Generalised Variational Inference for Gaussian Processes

James Wu

supervised by

Veit D. Wild & Jeremias Knoblauch

September 2023

MSc Computational Statistics and Machine Learning

This report is submitted as part requirement for the MSc Computational Statistics and Machine Learning at University College London. It is substantially the result of my own work except where explicitly indicated in the text.

Abstract

Proposed by Knoblauch et al. (2022), generalised variational inference (GVI) is a learning framework motivated by an optimisation-centric interpretation of Bayesian inference. Extending GVI to infinite dimensions, Wild et al. (2022) introduces Gaussian Wasserstein inference (GWI) in function spaces. GWI demonstrates a new inference approach for variational Gaussian processes (GPs), circumventing many limitations of previous approaches. Our work introduces various improvements to GWI for GPs, including new kernel parameterisations such as the neural network GP (NNGP) kernels from Novak et al. (2019). We also introduce a new learning framework that we call projected GVI (pGVI) for GPs. pGVI weakens the GVI assumption of a definite regulariser. Instead, we propose regularising between scalar projections of the stochastic processes, an approach we call projected regularisation. We demonstrate that pGVI is a highly flexible and well-performing variational inference framework with significantly cheaper linearly time computational costs compared to the cubic costs of existing approaches. We also present our learning frameworks through a comprehensive software implementation available on GitHub[1].

[1] For the most up-to-date version, see: https://github.com/jswu18/gvi-gaussian-process

Acknowledgements

This work would not have been possible without Jeremias Knoblauch and Veit D. Wild. Thank you for your guidance and endless patience as I navigated this world of theory while wearing the hardhat of an engineer. I look forward to continuing our collaboration into the future. I'd also like to thank my friends and family who have supported me throughout my life and in particular, this past year.

- James

Contents

0 Notation

We will first make some general remarks regarding the notation used throughout this report. We will denote $\mathcal{X}$ as any input space, where $\mathbf{X} \coloneqq \{x_n\}_{n=1}^N$ is a set of N elements from the input space such that $\mathbf{X} \in \mathcal{X}^N$. We also indicate N input and response pairs as $(\mathbf{X}, \mathbf{Y}) \coloneqq \{x_n, y_n\}_{n=1}^N$, where each input x_n has a corresponding y_n in a response space such as $\mathbb{R}$. $\boldsymbol{\mathcal{F}}$, $\boldsymbol{\mathcal{Q}}$, and $\boldsymbol{\Gamma}$ will also denote spaces, but will be defined throughout the text as needed.

We will frequently use mappings of the form $m : \mathcal{X} \to \mathbb{R}$. Given $\mathbf{X}$ and m, we will construct vectors of the form $\mathbf{m} \in \mathbb{R}^N$ such that each element is

$$[\mathbf{m}]_n \coloneqq m(x_n), \tag{0.1}$$

for $n = 1, \ldots, N$. Whenever we need additional clarity, we will denote $\mathbf{m}$ as $\mathbf{m}_{\mathbf{X}}$ or $m(\mathbf{X})$ to indicate its dependence on $\mathbf{X}$ and/or m.

We will also use mappings following $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. With $\mathbf{X}$ and k, we can construct matrices of the form $\mathbf{K} \in \mathbb{R}^{N \times N}$ where

$$[\mathbf{K}]_{n,n'} \coloneqq k(x_n, x_{n'}) \tag{0.2}$$

for $n, n' = 1, \ldots, N$. We denote $\mathbf{K}$ as $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ or $k(\mathbf{X}, \mathbf{X})$ whenever necessary.

A Gaussian distribution will be denoted $\mathcal{N}(\cdot, \cdot)$, where

$$Y \sim \mathcal{N}(\mathbf{m}, \mathbf{K}) \tag{0.3}$$

with $Y \in \mathbb{R}^N$, indicating that Y is a random vector following a Gaussian distribution having $\mathbf{m} \in \mathbb{R}^N$ as the mean and $\mathbf{K} \in \mathbb{R}^{N \times N}_{\succcurlyeq 0}$ as the covariance. The subscript $\succcurlyeq 0$ denotes that $\mathbf{K}$ is positive semi-definite.

For any random element F, we review the following notation conventions:

- sample notation:

$$F \sim \mathcal{N}(\cdot, \cdot) \tag{0.4}$$

  as seen previously, which becomes $F|\mathbf{Y}$ when F is conditioned on the known element $\mathbf{Y}$,

- measure notation:

$$\mathbb{P}^F = \mathcal{N}(\cdot, \cdot), \tag{0.5}$$

  where $\mathbb{P}^F$ is the probability measure of F or $\mathbb{P}(F) = \mathcal{N}(\cdot, \cdot)$, becoming $\mathbb{P}(F|\mathbf{Y})$ when conditioned, and

- probability density notation:

$$p(f) = \mathcal{N}(\cdot, \cdot), \tag{0.6}$$

  when there exists a density p with respect to some other measure, most commonly the Lebesgue measure. When conditioned, we denote $p(f|\mathbf{Y})$.

Sample, measure, and probability density notation will be used interchangeably, whenever appropriate. We will also define new notation such as $P \coloneqq \mathbb{P}(F)$ and $Q \coloneqq \mathbb{Q}(F)$, whenever convenient. We will use

$$\mathbb{D}[Q, P] \tag{0.7}$$

to denote a divergence between the measures Q and P, where a divergence satisfies

$$\mathbb{D}[Q, P] \geq 0 \tag{0.8}$$

and

$$\mathbb{D}[Q, P] = 0 \Leftrightarrow Q = P. \tag{0.9}$$

Other divergences will follow similar notation and will be defined whenever they are used.

Other conventions that we follow include:

- $\mathbb{E}_P[\,\cdot\,]$ to denote expectation with respect to P, and
- $[\,\cdot\,]^T$ to denote the matrix transpose.

1 Introduction

Uncertainty quantification (UQ) is an important research direction within the machine learning community. As an approach, UQ can demonstrate desirable prediction behaviours such as out-of-distribution (OOD) detection. In practice, quantifying uncertainty can provide valuable information during high-risk situations and enable better informed decision-making.

There are a number of modelling approaches for uncertainty quantification. Bayesian approaches are a common choice and generally provide more theoretically intuitive interpretations of uncertainty compared to other methods. Our work focuses on a well-known Bayesian approach called Gaussian processes (GPs), which we review in Section 2. GPs quantify uncertainty in the function space domain but like most Bayesian approaches, they suffer from mis-specifications and computational intractability. The problem of GP intractability has motivated many variational approaches, including the sparse variational GP (svGP) proposed by Titsias (2009). We review the svGP in Section 3, but by operating within restrictive approximation spaces, we also recognise the limitations of this approach. With a review of Knoblauch et al. (2022) in Section 4, we explain how generalised variational inference (GVI) addresses the issues of mis-specification and intractability prevalent in Bayesian inference. GVI re-contextualises Bayesian modelling within an optimisation-centric learning framework. This is followed by a review of Gaussian Wasserstein inference (GWI) from Wild et al. (2022) later in the same section, which extends GVI to function spaces. GWI offers a solution to the restrictive approximation spaces of previous variational GP approaches, such as the svGP.

Our contributions begin in Section 5, where we present a number of improvements to the GWI learning framework. This includes new parameterisations for the prior kernel such as the neural network GP (NNGP) kernel from Novak et al. (2019). We will also introduce new variational kernels as well as faster numerical approximations of the GWI objective. This is followed by Section 6, which introduces a new computationally cheap framework we call projected GVI (pGVI) for GPs. pGVI offers more flexibility for defining learning objectives for variational GPs with a new form of regularisation we call projected regularisation. This computes divergences between scalar projections of GPs, providing significantly cheaper linear time computational costs compared to the cubic costs of existing approaches. We also present an experimentation framework for our contributions, summarised in Section 7. This includes the development of an extensive software implementation designed for flexibility and scaleability, introducing appropriate abstraction architectures, and following best practices common in the software engineering community. This section will also present our experimentation setup, with examples to show it in action. We conclude with Section 8 proposing future research directions and final thoughts in Section 9.

2 Gaussian Processes

Gaussian processes (GPs) are powerful function approximators that can be used for both regression and classification tasks. The following sections will review GPs following Rasmussen (2003), Matthews (2017), and Wild et al. (2022).

2.1 The Gaussian Process

A GP is a stochastic process such that for any N points $\mathbf{X} \coloneqq \{x_n\}_{n=1}^{N}$ where $x_n \in \mathcal{X}$, the corresponding random response vector $Y \in \mathbb{R}^N$ has the Gaussian distribution

$$Y \sim \mathcal{N}\left(\mathbf{m}, \mathbf{K}\right), \tag{2.1}$$

where $\mathbf{m} \in \mathbb{R}^N$ and $\mathbf{K} \in \mathbb{R}_{\succcurlyeq 0}^{N \times N}$. The mean vector $\mathbf{m}$ is constructed through the selection of a mean function mapping $m : \mathcal{X} \to \mathbb{R}$ such that

$$\mathbf{m} \coloneqq \left[m(x_1) \cdots m(x_N)\right]^T, \tag{2.2}$$

while constructing $\mathbf{K}$ involves choosing a kernel function mapping $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that each element of the matrix is the evaluation

$$\left[\mathbf{K}\right]_{n,n'} \coloneqq k(x_n, x_{n'}), \tag{2.3}$$

for $n, n' = 1, \ldots, N$. $\mathbf{K}$ is also known as the Gram matrix. Appendix A.1 shows that choosing kernel functions defined as inner products of a feature space mapping will ensure that $\mathbf{K}$ is a valid positive semi-definite covariance matrix, as required in (2.1).

The distribution in (2.1) is a finite-dimensional instance of the GP random function mapping

$$F \sim \mathcal{GP}(m, k), \tag{2.4}$$

where $F \coloneqq \{F(x) : x \in \mathcal{X}\}$ such that a sample path from F has the mapping $f : \mathcal{X} \to \mathbb{R}$. In other words, choosing a mean and kernel to construct (2.4) ensures that *any* finite set of inputs will have a consistent joint Gaussian distribution adhering to (2.1) such that

$$Y \coloneqq \left[F(x_1), \ldots, F(x_N)\right]^T. \tag{2.5}$$

GPs are a powerful modelling approach. With minimal restrictions for choosing the mean and kernel function, there are endless possibilities for constructing expressive GP model spaces. This control and visibility into the model's behaviour is a strong advantage for GPs compared to other approaches. Novak et al. (2019) explains that the GP is also an important construct for understanding the theoretical properties of many neural network architectures at initialisation. Showing that the infinite-width limit of many such architectures can be expressed as a GP has provided a theoretical framework for analysing neural networks, which are typically viewed as black box approaches. This further motivates the potential and importance of GPs.

2.2 Gaussian Process Regression

Consider the regression task where we have N observation pairs $(\mathbf{X}, \mathbf{Y}) \coloneqq \{(x_n, y_n)\}_{n=1}^N$ with inputs $x_n \in \mathcal{X}$ and responses $y_n \in \mathbb{R}$. GP regression models the data generating process as

$$y_n \sim F(x_n) + \epsilon_n, \tag{2.6}$$

where the GP random function mapping F accounts for the epistemic (model) uncertainty and the random scalar ϵ_n accounts for the aleatoric (measurement) uncertainty. In this formulation, we assume that the aleatoric uncertainty is homoscedastic of the form

$$\epsilon_n \sim \mathcal{N}\left(0, \sigma^2\right). \tag{2.7}$$

In GP regression, the Bayesian posterior for a test point $x \in \mathcal{X}$ when conditioned on the training data $\mathbf{X}$ and $\mathbf{Y}$, acts as a 'prediction' for the epistemic uncertainty of the test data responses. With all terms being Gaussian, this posterior has a closed-form conditional Gaussian expression. Having also chosen the aleatoric data uncertainty to be modelled as Gaussian in (2.7), GP regression models the test data response $y \in \mathbb{R}$ as

$$y | \mathbf{X}, \mathbf{Y}, \sigma^2 \sim \mathcal{N}\left(\bar{m}(x), \bar{k}(x, x)\right), \tag{2.8}$$

with

$$\bar{m}(x) = m(x) + \mathbf{K}_{x,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}\right)^{-1}(\mathbf{Y} - \mathbf{m}_{\mathbf{X}}) \tag{2.9}$$

and

$$\bar{k}(x, x) = k(x, x) - \mathbf{K}_{x,\mathbf{X}}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} + \sigma^2 \mathbf{I}\right)^{-1}\mathbf{K}_{\mathbf{X},x}, \tag{2.10}$$

where $\mathbf{m}_{\mathbf{X}}$ is more verbose notation for (2.2) and $\mathbf{K}_{\mathbf{X},\mathbf{X}}$ is verbose for (2.3). $\mathbf{K}_{x,\mathbf{X}} \in \mathbb{R}^{1\times N}$ and $\mathbf{K}_{\mathbf{X},x} \in \mathbb{R}^{N\times 1}$ are row and column matrices respectively, constructed with $\mathbf{X}$ and x following (2.3). $\mathbf{I} \in \mathbb{R}^{N\times N}$ is the identity matrix.

