1 through the transport map $T : \mathcal{X} \to \mathcal{X}$ defined on a metric space $\mathcal{X}$. Intuitively, the metric measures the minimum amount of work required to transport probability mass from ν_1 to ν_2 with respect to the Euclidean norm.

In general, computing Equation (6.1) cannot be done analytically and its computation, therefore, requires solving a complex linear program. However, when ν_1 and ν_2 are multivariate Gaussian distributions $\nu_1 \sim \mathcal{N}(\mathbf{m}_1, \mathbf{\Sigma}_1)$ and $\nu_2 \sim \mathcal{N}(\mathbf{m}_2, \mathbf{\Sigma}_2)$, the Equation (6.1) has the closed form expression

$$W_2^2(\nu_1, \nu_2) = \|\mathbf{m}_1 - \mathbf{m}_2\|_2^2 + \mathrm{trace}\left(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2 - 2\left(\mathbf{\Sigma}_1^{1/2}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{1/2}\right)^{1/2}\right). \tag{6.2}$$

This result can be attributed to Takatsu (2011).

If we now allow ourselves to consider a set of T probability distributions $\xi = \{\nu_1, \nu_2, \ldots, \nu_T\}$, then a natural question that we may ask is "*what is the average probability distribution $\bar{\nu}$?*". Such a quantity is known as a barycentre and we can use the idea of Wasserstein distances given above to compute it (Agueh et al., 2011). In particular, when every element in ξ is a Gaussian distribution, the Wasserstein barycentre is analytically available and is itself a Gaussian distribution $\bar{\nu} \sim \mathcal{N}(\bar{\mathbf{m}}, \bar{\mathbf{\Sigma}})$[2]. We call barycentres of this form a *Bures barycentre* (Bhatia et al., 2019). To compute the mean and covariance of the Bures barycentre, we can calculate the following

$$\bar{\mathbf{m}} = \sum_{t=1}^{T} \beta_t \mathbf{m}_t \,, \quad \bar{\mathbf{\Sigma}} = \sum_{t=1}^{T} \beta_t \left(\bar{\mathbf{\Sigma}}^{1/2} \bar{\mathbf{\Sigma}}_t \bar{\mathbf{\Sigma}}^{1/2} \right)^{1/2} , \tag{6.3}$$

where $\{\beta_t\}_{t=1}^{T}$ is a set of weights. If we assume that elements in ξ are equally informative of the mean, then we can assign uniform weights. However, if we have some prior knowledge that certain elements in ξ more precisely represent the mean than others, then we are free to choose non-uniform weights under the constraint that $\sum_t \beta_t = 1$ and all weights are positive-valued. We visualise this in Figure 6.3 and utilise this idea in Section 6.4.2.

[2] A common over-simplification is that the Wasserstein barycentre is simply the average probability distribution. More precisely, the Wasserstein barycentre is the Fréchet mean defined on a Wasserstein space. Intuitively, the barycentre is the distribution that minimises the average Wasserstein distance to each of the sets' constituent measures.

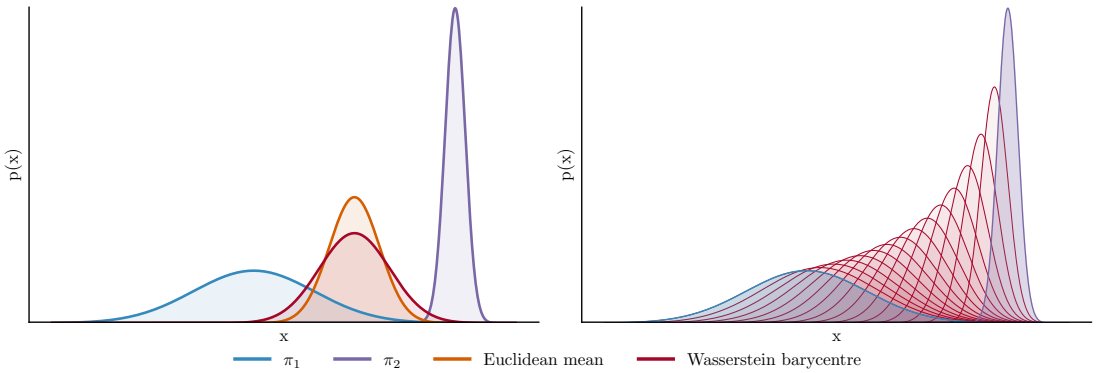


Figure 6.3: Left: Comparison of the naive Euclidean mean and the Wasserstein barycentre computed between two Gaussian distributions $\pi_1 \sim \mathcal{N}(10, 1.5)$ and $\pi_1 \sim \mathcal{N}(15, 0.25)$. Right: Visualisation of the changing Bures barycentre that is computed between two Gaussian distributions π_1 and π_2 as the weights (w_1, w_2) change from $(\alpha, 1-\alpha)$ for $0 < \alpha < 1$.

6.3 Data

6.3.1 Model output

We used monthly average surface temperature output from climate models within the CMIP6 ensemble (Eyring et al., 2016). These are sophisticated numerical models, built upon fundamental laws of physics and chemistry, which are used to generate future projections and explore a wide range of scientific questions. Specifically, we consider 20 models $\{m_w\}_{w=1}^{20}$ that simulated 3 or more realisations for both the historical and forecast simulations.

All historical simulations (1850–2014) are similarly forced with historic green house gas emissions, land use, sea surface temperature, sea ice concentrations and solar and volcanic events, such that they recreate the climatic behaviour of the past. For climate model forecasts (2016–2100) we focus on the 7 main SSP experiments which represent different ways that anthropogenic influences may change within the next century. All model output was conservatively regridded to a common $5^o \times 5^o$ to directly match the observational resolution.

We denote the set of realisations for the i^{th} model m_i as $\mathbf{Y}^{(i)} = \{\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, \ldots, \mathbf{y}_{R_i}^{(i)}\}$. Each element in $\mathbf{Y}^{(i)}$ is an estimate of the global surface temperature at an index set $\mathbf{X}$ that is comprised of a matrix of spatial coordinates $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2\}$ and vector of time points $\mathbf{t}$. This leads to $K = |\mathbf{s_1}| \times |\mathbf{s_2}| \times |\mathbf{t}|$ datapoints per realisation. To spare overloading notation, we index the individual values within one realisation (j^{th} realisation from the i^{th} model) using the notation $[\mathbf{y}_{j,1}^{(i)}, \mathbf{y}_{j,2}^{(i)}, \ldots, \mathbf{y}_{j,K}^{(i)}]$.

6.3.2 Observation data

Observations were sourced from HadCRUT5 (Morice et al., 2021) a global gridded dataset of historic surface temperature anomalies (relative to 1961–1990), which merges observational products of land and sea surface temperatures. The dataset contains 200 observational realisations, which captures the uncertainty due to measurement error, spatially non-uniform sampling of observations and from the statistical reconstruction. Like the models, the dataset spans 1850 to the present day and exists on a two-dimensional spatial grid. We align observations to model outputs through the spatiotemporal indexing and denote a single set of observations at the k^{th} spatiotemporal index as $\mathbf{y}_k^\star = [y_{k,1}^\star, y_{k,2}^\star, \ldots, y_{k,200}^\star]$.

6.4 A probabilistic ensemble

We will now proceed to describe the main contribution of this work; a three-step process that creates a probabilistic ensemble from sets of model realisations. Although the application of this work is to model global surface temperature, we present this section in a general setting as the framework can be effortlessly adapted to new problems.

6.4.1 From realisations to model emulation

Given a set of realisations from a single model, our goal is to probabilistically infer the latent function that generated the realisations. We make no assumptions of the underlying physics that drives global surface; imparting such information into the model is left as further work. To ease notation in this subsection, we will consider the task of emulating a single model and consequently drop the notational dependency on the model index, given by ^{i} in the previous subsection. We consider a set of model realisations $\mathbf{Y} = \{\mathbf{y}_r\}_{r=1}^{R}$ where $\mathbf{y}_r$ is a vector in $\mathbb{R}^N$. We make two assumptions: 1) there exists a latent function $\hat{\mathbf{y}}$ from which all $\mathbf{y}_r$ are noisy realisations of and 2) all realisations are defined across the same index set $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$. In our examples, these data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}_r\}_{r=1}^{R}$ are a collection of surface temperature projections across a common temporal range, simulated by multiple climate models, where we wish to extract the latent climate signal.

To model these data we use a Bayesian hierarchical model of GPs

$$g(\mathbf{X}) \sim \mathcal{GP}(\mathbf{0}, k_g(\mathbf{X}, \mathbf{X}')), \tag{6.4}$$

$$f^{(r)}(\mathbf{X}) \sim \mathcal{GP}(\boldsymbol{\mu}_{g\,|\,\mathbf{y}}, k^{(r)}(\mathbf{X}, \mathbf{X}'))\,, \tag{6.5}$$

where $g(\mathbf{X})$ is the latent function that underpins all $\mathbf{y}_r$. $f^{(r)}(\mathbf{X})$ is the latent function for an individual vector and has a mean given by the posterior mean of g that we denote $\boldsymbol{\mu}_{g\,|\,\mathbf{y}} = \mathbb{E}_{p(g\,|\,\mathbf{y})}[g(\mathbf{X})]$. The kernel k_g describes the covariance structure of the latent function and the set of kernels $\{k^{(r)}\}_{r=1}^{R}$, which have a one-to-one mapping with each $f^{(r)}$, describe the covariance structure of an individual realisation. Different kernel functions may be used for any of the kernel terms. This hierarchical model is linear, which means that for a single vector $f^{(r)}(\mathbf{X}) \sim \mathcal{GP}(\mathbf{0}, k_g(\mathbf{X}, \mathbf{X}') + k^{(r)}(\mathbf{X}, \mathbf{X}'))$, whereas the joint distribution

between different time series is described by the covariance function $k_g(\mathbf{X}, \mathbf{X}')$. This model can be extended to multiple hierarchies (e.g., Hensman et al., 2013b) but is computationally limited in its scalability; $\mathcal{O}(N^3)$ for data size as per standard GP regression and $\mathcal{O}(R^2)$ for the number of realisations.

To ameliorate the computational cost associated with fitting models of this form, we approximate each GP in Equations (6.4)–(6.5) with a low-rank (*sparse*) GP. To form such an approximation, we introduce a set of *inducing points* $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$ where $M \ll N$. We assume that the inducing points exist on the same support as our observed inputs $\mathbf{X}$, but they need not be a subset of $\mathbf{X}$ (Snelson et al., 2005; Titsias, 2009). Further, the set of inducing points used is shared across all GPs, as acknowledged by the absence of a subscript on $\mathbf{Z}$.

Our low-rank approximations are themselves GPs and therefore have an associated set of kernel and likelihood parameters. We collectively denote these as $\boldsymbol{\theta}$. Optimising the marginal log-likelihood of this model is intractable, so we instead use VI to form an ELBO (Jordan et al., 1998). The ELBO lower bounds the marginal log-likelihood and maximising its value is analogous to minimising the Kullback-Leibler divergence from the approximate low-rank processes to the true model. To achieve this, we optimise the model's hyperparameters and the inducing points' values using a first-order gradient-based optimiser (e.g., Ba et al., 2015). The cost of inference now scales linearly in N and quadratically in M, giving a total cost of $\mathcal{O}(NM^2R^2)$. We now proceed to give a full derivation of the ELBO, noting that similar results have been derived in the literature for GP regression (Titsias, 2009), latent variable GPs (Pinder et al., 2021), and heteroscedastic GP models (Lázaro-Gredilla et al., 2011; Saul et al., 2016).

Let $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$ be a set of *inducing points* whose corresponding function evaluations are given by $\mathbf{U}_f = \{\mathbf{u}_f^{(r)}\}_{r=1}^R = \{f^{(r)}(\mathbf{Z})\}_{r=1}^R$ and $\mathbf{u}_g = g(\mathbf{Z})$.

The function evaluations $\mathbf{U}_f$ and $\mathbf{u}_g$ are termed *inducing variables* as they are themselves random variables assigned the prior distributions of

$$p\left(\mathbf{u}_f^{(r)} \,|\, \mathbf{Z}\right) = \mathcal{N}\left(\mathbf{u}_f^{(r)} \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{f^{(r)}}\right) \tag{6.6}$$

$$p(\mathbf{u}_g \,|\, \mathbf{Z}) = \mathcal{N}\left(\mathbf{u}_g \,|\, \mathbf{0}, \mathbf{K}_{\mathbf{uu}}^{g}\right) \tag{6.7}$$

$$\tag{6.8}$$

where

$$\mathbf{K}_{\mathbf{uu}}^{f^{(r)}} = k_{f^{(r)}}(\mathbf{Z}, \mathbf{Z}) \tag{6.9}$$

$$\mathbf{K}_{\mathbf{uu}}^{g} = k_g(\mathbf{Z}, \mathbf{Z})\,. \tag{6.10}$$

Similarly, $\mathbf{K}_{\mathbf{xx}}^{f^{(r)}} = k_{f^{(r)}}(\mathbf{X}, \mathbf{X}')$ and $\mathbf{K}_{\mathbf{xx}}^{g} = k_g(\mathbf{X}, \mathbf{X}')$. When incorporating the inducing variables into our joint prior, we assume that our latent functions are independent

$$p(\mathbf{F}, \mathbf{g} \,|\, \mathbf{U}_f, \mathbf{u}_g) = p(\mathbf{F} \,|\, \mathbf{U}_f) p(\mathbf{g} \,|\, \mathbf{u}_g)\,, \tag{6.11}$$

where $\mathbf{F}$ is a matrix whose r^{th} column corresponds to the evaluation of $\mathbf{f}^{(r)} = f^{(r)}(\mathbf{X})$ and $\mathbf{g} = g(\mathbf{X})$. By the consistency of a GP, we can be sure that augmenting the GP's joint prior with the inducing variables and marginalising them out will leave no imprint on the true joint distribution.

