Multi-output Gaussian Processes for Large-scale Multi-class Classification and Hierarchical Data

Chunchao Ma

Department of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

July 2023

To my family.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Chunchao Ma
July 2023

Acknowledgements

First and foremost, I would like to thank my supervisor Mauricio A. Álvarez for his invaluable guidance, patience and support in my journey of studying for a PhD. I was very new to the computer science world and was struggling with English writing. I was eager to make algorithms work and improve my writing skills. Hence, I would like to express my sincere gratitude to Mauricio for his detailed guidance, essential feedback, and continuous encouragement. Further, I also thank him for encouraging me to free my mind, kiss life, and subtly change my thinking style. I was very fortunate to join his group.

I would like to thank my advisor and current supervisor Eleni Vasilaki for her support and encouragement. I really enjoyed our coffee and lunchtime together. I would also like to thank my panel chairs: Sebastian Ordyniak, Pan Peng and Mike Stannett for discussing my research. I really enjoyed beer and hiking with Sebastian and Pan.

I would like to thank Pablo A Alvarado, Pablo Moreno-Muñoz and Zhenwen Dai for all the discussion on Gaussian processes. Also, thanks to Will for giving me a chance to work with him on an applied project. I also thank Kat for proof reading my thesis.

I would like to thank my friends and colleagues in office 136 in Regent Court who let me experience a wonderful journey: Juan, Chao, Tianqi, Fariba, Senee, Wil, Mike, Arthur, Tom, Magnus, Areeb, Yan, Li, Shuo, Lawrence, Peizhen, Sina , Paul, August, Nada, Doyna, George, Ali, Abdulaziz.

The past four years have been one of the most challenging and rewarding periods of my life. This journey was made possible by the astonishing people I have met: Meizhu, Zexun, Wenyan, Xuefeng, Aplian, Helen, Chengyu, Zanjia, Meng, Yuanbo, Zhijia. Thanks.

I am so thankful to the Computer Science Department for providing an excellent working environment, and the helpful staff for always providing solutions to problems.

Finally I would like to thank my family for their unconditional love and support.

Abstract

Multi-output Gaussian processes (MOGPs) can concurrently deal with multiple tasks by exploiting the correlation between different outputs. MOGPs have been mostly used for multi-output regression datasets, where the responses of each output are continuous values. However, MOGPs have inferior performance in some complex structured datasets. For example, MOGPs demand a large computational complexity in large-scale multi-class classification. The most common type of data in multi-class classification problems consists of image data, and MOGPs are not specifically designed to handle image datasets so MOGPs have poor performance on image data that has the nature of high dimensionality. Most applications of MOGPs are restricted to regression problems with a reduced number of outputs; and particularly, MOGPs present a limited performance on hierarchical datasets, i.e., datasets where the observations are connected to each other by means of parent-child relationships forming a tree structure.

In this thesis, we address the aforementioned issues by proposing three new extensions of MOGPs separately. First, we develop a novel MOGP model to deal with large-scale multi-class classification by subsampling both training data sets and classes in each output. Second, we propose a novel model to deal with image input data sets by incorporating a convolutional kernel, which can effectively capture information from images, into our developed model above. Finally, we present a new hierarchical MOGP model with latent variables to handle hierarchical datasets, where we use a hierarchical kernel function to capture the correlation within hierarchical data structures and use latent variables to explore dependencies between outputs.

The new models are applied in various synthetic and real datasets. The results of this thesis indicate that our proposed models can improve prediction performance in corresponding datasets.

Table of contents

List of figures

List of tables

Nomenclature

Generalities

v	dimension of the input space
D	number of outputs
N	number of data points per output
$\mathcal{X}$	input space
$\mathcal{Y}$	output space
C	number of classes for all outputs $C = \sum_{d=1}^{D} C_d$ where $D \geq 1$
M	number of inducing points
$\mathbf{X}$	input training data, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ or $\mathbf{X} = \{\mathbf{X}_r\}_{r=1}^{R}$
$\mathbf{X}_r$	input training data at the r-th replica, $\mathbf{X}_r = [\mathbf{x}_r^{(1)}, \cdots, \mathbf{x}_r^{(N)}]^\top \in \mathbf{R}^{N \times v}$ where $\mathbf{x}_r^{(i)}$ is the i-th data pint in the r-th replica
$\mathbf{Z}_q$	set of inducing points per latent function $u_q(\cdot)$, $\mathbf{Z}_q = \{\mathbf{z}_q^{(m)}\}_{m=1}^{M}$
Q_H	dimension of the latent space
Q	number of latent functions $u_q(\cdot)$
$\mathbf{y}$	output training data, $\mathbf{y} = \{\mathbf{y}_d\}_{d=1}^{D}$, where $\mathbf{y}_d = [y_d(\mathbf{x}_1), \cdots, y_d(\mathbf{x}_N)]^\top$
N_*	number of test data points
$\mathbf{X}_*$	input test data, $\mathbf{X}_* = \{\mathbf{x}_{*j}\}_{j=1}^{N_*}$

Operators

$\mathbb{E}[\cdot]$	expected value

$\mathrm{cov}[\cdot,\cdot]$	covariance operator
$\otimes$	Kronecker product
$\lvert\lvert\mathbf{t}\rvert\rvert$	Euclidean length of vector $\mathbf{t}$, i.e., $\left(\sum_{i=1} t_i^2\right)^{\frac{1}{2}}$

Functions

$f(\cdot)$	latent parameter function $f(\cdot):\mathcal{X}\to\mathbf{R}$
$m(\cdot)$	mean function $m(\cdot):\mathcal{X}\to\mathbf{R}$
$k(\cdot,\cdot)$	covariance or kernel function $k(\cdot,\cdot):\mathcal{X}\times\mathcal{X}\to\mathbf{R}$
$f^c(\mathbf{x}_i)$	$f^c(\mathbf{x}_i)$ is the latent parameter function corresponding to the class c evaluated at input data points $\mathbf{x}_i$
$f_d(\mathbf{x})$	latent parameter function in d-th output evaluated at $\mathbf{x}$
$\mathbf{f}(\mathbf{x})$	vector-valued function, $\mathbf{f}(\cdot):\mathcal{X}\to\mathbf{R}^D$ and $\mathbf{f}(\mathbf{x})=[f_1(\mathbf{x}),\cdots,f_D(\mathbf{x})]^\top\in\mathbf{R}^D$
$\mathbf{K}(\mathbf{x},\mathbf{x}')$	vector-valued function $\mathbf{K}(\cdot,\cdot):\mathcal{X}\times\mathcal{X}\to\mathbf{R}^{D\times D}$ where $\mathbf{K}(\mathbf{x},\mathbf{x}')\in\mathbf{R}^{D\times D}$ is a positive semi-definite matrix
$u_q(\mathbf{x})$	q-th latent function evaluated at $\mathbf{x}$
$u_q^i(\cdot)$	i-th sample of $u_q^i(\cdot)$ drawn independent and identically distributed
$f_d^c(\mathbf{x})$	c-th latent parameter function in the d-th output evaluated at $\mathbf{x}$
$\widetilde{\mathbf{f}}_d(\mathbf{x})$	vector-valued function, $\widetilde{\mathbf{f}}_d(\mathbf{x})=\left[f_d^1(\mathbf{x}),\cdots,f_d^{C_d}(\mathbf{x})\right]^\top\in\mathbf{R}^{C_d\times 1}$
$\mathbf{f}(\mathbf{x})$	vector-valued function, $\mathbf{f}(\mathbf{x})=\left[\widetilde{\mathbf{f}}_1(\mathbf{x}),\cdots,\widetilde{\mathbf{f}}_D(\mathbf{x}))\right]^\top\in\mathbf{R}^{C\times 1}$
$k_q(\mathbf{x},\mathbf{x}')$	Gaussian process covariance function of $u_q(\mathbf{x})$
$k_{f_d^c f_{d'}^{c'}}(\mathbf{x},\mathbf{x}')$	covariance between latent functions $f_d^c(\mathbf{x})$ and $f_{d'}^{c'}(\mathbf{x}')$
$k_H(\cdot,\cdot)$	kernel function
$k_{\text{hierarchy}}(\cdot,\cdot)$	kernel function

Vectors and Matrices

$\mathbf{I}_N$	identity matrix of size N

$\mathbf{f}_*$	$\mathbf{f}_* = [f(\mathbf{x}_{*1}), \cdots, f(\mathbf{x}_{*N_*})]^\top$
$\mathbf{f}_{C,i}$	set of latent parameter functions $\{f^c(\cdot)\}_{c=1}^C$ evaluated at $\mathbf{x}_i$ and stacked in a column vector, $\mathbf{f}_{C,i} = \left[f^1(\mathbf{x}_i), f^2(\mathbf{x}_i), \cdots, f^C(\mathbf{x}_i)\right]^\top \in \mathbf{R}^{C\times 1}$
$\mathbf{f}^C$	vectors $\{\mathbf{f}_{C,i}\}_{i=1}^N$ stacked in a column vector
$\mathbf{u}_q$	$\mathbf{u}_q(\mathbf{x})$ evaluated at $\mathbf{Z}_q$, $\mathbf{u}_q = \left[u_q\left(\mathbf{z}_q^{(1)}\right), \cdots, u_q\left(\mathbf{z}_q^{(M)}\right)\right]^\top$
$\mathbf{u}$	vectors $\{\mathbf{u}_q\}_{q=1}^Q$ stacked in a column vector
$\mathbf{f}_d$	$f_d(\mathbf{x}_i)$ evaluated at all $\mathbf{x}_i \in \mathbf{X}$, $\mathbf{f}_d = [f_d(\mathbf{x}_1), \cdots, f_d(\mathbf{x}_N)]^\top$
$\mathbf{f}$	$\mathbf{f} = \left[\mathbf{f}_1^\top, \cdots, \mathbf{f}_D^\top\right]^\top \in \mathbf{R}^{DN\times 1}$
$\mathbf{B}$	positive semi-definite matrix $\mathbf{B} \in \mathbf{R}^{D\times D}$, where $\mathbf{a}^\top\mathbf{B}\mathbf{a} \geq 0$ for all $\mathbf{a} \in \mathbf{R}^D$
$\mathbf{K}_q$	covariance matrix with entries given by $k_q\left(\mathbf{z}_q^{(i)}, \mathbf{z}_q^{(j)}\right)$ with $\mathbf{z}_q^{(i)}, \mathbf{z}_q^{(j)} \in \mathbf{Z}_q$
$\mathbf{K}_{\mathbf{uu}}$	block-diagonal matrix based on $\mathbf{K}_q$ as each block
$\mathbf{K}_{\mathbf{f}_d^c \mathbf{f}_{d'}^{c'}}$	covariance matrix with entries given by $k_{f_d^c f_{d'}^{c'}}(\mathbf{x}_n, \mathbf{x}_m)$ with $\mathbf{x}_n, \mathbf{x}_m \in \mathbf{X}$
$\mathbf{H}$	latent variables, $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_D]^\top$ with $\mathbf{h}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{Q_H})$
$\mathbf{K}_{\mathbf{ff}}^H$	covariance matrix with entries given by with $k_H\left(\mathbf{h}_i, \mathbf{h}_j\right)$ with $\mathbf{h}_i, \mathbf{h}_j \in \mathbf{H}$
$\mathbf{K}_{\mathbf{ff}}^X$	covariance matrix with entries given by with $k_{\text{hierarchy}}\left(\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)}\right)$ with $\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)} \in \mathbf{X}$
$\mathbf{U}$	inducing variables, $\mathbf{U} \in \mathbf{R}^{M_\mathbf{X} \times M_\mathbf{H}}$ with $M_\mathbf{X}, M_\mathbf{H} \in \mathbf{R}$
$\mathbf{Z}^H$	inducing outputs, $\mathbf{Z}^H = \left[\mathbf{z}_1^H, \ldots, \mathbf{z}_{M_\mathbf{H}}^H\right]^\top$, $\mathbf{z}_m^H \in \mathbf{R}^{Q_H}$
$\mathbf{Z}^X$	inducing inputs, $\mathbf{Z}^X = \{\mathbf{Z}_r^X\}_{r=1}^R$ and $\mathbf{Z}_r^X = \left[\mathbf{z}_{r,1}^X, \ldots, \mathbf{z}_{r,M_r}^X\right]^\top$ in which $\mathbf{z}_{r,m}^X \in \mathbf{R}^\nu$ and $M_r \in \mathbf{R}$
$\mathbf{K}_{\mathbf{UU}}^H$	covariance matrix with entries given by $k_H\left(\mathbf{z}_i^H, \mathbf{z}_j^H\right)$ with $\mathbf{z}_i^H, \mathbf{z}_j^H \in \mathbf{Z}^H$
$\mathbf{K}_{\mathbf{UU}}^X$	covariance matrix with entries given by $k_{\text{hierarchy}}\left(\mathbf{z}_{r,i}^X, \mathbf{z}_{r',j}^X\right)$ with $\mathbf{z}_{r,i}^X, \mathbf{z}_{r',j}^X \in \mathbf{Z}^X$
$\mathbf{K}_{\mathbf{fU}}^H$	cross-covariance matrix with entries $k_H\left(\mathbf{h}_i, \mathbf{z}_j^H\right)$ evaluated at $\mathbf{H}$ and $\mathbf{Z}^H$

$\mathbf{K}_{\mathbf{fU}}^{X}$	cross-covariance matrix with entries $k_{\text{hierarchy}}\left(\mathbf{x}_r^{(i)}, \mathbf{z}_{r',j}^{X}\right)$ evaluated at $\mathbf{X}$ and $\mathbf{Z}^X$

Abbreviations

GP	Gaussian process
MOGPs	multi-output Gaussian processes
RBF	Radial Basis Function
MCMC	Markov Chain Monte Carlo
KL	Kullback-Leibler
ELBO	Evidence Lower Bound
GPLVM	Gaussian Process Latent Variable Model
ICM	Intrinsic Coregionalization Model
LMC	Linear Model of Coregionalization
MOGPs-AR	Multi-output Gaussian Processes with Augment & Reduce
RBF-ARD	The radial basis function kernel with automatic relevance determination
i.i.d.	Independent and identically distributed
G-M	Gaussian processes for multi-class classification
G-A	Gaussian processes multi-class classification with additive noise model
MG-M	Multi-output Gaussian processes for multi-class classification problems
S-20	A generated synthetic data has 20 classes
MOCGPs-AR	Multi-output Convolutional Gaussian Processes with Augment & Reduce
HMOGP-LV	Hierarchical Multi-output Gaussian Processes with Latent Variable
HGP	Gaussian process with a hierarchical kernel matrix (two-layer)
HGPInd	HGP with inducing variables
DHGP	Gaussian process with a hierarchical kernel matrix (three-layer)

LVMOGP	Multi-output Gaussian processes model that uses latent information to refer to the correlation between each output

Chapter 1

Introduction

Many tasks in machine learning can be modelled by the means of an underlying function that maps our data observations. We can build or infer such a function using existing data and then use it to predict new data. To infer the underlying function, we are bound to assume a particular class of functions due to an infinite number of possible candidates. A usual trade-off happens when assuming the function; for example, if we decide to use a simple model based on a linear function where the output adjusts proportionally to the input, then our model arguably remains too restrictive leading to poor predictions; if we choose a complex model based on a non-linear function that has enough flexibility to provide a good fit of the data, then we could incur in over-fitting that deteriorates the generalisation performance. It is unclear how to choose the underlying function to properly model our data. Thus, the application of the Gaussian process allows us to consider a broader set of functions and select them with higher probabilities whilst helping to tackle the issues of poor performance and over-fitting.

A Gaussian process (GP), derived from the conception of a Gaussian distribution, is a flexible non-parametric distribution over non-linear functions used to solve unsupervised tasks, and supervised problems for either regression or classification tasks (Williams and Rasmussen, 2006). Such a non-parametric distribution refers to *a prior distribution* that incorporates prior knowledge or beliefs about the functions. On the other hand, there is *a likelihood function* that can provide information for observations. By combining the prior distribution and the likelihood function, our knowledge can be updated through a distribution known as *a posterior distribution* that could predict unobserved points.

Gaussian processes can properly tackle single-output problems, however, solving multi-output problems simultaneously involves dependencies between the related tasks that limit their predictive performance. A multi-output Gaussian processes (MOGPs) model (Álvarez et al., 2012) is a generalisation framework of Gaussian processes to deal with multiple problems together such as multiple classifications, multiple regression problems. This model could exploit dependencies between the related outputs, where each output corresponds to a problem (e.g., regression or classification). This could lead to an improved prediction for each output and cope with the scenario of data scarcity (Álvarez et al., 2012). For example, in a student's mathematics performance (Cortez and Silva, 2008), there is a strong correlation between students' marks in the first and final periods. This correlation could assist to predict marks in the first and final periods simultaneously. Since MOGPs follow the elegant Bayes' theorem, it can provide the uncertainty for each output.

1.1 Motivation and Research Questions

The main motivation of this thesis is to use MOGPs to exploit correlations between outputs where each output is a multi-class classification problem or a regression problem. In such problems, MOGPs have poor performance in some complex structured datasets. MOGPs scale badly in large-scale multi-class datasets (see Figure 1.1 (a)). MOGPs also have weak performance in image datasets in classification problems because of the curse of high dimensionality (see Figure 1.1 (b)). Finally, current MOGPs approaches do not perform well on hierarchical datasets due to the complex structure of the dataset where such hierarchical dataset are considered to have a two-layer top-down tree-like data structure, where we refer to instances in tree-leaf nodes on the same hierarchy as replicas (see Figure 1.1 (c) for more detail).

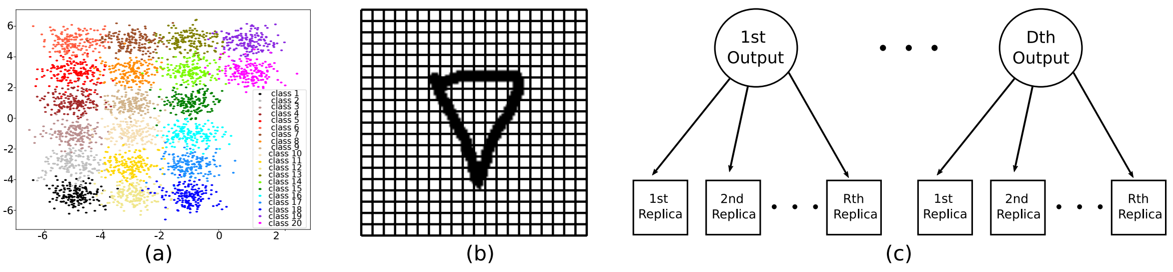


Fig. 1.1 Three different data structures: (a) a classification dataset; (b) an image data; (c) a hierarchical dataset where there are D outputs and each output has R replicas.

In terms of classification problems, MOGPs have been used to address a single-output multi-class classification problem (Chai, 2012; Dezfouli and Bonilla, 2015). Additionally, Moreno-Muñoz et al. (2018) use MOGPs to tackle multi-output multi-class classification problems. However, the approaches described above scale poorly when the number of classes is large (e.g., on the order of tens). Further, most common types of data in multi-class classification problems consist of image data. However, MOGPs are not specifically designed for image data. Therefore, we first develop a new MOGP model that aims to deal with large-scale multi-class classification datasets and then derive another novel MOGP model for the image classification datasets.

With regard to regression problems, there are many research works using various MOGP models, such as the intrinsic coregionalisation model (Bonilla et al., 2008), linear model of coregionalisation (Álvarez et al., 2012). Generally, these MOGPs models assume a flat correlation structure between the outputs. However, such a formulation does not account for more elaborate relationships, for instance, if several replicates were observed for each output (which is a typical setting in biological experiments) like hierarchical datasets. A hierarchical dataset can generally be represented as a top-down tree-like architecture. We refer to all leaf nodes of the same level as replicas since they inherit from the same parent node. The authors of (Kalinka et al., 2010) proposed a dataset where gene expression is observed through eight replicas. Gene expression is a biological process indicating how the information of a particular gene can affect the phenotype, and many practitioners aim to understand this phenomenon better. In real-world applications, many datasets present a hierarchical structure, such as the one observed in this gene expression dataset. However, it is unclear how to use MOGPs to study this kind of dataset. Further, if there is a missing replica for the same gene, it is unknown how to use MOGPs to predict this missing replica. Thus, we aim to develop MOGPs for a hierarchical structure dataset.

As mentioned above, it is not obvious how to employ MOGPs in those complex structured datasets: large-scale multi-class classification datasets, image datasets and hierarchical structure datasets. Hence, the main goal of this thesis is to use MOGPs to exploit correlations between outputs in the aforementioned datasets. We provided extensions of MOGPs to deal with said datasets.

1.2 Outline of the Thesis and Contributions

The structure of this thesis along with research contributions are presented below:

- **Chapter 2: "Gaussian Processes Overview"** contains background knowledge relating to the research in this thesis. This chapter starts by describing Gaussian processes (GPs) regression, covariance functions and GP classification. It then shows one inducing variables framework to reduce the computational complexity of GPs and variational inference to approximate intractable posterior distribution for GPs. Additionally, this chapter presents the Bayesian Gaussian process latent variable model where we integrate this methodology to obtain latent variables in Chapter 5. Finally, this chapter explains multi-output GPs which will be the foundation of our developed models.

- **Chapter 3: "Multi-output Gaussian Processes for Large-scale Multi-class Classification"** is concerned with MOGPs in large scale multi-class classification datasets. We extend multi-output Gaussian processes to deal with the classification problems by subsampling both training input data and classes in each output. The scalability of our model is achieved by using stochastic variational inference (Hensman et al., 2013a; Moreno-Muñoz et al., 2018) and by choosing a softmax likelihood function via Gumbel noise error for all outputs. We show empirically that our proposed model outperforms single-output Gaussian processes in terms of different performance metrics and multi-output Gaussian processes with respect to scalability, both in synthetic and in real classification problems.

- **Chapter 4: "Multi-output Convolutional Gaussian Processes for Images"** is about applying MOGPs to image classification datasets. A novel MOGP model is developed by integrating the convolutional kernel (Van der Wilk et al., 2017) into our proposed model in Chapter 3. To show that the model can deal with large-scale classification and image data, we apply our model to an example with the Omniglot dataset that has 1623 varied handwritten characters from 50 distinct alphabets, which means that there are 50 outputs with a total of 1623 classes. Our model is superior in this dataset compared to single output GP models and the model in Chapter 3.

- **Chapter 5: "Hierarchical Multi-output Gaussian Processes with Latent Information"** focuses on MOGPs in hierarchically structured datasets where a new MOGP model is derived. In the proposed model, to encapsulate the structure of the input domain, we create a GP with a hierarchical structure where the GP's mean is another GP. To reduce computational complexity, the proposed model also applies variational inference. We conduct experiments on the new model to show its performance compared to the single-output and multi-output Gaussian processes models and also illustrate predicting missing replicas well.

- **Chapter 6: "Conclusions and Future Work"** summarises the contributions of this thesis and also presents some ideas to improve and extend our models for future work.