2.3 Gaussian Process Classification

Consider the classification task where we have N observation pairs $(\mathbf{X}, \mathbf{Y}) \coloneqq \{(x_n, y_n)\}_{n=1}^N$ with inputs $x_n \in \mathcal{X}$ and label responses $y_n \in \{1, \dots, J\}$. In other words, we wish to map each input x_n to one of J labels. Following the approach from Matthews (2017), GPs can be used for classification through the model

$$Y \sim \mathcal{C}\Big(s\left(F_1(x), \dots, F_J(x)\right)\Big), \tag{2.11}$$

where we construct $F_j \sim \mathcal{GP}\left(m_j, k_j\right)$ for each label $j = 1, \dots J$ such that $F_1, \dots, F_J$ are stochastically independent, and $s : \mathbb{R}^J \to \Delta(J)$ is a mapping to a J dimensional probability simplex, which parameterises the categorical distribution $\mathcal{C}$. This means that the probability of the j^{th} label is given as

$$\mathbb{P}(Y = j) = s_j(F_1(x), \ldots, F_J(x)), \tag{2.12}$$

the j^{th} element of the probability simplex from s.

The Robust Max Function Matthews (2017) provides a few different choices for s in (2.11). We follow Wild et al. (2022), using the robust max function to define the j^{th} element of the probability simplex with

$$s_j\left(f_1, \ldots, f_J\right) = \begin{cases} 1-\delta, & \text{if } j = \arg\max_{j=1\ldots J}\left(f_j\right), \\ \frac{\delta}{J-1}, & \text{otherwise,} \end{cases} \tag{2.13}$$

where $\delta \in [0,1]$. Typically, δ is chosen as a very small value (i.e. $1e^{-2}$). Constructing the $\Delta(J)$ vector with (2.13), we have the probability value of $1-\delta$ for the label of maximum value and $\frac{\delta}{J-1}$ otherwise. Wild et al. (2022) explains that this formulation provides robustness to outliers, as it only considers the ranking of the GP models for each label.

A benefit of the robust max function is that the expected log-likelihood under the categorical distribution in (2.11) becomes analytically tractable. Wild et al. (2022) shows that with N input and response pairs, the expected log-likelihood is

$$\mathbb{E}\left[\log p\left(y|x\right)\right] \approx \sum_{n=1}^{N} \log(1-\epsilon)S(x_n, y_n) + \log\left(\frac{\epsilon}{J-1}\right)\left(1 - S(x_n, y_n)\right), \tag{2.14}$$

with

$$S(x,j) \coloneqq \frac{1}{\sqrt{\pi}} \sum_{i=1}^{I} w_i \left(\prod_{j'=1, j' \neq j}^{J} \phi\left(\frac{\xi_i \sqrt{(2k_{j'}(x,x)} + m_j(x) - m_{j'}(x)}{\sqrt{k_{j'}(x,x)}} \right) \right) \tag{2.15}$$

and $\{w_i, \xi_i\}_{i=1}^{I}$ being the weights and roots of the Hermite polynomial of order $I \in \mathbb{N}$. ϕ is the standard normal cumulative distribution function.

3 Standard Variational Inference for Gaussian Processes

Although they are *analytically* tractable, a major drawback of GP models has been their inability to *computationally* scale with respect to N, the number of training points. Both classification and regression predictive posteriors rely on evaluating the inversion of an $\mathbb{R}^{N\times N}$ matrix in (2.9) and (2.10). This operation has time complexity $\mathcal{O}(N^3)$ and space complexity $\mathcal{O}(N^2)$, both of which quickly become problematic when scaling to larger-sized training sets. This problem has been a serious limitation of GPs and has restricted their use to problem domains having smaller-sized data sets.

This section will review standard variational inference (VI) for GPs and in particular, the sparse variational GP (svGP) from Titsias (2009), to obtain computationally cheaper approximations of the true predictive posterior. We will also discuss the challenges of learning within this framework.

3.1 Standard Variational Inference in Finite Dimensions

GPs are objects constructed in an infinite dimensional setting, however this section will first review standard VI in a finite dimensional setting. A Bayesian modelling approach begins by assuming that the data generating process for an observation is conditionally dependent on M unobserved latent random variables $\mathbf{Z} \in \mathcal{X}^M$ through the observation likelihood $p(Y = y|\mathbf{Z})$ and the prior $p(\mathbf{Z})$. These construct a belief update for the generating process called the Bayesian posterior given as

$$p(\mathbf{Z}|Y = y) \propto p(Y = y|\mathbf{Z})p(\mathbf{Z}), \tag{3.1}$$

which is often computationally and/or analytically intractable. This has motivated the need for variational methods like VI to approximate (3.1). VI is based on the observation that the Kullback-Leibler (KL) divergence between an arbitrary probability measure $Q := \mathbb{Q}(Z)$ and the true Bayesian posterior in (3.1), which we denote $\mathbb{P}(Z|Y)$, can be rewritten as

$$\mathbb{KL}\left[Q, \mathbb{P}(\mathbf{Z}|Y)\right] = L(Q) - \log p(y), \tag{3.2}$$

where $p(y) = \int p(y|z)p(z)dz$ is the marginal log-likelihood and

$$L(Q) := -\mathbb{E}_Q\left[\log p(y|\mathbf{Z})\right] + \mathbb{KL}\left[Q, P\right], \tag{3.3}$$

such that $P := \mathbb{P}(Z)$ is the prior and $\mathbb{KL}\left[\,\cdot\,,\,\cdot\,\right]$ denotes the KL divergence. It follows that

$$\operatorname*{arg\,min}_{Q\in\boldsymbol{Q}} \mathbb{KL}\left[Q, \mathbb{P}(\mathbf{Z}|Y)\right] = \operatorname*{arg\,min}_{Q\in\boldsymbol{Q}} L(Q), \tag{3.4}$$

where $\boldsymbol{Q}$ is a set of candidate probability measures. In other words, approximating the posterior by minimising (3.2) is equivalent to minimising (3.3). Typically, $\boldsymbol{Q}$ is constructed with respect to a parameter set $\boldsymbol{\Gamma}$ such that

$$\boldsymbol{Q} := \left\{Q_\gamma : \gamma \in \boldsymbol{\Gamma}\right\}, \tag{3.5}$$

where $\mathbf{\Gamma}$ is a Euclidean parameter space. Therefore solving

$$\gamma^* \in \underset{\gamma \in \mathbf{\Gamma}}{\arg\min}\, L(Q_\gamma) \tag{3.6}$$

obtains the VI approximation of the Bayesian posterior minimising (3.2), denoted as Q_{γ^*}. Standard VI depends on three important assumptions to ensure a reasonable approximation:

1. the parameterised set of measures $\boldsymbol{Q}$ is large enough to contain a reasonable approximation of the true Bayesian posterior,
2. the parameterisation of Q ensures that $L(Q)$ is tractable or easy to approximate, and
3. there exists an optimisation procedure that can find a reasonable minimiser γ^*.

These three assumptions are in tension with each other. For example, a larger approximation space $\boldsymbol{Q}$ can cause $L(Q)$ to be intractable or create a more difficult optimisation setup. However, VI can be quite successful if employed by well-informed practitioners of the method.

3.2 Sparse[1] Variational Gaussian Processes

To overcome the computationally intractable GP we review Titsias (2009), which proposes a variational approximation for the Bayesian predictive posterior in (2.8) as

$$\mathbb{Q}(F) \coloneqq \mathcal{GP}(m_Q, r), \tag{3.7}$$

where the mean function is

$$m_Q(x) = m_P(x) + \mathbf{K}_{x,\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1} \boldsymbol{\mu}, \tag{3.8}$$

parameterised by $\boldsymbol{\mu} \in \mathbb{R}^M$, and the kernel function is

$$r(x,x) = k(x,x) - \mathbf{K}_{x,\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1} \mathbf{K}_{\mathbf{Z},x} + \mathbf{K}_{x,\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1} \mathbf{\Sigma} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1} \mathbf{K}_{\mathbf{Z},x} \tag{3.9}$$

with m_P and k being the mean and kernel functions of the target GP that we want to approximate, and parameterised by $\mathbf{\Sigma} \in \mathbb{R}^{M \times M}_{\succcurlyeq 0}$. $\mathbf{Z} \in \mathcal{X}^M$ is M inducing points, typically chosen as some subset of $\mathbf{X}$. This defines the parameter space

$$\mathbf{\Gamma} = \left\{ \boldsymbol{\mu} \in \mathbb{R}^M, \mathbf{\Sigma} \in \mathbb{R}^{M \times M}_{\succcurlyeq 0}, \mathbf{Z} \in \mathcal{X}^M \right\}. \tag{3.10}$$

Following (3.3), the variational loss for a candidate $Q \coloneqq \mathbb{Q}(F)$ becomes

$$L(Q) = \frac{1}{N} \sum_{n-1}^{N} \mathbb{E}_Q \left[-\log p(y_n | F(x_n) \right] + \mathbb{KL} \left[\mathbb{Q}^F, \mathbb{P}^F \right], \tag{3.11}$$

where the expectation in (3.11) is tractable, since $F(x_n)$ is Gaussian under Q. However it is unclear if the KL divergence between two GPs in the second term is even well-defined and from a practical viewpoint, computable. Matthews et al. (2016) points out that the choice of m_Q and r by Titsias

[1] Also known as stochastic or scaleable variational Gaussian processes (svGPs)

(2009) ensures that

$$\mathbb{KL}\left[\mathcal{GP}\left(m_Q, r\right), \mathcal{GP}\left(m_P, k\right)\right] = \mathbb{KL}\left[\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right), \mathcal{N}\left(\mathbf{m}_{\mathbf{Z}}, \mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)\right], \tag{3.12}$$

reducing the KL divergence between two stochastic processes to

$$\mathbb{KL}\left[\mathbb{Q}^F, \mathbb{P}^F\right] = \frac{1}{2}\left(\operatorname{tr}\left(\left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1}\boldsymbol{\Sigma}\right) - M + \left(\mathbf{m}_{\mathbf{Z}} - \boldsymbol{\mu}\right)^T \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\left(\mathbf{m}_{\mathbf{Z}} - \boldsymbol{\mu}\right) + \log\left(\frac{\det \mathbf{K}_{\mathbf{Z},\mathbf{Z}}}{\det \boldsymbol{\Sigma}}\right)\right), \tag{3.13}$$

the KL divergence between two finite dimensional Gaussian distributions on $\mathbb{R}^M$. Titsias (2009) shows that for a given set of inducing points $\mathbf{Z}$, the optimal choices $\boldsymbol{\mu}^*$ and $\boldsymbol{\Sigma}^*$ to minimise (3.11) have the closed forms

$$\boldsymbol{\mu}^* = \sigma^{-2}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\boldsymbol{\Psi}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{X}}\left(\mathbf{Y} - \mathbf{m}_{\mathbf{X}}\right) \tag{3.14}$$

and

$$\boldsymbol{\Sigma}^* = \mathbf{K}_{\mathbf{Z},\mathbf{Z}}\boldsymbol{\Psi}^{-1}\mathbf{K}_{\mathbf{Z},\mathbf{Z}}, \tag{3.15}$$

where

$$\boldsymbol{\Psi} \coloneqq \mathbf{K}_{\mathbf{Z},\mathbf{Z}} + \sigma^{-2}\mathbf{K}_{\mathbf{Z},\mathbf{X}}\mathbf{K}_{\mathbf{X},\mathbf{Z}}, \tag{3.16}$$

conveniently eliminating the need for gradient-based optimisations such that $\gamma = (\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*, \mathbf{Z})$.