The marginal log-likelihood of the model presented in Equations (6.4)–(6.5) is

$$\log p(\mathbf{Y}) = \sum_{r=1}^{R} \log \int p(\mathbf{y}_r \,|\, \mathbf{f}^{(r)}, \mathbf{g}) p(\mathbf{f}^{(r)}, \mathbf{g} \,|\, \mathbf{u}_f^{(r)}, \mathbf{u}_g) p(\mathbf{u}_f^{(r)}) p(\mathbf{u}_g) \mathrm{d}\mathbf{f}^{(r)} \mathrm{d}\mathbf{g} \mathrm{d}\mathbf{u}_f^{(r)} \mathrm{d}\mathbf{u}_g \tag{6.12}$$

To enable a tractable model, our approach is to introduce a variational approximation to the model's posterior distribution. By assuming that the latent variables $\mathbf{F}$ and $\mathbf{g}$ factor within the variational posterior distribution, we are then able to enable tractable computations (Saul et al., 2016). We will now proceed to derive an ELBO term following the approaches given in Titsias (2009); Lázaro-Gredilla et al. (2011) that will allow us to optimise the parameters of our variational approximation such that they offer the best approximation of the true posterior. We begin by defining the augmented joint prior

$$p(\mathbf{F}, \mathbf{g}, \mathbf{U}_f, \mathbf{u}_g \,|\, \mathbf{y}) \approx p(\mathbf{F} \,|\, \mathbf{U}_f) p(\mathbf{g} \,|\, \mathbf{u}_g) q(\mathbf{U}_f) q(\mathbf{u}_g) \tag{6.13}$$

$$= p(\mathbf{g} \,|\, \mathbf{u}_g) q(\mathbf{u}_g) \prod_{r=1}^{R} p(\mathbf{f}^{(r)} \,|\, \mathbf{u}_f^{(r)}) q(\mathbf{u}_f^{(r)}) \,, \tag{6.14}$$

where $q\left(\mathbf{u}_f^{(r)}\right)$ and $q(\mathbf{u}_g)$ are multivariate Gaussian distributions of the form

$$q\left(\mathbf{u}_f^{(r)}\right) = \mathcal{N}\left(\mathbf{u}_f^{(r)} \,|\, \mathbf{m}_f^{(r)}, \mathbf{\Sigma}_f^{(r)}\right) \tag{6.15}$$

$$q\left(\mathbf{u}_g\right) = \mathcal{N}\left(\mathbf{u}_g \,|\, \mathbf{m}_g, \mathbf{\Sigma}_g\right) \,. \tag{6.16}$$

We now seek to obtain a tractable bound on the marginal log-likelihood from Equation (6.12). To achieve this, we first apply the assumptions of factorisation and latent prior independence to Equation (6.12) and incorporate Equation (6.13). Application of Jensen's inequality to the refactored marginal log-likelihood will then yield a tractable lower bound. Applying the assumptions of factorisation and independence the marginal log-likelihood from Equation (6.12) and incorporating

Equation (6.13) gives

$$\log p(\mathbf{Y}) = \sum_{r=1}^{R} \log \int p(\mathbf{y}_r \,|\, \mathbf{f}^{(r)}, \mathbf{g}) p(\mathbf{f}^{(r)} \,|\, \mathbf{u}_f^{(r)}) p(\mathbf{g} \,|\, \mathbf{u}_g) p(\mathbf{u}_f^{(r)}) p(\mathbf{u}_g) \mathrm{d}\mathbf{f}^{(r)} \mathrm{d}\mathbf{g} \mathrm{d}\mathbf{u}_f^{(r)} \mathrm{d}\mathbf{u}_g \tag{6.17}$$

$$\geq \sum_{r=1}^{R} \mathbb{E}_{q(\mathbf{f})q(\mathbf{g})} \left[\log p(\mathbf{y} \,|\, \mathbf{f}, \mathbf{g})\right] - \mathrm{KL}(q(\mathbf{u}_f^{(r)}) \,||\, p(\mathbf{u}_f^{(r)})) - \mathrm{KL}(q(\mathbf{u}_g) \,||\, p(\mathbf{u}_g)) \,. \tag{6.18}$$

The forms of $q\left(\mathbf{f}^{(r)}\right)$ and $q(\mathbf{g})$ are given by

$$q(\mathbf{f}^{(r)}) = \int p(\mathbf{f}^{(r)} \,|\, \mathbf{u}_f^{(r)}) q\left(\mathbf{u}_f^{(r)}\right) \mathrm{d}\mathbf{u}_f^{(r)} \tag{6.19}$$

$$= \int \mathcal{N}\left(\mathbf{f}^{(r)} \,|\, \mathbf{K}_{\mathbf{fu}}^{f^{(r)}} \left[\mathbf{K}_{\mathbf{uu}}^{f^{(r)}}\right]^{-1} \mathbf{u}_f^{(r)}, \mathbf{K}_{\mathbf{ff}}^{(r)} - \mathbf{Q}_{\mathbf{ff}}^{(r)}\right) q\left(\mathbf{u}_f^{(r)}\right) \mathrm{d}\mathbf{u}_f^{(r)} \tag{6.20}$$

$$q(\mathbf{g}) = \int p(\mathbf{g} \,|\, \mathbf{u}_g) q(\mathbf{u}_g) \mathrm{d}\mathbf{u}_g \tag{6.21}$$

$$= \int \mathcal{N}\left(\mathbf{g} \,|\, \mathbf{K}_{\mathbf{gu}}^{g} \left[\mathbf{K}_{\mathbf{uu}}^{g}\right]^{-1} \mathbf{u}_g, \mathbf{K}_{\mathbf{gg}} - \mathbf{Q}_{\mathbf{gg}}\right) q(\mathbf{u}_g) \mathrm{d}\mathbf{u}_g \,. \tag{6.22}$$

where $\mathbf{Q}_{\mathbf{ff}}^{(r)} = \mathbf{K}_{\mathbf{fu}}^{f^{(r)}} \left[\mathbf{K}_{\mathbf{uu}}^{f^{(r)}}\right]^{-1} \mathbf{K}_{\mathbf{uf}}^{f^{(r)}}$ and $\mathbf{Q}_{\mathbf{gg}} = \mathbf{K}_{\mathbf{gu}}^{g} \left[\mathbf{K}_{\mathbf{uu}}^{g}\right]^{-1} \mathbf{K}_{\mathbf{ug}}^{g}$.

By virtue of the likelihood function being Gaussian, we can analytically evaluate the expectation in Equation (6.18) using the results outlined in Titsias (2009). Further, both of the KLD terms in Equation (6.18) are being evaluated on a pair of multivariate Gaussian distributions, meaning that they have an analytical form that can easily be computed. As such, when we bring these results together, we are provided with the following ELBO term

$$\mathcal{L}_q^* = \sum_{i=1}^{R} \sum_{j=1}^{R} \Big[\log \mathcal{N}\left(\mathbf{y} \,|\, \mathbf{0}, \sigma^2 \mathbf{I}_n + \mathbf{Q}_{\mathbf{gg}} + \delta_{i,j} \mathbf{Q}_{\mathbf{ff}}^{(i)}\right) - \frac{1}{2\sigma^2} \operatorname{trace}\left(\mathbf{K}_{\mathbf{xx}}^{g} + \mathbf{Q}_{\mathbf{gg}} - \delta_{i,j}\left(\mathbf{K}_{\mathbf{xx}}^{f^{(i)}} + \mathbf{Q}_{\mathbf{ff}}^{(i)}\right)\right) \Big] , \tag{6.23}$$

where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise i.e., a Dirac delta function.

In the forthcoming sections, we use the posterior distribution of the latent function g to emulate the latent behaviour of a climate model. The posterior distributions that correspond to the latent functions $\{f^{(r)}\}$ help diagnose the emulator's behaviour; however, they are not used in the following sections. As such, we will not discuss them further.

6.4.2 Weighting models

With a set of posterior distributions now learned for each model, the next step in our framework is to score each of the models. To do this, we make use of the observational data that was introduced in Section 6.3. Under our approach, we assign a higher score $c \in \mathbb{R}$ to models where the observed data is more plausible under the model's predictive density. To quantify this, we consider three metrics: EPLD, KSD, and CRPS that we present here before empirically exploring each of them in Section 6.5.

Expected predictive likelihood density The EPLD metric measures how well an emulator's posterior distribution explains the observed data points. For EPLD, a higher score is better and we, therefore, compute its negative value at each of the K index points by

$$\mathbf{c}_{\text{EPLD}} = \mathbb{E}\left[\log p(\mathbf{y}^\star \mid \mathbf{f}, \widehat{\mathbf{y}})\right] \tag{6.24}$$

$$\approx \frac{1}{200} \sum_{i=1}^{200} \log p(y_i^\star \mid \mathbf{f}, \widehat{\mathbf{y}})\,. \tag{6.25}$$

A larger EPLD score will correspond to a better performing model. However, to be consistent with other metrics, we will use the negative of this value as our score.

Kernel stein discrepancy KSD is a statistical divergence metric that can be used to measure how well an empirical density that is defined over a finite sample set approximates a true distribution (Liu et al., 2016a; Gorham et al., 2015). Letting p be the emulator's posterior distribution, q be the empirical density over our sample set $\mathbf{y}^\star$, and $k(x, x') = (\alpha^2 + \|x - x'\|_2^2)^\beta$ be the inverse multiquadratic kernel with $\alpha > 0$ and $\beta < 0$ to give

$$\mathbf{c}_{\text{KSD}} = \mathbb{E}_{y_i^\star, y_j^\star \sim q} \left[\mathcal{A}_p^{y_i^\star} \mathcal{A}_p^{y_j^\star} k(y_i^\star, y_j^\star) \right] \tag{6.26}$$

where the Stein operator $\mathcal{A}_p$ is given by

$$\mathcal{A}_p^{y_i^\star} = k(y_i^\star, \mathbf{y}^\star) \nabla_{y_i^\star} \log p(y_i^\star) + \nabla_{y_i^\star} k(y_i^\star, \mathbf{y}^\star) \,. \tag{6.27}$$

KSD is a strictly positive metric and a score of 0 is attained if, and only if, the samples were generated exactly from the target distribution. For a more rigorous presentation of KSD, we refer the reader to Section 2.2.

Continuous ranked probability score When evaluating the performance of a probabilistic forecast, CRPS is a commonly used metric that can intuitively be thought of as the squared difference between the empirical CDF of over the observations and the CDF of the emulator's predictive posterior distribution (Brown, 1974; Matheson et al., 1976). We can formalise this as follows

$$\mathbf{c}_{\text{CRPS}} = -\int_{-\infty}^{\infty} \left(p(x) - \mathbb{1}_{\mathbf{y}^\star \geq x}\right)^2 \mathrm{d}x \,. \tag{6.28}$$

When the predictive posterior distribution p is a Gaussian with mean μ and standard deviation σ, the CRPS has the following analytical form

$$\mathbf{c}_{\text{CRPS}} = \sigma \left[\frac{1}{\sqrt{\pi}} - 2\varphi\left(\frac{x-\mu}{\sigma}\right) - \frac{x-\mu}{\sigma}\left(2\Phi\left(\frac{x-\mu}{\sigma}\right) - 1\right)\right] , \tag{6.29}$$

where φ and Φ denote the PDF and CDF of a unit Gaussian distribution, respectively. As with KSD, CRPS is a strictly positive metric.

Proper scoring rules EPLD, KSD, and CRPS are three evaluation functions that we could use to measure the quality of our emulators for a set of observations. However, it is non-trivial how we should choose between them. Whilst we empirically explore this in Section 6.5, observing this problem through the lens of scoring rules can equip us with a more general framework for evaluating the quality of a set of emulators.

If we assume that our observations are given by the latent distribution Q, then a *scoring function* γ_S is a function based on a *scoring rule* S that can be defined by

$$\gamma_S(P, Q) = \mathbb{E}_Q \left[S(P, Q) \right] , \tag{6.30}$$

where $P, Q \in \mathcal{Q}$ are two probability distributions over the same space where P is used to denote the probabilistic forecast of the model.