Throughout my Ph.D. journey, I have delivered the following presentations and published a paper:

- Presentation: Multi-output Gaussian Processes with Transfer Learning, Machine Learning Retreat - Sheffield - 06/2019

- Poster presentation: Multi-output Gaussian Processes with Transfer Learning, Machine Learning Summer School - London - 07/2019

- Presentation: Multi-output Gaussian Processes for Large-scale Multi-class Classification and Hierarchical Data, Machine Learning Retreat - Sheffield - 07/2022
- Journal article: **Ma, C.**, & Álvarez, M. A. (2023). Large scale multi-output multi-class classification using Gaussian processes. *Machine Learning*, 1-30. The work presented in Chapter 3 and Chapter 4 is based on this journal.

Chapter 2

Gaussian Processes Overview

This chapter aims to show the background of Gaussian processes (Rasmussen and Williams, 2006) and multi-output Gaussian processes (MOGPs) (Álvarez et al., 2012). It first introduces Gaussian processes (GPs), also known as single-output GPs, for regression problems. GPs can obtain a tractable posterior distribution for inference due to the property of Gaussian distributions in regression problems with a Gaussian likelihood (Rasmussen and Williams, 2006), as discussed in Section 2.1. The next section describes covariance functions that play a significant role in GPs. In Section 2.3, we focus on GPs for classification problems where the posterior distribution is intractable. The following section discusses an inducing variables framework that can reduce the computational complexity of GPs to handle large-scale datasets. Section 2.5 goes on to present a variational inference to handle the posterior distribution with non-Gaussian likelihood. Further, Section 2.6 describes latent variable models in which we could consider the inputs as unobserved; we have only observed outputs in GPs. Finally, we explain how single-output GPs become MOGPs and present various current MOGP models in Section 2.7.

2.1 Gaussian Process Regression

A Gaussian process (GP) is a non-parametric stochastic process that allows us to place a prior distribution on an unknown mapping function $f(\cdot)$ called a latent parameter function. Using a number of observed data points and Bayes' rule, we can convert the prior distribution into a posterior distribution of $f(\cdot)$ in order to fit the data. Through learning the latent parameter function $f(\cdot)$, a GP can address different kinds of problems (e.g., regression, classification and dimensionality reduction). It can be formally defined as (Rasmussen and Williams, 2006):

Definition 1. *"A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution."*

GPs constitute the generalisation of an infinite multivariate Gaussian distribution over functions (Rasmussen and Williams, 2006). Through the marginalisation properties of this type of distribution, we can work only with a finite collection of function evaluations $\mathbf{f} = [f(\mathbf{x}_1), \cdots, f(\mathbf{x}_N)]^\top \in \mathbf{R}^N$ that are evaluated from an input dataset of $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^\top \in \mathbf{R}^{N \times v}$ where N is the number of data points and v is the dimension of each point $\mathbf{x}_i$. The input dataset also belongs to the input feature space $\mathcal{X}$. There is a latent parameter function $f(\cdot) : \mathcal{X} \to \mathbf{R}$. The output data $\mathbf{y} = [y_1, \cdots, y_N] \in \mathbf{R}^N$, a vector of observed outputs (continuous values), also belongs to an output space $\mathcal{Y}$. The S refers to the N input-output pairs dataset: $S = (\mathbf{X}, \mathbf{Y}) = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. A Gaussian process is specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, with $\mathbf{x} \in \mathbf{R}^v$ and $\mathbf{x}' \in \mathbf{R}^v$. The function $f(\cdot)$ follows a Gaussian process ($\mathbb{E}$ refers to the notation of expectation):

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{2.1}$$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{2.2}$$

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \mathbb{E}\left[\left(f\left(\mathbf{x}\right) - m\left(\mathbf{x}\right)\right)\left(f\left(\mathbf{x}'\right) - m\left(\mathbf{x}'\right)\right)\right]. \tag{2.3}$$

We usually assume the mean function $m(\mathbf{x})$ is zero. The covariance function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$ is positive semi-definite and determines the characteristics of the process, e.g., periodicity and smoothness. There are various forms and hyper-parameters of covariance functions, where hyperparameters denote the parameters of kernel functions. Covariance functions are described in Section 2.2.

For the case of regression in the dataset S, each entry of y_i is assumed to be equal to each function $f(\mathbf{x}_i)$ corrupted with a Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$:

$$y_i \mid f\left(x_i\right) \sim \mathcal{N}\left(f\left(x_i\right), \sigma^2\right), \quad \text{independently for} \quad i = 1, \ldots, N, \tag{2.4}$$

where σ^2 is a Gaussian noise variance. This leads to the likelihood function:

$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}\big(\mathbf{f}, \sigma^2 \mathbf{I}_N\big), \tag{2.5}$$

where $\mathbf{I}_N$ is an identity matrix of size N. The Gaussian likelihood function helps handle regression problems. Other kinds of likelihood functions can deal with different problems (e.g., softmax likelihoods for classification). Besides the likelihood function, the prior distribution of the vector **f** is given as

$$\begin{aligned} p(\mathbf{f}) &= \mathcal{N}\big(\mathbf{0}, k(\mathbf{X},\mathbf{X})\big) \\ &= (2\pi)^{-\frac{N}{2}} |k(\mathbf{X},\mathbf{X})|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{f}^\top k(\mathbf{X},\mathbf{X})\mathbf{f}\right), \end{aligned} \tag{2.6}$$

where $k(\mathbf{X},\mathbf{X})$ is a $N \times N$ covariance matrix built from evaluations of the covariance function between all pairs of data points $\mathbf{X}$ and $(k(\mathbf{X},\mathbf{X}))_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j)$. The prior distribution expresses the possible behaviour of the latent parameter function f before obtaining any data point (see Figure 2.1 (a)). The log of the prior is

$$\log p(\mathbf{f}) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|k(\mathbf{X},\mathbf{X})| - \frac{1}{2}\mathbf{f}^\top k(\mathbf{X},\mathbf{X})\mathbf{f}. \tag{2.7}$$

The kernel function is usually determined by a set of parameters that we refer to as hyper-parameters. We can select the optimisation hyper-parameters and a parameter in GPs, which are the hyper-parameters of the covariance function and the Gaussian noise variance in Eq. (2.4), using the maximum of log marginal likelihood. The marginal likelihood is obtained by integrating out the latent parameter function **f** in the likelihood function (2.5). Since both prior and likelihood are Gaussian distributions, the marginal likelihood is tractable. Then the corresponding log marginal likelihood is:

$$\begin{aligned} \log p(\mathbf{y} \mid \theta) &= \log \int p(\mathbf{y} \mid \mathbf{f})\, p(\mathbf{f})\, \mathrm{d}\mathbf{f} \\ &= \log \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, k(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}_N\right) \\ &= -\frac{1}{2}\mathbf{y}^\top \left(k(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}_N\right)^{-1}\mathbf{y} - \frac{1}{2}\log\left|k(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}_N\right| - \frac{N}{2}\log 2\pi, \end{aligned} \tag{2.8}$$

where $|\mathbf{A}|$ represents the determinant of $\mathbf{A}$ ($\mathbf{A}$ is a matrix), θ refers to all hyper-parameters for the kernel k and σ. Optimising the θ is to maximise the expression in Eq. (2.8).

After obtaining the optimisation parameters θ, we can employ a posterior distribution to make a prediction. Based on the Bayesian theorem, we obtain the posterior distribution of **f**:

$$\begin{aligned} p(\mathbf{f} \mid \mathbf{y}) &= \frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f})}{p(\mathbf{y} \mid \theta)} \\ &\propto \mathcal{N}\left(\mathbf{y} \mid \mathbf{f}, \sigma^2\mathbf{I}_N\right)\mathcal{N}\left(\mathbf{f} \mid \mathbf{0}, k(\mathbf{X},\mathbf{X})\right). \end{aligned} \tag{2.9}$$

The posterior distribution is proportional to the product of the likelihood function with the prior distribution. The posterior distribution can incorporate the observed data information into the latent parameter function **f**. It is also tractable since the prior distribution and the likelihood function are both Gaussian. Figure 2.1 shows samples from the GP prior and GP posterior of $\mathbf{f}$. The samples in the posterior distribution have a clear shape along with the data points as expected in Figure 2.1 (b).

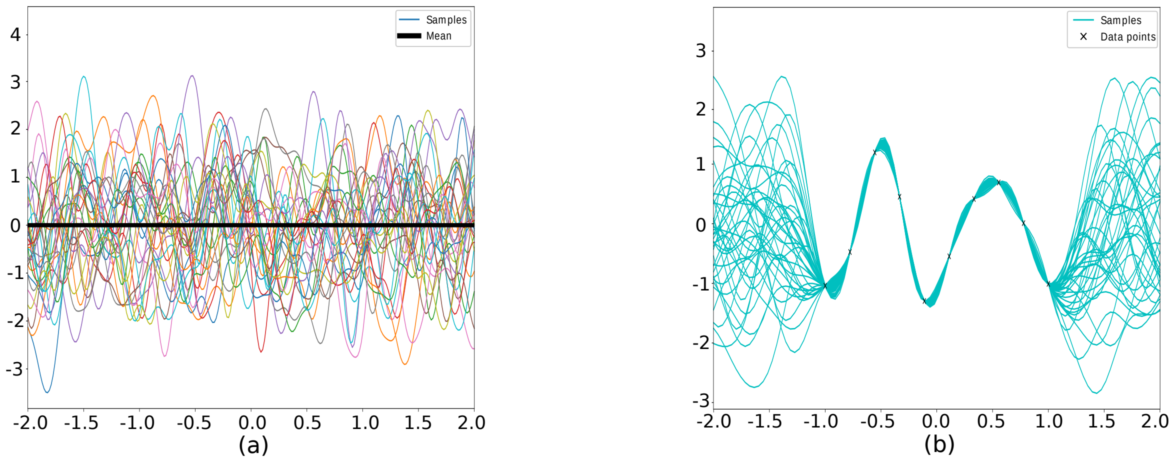


Fig. 2.1 (a): Samples from the prior distribution of $\mathbf{f}$. (b): Samples from the posterior distribution of $\mathbf{f}$ after receiving some data observations.

This is another way to derive the posterior distribution. Since both the likelihood function and the prior distribution are Gaussian distributions, the joint vector $\mathbf{y}$ and $\mathbf{f}$ follows a joint Gaussian distribution with

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X},\mathbf{X}) & k(\mathbf{X},\mathbf{X}) \\ k(\mathbf{X},\mathbf{X}) & k(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I}_N\end{bmatrix}\right). \tag{2.10}$$

Due to the properties of a Gaussian distribution (see appendix A.2), the posterior distribution of $\mathbf{f}$ is

$$\begin{aligned} p(\mathbf{f}\mid\mathbf{y}) =& \mathcal{N}\Big(\mathbf{f}\mid k(\mathbf{X},\mathbf{X})\left(k(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I}_N\right)^{-1}\mathbf{y}, \\ & k(\mathbf{X},\mathbf{X})-k(\mathbf{X},\mathbf{X})\left(k(\mathbf{X},\mathbf{X})+\sigma^2\mathbf{I}_N\right)^{-1}k(\mathbf{X},\mathbf{X})\Big). \end{aligned} \tag{2.11}$$

To make predictions for unknown data points or test data points, we can apply this formalism to test data points. Suppose $\mathbf{X}_* = \{\mathbf{x}_{*j}\}_{j=1}^{N_*} \in \mathbf{R}^{N_*\times v}$ is a test set and $\mathbf{f}_* = [f(\mathbf{x}_{*1}),\cdots,f(\mathbf{x}_{*N_*})]^\top \in \mathbf{R}^{N_*}$, where N_* is the number of test data. We can incorporate $\mathbf{f}_*$ into our model. For example, $\mathbf{f}_*$ can be included in the prior (2.6). Then the joint distribution of $p(\mathbf{f})$ and $p(\mathbf{f}_*)$ is

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X},\mathbf{X}) & k(\mathbf{X},\mathbf{X}_*) \\ k(\mathbf{X}_*,\mathbf{X}) & k(\mathbf{X}_*,\mathbf{X}_*)\end{bmatrix}\right), \tag{2.12}$$

where $k(\mathbf{X}_*,\mathbf{X}_*) \in \mathbf{R}^{N_* \times N_*}$ is the covariance matrix built for the test data points, e.g., $\left(k\left(\mathbf{X}_*,\mathbf{X}_*\right)\right)_{i,j} = k\left(\mathbf{x}_{*i},\mathbf{x}_{*j}\right)$. $k(\mathbf{X},\mathbf{X}_*) \in \mathbf{R}^{N \times N_*}$ is the covariance matrix between training and test data points, such as $\left(k(\mathbf{X},\mathbf{X}_*)\right)_{i,j} = k(\mathbf{x}_i,\mathbf{x}_{*j})$, and $k(\mathbf{X}_*,\mathbf{X})$ is the transpose of $k(\mathbf{X},\mathbf{X}_*)$, i.e., $k(\mathbf{X}_*,\mathbf{X}) = k(\mathbf{X},\mathbf{X}_*)^\top$. Similarly, the joint distribution of $\mathbf{f}_*$ and $\mathbf{y}$ is:

$$\begin{bmatrix} \mathbf{f}_* \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X}_*,\mathbf{X}_*) & k(\mathbf{X}_*,\mathbf{X}) \\ k(\mathbf{X},\mathbf{X}_*) & k(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}_N \end{bmatrix}\right). \tag{2.13}$$

Again, through the properties of a Gaussian distribution, the predictive posterior distribution of $\mathbf{f}_*$ is:

$$\begin{aligned} p\left(\mathbf{f}_* \mid \mathbf{y}\right) =& \mathcal{N}\big(\mathbf{f}_* \mid k(\mathbf{X}_*,\mathbf{X})(k(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}_N)^{-1}\mathbf{y}, \\ & k(\mathbf{X}_*,\mathbf{X}_*) - k(\mathbf{X}_*,\mathbf{X})(k(\mathbf{X},\mathbf{X}) + \sigma^2 \mathbf{I}_N)^{-1} k(\mathbf{X},\mathbf{X}_*)\big). \end{aligned} \tag{2.14}$$

This distribution provides the predictive mean and variance for the test data points.

2.2 Covariance Functions

As the above section shows, a Gaussian process is defined by its mean (usually selected as $\mathbf{0}$) and covariance function. This section is devoted to describing the covariance function which plays an important role in the Gaussian process. A covariance function, also called a kernel, encapsulates inputs into a real function: $k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$ and $k(\cdot,\cdot)$ must be a positive semi-definite. A kernel can measure the similarity between data points. For example, if data points are close to one another, those points have a high correlation and behave similarly. Kernels broadly fall into two groups: stationary kernels and non-stationary kernels. A stationary kernel function is a function of $\tau = \mathbf{x} - \mathbf{x}'$ (Rasmussen and Williams, 2006). This thesis only considers stationary kernels. In the area of Gaussian processes, one of the research focuses is on kernel functions, e.g., we construct a new kernel to build a novel multi-output Gaussian process in Chapter 5.

There are many different kernels with various formulas and hyper-parameters such as radial basis function (RBF), Matérn class of kernels (e.g., Matérn 32 kernel), periodic kernels and polynomial kernels (Rasmussen and Williams, 2006). Each kernel has its own property. For instance, RBF is infinitely differentiable and captures more smooth functions, Matérn 32 kernel has finite differentiability. Further, each kernel function has its own hyper-parameters. For example, the RBF and Matérn 32 kernel have two hyper-parameters, respectively (variance (σ^2_{RBF}, $\sigma^2_{\text{Matérn32}}$) and length scale ($l_{\text{RBF}}$, $l_{\text{Matérn32}}$)):

$$k_{\text{RBF}}\left(\mathbf{x},\mathbf{x}'\right) = \sigma^2_{\text{RBF}} \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2l^2_{\text{RBF}}}\right), \tag{2.15}$$

$$k_{\text{Matérn32}}\left(\mathbf{x},\mathbf{x}'\right) = \sigma^2_{\text{Matérn32}}\left(1+\frac{\sqrt{3}||\mathbf{x}-\mathbf{x}'||}{l_{\text{Matérn32}}}\right)\exp\left(-\frac{\sqrt{3}||\mathbf{x}-\mathbf{x}'||}{l_{\text{Matérn32}}}\right). \tag{2.16}$$

Figure 2.2 shows that hyper-parameters influence the correlation between data points in the RBF. We generate samples ($\mathbf{x} \in [-2,2]$) from a normal distribution where the mean is always zero and covariance is RBF with different scale hyper-parameters. The samples change faster with the smaller length scale (see Figures 2.2 (a) and (c)), while with the larger length scale, the functions or samples change much slower and become flat (see Figures 2.2 (b) and (d)). This indicates that samples probably have a high correlation. In terms of the magnitude of variance, the smaller the variance, the smaller the magnitude of the samples.

There is one particular kernel function called a convolutional kernel (Van der Wilk et al., 2017), which is designed for image data. In image data, we usually assume $\mathbf{x} \in \mathbf{R}^{W\times H}$ is an image data point where W and H are the width and height of the data point, respectively. Further, there are $E = w \times h$ patches for each data point, where w and h are the width and height of each patch, respectively. In total, there are $P = (W-w+1)\times(H-h+1)$ patches for each image. Then, the convolutional kernel is

$$k_{conv}\left(\mathbf{x},\mathbf{x}'\right) = \sum_{p=1}^{P}\sum_{p'=1}^{P} k\left(\mathbf{x}^{[p]},\mathbf{x}'^{[p']}\right), \tag{2.17}$$

where the k is a kernel function and $\mathbf{x}^{[p]}$ is the p-th patch of the image $\mathbf{x}$. This convolutional structure is more suitable for the image dataset. To extend the convolutional structure, Van der Wilk et al. (2017) also add weight to the expression (2.17):

$$k_{wconv}\left(\mathbf{x},\mathbf{x}'\right) = \sum_{p=1}^{P}\sum_{p'=1}^{P} w_p w_{p'} k\left(\mathbf{x}^{[p]},\mathbf{x}'^{[p']}\right), \tag{2.18}$$

where w_p is the weight along with $\mathbf{x}^{[p]}$. In Chapter 4, we combine this kernel into our model.

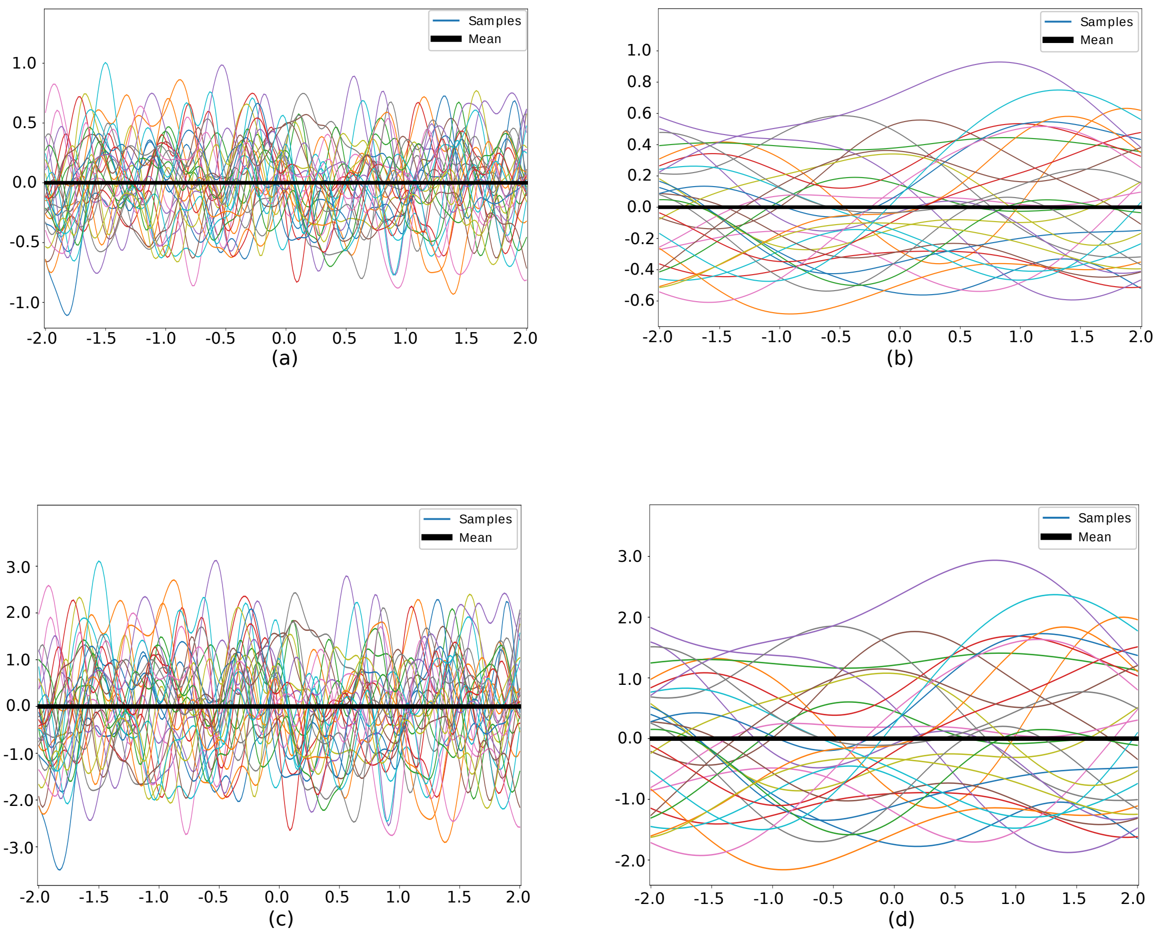


Fig. 2.2 (a): RBF with σ^2_{RBF} = 0.1 and l_{RBF} = 0.1; (b): RBF with σ^2_{RBF} = 0.1 and l_{RBF} = 1; (c): RBF with σ^2_{RBF} = 1 and l_{RBF} = 0.1; (d): RBF with σ^2_{RBF} =1 and l_{RBF} = 1.

2.3 Gaussian Process Classification

Gaussian processes can not only deal with regression problems but can also handle other kinds of problems, such as classification. In a regression problem, the observed outputs are continuous values, e.g., $\mathbf{y} = \{y_i\}_{i=1}^N \in \mathbf{R}^N$ while the outputs in a classification problem are categorical values, e.g., $y_i \in [1, 2, \cdots, C]$, and C is the number of classes. Classification problems are classified as binary ($C = 2$) or multi-class ($C > 2$). Multi-class classification is a generalisation of binary classification. This section focuses on multi-class classification problems.

Generally in classification problems based on a Gaussian process model, the likelihood function is not a Gaussian distribution. Therefore, the posterior distribution is intractable because the Gaussian process prior is not conjugate to the likelihood function. Thus it is necessary to approximate the posterior distribution. Many approximation methods exist in the literature such as Markov Chain Monte Carlo (MCMC) sampling (Neal, 1997), variational approximation (Gibbs and MacKay, 2000; Hensman et al., 2015a), expectation propagation (Minka, 2013), and the Laplace approximation (Barber and Williams, 1996). In this chapter and throughout the thesis, we use variational inference because it is expected to be faster than traditional approximation methods (Blei et al., 2017). We describe variational inference for approximating intractable integrals (more detail in Section 2.5) and reducing the computational complexity (more detail in Section 2.4) for GPs.