The svGP ensures matrix inversions of $\mathbb{R}^{M\times M}$ matrices with $\mathcal{O}\left(M^3\right)$ time complexity , while the operation $\mathbf{K}_{\mathbf{Z},\mathbf{X}}\mathbf{K}_{\mathbf{X},\mathbf{Z}}$ in (3.16) is $\mathcal{O}\left(NM^2\right)$. Thus, the overall time complexity of this approach is $\mathcal{O}\left(NM^2\right)$ with space complexity $\mathcal{O}\left(NM\right)$. This significantly improves the scaleability of GP approaches from the standard GP in Section 2. M is typically chosen as $\mathcal{O}(N^{1/2})$ such that the svGP has $\mathcal{O}(N^2)$ and $\mathcal{O}(N^{3/2})$ time and space complexity respectively.

This svGP formulation provides a solution to the scaling issues of the GP, but illustrates how the three assumptions of standard VI discussed in Section 3.1 can break down. In particular, the variational set $\mathbf{Q}$ defined by the parameter set $\boldsymbol{\Gamma}$ may not be expressive enough to contain a reasonable approximation of the true Bayesian posterior. For example, the mean function of the true posterior with a zero mean GP can be expressed as the linear combination

$$\hat{m}(x) = \sum_{n=1}^{N} \alpha_n k(x, x_n) \in \operatorname{span}\left(\{k(\cdot, x_1), \ldots, k(\cdot, x_N)\}\right), \tag{3.17}$$

with $\left[\alpha_1 \cdots \alpha_N\right]^T = \mathbf{K}_{\mathbf{X},\mathbf{X}}^{-1}\mathbf{Y}$. On the other hand, the corresponding variational mean m_Q in (3.8) is only a linear combination within the inducing point space such that

$$m_Q(x) = \sum_{m=1}^{M} \beta_m k(x, z_m) \in \operatorname{span}\left(\{k(\cdot, z_1), \ldots, k(\cdot, z_M))\}\right), \tag{3.18}$$

with $\left[\beta_1 \cdots \beta_M\right]^T = \mathbf{K}_{\mathbf{Z},\mathbf{Z}}^{-1}\boldsymbol{\mu}$. Choosing M as $\mathcal{O}(N^{1/2})$, it is not unlikely that the inducing point space will be too small to contain a reasonable approximation of $\hat{m}$. Wild et al. (2021) also makes a similar argument for the posterior kernel. Burt et al. (2020a) explains that the KL divergence between two GPs is not generally tractable or even finite. Thus within the Bayesian context of functional VI, we are forced to restrict the variational set $\boldsymbol{Q}$, to obtain a tractable loss L.

3.3 Inducing Point Selection

Within the context of VI, we wish to find an optimal set of inducing points $\mathbf{Z}$ such that

$$\operatorname{span}\left(\{k(\cdot, x_1), \ldots, k(\cdot, x_N)\}\right) \approx \operatorname{span}\left(\{k(\cdot, z_1), \ldots, k(\cdot, z_M))\}\right), \tag{3.19}$$

where the inducing points can approximate the majority of the input space. Several existing approaches to inducing point selection include Smola (2000), Hensman et al. (2015), Li et al. (2016), and Alaoui and Mahoney (2015). Algorithm 1 reviews an approach from Burt et al. (2020b), which proposes an iterative procedure that greedily chooses the next inducing point based on the highest marginal variance in the prior, when conditioned on the currently selected set of inducing points.

Algorithm 1 Greedy Variance Inducing Point Selection

```
  $m \leftarrow 1$
  $z_1 \leftarrow \arg\max_{x \in \mathbf{X}} k(x, x)$
  $\mathbf{z} \leftarrow \{z_1\}$                                      ▷ initialise selection set
  while $m < M$ do
      $z_{m+1} \leftarrow \arg\max_{x \in \mathbf{X}} \left(\operatorname{diag}\left(\mathbf{K}_{\mathbf{X},\mathbf{X}} - \mathbf{K}_{\mathbf{X},z_m} k\left(z_m, z_m\right)^{-1} \mathbf{K}_{z_m,\mathbf{X}}\right)\right)$
      $\mathbf{z} \leftarrow \mathbf{z} \cup \{z_{m+1}\}$                  ▷ add to the current selection set
      $m \leftarrow m + 1$
  end while
  return $\mathbf{z}$
```

Figure 3.1 compares random selection to Algorithm 1 with MNIST data, selecting 10 inducing points from 5000 for each digit. We see that the greedy variance approach is much more effective at selecting a diverse set of images. Moreover, the images in Figure 3.1 were selected in order from left to right. The first two images selected by greedy variance are generally two very different variations of the digit. For example, the first image for '1' is a single line slanted to the right while the second image involves more strokes and slants left, showing how the algorithm greedily minimises the variance across the candidate images.

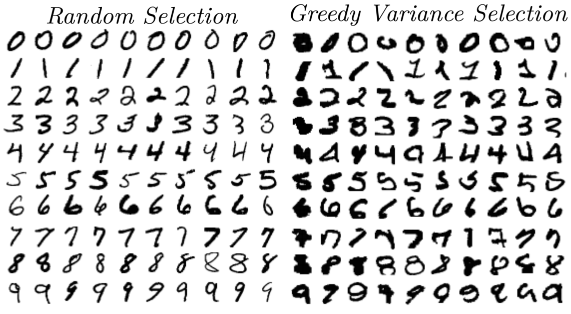


Figure 3.1: Random Selection versus Algorithm 1

Algorithm 2 Prior Kernel Learning and Inducing Points Selection

```
Require: $\theta_0$                                        ▷ initial hyper-parameters of $k$
  $i \leftarrow 0$
  $\mathbf{Z}_0 \leftarrow$ GreedyVarianceSelection($\mathbf{X}$, $\theta_0$)     ▷ via Algorithm 1 with $k$
  $\mathbf{U}_0 \leftarrow \mathbf{Y}[\mathbf{X}.\text{index}(\mathbf{Z}_0)]$        ▷ corresponding inducing point responses
  while $\mathbf{Z}_i \neq \mathbf{Z}_{i-1}$ do
      $\theta_{i+1} \leftarrow$ OptimiseNLL($\theta_0$, $\mathbf{Z}_i$, $\mathbf{U}_i$)   ▷ via type-II maximum likelihood on inducing points
      $\mathbf{Z}_{i+1} \leftarrow$ GreedyVarianceSelection($\mathbf{X}$, $\theta_{i+1}$)
      $\mathbf{U}_{i+1} \leftarrow \mathbf{Y}[\mathbf{X}.\text{index}(\mathbf{Z}_{i+1})]$
      $i \leftarrow i + 1$
  end while
  return ($\theta_i$, $\mathbf{Z}_i$)
```

The inducing point selection method in Algorithm 1 requires a pre-selected kernel. However in practice, kernel selection generally involves negative log-likelihood (NLL) optimisation on training data. In the case of learning the prior GP, this would involve learning the kernel hyper-parameters on the selected inducing points. This presents a chicken and egg problem. Burt et al. (2020b) proposes an EM-like approach, iteratively learning the kernel hyper-parameters and selecting the inducing points with Algorithm 1 until the convergence of an evidence lower bound. Our implementation takes a more naive approach, iterating between point selection and NLL minimisation until the inducing points do not change. We summarise this approach in Algorithm 2.

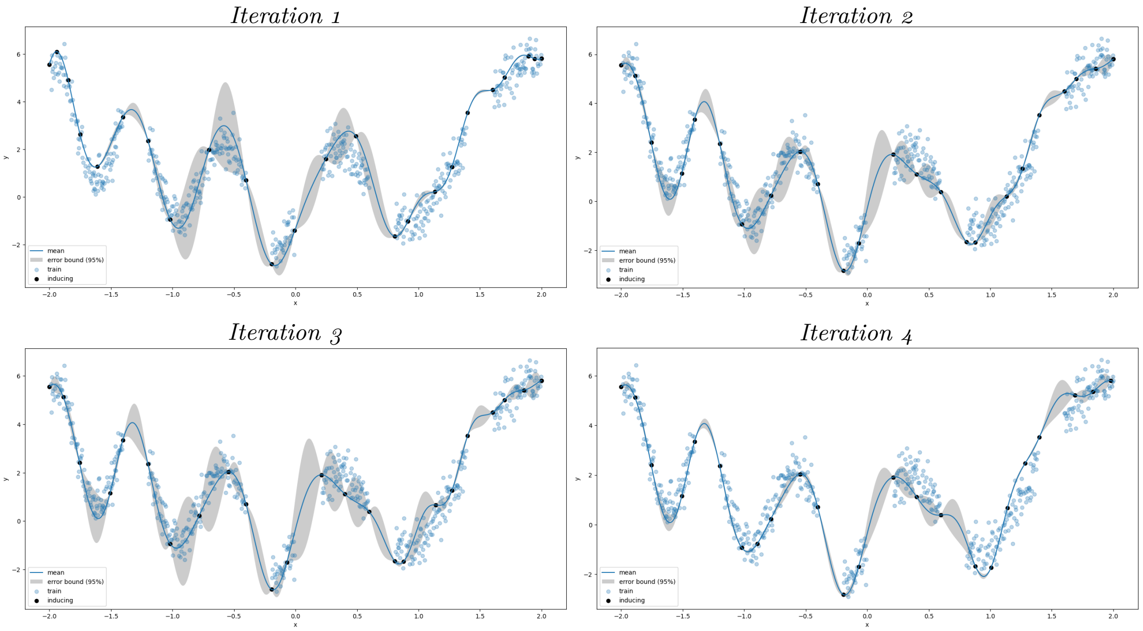


Figure 3.2: Inducing Point Selection for Four Iterations of Algorithm 2 *(black inducing points and blue training/candidate points)*

Algorithm 2 incorporates response data into the inducing point selection process, which did not exist in Algorithm 1. Learning the kernel with the response data can give more appropriate covariances when comparing candidate points during the inducing points selection of Algorithm 1. Figure 3.2 visualises the inducing points selected in the first four iterations of Algorithm 2 for a regression problem. This is overlaid with the Bayesian posterior of a GP parameterised with the learned kernel and zero mean, when conditioned on those selected inducing points. This visualises the selection procedure as it attempts to minimise uncertainty over training points, ensuring maximum coverage over the input space. By learning the kernel in-between iterations, we see that Algorithm 2 can appropriately adjust the variances on the training data, which would be held fixed during Algorithm 1.

In the loop of Algorithm 1, each element of the diagonal evaluation has time complexity $\mathcal{O}(M)$ from matrix multiplication, so computing for all N candidate points along the diagonal is $\mathcal{O}(NM)$. Selecting M inducing points, Algorithm 1 has $\mathcal{O}(NM^2)$ time complexity . Choosing M as $\mathcal{O}(N^{1/2})$, we have $\mathcal{O}(N^2)$ time complexity and $\mathcal{O}(N)$ space complexity. Assuming Algorithm 2 loops for k iterations, we have $\mathcal{O}(kN^2)$ time complexity and the same space complexity as Algorithm 1.

4 Generalising Variational Inference for Gaussian Processes

The standard VI objective in (3.11) is the root cause of the restrictive variational set for the svGP. This is because learning in the standard VI framework necessitates a KL divergence, which is often ill-defined for GPs.

This section reviews generalised variational inference (GVI) in finite dimensions from Knoblauch et al. (2022), a framework that does not rely on the KL divergence. GVI is a fundamental construct for new realisations of variational inference frameworks. We also review Wild et al. (2022), which extends GVI to an infinite dimensional setting and proposes Gaussian Wasserstein inference (GWI) in function spaces. GWI constructs a learning framework that avoids the KL divergence for GPs, providing richer approximation spaces for variational learning.