Within the literature, scoring rules and scoring functions are presented as both negatively and positively oriented. To align with the orientation of EPLD, KSD, and CRPS, we here focus on the negatively oriented case where a smaller score is indicative of a better model. In this case, a scoring rule S is termed a *proper scoring rule* if and only if $\gamma_S(P, Q) \geq s_\gamma(Q, Q)$ holds for all $P, Q \in \mathcal{Q}$. Further, the scoring rule is *strictly proper* if the equality $\gamma_S(P, Q) = s_\gamma(Q, Q)$ is satisfied if

and only if $P = Q$ (Savage, 1971; Gneiting et al., 2007).

Intuitively, a scoring rule is proper if the best expected value is attained by predicting the observations' distribution and strictly proper if no other prediction can achieve this distribution. Within our emulator framework, comparing models using a proper scoring rule allows us to be confident that the model whose predictive distribution has the highest fidelity with the observations will receive the largest weight in the ensemble. Both EPLD and CRPS are proper scoring rules; however, as we shall see in Section 6.5, ranking models using KSD yields a well-performing ensemble and further work should be done to theoretically investigate whether KSD is a proper scoring rule.

6.4.3 Computing weights

For each of the three scoring metrics, the process is repeated independently for each model, leaving us with a set of W score vectors, each of length K. We transform these scores into weights $\boldsymbol{\beta}$ by calculating the relative score of each model w at each index location k through[3]

$$\beta_{w,k} = 1 - \frac{c_{w,k}}{\sum_{w=1}^{W} c_{w,k}} \,. \tag{6.31}$$

Clearly, if we were to compute $\sum_{w=1}^{W} \beta_{w,k}$ then we would yield a K-long vector of ones, thus satisfying our requirement from Section 6.2.1, that weights should sum to one.

[3] As the scores outlined in Section 6.4.2 are negatively oriented, we subtract each term below from 1 to ensure the *best* model is assigned the largest weight.

6.4.4 Creating an ensemble

Now that we have inferred a GP posterior distribution for each model and assigned it a relative weight vector for each index location, the only remaining task required to compute our ensemble is to combine the set of posterior distributions. Having taken care to work under a probabilistic framework thus far, we seek to ensure that our ensemble takes into account not only the mean of each posterior distribution, but also its variance, thus ensuring that our final ensemble will exhibit well-characterised uncertainty estimates. For this reason, we compute the Bures barycentre from Section 6.2.1 where the constituent set of probability distributions is the individual GP posterior distributions. We use the weights from Section 6.4.2 to assign a larger amount of probability mass to those models that represented the underlying observations accurately and conversely penalise models which are unrepresentative of the truth.

Although we are computing the Bures barycentre over a finite index set here, the Bures barycentre is well defined over a set of infinite dimension (Mallasto et al., 2017; Masarotto et al., 2019). Such a theoretical result guarantees to us that computing the Bures barycentre here as our final ensemble will yield a unique and well-defined function if, theoretically, the index set size was to tend toward infinity.

6.5 Experimental results

In this section, we empirically validate our proposed ensemble methodology and present initial results that project global mean surface temperature. Fundamentally, this involves fitting a 1-dimensional temporal regression model. Soon, we plan to conduct further experiments to evaluate global surface temperature in a spatiotemporally resolved manner using the workflow presented in this chapter.

6.5.1 Projecting mean surface temperature

In this section, we answer the question "*How will the global mean surface temperature change between now and 2100*". To achieve this, we apply our model from Section 6.4 to climate model outputs that have been constrained by each of the five SSPs and the set of RCPs for which each climate model was simulated for.

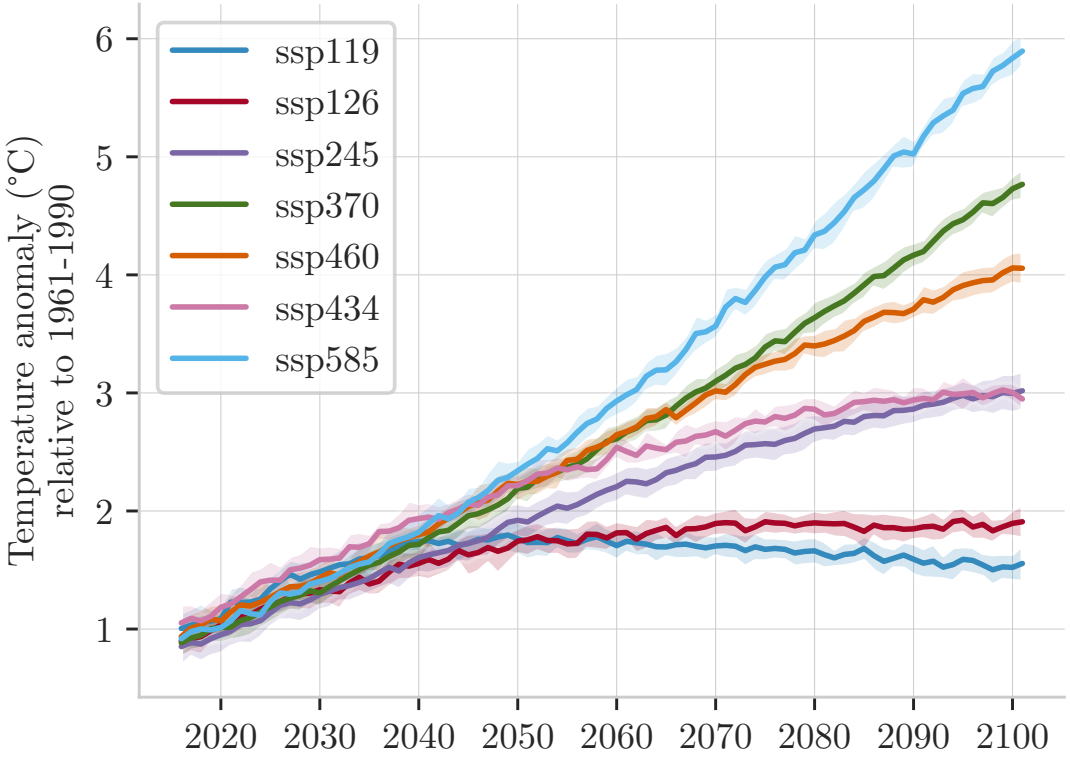


Figure 6.4: The predictions of our ensemble for the global mean surface temperature constrained by each of the five SSPs. The solid line represent the ensemble's predictive mean and the surrounding shaded region represents one standard deviation.

In Figure 6.4, we plot the predictions of our ensemble for the global mean surface temperature constrained by each of the five SSP. For all scenarios, we see an increase in global mean surface temperature until 2040. In 2040, even the most optimistic scenario, SSP119, sees an increase in global mean surface temperature of 1.5°C. It is of significance, that limiting global mean surface temperature to

2°C with a goal of 1.5°C is a fundamental statement in the 2015 Paris Agreement (UNFCCC, 2015).

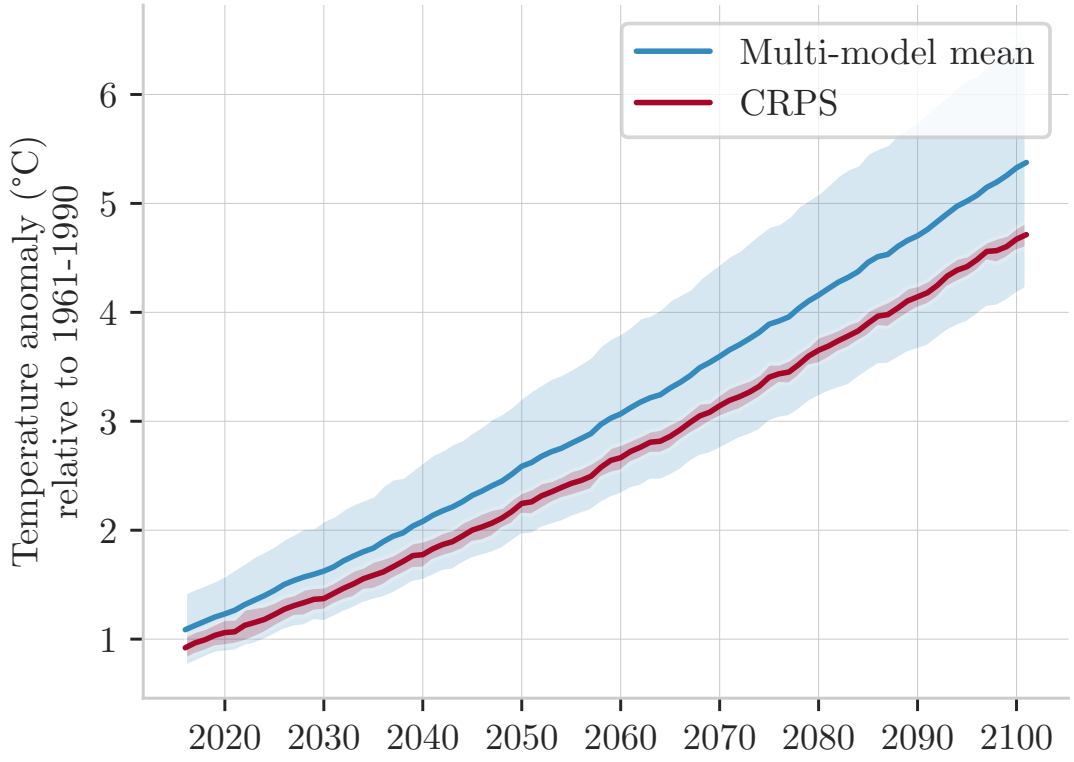


Figure 6.5: Comparison of the projections of our ensemble (red) to the multi-model mean (blue). The data used in each model has been constrained according to SSP370. The solid line represent the ensemble's predictive mean and the surrounding shaded region represents one standard deviation.

Projections for each SSP begin to diverge from one another after 2040. The most pessimistic scenario, SSP585 that sees rapid and unconstrained economic growth and energy consumption, results in a global mean surface temperature increase of 5.9°C by 2100. Such warming would be catastrophic for the planet and, at warmings of this magnitude, Earth would be incompatible with human civilisation. Conversely, by the projections given by our model, we can see that if we follow SSP119 or SSP126, we would be able to suppress global mean surface temperature increases to less than 2°C by 2100. In between these two extremes, we see that

all scenarios yield a global mean surface temperature of between 3 and 4.8°C by 2100.

An important observation to make when interpreting the projections from our model is that they should not be treated as binary outcomes. It is untrue to state that if Earth's warming can be limited to less than 1.5°C, then the human civilisation will be fine and warming of 2°C will result in the collapse of humanity. Instead, the correct way to interpret the results is that every increase in warming corresponds to an increased risk of catastrophic events occurring (Kopp, 2021).

Within the climate modelling community, the multi-model mean (MMM) is the default ensemble model that is used in the Intergovernmental Panel on Climate Change (IPCC) reports. For this reason, we compare our ensemble to the MMM and argue for the benefits of our proposed approach in the context of both precision and interpretability. The climate models and corresponding realisation sets used to compute the MMM and our ensemble are equal and correspond to SSP370. Observing Figure 6.5, we can see the difference in projections that our ensemble model yields in comparison to those given by the MMM. The variance of the MMM's predictions is significantly wider than that of our model as it is capturing the variance across all models' projections. In contrast, the variance of our model is a characterisation of the range of possible projections that could be made by our ensemble model. In the framework of ensemble modelling, we argue that the variances produced by our model are significantly more useful than those of the MMM as they inform the practitioner of a precise range of possible temperatures that could be experienced, not the range of model outputs, as is the interpretation of the MMM.

The projections given by our model are strictly less than those given by the MMM. This can be explained by the fact that a subset of climate models are biased towards

overestimating surface temperature due to complex dynamics required to resolve cloud formations (Zelinka et al., 2020). We follow convention and refer to these as *hot models* (Hausfather et al., 2022). However, the MMM does not assign more credibility to models that are more representative of the truth and, consequently, the hot models over-contribute to the projections given by the MMM. In contrast, the weights assigned to each climate model within our ensemble are determined by the accuracy of the model with respect to the fidelity between its hindcast and the true observations. As a result, the hot models are penalised and the resulting projections are more conservative when compared to those of the MMM. We see this as a further benefit of our proposed approach.

In Figures 6.4 and 6.5, we note that the standard deviation of our ensemble does not increase as we extrapolate into the future. This may seem counter-intuitive as we would expect the uncertainty of a model to increase as we move further away from the data that it was trained on. However, in this work, all meteorological effects have been removed from the data in the preprocessing. Further, by constraining the model runs to a single SSP, all socio-economic variation has been explained. With these two constraints applied, the variance between the constituent models becomes small which, consequenlty, makes the ensemble model's variance narrow.

6.5.2 Ensemble validation

Within the above work, we used CRPS to weight our emulators. However, we also proposed EPLD and KSD in Section 6.4.2. We justify the use of CRPS for two reasons, firstly, despite offering good performance, KSD is not a proper scoring rule and is therefore not accompanied by the same guarantees as CRPS. Second, the range of values that EPLD can take are broad and its value is very sensitive. We see this in Figure 6.6 where the weights assigned to each model by EPLD are

very rough and many models will not contribute to the ensemble prediction at all as their weight is zero. This behaviour is undesirable from a climate model ensemble as we know that all models are reasonable approximations of the truth and, therefore, should at least offer small contributions to the ensemble prediction. We acknowledge the uniformity of the weights plotted in Figure 6.6, however, when extending the model to the spatiotemporal setting we would expect to see more inter-model variation as it is well known that all models do not perform equally well across the globe.