There are several likelihood functions that can be used for multi-class classification such as the softmax likelihood (Galy-Fajou et al., 2020; Kim and Ghahramani, 2006; Williams and Rasmussen, 2006), the multinomial probit likelihood function (Girolami and Rogers, 2006), the step function (Hernández-Lobato et al., 2011). This section will describe the softmax likelihood function that will be used in Chapters 3 and 4.

Since there are C classes, we assume there are C latent parameter functions. Suppose $f^c(\mathbf{x}_i)$ is the latent parameter function corresponding to the class c evaluated at input data points $\mathbf{x}_i$, $\mathbf{f}_C(\mathbf{x}_i)$ is a set of latent parameter functions $\{f^c(\cdot)\}_{c=1}^C$ evaluated at $\mathbf{x}_i$ and are stacked in a column vector $\mathbf{f}_C(\mathbf{x}_i) = \left[f^1(\mathbf{x}_i), f^2(\mathbf{x}_i), \cdots, f^C(\mathbf{x}_i)\right]^\top \in \mathbf{R}^{C\times 1}$, where the C latent parameter functions are uncorrelated. Then $\mathbf{f}_C$ represents $\mathbf{f}_C(\cdot)$ evaluated at $\mathbf{X}$:

$$\mathbf{f}_C = \left[f^1(\mathbf{x}_1), f^1(\mathbf{x}_2), \cdots, f^1(\mathbf{x}_N), f^2(\mathbf{x}_1), \cdots, f^2(\mathbf{x}_N), \cdots, f^C(\mathbf{x}_1), \cdots, f^C(\mathbf{x}_N)\right]^\top \in \mathbf{R}^{CN\times 1}.$$

If $\mathbf{x}$ is the input for the class c, ideally $f^c(\mathbf{x}) = \underset{c}{\operatorname{argmax}}\, \mathbf{f}_C(\mathbf{x})$. The corresponding softmax likelihood function is defined as:

$$p\left(y_i \mid \mathbf{f}_C\left(\mathbf{x}_i\right)\right) = \frac{\exp\left(f^{y_i}\left(\mathbf{x}_i\right)\right)}{\sum_{c=1}^C \exp\left(f^c\left(\mathbf{x}_i\right)\right)}. \tag{2.19}$$

Based on the conditional independence assumption, the likelihood function over the whole training dataset is:

$$p(\mathbf{y} \mid \mathbf{f}_C) = \prod_{i=1}^N p\left(y_i \mid \mathbf{f}_C\left(\mathbf{x}_i\right)\right) = \prod_{i=1}^N \frac{\exp\left(f^{y_i}\left(\mathbf{x}_i\right)\right)}{\sum_{c=1}^C \exp\left(f^c\left(\mathbf{x}_i\right)\right)}. \tag{2.20}$$

The prior function is:

$$\mathbf{f}_C \sim \mathcal{N}\left(\mathbf{0}, \mathbf{J}\right), \tag{2.21}$$

where the covariance matrix $\mathbf{J}$ is a block diagonal matrix with $\mathbf{J}_1, \cdots, \mathbf{J}_C$. It is a block diagonal matrix because we assume C latent parameter functions are independent of each other and each $\mathbf{J}_c$ presents a covariance matrix for the latent parameter function $f^c\left(\cdot\right)$ evaluated at $\mathbf{X}$

$\left(f^c(\mathbf{X}) = [f^c(\mathbf{x}_1), \cdots, f^c(\mathbf{x}_N)]^\top \in \mathbf{R}^{N \times 1}\right)$. The independence assumption will be relaxed in Chapters 3 and 4 by using multi-output Gaussian processes. Combining the likelihood function and the prior distribution through the Bayesian rule, the posterior distribution is:

$$p(\mathbf{f}_C \mid \mathbf{y}) = \frac{p(\mathbf{f}_C)\prod_{i=1}^N p(y_i \mid \mathbf{f}_C(\mathbf{x}_i))}{\int p(\mathbf{f}_C)\prod_{i=1}^N p(y_i \mid \mathbf{f}_C(\mathbf{x}_i))\,\mathrm{d}\mathbf{f}} = \frac{\mathcal{N}(\mathbf{f}_C \mid \mathbf{0}, \mathbf{J})}{p(\mathbf{y})}\prod_{i=1}^N \frac{\exp(f^{y_i}(\mathbf{x}_i))}{\sum_{c=1}^C \exp(f^c(\mathbf{x}_i))}. \tag{2.22}$$

The posterior is intractable due to the non-Gaussian likelihood function. The variational inference method approximates the posterior distribution, the detail of which is explained in Section 2.5. The predictive posterior distribution at a test data point $\mathbf{x}_*$ (the corresponding class $y_* = c$) is:

$$\begin{aligned} p(\mathbf{f}_C(\mathbf{x}_*) \mid \mathbf{y}) &= \int p(\mathbf{f}_C(\mathbf{x}_*) \mid \mathbf{f}_C)\, p(\mathbf{f}^C \mid \mathbf{y})\mathrm{d}\mathbf{f}_C, \\ p(y_* = c \mid \mathbf{y}) &= \int p(\mathbf{f}_C(\mathbf{x}_*) \mid \mathbf{y})\, p(\mathbf{f}_C \mid \mathbf{y})\mathrm{d}\mathbf{f}_C. \end{aligned} \tag{2.23}$$

The predictive posterior distribution is intractable so we use the approximated posterior distribution (2.22) through variational inference to approximate it.

2.4 Inducing Variables Framework

Before introducing the variational inference method for approximating the posterior distribution, we describe the computational complexity for Gaussian processes and the inducing variables framework to deal with it. The computational complexity is $\mathcal{O}(N^3)$ for inverting the covariance matrix $k(\mathbf{X}, \mathbf{X}) \in \mathbf{R}^{N \times N}$ and storage is $\mathcal{O}(N^2)$, where N is the number of training data points. It is intractable when N is large, limiting the application of Gaussian processes.

To tackle this kind of problem, sparse approximation methods have arisen to reduce the computational complexity (Hensman et al., 2013a; Snelson and Ghahramani, 2006; Titsias, 2009; Williams and Rasmussen, 2006). Gaussian processes, including the sparse approximation method, are referred to as sparse Gaussian processes. One popular method is to approximate the covariance matrix by introducing a set of inducing variables $\mathbf{u} = [f(\mathbf{z}_1), \cdots, f(\mathbf{z}_M)]^\top$. The set of inducing variables $\mathbf{u}$ that contain values of the latent parameter function $f(\cdot)$ are evaluated at an unknown inducing input set $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M \in \mathbf{R}^{M \times v}$ where $M \ll N$. It can ease the computational complexity to $\mathcal{O}(NM^2)$ and storage to $\mathcal{O}(NM)$ (Snelson and Ghahramani, 2006; Titsias, 2009). To demonstrate this method, we first augment the GP prior as:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(\mathbf{X},\mathbf{X}) & k(\mathbf{X},\mathbf{Z}) \\ k(\mathbf{Z},\mathbf{X}) & k(\mathbf{Z},\mathbf{Z}) \end{bmatrix}\right), \tag{2.24}$$

where $k(\mathbf{Z},\mathbf{Z}) \in \mathbf{R}^{M\times M}$ is the covariance matrix built for the inducing points, $k(\mathbf{X},\mathbf{Z}) \in \mathbf{R}^{N\times M}$ is the covariance matrix between training data observations and inducing points. By applying the properties of a Gaussian distribution (see appendix A.2) and $p(\mathbf{f},\mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u})$, the prior distribution of the inducing points $p(\mathbf{u})$ and the conditional distribution $p(\mathbf{f} \mid \mathbf{u})$ are:

$$\mathbf{u} \sim \mathcal{N}(\mathbf{0}, k(\mathbf{Z},\mathbf{Z})), \tag{2.25}$$

$$\begin{aligned} p(\mathbf{f} \mid \mathbf{u}) =& \mathcal{N}\big(\mathbf{f} \mid k(\mathbf{X},\mathbf{Z})k(\mathbf{Z},\mathbf{Z})^{-1}\mathbf{u}, \\ & k(\mathbf{X},\mathbf{X}) - k(\mathbf{X},\mathbf{Z})k(\mathbf{Z},\mathbf{Z})^{-1}k(\mathbf{Z},\mathbf{X})\big). \end{aligned} \tag{2.26}$$

Equation (2.26) shows that we only need a computational complexity $\mathcal{O}(NM^2)$ (e.g., a product like $k(\mathbf{X},\mathbf{Z})k(\mathbf{Z},\mathbf{Z})^{-1}$) to derive a conditional distribution $p(\mathbf{f}|\mathbf{u})$, which is key for sparse GPs. To some extent, the set of inducing inputs $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^{M}$ and inducing variables $\mathbf{u}$ represent the real input $\mathbf{X}$ and function $\mathbf{f}$. After obtaining the conditional distribution $p(\mathbf{f} \mid \mathbf{u})$, it will be used to derive a marginal likelihood and a predictive distribution, which is discussed in the next section.

2.5 Variational Inference for Gaussian Processes

This section aims to introduce a variational inference strategy. Variational inference introduces a family of distributions feasible to approximate difficult-to-compute probability densities (Blei et al., 2017). In our case, the variational inference method can be implemented to approximate the predictive posterior distribution given a non-Gaussian likelihood function such as the intractable posterior distribution in Section 2.3. Additionally, we can combine variational inference with the inducing variable framework (Hensman et al., 2013a; Titsias, 2009) to address the computational complexity of Gaussian processes as mentioned in Section 2.4.

As the above section mentioned, there is a joint distribution $p(\mathbf{f},\mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u})$ in the inducing variable framework. Further, the posterior distribution is $p(\mathbf{f},\mathbf{u} \mid \mathbf{y}) = p(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u} \mid \mathbf{y})$. Then, the marginal likelihood of sparse Gaussian processes is defined as

$$p(\mathbf{y}) = \iint p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u}. \tag{2.27}$$

If the likelihood function is a non-Gaussian likelihood function like Equation (2.20) in Section 2.3, both the integral above (2.27) and posterior distribution $p(\mathbf{f},\mathbf{u} \mid \mathbf{y})$ are intractable. To approximate the marginal likelihood (2.27) and obtain an approximation for the true posterior distribution, the variational inference introduces a variational distribution $q(\mathbf{f},\mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})$ where $q(\mathbf{f},\mathbf{u}) \approx p(\mathbf{f},\mathbf{u} \mid \mathbf{y})$. More precisely, $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m},\mathbf{S})$ is a Gaussian distribution with mean $\mathbf{m}$ and variance $\mathbf{S}$. $q(\mathbf{u})$ is a variational distribution of $\mathbf{u}$ and aims to approximate $p(\mathbf{u})$. By applying Jensen's inequality, we add $q(\mathbf{f},\mathbf{u})$ into the log of marginal likelihood of $\mathbf{y}$:

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \iint p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} && (2.28)\\
&= \log \iint q(\mathbf{f},\mathbf{u})\frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u})}{q(\mathbf{f},\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} && (2.29)\\
&\geq \iint p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u})}{p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} && (2.30)\\
&= \iint p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{u})}{q(\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} && (2.31)\\
&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}\mathbb{E}_{q(\mathbf{u})}\left\{\log\left(\frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{u})}{q(\mathbf{u})}\right)\right\} && (2.32)\\
&= \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}\mathbb{E}_{q(\mathbf{u})}\{\log p(\mathbf{y} \mid \mathbf{f})\} + \mathbb{E}_{q(\mathbf{u})}\left\{\log\left(\frac{p(\mathbf{u})}{q(\mathbf{u})}\right)\right\} && (2.33)\\
&= \mathbb{E}_{q(\mathbf{f})}\{\log p(\mathbf{y} \mid \mathbf{f})\} - \mathrm{KL}(q(\mathbf{u})\|p(\mathbf{u})), && (2.34)
\end{aligned}$$

where $\mathrm{KL}(q(\mathbf{u})\|p(\mathbf{u})) = -\mathbb{E}_{q(\mathbf{u})}\left\{\log\left(\frac{p(\mathbf{u})}{q(\mathbf{u})}\right)\right\}$ is the called Kullback-Leibler divergence between $q(\mathbf{u})$ and $p(\mathbf{u})$, and $q(\mathbf{f}) = \int p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u}$ refers to the approximation of the posterior distribution of $\mathbf{f}$. The predictive posterior distribution (2.23) can be obtained by applying $q(\mathbf{f})$.

To obtain the optimisation parameters and hyper-parameters of the Gaussian process, we optimise the lower bound of log marginal likelihood, also called evidence lower bound (ELBO), with respect to parameters and hyper-parameters.

To apply GPs to large datasets, we can apply stochastic variational inference for GPs, which is described in subsection 2.7.2 for multi-output Gaussian processes. Based on Hensman et al. (2013a), we can factorise the ELBO so we can perform optimisation using a small number of mini-batch of training samples to reduce the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^3)$ and reduce the storage from $\mathcal{O}(N^2)$ to $\mathcal{O}(M^2)$ where $M \ll N$.

As the approximation of the posterior distribution $q(\mathbf{f}) = \int p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u}$ and the approximated posterior $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m},\mathbf{S})$, the approximated predictive posterior distribution

$q(\mathbf{f}_*)$ is

$$\begin{aligned} q(\mathbf{f}_*) &= \int p(\mathbf{f}_* \mid \mathbf{u}) q(\mathbf{u}) \mathrm{d}\mathbf{u} \\ &= \mathcal{N}\Big(\mathbf{f}_* \mid k(\mathbf{X}_*, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} \mathbf{m}, \qquad (2.35) \\ &\quad k(\mathbf{X}_*, \mathbf{X}_*) + k(\mathbf{X}_*, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} \left(\mathbf{S} - k(\mathbf{Z}, \mathbf{Z})\right) k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{X}_*)\Big). \qquad (2.36) \end{aligned}$$

The posterior distribution $q(\mathbf{f}_*)$ provides mean and variance for the test data points $\mathbf{X}_*$.

2.6 Bayesian Gaussian Process Latent Variable Model

In this section, we describe a Gaussian process model, Bayesian Gaussian process latent variable model (Bayesian GPLVM) (Titsias and Lawrence, 2010), to handle a dataset with only inputs. We assume there are latent variables in a low-dimensional space to represent the inputs. In Chapter 5, we build a multi-output Gaussian process model with the basic idea of latent variables. This section aims to show the idea of how Gaussian processes handle latent variables.

Suppose the observed inputs $\mathbf{X} \in \mathbf{R}^{N \times v}$ are associated with latent variables $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^N \in \mathbf{R}^{N \times Q_H}$ where $Q_H \ll v$. We assume there is a normal prior distribution over the latent variables

$$p(\mathbf{H}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{h}_n \mid \mathbf{0}, \mathbf{I}_{Q_H}), \qquad (2.37)$$

where $\mathbf{I}_{Q_H}$ is an identity matrix of size Q_H. The GPs are independently defined across the features in $\mathbf{X}$ so the likelihood factorises across feature dimensions:

$$p(\mathbf{X} \mid \mathbf{H}) = \prod_{j=1}^{v} p\left(\mathbf{x}^j \mid \mathbf{H}\right), \qquad (2.38)$$

where $\mathbf{x}^j$ refers to the j-th column of $\mathbf{X}$. As in all Bayesian models, we are interested in computing the posterior distribution $p(\mathbf{H} \mid \mathbf{X})$. However, the computation of the posterior distribution in closed form is intractable since the marginal likelihood $p(\mathbf{X}) = \int p(\mathbf{X} \mid \mathbf{H}) p(\mathbf{H}) \mathrm{d}\mathbf{H}$ cannot be computed in closed form. To deal with this problem, Titsias and

Lawrence (2010) provided a sparse GP formulation (Bayesian GPLVM), as in Section 2.4:

$$p\left(\mathbf{x}^j \mid \mathbf{H}, \mathbf{f}_j, \mathbf{u}_j\right) = p\left(\mathbf{x}^j \mid \mathbf{f}_j\right) p\left(\mathbf{f}_j \mid \mathbf{H}, \mathbf{u}_j, \mathbf{Z}\right) p\left(\mathbf{u}_j \mid \mathbf{Z}\right), \tag{2.39}$$

$$p\left(\mathbf{x}^j \mid \mathbf{f}_j\right) = \mathcal{N}\left(\mathbf{x}^j \mid \mathbf{f}_j, \sigma^2 \mathbf{I}_N\right), \tag{2.40}$$

$$p\left(\mathbf{f}_j \mid \mathbf{u}_j, \mathbf{H}, \mathbf{Z}\right) = \mathcal{N}(\mathbf{f}_j \mid k(\mathbf{H}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} \mathbf{u}_j, \tag{2.41}$$

$$k(\mathbf{H}, \mathbf{H}) - k(\mathbf{H}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{H})), \tag{2.42}$$

$$p\left(\mathbf{u}_j \mid \mathbf{Z}\right) = \mathcal{N}\left(\mathbf{u}_j \mid \mathbf{0}, k(\mathbf{Z}, \mathbf{Z})\right), \tag{2.43}$$

where each $\mathbf{x}^j$ is assumed equal to a corresponding vector of latent parameter function values $\mathbf{f}_j$ corrupted with a zero-mean Gaussian noise and σ^2 variance. For each $\mathbf{f}_j$, there is an inducing variable $\mathbf{u}_j$ evaluated at a set of inducing inputs $\mathbf{Z} \in \mathbf{R}^{M \times Q_H}$. As in Section 2.4, we need to approximate the true posteriors $p(\mathbf{f}_j, \mathbf{u}_j \mid \mathbf{x}^j, \mathbf{H}) = p\left(\mathbf{f}_j \mid \mathbf{u}_j, \mathbf{x}^j, \mathbf{H}\right) p\left(\mathbf{u}_j \mid \mathbf{x}^j, \mathbf{H}\right)$ and $p(\mathbf{H})$; the variational formulation is

$$q\left(\mathbf{f}_j, \mathbf{u}_j\right) = p\left(\mathbf{f}_j \mid \mathbf{u}_j, \mathbf{H}\right) q\left(\mathbf{u}_j\right), \tag{2.44}$$

$$q(\mathbf{H}) = \prod_{n=1}^{N} \mathcal{N}\left(\mathbf{h}_n \mid \boldsymbol{\mu}_n, \mathcal{S}_n\right), \tag{2.45}$$

where $\boldsymbol{\mu}_n$ and $\mathcal{S}_n$ are the variational parameters that follow the notation as in Titsias and Lawrence (2010). Those variational formulas bring a tractable lower bound for $p(\mathbf{X})$ (Titsias and Lawrence, 2010). Similar to Section 2.5, it is possible to derive an evidence lower bound of the log marginal distribution. Refer to Titsias and Lawrence (2010) for a detailed derivation of the lower bound. In Chapter 5, we apply latent variables to our model, an idea which originates from Bayesian GPLVM. Therefore, we explain the key part to derive latent variables in this section and we encourage the reader to refer to Titsias and Lawrence (2010) for more detail such as a predictive distribution.

2.7 Multi-output Gaussian Process Overview

Previous sections describe the background of Gaussian processes while this section focuses on multi-output Gaussian processes (MOGPs) (Álvarez et al., 2012). Nowadays, many model applications require solving several decision making or prediction problems to be solved simultaneously. The aim of multi-task learnings is to leverage useful information contained in multiple related tasks to improve the performance of all the tasks (Zhang and Yang, 2017). The MOGP is a kind of multi-task learning that can make several predictions at the same time. It manages to improve predictions of each output by using the relationship between several tasks (Álvarez et al., 2012; Bonilla et al., 2008). MOGP models exist in various areas such as geostatistics (Goovaerts et al., 1997), latent force models (Alvarez et al., 2009), and medicine (Alaa and Van Der Schaar, 2017). Many models exist, including the intrinsic coregionalisation model (ICM), the semiparametric latent factor model, the linear model of coregionalisation (LMC), and process convolutions. We refer the reader to Álvarez et al. (2012) for a comprehensive review of these models. Today, there are many advanced MOGPs models, including MOGPs with spectral mixture kernels (Parra and Tobar, 2017), MOGPs with heterogeneous outputs (Moreno-Muñoz et al., 2018), non-linear process convolutions (Álvarez et al., 2019), and nonstationary MOGPs (Altamirano and Tobar, 2022). This section briefly explains the LMC and ICM since our models are based on them.

The MOGP model is a generalisation of GPs, entailing a dependence on their own kernel function. Compared with single-output GPs, multi-output GPs have a multi-output function. This is known as a vector-valued function, with a corresponding reproducing kernel.

Suppose there is a multi-output dataset:

$$\mathbf{S} = \{S_d\}_{d=1}^{D} = \{\mathbb{X}, \mathbf{y}\} \tag{2.46}$$

where $\mathbb{X}$=$[\mathbf{X}_1^\top, \cdots, \mathbf{X}_D^\top]^\top$ and, for notation simplicity, we assume all different input datasets have the same input set and each dataset has N data points. For example, $\mathbf{X}_d = \mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^\top \in \mathbf{R}^{N\times v}$ for $d \in \{1, \cdots, D\}$; then $\mathbf{y}_d$=$[y_d(\mathbf{x}_1), \cdots, y_d(\mathbf{x}_N)]^\top \in \mathbf{R}^N$ is the d-th output dataset, $y_d(\mathbf{x}_n)$ is the n-th data point for the d-th output dataset and $\mathbf{y}$=$[\mathbf{y}_1^\top, \cdots, \mathbf{y}_D^\top]^\top$.

Each latent parameter function corresponds to each output. In MOGPs, those latent parameter functions are stacked into a vector-valued function $\mathbf{f}(\cdot) = \{f_d(\cdot)\}_{d=1}^{D}$ ($\mathbf{f}(\cdot) : \mathcal{X} \to \mathbf{R}^D$): $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \cdots, f_D(\mathbf{x})]^\top \in \mathbf{R}^D$, where $f_d(\mathbf{x})$ is a latent parameter function in the d-th output evaluated at $\mathbf{x}$. The vector-valued function $\mathbf{f}(\cdot)$ is assumed to follow a Gaussian process:

$$\mathbf{f}(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{m}(\mathbf{x}), \mathbf{K}(\mathbf{x}, \mathbf{x}')\right), \tag{2.47}$$

where $\mathbf{m}(\cdot)$=$\{m_d(\cdot)\}_{d=1}^{D}$ and $m_d(\cdot)$ is the mean function for d-th output. Those mean functions are usually assumed as $\mathbf{0}$. $\mathbf{K}(\cdot,\cdot)$ refers to a reproducing kernel and is also a matrix-valued function $\mathbf{K}(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbf{R}^{D\times D}$ where $\mathbf{K}(\mathbf{x}, \mathbf{x}') \in \mathbf{R}^{D\times D}$ is a positive semi-definite matrix and $(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'} \in \mathbf{R}$ expresses the correlation between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$ where $d, d' \in \{1, \cdots, D\}$.