4.1 Generalised Variational Inference in Finite Dimensions

Extending the Bayesian posterior from (3.1) to N observations we have

$$q_B(\mathbf{Z}) \coloneqq p(\mathbf{Z}|Y=y) \propto p(\mathbf{Z}) \prod_{n=1}^{N} p(Y=y_n|\mathbf{Z}), \tag{4.1}$$

where $p(\mathbf{Z})$ is the prior for the latent variables and $p(Y=y|\mathbf{Z})$ is the likelihood of an observation. The Bayesian posterior $q_B(\mathbf{Z})$ is traditionally viewed as an updated belief for the latent variables by incorporating the likelihood evaluation of new observations. This interpretation is commonly cited in the context of statistical modelling, where practitioners are focused on obtaining the model specification that characterises an underlying data generating process. The validity of this belief update relies on three assumptions:

1. a well-specified prior,
2. a well-specified likelihood, and
3. an analytically and/or computationally tractable posterior, or the existence of a tractable and reasonable VI approximation,

which are often violated in settings of larger-scaled models like Bayesian Neural Networks (BNNs).

In BNNs, Gaussian priors are generally chosen for computational convenience and are most likely mis-specified in the Bayesian context. The likelihood evaluation of BNNs are also most definitely mis-specified, as it's unlikely that evaluating such an over-parameterised model would provide any meaningful insights into the data likelihood. Finally, the posterior of BNNs are often approximated either through samplers or variational approximations. To achieve sampling convergence, larger-scaled models may require much more computational resources and time than is practically available. Moreover a reasonable variational approximation is also not always guaranteed, as discussed in Section 3.1.

Knoblauch et al. (2022) introduces a new interpretation of the Bayesian posterior, showing that it solves the optimisation problem

$$q_B(Z) = \underset{Q \in \boldsymbol{Q}}{\arg\min} \left\{ \mathbb{E}_Q \left[\sum_{n=1}^{N} \ell\left(\mathbf{Z}, y_n\right) \right] + \mathbb{D}\left[Q, P\right] \right\} \tag{4.2}$$

when choosing $\boldsymbol{Q}$ as the space of all probability measures on $\mathbf{Z}$, the negative log-likelihood $\ell(\mathbf{Z}, y) = -\log p\,(y|\mathbf{Z})$, and the KL divergence $\mathbb{D}\,[Q, P] = \mathbb{KL}\,[Q, P]$ with P as the prior. More generally, ℓ is called the loss function and $\mathbb{D}$ is the divergence. With this interpretation, the Bayesian posterior is the solution of a regularised empirical risk minimisation problem.

Framed through optimisation, the Bayesian posterior will always be a valid solution of (4.2), regardless of the three assumptions required to ensure a valid Bayesian belief update. This is more in-tune with practitioners of larger-scaled models who focus on obtaining superior predictive performance rather than correct model specification. Knoblauch et al. (2022) also shows that by generalising the Bayesian posterior within the learning framework of (4.2), any choice of prior P, valid divergence $\mathbb{D}$, loss ℓ, and approximation set $\boldsymbol{Q}$ will result in a generalised posterior that maintains interpretations as a belief update. This more flexible inference approach will motivate the replacement of the KL divergence between GPs that was problematic in standard VI.

4.2 Generalised Variational Inference on Function Spaces

This section reviews Wild et al. (2022), which extends GVI to infinite dimensional settings to quantify uncertainties on function spaces and proposes a new framework for variational learning of GPs. Wild et al. (2022) shows that GVI in function spaces involves solving

$$Q^* = \underset{Q \in \boldsymbol{Q}}{\arg\min} \left\{ \mathbb{E}_Q \left[\sum_{n=1}^{N} \ell(F, y_n) \right] + \mathbb{D}\left[Q, \mathbb{P}(F)\right] \right\}, \tag{4.3}$$

where $Q \in \boldsymbol{Q}$ is a variational family of probability measures on a function space $\boldsymbol{\mathcal{F}}$ and $\mathbb{P}(F)$ is a prior on $\boldsymbol{\mathcal{F}}$. Wild et al. (2022) explains that the Kolmogorov Extension Theorem guarantees that for every GP, there exists a corresponding Gaussian measure (GM). This GM is usually over the trivial function space $\boldsymbol{\mathcal{F}} = \{f : \mathcal{X} \to \mathbb{R}\}$. However, this space is highly prone to support mis-match and ensuring a tractable KL on $\boldsymbol{\mathcal{F}}$ requires heavy restrictions on the variational family $\boldsymbol{Q}$, as discussed in Section 3.2. Instead, Wild et al. (2022) identifies the existence of GPs with corresponding GMs on the Hilbert space of square integrable functions $\mathcal{L}^2\left(\mathcal{X}, \rho, \mathbb{R}\right) = \left\{ f \in \boldsymbol{\mathcal{F}} : \left[\int_{\mathcal{X}} f(x)^2 d\rho(x)\right]^{1/2} < \infty \right\}$ when the mean satisfies

$$m \in \mathcal{L}^2\left(\mathcal{X}, \rho, \mathbb{R}\right) \tag{4.4}$$

and the kernel satisfies

$$\int k(x, x) d\rho(x) < \infty, \tag{4.5}$$

also known as the trace-class kernel condition. We will denote $\mathcal{L}^2\left(\mathcal{X}, \rho, \mathbb{R}\right)$ as $\mathcal{L}^2$ for convenience. These conditions guarantee that sample functions from the GP will have square-integrable paths such that there exists $\mathbb{P} \coloneqq \mathcal{N}(m, C)$, a GM on $\mathcal{L}^2$ with the same mean m and a covariance operator

$$C(f) \coloneqq \int_{\mathcal{X}} k(\cdot, x) f(x) d\rho(x), \tag{4.6}$$

for any function $f \in \mathcal{L}^2$. With the GVI objective in (4.3), Wild et al. (2022) proposes Gaussian Wasserstein inference (GWI), which replaces the KL divergence from standard VI with the squared

2-Wasserstein distance between GMs on Hilbert spaces

$$\mathbb{W}\left[Q,P\right]^2 = \|m_P - m_Q\|_2^2 + \operatorname{tr}\left(C_P\right) + \operatorname{tr}\left(C_Q\right) - 2\operatorname{tr}\left[\left(\left(C_P\right)^{1/2} C_Q \left(C_P\right)^{1/2}\right)^{1/2}\right], \tag{4.7}$$

which is always well-defined and does not have the support mis-match problem of the KL divergence. The subscripts denote the respective GM of each term, tr is the trace of an operator, and $(C)^{1/2}$ denotes the square root of the positive self-adjoint operator C. Most mean functions satisfy (4.4) and most kernels satisfy (4.5), ensuring the existence of the corresponding Gaussian measures P and Q on $\mathcal{L}^2$. This provides significantly more freedom for the variational set $\boldsymbol{Q}$ than the svGP approach from Titsias (2009).

With GWI, Wild et al. (2022) proposes GWI-net, a variational GP parameterised with a neural network mean to replace the svGP mean in (3.8). The experimental results of this approach in Wild et al. (2022) show promising potential in both regression and classification tasks.

5 Improving Gaussian Wasserstein Inference

In this section, we propose improvements to GWI for GPs, the function space GVI framework from Wild et al. (2022). We introduce new prior and variational kernels for GWI, further exploring the flexibility of the learning framework. We also introduce a new modular GWI learning procedure. GPs learned with this procedure will be called GWI-GPs. Finally, we introduce numerical approximations of the Wasserstein distance that significantly improves inference speed.

5.1 Improved Kernel Selection

The GWI framework is parameterised in terms of the infinite-dimensional parameters:

- $m_P \in \mathcal{L}^2$, a prior mean function,
- $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, a prior (trace-class) kernel,
- $m_Q \in \mathcal{L}^2$, a variational mean function, and
- $r : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, a variational (trace-class) kernel

to construct the objective in (4.3). Wild et al. (2022) proposes GWI-net, parameterising m_P as the zero mean function, k as the ARD kernel, m_Q as a neural network, and r as the svGP kernel. The following sections will introduce new parameterisations for the prior and variational kernels.

5.1.1 Prior Kernels

The GWI-net construction from Wild et al. (2022) uses an ARD kernel for the prior given as

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{1}{\alpha_d^2}(x_d - x'_d)^2\right), \tag{5.1}$$

where $\sigma_f > 0$ is the kernel scaling factor and $\alpha_d > 0$ is the length-scale for each dimension $d = 1, \ldots, D$. In structured data settings like images when it is important to learn correlations between features, having an independent length-scale for each dimension can be less effective. We propose using kernels that are better suited for structured data settings. These include the neural network GP (NNGP) kernels from Novak et al. (2019) and a form of neural network kernels that we will call custom feature mapping kernels.

NNGP Kernels Novak et al. (2019) shows that NNGP kernels are the infinite-width limit of neural network architectures at initialisation. We suggest choosing NNGP kernels with architectures known to be suitable for the given data domain (i.e. a convolutional neural network (CNN) for image data). The learnable hyper-parameters of NNGP kernels are the variances of the weights and biases for each infinite-width limit layer.

Figure 5.1 compares inducing point selection of different kernels for MNIST data (FCNN denotes a fully connected neural network). Using Algorithm 1, we selected 10 images from 5000 for each digit. We also included random selection as a control for visual comparison. We observed that NNGP kernels are much better at selecting different instances of the same digit, providing better coverage of the data variation. On the other hand, the ARD kernel is comparable to random selection. This motivates the use of NNGP kernels in more structured data settings. These experiments were performed without any training on the kernel hyper-parameters.

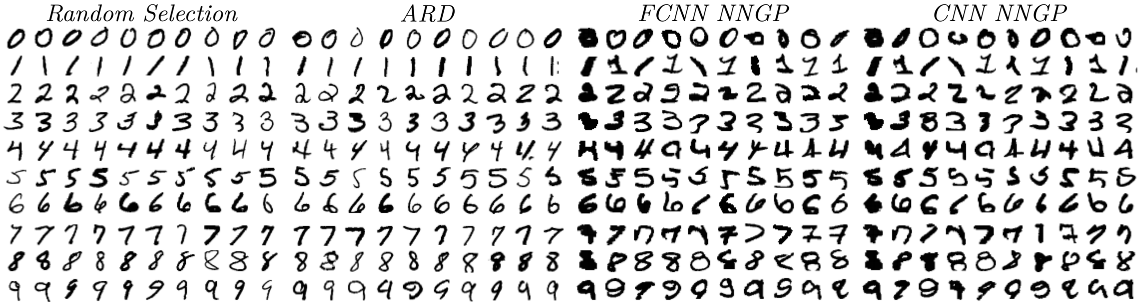


Figure 5.1: Inducing Point Selection with Different Kernels

Custom Feature Mapping Kernels We also explored neural network kernels of the form

$$k(x, x') = k_0(h(x), h(x')), \tag{5.2}$$

where $h : \mathcal{X} \to \mathbb{R}^D$ is a mapping to a D-dimensional feature space and $k_0 : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a base kernel that can be any kernel function. We use Algorithm 2 to learn the hyper-parameters of k_0 and h. In our experiments, we chose non-stationary kernels for k_0 to ensure non-zero gradients for the Gram matrix diagonals $k_0(h(x), h(x))$ with respect to the hyper-parameters of h. Like with NNGP kernels, we suggest choosing a suitable neural network architecture for h depending on the data domain.

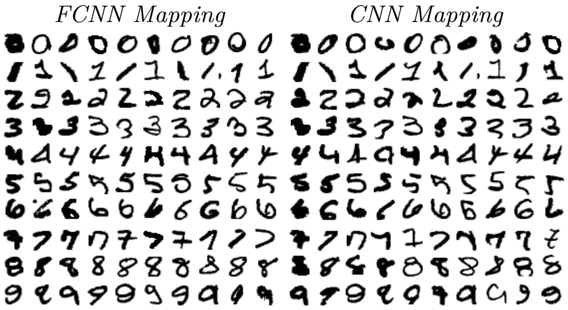


Figure 5.2: Inducing Point Selection with Randomly Initialised Feature Mapping Kernels

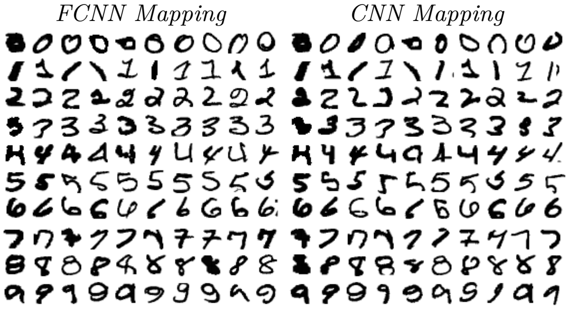


Figure 5.3: Inducing Point Selection with Trained Feature Mapping Kernels

To motivate our approach, Figure 5.2 shows inducing points selected with Algorithm 1 using randomly initialised custom feature mapping kernels and selecting from the same 5000 images of each digit as for Figure 5.1. We chose k_0 as a linear kernel and h as a neural network (FCNN or

CNN) with an output layer of $D = 256$ features. We observed that even with random initialisation, the selected images are more diverse than the ARD kernel. Figure 5.3 visualises the inducing points after learning the hyper-parameters of k with Algorithm 2. These images are qualitatively comparable to those selected by NNGP kernels.