6.5.3 Validating our model

Within this work, we have introduced a novel GP methodology that enables us to probabilistically learn a latent function from a set of vector-valued observations. The posterior distribution of this latent function is then used as a component of our final ensemble. For this reason, it is therefore prudent to evaluate our model and validate two points: 1) the GP is capable of learning the underlying latent function, and 2) the variance of the GP's posterior distribution is calibrated.

Data To empirically validate the model from Section 6.4.1, we simulate four datasets where the latent function in each is a single realisation of a Matérn process. The lengthscale used in the latent functions' draws are 0.1, 0.2, 0.5, and 1. 10 realisations are produced from the latent function where each is corrupted by a homoscedastic, zero-mean Gaussian noise vector with a scalar variance of 0.1. The relationship between the latent function and each realisation is further corrupted by shifting each realisation by a constant factor whose value is drawn from a uniform distribution with limits of -1.5 and 1.5. We plot the four datasets used in Figure 6.7.

A priori, we should expect our model to perform better on datasets whose latent

function was drawn from a Matérn process with a larger lengthscale as this will correspond to a smoother function that will be easier to model, particularly when the ratio of data points to inducing points is high.

Predicitve quality The first question we seek to answer is "*How well can our model recover the true latent function using only a set of realisations?*". To this end, we simulate 10 realisations from the four Matérn processes described above. We further vary the number of data points within a single realisation to be one of 50, 100, or 300. The latent function's lengthscale and the number of data points within a single realisation are our independent variables. For each of the four datasets, we fit our model from Section 6.4.1 and report the R^2 coefficient to assess how much of the variability within the latent function is explained by our model. This is our dependent response variable.

From Figure 6.8 we can see that our model can consistently model the variance in the latent function to a high fidelity. The only time we see a degradation in performance is when the latent function's lengthscale is 0.1 or 0.2, and we use just 10 inducing points in our model. This is not surprising though as we should never expect to recover such rough functions with so few inducing points as the GP will be forced to smooth out the underlying function's roughness when interpolating from one inducing point to the next.

Posterior calibration We further validate our model by answering the question "*How well calibrated is the uncertainty given by our model when representing the latent function?*". We use the same data generating process as above, however, we now let our independent variables be the true latent function's lengthscale and the number of realisations within a dataset. To test calibration, we query our GP at a set of test locations and report the percentage of times that the true response

value fell within our model's 95% credible interval. An over-confident model would report values less than 95%, whilst an under-confident model would report values larger than 95%.

From Figure 6.9 we can see that the uncertainty estimates of our model are well calibrated. There is a tendency for the posterior distribution of our model to be under-confident in its predictions. However, this is a common artefact of variational GP models of the form given in Titsias (2009) due to the different rank of $\mathbf{K}_{\mathbf{xx}}^{g}$ and $\mathbf{K}_{\mathbf{xx}}^{f^{(r)}}$ when compared to $\mathbf{Q}_{\mathbf{gg}}$ and $\mathbf{Q}_{\mathbf{ff}}^{(r)}$ (Turner et al., 2011; M. Bauer et al., 2016). Further, as the number of inducing points grows, the calibration of the model's posterior improves.

6.6 Conclusions

Within this chapter, we have developed a probabilistic ensemble model that is capable of propagating uncertainty from the underlying climate models' output through to the ensemble model's predictions. To achieve this, we have developed a novel GP method that allows us to infer a posterior distribution over the realisations produced by a single climate model. Further, we have leveraged connections between our GP and the literature within the fields of optimal transport and scoring rules to effectively weight and combine our set of GP posteriors into an ensemble. Finally, we have applied our ensemble methodology to surface temperature data to produce probabilistic projections of global mean surface temperature from the present day through to 2100.

The model we have developed is flexible and modular, meaning that its usage is not limited to just modelling surface temperature. Through a different likelihood function or probabilistic model altogether, the hierarchical GP presented in

Section 6.4.1 can be adapted to model other environmental variables such as precipitation or air pollution. Further, we explored the use of three different scoring rules to weight models within our ensemble but the literature on scoring rules is vast, and exploring the use of other scoring rules is a natural extension of this work.

Finally, a key finding of the most recent IPCC report is that "*there is no single 1.5°C warmer world*". This means that the warming of our planet is increasing at different rates in different parts of the world. Many regions of the world have already surpassed a 1.5°C increase in surface temperature compared to pre-industrial levels. For this reason, the immediate next step in this work is to extend our model to the spatiotemporal domain. Doing so will allow us to create projections of surface temperature in a grid of locations across the globe. These projections can then be used to report the likely range of surface temperature that each region of the world is likely to experience. The GP foundations of our model make it a perfect candidate for such a task as, through careful kernel design, we will be able to construct a model that can capture the spatial and temporal dependencies within the climate models' output.

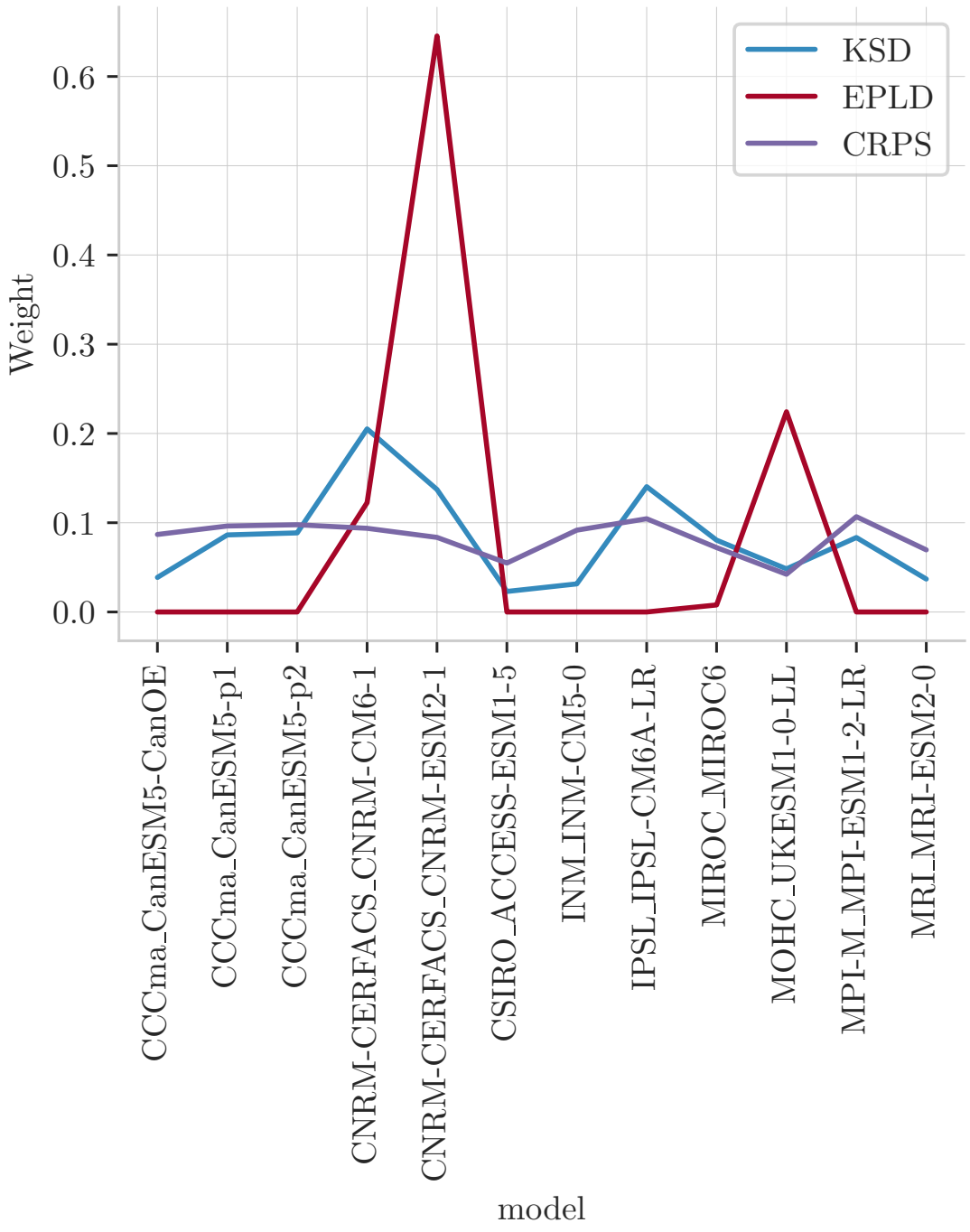


Figure 6.6: The average weight assigned to each climate model used when ensembling the SSP370 scenario.

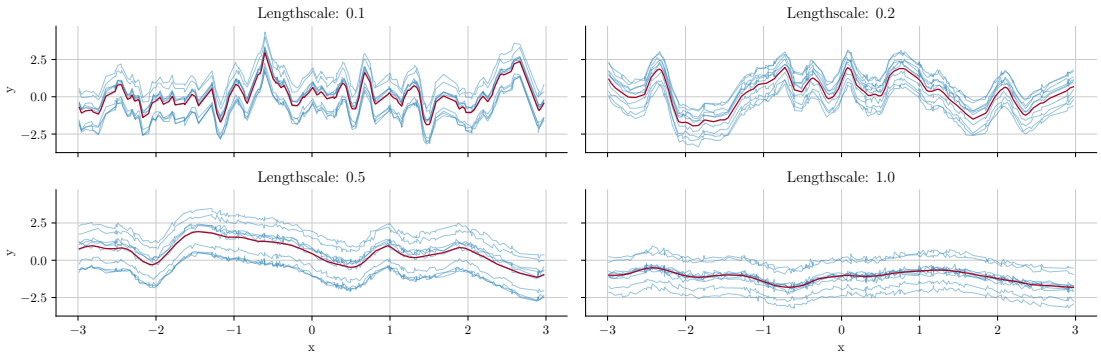


Figure 6.7: Four simulated datasets used for model validation in Section 6.5.3. The value of ℓ in each figure denotes the lengthscale used in the latent Matérn function from which each dataset is generated. The red line represents the true latent function and each blue line is a noisy realisation of the latent function.

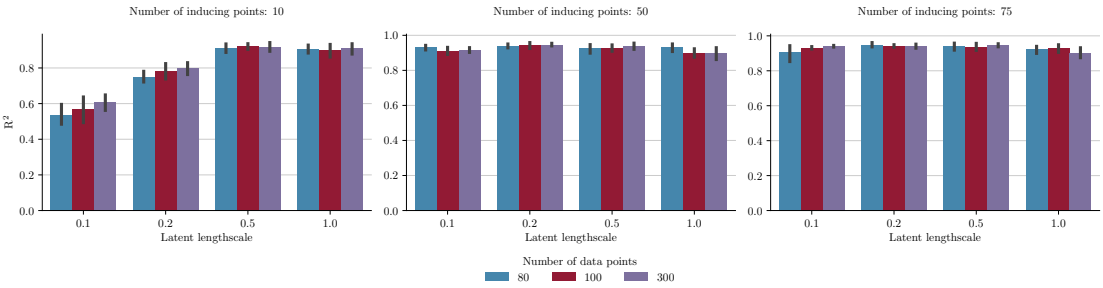


Figure 6.8: R^2 scores of our model plotted as a function of latent function's true lengthscale and the number of data points within each realisation.

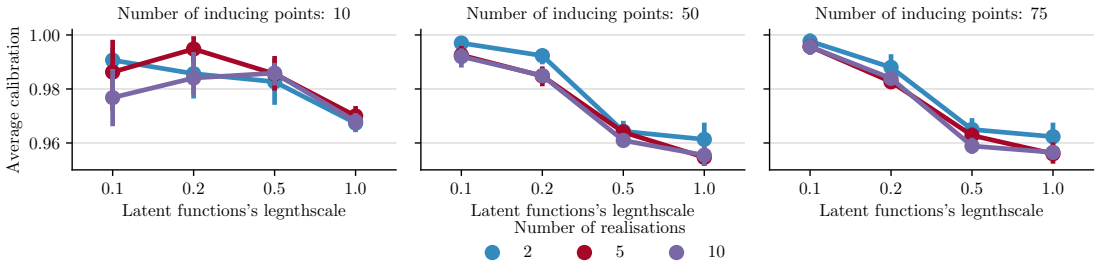


Figure 6.9: The posterior calibration of our model reported as a function of the number of realisations.

Chapter 7

Conclusions

This thesis is concluded by first summarising the key findings of the research and discussing the unifying themes that relate chapters together. We then proceed to close the thesis by discussing the implications of this work for future research and outline potential directions of research.

7.1 Discussion of main results

To understand the process of scientific work, George Box advocated the use of a loop to aid in understanding physical processes (Box et al., 1962). This workflow has been assigned the name *Box's loop* in more recent literature (Blei, 2014). We adapt this loop for the process of developing GP models in Figure 7.1 and use this framework to consolidate the key findings of this thesis.