2.7.1 Intrinsic Coregionalisation Model

This subsection describes one of the well-known MOGP models: the intrinsic coregionalisation model (ICM) (Álvarez et al., 2012), which is also known as the multitask GP model (Bonilla et al., 2008). ICM has a special multi-output kernel that can be divided into two parts: one part is for a kernel function to encode input space knowledge only; the other is another kernel to capture correlations among outputs.

The multi-output kernel can be written as

$$(\mathbf{K}(\mathbf{x},\mathbf{x}'))_{d,d'} = k_T(d,d') \times k(\mathbf{x},\mathbf{x}'), \tag{2.48}$$

where $k_T : \{1,2,\cdots,D\} \times \{1,2,\cdots,D\} \to \mathbf{R}$ and $k(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbf{R}$ are kernels for output space and input space respectively. Then,

$$\mathbf{K}(\mathbf{x},\mathbf{x}') = \mathbf{B} \times k(\mathbf{x},\mathbf{x}'), \tag{2.49}$$

with

$$\mathbf{B} = \begin{bmatrix} B_{1,1} & \dots & B_{1,D} \\ B_{2,1} & \dots & B_{2,D} \\ \vdots & \ddots & \vdots \\ B_{D,1} & \dots & B_{D,D} \end{bmatrix}, \tag{2.50}$$

where $\mathbf{B} \in \mathbf{R}^{D\times D}$ is a positive semi-definite matrix and $B_{d,d'} \in \mathbf{R}$ $(d,d' \in \{1,2\cdots,D\})$ describes the correlation between $f_d(\mathbf{x})$ and $f_{d'}(\mathbf{x}')$.

Similar to the single-output Gaussian process, there is also the prior distribution of the vector $\mathbf{f}$:

$$\mathbf{f} \sim \mathcal{N}\big(\mathbf{0},\mathbf{K}(\mathbf{X},\mathbf{X})\big), \tag{2.51}$$

where $\mathbf{f}$ is $\mathbf{f} = [\mathbf{f}_1,\cdots,\mathbf{f}_d,\cdots,\mathbf{f}_D]^\top$ is a vector function in which $\mathbf{f}_d = [f_d(\mathbf{x}_1),\cdots,f_d(\mathbf{x}_N)]^\top$ and $f_d(\mathbf{x}_i)$ is the function $f_d(\cdot)$ evaluated at $\mathbf{x}_i$; $\mathbf{K}(\mathbf{X},\mathbf{X})$ is a semi-positive definite matrix:

$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X},\mathbf{X}) \tag{2.52}$$

where $\otimes$ is Kronecker product and $k(\mathbf{X},\mathbf{X}) \in \mathbf{R}^{N\times N}$ is a positive semi-definite matrix.

For a regression problem, the likelihood function often takes a Gaussian distribution for each output, so

$$\mathbf{y}|\mathbf{f}(\mathbf{X}) \sim \mathcal{N}\big(\mathbf{f}(\mathbf{X}), \boldsymbol{\Sigma}\big), \tag{2.53}$$

where $\boldsymbol{\Sigma} = \Sigma \otimes \mathbf{I}_N$, Σ is a diagonal matrix with elements $\{\sigma_d^2\}_{d=1}^D$ in the diagonal and $\{\sigma_d^2\}_{d=1}^D$ are the variances of the noise for each output.

Similar to single-output Gaussian processes, based on Bayesian theorem and the properties of a Gaussian distribution, the predictive distribution for an input test point $\mathbf{x}_*$ in all outputs is

$$p\left(\mathbf{f}(\mathbf{x}_*)|\mathbf{y}\right) = \mathcal{N}(\mathbf{f}(\mathbf{x}_*)|\mathbf{m}_*, \, \mathbf{V}_*), \tag{2.54}$$

with

$$\mathbf{m}_* = \mathbf{K}^\top\left(\mathbf{x}_*, \mathbf{X}\right)\left(\mathbf{K}\left(\mathbf{X}, \mathbf{X}\right) + \boldsymbol{\Sigma}\right)^{-1}\mathbf{y}, \tag{2.55}$$

$$\mathbf{V}_* = \mathbf{K}\left(\mathbf{x}_*, \mathbf{x}_*\right) - \mathbf{K}\left(\mathbf{x}_*, \mathbf{X}\right)\left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \boldsymbol{\Sigma}\right)^{-1}\mathbf{K}^\top\left(\mathbf{x}_*, \mathbf{X}\right), \tag{2.56}$$

where $\mathbf{m}_* \in \mathbf{R}^D$ and $\mathbf{V}_* \in \mathbf{R}^{D\times D}$ are the mean and variance for the vector value function's predictive distribution evaluated at $\mathbf{x}_*$. $\mathbf{K}(\mathbf{x}_*, \mathbf{X}) \in \mathbf{R}^{D\times ND}$ has entries $(\mathbf{K}(\mathbf{x}_*, \mathbf{x}_i))_{d,d'}$ for $i \in \{1, \cdots, N\}$ and $d, d' \in \{1, \cdots, D\}$.

The ICM model also suffers from computational complexity and intractable posterior distributions. The ICM model demands a large computational complexity similar to the single GPs ($\mathcal{O}(N^3)$) but increased by the number of D outputs. Its computational complexity is $\mathcal{O}((ND)^3)$ and its storage is $\mathcal{O}((ND)^2)$. In terms of different likelihood functions, the posterior distribution is intractable. By using the knowledge of sparse GP and variational inference, we can speed up the computation in the MOGP to address the large computational complexity and intractable posterior distribution (Moreno-Muñoz et al., 2018), the detail of which is described in the next subsection.

2.7.2 Linear Model of Coregionalisation

The above subsection described ICM, a special case of the linear model of coregionalisation (LMC) (Álvarez et al., 2012). This subsection focuses on LMC.

Suppose there is a vector value function $\mathbf{f}(\cdot) = \{f_d(\cdot)\}_{d=1}^D$, where each output function is represented as a linear combination of a group of Gaussian processes, then each $f_d(\mathbf{x})$ is described as

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^{i} u_q^{i}(\mathbf{x}), \tag{2.57}$$

where $a_{d,q}^{i} \in \mathbf{R}$, $u_q^i(\mathbf{x})$ is an independent and identically distributed (i.i.d) sample from GPs $u_q(\cdot) \sim \mathcal{GP}\left(0, k_q(\cdot,\cdot)\right)$. $u_q(\cdot)$ refers to the q-th latent function in $\mathcal{U}(\cdot) = \{u_q(\cdot)\}_{q=1}^{Q}$ and $u_q(\cdot)$ is independent of each other. Suppose the mean function of any function $f_d(\mathbf{x})$ is zero and the cross-covariance between any two output functions is

$$\begin{aligned} k_{f_d f_{d'}}\left(\mathbf{x},\mathbf{x}'\right) =& \operatorname{cov}\left[f_d(\mathbf{x}), f_{d'}\left(\mathbf{x}'\right)\right] \\ =& \sum_{q=1}^{Q} \sum_{q'=1}^{Q} \sum_{i=1}^{R_q} \sum_{i'=1}^{R_q} a_{d,q}^{i} a_{d',q'}^{i'} \operatorname{cov}\left[u_q^i(\mathbf{x}), u_{q'}^{i'}(\mathbf{x}')\right], \end{aligned} \tag{2.58}$$

where $d, d' \in \{1, \cdots, D\}$ and $\operatorname{cov}[\cdot,\cdot]$ represents covariance operator. The $k_{f_d f_{d'}}\left(\mathbf{x},\mathbf{x}'\right)$ is the same as $\left(\mathbf{K}(\mathbf{x},\mathbf{x}')\right)_{d,d'}$ above. Because $u_q^i(\cdot)$ is i.i.d drawn from $u_q(\cdot)$ and $\mathcal{U}(\cdot)$ are mutually independent, the above function (2.58) can be represented as

$$\left(\mathbf{K}(\mathbf{x},\mathbf{x}')\right)_{d,d'} = k_{f_d f_{d'}}\left(\mathbf{x},\mathbf{x}'\right) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^{i} a_{d',q}^{i} k_q\left(\mathbf{x},\mathbf{x}'\right) = \sum_{q=1}^{Q} b_{d,d'}^{q} k_q\left(\mathbf{x},\mathbf{x}'\right), \tag{2.59}$$

where $b_{d,d'}^{q} = \sum_{i=1}^{R_q} a_{d,q}^{i} a_{d',q}^{i}$. Similar to Eq. (2.49), we can write $\mathbf{K}(\mathbf{x},\mathbf{x}')$ here as:

$$\mathbf{K}\left(\mathbf{x},\mathbf{x}'\right) = \sum_{q=1}^{Q} \mathbf{B}_q k_q\left(\mathbf{x},\mathbf{x}'\right), \tag{2.60}$$

where each $\mathbf{B}_q \in \mathbf{R}^{D\times D}$ is also a positive semi-definite matrix and is referred to as a coregionalisation matrix. When $Q = 1$, the above function is the same as expression (2.49). Since we assume all outputs have the same input dataset $\mathbf{X}$, the kernel matrix $\mathbf{K}(\mathbf{X},\mathbf{X})$ can be expressed as a sum of Kronecker products:

$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \sum_{q=1}^{Q} \mathbf{B}_q \otimes k_q\left(\mathbf{X},\mathbf{X}\right). \tag{2.61}$$

Following the same idea as ICM, we can obtain our predictive posterior distribution. For simplicity, we assume $R_q = 1$.

Similar to single-output Gaussian process, the multi-output Gaussian processes also suffers from computational complexity and has an intractable posterior distribution when dealing with a non-Gaussian likelihood function. Here we use the methodology of stochastic variational inference (SVI) to cope with these problems (Moreno-Muñoz et al., 2018), which is also implemented in Chapter 3. SVI reduces the computational complexity from $\mathcal{O}((ND)^3)$ to $\mathcal{O}\left(QM^3 + DNQM^2\right)$ and reduces the storage from $\mathcal{O}((ND)^2)$ to $\mathcal{O}\left(QM^2 + DNQM\right)$. It can also approximate the posterior distribution given different types of likelihoods so it can handle heterogeneous outputs, i.e., a mix of classification and regression outputs.

To apply SVI in MOGPs, we employ the inducing variable framework in Section 2.4 into a multi-output Gaussian process format. We define a group of inducing points $\mathbf{Z} = \{\mathbf{Z}_q\}_{q=1}^{Q}$ and each set of M inducing points $\mathbf{Z}_q = \left[\mathbf{z}_q^{(1)}, \cdots, \mathbf{z}_q^{(M)}\right]^\top \in \mathbf{R}^{M\times v}$ is associated to each latent function $u_q(\cdot)$. Each $\mathbf{u}_q = \left[u_q\left(\mathbf{z}_q^{(1)}\right), \cdots, u_q\left(\mathbf{z}_q^{(M)}\right)\right]^\top$ refers to $u_q(\cdot)$ evaluated at $\mathbf{Z}_q$. Then, we assume $\mathbf{u} = \left[\mathbf{u}_1^\top, \cdots, \mathbf{u}_Q^\top\right]^\top \in \mathbf{R}^{QM\times 1}$. Due to the mutual independence of $\mathcal{U}(\cdot)$, the distribution $p(\mathbf{u})$ is expressed as $p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{\mathbf{uu}}) = \prod_{q=1}^{Q} p\left(\mathbf{u}_q\right)$ and $\mathbf{u}_q \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_q\right)$, where in $\mathbf{K}_q \in \mathbf{R}^{M\times M}$ each entry is $k_q\left(\mathbf{z}_q^{(i)}, \mathbf{z}_q^{(j)}\right)$ with $\mathbf{z}_q^{(i)}, \mathbf{z}_q^{(j)} \in \mathbf{Z}_q$. $\mathbf{K}_{\mathbf{uu}}$ is a block-diagonal matrix with each diagonal block shown as $\mathbf{K}_q$. Suppose now $\mathbf{f}_d$ is $f_d(\mathbf{x}_i)$ evaluated at all $\mathbf{x}_i \in \mathbf{X}$, $\mathbf{f}_d = [f_d(\mathbf{x}_1), \cdots, f_d(\mathbf{x}_N)]^\top$ and assume $\mathbf{f} = \left[\mathbf{f}_1^\top, \cdots, \mathbf{f}_D^\top\right]^\top \in \mathbf{R}^{DN\times 1}$. Each output $\mathbf{f}_d$ are conditionally independent given $\mathbf{u}$.

Now, we assume the joint distribution between outputs $\mathbf{f}$ and inducing variables $\mathbf{u}$ as:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u}) = \prod_{d=1}^{D} p\left(\mathbf{f}_d \mid \mathbf{u}\right) \prod_{q=1}^{Q} q\left(\mathbf{u}_q\right), \tag{2.62}$$

where $p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{uf}}\right)$, $\mathbf{K}_{\mathbf{ff}}$ is the covariance matrix for the input $\mathbf{X}$ (see equation (2.61)) and $\mathbf{K}_{\mathbf{fu}}$ is the covariance matrix between the input $\mathbf{X}$ and inducing variables; $q\left(\mathbf{u}_q\right) = \mathcal{N}\left(\mathbf{u}_q \mid \boldsymbol{\mu}_{\mathbf{u}_q}, \mathbf{S}_{\mathbf{u}_q}\right)$ and $\left\{\boldsymbol{\mu}_{\mathbf{u}_q}, \mathbf{S}_{\mathbf{u}_q}\right\}_{q=1}^{Q}$ are considered as the variational parameters to be optimised.

The log-marginal likelihood is

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \int\int p(\mathbf{y}, \mathbf{f}, \mathbf{u})\, \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \log \int\int \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u})\, q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&\geq \int\int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \int\int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \mathcal{L},
\end{aligned} \tag{2.63}$$

where $\mathcal{L}$ is also called a variational lower bound. We can simplify $\mathcal{L}$ further:

$$\begin{aligned}
\mathcal{L} &= \int\int p(\mathbf{f}|(\mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}))}{p(\mathbf{f}|(\mathbf{u})q(\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \int\int \prod_{d=1}^{D} p(\mathbf{f}_d \mid \mathbf{u})\, q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{u}))}{q(\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \\
&= \int\int \prod_{d=1}^{D} p(\mathbf{f}_d \mid \mathbf{u})\, q(\mathbf{u})\log p(\mathbf{y}|\mathbf{f})\,\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} - \sum_{q=1}^{Q}\mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right) \\
&= \sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{f}_d)}\left[\log p\left(\mathbf{y}_d \mid \mathbf{f}_d\right)\right] - \sum_{q=1}^{Q}\mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right), \qquad (2.64)
\end{aligned}$$

where $\sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{f}_d)}\left[\log p\left(\mathbf{y}_d \mid \mathbf{f}_d\right)\right]$ is also valid across data points. Therefore, the lower bound can be expressed as

$$\mathcal{L} = \sum_{d=1}^{D}\sum_{n=1}^{N}\mathbb{E}_{q(\mathbf{f}_d)}\left[\log p\left(y_d(\mathbf{x}_n) \mid f_d(\mathbf{x}_n)\right)\right] - \sum_{q=1}^{Q}\mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right). \qquad (2.65)$$

By maximising $\mathcal{L}$, we can obtain the optimisation hyper-parameters in LMC. Further, since $\mathcal{L}$ factorises across data observations, it allows the use of a small number of mini-bath of samples to perform optimisation so that it reduces the computational complexity of LMC (Moreno-Muñoz et al., 2018).

To make a prediction, the approximate posterior distribution $q(\mathbf{f})$ is

$$\begin{aligned}
q(\mathbf{f}) &= \int p(\mathbf{f} \mid \mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u} \\
&= \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\boldsymbol{\mu}, \mathbf{K}_{\mathbf{ff}} + \mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\left(\mathbf{S}-\mathbf{K}_{\mathbf{uu}}\right)\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{fu}}^{\top}\right), \qquad (2.66)
\end{aligned}$$

where $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_{\mathbf{u}_1}^{\top}, \cdots, \boldsymbol{\mu}_{\mathbf{u}_Q}^{\top}\right]^{\top}$ and $\mathbf{S}$ is a block diagonal matrix with blocks given as $\mathbf{S}_{\mathbf{u}_q}$.

2.8 Summary

In this chapter, we review the background of Gaussian processes and multi-output Gaussian processes. First, the basics of the Gaussian process frameworks are described. For example, we introduce Gaussian processes for regression and classification problems and Gaussian processes with latent variables such as the Bayesian Gaussian process latent variable model. As the non-Gaussian likelihood leads to intractable computation and computational complexity limits the application of Gaussian processes, variational inference and inducing variables strategies are described. Second, the basic multi-output Gaussian process frameworks are explained. We introduce two traditional MOGP models: the intrinsic coregionalisation model (ICM) and the linear model of coregionalisation (LMC). Further, we describe the stochastic variational inference for MOGPs to deal with larger-scale datasets and heterogeneous outputs. In the following chapters, we take these base models or strategies as fundamental elements to build advanced multi-output Gaussian process models to deal with large-scale multi-class classification (Chapter 3), large scale multi-class image classification (Chapter 4) and hierarchical structure data regression (Chapter 5) problems.

Chapter 3

Multi-output Gaussian Processes for Large-scale Multi-class Classification

Multi-output Gaussian processes (MOGPs) can solve multiple problems concurrently and exploit the correlations not only with the input variables but also with output variables (Alvarez, 2011; Álvarez et al., 2012; Bonilla et al., 2008; Dahl and Bonilla, 2019; Nguyen et al., 2018; Wistuba et al., 2018). For example, in a sensor network, prediction of missing signals from some sensors may be done by exploiting dependencies with signals obtained from nearby sensors (Osborne et al., 2008).

Our main purpose in this chapter[1] is to use MOGPs to study the problem of multiple outputs where each output is a multi-class classification problem. The setting considered here goes beyond multi-label classification since we allow each output to potentially have its own inputs moving into the multi-task setting.

MOGPs have mainly been used for multi-output regression to predict continuous variables (Álvarez et al., 2012; Bonilla et al., 2008; Dai et al., 2017). In this setting, the assumption is that each output follows a Gaussian likelihood and the mean of the Gaussian likelihood is given by one output of the MOGP. Due to the properties of the Gaussian distribution, Bayesian inference is tractable in this case.

Beyond the muti-output regression problem, there is some research on other types of outputs in MOGPs. For example, Skolidis and Sanguinetti (2011) use MOGPs to model a setting where each output corresponds to a binary classification problem. Each binary outcome is modelled using a probit likelihood. The MOGP corresponds to the so called intrinsic coregionalisation model (ICM) (Bonilla et al., 2008). Since Bayesian inference is

[1]This chapter is based on **Ma, C.**, & Álvarez, M. A. (2023). Large scale multi-output multi-class classification using Gaussian processes. *Machine Learning*, 1-30.

intractable in this model, the authors approximate posterior distributions using expectation-propagation and variational Bayes.

Several research works have addressed the case of multi-class classification using GPs. Previous works have used the softmax likelihood (Galy-Fajou et al., 2020; Kim and Ghahramani, 2006; Williams and Rasmussen, 2006), the multinomial probit likelihood function (Girolami and Rogers, 2006) and the step function (Hernández-Lobato et al., 2011). Recently, Liu et al. (2019) have used all of the above likelihoods through additive noise terms. The parameters in these likelihood functions are assumed to follow independent Gaussian processes. Another strand of works generalise this setting by allowing correlated Gaussian processes for the latent parameters of the likelihood functions, typically using MOGPs. Both Dezfouli and Bonilla (2015) and Chai (2012) use an ICM for a single-output multi-class classification problem modelled through a multinomial logistic likelihood, i.e. the softmax likelihood. In terms of Bayesian inference, Chai (2012) proposes a variational sparse approximation for the posterior distribution, and based on scalable automated variational inference, Dezfouli and Bonilla (2015) approximate the posterior distribution using a mixture of Gaussians. Moreno-Muñoz et al. (2018) build a heterogeneous multi-output Gaussian process, where each output has its own likelihood, through a linear model of coregionalisation (LMC) (Álvarez et al., 2012). Moreno-Muñoz et al. (2018) use a stochastic variational approach for Bayesian inference.

The approaches for single-output multi-class classification described above are restricted to the case where the number of classes is small. They scale poorly when the number of classes go beyond a few tens. Scalability is also poorly handled by the more general model of Moreno-Muñoz et al. (2018) for the multi-output multi-class classification case, where once again, problems that go beyond a few tens of classes are out of reach.

Our main contribution in this chapter is that we introduce a new extension of multi-output GPs able to handle large scale multi-output multi-class classification problems, typically in the range of hundreds and even thousands of classes. We achieve scalability by subsampling both training input data and classes in each output, by using stochastic variational inference (Hensman et al., 2013a; Moreno-Muñoz et al., 2018) and by choosing a softmax likelihood function via Gumbel noise error for all outputs. We refer to this model as *multi-output Gaussian processes with augment & reduce (MOGPs-AR)*[2].

[2]The model is implemented based on Python 3.7, mainly depending on the library GPflow 2.1.3 which was built on the library TensorFlow 2+. Our code is publicly available in the repository https://github.com/ChunchaoPeter/MOGPs-AR.

3.1 Related Work

As mentioned earlier, the multi-class classification problem has mainly been studied using single-output GPs (Girolami and Rogers, 2006; Hernández-Lobato et al., 2011; Kim and Ghahramani, 2006; Liu et al., 2019; Williams and Rasmussen, 2006). The model introduced in this paper, MOGPs-AR, uses the softmax likelihood through additive noise errors, which is the same as Liu et al. (2019). However, MOGPs-AR solves multiple output problems together, while the model used by Liu et al. (2019), like all single-output GPs, only solves single-output problems. Regarding single-output problems, MOGPs-AR can also improve prediction using a correlation between all latent parameter functions whereas single-output GPs cannot capture the correlation.

The works more relevant to this study are Chai (2012); Dezfouli and Bonilla (2015); Moreno-Muñoz et al. (2018); Skolidis and Sanguinetti (2011). Both Chai (2012) and Dezfouli and Bonilla (2015) can only handle a single-output multi-class classification problem even if they use MOGPs. Nevertheless, our model can tackle multiple outputs where each output is a multi-class classification problem. Skolidis and Sanguinetti (2011) only solve multi-output binary classification problems, which is different to ours. Compared with Skolidis and Sanguinetti (2011), our inference method is also suited to large scale datasets. Moreno-Muñoz et al. (2018) can tackle multi-output multi-class classification problems and have developed a stochastic variational inference method similar to us. However, we are different to Moreno-Muñoz et al. (2018) since we can cope with a large number of classes by subsampling classes.

The work by Panos et al. (2021) is very relevant to us since we use a similar subsampling method. Panos et al. (2021) extend a semiparametric latent model, a special case of LMC, to address the multi-label problem by using sigmoidal/Bernoulli likelihood for each latent parameter function. Panos et al. (2021) can doubly subsample data points and classes to reduce computational complexity based on stochastic variational inference, which is analogous to us. However, our work is different in other aspects. First, we solve multi-class classification problems using the softmax likelihood instead of multi-label problems using sigmoidal/Bernoulli likelihood. Further, our model can deal with multi-output problems rather than only tackling single-output problems.