5.1.2 Variational Kernels

The svGP kernel is given as

$$r(x, x') = k(x, x') - \mathbf{K}_{x,\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1} \mathbf{K}_{\mathbf{Z},x'} + \mathbf{K}_{x,\mathbf{Z}} \mathbf{\Sigma} \mathbf{K}_{\mathbf{Z},x'}, \tag{5.3}$$

where $\mathbf{\Sigma} \in \mathbb{R}^{M \times M}_{\succcurlyeq 0}$ is the hyper-parameter of r. Wild et al. (2022) proposes learning the positive semi-definite matrix $\mathbf{\Sigma}$ by learning the parameters of $\mathbf{L} \in \mathbb{R}^{M \times M}$, a lower triangle matrix with positive diagonal elements. As a Cholesky decomposition, this ensures that $\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^T \in \mathbb{R}^{M \times M}_{\succcurlyeq 0}$. We call this the *Cholesky parameterisation*. This section introduces new parameterisations for the svGP kernel followed by new forms of variational kernels.

svGP Parameterisations We introduce a new parameterisation of $\mathbf{\Sigma}$ called the *diagonal parameterisation*. The diagonal parameterisation learns

$$\operatorname{diag}(\mathbf{v}) : \log(\mathbf{v}) \in \mathbb{R}^M, \tag{5.4}$$

where $\mathbf{\Sigma} = \operatorname{diag}(\mathbf{v}) \in \mathbb{R}^{M \times M}_{\succcurlyeq 0}$, a diagonal matrix of positive elements, and $\operatorname{diag} : \mathbb{R}^M \to \mathbb{R}^{M \times M}$, using a vector in $\mathbb{R}^M$ to construct a diagonal matrix in $\mathbb{R}^{M \times M}$. Although this new parameterisation is more restrictive than the Cholesky parameterisation, it is computationally cheaper during gradient-based learning and has shown to have comparable performance during our experiments. This suggests that a Cholesky decomposition may be more expressive than needed for our purposes.

We also propose a *base kernel parameterisation* replacing $\mathbf{K}_{x,\mathbf{Z}} \mathbf{\Sigma} \mathbf{K}_{\mathbf{Z},x}$, the last term of (5.3) such that

$$r(x, x') = k(x, x') - \mathbf{K}_{x,\mathbf{Z}} \left(\mathbf{K}_{\mathbf{Z},\mathbf{Z}}\right)^{-1} \mathbf{K}_{\mathbf{Z},x'} + r_0(x, x'), \tag{5.5}$$

where r_0 is the base kernel, which can be any kernel function. We learn the hyper-parameters of r_0 with GWI through gradient-based optimisation.

Sparse Posterior Kernels Inspired by the GP posterior covariance in (2.10) and the svGP kernel from Titsias (2009), we present a new form of variational kernel we call *sparse posterior kernels* with the formulation

$$r(x, x') = r_0(x, x') - r_0\left(x, \mathbf{Z}\right) r_0\left(\mathbf{Z}, \mathbf{Z}\right)^{-1} r_0\left(\mathbf{Z}, x'\right), \tag{5.6}$$

where r_0 is any base kernel learned during GWI such that $r_0\left(x, \mathbf{Z}\right) \in \mathbb{R}^{1 \times M}$, $r_0\left(\mathbf{Z}, \mathbf{Z}\right) \in \mathbb{R}^{M \times M}$, and $r_0\left(\mathbf{Z}, x'\right) \in \mathbb{R}^{M \times 1}$ are Gram matrices of r_0 following the construction in (2.3). In our experiments, we chose r_0 as the same kernel as the prior kernel k. We also initialised its hyper-parameters with the hyper-parameters of k. Treating $k\left(\mathbf{Z}, \mathbf{Z}\right)^{-1}$ as a pre-computed constant, sparse posterior kernels are $\mathcal{O}(M^2)$ to evaluate.

Fixed Sparse Posterior Kernels During GP prediction with sparse posterior kernels, the inversion $r_0\left(\mathbf{Z}, \mathbf{Z}\right)^{-1}$ is a constant. However during inference, gradients must pass through it to learn the parameters of the base kernel. Thus, we also present a *fixed* version of the sparse posterior kernel

$$r(x, x') = r_0(x, x') - r_0\left(x, \mathbf{Z}\right) k\left(\mathbf{Z}, \mathbf{Z}\right)^{-1} r_0\left(\mathbf{Z}, x'\right), \tag{5.7}$$

where the matrix inversion is a constant during inference and defined with respect to the prior kernel. This option provides faster training, no longer needing gradients to pass through a matrix inversion.

5.2 An Improved Learning Procedure

We improve on the GWI learning procedure from Wild et al. (2022) with Algorithm 3. This algorithm abstracts the specific parameterisation of GWI-nets. It also includes an inducing point selection method, whereas Wild et al. (2022) selected inducing points randomly. The rest of this section provides further details for certain steps of this procedure.

Algorithm 3 GWI-GP Learning

```
Require: $k$, $m_Q$, $r$                                  ▷ use zero mean for $m_P$
  $(\theta, \mathbf{Z}) \leftarrow$ KernelAndInducingPoints($k$)    ▷ via Algorithm 2, $\theta$ are hyper-parameters of $k$
  $P \leftarrow (k, \theta)$                                ▷ construct prior/regulariser GP
  $Q \leftarrow (m_Q, r, \mathbf{Z})$                            ▷ construct GWI-GP
  $(\gamma_{m_Q}, \gamma_r) \leftarrow$ OptimiseGWI($P$, $Q$)       ▷ via GWI, $\gamma_{m_Q}$, $\gamma_r$ are hyper-parameters of $m_Q$, $r$
  $\gamma_r \leftarrow$ OptimiseTemper($\gamma_r$)             ▷ via type-II maximum likelihood on a validation set
  return $(\gamma_{m_Q}, \gamma_r)$
```

The Regulariser The prior is an essential component of a Bayesian posterior. However in the context of GVI, the prior can be interpreted as a regulariser for empirical risk minimisation. To prevent confusion in this discussion, we will call this the regulariser. In our experiments, we explored traditional regularisers constructed with the standard GP prior, having $\mathcal{O}(1)$ time complexity . Additionally, we propose another option for the GP regulariser

$$y|\mathbf{Z}, \mathbf{U}, \sigma^2 \sim \mathcal{N}\left(\bar{m}_P(x), \bar{k}(x, x)\right), \tag{5.8}$$

a 'lightweight' Bayesian posterior using the inducing points $\mathbf{Z} \in \mathcal{X}^M$ and corresponding responses $\mathbf{U} \in \mathbb{R}^M$. This regulariser would violate the traditional interpretation of a prior as it would explicitly use training data twice during inference. However, we weaken this interpretation with the optimisation-centric nature of GVI, viewing this as a potentially better-informed regulariser than the unconditioned GP prior. Our experiments show that this less-than traditional regulariser can sometimes construct a better learning objective for GWI-GPs. Moreover, with inversions of $\mathbb{R}^{M \times M}$ matrices only, we maintain computational tractability when evaluating the GWI objective.

Tempering Wild et al. (2022) and Adlam et al. (2020) explain that tempering the predictive posterior of GPs can improve predictive performance and has theoretical interpretations for correcting against a mis-specified prior. Following Wild et al. (2022), we temper GWI-GPs by learning a factor $\alpha_T > 0$ for the tempered kernel formulated as

$$r_T(\mathbf{X}, \mathbf{X}) = \alpha_T \left(r(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} \right), \tag{5.9}$$

where α_T is learned through negative log-likelihood optimisation on $Q\left(m_Q, r_T\right)$ with a hold-out validation set. The rest of the GP hyper-parameters are held fixed during this process. This forms the final GWI-GP such that the predictive posterior is given as

$$y \sim \mathcal{N}\left(m_Q(x), \alpha_T \left(r(x,x) + \sigma^2 \right) \right). \tag{5.10}$$

Tempering is also applied to classification by constructing $\left[\alpha_T^1 \dots \alpha_T^J\right]^T \in [0, \infty)^J$, a separate tempering factor for each GP, and optimising the classification log-likelihood in (2.14).

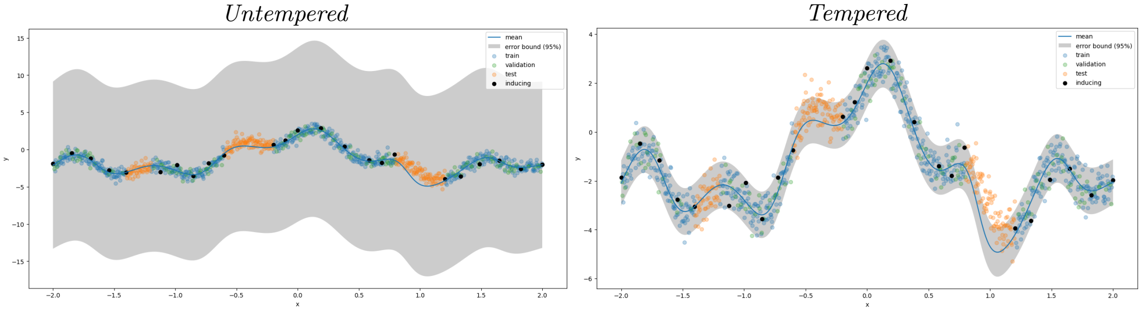


Figure 5.4: Tempering the GWI-GP *(black inducing points, blue training points, green validation points, and orange test points)*

Figure 5.4 visualises an extreme example of a GP before and after tempering. We observe that α_T is essentially 'squeezing' the variance to more appropriate uncertainty bounds for the data, while the rest of the GP's behaviour remains the same.

5.3 Wasserstein Distance Approximations

Wild et al. (2022) provides the batched approximation of the Wasserstein distance from (4.7) for N data points as

$$\begin{aligned} \mathbb{W}\left[Q, P\right]^2 \approx & \frac{1}{N_B} \sum_{n_b=1}^{N_B} \left(m_P(x_{n_b}) - m_Q(x_{n_b}) \right)^2 \\ & + \frac{1}{N_B} \sum_{n_b=1}^{N_B} k(x_{n_b}, x_{n_b}) + \frac{1}{N_B} \sum_{n_b=1}^{N_B} r(x_{n_b}, x_{n_b}) \\ & - \frac{2}{\sqrt{N_B N_S}} \sum_{n_s=1}^{N_S} \sqrt{\lambda_{n_s} \left(r\left(\mathbf{X}_{N_S}, \mathbf{X}_{N_B}\right) k\left(\mathbf{X}_{N_B}, \mathbf{X}_{N_S}\right) \right)}, \end{aligned} \tag{5.11}$$

where $N_B < N$ and $N_S < N$ for two independent batches. $\lambda_n : \mathbb{R}^{N_S \times N_S} \to \mathbb{R}$ evaluates the n^{th} eigenvalue. We introduce further numerical approximations of (5.11) to improve computation speed.

Symmetric Matrices In our experiments, we use the same data sample for $\mathbf{X}_{N_B}$ and $\mathbf{X}_{N_S}$, to ensure symmetric square matrices for $r\left(\mathbf{X}_{N_S}, \mathbf{X}_{N_B}\right)$ and $k\left(\mathbf{X}_{N_B}, \mathbf{X}_{N_S}\right)$ in the eigendecomposition term of (5.11). This allows us to take advantage of eigenvalue properties of symmetric matrices, further explained in Appendix A.2. We use symmetric matrices to leverage the eigendecomposition implementation in version 0.4 of Python JAX, which is only available on GPUs for symmetric matrices. Our experiments showed that using the same batch for $\mathbf{X}_{N_S}$ and $\mathbf{X}_{N_B}$ doesn't seem to make any significant difference on the resulting GWI-GP.