The process of science is iterative, and whilst we may devote more time to certain elements of Figure 7.1 than others, the whole process must be considered in order to develop a successful model. We begin by devising our model. For a GP, this requires us to specify a kernel and mean function, perhaps with prior distributions

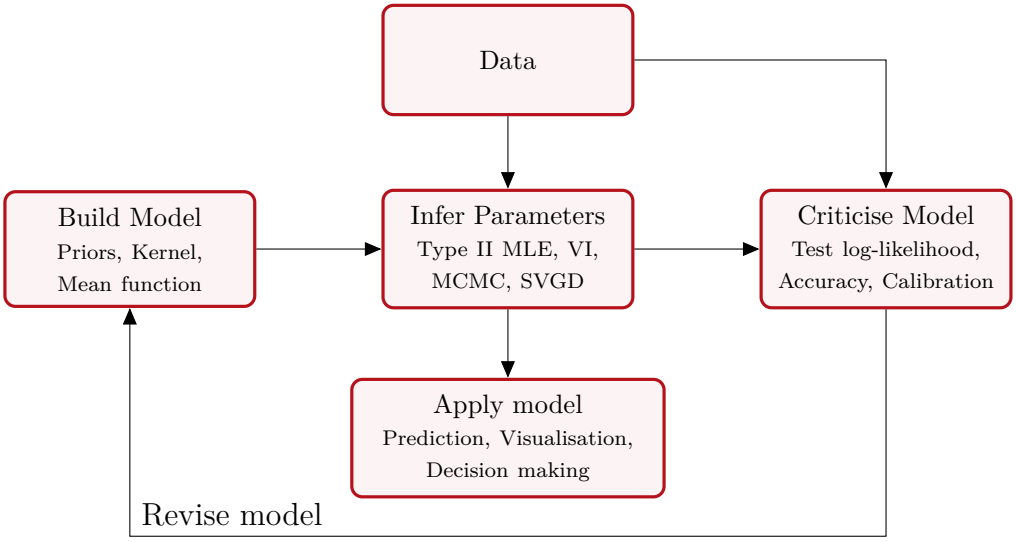


Figure 7.1: Box's loop of model development adapted for a Gaussian process workflow.

placed on the parameters of these functions. We then use an observed dataset to infer the optimal parameters of our model. For conjugate regression problems with small data, this task is likely straightforward optimisation of the marginal log-likelihood. However, for more complex models, this may involve the use of an approximate inference algorithm. We then criticise, or evaluate, our model using a test dataset that is disjoint to the data used for inference. The criteria for which we evaluate our model will vary greatly depending upon the task at hand. If we are satisfied by our model's performance, then we may apply it in the context of prediction, visualisation or as part of a decision-making loop. If we are not satisfied, then we may revise our model. In practice, this may involve building more complex kernel functions or altering our inference scheme to better identify the parameters of our model.

Build models Within this thesis, we have spent time considering each component of Figure 7.1, and across all chapters, we have constructed GP models. In Chapter 2 we built a spatiotemporal GP with a custom kernel function to

precisely represent our beliefs surround the underlying dynamics of pollution. In Chapter 3 this involved the use of a kernel function that would allow us to capture the correlation structure of vertices in a network. Meanwhile, Chapter 4 detailed the construction of a latent variable model that could accurately represent the implicit structure contained with a hypergraph. Finally, Chapter 6 saw us develop a hierarchical GP that could identify and propagate the latent uncertainty across a set of realisations into a single latent GP function.

Infer parameters In Chapter 2, we investigate the use of SVGD as an alternative form of inference for GP models whose latent function cannot be analytically integrated. Meanwhile, Chapters 3–4 and Chapter 6 detailed the derivation of a variational approximation to the true posterior. Such a form of inference allowed us to include a heteroscedastic noise term in Chapter 3, embed binary hypergraph data into a latent space in Chapter 4, and compose a sequence of GPs in a sparse hierarchical model in Chapter 6.

Data and model criticism Within each chapter, we have considered complex, real-world datasets to evaluate and critique our model. In Chapter 2, we considered spatiotemporal air quality sensor measurements throughout the time window of the United Kingdom's Covid-19 lockdown. This data was supplemented with covariates such as spatially-indexed land type and spatiotemporally-indexed wind speeds. Chapter 3 considered mobile air quality data which we incorporated into a network to describe the streets of London. This was proceeded by our campaign to collect a mobile dataset of ultrafine particulate matter in Lancaster, UK. Hypergraph representations of political networks within the United States of America and Peru were considered in Chapter 4 along with a comparison against their pairwise graph counterparts. Finally, the vector-valued output of climate

simulators was used in Chapter 6. In each case, the data has been collected, processed and analysed to provide a fair and robust evaluation of our model through a process described in the relevant chapter. To criticise our GP models, metrics such as test log-likelihood, posterior calibration and expected calibration error have been used as these measures allow us to assess not only the quality of our GP's posterior mean but also the posterior variance; an often overlooked area of GP model criticism.

Apply model Model application has centred around trying to answer one, or more, research questions. In Chapter 2 we sought to understand the extent to which the United Kingdom's Covid-19 lockdown affected air quality levels. In Chapter 3, we quantified the NO_2 exposure that one would experience whilst walking between any two locations in Mitcham, London. We also applied our model to identify the roads with the greatest NO_2 levels in Mitcham. We tried to understand the latent structure of political networks in Chapter 4. This not only provided us with vectors that could more easily be fed into downstream machine learning tasks but it also provided us with a way to visualise the dependency structure of the hypergraph. Finally, in Chapter 6, we used our probabilistic ensemble model to build probabilistic global surface temperature projections.

Across all of this work, there has been a strong focus on supplementing any research with efficient software containing a set of abstractions that make it easy to use. This principle is detailed in Chapter 5 where we discuss the design of the `GPJax` software and how it provides practitioners with a tool that can enable them to more easily conduct their research into GP methodology.

7.2 Future directions

Research into accelerating inference in GP models blossomed in the early 2000s through the works of Snelson et al. (2005) and Titsias (2009), to highlight just two seminal works from this period. The field has continued to mature until the late-2010s through works such as Hensman et al. (2013a). However, as a community, inference in GPs is now such that millions of data points can be handled in a matter of minutes using consumer-grade laptops (e.g., K. Wang et al., 2019; Hensman et al., 2018). As a consequence of this, the GP community has reached an inflection point where the focus of research is shifting from inferential efficiency to effectively handling complex data supports with non-trivial modelling constraints.

Within this thesis alone, we have considered modelling spatiotemporal data in Chapter 2, network data in Chapter 3, hypergraphs in Chapter 4, and hierarchical vector-valued data in Chapter 6. Beyond this thesis, research has been conducted to enable modelling with GPs on the following supports: Riemannian manifolds (Borovitskiy et al., 2020), Lie groups (Azangulov et al., 2022), vector fields (Hutchinson et al., 2021; Lange-Hegermann, 2021), and probability measures (Meunier et al., 2022). Beyond this, recent work in Lu et al. (2022) has extended the additive GP model (Duvenaud et al., 2011) to orthogonally constrain the additive kernel terms, thus making the relationship between inputs and outputs fully explainable. Across all of this work, the common theme is the shift from trying to develop faster inference schemes to develop more expressive models that can handle increasingly more complex data.

This paradigm shift is unsurprising as the way we collect data has evolved significantly in recent years and we now have access to volumes of descriptive data that is suitable for modelling. As such, machine learning models have transitioned from isolated academic environments into businesses seeking to extract value from

their data. In such environments, the singular priority is seldom speed and is instead a complex balance of explainability, accuracy, and speed.

Complimenting the existing body of literature, the work presented in this thesis could naturally be extended in several ways. Lifting the work given in Chapter 3 to an experimental design setting would be a natural starting point. Whilst monitoring pollution levels using mobile sensors is of high utility, as shown in Chapter 3, developing schemes that enable this data collection requires a significant investment in infrastructure. Consequently, much of the interest still concerns installing static measurement stations. The work presented in Chapter 3 could be used as a tool to help decide where next to place a measurement station as, often, sensors are placed in areas where we a priori anticipate pollution levels to be high. Positioning sensors in this way results in a biased estimate of the expected pollution levels in the area. Therefore, one solution would be to place sensors in areas where the predictive variance of the model presented in Chapter 3 is greatest. Pollution maps could then be built by modelling a network of sensors where the predictive mean of the street-level GP presented in Chapter 3 is used as a control variate to minimise the variance of the full model.

Chapter 4 enables a natural extension to embed network data into a manifold. We assumed no explicit geometrical structure of latent space in this work, but there is a body of literature that has considered embedding networks into elliptic and hyperbolic manifolds (e.g., A. L. Smith et al., 2019). Using the kernel given in (Borovitskiy et al., 2020), it may be possible to assume that the latent space has a Riemannian metric and the assumption of a Euclidean latent space is relaxed. Equipping the latent space with this structure would allow a rich inspection of the latent space to be accomplished using the tools developed by the probabilistic geometry community (Tosi et al., 2014; Arvanitidis et al., 2017).

We considered the problem of modelling global surface temperature in Chapter 6. However, throughout this work, we ensured that our modelling approach was flexible enough to incorporate any climate model's output. Consequently, extending the model presented in Chapter 6 to the modelling of precipitation or pollution data would be a natural extension with a large potential impact on the environmental sciences. Logistical challenges will present themselves whilst obtaining the relevant dataset, and some of the modelling assumptions made in Chapter 6 may be inappropriate for the new dataset e.g., precipitation data is zero-inflated and positive and a non-Gaussian likelihood function will undoubtedly be required. However, due to the modularity of our framework, these assumptions can be incorporated into the model.

Finally, the development of open source software is imperative for any significant progress to be achieved in machine learning. Therefore, the ongoing maintenance and contributions to packages such as `GPJax` Chapter 5 is one avenue of future research that would be of great benefit to the community. At present, `GPJax` is written in a way that is conducive to machine learning researchers being able to quickly prototype and develop new models. However, the abstractions that enable this are superfluous to applied scientists who simply wish to use a GP as part of their modelling workflow. Therefore, developing an abstraction of `GPJax` that provides a simple interface for applied scientists to leverage the efficiency of `GPJax` would be of great utility to the broader science community.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: a system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016. Cited on pages 46, 130, 131.

[2] M. Abramowitz and I. A. Stegun. Handbook of mathematical functions with formulas, graphs, and mathematical tables. national bureau of standards applied mathematics series 55. tenth printing. 1972. Cited on page 3.

[3] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. J. Kriegman, and S. J. Belongie. Beyond pairwise clustering. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005. Cited on page 120.

[4] M. Agueh and G. Carlier. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 2011. Cited on page 150.

[5] S.-I. Amari. Natural gradient works efficiently in learning. *Neural computation*, 1998. Cited on page 38.

[6] M. Amos, P. J. Young, J. S. Hosking, J.-F. Lamarque, N. L. Abraham, H. Akiyoshi, A. T. Archibald, S. Bekki, M. Deushi, P. Jöckel, D. Kinnison,

O. Kirner, M. Kunze, M. Marchand, D. A. Plummer, D. Saint-Martin, K. Sudo, S. Tilmes, and Y. Yamashita. Projecting ozone hole recovery using an ensemble of chemistry–climate models weighted by model performance and independence. *Atmospheric Chemistry and Physics*, 2020. Cited on page 147.

[7] M. Andjelković, B. Tadić, S. Maletić, and M. Rajković. Hierarchical sequencing of online social graphs. *Physica A: Statistical Mechanics and its Applications*, 2015. Cited on page 107.

[8] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 1950. Cited on page 46.

[9] G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017. Cited on page 178.

[10] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th annual foundations of computer science*. IEEE, 1995. Cited on page 142.

[11] I. Azangulov, A. Smolensky, A. Terenin, and V. Borovitskiy. Stationary kernels and Gaussian processes on Lie groups and their homogeneous spaces i: the compact case. *arXiv preprint arXiv:2208.14960*, 2022. Cited on page 177.

[12] J. Ba and D. Kingma. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. Cited on pages 7, 38, 154.

[13] I. Babuschkin, K. Baumli, A. Bell, S. Bhupatiraju, J. Bruce, P. Buchlovsky, D. Budden, T. Cai, A. Clark, I. Danihelka, C. Fantacci, J. Godwin, C. Jones, T. Hennigan, M. Hessel, S. Kapturowski, T. Keck, I. Kemaev, M. King, L. Martens, V. Mikulik, T. Norman, J. Quan, G. Papamakarios, R. Ring, F. Ruiz, A. Sanchez, R. Schneider, E. Sezener, S. Spencer, S. Srinivasan, W. Stokowiec, and F. Viola. The DeepMind JAX Ecosystem, 2020. URL: HTTP://GITHUB.COM/DEEPMIND. Cited on pages 131, 132.

[14] P. Barberá. Birds of the same feather tweet together: Bayesian ideal point estimation using Twitter data. *Political analysis*, 2015. Cited on page 104.

[15] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri. Networks beyond pairwise interactions: Structure and dynamics. en. *Physics Reports*, 2020. Cited on page 107.