3.2 Methodology

In this section, we derive the MOGPs-AR model. We first define the LMC model. We then define the softmax likelihood through augmenting noise data. Finally, we describe stochastic variational inference and the approximated predictive distribution for our model.

We assume there are D different outputs. The vector $\mathbf{y}(\mathbf{x}) \in \mathbf{R}^D$ groups all the D different outputs:

$$\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \cdots, y_D(\mathbf{x})]^\top, \tag{3.1}$$

where $\mathbf{x} \in \mathbf{R}^v$. Each output $y_d(\mathbf{x}) \in \{1, \cdots, C_d\}(C_d \geq 2$ and $d \in \{1, \cdots, D\})$ is a categorical variable and C_d is the number of classes in the d-th output. Like Moreno-Muñoz et al. (2018), we assume that those outputs are conditionally independent given parameters $\boldsymbol{\theta}(\mathbf{x}) = [\theta_1(\mathbf{x}), \theta_2(\mathbf{x}), \cdots, \theta_D(\mathbf{x})]^\top$, where $\boldsymbol{\theta}(\mathbf{x})$ is defined by latent parameter functions:

$$\mathbf{f}(\mathbf{x}) = \left[f_1^1(\mathbf{x}), f_1^2(\mathbf{x}), \cdots f_1^{C_1}(\mathbf{x}), f_2^1(\mathbf{x}), f_2^2(\mathbf{x}), \cdots, f_D^{C_D}(\mathbf{x})\right]^\top \in \mathbf{R}^{C\times 1}, \tag{3.2}$$

where $C = \sum_{d=1}^D C_d$ and $f_d^c(\mathbf{x})$ is c-th latent parameter function in the d-th output evaluated at $\mathbf{x}$. We then obtain:

$$\begin{aligned} p(\mathbf{y}(\mathbf{x})|\boldsymbol{\theta}(\mathbf{x})) &= p(\mathbf{y}(\mathbf{x})|\mathbf{f}(\mathbf{x})) \\ &= \prod_{d=1}^D p\left(y_d(\mathbf{x})|\boldsymbol{\theta}_d(\mathbf{x})\right) \\ &= \prod_{d=1}^D p\left(y_d(\mathbf{x})|\widetilde{\mathbf{f}}_d(\mathbf{x})\right), \end{aligned} \tag{3.3}$$

where $\widetilde{\mathbf{f}}_d(\mathbf{x}) = \left[f_d^1(\mathbf{x}), \cdots, f_d^{C_d}(\mathbf{x})\right]^\top \in \mathbf{R}^{C_d\times 1}$ is a group of latent parameter functions defining the parameters in $\boldsymbol{\theta}_d(\mathbf{x})$.

3.2.1 Linear Model of Coregionalisation

We use the linear model of coregionalisation (LMC). The LMC is a popular model in MOGPs, where each output is expressed as a linear combination of a collection of Gaussian processes (Álvarez et al., 2012).

We set up mutually independent latent functions $\mathcal{U}(\cdot) = \{u_q(\cdot)\}_{q=1}^Q$ where $u_q(\cdot)$ follows a Gaussian process and each latent parameter function $f_d^c(\cdot)$ is a linear combination of the latent functions $\mathcal{U}(\cdot)$. Each function $u_q(\cdot)$ is drawn from an independent GP prior: $u_q(\cdot) \sim \mathcal{GP}\left(0, k_q(\cdot,\cdot)\right)$, where $k_q(\cdot,\cdot)$ can be any kernel function. In this chapter, we use the radial basis function kernel with automatic relevance determination (RBF-ARD) (Williams and Rasmussen, 2006):

$$k_{\text{ard}}\left(\mathbf{x},\mathbf{x}'\right)=\sigma_{\text{ard}}^2\exp\left(-\frac{1}{2}\sum_{j=1}^{v}\frac{\left(x_j-x_j'\right)^2}{l_j^2}\right), \tag{3.4}$$

where x_j is the j-th dimension of $\mathbf{x}$, σ_{ard}^2 is a variance parameter and l_j is the length scale for the j-th input dimension. When all length scales are the same, the kernel is called radial basis function kernel (RBF) (Lawrence and Hyvärinen, 2005). Hence, each $f_d^c(\mathbf{x})$ is defined as

$$f_d^c(\mathbf{x})=\sum_{q=1}^{Q}\sum_{i=1}^{R_q}a_{d,c,q}^i u_q^i(\mathbf{x}), \tag{3.5}$$

where $a_{d,c,q}^i\in\mathbf{R}$ can be considered as a weight on $\mathcal{U}$ and we assume $\{\phi_q\}_{q=1}^Q$ are the hyperparameters for $\{k_q(\cdot,\cdot)\}_{q=1}^Q$. R_q represents the number of latent functions $u_q^i(\mathbf{x})$ that are sampled independently and identically from the Gaussian processes $u_q(\cdot)\sim\mathcal{GP}\left(0,k_q(\cdot,\cdot)\right)$. With $q=1,...,Q$ and $i=1,...,R_q$, the function $u_q^i(\mathbf{x})$ have a zero mean and covariance $\text{cov}\left[u_q^i(\mathbf{x}),u_{q'}^{i'}(\mathbf{x}')\right]=k_q\left(\mathbf{x},\mathbf{x}'\right)$ if $i=i'$ and $q=q'$. Let the mean function of $f_d^c(\mathbf{x})$ be zero and the cross-covariance function of $f_d^c(\mathbf{x})$ be

$$\begin{aligned}k_{f_d^c f_{d'}^{c'}}\left(\mathbf{x},\mathbf{x}'\right)&=\text{cov}\left[f_d^c(\mathbf{x}),f_{d'}^{c'}\left(\mathbf{x}'\right)\right]\\&=\text{cov}\left[\sum_{q=1}^{Q}\sum_{i=1}^{R_q}a_{d,c,q}^i u_q^i(\mathbf{x}),\sum_{q'=1}^{Q}\sum_{i'=1}^{R_q}a_{d',c',q'}^{i'}u_{q'}^{i'}(\mathbf{x}')\right]\\&=\sum_{q=1}^{Q}\sum_{q'=1}^{Q}\sum_{i=1}^{R_q}\sum_{i'=1}^{R_q}a_{d,c,q}^i a_{d',c',q'}^{i'}\,\text{cov}\left[u_q^{i'}(\mathbf{x}),u_{q'}^{i'}(\mathbf{x}')\right].\end{aligned} \tag{3.6}$$

Because $u_q^i(\cdot)$ is independently and identically drawn from $u_q(\cdot)$ and $\mathcal{U}(\cdot)$ are mutually independent

$$k_{f_d^c f_{d'}^{c'}}\left(\mathbf{x},\mathbf{x}'\right)=\sum_{q=1}^{Q}b_{(d,c),(d',c')}^q k_q\left(\mathbf{x},\mathbf{x}'\right), \tag{3.7}$$

where $b_{(d,c),(c',c')}^q=\sum_{i=1}^{R_q}a_{d,c,q}^i a_{d',c',q}^i$. For simplicity in the presentation, we assume that all outputs $y_d(\mathbf{x})$ have a collection of the same input vectors $\mathbf{X}=\{\mathbf{x}_n\}_{n=1}^N\in\mathbf{R}^{N\times v}$. Our model also works for each output with a different input dataset. Then $\mathbf{y}_d$=$[y_d(\mathbf{x}_1),\cdots,y_d(\mathbf{x}_N)]^\top\in\mathbf{R}^N$ is the d-output dataset and $\mathbf{y}$=$[\mathbf{y}_1^\top,\cdots,\mathbf{y}_D^\top]^\top$ is the dataset for all outputs. For notation simplicity, we define

$$\mathbf{f}_d^c = \left[f_d^c(\mathbf{x}_1), \cdots, f_d^c(\mathbf{x}_N)\right]^\top \in \mathbf{R}^{N\times 1} \tag{3.8}$$

$$\widetilde{\mathbf{f}}_d = \left[\left(\mathbf{f}_d^1\right)^\top, \cdots, \left(\mathbf{f}_d^{C_d}\right)^\top\right]^\top \in \mathbf{R}^{C_d N\times 1} \tag{3.9}$$

$$\mathbf{f} = \left[\widetilde{\mathbf{f}}_1^\top, \cdots, \widetilde{\mathbf{f}}_D^\top\right]^\top \in \mathbf{R}^{CN\times 1} \tag{3.10}$$

The prior distribution of $\mathbf{f}$ is given by $\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ is a block-wise matrix based on $\left\{\mathbf{K}_{\mathbf{f}_d^c \mathbf{f}_{d'}^{c'}}\right\}_{d=1,d'=1,c=1,c'=1}^{D,D,C_d,C_{d'}}$ as each block and $\mathbf{K}_{\mathbf{f}_d^c \mathbf{f}_{d'}^{c'}}$ has entries computed using $k_{f_d^c f_{d'}^{c'}}(\mathbf{x}_n, \mathbf{x}_m)$ with $\mathbf{x}_n, \mathbf{x}_m \in \mathbf{X}$. $\mathbf{K}$ can be formulated as a sum of Kronecker products $\mathbf{K} = \sum_{q=1}^{Q} \mathbf{A}_q \mathbf{A}_q^\top \otimes \mathbf{K}_q = \sum_{q=1}^{Q} \mathbf{B}_q \otimes \mathbf{K}_q$ as well, where $\mathbf{A}_q \in \mathbf{R}^{C\times R_q}$ and $\mathbf{B}_q$ have entries $\left\{a_{d,c,q}^i\right\}_{d=1,c=1,i=1}^{D,C_d,R_q}$ and $\left\{b_{(d,c),(d',c')}^q\right\}_{d=1,d'=1,c=1,c'=1}^{D,D,C_d,C_{d'}}$, respectively. $\mathbf{K}_q \in \mathbf{R}^{N\times N}$ has entries computed using $k_q(\mathbf{x}_n, \mathbf{x}_m)$ for $\mathbf{x}_n, \mathbf{x}_m \in \mathbf{X}$. Each matrix $\mathbf{B}_q \in \mathbf{R}^{C\times C}$ is known as a *coregionalisation matrix* and it controls the correlation between each latent parameter function.

3.2.2 Augmenting Model by Noise Data

In this section, we generalise the model in the last subsection to cope with the multi-output multi-class classification problem using the softmax likelihood. We derive a softmax likelihood function through Gumbel noise error for a generic output $\mathbf{y}_d$.

We take the d-th output $y_d(\mathbf{x})$ with the latent parameter function $\widetilde{\mathbf{f}}_d(\mathbf{x}) = [f_d^1(\mathbf{x}), f_d^2(\mathbf{x}), \cdots, f_d^{C_d}(\mathbf{x})]^\top$. We first add a Gumbel noise error to each latent parameter function in $\widetilde{\mathbf{f}}_d(\mathbf{x})$ to get a new vector function $\mathbf{h}_d(\mathbf{x}) = \left\{h_d^c(\mathbf{x})\right\}_{c=1}^{C_d}$ for each of the classes in the d-th output. We thus obtain:

$$h_d^c(\mathbf{x}) = f_d^c(\mathbf{x}) + \varepsilon_{d,i}^c, \tag{3.11}$$

$$y_d(\mathbf{x}) = \underset{c}{\operatorname{argmax}}\, h_d^c(\mathbf{x}), \tag{3.12}$$

where $\varepsilon_{d,i}^c$ is the i-th i.i.d. Gumbel noise error for class c in the d-th output. We define $\mathbf{h}_d(\mathbf{x}_i) = \left(h_d^1(\mathbf{x}_i), \cdots, h_d^{C_d}(\mathbf{x}_i)\right)^\top \in \mathbf{R}^{C_d\times 1}$. We then employ the internal step likelihood (Liu et al., 2019):

$$
\begin{aligned}
p\left(y_d\left(\mathbf{x}_i\right)\left|\mathbf{h}_d(\mathbf{x}_i)\right.\right) &= \prod_{c \neq y_d(\mathbf{x}_i)} H\left(h_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right) - h_d^c\left(\mathbf{x}_i\right)\right) \\
&= \prod_{c \neq y_d(\mathbf{x}_i)} H\left(f_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right) + \varepsilon_{d,i}^{y_d(\mathbf{x}_i)} - f_d^c\left(\mathbf{x}_i\right) - \varepsilon_{d,i}^c\right),
\end{aligned}
\tag{3.13}
$$

where $H(z) = 1$ when $z > 0$; otherwise, $H(z) = 0$; $y_d\left(\mathbf{x}_i\right)$ is i-th point in d-th output. By integrating $\mathbf{h}_d(\mathbf{x}_i)$ out, we get

$$
\begin{aligned}
p\left(y_d\left(\mathbf{x}_i\right)\left|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right.\right) &= \int p\left(y_d\left(\mathbf{x}_i\right)\left|\mathbf{h}_d(\mathbf{x}_i)\right.\right) p\left(\mathbf{h}_d(\mathbf{x}_i)\left|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right.\right) \mathrm{d}\mathbf{h}_d(\mathbf{x}_i) \\
&= \int \phi_{\mathcal{G}}\left(\varepsilon_{d,i}\right) \prod_{c \neq y_d(\mathbf{x}_i)} \Phi_{\mathcal{G}}\left(\varepsilon_{d,i} + f_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right) - f_d^c\left(\mathbf{x}_i\right)\right) \mathrm{d}\varepsilon_{d,i},
\end{aligned}
\tag{3.14}
$$

where $\phi_{\mathcal{G}}(\cdot)$ and $\Phi_{\mathcal{G}}(\cdot)$ are the probability density function and the cumulative distribution function of the Gumbel distribution, respectively. (NB: We drop out the c in $\varepsilon_{d,i}^c$ for convenience since all the $\varepsilon_{d,i}^c$ are from the same Gumbel error distribution). Now, we assume the Gumbel error $\varepsilon_{d,i} \sim \phi_{\mathcal{G}}(\varepsilon_{d,i}|0,1)$ so we obtain $\phi_{\mathcal{G}}(\varepsilon_{d,i}) = \exp\left(-\varepsilon_{d,i} - e^{-\varepsilon_{d,i}}\right)$ and $\Phi_{\mathcal{G}}(\varepsilon_{d,i}) = \exp\left(-e^{-\varepsilon_{d,i}}\right)$. Eq. (3.14) can be obtained by eliminating $\varepsilon_{d,i}$ from Eq. (3.15) through marginalization. The log-joint of Eq. (3.15), involving a sum over possible classes c, allows unbiased estimates of the log probability and its gradient (Ruiz et al., 2018).

$$
p\left(y_d\left(\mathbf{x}_i\right), \varepsilon_{d,i}\left|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right.\right) = \phi_{\mathcal{G}}\left(\varepsilon_{d,i}\right) \prod_{c \neq y_d(\mathbf{x}_i)} \Phi_{\mathcal{G}}\left(\varepsilon_{d,i} + f_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right) - f_d^c\left(\mathbf{x}_i\right)\right).
\tag{3.15}
$$

From expression (3.14), we recover a softmax likelihood (Liu et al., 2019; Ruiz et al., 2018):

$$
p\left(y_d\left(\mathbf{x}_i\right)\left|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right.\right) = \frac{\exp\left(f_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right)\right)}{\sum_{c=1}^{C_d} \exp\left(f_d^c\left(\mathbf{x}_i\right)\right)}.
\tag{3.16}
$$

The softmax likelihood is a common likelihood used in multi-class classification with Gaussian processes (Williams and Rasmussen, 2006). As mentioned in expression (4.3), all outputs are conditionally independent given the corresponding latent parameter functions so each output has its own likelihood expression (3.14).

Since our work shares similarities with the works of Liu et al. (2019) and Ruiz et al. (2018), we would like to emphasize the differences between our work and theirs here. Ruiz et al. (2018) employed stochastic subsampling over both training data sets and classes for a single output problem. Our approach is similar in that we also subsample both training data and classes. To some extent, we consider that we generalise a parametric model from Ruiz et al. (2018) to our non-parametric model. MOGPs-AR uses the softmax likelihood with additive noise errors, which is the same as Liu et al. (2019). However, MOGPs-AR solves multiple output problems simultaneously, whereas the model used by Liu et al. (2019) only solves single-output problems and does not apply stochastic subsampling over classes.

3.2.3 Scalable Variational Inference

We have derived the LMC model and used the softmax likelihood. However, a computational challenge exists if there are a very large number of classes and training instances in each output. We thus use scalable variational inference to reduce the computational complexity using the techniques of inducing points and subsampling, where we refer our model to *multi-output Gaussian processes with augment & reduce (MOGPs-AR)*. Inducing points can ease the computational complexity of the inversion of a kernel matrix from $\mathcal{O}\left(N^3\right)$ to $\mathcal{O}\left(NM^2\right)$, where N is the number of data points per output and M is the number of inducing points ($M \ll N$). Subsampling reduces the computational complexity of our model using a subset of both training data and classes for each output during hyperparameter and parameter optimisation.

3.2.3.1 Inducing Points for MOGPs-AR

We first define a group of inducing points $\mathbf{Z} = \{\mathbf{Z}_1\}_{q=1}^{Q}$ and each set has a collection of M inducing points (Hensman et al., 2013a) $\mathbf{Z}_q = \left\{\mathbf{z}_q^m\right\}_{m=1}^{M} \in \mathbf{R}^{M\times v}$ for each latent function u_q. We then obtain $\mathbf{u}_q = \left[u_q\left(\mathbf{z}_q^{(1)}\right), \cdots, u_q\left(\mathbf{z}_q^{(M)}\right)\right]^\top$ evaluated at $\mathbf{Z}_q$. We assume $\mathbf{u} = \left[\mathbf{u}_1^\top, \cdots, \mathbf{u}_Q^\top\right]^\top \in \mathbf{R}^{QM\times 1}$. Since the latent functions $\mathcal{U}(\cdot) = \left\{u_q(\cdot)\right\}_{q=1}^{Q}$ are mutually independent, the distribution $p(\mathbf{u})$ factorises as $p(\mathbf{u}) = \prod_{q=1}^{Q} p\left(\mathbf{u}_q\right)$, with $\mathbf{u}_q \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_q\right)$, where $\mathbf{K}_q \in \mathbf{R}^{M\times M}$ has entries given by $k_q\left(\mathbf{z}_q^{(i)}, \mathbf{z}_q^{(j)}\right)$ with $\mathbf{z}_q^{(i)}, \mathbf{z}_q^{(j)} \in \mathbf{Z}_q$. The latent parameter functions $\mathbf{f}_d^c$ are conditionally independent given $\mathbf{u}$. We therefore obtain the conditional distribution $p(\mathbf{f}|\mathbf{u})$

$$\begin{aligned} p(\mathbf{f}|\mathbf{u}) &= \prod_{d=1}^{D}\prod_{c=1}^{C_d} p\left(\mathbf{f}_d^c|\mathbf{u}\right) \\ &= \prod_{d=1}^{D}\prod_{c=1}^{C_d} \mathcal{N}\left(\mathbf{f}_d^c|\mathbf{K}_{\mathbf{f}_d^c\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{f}_d^c\mathbf{f}_d^c} - \mathbf{K}_{\mathbf{f}_d^c\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{f}_d^c\mathbf{u}}^\top\right), \end{aligned} \tag{3.17}$$

where $\mathbf{K}_{\mathbf{uu}} \in \mathbf{R}^{QM \times QM}$ is a block-diagonal matrix based on $\mathbf{K}_q$ as each block.

Based on Liu et al. (2019); Moreno-Muñoz et al. (2018), we obtain the lower bound $\mathcal{L}$ for $\log p(\mathbf{y})$:

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \int\int\int p(\mathbf{y}, \varepsilon, \mathbf{f}, \mathbf{u})\, \mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
&= \log \int\int\int \frac{p(\mathbf{y}, \varepsilon, \mathbf{f}, \mathbf{u})\, q(\mathbf{f}, \mathbf{u}, \varepsilon)}{q(\mathbf{f}, \mathbf{u}, \varepsilon)} \mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
&\geq \int\int\int q(\mathbf{f}, \mathbf{u}, \varepsilon) \log \frac{p(\mathbf{y}, \varepsilon, \mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u}, \varepsilon)} \mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
&= \int\int\int q(\varepsilon|\mathbf{f}, \mathbf{u}) q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \varepsilon, \mathbf{f}, \mathbf{u})}{q(\varepsilon|\mathbf{f}, \mathbf{u}) q(\mathbf{f}, \mathbf{u})} \mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
&= \mathcal{L},
\end{aligned} \tag{3.18}$$

where

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u}) = \prod_{d=1}^{D} \prod_{c=1}^{C_d} p(\mathbf{f}_d^c \mid \mathbf{u}) \prod_{q=1}^{Q} q(\mathbf{u}_q),$$

where $q(\mathbf{u}_q) = \mathcal{N}\left(\mathbf{u}_q \mid \boldsymbol{\mu}_{\mathbf{u}_q}, \mathbf{S}_{\mathbf{u}_q}\right)$ and we refer $\left\{\boldsymbol{\mu}_{\mathbf{u}_q}, \mathbf{S}_{\mathbf{u}_q}\right\}_{q=1}^{Q}$ as the variational hyperparameters that need to be optimised. Further, we get (see Appendix B for detail):

$$\begin{aligned}
\mathcal{L} =& \sum_{d}^{D} \sum_{i}^{N} \left\langle \log p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right) \right\rangle_{q(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)) q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)} \\
& - \sum_{q=1}^{Q} \mathrm{KL}\left(q(\mathbf{u}_q) \| p(\mathbf{u}_q)\right) - \sum_{i=d}^{D} \sum_{i=1}^{N} \mathrm{KL}\left(q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \| p(\varepsilon_{d,i})\right),
\end{aligned} \tag{3.19}$$

where $q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)$ approximates the posterior $p\left(\varepsilon_{d,i} | y_d(\mathbf{x}_i), \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)$:

$$p\left(\varepsilon_{d,i} | y_d(\mathbf{x}_i), \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \propto p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right) p(\varepsilon_{d,i}), \tag{3.20}$$

$$\begin{aligned}
p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right) &= \prod_{c \neq y_d(\mathbf{x}_i)} \Phi_{\mathcal{G}}\left(\varepsilon_{d,i} + f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i) - f_d^c(\mathbf{x}_i)\right), \\
&= \prod_{c \neq y_d(\mathbf{x}_i)} \exp\left(-e^{-\varepsilon_{d,i} - f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i) + f_d^c(\mathbf{x}_i)}\right).
\end{aligned} \tag{3.21}$$

$p\left(\varepsilon_{d,i}\right)$ is a probability density function of the standard Gumbel distribution. $q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)$ approximates the marginal posterior for $\widetilde{\mathbf{f}}_d(\mathbf{x}_i)$:

$$\begin{aligned}
q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) =& \int p\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)|\mathbf{u}\right) q(\mathbf{u})\mathrm{d}\mathbf{u} \\
=& \mathcal{N}\Big(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)|\mathbf{K}_{\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\boldsymbol{\mu}_{\mathbf{u}}, \\
& \mathbf{K}_{\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\widetilde{\mathbf{f}}_d(\mathbf{x}_i)} + \mathbf{K}_{\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\left(\mathbf{S}_{\mathbf{u}} - \mathbf{K}_{\mathbf{uu}}\right)\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\mathbf{u}}^{\top}\Big),
\end{aligned} \tag{3.22}$$

where $\boldsymbol{\mu}_{\mathbf{u}} = \left[\boldsymbol{\mu}_{\mathbf{u}_1}^{\top}, \cdots, \boldsymbol{\mu}_{\mathbf{u}_Q}^{\top}\right]^{\top}$ and $\mathbf{S}_{\mathbf{u}}$ is a block diagonal matrix with blocks given as $\mathbf{S}_{\mathbf{u}_q}$. After calculation (NB: detail can be found in Appendix B), we obtain:

$$\mathcal{L} = -\sum_{d=1}^{D}\sum_{i=1}^{N} \log\left(\mathcal{P}_{d,i} + 1\right) - \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right) \| p\left(\mathbf{u}_q\right)\right), \tag{3.23}$$

where

$$\mathcal{P}_{d,i} = \exp\left(\frac{\nu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)}{2} - \mu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)\right) \sum_{c \neq y_d(\mathbf{x}_i)} \exp\left(\frac{\nu_{f_d^c}(\mathbf{x}_i)}{2} + \mu_{f_d^c}(\mathbf{x}_i)\right), \tag{3.24}$$

where $\mu_{f_d^c}(\mathbf{x}_i)$ and $\nu_{f_d^c}(\mathbf{x}_i)$ are the mean and variance of $q\left(f_d^c(\mathbf{x}_i)\right)$, respectively.