Dropping the Eigendecomposition The eigendecomposition in the last term of (5.11) is computationally expensive with $\mathcal{O}(N^3)$ complexity, which in our case is the sample size N_S. In our GWI experiments we often dropped this term with negligible practical effects. This suggests that for our use cases, the correlation constraints imposed by the eigendecomposition is a less essential contributor to achieving good prediction performance. This imprecise regulariser, which caused minimal change to our inference results, was significantly counteracted by much faster training speeds.

6 Projected Generalised Variational Inference for Gaussian Processes

This section proposes projected GVI (pGVI) for GPs, a new framework to learn variational GPs that we call pGVI-GPs. pGVI uses the same learning procedure in Algorithm 3, except it replaces the Wasserstein regularisation with a new form of regularisation that we call projected regularisation. Projected regularisations are computationally cheap, making pGVI an extremely attractive option in practice. We will present the general projected regularisation formulation followed by different constructions of the pGVI objective.

6.1 Projected Regularisation

The GVI framework from Knoblauch et al. (2022) assumes

$$\mathbb{D}\left[Q, P\right] = 0 \Leftrightarrow P = Q, \tag{6.1}$$

in other words, that the regulariser $\mathbb{D}$ is definite. We weaken this assumption and propose regularisers of the form

$$\mathbb{D}\left[Q, P\right] = \frac{1}{N}\sum_{n=1}^{N} \mathbb{D}_0\Big[\mathbb{Q}\left(F(x_n)\right), \mathbb{P}\left(F(x_n)\right)\Big], \tag{6.2}$$

where $x_n \in \mathcal{X}$ and $\mathbb{D}_0$ is a (base) regulariser between two probability measures on $\mathbb{R}$. Specifically, since both the variational approximation and regulariser are GPs, we have that $\mathbb{Q}\left(F(x_n)\right) = \mathcal{N}\left(m_Q(x_n), r(x_n, x_n)\right)$ and $\mathbb{P}\left(F(x_n)\right) = \mathcal{N}\left(m_P(x_n), k(x_n, x_n)\right)$. Therefore any base regulariser $\mathbb{D}_0$ that can be computed between two scalar Gaussian random variables will lead to a tractable regulariser computation and therefore a tractable variational loss. We call regularisation schemes following (6.2), projected regularisation.

Projected regularisations were inspired by the experimental observation that dropping the eigendecomposition in the GWI objective (5.11) had negligible effect on the training procedure. By only regularising against the marginal function values in (6.2), we are similarly unconstrained with respect to correlations. However our experiments show that this can be a sufficient regulariser to recover most of the benefits of GVI in terms of uncertainty quantification. This suggests that in practice, the assumption of a definite regulariser may sometimes be overly strict. In most VI use cases where Q is an approximation, we would rarely expect to actually be in a setting where Q matches our target posterior. The experimental success of projected regularisation suggests that a regulariser $\mathbb{D}[Q, P] \geq 0$ having capacity to prevent overfitting to the empirical risk may be sufficient to provide a reasonable learning objective for GVI.

The experimentally-driven approach of pGVI is attractive because it is computationally cheaper than GWI or any other previous approaches to function space VI. The base regularisers that we propose have $\mathcal{O}(1)$ time complexity with respect to a training input x_n. For a batch of N_B points we have $\mathcal{O}(N_B)$, much faster than $\mathcal{O}(N_B^3)$ for the Wasserstein distance in (5.11). Other approaches like Rudner et al. (2020), Sun et al. (2019), Ma et al. (2019), and Ma and Hernández-Lobato (2021) generally attempt to stochastically approximate a function space KL divergence with some finite N-dimensional realisation, also having complexity $\mathcal{O}(N_B^3)$. This makes the linear time complexity of pGVI an extremely attractive approach in practice.

6.2 Base Regularisers

We now review the base regularisers $\mathbb{D}_0$ used in our experiments. These provide different constructions of the pGVI objective, demonstrating the modularity of our approach. In the following, we denote $Q_x \coloneqq \mathbb{Q}\left(F(x)\right)$, $P_x \coloneqq \mathbb{P}\left(F(x)\right)$, $\mu_p \coloneqq m_P(x)$, $\mu_q \coloneqq m_Q(x)$, $\sigma_p^2 \coloneqq k(x,x)$, and $\sigma_q^2 \coloneqq r(x,x)$.

The Bhattacharyya distance Given as

$$\mathbb{B}\left[Q_x, P_x\right] = \frac{1}{4}\frac{\left(\mu_p - \mu_q\right)^2}{\sigma_p^2 + \sigma_q^2} + \frac{1}{2}\log\left(\frac{\sigma_p^2 + \sigma_q^2}{2\sigma_p\sigma_q}\right). \tag{6.3}$$

The Wasserstein distance Given as

$$\mathbb{W}\left[Q_x, P_x\right]^2 = \left(\mu_p - \mu_q\right)^2 + \sigma_p^2 + \sigma_q^2 - 2\sigma_p\sigma_q \tag{6.4}$$

for Gaussians on $\mathbb{R}$.

The Hellinger divergence Given as

$$\mathbb{H}\left[Q_x, P_x\right] = 1 - \sqrt{\frac{2\sigma_p\sigma_q}{\sigma_p^2 + \sigma_q^2}}\exp\left(-\frac{1}{4}\frac{\left(\mu_p - \mu_q\right)^2}{\sigma_p^2 + \sigma_q^2}\right). \tag{6.5}$$

The KL divergence Given as

$$\mathbb{KL}\left[Q_x, P_x\right] = \log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + \left(\mu_p - \mu_q\right)^2}{2\sigma_q^2} - \frac{1}{2} \tag{6.6}$$

for Gaussian on $\mathbb{R}$.

The α-Renyi divergence Given as

$$\mathbb{R}_\alpha\left[Q_x, P_x\right] = \log\frac{\sigma_p}{\sigma_q} + \frac{1}{2(\alpha-1)}\log\left(\frac{\sigma_p^2}{\alpha\sigma_p^2 + (1-\alpha)\sigma_q^2}\right) + \frac{1}{2}\frac{\alpha\left(\mu_p - \mu_q\right)^2}{\alpha\sigma_p^2 + (1-\alpha)\sigma_q^2}, \tag{6.7}$$

with $\alpha\sigma_p^2 + (1-\alpha)\sigma_q^2 > 0$.

A naive divergence Finally, we also experimented with a naive divergence

$$\mathbb{N}\left[Q_x, P_x\right] = \left(\mu_p - \mu_q\right)^2 + \left(\sigma_q^2 - \sigma_p^2\right)^2, \tag{6.8}$$

combining the squared difference of the means and covariances.

7 Experimentation Framework

This section presents the experimentation framework that we developed for GWI-GPs and pGVI-GPs. We begin by summarising our implementation architecture followed by a description of our experimentation setup. This is followed by some results for 1-D regression problems to show our framework in action. Our Python implementation is openly available on GitHub[1]. It should be noted that our implementation collects the terms svGP, GWI-GP, and pGVI-GP under the umbrella name 'approximate GP'.

7.1 Implementation Architecture

GWI and pGVI are both highly flexible learning frameworks. As such, abstraction was critical for ensuring scaleable and maintainable implementations. Luckily, the GVI framework proposed by Knoblauch et al. (2022) naturally translates to an architecture that we outline with the UML class diagram in Figure 7.1. For visual clarity, we only included relevant attributes and methods. We see that the GVI objective in (4.3) with ℓ and $\mathbb{D}$ has been abstracted to accommodate any valid empirical risk and regularisation, respectively. These abstractions are exactly mirrored in our implementation architecture as abstract base classes, ensuring that the same interface is maintained for all child classes.

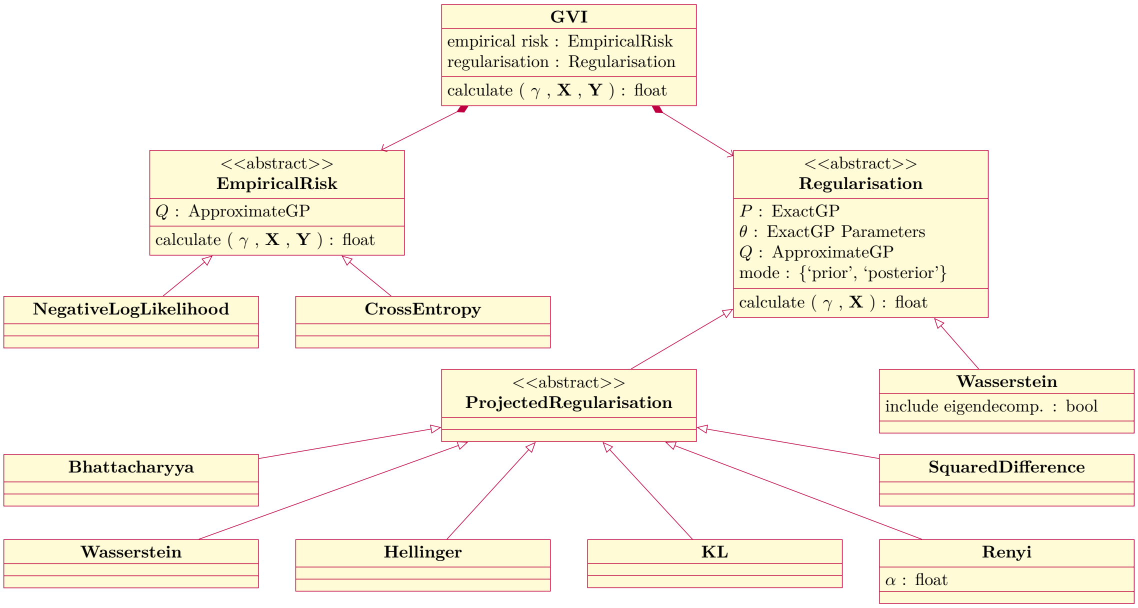

GVI
empirical risk : EmpiricalRisk
regularisation : Regularisation
calculate (γ , $\mathbf{X}$, $\mathbf{Y}$) : float
<<abstract>>
EmpiricalRisk
Q : ApproximateGP
calculate (γ , $\mathbf{X}$, $\mathbf{Y}$) : float
<<abstract>>
Regularisation
P : ExactGP
θ : ExactGP Parameters
Q : ApproximateGP
mode : {'prior', 'posterior'}
calculate (γ , $\mathbf{X}$) : float
NegativeLogLikelihood
CrossEntropy
<<abstract>>
ProjectedRegularisation
Wasserstein
include eigendecomp. : bool
Bhattacharyya
SquaredDifference
Wasserstein
Hellinger
KL
Renyi
α : float

Figure 7.1: GVI Implementation Architecture

Algorithms 1, 2, and 3 are abstracted for any mean, kernel, and variational kernel. We developed the architecture in Figure 7.2 for our implementation of GPs, Figure 7.3 for mean functions, and Figure 7.4 for kernels. These architectures are also exactly reflected in our code base.