[16] M. Bauer, M. van der Wilk, and C. E. Rasmussen. Understanding probabilistic sparse Gaussian process approximations. *Advances in neural information processing systems*, 2016. Cited on pages 30, 35, 169.

[17] T. Bayes. LII. an essay towards solving a problem in the doctrine of chances. by the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, 1763. Cited on page 2.

[18] J. Bennett. *OpenStreetMap*. Packt Publishing Ltd, 2010. Cited on page 84.

[19] C. Berge. *Graphs and hypergraphs*. North-Holland Pub. Co., 1973. Cited on page 106.

[20] A. Berlinet and C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2004. Cited on pages 46, 48, 61.

[21] R. Bhatia, T. Jain, and Y. Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 2019. Cited on page 150.

[22] C. M. Bishop. *Pattern recognition and Machine Learning*. Springer, 2006. Cited on pages 30, 109.

[23] C. M. Bishop and G. D. James. Analysis of multiphase flows using dual-energy Gamma densitometry and neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1993. Cited on page 112.

[24] BlackJax. BlackJAX; a sampling library designed for ease of use, speed and modularity. 2021. URL: HTTPS://GITHUB.COM/BLACKJAX-DEVS/BLACKJAX/. Cited on page 130.

[25] D. M. Blei. Build, compute, critique, repeat: data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 2014. Cited on page 173.

[26] D. Bolin. Spatial matérn fields driven by non-Gaussian noise. *Scandinavian Journal of Statistics*, 2014. Cited on page 106.

[27] D. Bolin, A. B. Simas, and J. Wallin. Gaussian Whittle-Matérn fields on metric graphs. *arXiv preprint arXiv:2205.06163*, 2022. Cited on page 106.

[28] V. Borovitskiy, I. Azangulov, A. Terenin, P. Mostowsky, M. Deisenroth, and N. Durrande. Matérn Gaussian processes on graphs. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021. Cited on pages 77, 84, 105, 113, 115, 121, 133.

[29] V. Borovitskiy, A. Terenin, P. Mostowsky, et al. Matérn Gaussian processes on Riemannian manifolds. *Advances in Neural Information Processing Systems*, 2020. Cited on pages 105, 177, 178.

[30] G. E. Box and W. G. Hunter. A useful method for model-building. *Technometrics*, 1962. Cited on page 173.

[31] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, version 0.2.5, 2018. URL: HTTP://GITHUB.COM/GOOGLE/JAX. Cited on page 130.

[32] L. Breiman. Bagging predictors. *Machine learning*, 1996. Cited on page 142.

[33] A. Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 2013. Cited on page 107.

[34] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021. Cited on page 100.

[35] T. A. Brown. *Admissible Scoring Systems for Continuous Distributions.* ERIC, 1974. Cited on page 159.

[36] W. Bruinsma. Stheno, 2022. URL: HTTPS://GITHUB.COM/WESSELB/STHENO. Cited on page 133.

[37] L. Brunner, R. Lorenz, M. Zumwald, and R. Knutti. Quantifying uncertainty in European climate projections using combined performance-independence weighting. *Environmental Research Letters*, 2019. Cited on page 147.

[38] T. D. Bui, C. Nguyen, and R. E. Turner. Streaming sparse Gaussian process approximations. *Advances in Neural Information Processing Systems*, 2017. Cited on page 36.

[39] D. Burt, C. Rasmussen, and M. van der Wilk. Convergence of sparse variational inference in Gaussian processes regression. *Journal of Machine Learning Research*, 2020. Cited on pages 36, 58, 71.

[40] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: a probabilistic programming language. *Journal of statistical software*, 2017. Cited on page 133.

[41] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 2001. Cited on page 93.

[42] L. H. Chen. Poisson approximation for dependent trials. *The Annals of Probability*, 1975. Cited on page 50.

[43] C. Cheng and B. Boots. Variational inference for Gaussian process models with linear complexity. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017. Cited on page 44.

[44] P. S. Chodrow, N. Veldt, and A. R. Benson. Hypergraph clustering: from blockmodels to modularity. *Science Advances*, 2021. Cited on page 125.

[45] C. Chu, K. Minami, and K. Fukumizu. The equivalence between Stein variational gradient descent and black-box variational inference. In *arXiv:2004.01822 [cs, stat]*, 2020. arXiv: 2004.01822. Cited on page 55.

[46] K. Chwialkowski, H. Strathmann, and A. Gretton. A kernel test of goodness of fit. In *International conference on machine learning*. PMLR, 2016. Cited on pages 49, 51.

[47] A. J. Cohen, M. Brauer, R. Burnett, H. R. Anderson, J. Frostad, K. Estep, K. Balakrishnan, B. Brunekreef, L. Dandona, R. Dandona, et al. Estimates and 25-year trends of the global burden of disease attributable to ambient air pollution: an analysis of data from the global burden of diseases study 2015. *The Lancet*, 2017. Cited on page 91.

[48] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*. PMLR, 2018. Cited on page 10.

[49] N. De Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. 1785. Cited on page 142.

[50] V. M. De Oliveira and J. Fontanari. Random replicators with high-order interactions. *Physical review letters*, 2000. Cited on page 107.

[51] M. Deisenroth and J. W. Ng. Distributed Gaussian processes. In *International Conference on Machine Learning*. PMLR, 2015. Cited on page 142.

[52] M. Deisenroth and C. Rasmussen. PILCO: a model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer, 2011. Cited on page 130.

[53] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. Cited on page 44.

[54] C. Deser, F. Lehner, K. B. Rodgers, T. Ault, T. L. Delworth, P. N. DiNezio, A. Fiore, C. Frankignoul, J. C. Fyfe, D. E. Horton, J. E. Kay, R. Knutti, N. S. Lovenduski, J. Marotzke, K. A. McKinnon, S. Minobe, Randerson, J. A, I. Screen, I. R. Simpson, and T. M. Insights from Earth system model initial-condition large ensembles and future prospects. *Nature Climate Change*, 2020. Cited on pages 146, 147.

[55] G. Detommaso, T. Cui, Y. Marzouk, A. Spantini, and R. Scheichl. A Stein variational Newton method. *Advances in Neural Information Processing Systems*, 2018. Cited on page 52.

[56] S. S. Dhomse, D. Kinnison, M. P. Chipperfield, R. J. Salawitch, I. Cionni, M. I. Hegglin, N. L. Abraham, H. Akiyoshi, A. T. Archibald, E. M. Bednarz, S. Bekki, P. Braesicke, N. Butchart, M. Dameris, M. Deushi, S. Frith, S. C. Hardiman, B. Hassler, L. W. Horowitz, R.-M. Hu, P. Jöckel, B. Josse, O. Kirner, S. Kremser, U. Langematz, J. Lewis, M. Marchand, M. Lin, E. Mancini, V. Marécal, M. Michou, O. Morgenstern, F. M. O'Connor, L. Oman, G. Pitari, D. A. Plummer, J. A. Pyle, L. E. Revell, E. Rozanov, R. Schofield, A. Stenke, K. Stone, K. Sudo, S. Tilmes, D. Visioni, Y. Yamashita, and G. Zeng. Estimates of ozone return dates from chemistry-climate model initiative simulations. *Atmospheric Chemistry and Physics*, 2018. Cited on page 147.

[57] P. J. Diggle, R. Menezes, and T.-l. Su. Geostatistical inference under preferential sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 2010. Cited on page 93.

[58] A. Duncan, N. Nuesken, and L. Szpruch. On the geometry of Stein variational gradient descent. *arXiv:1912.00894 [cs, math, stat]*, 2019. arXiv: 1912.00894. Cited on page 55.

[59] V. Dutordoir, N. Durrande, and J. Hensman. Sparse Gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*. PMLR, 2020. Cited on page 39.

[60] D. K. Duvenaud, H. Nickisch, and C. Rasmussen. Additive Gaussian processes. *Advances in neural information processing systems*, 2011. Cited on page 177.

[61] V. Eyring, S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, and K. E. Taylor. Overview of the coupled model intercomparison project phase 6 (CMIP6) experimental design and organization. *Geoscientific Model Development*, 2016. Cited on pages 147, 148, 151.

[62] J. Fairbrother, C. Nemeth, M. Rischard, J. Brea, and T. Pinder. GaussianProcesses.jl: a nonparametric Bayes package for the Julia language. *Journal of Statistical Software*, 2022. Cited on page 133.

[63] D. Foreman-Mackey. TinyGP, 2021. URL: HTTPS://GITHUB.COM/DFM/TINYGP. Cited on page 132.

[64] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 2001. Cited on page 142.

[65] Y. Gal, Y. Chen, and Z. Ghahramani. Latent Gaussian processes for distribution estimation of multivariate categorical data. In *International Conference on Machine Learning*. PMLR, 2015. Cited on page 119.

[66] T. Galy-Fajou, F. Wenzel, and M. Opper. Automated augmented conjugate inference for non-conjugate Gaussian process models. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, 2020. Cited on page 133.

[67] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in neural information processing systems*, 2018. Cited on page 132.

[68] D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017. Cited on page 64.

[69] A. E. Gelfand, P. Diggle, P. Guttorp, and M. Fuentes. *Handbook of spatial statistics*. eng. CRC Press, 2010. ISBN: 978-1-4200-7287-7. Cited on page 56.

[70] F. Giorgi and L. O. Mearns. Calculation of average, uncertainty range, and reliability of regional climate changes from AOGCM simulations via the "reliability ensemble averaging"(rea) method. *Journal of Climate*, 2002. Cited on page 147.

[71] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 2007. Cited on page 161.

[72] C. Gong, J. Peng, and Q. Liu. Quantile Stein variational gradient descent for batch Bayesian optimization. In *International Conference on Machine Learning*. PMLR, 2019. Cited on page 55.

[73] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 2014. Cited on page 108.

[74] J. Gorham and L. Mackey. Measuring sample quality with Stein's method. *Advances in Neural Information Processing Systems*, 2015. Cited on pages 52, 53, 62, 159.

[75] J. Gorham, A. Raj, and L. Mackey. Stochastic Stein discrepancies. *Advances in Neural Information Processing Systems*, 2020. Cited on page 49.

[76] GPy. GPy: a Gaussian process framework in Python, 2012. URL: HTTP://GITHUB.COM/SHEFFIELDML/GPY. Cited on page 132.

[77] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and R. Zemel. Cutting out the Middle-Man: Training and Evaluating Energy-Based Models without Sampling. *arXiv:2002.05616 [cs, stat]*, 2020. arXiv: 2002.05616. Cited on page 55.

[78] G. Green. *An essay on the application of mathematical analysis to the theories of electricity and magnetism*, volume 3. author, 1889. Cited on page 105.

[79] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. Cited on page 97.

[80] O. Hamelijnck, T. Damoulas, K. Wang, and M. Girolami. Multi-resolution multi-task Gaussian processes. *Advances in Neural Information Processing Systems*, 2019. Cited on page 79.

[81] O. Hamelijnck, W. Wilkinson, N. Loppi, A. Solin, and T. Damoulas. Spatio-temporal variational Gaussian processes. *Advances in Neural Information Processing Systems*, 2021. Cited on page 46.

[82] C. A. Hasenkopf, J. Flasher, O. Veerman, and H. L. DeWitt. OpenAQ: a platform to aggregate and freely share global air quality data. In *AGU Fall Meeting Abstracts*, 2015. Cited on page 83.

[83] Z. Hausfather, K. Marvel, G. A. Schmidt, J. W. Nielsen-Gammon, and M. Zelinka. Climate simulations: recognize the 'hot model' problem, 2022. Cited on page 166.

[84] M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes. Inference in deep Gaussian processes using stochastic gradient hamiltonian monte carlo. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018. Cited on page 44.

[85] E. Hawkins and R. Sutton. The potential to narrow uncertainty in regional climate predictions. *Bulletin of the American Meteorological Society*, 2009. Cited on page 147.

[86] J. Hensman. Sparse GPs: approximate the posterior, not the model, 2020. HTTPS://WWW.SECONDMIND.AI/LABS/SPARSE-GPS-APPROXIMATE-THE-POSTERIOR-NOT-THE-MODEL/, Last accessed on 5th August 2022. Cited on page 31.

[87] J. Hensman, N. Durrande, and A. Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 2018. Cited on pages 39, 177.

[88] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In A. Nicholson and P. Smyth, editors, *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*. AUAI Press, 2013. Cited on pages 37, 43, 44, 71, 133, 177.

[89] J. Hensman, N. D. Lawrence, and M. Rattray. Hierarchical Bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. *BMC bioinformatics*, 2013. Cited on page 154.

[90] J. Hensman, A. Matthews, M. Filippone, and Z. Ghahramani. MCMC for variationally sparse Gaussian processes. *Advances in Neural Information Processing Systems*, 2015. Cited on pages 44, 57.

[91] J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015. Cited on page 38.

[92] D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont. Robust multi-class Gaussian process classification. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, 2011. Cited on page 121.

[93] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020. Cited on page 108.

[94] T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*. IEEE, 1995. Cited on page 142.