3.2.3.2 Reducing Computational Complexity by Subsampling

To reduce the computational complexity of our model, we use only a subset of the data observations and a subset of the classes to estimate the parameters and hyperparameters. The optimal parameters and hyperparameters are chosen by maximising an unbiased estimator of $\mathcal{L}$ (3.23), where the unbiased estimator is obtained through a subset of both training data points (one sum: $\sum_{i=1}^{N}$ in $\mathcal{L}$) and classes in each output (another sum: $\sum_{c \neq y_d(\mathbf{x}_i)}$ in $\mathcal{L}$).

In our model, the hyperparameters are $\mathbf{Z}$, $\left\{a_{c,q}\right\}_{c=1,q=1}^{C,Q}$, $\{\phi_q\}_{q=1}^{Q}$, $\{w_p\}_{p=1}^{P}$ and the variational parameters are $\left\{\mathbf{u}_q, \mathbf{S}_{\mathbf{u}_q}\right\}_{q=1}^{Q}$ for $\{q\left(\mathbf{u}_q\right)\}_{q=1}^{q=Q}$. We refer to all those parameters as Θ. To obtain the optimal values of Θ, we use a gradient descent to maximise $\mathcal{L}$ with respect to Θ:

$$\begin{aligned}
\nabla_\Theta \mathcal{L} =& -\nabla_\Theta \sum_{d=1}^{D}\sum_{i=1}^{N} \log\left(\mathcal{P}_{d,i}+1\right) - \nabla_\Theta \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right) \\
=& -\nabla_\Theta \sum_{d=1}^{D}\sum_{i=1}^{N} \log\left(\left(\psi_{y_d(\mathbf{x}_i)} \sum_{c\neq y_d(\mathbf{x}_i)} \gamma_{d,c(\mathbf{x}_i)}\right)+1\right) \\
& -\nabla_\Theta \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right), \qquad (3.25)
\end{aligned}$$

where, for notation simplicity, we define

$$\psi_{y_d(\mathbf{x}_i)} = \exp\left(\frac{\nu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)}{2} - \mu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)\right), \qquad (3.26)$$

$$\gamma_{d,c(\mathbf{x}_i)} = \exp\left(\frac{\nu_{f_d^c}(\mathbf{x}_i)}{2} + \mu_{f_d^c}(\mathbf{x}_i)\right). \qquad (3.27)$$

We then estimate $\nabla_\Theta \mathcal{L}$ by randomly sampling a subset of data points $\mathcal{B} = \{\mathbf{b}_1, ..., \mathbf{b}_{|\mathcal{B}|}\} \subseteq \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ of size $|\mathcal{B}|$ and a subset of classes $\mathcal{S} = \{s_1, ..., s_{|\mathcal{S}|}\} \subseteq \{1, \ldots, C_d\} \backslash \{y_d(\mathbf{x})\}$ with size $|\mathcal{S}|$ ("$\backslash$" means $\{y_d(\mathbf{x})\}$ is excluded from $\{1, \ldots, C_d\}$) for each multi-class classification output:

$$\begin{aligned}
\nabla_\Theta \widetilde{\mathcal{L}} =& -\nabla_\Theta \sum_{d=1}^{D}\sum_{\mathbf{x}_i\in\mathcal{B}} \frac{N}{|\mathcal{B}|} \log\left(\frac{K-1}{|\mathcal{S}|}\left(\psi_{y_d(\mathbf{x}_i)} \sum_{c\in\mathcal{S}} \gamma_{d,c(\mathbf{x}_i)}\right)+1\right) \\
& -\nabla_\Theta \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right). \qquad (3.28)
\end{aligned}$$

$\nabla_\Theta \widetilde{\mathcal{L}}$ is an unbiased estimator for $\nabla_\Theta \mathcal{L}$ where the computational complexity of MOGPs-AR is dominated by optimising the parameters by maximising $\mathcal{L}$.

Our sampling strategy is in Algorithm 1. The computational complexity of MOGPs-AR mainly depends on the inversion of $\mathbf{K}_{\mathbf{uu}}$ with complexity $\mathcal{O}(QM^3)$ and products like $\mathbf{K}_{\tilde{\mathbf{f}}\mathbf{u}}$ with complexity $\mathcal{O}\left(D|\mathcal{S}||\mathcal{B}|QM^2\right)$; if we do not use the subsampling of classes, we have to calculate products like $\mathbf{K}_{\bar{\mathbf{f}}\mathbf{u}}$ with a cost of $\mathcal{O}\left(C|\mathcal{B}|QM^2\right)$, where the notations are defined as below:

$$\widetilde{\mathbf{f}} = \left[\widetilde{\mathbf{f}}_{1,\mathcal{B}}^{\top}, \cdots, \widetilde{\mathbf{f}}_{D,\mathcal{B}}^{\top}\right]^{\top} \in \mathbf{R}^{D|\mathcal{S}||\mathcal{B}|\times 1} \tag{3.29}$$

$$\bar{\mathbf{f}} = \left[\bar{\mathbf{f}}_{1,\mathcal{B}}^{\top}, \cdots, \bar{\mathbf{f}}_{D,\mathcal{B}}^{\top}\right]^{\top} \in \mathbf{R}^{C|\mathcal{B}|\times 1} \tag{3.30}$$

$$\widetilde{\mathbf{f}}_{d,\mathcal{B}} = \left[\mathbf{f}_{d,\mathcal{B}}^{\mathcal{S}_1}, \cdots, \mathbf{f}_{d,\mathcal{B}}^{\mathcal{S}_{|\mathcal{S}|}}\right]^{\top} \in \mathbf{R}^{|\mathcal{S}||\mathcal{B}|\times 1} \tag{3.31}$$

$$\bar{\mathbf{f}}_{d,\mathcal{B}} = \left[\mathbf{f}_{d,\mathcal{B}}^{1}, \cdots, \mathbf{f}_{d,\mathcal{B}}^{C_d}\right]^{\top} \in \mathbf{R}^{C_d|\mathcal{B}|\times 1} \tag{3.32}$$

$$\mathbf{f}_{d,\mathcal{B}}^{c} = \left[f_d^c(\mathbf{b}_1), \cdots, f_d^c(\mathbf{b}_{|\mathcal{B}|})\right]^{\top} \in \mathbf{R}^{|\mathcal{B}|\times 1}. \tag{3.33}$$

We observe that $D|\mathcal{S}| \ll C$ ($C = \sum_{d=1}^{D} C_d$) so MOGPs-AR alleviates the computational complexity of the product $\mathbf{K}_{\widetilde{\mathbf{f}}\mathbf{u}}$ from $\mathcal{O}\left(C|\mathcal{B}|QM^2\right)$ to $\mathcal{O}\left(D|\mathcal{S}||\mathcal{B}|QM^2\right)$.

Algorithm 1 Sampling strategy

```
1:  **Input:** data ($\mathbf{X}$, $\mathbf{y}$), minibatch sizes $|\mathcal{B}|$ and $|\mathcal{S}|$
2:  **Output:** parameters $\Theta$
3:  Initialise all parameters and hyperparameters
4:  **for** iteration $t = 1, 2, 3,...,$until convergence **do**
5:      *# Sampling minibatches*
6:      **Sample a minibatch of data,** $\mathcal{B} \subseteq \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
7:      **for** $i \in \mathcal{B}$ **do**
8:          *# Sampling Classes*
9:          **Sample a set of classes,** $\mathcal{S}_{d,i} \subseteq \{1, \dots, C_d\} \setminus \{y_d(\mathbf{x}_i)\}$, $d \in \{1,...,D\}$
10:         Compute $q\left(f_d^c(\mathbf{x}_i)\right)$ , where $c \in \mathcal{S}_{d,i} \cup \{y_d(\mathbf{x}_i)\}$, $d \in \{1,...,D\}$ and $|\mathcal{S}| = |\mathcal{S}_{d,i}| + 1$
11:     **end for**
12:     Gradient step on parameters: $\Theta \leftarrow \Theta + \alpha \nabla_{\Theta} \widetilde{\mathcal{L}}$
13: **end for**
```

3.2.4 Prediction of MOGPs-AR

In this subsection, we derive the predictive distribution of MOGPs-AR. Considering a new test input $\mathbf{x}_*$ in the d-th output, we assume $p(\mathbf{u}|\mathbf{y}) \approx q(\mathbf{u})$ and approximate the predictive distribution $p\left(y_d\left(\mathbf{x}_*\right)\right)$ by

$$p\left(y_d\left(\mathbf{x}_*\right)|\mathbf{y}\right) \approx \int p\left(y_d\left(\mathbf{x}_*\right) \mid \widetilde{\mathbf{f}}_d\left(\mathbf{x}_*\right)\right) q\left(\widetilde{\mathbf{f}}_d\left(\mathbf{x}_*\right)\right) \mathrm{d}\widetilde{\mathbf{f}}_d\left(\mathbf{x}_*\right), \tag{3.34}$$

where $q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_*)\right) = \int p\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_*) \mid \mathbf{u}\right) q(\mathbf{u}) d\mathbf{u} = \prod_{c=1}^{C_d} q\left(f_d^c(\mathbf{x}_*)\right)$. The approximated latent parameter functions $q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_*)\right)$ are mutually independent, so we obtain

$$\begin{aligned} p\left(y_d(\mathbf{x}_*) \mid \mathbf{y}\right) &\approx \int p\left(y_d(\mathbf{x}_*) \mid \widetilde{\mathbf{f}}_d(\mathbf{x}_*)\right) q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_*)\right) d\widetilde{\mathbf{f}}_d(\mathbf{x}_*) \\ &= \int \frac{\exp\left(f_d^{y_d(\mathbf{x}_*)}(\mathbf{x}_*)\right)}{\sum_{c=1}^{C_d} \exp\left(f_d^c(\mathbf{x}_*)\right)} \prod_{c=1}^{C_d} \mathcal{N}\left(f_d^c(\mathbf{x}_*) \mid \mu_{f_d}^c(\mathbf{x}_*), v_{f_d}^c(\mathbf{x}_*)\right) d\widetilde{\mathbf{f}}_d(\mathbf{x}_*). \end{aligned} \tag{3.35}$$

We can use Monte Carlo to approximate the integral in the same way as Liu et al. (2019).

Table 3.1 *Setting and parameters of different GP models in different datasets. All the models for all datasets use 100 inducing variables and 200 mini-batch sizes. "# of Folds" indicates the number of folds for cross-validation. "Q" refers to the optimal number Q of latent functions $\mathcal{U}$.*

Dataset	Model	Q	# of Folds
S-20	MOGPs-AR	[5, 10, 15, 20]	5
S-20	MG-M	[5, 10, 15, 20]	5
Balance	MOGPs-AR (1)	[1, 2, 3]	5
Balance	MG-M	[1, 2, 3]	5
Balance	G-A	None	5
Balance	G-M	None	5
CANE	MOGPs-AR (5)	[6, 9]	5
CANE	MG-M	[6, 9]	5
CANE	G-A	None	5
CANE	G-M	None	5
Mediamill	MOGPs-AR (5)	[10, 15, 20]	5
Mediamill	MG-M	[10, 15, 20]	5
Mediamill	G-A	None	5
Mediamill	G-M	None	5
Bibtex	MOGPs-AR (20)	[5, 10, 15, 20]	3
Bibtex	G-A	None	3
Bibtex	G-M	None	3
UJIIndoorLoc	MOGPs-AR (2)	[4, 8, 12]	3
UJIIndoorLoc	G-A	None	3
UJIIndoorLoc	G-M	None	3

3.3 Experiments

In this section, we evaluate MOGPs-AR in various datasets. We apply MOGPs-AR in a synthetic dataset to show its scalability in the number of classes compared to multi-output Gaussian processes. We also compare MOGPs-AR to other models in different real datasets.

Baselines: We compare the MOGPs-AR with the following two single-output and one multi-output Gaussian process models: 1) A Gaussian process for multi-class classification model (**G-M**), an independent Gaussian process using the softmax likelihood. 2) A Gaussian process multi-class classification with additive noise model (**G-A**), an independent Gaussian process using the softmax likelihood via Gumbel noise. 3) A multi-output Gaussian process model for multi-class classification problems (**MG-M**), a standard linear model of coregionalisation for MOGPs using the softmax likelihood. All models are trained by 4000 iterations using the Adam optimiser with 0.01 learning rate (Kingma and Ba, 2014). We use the same 80% training and 20% validation dataset to choose the optimal number Q of latent functions $\mathcal{U}$, where we re-optimise all hyperparameters during cross-validation, for MOGPs-AR and MG-M.

Evaluation Metrics: There are three different evaluation metrics in this chapter, precisions weighted, recall weighted and F1 weighted:

$$\text{Precision-Weighted} = \frac{1}{\sum_{l\in\mathbb{L}} \left|\mathbb{O}^l_{true}\right|} \sum_{l\in\mathbb{L}} \left|\mathbb{O}^l_{true}\right| P\left(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}\right), \tag{3.36}$$

$$\text{Recall-Weighted} = \frac{1}{\sum_{l\in\mathbb{L}} \left|\mathbb{O}^l_{true}\right|} \sum_{l\in\mathbb{L}} \left|\mathbb{O}^l_{true}\right| R\left(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}\right), \tag{3.37}$$

$$\text{F1-Weighted} = \frac{1}{\sum_{l\in\mathbb{L}} \left|\mathbb{O}^l_{true}\right|} \sum_{l\in\mathbb{L}} \left|\mathbb{O}^l_{true}\right| F_1\left(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}\right), \tag{3.38}$$

$$P(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}) = \frac{|\mathbb{O}^l_{prediction} \cap \mathbb{O}^l_{true}|}{|\mathbb{O}^l_{prediction}|}, \tag{3.39}$$

$$R(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}) = \frac{|\mathbb{O}^l_{prediction} \cap \mathbb{O}^l_{true}|}{|\mathbb{O}^l_{true}|}, \tag{3.40}$$

$$F_1(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}) = 2\frac{P(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}) \times R(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true})}{P(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true}) + R(\mathbb{O}^l_{prediction}, \mathbb{O}^l_{true})}, \tag{3.41}$$

where $\mathbb{O}_{true}$ and $\mathbb{O}_{prediction}$ are separate sets of true and predicted pairs (input data point, class) (e.g., $\mathbb{O}_{true,n} = (\mathbf{x}_n, y_{true,n})$ where $\mathbf{x}_n$ is the n-th input data point and $y_{true,n}$ is the corresponding class for $\mathbf{x}_n$. The $\mathbb{O}^l_{true}$ and $\mathbb{O}^l_{prediction}$ are separate subsets of $\mathbb{O}_{true}$ and

$\mathbb{O}_{prediction}$ (e.g., $\mathbb{O}^{l}_{true} = \{(\mathbf{x}_n, y_{true,n}) \in \mathbb{O}_{true} \mid y_{true,n} = l, n \in \mathbb{N}\}$). The $\mathbb{L}$ and $\mathbb{N}$ are the sets of classes and input data points, respectively. The formulas use $P(\mathbb{O}^{l}_{prediction}, \mathbb{O}^{l}_{true}) = 0$ or $R(\mathbb{O}^{l}_{prediction}, \mathbb{O}^{l}_{true}) = 0$ if $\mathbb{O}^{l}_{prediction} = \emptyset$ or $\mathbb{O}^{l}_{true} = \emptyset$.

The synthetic data experiment was performed on a Dell PowerEdge C6320 with an Intel Xeon E5-2630 v3 at 2.40 GHz and 64GB of RAM. All real data experiments were performed on a PowerEdge R740XD server with NVIDIA Tesla v100 32GB GDDR.

3.3.1 B matrix evaluation

In this subsection, we evaluate the **B** matrix in our model on a synthetic data. We created a synthetic two-class classification dataset and fit our model to the dataset. After training, we obtained $\mathbf{B} = \begin{pmatrix} 11.24525249 & -11.57961005 \\ -11.57961005 & 11.92390914 \end{pmatrix}$. The absolute values of each element in **B** are very similar to each other, and the covariance is negative (e.g., -11.57961005). Following this, we create a two-class classification synthetic dataset by sampling latent parameter functions ($f(.)$) from a normal distribution with zero mean and covariance matrix $\mathbf{B} \bigotimes \mathbf{K}$, where $\mathbf{K} \in \mathbf{R}^{N \times N}$ has entries computed using $k(\mathbf{x}_n, \mathbf{x}_m)$, which is assumed to be RBF with 0.1 lengthscale and 1 variance and $\mathbf{x}_n$ or $\mathbf{x}_m$ ranges between -10 and 10. Figure 3.1 (Left) shows sample points for two latent parameter functions, f_1 and f_2, respectively. If a point in f_1 is greater than its corresponding point in f_2 for the same input, the sample belongs to class one; otherwise, it belongs to class two. The total number of data points is 2000, as shown in Figure 3.1 (Right).

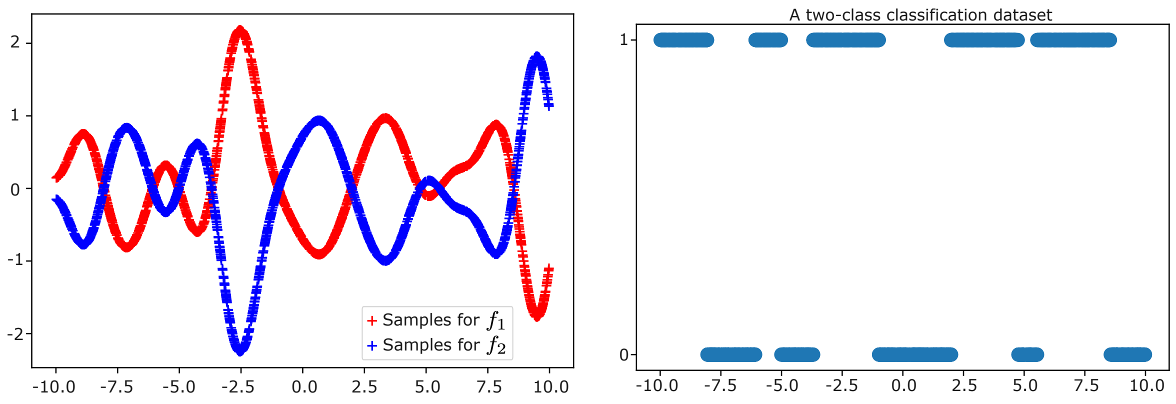


Fig. 3.1 Left: Sample data points corresponding to latent parameter functions f_1 and f_2, respectively; Right: The synthetic dataset consists of data points with a value of one on the y-axis belonging to class one, while data points with a value of zero belong to class two.

To check the robustness of the **B** estimation, we performed five-fold cross-validation on the synthetic dataset using our model. To make it easy to compare the estimated **B** with the true **B**, we set the number of latent functions, $\mathcal{U}(\cdot) = \{u_q(\cdot)\}_{q=1}^{Q}$, to one ($Q = 1$). Table 3.2

Table 3.2 *The values of* **B** *matrix and Recall-Weighted in each fold.*

Order of fold	**B**	Recall-Weighted
1st fold	$\begin{pmatrix} 8.21637951 & -8.40607578 \\ -8.40607578 & 8.60015167 \end{pmatrix}$	0.995
2nd fold	$\begin{pmatrix} 11.82047399 & -12.08549469 \\ -12.08549469 & 12.35645729 \end{pmatrix}$	0.99
3rd fold	$\begin{pmatrix} 13.68672327 & -14.18900248 \\ -14.18900248 & 14.70971447 \end{pmatrix}$	1.0
4th fold	$\begin{pmatrix} 13.26049788 & -13.80302979 \\ -13.80302979 & 14.36775851 \end{pmatrix}$	0.9925
5th fold	$\begin{pmatrix} 14.31689019 & -14.75963664 \\ -14.75963664 & 15.21607492 \end{pmatrix}$	0.9975

shows our Recall-Weighted performance and **B** estimation results. We observed that the values of **B** in the second fold are much closer to the true **B** values. However, they are not identical to the true **B**, possibly due to the non-convex nature of our model and the presence of multiple parameters, such as inducing variables. Nonetheless, the main pattern of **B** in all folds matches the true **B**: all absolute values for each element in the B-matrix are similar and the correlation between latent parameter functions is negative.

3.3.2 Synthetic Data

In this subsection, we compare the performance of MOGPs-AR with MG-M on synthetic data where we generate a single-output classification synthetic dataset.[3] We also run an experiment on the synthetic dataset to explain the choice of the number of inducing points in our model. We create a 20-class dataset by assigning a cluster of 100 points normally distributed, where each data point has five features, to each class. In total, there are 2000 samples. Since the synthetic data has 20 classes, we refer to it as S-20. We use 20 classes to compare MOGPs-AR with MG-M in terms of scalability. MOGPs-AR and MG-M use the same parameter setting (see Table 3.1) apart from that MOGPs-AR use a different number of subset classes.