Our implementation is in Python and predominantly uses JAX, a high-performance numerical computing package developed by Bradbury et al. (2018). Because there is currently no existing sta-

[1] For the most up-to-date version, see: https://github.com/jswu18/gvi-gaussian-process

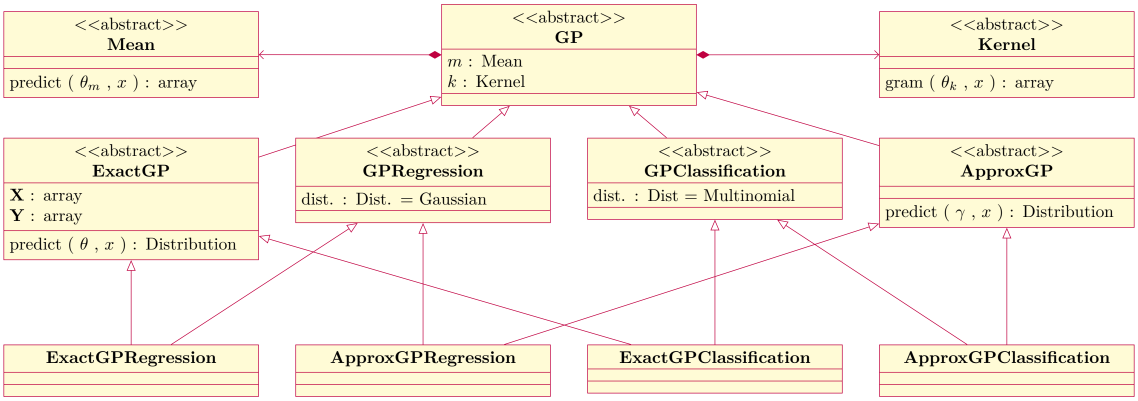


Figure 7.2: GP Implementation Architecture

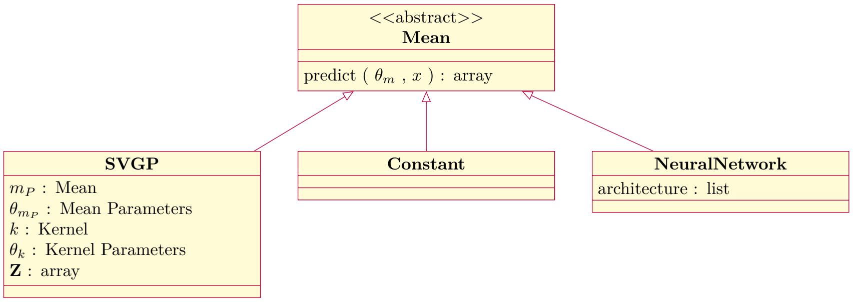


Figure 7.3: Mean Function Implementation Architecture

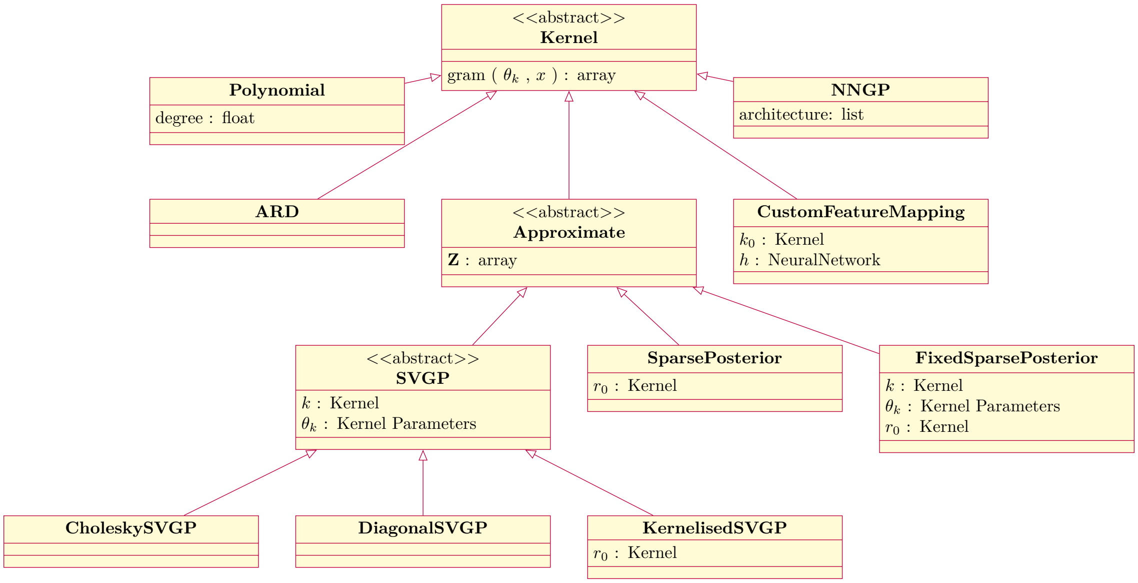


Figure 7.4: Kernel Function Implementation Architecture

ble, well-maintained, and well-documented implementation of GPs using JAX, we developed many of our implementations from scratch. We chose JAX for its currently growing user-base, unique customisations, and to ensure compatibility with the implementation of NNGP kernels by Novak et al. (2019), which is also developed in JAX. We chose Python for its current popularity within the machine learning community. We also incorporated a number of commonly used implementation techniques whenever necessary, such as adding jitter and using the Cholesky decomposition for matrix inversions.

We use a number of standard software engineering tools to ensure a scaleable, maintainable, and controlled environment for our growing code base. We introduced strict typing controls for all model parameter classes with Pydantic base models from Colvin et al. (2023). Pydantic models enforce typing hints during run-time that would otherwise be considered 'syntactic sugar' in Python. This also provides clear guidelines when constructing models within our implementation framework. We also included a rigorous number of tests that currently has 95% test coverage over the implementations found in the `src/` directory, much higher than needed in most software development projects. This also includes the use of mockers for isolated testing environments of different components (i.e. mean and kernel mockers to isolate testing of GP implementations). Finally, we use Poetry developed by Eustace to strictly control package requirements, ensuring the version compatibility of dependencies.

7.2 Experiment Scaling

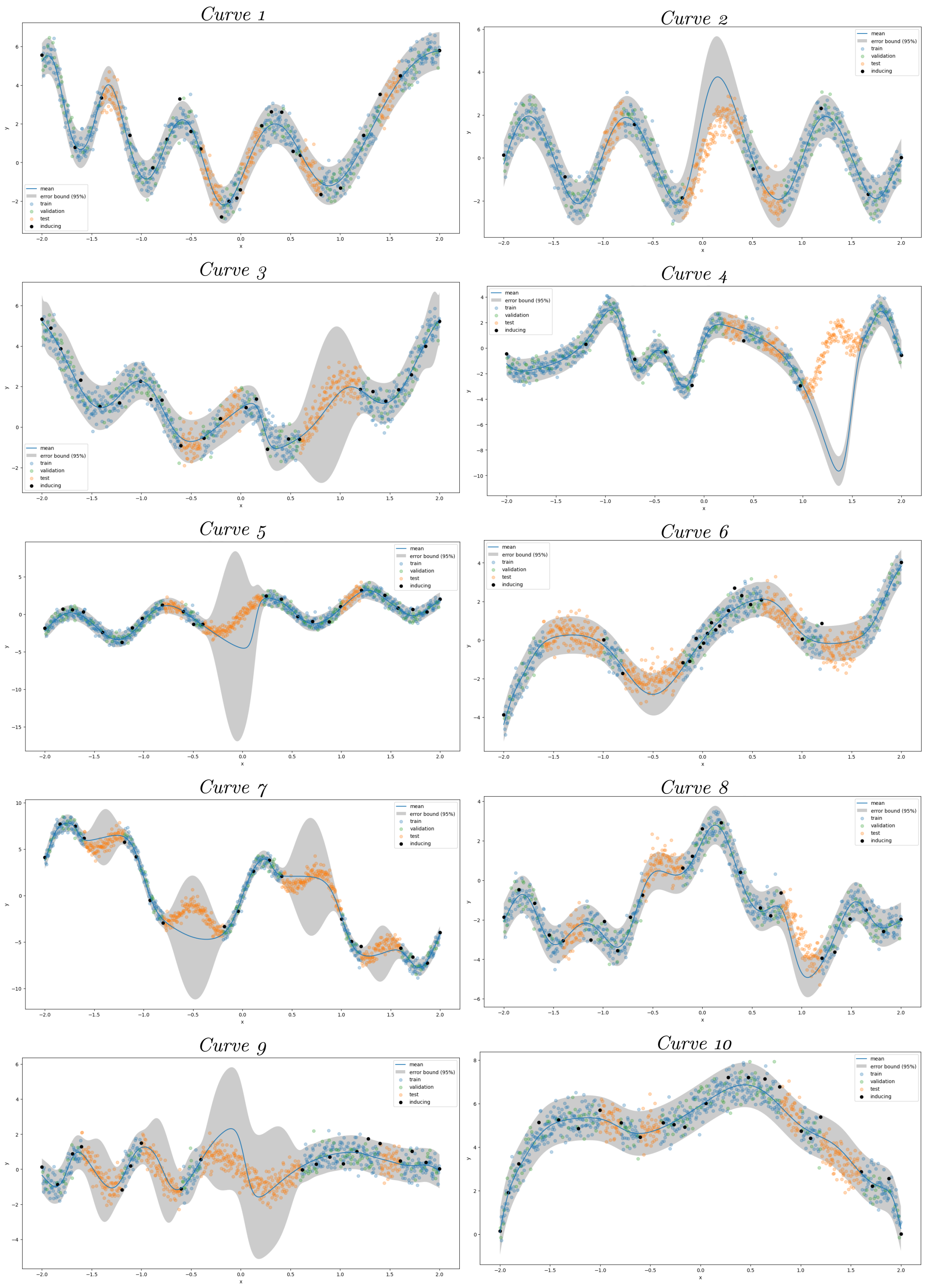
In addition to our implementation architecture, we developed a scaleable approach to experimentation. Implemented in the `experiments/` directory, we constructed schemas and resolvers for each abstraction that we described in Section 7.1. This allowed us to define and construct experiments through .`yaml` configuration files. Each .`yaml` outlines the settings of an experiment (the GP, learning objective, learning rate, optimiser, etc.). Appendix A.3 shows example .`yaml` configuration files for training a regulariser GP and an approximate GP .

Generating configuration files was automated, allowing us to scale our experiments to all possible combinations of empirical risks, regularisers, mean functions, and kernels. These configurations were then submitted as jobs to a computing cluster, running our experiments in a well-organised environment.

7.3 Regression Curve Experiments

To demonstrate our experimentation framework, we trained approximate GPs for ten different regression curve problems. Randomly selected segments of each curve were removed from the training data to simulate an out-of-distribution (OOD) setting.

Using our framework, we constructed 1260 combinations of experiment settings, each defined through .`yaml` configuration files. Each setting includes a unique combination of constructions for the regulariser GP, the approximate GP, and the GVI objective, among other parameters. The lists of options for each section that we explored are outlined in Appendix A.4. After running all 1260 combinations, we selected the experiment having the best validation set negative log-likelihood.


Curve 1

Curve 2

Curve 3

Curve 4

Curve 5

Curve 6

Curve 7

Curve 8

Curve 9

Curve 10

Figure 7.5: Best Performing Approximate GPs with respect to Validation Set NLL *(black inducing points, blue training points, green validation points, and orange test points)*

The best performing approximate GPs are shown in Figure 7.5. Our framework is quite successful at learning the training data for all the curve examples. With the exception of Curve 4, they also demonstrate reasonable behaviours in the OOD setting. Table 7.1 outlines the configurations for each best performing GP. Appendix A.4 contains further details for each setting. Interestingly, in some cases it was actually better to use fewer inducing points with $M = \mathcal{O}(N^{1/3})$. We also see that optimal settings can vary quite significantly. This demonstrates the flexibility of GWI and pGVI to accommodate the specific variational space and learning objective that may be best suited for a given problem. Please refer to the `experiments/toy_curves/` directory to reproduce our results.

Problem	$M = \mathcal{O}(N^{1/\alpha})$	$k(x, x')$	$r(x, x')$	P Mode	$\mathbb{D}[\,\cdot\,,\,\cdot\,]$	$\ell.r.$
Curve 1	$\alpha = 2$	1-layer-tanh-fcnn-nngp	diagonal-svgp	prior	proj-renyi	1e-2
Curve 2	$\alpha = 3$	1-layer-tanh-fcnn-mapping-linear-base	sparse-posterior	prior	proj-bhattacharyya	1e-4
Curve 3	$\alpha = 2$	ard	sparse-posterior	posterior	proj-renyi	1e-4
Curve 4	$\alpha = 3$	1-layer-tanh-fcnn-nngp	diagonal-svgp	posterior	proj-bhattacharyya	1e-4
Curve 5	$\alpha = 2$	ard	fixed-sparse-posterior	prior	proj-bhattacharyya	1e-4
Curve 6	$\alpha = 2$	1-layer-tanh-fcnn-nngp	kernelised-svgp	prior	proj-hellinger	1e-2
Curve 7	$\alpha = 2$	ard	kernelised-svgp	posterior	proj-bhattacharyya	1e-3
Curve 8	$\alpha = 2$	1-layer-tanh-fcnn-nngp	diagonal-svgp	prior	proj-renyi	1e-4
Curve 9	$\alpha = 2$	ard	kernelised-svgp	posterior	proj-hellinger	1e-4
Curve 10	$\alpha = 2$	ard	kernelised-svgp	prior	proj-bhattacharyya	1e-2

Table 7.1: Best Performing Settings with respect to Validation Set NLL

8 Future Work

This section discusses potential future extensions of our work. In particular, we will discuss applications to different problem domains that can push the limits of GWI and pGVI. This will hopefully provide additional feedback and understanding into the strengths and limitations of our approaches. By leveraging our code base, most of these extensions will involve minimal implementation work.