[95] O. Hoegh-Guldberg, D. Jacob, M. Bindi, S. Brown, I. Camilloni, A. Diedhiou, R. Djalante, K. Ebi, F. Engelbrecht, J. Guiot, Y. Hijioka, S. Mehrotra, A. Payne, S. I. Seneviratne, A. Thomas, R. Warren, and G. Zhou. Impacts of 1.5C global warming on natural and human systems, Chapter 3. In 2018. Cited on pages 146, 147.

[96] Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph cut. In *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009. Cited on page 100.

[97] M. Hutchinson, A. Terenin, V. Borovitskiy, S. Takao, Y. Teh, and M. Deisenroth. Vector-valued Gaussian processes on Riemannian manifolds via gauge independent projected kernels. *Advances in Neural Information Processing Systems*, 2021. Cited on page 177.

[98] A. Inness, M. Ades, A. Agustí-Panareda, J. Barré, A. Benedictow, A.-M. Blechschmidt, J. J. Dominguez, R. Engelen, H. Eskes, J. Flemming, et al. The CAMS reanalysis of atmospheric composition. *Atmospheric Chemistry and Physics*, 2019. Cited on page 91.

[99] IPCC. Meeting report of the intergovernmental panel on climate change expert meeting on assessing and combining multi model climate projections. *Intergovernal Panel Climate Change*, 2010. R. Knutti, G. Abramowitz, M. Collins, V. Eyring, P. J. Gleckler, B. Hewitson, L. Mearns, T. Stocker, Q. Dahe, et al., editors. Cited on page 147.

[100] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation*, 1991. Cited on page 142.

[101] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 1906. Cited on page 9.

[102] B. Jiang, C. Ding, and J. Tang. Graph-Laplacian PCA: Closed-form solution and robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013. Cited on page 102.

[103] K. Jordahl, J. V. den Bossche, M. Fleischmann, J. Wasserman, J. McBride, J. Gerard, J. Tratner, M. Perry, A. G. Badaracco, C. Farmer, G. A. Hjelle, A. D. Snow, M. Cochran, S. Gillies, L. Culbertson, M. Bartos, N. Eubank, maxalbert, A. Bilogur, S. Rey, C. Ren, D. Arribas-Bel, L. Wasser, L. J. Wolf, M. Journois, J. Wilson, A. Greenhall, C. Holdgraf, Filipe, and F. Leblanc. Geopandas/geopandas: v0.8.1. In Zenodo, 2020. Cited on page 97.

[104] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Learning in Graphical Models*, 1998. M. Jordan, editor. Cited on pages 5, 154.

[105] M. Kac. *Probability and related topics in physical sciences*, number Volume 1. A. Am. Mathematical Soc, 2. printing edition, 1976. ISBN: 978-0-8218-0047-8. OCLC: 223864829. Cited on page 54.

[106] H. Kanagawa, W. Jitkrittum, L. Mackey, K. Fukumizu, and A. Gretton. A kernel Stein test for comparing latent variable models. *arXiv*, 2019. Cited on page 50.

[107] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. Cited on page 108.

[108] S. Klamt, U.-U. Haus, and F. Theis. Hypergraphs and cellular networks. *PLoS computational biology*, 2009. Cited on page 100.

[109] J. Knoblauch, J. Jewson, and T. Damoulas. An optimization-centric view on Bayes' rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research*, 2022. Cited on page 52.

[110] R. Knutti. The end of model democracy? *Climatic Change*, 2010. Cited on page 147.

[111] R. Knutti and J. Sedláček. Robustness and uncertainties in the new CMIP5 climate model projections. *Nature climate change*, 2013. Cited on page 146.

[112] L. Kong, X. Tang, J. Zhu, Z. Wang, J. Li, H. Wu, Q. Wu, H. Chen, L. Zhu, W. Wang, et al. A 6-year-long (2013–2018) high-resolution air quality reanalysis dataset in china based on the assimilation of surface observations from CNEMC. *Earth System Science Data*, 2021. Cited on page 91.

[113] R. Kopp. A hotter future is certain, climate panel warns. but how hot is up to us, 2021. HTTPS://MATANGITONGA.TO/2021/08/10/HOTTER-CERTAIN-CLIMATE-PANEL-WARNS, Last accessed on 21st Septemeber 2022. Cited on page 165.

[114] A. Korba, A. Salim, M. Arbel, G. Luise, and A. Gretton. A non-asymptotic analysis for Stein variational gradient descent. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. Cited on page 58.

[115] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 1951. Cited on page 5.

[116] V. Lalchand, A. Ravuri, and N. D. Lawrence. Generalised GPLVM with stochastic variational inference. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022. Cited on page 116.

[117] J.-F. Lamarque, F. Dentener, J. McConnell, C.-U. Ro, M. Shaw, R. Vet, D. Bergmann, P. Cameron-Smith, S. Dalsoren, R. Doherty, G. Faluvegi, S. J. Ghan, B. Josse, Y. H. Lee, I. A. MacKenzie, D. Plummer, D. T. Shindell, R. B. Skeie, D. S. Stevenson, S. Strode, G. Zeng, M. Curran, D. Dahl-Jensen, S. Das, D. Fritzsche, and M. Nolan. Multi-model mean nitrogen and sulfur deposition from the atmospheric chemistry and climate model intercomparison project (ACCMIP): evaluation of historical and projected future. *Atmospheric Chemistry and Physics*, 2013. Cited on page 146.

[118] M. Lange-Hegermann. Linearly constrained Gaussian processes with boundary conditions. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021. Cited on page 177.

[119] P. S. Laplace. Mémoire sur la probabilité de causes par les évenements. *Mémoire de l'académie royale des sciences*, 1774. Cited on page 2.

[120] H. C. Law, D. Sejdinovic, E. Cameron, T. Lucas, S. Flaxman, K. Battle, and K. Fukumizu. Variational learning on aggregate outputs with Gaussian processes. *Advances in neural information processing systems*, 2018. Cited on page 79.

[121] N. Lawrence and A. Hyvärinen. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research*, 2005. Cited on pages 102, 108, 116, 128.

[122] M. Lázaro-Gredilla and A. Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2009. Cited on page 39.

[123] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 2010. Cited on page 39.

[124] M. Lázaro-Gredilla and M. K. Titsias. Variational heteroscedastic Gaussian process regression. In *ICML*, 2011. Cited on pages 86, 154, 156.

[125] A. Lederer, J. Umlauft, and S. Hirche. Uniform error bounds for Gaussian process regression with application to safe control. *Advances in Neural Information Processing Systems*, 2019. Cited on page 62.

[126] S. H. Lee, J. M. Magallanes, and M. A. Porter. Time-dependent community structure in legislation cosponsorship networks in the Congress of the Republic of Peru. *Journal of Complex Networks*, 2017. Cited on page 120.

[127] F. Lehner, C. Deser, N. Maher, J. Marotzke, E. M. Fischer, L. Brunner, R. Knutti, and E. Hawkins. Partitioning climate projection uncertainty with multiple large ensembles and CMIP5/6. *Earth System Dynamics*, 2020. Cited on page 146.

[128] J. Lelieveld, K. Klingmüller, A. Pozzer, R. Burnett, A. Haines, and V. Ramanathan. Effects of fossil fuel and total anthropogenic emission removal on public health and climate. *Proceedings of the National Academy of Sciences*, 2019. Cited on page 78.

[129] Y. Li and R. E. Turner. Rényi divergence variational inference. *Advances in neural information processing systems*, 2016. Cited on page 52.

[130] Y. Liang, N. P. Gillett, and A. H. Monahan. Climate model projections of 21st century global warming constrained using the observed warming trend. *Geophysical Research Letters*, 2020. Cited on page 147.

[131] S. Lim, I. Mudway, N. Molden, J. Holland, and B. Barratt. Identifying trends in ultrafine particle infiltration and carbon dioxide ventilation in 92 vehicle models. *Science of The Total Environment*, 2022. Cited on page 94.

[132] F. Lindgren, D. Bolin, and H. Rue. The SPDE approach for Gaussian and non-Gaussian fields: 10 years and still running. *Spatial Statistics*, 2022. Cited on page 106.

[133] F. Lindgren, H. Rue, and J. Lindström. 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)*, 2011. Cited on page 105.

[134] Q. Liu. Stein variational gradient descent as gradient flow. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing*

Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017. Cited on pages 55, 58.

[135] Q. Liu, J. D. Lee, and M. I. Jordan. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. *arXiv:1602.03253 [stat]*, 2016. arXiv: 1602.03253. Cited on pages 51, 52, 159.

[136] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016. Cited on pages 45, 54, 64.

[137] S. V. Lototsky, B. L. Rozovsky, et al. *Stochastic partial differential equations*. Springer, 2017. Cited on page 104.

[138] X. Lu, A. Boukouvalas, and J. Hensman. Additive Gaussian processes revisited. In *International Conference on Machine Learning*. PMLR, 2022. Cited on page 177.

[139] D. J. C. Mackay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992. Cited on page 2.

[140] A. Mallasto and A. Feragen. Learning from uncertain curves: the 2-Wasserstein metric for Gaussian processes. *Advances in Neural Information Processing Systems*, 2017. Cited on pages 133, 162.

[141] J. Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 2020. Cited on page 38.

[142] V. Masarotto, V. M. Panaretos, and Y. Zemel. Procrustes metrics on covariance operators and optimal transportation of Gaussian processes. *Sankhya A*, 2019. Cited on page 162.

[143] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *Advances in neural information processing systems*, 1999. Cited on page 142.

[144] B. Matérn. *Spatial variation : Stochastic models and their application to some problems in forest surveys and other sampling investigations*. PhD thesis, Stockholm University, 1960. Cited on page 21.

[145] G. Matheron. Principles of geostatistics. *Economic geology*, 1963. Cited on pages 95, 130.

[146] J. E. Matheson and R. L. Winkler. Scoring rules for continuous probability distributions. *Management science*, 1976. Cited on page 159.

[147] A. G. d. G. Matthews, M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPFlow: a Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 2017. Cited on pages 46, 74, 132.

[148] D. Meunier, M. Pontil, and C. Ciliberto. Distribution regression with sliced Wasserstein kernels. *arXiv preprint arXiv:2202.03926*, 2022. Cited on page 177.

[149] J. Mockus. *Bayesian Approach to Global Optimization: Theory and Applications*. English. Springer, 2012. ISBN: 978-94-009-0909-0. OCLC: 851374758. Cited on page 44.

[150] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards global optimization*, 1978. Cited on page 130.

[151] G. Monge. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, 1781. Cited on page 149.

[152] C. P. Morice, J. J. Kennedy, N. A. Rayner, J. P. Winn, E. Hogan, R. E. Killick, R. J. H. Dunn, T. J. Osborn, P. D. Jones, and I. R. Simpson. An updated assessment of near-surface temperature change from 1850: the HadCRUT5 data set. *Journal of Geophysical Research: Atmospheres*, 2021. Cited on page 152.

[153] J. M. Murphy, D. M. Sexton, D. N. Barnett, G. S. Jones, M. J. Webb, M. Collins, and D. A. Stainforth. Quantification of modelling uncertainties in a large ensemble of climate change simulations. *Nature*, 2004. Cited on page 146.

[154] I. Murray, R. Adams, and D. MacKay. Elliptical slice sampling. en. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010. Cited on page 57.

[155] S. Murray and H. Kjellström. Mixed likelihood Gaussian process latent variable model. *arXiv preprint arXiv:1811.07627*, 2018. Cited on page 116.

[156] R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. *arXiv:physics/9701026*, 1997. arXiv: physics/9701026. Cited on page 67.

[157] J. Ngiam, Z. Chen, P. W. Koh, and A. Y. Ng. Learning deep energy models. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011. Cited on page 108.

[158] T. V. Nguyen and E. V. Bonilla. Automated variational inference for Gaussian process models. *Advances in Neural Information Processing Systems*, 2014. Cited on page 44.

[159] A. V. Nikitin, S. John, A. Solin, and S. Kaski. Non-separable spatio-temporal graph kernels via SPDEs. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022. Cited on page 95.

[160] NOAA. State of the climate: monthly global climate report for annual 2021. 2021. URL: HTTPS://WWW.NCEI.NOAA.GOV/ACCESS/MONITORING/MONTHLY-REPORT/GLOBAL/202113. (visited on 09/20/2022). Cited on page 143.

[161] C. J. Oates, M. Girolami, and N. Chopin. Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2017. Cited on pages 44, 49, 50.

[162] Objax Developers. Objax, version 1.2.0, 2020. URL: HTTPS://GITHUB.COM/GOOGLE/OBJAX. Cited on page 132.

[163] S. Ohlwein, R. Kappeler, M. Kutlar Joss, N. Künzli, and B. Hoffmann. Health effects of ultrafine particles: a systematic literature review update of epidemiological evidence. *International journal of public health*, 2019. Cited on page 91.

[164] B. Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013. Cited on page 105.

[165] M. Opper and C. Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 2008. Cited on page 44.

[166] M. B. Palacios and M. F. J. Steel. Non-Gaussian Bayesian geostatistical modeling. *Journal of the American Statistical Association*, 2006. Cited on page 56.