[3]This data is generated from scikit-learn (Pedregosa et al., 2011)

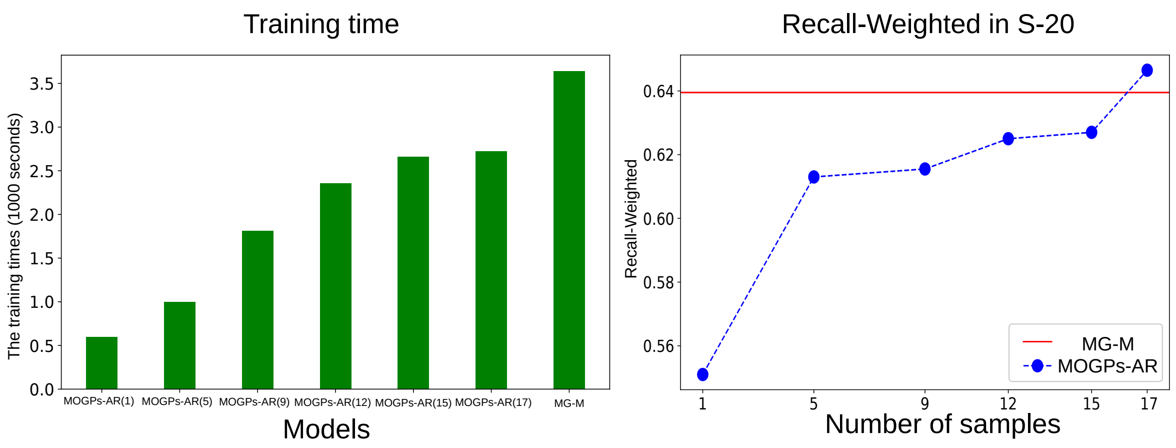


Fig. 3.2 Left: the training time in MG-M and MOGPs-AR model in S-20 (MOGPs-AR(5) means that MOGPs-AR with a subset of classes $\mathcal{S}_d \subseteq \{1,\ldots,C_d\}\backslash\{y_d(\mathbf{x})\}$ with size $|\mathcal{S}_d| = 5$ ($|\mathcal{S}| = |\mathcal{S}_d| + 1$) and we use $y_d(\mathbf{x})$ and C_d for notation consistency but here there is only one output); Right: the recall weight between MG-M and MOGPs-AR with a different number of samples (e.g., 5 means $|\mathcal{S}_d| = 5$).

We compare MOGPs-AR with MG-M in terms of training time and recall-weighted performance. Figure 3.2 shows the mean training time for MOGPs-AR is less than MG-M in five folds cross-validation. This is because the computational complexity of MOGPs-AR is less than MG-M. As mentioned in 3.2.3.2, compared to MG-M, MOGPs-AR reduce the computational complexity of the product $\mathbf{K}_{\tilde{\mathbf{f}}\mathbf{u}}$ from $\mathcal{O}\left(C|\mathcal{B}|QM^2\right)$ to $\mathcal{O}\left(D|\mathcal{S}||\mathcal{B}|QM^2\right)$ where $D|\mathcal{S}| \ll C$. Figure 3.2 empirically shows the mean training time of MOGPs-AR (1) with 596s is nearly one-sixth of MG-M with 3641s. The mean training time in MOGPs-AR increases as the number of $|\mathcal{S}_d|$ increases but it is still less than MG-M. While MOGPs-AR has less training time than MG-M, it performs similarly in recall-weighted with MG-M for S-20. Even if we use a small subset of classes, e.g., five classes, MOGPs-AR also has a close performance to MG-M (see Figure 3.2 right panel). The recall-weighted of MOGPs-AR slightly increases as the number of samples increase. Further, we notice that MOGPs-AR (17) has a better performance than MG-M. In theory, MOGPs-AR should have the same performance as MG-M, however, we can not perform convex optimisation for both MG-M and MOGPs-AR; so, in practice, MOGPs-AR may outperform MG-M in various performance metrics.

In general, the number of inducing points can be determined using cross-validation strategies. In our case, to identify a reasonable number of inducing points for our model, we use the S-20 dataset and implemented MOGPs-AR(5) with a range of inducing point values, varying from 10 to 200. Figure 3.3 illustrates the Recall-Weighted of MOGPs-AR(5) with different numbers of inducing points on S-20, demonstrating the robustness of our model to the number of inducing points. Surprisingly, our results show that the number of inducing points of 40 provides the best performance, despite the expectation that more inducing points would result in better performance. This might be due to the fact that our model is a non-convex model. The chapter's primary focus is on presenting a new MOGP extension capable of handling large-scale multi-output multi-class classification problems, such as those with hundreds of classes. Therefore, we employed a fixed number of inducing points for all GP models in each experiment to ensure consistency across all datasets. Specifically, we selected 100 inducing points, which is the same number used in Moreno-Muñoz et al. (2018) study, on which our model is based. This choice was also influenced by computational cost. Similarly, we determined other parameters such as the mini-batch size based on computational cost considerations.

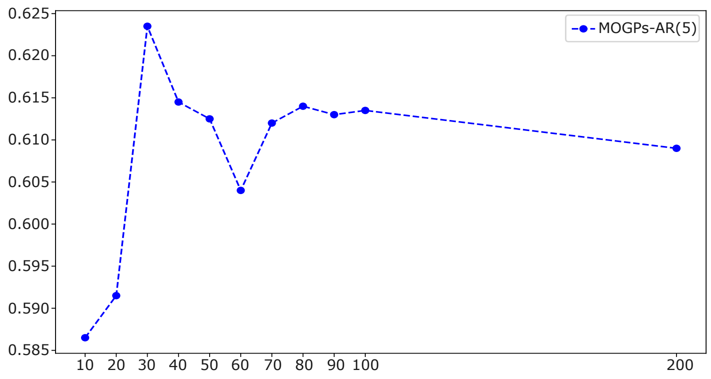


Fig. 3.3 Recall-Weighted in S-20 (MOGPs-AR(5) with a different number of inducing points).

3.3.3 Single-output GP Classification: Four Real Datasets

We use the following four real datasets to test the performance of the different GP classifiers: 1) **Balance** (Dua and Graff, 2017) is a dataset for the results of psychology experiments. There are 625 data points with four discrete variables: Left-Weight, Left-Distance, Right-Weight and Right-Distance. The value of all four discrete variables ranges from one to five. The dataset consists of three classes: the balance scale tipped to the right (R), tipped to the left (L) or balanced (B). 2) **CANE9** (Dua and Graff, 2017) contains 1080 documents of free text business descriptions of Brazilian companies. Those documents are divided into nine different categories. Each document has 856 integer variables (word frequency). 3) **Mediamill** (Snoek et al., 2006) is a multi-label dataset for generic video indexing. To apply multi-classification, we only maintained one label, which is the first label to appear, for each data point. Further, we only use part of this dataset since the original dataset is highly imbalanced. We then obtain the number of data points for each class ranged from 31 to 545. In total, we have 6689 data points with 120 numeric features and 35 classes. 4) **Bibtex** dataset (Katakis et al., 2008) is also a multi-label data that contains 7395 Bibtex entries with 1836 variables. Similarly, we only maintained one label, which is the first label to appear, obtaining 148 classes.

In all three performance measures, MOGPs-AR outperforms G-A and G-M on all four datasets (see Figure 3.4). This is because MOGPs-AR can use each latent parameter function in **f**, which is a linear combination of latent functions $\mathcal{U}$, to predict each class. The underlying function of the latent functions $\mathcal{U}$ and $\mathbf{B}_q$ can transfer knowledge between each class in the same output. However, G-A and G-M only have independent Gaussian processes that cannot capture the similarity between each class. Further, Figure 3.4 indicates that using a small subset of classes (e.g., MOGPs-AR(1) or MOGPs-AR(5)), MOGPs-AR obtains a similar result as MG-M for Balance, CANE9 and Mediamill datasets as discussed as in 3.3.2.

Compared with single-output Gaussian processes, MOGPs-AR can achieve around 10% improvement in terms of three performance metrics on Balance and CANE9 dataset (Figure 3.4 upper panel). The optimal number (Q) of latent functions $\mathcal{U}$ is two and nine for the Balance and CANE9 datasets, respectively. Those latent functions share the knowledge between each class and help to improve the performance. There is also a connection between single output and multi-output Gaussian processes. Considering an extreme case, we assume there is only one class, Q=1 and $\mathbf{B}_q$ = 1, in theory, MOGPs-AR and MG-M have the same structure as G-A and G-M, respectively.

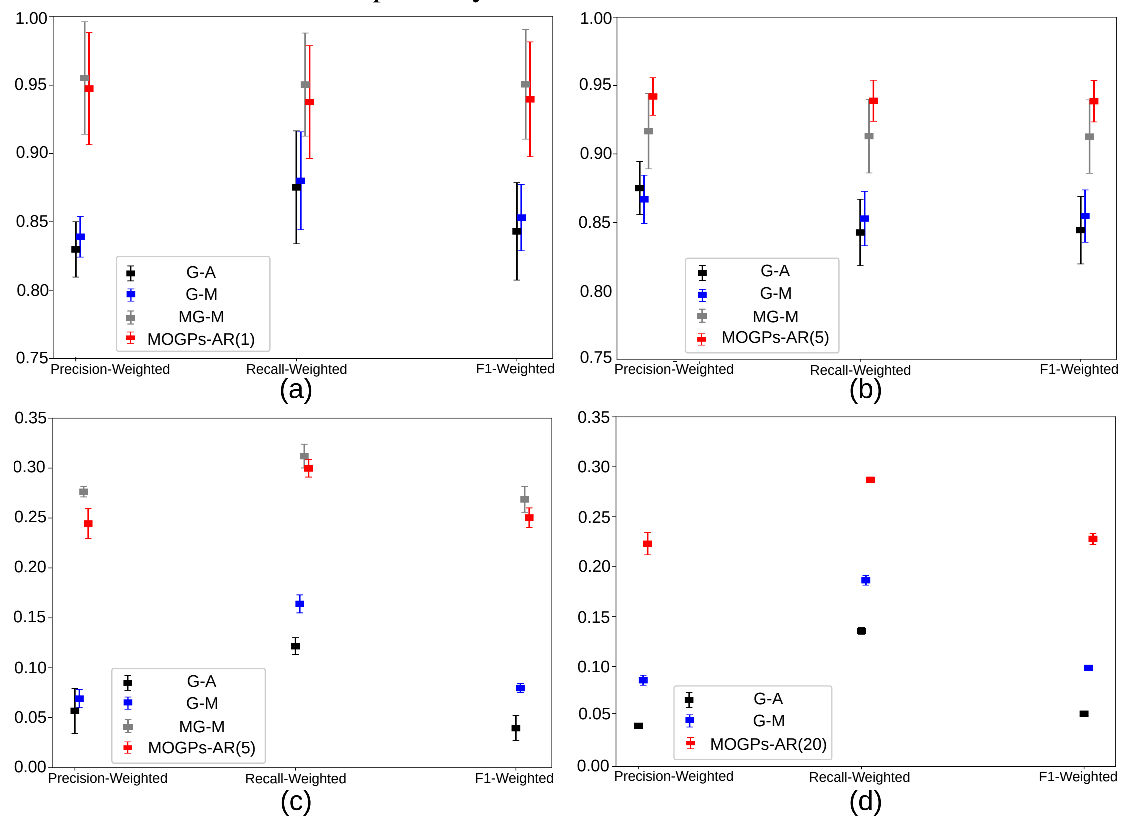


Fig. 3.4 Performance in five folds cross-validation (mean ± standard deviation) in Balance, CANE9 and Mediamill datasets (Three fixed training/test datasets in the Bibtex). (a): Balance data results; (b): CANE9 data results; (c): Mediamill data results; (d): Bibtex data results.

Regarding both Mediallmill (35 classes) and Bibtex (148 classes), MOGPs-AR has excellent performance compared to the single-output Gaussian processes and MG-M. For the Mediamill dataset, based on capturing dependency between each class, MOGPs-AR is nearly six times better than G-A and four times better than G-M in terms of F1-Weighted, where the mean of F1-Weighted is 0.04 for G-A, 0.08 for G-M and 0.25 for MOGPs-AR. Further, we cannot apply MG-M in the Bibtex dataset since it is not able to compute $\mathbf{K}_{\bar{\mathbf{f}}\mathbf{u}}$ (out of memory). However, MOGPs-AR scales well since it only uses a subset of classes (MOGPs-AR (20)) for prediction.

3.3.4 Multi-output GP Classifications: UJIIndoorLoc

To compare the performance of MOGPs-AR in multi-output multi-class classification problems, we apply MOGPs-AR to UJIIndoorLoc dataset (Torres-Sospedra et al., 2014). There are 21048 instances that rely on a WIFI fingerprint for three buildings of Universitat Jaume I where Building I and Building II each has four floors and Building III has five floors. Each instance has 520 features based on signal strength intensity. We randomly sample 200 data points from each floor so there are 800 data points each for Building I and Building II and 1000 data points for Building III. Further, we standardise the dataset for each building. We make predictions for each floor depending on the 520 features. Since Universitat Jaume I has three buildings, we assume there is a strong correlation between each building. We regard each building as its own output and different floors as different classes in our model. The UJIIndoorLoc is considered as a multi-output multi-class classification problem. In this and the following experiment, we do not apply the MG-M model due to its computational complexity. MOGPs-AR can be an alternative model for MG-M so we only consider MOGPs-AR and two single-output GP models.

Figure 3.5 shows that MOGPs-AR outperforms single-output Gaussian processes in Buildings I, II and III in all three performance measures. For example, MOGPs-AR can achieve around 50% improvement in terms of recall-weighted on Building I compared with single output Gaussian processes. The reason is that MOGPs-AR can capture intra- and inter-dependencies in all three buildings. The dependencies can help improve the prediction for all buildings. The single-output Gaussian process cannot use the dependency so it does not perform well in UJIIndoorLoc.

We built Hinton diagrams of coregionalisation matrices to analyse the estimated hyperparameters and correlations between classes. A coregionalisation matrix $\mathbf{B}_q$ can have positive or negative entries, indicating positive or negative cross-covariances. In the Hinton diagram of any $\mathbf{B}_q$, the white and black squares represent the positive and negative correlation values, respectively, and the magnitude of each value is represented by the size of each square (see

Figure 3.6). In each fold of cross-validation, there are different training and test datasets and our model is also a non-convex model so we could obtain different parameters for our trained model for each fold. Hence, we obtain different Hinton diagrams for the same dataset in the different folds. We still obtain a similar pattern in different folds for the same dataset probably because the training and test datasets are from the same distribution. Figure 3.6 shows Hinton diagrams of a few different coregionalisation matrices and the UJIIndoorLoc dataset. Figure 3.6 (a) and (b) indicate that Building III correlates less with Building I and II.

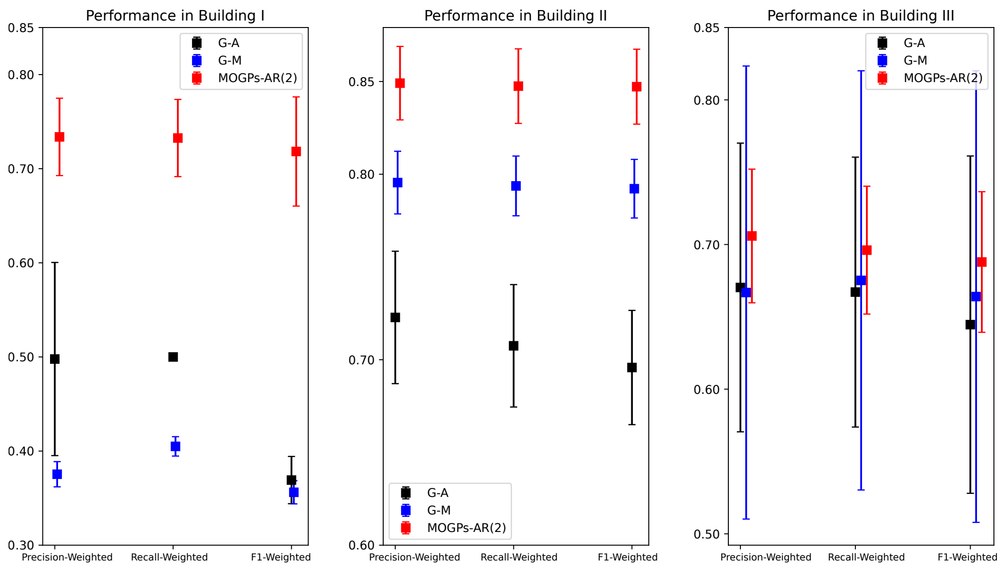
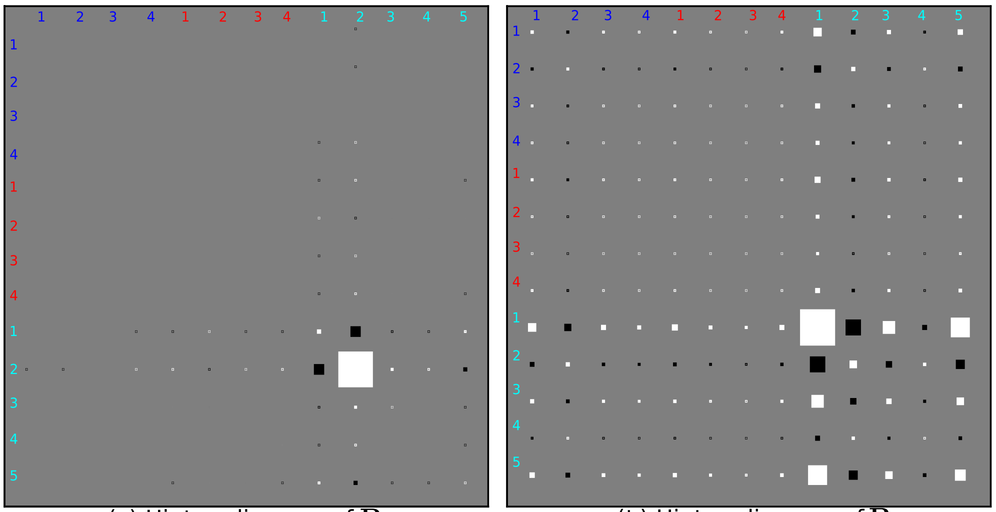


Fig. 3.5 Performance in cross-validation (mean ± standard deviation).



(a) Hinton diagram of $\mathbf{B}_4$ (b) Hinton diagram of $\mathbf{B}_6$

Fig. 3.6 Hinton diagram of a few coregionalisation matrices. (a) and (b) are Hinton diagrams of $\mathbf{B}_4$ and $\mathbf{B}_6$ for the UJIIndoorLoc dataset during the second fold cross-validation where the blue, red and cyan colours are the index of each floor in Buildings I, II and III.

To investigate the correlation between intra- and inter-output, we create a *global absolute coregionalisation matrix*. First, we create absolute coregionalisation matrices $\left\{\mathbf{B}_q^{abs}\right\}_{q=1}^{Q}$, where $\mathbf{B}_q^{abs} \in \mathbf{R}^{C\times C}$, by taking the absolute value of each entry in $\mathbf{B}_q$. We use the absolute value of the entries to avoid the cancellation of positive or negative values when summing them together, which indicates the absence of the correlation. Second, we obtain the mean of those absolute coregionalisation matrices: $\overline{\mathbf{B}} = \frac{1}{Q}\sum_{q=1}^{Q}\mathbf{B}_q^{abs}$ and $\overline{\mathbf{B}} \in \mathbf{R}^{C\times C}$. Since we are performing K-fold cross-validation, we have different K mean absolute coregionalisation matrices: $\left\{\overline{\mathbf{B}}^i\right\}_{i=1}^{K}$, where $\overline{\mathbf{B}}^i \in \mathbf{R}^{C\times C}$ refers to the mean absolute coregionalisation matrices during the i-th fold cross-validation. Further, we calculate the mean of $\left\{\overline{\mathbf{B}}^i\right\}_{i=1}^{K}$ for all K-fold cross-validations so $\tilde{\mathbf{B}} \in \mathbf{R}^{C\times C} = \frac{1}{K}\sum_{i=1}^{K}\overline{\mathbf{B}}^i$:

$$\tilde{\mathbf{B}} = \begin{bmatrix} \left(\tilde{\mathbf{B}}\right)_{1,1} & \cdots & \left(\tilde{\mathbf{B}}\right)_{1,D} \\ \left(\tilde{\mathbf{B}}\right)_{2,1} & \cdots & \left(\tilde{\mathbf{B}}\right)_{2,D} \\ \vdots & \ddots & \vdots \\ \left(\tilde{\mathbf{B}}\right)_{D,1} & \cdots & \left(\tilde{\mathbf{B}}\right)_{D,D} \end{bmatrix}, \tag{3.42}$$

where $\left(\tilde{\mathbf{B}}\right)_{i,j} \in \mathbf{R}^{C_i\times C_j}$ indicates the correlations for all latent parameter functions between i-th output and j-th output. Finally, in order to find the correlation for outputs independently, we calculate a scalar $\tilde{B}_{i,j} = \frac{1}{C_iC_j}\sum_m\sum_n\left\{\left(\tilde{\mathbf{B}}\right)_{i,j}\right\}_{m,n}$, which represents dependence between i-th output and j output. We therefore define a global absolute coregionalisation matrix (GACM $\in \mathbf{R}^{D\times D}$) as the following:

$$\text{GACM} = \begin{bmatrix} \tilde{B}_{1,1} & \cdots & \tilde{B}_{1,D} \\ \tilde{B}_{2,1} & \cdots & \tilde{B}_{2,D} \\ \vdots & \ddots & \vdots \\ \tilde{B}_{D,1} & \cdots & \tilde{B}_{D,D} \end{bmatrix}. \tag{3.43}$$

Figure 3.7 shows the correlation between each building captured by our model. We can see that there is a strong correlation between the different buildings. Building I and Building II have a relatively strong correlation compared to Building I and Building III or Building II and Building III. Building II has the strongest intra-output correlation while Building III has the smallest intra-output correlation among those three buildings.

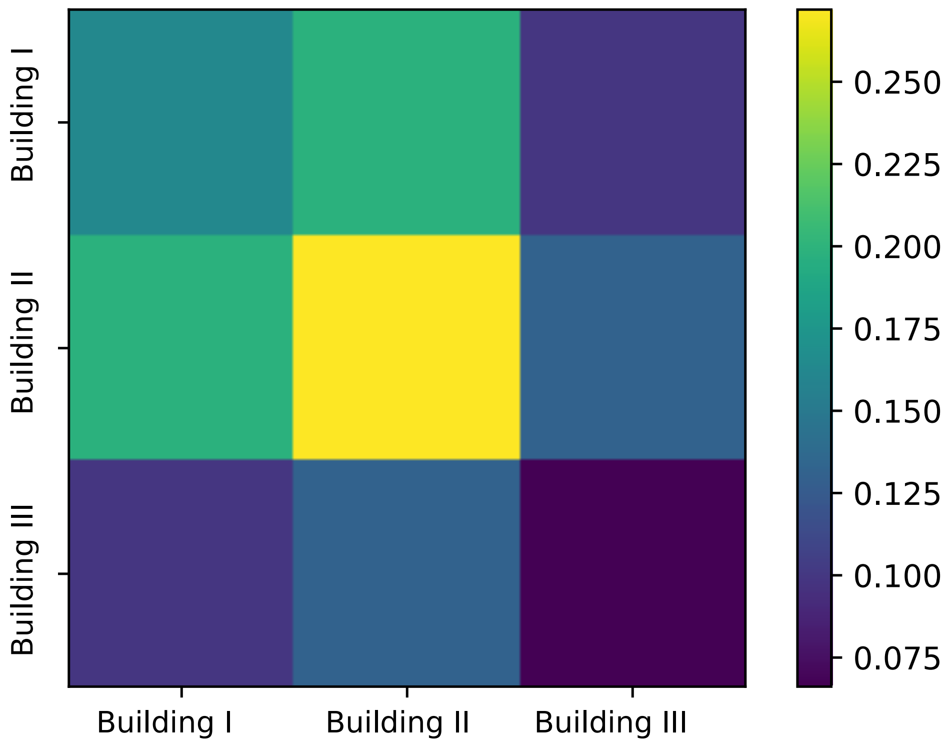


Fig. 3.7 Global absolute coregionalisation matrix of UJIIndoorLoc dataset.