UCI Regression Benchmarking Many variational GP approaches such as Wild et al. (2022), Blundell et al. (2015), Gal and Ghahramani (2015), Li and Gal (2017), Ma and Hernández-Lobato (2021), Ma et al. (2019), and Sun et al. (2019) benchmark their approaches with standard UCI regression datasets. The next step in our work will be to benchmark the experimentation framework we proposed in Section 7 with these datasets. This will provide better understanding of GWI-GPs and pGVI-GPs within the context of the existing literature for function space VI.

Image Classification A differentiating feature of GWI is the freedom to parameterise the variational GP with any mean and kernel. Wild et al. (2022) shows promising experimental results for GWI within the context of image classification by parameterising the variational mean with a CNN. However, they use an ARD kernel to construct their svGP variational kernel.

We showed that the kernels in Section 5.1.1 have the potential to better quantify uncertainty in images by leveraging the more structured nature of the data. A natural next step is to explore the performance of GWI-GPs and pGVI-GPs with these NNGP kernels and custom feature mapping kernels in combination with the new variational kernels proposed in Section 5.1.2. With the classification GP implementations in our code base, one could extend our current work to this use-case with relative ease.

NLP Classification GWI and pGVI introduces flexibility and computational performance previously unavailable to variational GP learning. This motivates us to explore the limits of these learning frameworks with larger data settings, such as problems in the Natural Language Processing (NLP) domain. Candidate problems include sentiment analysis and named-entity recognition (NER), which would both be implementation extensions of an image classification model.

Attention networks are a fundamental building block for transformer models, the default architecture used by the NLP community at the moment. Hron et al. (2020) proposes the existence of infinite-width attention networks. These could be used to parameterise GPs within our VI frameworks. Alternatively, the custom feature mapping kernel proposed in (5.2) can accommodate any feature mapping for h, including pre-trained transformer embedding models. These would both be reasonable approaches for constructing variational GPs in the NLP domain.

9 Conclusions

Generalised variational inference (GVI) proposed by Knoblauch et al. (2022) is an exciting development within the field of Bayesian inference. An infinite dimensional realisation of GVI, Gaussian Wasserstein inference (GWI) from Wild et al. (2022), establishes a new direction for function space variational inference, pivoting from previous approaches to the problem. This has only started to demonstrate the potential of GVI, when applied to Gaussian processes (GPs).

Through the modular nature of Algorithm 3 and new kernel parameterisations, we further explored the flexibility of GWI. With projected GVI (pGVI), we experimentally showed that weakening the assumption of a definite regulariser in GVI can still maintain a sensible loss objective for GP variational learning. Compared to the cubic time complexity of previous methods, projected regularisation is shown to be an attractive and practical linear time alternative. pGVI is an even more flexible learning framework and improves the accessibility of GPs for new problem domains that we hope to explore in the future.

Our proposed methods were developed in an experimentally-driven environment, primarily motivated to improve the computational and predictive performance of variational GPs. We hope that our contributions have inspired helpful insights into function space variational inference and will provide practical tools for future work with Gaussian processes.

Bibliography

Ben Adlam, Jasper Snoek, and Samuel L Smith. Cold posteriors and aleatoric uncertainty. *arXiv preprint arXiv:2008.00029*, 2020.

Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. *Advances in neural information processing systems*, 28, 2015.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

David R Burt, Sebastian W Ober, Adrià Garriga-Alonso, and Mark van der Wilk. Understanding variational inference in function-space. *arXiv preprint arXiv:2011.09421*, 2020a.

David R Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Convergence of sparse variational inference in gaussian processes regression. *The Journal of Machine Learning Research*, 21(1): 5120–5182, 2020b.

Samuel Colvin, David Montague, Adrian Garcia Badaracco, Hasan Ramezani, Eric Jolibois, Marcelo Trylesinski, Terrence Dorsey, pyup.io bot, Serge Matveenko, Arseny Boykov, Sebastián Ramírez, David Hewitt, Nikita Grishko, Koudai Aono, Yurii Karabas, Vitaly Samigullin, Stephen Brown II, Viicos, Amin Alaee, Davis Kirkendall, layday, Yasser Tahiri, Daniel Smith, Marc Mueller, Nuno André, Hmvp, John Carter, and Ofek Lev. pydantic/pydantic: v2.3.0, August 2023. URL `https://doi.org/10.5281/zenodo.8277473`.

Sébastien Eustace. Poetry. URL `https://github.com/python-poetry/poetry`.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Insights and applications. In *deep learning workshop, ICML*, volume 1, page 2, 2015.

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

Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: NNGP and NTK for deep attention networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4376–4386. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/hron20a.html`.

Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. An optimization-centric view on bayes' rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research*, 23(132):1–109, 2022.

Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Fast dpp sampling for nystrom with application to kernel methods. In *International Conference on Machine Learning*, pages 2061–2070. PMLR, 2016.

Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International conference on machine learning*, pages 2052–2061. PMLR, 2017.

Chao Ma and José Miguel Hernández-Lobato. Functional variational inference based on stochastic process generators. *Advances in Neural Information Processing Systems*, 34:21795–21807, 2021.

Chao Ma, Yingzhen Li, and José Miguel Hernández-Lobato. Variational implicit processes. In *International Conference on Machine Learning*, pages 4222–4233. PMLR, 2019.

Alexander G de G Matthews, James Hensman, Richard Turner, and Zoubin Ghahramani. On sparse variational methods and the kullback-leibler divergence between stochastic processes. In *Artificial Intelligence and Statistics*, pages 231–239. PMLR, 2016.

Alexander Graeme de Garis Matthews. *Scalable Gaussian process inference using variational methods.* PhD thesis, University of Cambridge, 2017.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A Alemi, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. *arXiv preprint arXiv:1912.02803*, 2019.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

Tim GJ Rudner, Zonghao Chen, and Yarin Gal. Rethinking function-space variational inference in bayesian neural networks. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

Alexander J Smola. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th international conference on machine learning, June 29-July 2 2000.* Morgan Kaufmann, 2000.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

Veit Wild, Motonobu Kanagawa, and Dino Sejdinovic. Connections and equivalences between the nyström method and sparse variational gaussian processes. *arXiv preprint arXiv:2106.01121*, 2021.

Veit David Wild, Robert Hu, and Dino Sejdinovic. Generalized variational inference in function spaces: Gaussian measures meet bayesian deep learning. *Advances in Neural Information Processing Systems*, 35:3716–3730, 2022.

A Appendix

A.1 Positive Semi-Definite Kernels

Given a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined as an inner product of features in some feature space $\mathcal{H}$ such that

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}, \tag{A.1}$$

where $\phi : \mathcal{X} \to \mathcal{H}$, a Gram matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$ defined element-wise as

$$[\mathbf{K}]_{n,n'} = k(x_n, x_{n'}) \tag{A.2}$$

for any N points $\mathbf{X} = \{x_n\}_{n=1}^{N}$ with $x_n \in \mathcal{X}$, and any vector $\mathbf{v} \in \mathbb{R}^N$ then

$$\begin{aligned}
\mathbf{v}^T \mathbf{K} \mathbf{v} &= \sum_{n=1}^{N} \sum_{n'=1}^{N} v_n v_{n'} \left\langle \phi(x_n), \phi(x_{n'}) \right\rangle_{\mathcal{H}} &\text{(A.3)}\\
&= \left\langle \sum_{n=1}^{N} v_n \phi(x_n), \sum_{n'=1}^{N} v_{n'} \phi(x_{n'}) \right\rangle_{\mathcal{H}} &\text{(A.4)}\\
&= \left\| \sum_{n=1}^{N} v_n \phi(x_n) \right\|^2 \geq 0 &\text{(A.5)}
\end{aligned}$$

showing that $\mathbf{K}$ is a positive semi-definite matrix.

A.2 Symmetric Matrix Eigenvalues

For square symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{N \times N}$, $\sqrt{\mathbf{A}}\mathbf{B}\sqrt{\mathbf{A}}$ is also a symmetric matrix where $\sqrt{\mathbf{A}}$ is the square root such that $\mathbf{A} = \sqrt{\mathbf{A}}^T \sqrt{\mathbf{A}}$. Moreover, it can be shown that

$$\lambda_n \left(\mathbf{AB}\right) = \lambda_n \left(\sqrt{\mathbf{A}}\mathbf{B}\sqrt{\mathbf{A}}\right) \tag{A.6}$$

for all $n = 1, \ldots, N$, where $\lambda_n(\cdot)$ computes the n^{th} eigenvalue.

A.3 Example Experiment Configurations

```yaml
data_name: 604913dc3af2417fb1d5a21ea26e4afd
empirical_risk_break_condition: -10
empirical_risk_schema: negative_log_likelihood
inducing_points:
  inducing_points_factor: 1.0
  inducing_points_power: 3
  inducing_points_selector_schema: conditional_variance
kernel:
  kernel_kwargs:
    input_shape:
    - 1
    layers:
      layer_1:
        layer_kwargs:
          features: 10
        layer_schema: dense
      layer_2:
        layer_kwargs: null
        layer_schema: tanh
  kernel_parameters: null
  kernel_schema: nngp
number_of_iterations: 5
save_checkpoint_frequency: 1000
trainer_settings:
  batch_drop_last: false
  batch_shuffle: true
  batch_size: 1000
  learning_rate: 0.0001
  number_of_epochs: 10000
  optimiser_schema: adabelief
  seed: 0
```

Figure A.3.1: Regulariser GP Example Configuration YAML

```yaml
empirical_risk_schema: negative_log_likelihood
kernel:
  kernel_kwargs:
    diagonal_regularisation: 1.0e-10
    is_diagonal_regularisation_absolute_scale: false
  kernel_parameters: null
  kernel_schema: sparse_posterior
mean:
  mean_kwargs:
    nn_function_kwargs:
      input_shape:
      - 1
      layers:
        layer_1:
          layer_kwargs:
            features: 10
          layer_schema: dense
        layer_2:
          layer_kwargs: null
          layer_schema: tanh
        layer_3:
          layer_kwargs:
            features: 1
          layer_schema: dense
      seed: 0
    number_output_dimensions: 1
  mean_parameters: null
  mean_schema: custom
reference_name: 7b386a3faf1f4b79ac6ff6604b6bc932
regularisation:
  regularisation_kwargs:
    mode: posterior
    alpha: 0.5
  regularisation_schema: projected_renyi
save_checkpoint_frequency: 1000
trainer_settings:
  batch_drop_last: false
  batch_shuffle: true
  batch_size: 1000
  learning_rate: 0.001
  number_of_epochs: 50000
  optimiser_schema: adabelief
  seed: 0
```

Figure A.3.2: Approximate GP Example Configuration YAML

A.4 Regression Curve Experiment Settings

The following tables outline the options for each setting in Table 7.1. Taking all combinations of these settings generates 1260 unique experiments.

$M = \mathcal{O}\left(N^{1/\alpha}\right)$
$\alpha = 2$
$\alpha = 3$

Table A.4.1: Number of Inducing Points *(α is the root factor)*

$k(x, x')$	Description
1-layer-tanh-fcnn-mapping-linear	Feature Mapping Kernel: single-layer FCNN & tanh activations with linear base kernel
1-layer-tanh-fcnn-nngp	NNGP Kernel: single-layer FCNN & tanh activations
ard	ARD Kernel

Table A.4.2: Regulariser GP Kernel

$r(x, x')$
cholesky-svgp
diagonal-svgp
fixed-sparse-posterior
kernelised-svgp
sparse-posterior

Table A.4.3: Approximate GP Kernel

P Mode
posterior
prior

Table A.4.4: Regulariser GP Mode

$\mathbb{D}[\,\cdot\,,\,\cdot\,]$
proj-bhattacharyya
proj-hellinger
proj-kl
proj-renyi
proj-squared-difference
proj-wasserstein
wasserstein-partial

Table A.4.5: GVI Regularisation *('partial' indicates dropping the eigendecomposition term)*

$\ell.r.$
1e-2
1e-3
1e-4

Table A.4.6: GVI Learning Rate