[167] T. Palmer, F. Doblas-Reyes, R. Hagedorn, and A. Weisheimer. Probabilistic prediction of climate using multi-model ensembles: from basics to applications. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2005. Cited on page 147.

[168] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *arXiv:1912.02762 [cs, stat]*, 2019. arXiv: 1912.02762. Cited on page 53.

[169] A. Patania, G. Petri, and F. Vaccarino. The shape of collaborations. *EPJ Data Science*, 2017. Cited on page 100.

[170] S. Paul and Y. Chen. Consistent community detection in multi-relational data through restricted multi-layer stochastic blockmodel. *Electronic Journal of Statistics*, 2016. Cited on page 103.

[171] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 1901. Cited on page 26.

[172] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 2011. Cited on page 101.

[173] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014. Cited on page 102.

[174] G. Peyré, M. Cuturi, et al. Computational optimal transport: with applications to data science. *Foundations and Trends® in Machine Learning*, 2019. Cited on page 149.

[175] D. Phan, N. Pradhan, and M. Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019. Cited on page 133.

[176] T. Pinder, M. Amos, D. Booker, R. Duncan, L. Gouldsbrough, and P. Young. LancasterAQ: a mobile air quality dataset measuring ultrafine levels in Lancaster. *Under review*, 2022. Cited on pages iv, 40.

[177] T. Pinder, M. Amos, and P. Young. Identifying latent climate signals using sparse hierarchical Gaussian processes. *NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems*, 2022. Cited on page iv.

[178] T. Pinder and D. Dodd. GPJax: a Gaussian process framework in jax. *Journal of Open Source Software*, 2022. Cited on pages iv, 41.

[179] T. Pinder, C. Nemeth, and D. Leslie. Stein variational Gaussian processes. *arXiv preprint arXiv:2009.12141*, 2020. Cited on pages iv, 40, 43.

[180] T. Pinder, K. Turnbull, C. Nemeth, and D. Leslie. Gaussian processes on hypergraphs. *arXiv preprint arXiv:2106.01982*, 2021. Cited on pages iv, 40, 154.

[181] T. Pinder, K. Turnbull, C. Nemeth, and D. Leslie. Street-level air pollution modelling with graph Gaussian processes. *ICLR: AI for Earth and Space Science*, 2022. Cited on pages iv, 40, 95, 97.

[182] H. Pörtner, D. Roberts, M. Tignor, E. Poloczanska, K. Mintenbeck, A. Alegrìa, M. Craig, S. Langsdorf, S. Löschke, V. Möller, et al. Contribution of working group ii to the sixth assessment report of the intergovernmental panel on climate change, 2022. Cited on page 143.

[183] Y. Pu, Z. Gan, R. Henao, C. Li, S. Han, and L. Carin. VAE learning via Stein variational gradient descent. *Advances in Neural Information Processing Systems*, 2017. Cited on pages 50, 55.

[184] J. Quinonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 2005. Cited on page 28.

[185] A. E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski. Using Bayesian model averaging to calibrate forecast ensembles. *Monthly weather review*, 2005. Cited on page 147.

[186] S. Ramchandran, M. Koskinen, and H. Lähdesmäki. Latent Gaussian process with composite likelihoods and numerical quadrature. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021. Cited on page 116.

[187] D. A. Randall, R. A. Wood, S. Bony, R. Colman, T. Fichefet, J. Fyfe, V. Kattsov, A. Pitman, J. Shukla, J. Srinivasan, R. J. Stouffer, A. Sumi, and K. E. Taylor. Climate models and their evaluation. In *Climate change 2007: The physical science basis. Contribution of Working Group I to the Fourth Assessment Report of the IPCC (FAR)*, pages 589–662. Cambridge University Press, 2007. Cited on page 143.

[188] A. Rapoport. Spread of information through a population with socio-structural bias: i. assumption of transitivity. *The bulletin of mathematical biophysics*, 1953. Cited on page 103.

[189] C. E. Rasmussen and H. Nickisch. Gaussian processes for machine learning (GPML) toolbox. *The Journal of Machine Learning Research*, 2010. Cited on page 133.

[190] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*, number 3. MIT press Cambridge, MA, 2006. Cited on pages 11, 44, 104, 130.

[191] T. Reichler and J. Kim. How well do coupled models simulate today's climate? *Bulletin of the American Meteorological Society*, 2008. Cited on page 147.

[192] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*. PMLR, 2015. Cited on page 108.

[193] K. Riahi, D. P. Van Vuuren, E. Kriegler, J. Edmonds, B. C. O'neill, S. Fujimori, N. Bauer, K. Calvin, R. Dellink, O. Fricko, et al. The shared socioeconomic pathways and their energy, land use, and greenhouse gas emissions implications: an overview. *Global environmental change*, 2017. Cited on pages 144, 145.

[194] D. Rudoy and P. J. Wolfe. Monte Carlo Methods for Multi-Modal Distributions. In *2006 Fortieth Asilomar Conference on Signals, Systems and Computers*, 2006. ISSN: 1058-6393. Cited on page 44.

[195] A. Ruepp, B. Waegele, M. Lechner, B. Brauner, I. Dunger-Kaltenbach, G. Fobo, G. Frishman, C. Montrone, and H.-W. Mewes. Corum: the

comprehensive resource of mammalian protein complexes—2009. *Nucleic acids research*, 2010. Cited on page 107.

[196] R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In D. A. V. Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*. JMLR.org, 2009. Cited on page 108.

[197] H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017. Cited on page 57.

[198] H. Salimbeni, S. Eleftheriadis, and J. Hensman. Natural gradients in practice: non-conjugate variational inference in Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018. Cited on page 38.

[199] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck. Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, 2016. Cited on page 133.

[200] C. Salvi and M. Lemercier. Neural stochastic partial differential equations. *arXiv preprint arXiv:2110.10249*, 2021. Cited on page 106.

[201] B. M. Sanderson, M. Wehner, and R. Knutti. Skill and independence weighting for multi-model assessments. *Geoscientific Model Development*, 2017. Cited on page 147.

[202] S. Särkkä and A. Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019. Cited on page 104.

[203] A. D. Saul, J. Hensman, A. Vehtari, and N. D. Lawrence. Chained Gaussian processes. In *Artificial Intelligence and Statistics*. PMLR, 2016. Cited on pages 154, 156.

[204] L. J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 1971. Cited on page 161.

[205] R. E. Schapire. The strength of weak learnability. *Machine learning*, 1990. Cited on page 142.

[206] G. Schrotter and C. Hürzeler. The digital twin of the city of Zurich for urban planning. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 2020. Cited on page 93.

[207] M. W. Seeger, C. K. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2003. Cited on page 28.

[208] D. K. Sewell and Y. Chen. Latent space models for dynamic networks. *Journal of the American Statistical Association*, 2015. Cited on page 103.

[209] D. K. Sewell and Y. Chen. Latent space models for dynamic networks with weighted edges. *Social Networks*, 2016. Cited on page 103.

[210] B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1985. Cited on pages 29, 84.

[211] A. L. Smith, D. M. Asta, and C. A. Calder. The geometry of continuous latent space models for network data. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 2019. Cited on page 178.

[212] J. D. Smith, S. Hall, G. Coombs, J. Byrne, M. A. Thorne, J. A. Brearley, D. Long, M. Meredith, and M. Fox. Autonomous passage planning for a polar vessel. *arXiv preprint arXiv:2209.02389*, 2022. Cited on page 93.

[213] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003. Cited on pages 82, 113.

[214] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 2005. Cited on pages 29, 154, 177.

[215] E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen. Derivative observations in Gaussian process models of dynamic systems. *Advances in neural information processing systems*, 2002. Cited on page 62.

[216] S. Srivastava, V. Cevher, Q. Dinh, and D. Dunson. WASP: Scalable Bayes via barycenters of subset posteriors. In G. Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015. Cited on page 142.

[217] C. Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proc. Sixth Berkeley Symp. Math. Stat. Prob.* 1972. Cited on pages 46, 49.

[218] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008. Cited on pages 46, 48.

[219] M. Strak, G. Hoek, M. Steenhof, E. Kilinc, K. J. Godri, I. Gosens, I. S. Mudway, R. van Oerle, H. M. Spronk, F. R. Cassee, et al. Components of ambient air pollution affect thrombin generation in healthy humans: the raptes project. *Occupational and environmental medicine*, 2013. Cited on page 78.

[220] M. Strak, N. A. Janssen, K. J. Godri, I. Gosens, I. S. Mudway, F. R. Cassee, E. Lebret, F. J. Kelly, R. M. Harrison, B. Brunekreef, et al. Respiratory health effects of airborne particulate matter: the role of particle size, composition, and oxidative potential—the RAPTES project. *Environmental health perspectives*, 2012. Cited on page 78.

[221] A. Takatsu. Wasserstein geometry of Gaussian measures. *Osaka Journal of Mathematics*, 2011. Cited on page 150.

[222] K. E. Taylor, R. J. Stouffer, and G. A. Meehl. An overview of CMIP5 and the experiment design. *Bulletin of the American Meteorological Society*, 2012. Cited on page 143.

[223] C. Tebaldi and R. Knutti. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2007. Cited on pages 146, 147.

[224] W. Tebbutt, A. Solin, and R. E. Turner. Combining pseudo-point and state space approximations for sum-separable Gaussian processes. In *Uncertainty in Artificial Intelligence*. PMLR, 2021. Cited on page 46.

[225] T. R. Tebbutt W Bruinsma W. Stheno.jl. 2022. URL: https://github.com/JuliaGaussianProcesses/Stheno.jl. Cited on page 133.

[226] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 1999. Cited on page 108.

[227] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*. PMLR, 2009. Cited on pages 31, 35, 43, 44, 154, 156, 157, 169, 177.

[228] M. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010. Cited on page 36.

[229] A. Tosi, S. Hauberg, A. Vellido, and N. D. Lawrence. Metrics for probabilistic geometries. *arXiv preprint arXiv:1411.7432*, 2014. Cited on pages 128, 178.

[230] R. E. Turner and M. Sahani. Two problems with variational expectation maximisation for time-series models. In D. Barber, T. Cemgil, and S. Chiappa, editors, *Bayesian Time series models*, chapter 5, pages 109–130. Cambridge University Press, 2011. Cited on page 169.

[231] UNFCCC. Paris agreement. *United Nations Framework Convention on Climate Change*, 2015. Cited on page 164.

[232] J. Vanhatalo, J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari. GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*, 2013. Cited on page 133.

[233] R. Vautard, N. Kadygrov, C. Iles, F. Boberg, E. Buonomo, K. Bülow, E. Coppola, L. Corre, E. van Meijgaard, R. Nogherotto, et al. Evaluation of the large EURO-CORDEX regional climate model ensemble. *Journal of Geophysical Research: Atmospheres*, 2021. Cited on page 146.

[234] C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2009. Cited on page 149.

[235] G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-variance tradeoff and the randomized gacv. *Advances in Neural Information Processing Systems*, 1998. Cited on page 29.

[236] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact Gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 2019. Cited on page 177.

[237] Y. Wang, Y. Wu, Z. Li, K. Liao, C. Li, and G. Song. Route planning for active travel considering air pollution exposure. *Transportation Research Part D: Transport and Environment*, 2022. Cited on page 93.

[238] M. D. Ward, K. Stovel, and A. Sacks. Network analysis and political science. *Annual Review of Political Science*, 2011. Cited on page 103.

[239] P. Whittle. Stochastic processes in several dimensions. *Bulletin of the International Statistical Institute*, 1963. Cited on page 105.

[240] W. J. Wilkinson, S. Särkkä, and A. Solin. Bayes-Newton methods for approximate Bayesian inference with PSD guarantees. *arXiv preprint arXiv:2111.01721*, 2021. Cited on page 132.

[241] WMO. *Scientific Assessment of Ozone Depletion: 2018, Global Ozone Research and Monitoring Project-Report No. 58*. WMO, 2018. Cited on page 147.

[242] World Health Organization. Nitrogen oxides environmental health criteria 188. *International programme on chemical safety*, 1997. Cited on page 89.

[243] X. Yu and A. Rodriguez. A Bayesian approach to spherical factor analysis for binary data. *arXiv preprint arXiv:2008.05109*, 2020. Cited on page 103.

[244] M. D. Zelinka, T. A. Myers, D. T. McCoy, S. Po-Chedley, P. M. Caldwell, P. Ceppi, S. A. Klein, and K. E. Taylor. Causes of higher climate sensitivity in CMIP6 models. *Geophysical Research Letters*, 2020. Cited on page 166.

[245] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: clustering, classification, and embedding. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2006. Cited on pages 107, 123, 124, 126, 127.

[246] D. X. Zhou. Derivative reproducing properties for kernel methods in learning theory. en. *Journal of Computational and Applied Mathematics*, 2008. Cited on page 62.

[247] H. Zhu, A. Howes, O. van Eer, M. Rischard, Y. Li, D. Sejdinovic, and S. Flaxman. Aggregated Gaussian processes with multiresolution Earth observation covariates. *arXiv preprint arXiv:2105.01460*, 2021. Cited on page 79.