3.4 Summary

In this chapter, we have introduced MOGPs-AR, a novel framework that allows the use of multi-output Gaussian processes for multi-output multi-class classification. MOGPs-AR can tackle large scale datasets and a large number of classes in each output.

We show experimentally that MOGPs-AR has a similar result to MG-M, which is a linear model of coregionalisation and uses a similar stochastic variational inference method to us. However, the training time of MOGPs-AR is less than MG-M. Experimental results in various datasets also indicate that MOGPs-AR significantly improves the performance compared to single-output Gaussian processes.

In the next chapter, we will propose an extension of MOGPs-AR via integrating a convolutional kernel (Van der Wilk et al., 2017) to replace the RBF-ARD. Compared to the MOGPs-AR, the novel extension can effectively extract features from image data, and therefore its performance is superior to that of MOGPs-AR in image classification problems.

Chapter 4

Multi-output Convolutional Gaussian Processes for Images

Our main contribution in this chapter [1] we provide an extension of the MOGPs-AR called *multi-output convolutional Gaussian processes with augment & reduce (MOCGPs-AR)*[2]. We incorporated a convolutional kernel into our model to handle image datasets. The MOCGPs-AR also inherits the properties of MOGPs-AR, allowing it to deal with large-scale classification of images by subsampling both training datasets and classes for each output.

In Chapter 3, we introduced MOGPs-AR, which is a new extension of MOGPs, that handles large scale multi-output multi-class classification. However, the most common type of data in multi-class classification problems consists of image data, and MOGPs-AR is not specifically designed to handle such high-dimensional data.

In this chapter, we introduce MOCGPs-AR and enable it to allow downsized images as input data. To efficiently deal with downsized images, we employ convolutional kernels (Van der Wilk et al., 2017), computing the entries of the kernel matrices using kernels over patches of the images and integrating these kernels within a MOGP. Since our model is able to capture both intra- and inter-output dependencies, it also provides a means to perform transfer learning in the multi-task setting. We show an example of the multi-task learning ability of our model in the Ommiglot dataset. To the best of our knowledge, this is the first time that a multi-task multi-class Gaussian process model has been used over such a dataset.

[1]This chapter is based on **Ma, C.**, & Álvarez, M. A. (2023). Large scale multi-output multi-class classification using Gaussian processes. *Machine Learning*, 1-30.

[2]The model is implemented based on Python 3.7, mainly depending on the library GPflow 2.1.3 which was built on the library TensorFlow 2+. Our code is publicly available in the repository https://github.com/ChunchaoPeter/MOGPs-AR.

4.1 Related Work

Several research works have dealt with multi-class classification with image datasets using single-output GPs (Blomqvist et al., 2019; Hensman et al., 2015b; Krauth et al., 2016; Van der Wilk et al., 2017). Hensman et al. (2015b) proposed an inference scheme that can help GPs deal with classification problems by combining variational and MCMC methods. The work of Krauth et al. (2016) used a leave-one-out objective function for optimising the hyper-parameters and a radial basis function kernel with automatic relevance determination (RBF-ARD). Van der Wilk et al. (2017) introduced a convolutional kernel that made Gaussian processes more suited to data with high-input dimension like images. In terms of MNIST dataset, Van der Wilk et al. (2017) obtained a better accuracy performance than Hensman et al. (2015b) and Krauth et al. (2016). Based on Van der Wilk et al. (2017), Blomqvist et al. (2019) proposed a deep Gaussian process with convolutional structure. Similarly, our model (MOCGPs-AR) is built on that of Van der Wilk et al. (2017). However, different from single-output GPs, our model can boost predictions by exploiting the correlation between all latent parameter functions. Further, our model can cope with multiple output problems while single-output GPs can only tackle single-output problems.

MOGPs-AR is the most relevant work to MOCGPs-AR so the related work in Chapter 3 also relates to the work in this Chapter to some extent. In terms of structure, MOGPs-AR is a special case of MOCGPs-AR. MOCGPs-AR is a MOGPs-AR with convolutional structure or convolutional kernel (Van der Wilk et al., 2017). Because of the convolutional kernel, to reduce the computational complexity, MOGPs-AR uses the inducing points method while MOCGPs-AR uses the inducing patches method. The difference between inducing points and inducing patches is the dimension size. The dimensions of the inducing points are the same as the input data, whereas the dimensions of the inducing patches match the patch of the images. In terms of a practical application, compared to MOGPs-AR, MOCGPs-AR can effectively deal with downsized images through convolutional kernels (Van der Wilk et al., 2017).

4.2 Methodology

In this section, we will derive the MOCGPs-AR model. We first develop the LMC model with a convolutional kernel. We then describe stochastic variational inference for MOCGPs-AR. MOCGPs-AR use the same likelihood function and prediction as MOGPs-AR (please see Sections 3.2.2 and 3.2.4 for more detail)

Similar to Chapter 3, we assume there is the vector $\mathbf{y}(\mathbf{x}) \in \mathbf{R}^D$ to represent different D outputs:

$$\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), y_2(\mathbf{x}), \cdots, y_D(\mathbf{x})]^\top, \tag{4.1}$$

where $\mathbf{x} \in \mathbf{R}^v$. The latent parameter functions are:

$$\mathbf{f}(\mathbf{x}) = \left[f_1^1(\mathbf{x}), f_1^2(\mathbf{x}), \cdots f_1^{C_1}(\mathbf{x}), f_2^1(\mathbf{x}), f_2^2(\mathbf{x}), \cdots, f_D^{C_D}(\mathbf{x})\right]^\top \in \mathbf{R}^{C\times 1}. \tag{4.2}$$

The connection between $\mathbf{y}(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$ is

$$p(\mathbf{y}(\mathbf{x})|\mathbf{f}(\mathbf{x})) = \prod_{d=1}^{D} p\left(y_d(\mathbf{x})|\widetilde{\mathbf{f}}_d(\mathbf{x})\right). \tag{4.3}$$

For the notation and more detail please see Section 3.2.

4.2.1 Extending MOGPs-AR by Including a Convolutional Kernel

We combine the linear model of coregionalisation (LMC) (Álvarez et al., 2012) with the convolutional kernel. The convolutional kernel (Van der Wilk et al., 2017) can effectively exploit features in an image dataset.

Invariances can help us constrain models to generalise well in high-dimensional datasets (Van der Wilk, 2019) so incorporating invariances into GPs can help them deal with images well. GPs' prior constrains the GP models via kernel functions where traditional kernel functions (e.g., RBF-ARD) depend on local metrics, such as the Euclidean distance, to constrain variations. Van der Wilk (2019) provides a way to encode invariance over patches in images into kernels to handle images, as small patches can contain a lot of information for image labels. The general idea is to divide each image into a group of patches first. Then, each patch is applied to the same function (the function follows a GP prior); further, they sum over all the patches' responses for each image. Based on this structure, a translation invariant kernel (the same as Equation 2.17) has been developed, which is invariant to the image including the same patches. However, in the translation invariant kernel, for image classification, the different class labels could have the same feature but in different locations for the image input. To elude this situation, Van der Wilk (2019) added weight for each patch response, resulting in the new kernel, as in Equation 2.18.

Following Van der Wilk (2019)'s idea, we construct a convolutional structure for mutually independent latent functions $\mathcal{U}(\cdot) = \{u_q(\cdot)\}_{q=1}^{Q}$ (a set of Q GPs). Here, we assume $\mathbf{x} \in \mathbf{R}^{W\times H}$ is an image data point that has a $v = W \times H$ size where W and H are the width and height of the image, respectively. We also assume $\mathbf{x}^{[p]}$ is the p^{th} patch of $\mathbf{x}$ with patches of size $E = w \times h$ where w and h are the width and height of each patch, respectively.

After dividing an image into patches, we get a total of $P = (W - w + 1) \times (H - h + 1)$ patches. We begin with a patch response function $u_q\left(\mathbf{x}^{[p]}\right) : \mathbf{R}^{w \times h} \to \mathbf{R}$, which maps a patch of size $E = w \times h$ to a real number in $\mathbf{R}$. Then we add weight with each patch response function and get a latent function $u_q(\mathbf{x}) : \mathbf{R}^{W \times H} \to \mathbf{R}$, where $u_q(\mathbf{x})$ is the sum of all patch responses with weights: $u_q(\mathbf{x}) = \sum_p w_p u_q\left(\mathbf{x}^{[p]}\right)$. Each function $u_q(\cdot)$ is drawn from an independent GP prior: $u_q(\cdot) \sim \mathcal{GP}\left(0, k_q(\cdot,\cdot)\right)$, where k_q is the radial basis function kernel with automatic relevance determination (RBF-ARD) (Williams and Rasmussen, 2006) in this chapter:

$$k_{\text{ard}}\left(\mathbf{x}^{[p]}, \mathbf{x}^{[p']}\right) = \sigma_{\text{ard}}^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{E} \frac{\left(x_j^{[p]} - x_j^{[p']}\right)^2}{l_j^2}\right), \tag{4.4}$$

where $x_j^{[p]}$ is the j-th dimension of $\mathbf{x}^{[p]}$, σ_{ard}^2 is a variance parameter and l_j is the length scale for the j-th input dimension. Then, each $f_d^c(\mathbf{x})$ is defined as

$$f_d^c(\mathbf{x}) = \sum_{q=1}^{Q}\sum_{i=1}^{R_q} a_{d,c,q}^i u_q^i(\mathbf{x}) = \sum_{q=1}^{Q}\sum_{i=1}^{R_q} a_{d,c,q}^i \left(\sum_{p=1}^{P} w_p u_q^i(\mathbf{x}^{[p]})\right), \tag{4.5}$$

where $a_{d,c,q}^i \in \mathbf{R}$ and we assume $\{k_q(\cdot,\cdot)\}_{q=1}^Q$ have hyperparameters $\{\phi_q\}_{q=1}^Q$. The difference between the convolutional kernel model and a more classic kernel, e.g., RBF, is that we use the convolutional structure term $\sum_{p=1}^P w_p u_q^i(\mathbf{x}^{[p]})$ instead of solely $u_q^i(\mathbf{x})$. Further, $f_d^c(\mathbf{x})$ has a zero mean and the cross-covariance function of $f_d^c(\mathbf{x})$ is

$$\begin{aligned} k_{f_d^c f_{d'}^{c'}}\left(\mathbf{x}, \mathbf{x}'\right) &= \operatorname{cov}\left[f_d^c(\mathbf{x}), f_{d'}^{c'}\left(\mathbf{x}'\right)\right] \\ &= \sum_{q=1}^{Q} b_{(d,c),(d',c')}^q \left[\sum_{p=1}^{P}\sum_{p'=1}^{P} w_p w_{p'} k_q\left(\mathbf{x}^{[p]}, \mathbf{x}'^{[p']}\right)\right], \end{aligned} \tag{4.6}$$

where $b_{(d,c),(c',c')}^q = \sum_{i=1}^{R_q} a_{d,c,q}^i a_{d',c',q}^i$. Our model works for each output with different input datasets but now we assume all output has the same input vector for simplicity in the presentation. Now, the kernel can be represented as:

$$\mathbf{K} = \sum_{q=1}^{Q} \mathbf{B}_q \otimes \mathbf{K}_q, \tag{4.7}$$

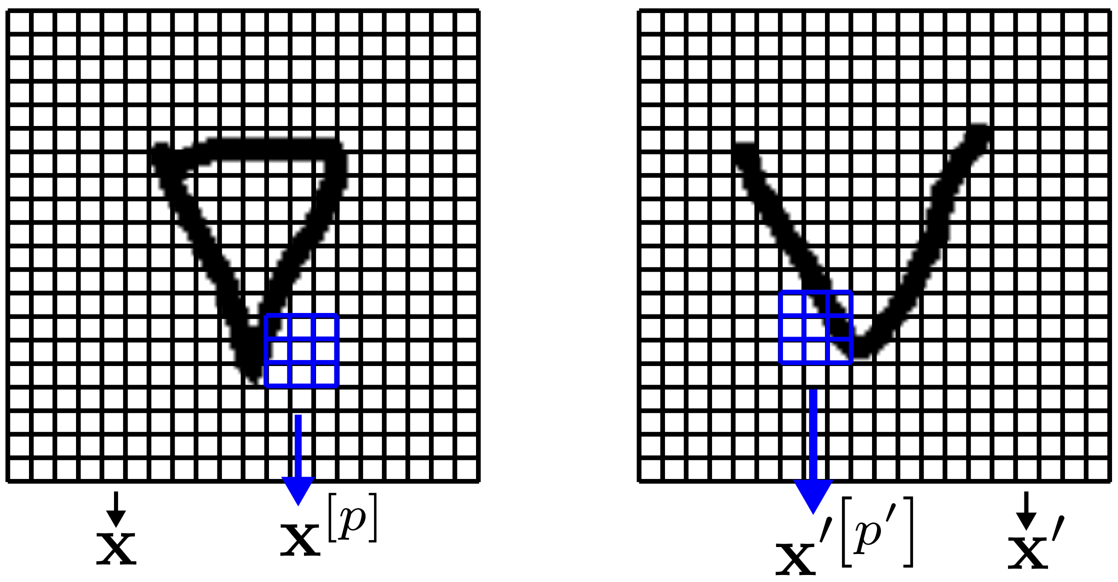


Fig. 4.1 An example of two images for our kernel inputs. The two images are two characters in the Ojibwe alphabet (please see section 4.3 for more detail). We consider two characters as two classes. The two images are one data point for each class separately. Left: The whole image is considered as an input data point $\mathbf{x}$ and the blue grid represents the p-th patch $\mathbf{x}^{[p]}$. Right: The whole image is considered as an input data point $\mathbf{x}'$ and the blue grid represents the p'-th patch $\mathbf{x}'^{[p']}$.

where $\mathbf{B}_q$ has elements $\left\{b^q_{(d,c),(d',c')}\right\}_{d=1,d'=1,c=1,c'=1}^{D,D,C_d,C_{d'}}$. $\mathbf{K}_q \in \mathbf{R}^{N\times N}$ has elements computed using $\sum_{p=1}^{P}\sum_{p'=1}^{P} w_p w_{p'} k_q\left(\mathbf{x}_n^{[p]}, \mathbf{x}_m^{[p']}\right)$ for $\mathbf{x}_n, \mathbf{x}_m \in \mathbf{X}$ and each $\mathbf{B}_q \in \mathbf{R}^{C\times C}$ is a coregionalisation matrix.

4.2.2 Scalable Variational Inference

Similar to MOGPs-AR, MOCGPs-AR also suffer from computational complexity. To reduce the computational complexity, we use inducing patches and subsampling techniques. The subsampling strategy here is the same as MOGPs-AR (see Section 3.2.3.2 for more detail).

Since we use image datasets in this section we define the inducing patches (Van der Wilk et al., 2017) at the latent functions $\mathcal{U}(\cdot)$. Let $\mathbf{Z}_q = \left\{\mathbf{z}_q^{(m)}\right\}_{m=1}^{M} \in \mathbf{R}^{M\times E}$ be a set of M inducing patches (Van der Wilk et al., 2017) for each latent function u_q and $\mathbf{Z} = \{\mathbf{Z}_1\}_{q=1}^{Q}$. Let $\mathbf{u} = \left[\mathbf{u}_1^\top, \cdots, \mathbf{u}_Q^\top\right]^\top \in \mathbf{R}^{QM\times 1}$ be a set, where each latent function is assumed to have its own inducing patches and $\mathbf{u}_q = \left[u_q\left(\mathbf{z}_q^{(1)}\right), \cdots, u_q\left(\mathbf{z}_q^{(M)}\right)\right]^\top$ evaluated at $\mathbf{Z}_q$.

Except for $\mathbf{u} = \left[\mathbf{u}_1^\top, \cdots, \mathbf{u}_Q^\top\right]^\top$, the lower bound $\mathcal{L}$ for $\log p(\mathbf{y})$ for MOCGPs-AR is the same as MOGPs-AR (see Section 3.2.3.1 for more detail).

Table 4.1 *Omniglot data: we show the number of data points and classes for each alphabet in the Omniglot dataset. The columns of the background set and evaluation set show 30 and 20 alphabets, respectively.*

Omniglot-evaluation	N_{data}	classes	Omniglot-background	N_{data}	classes
Angelic	400	20	Alphabet-of-the-Magi	400	20
Atemayar-Qelisayer	520	26	Anglo-Saxon-Futhorc	580	29
Atlantean	520	26	Arcadian	520	26
Aurek-Besh	520	26	Armenian	820	41
Avesta	520	26	Asomtavruli-(Georgian)	800	40
Ge-ez	520	26	Balinese	480	24
Glagolitic	900	45	Bengali	920	46
Gurmukhi	900	45	Blackfoot (Canadian-Aboriginal-Syllabics)	280	14
Kannada	820	41	Braille	520	26
Keble	520	26	Burmese-(Myanmar)	680	34
Malayalam	940	47	Cyrillic	660	33
Manipuri	800	40	Early-Aramaic	440	22
Mongolian	600	30	Futurama	520	26
Old-Church-Slavonic (Cyrillic)	900	45	Grantha	860	43
Oriya	920	46	Greek	480	24
Sylheti	560	28	Gujarati	960	48
Syriac-(Serto)	460	23	Hebrew	440	22
Tengwar	500	25	Inuktitut-(Canadian-Aboriginal-Syllabics)	320	16
Tibetan	840	42	Japanese-(hiragana)	1040	52
ULOG	520	26	Japanese-(katakana)	940	47
			Korean	800	40
			Latin	520	26
			Malay-(Jawi-Arabic)	800	40
			Mkhedruli-(Georgian)	820	41
			N-Ko	660	33
			Ojibwe-(Canadian-Aboriginal-Syllabics)	280	14
			Sanskrit	840	42
			Syriac-(Estrangelo)	460	23
			Tagalog	340	17
			Tifinagh	1100	55

4.3 Omniglot Dataset

In this section, we apply MOCGPs-AR to the Omniglot image dataset (Lake et al., 2015). The Omniglot dataset includes 1623 various handwritten characters from 50 distinct alphabets. Each of the 1623 characters was drawn by 20 different people (the total number of images is 32460). In terms of the number of data points and classes for each alphabet in the Omniglot dataset, please see the detailed explanation in Table 4.1. Although traditional MOGPs are not specifically designed to deal with image data, MOCGPs-AR can handle image data by incorporating a convolutional kernel (Van der Wilk et al., 2017). The size of each image is 105×105 pixels. To help speed up the computation and reduce the computational complexity in the convolutional kernel, we resize the images from 105×105 to 20×20 as per Santoro et al. (2016). We regard each alphabet as an output in our model. Each alphabet has different characters which are considered as different classes. Therefore, we consider the Omniglot dataset as multi-output multi-class classification problems. We compare MOCGPs-AR with MOGPs-AR, G-M, G-A and MG-M from Chapter 3. The evaluation metrics are precision-weighted, recall-weighted and F1-weighted as in Chapter 3. All data experiments were performed on a PowerEdge R740XD server with NVIDIA Tesla v100 32GB GDDR.

4.3.1 Ojibwe and Blackfoot Alphabets

To compare the performance of MOCGPs-AR in multi-output multi-class classification problems and image input data, we first consider Ojibwe and Blackfoot alphabets as two different multi-class classification problems (see Figure 4.2). Since the two alphabets are from Canadian Aboriginal syllabics, we assume there is a strong correlation between them. Our model can capture the correlation through joint modelling of the two alphabets to improve predictive performance for each multi-class classification problem. There are 14 different characters in each output so there are 14 classes, and each class has 20 data points. We compare MOGPs-AR with MOCGPs-AR. Table 4.2 shows the parameter setting in the Omniglot dataset: Regarding the number of inducing points, we chose 40 inducing points for the Ojibwe and Blackfoot dataset since the dataset has 280 data points for each alphabet. This number of inducing points is sufficient for a total of 280 data points. We chose 100 inducing patches for the Ojibwe and Blackfoot dataset to maintain consistency with Chapter 3, where we used 100 inducing points. For the other datasets, we chose 200 inducing patches as we wanted relatively large inducing patches to provide more information, based on a relatively small mini-batch size of 9.

Figure 4.3 shows that MOCGPs-AR outperforms single-output Gaussian processes in both alphabets in terms of the convolutional kernel or RBF-ARD. The reason is that MOCGPs-AR can capture the dependency between the two alphabets. The dependency can help improve the prediction for both alphabets. The single-output Gaussian processes cannot use the dependency so the single-output Gaussian process with either the convolutional kernel or RBF-ARD does not perform well in both Ojibwe and Blackfoot. The size of the mini-batch is too small that has also a negative influence on the single-output Gaussian processes (Figure 4.4). Especially, the values of the three performance metrics are closed to 0.05 for G-A with the convolutional kernel on Ojibwe.

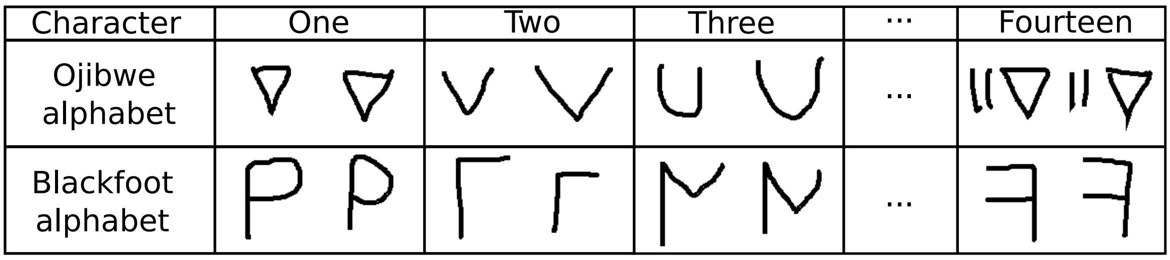


Fig. 4.2 Both the Ojibwe and the Blackfoot alphabets have 14 characters each. For each character, we only illustrate two typical samples.

Table 4.2 *Setting and parameters of different GP models in Omniglot. There are three cross-validations for all models and the optimal number $Q \in [10, 15, 20]$ of latent functions $\mathcal{U}$ for MOGPs-AR. "M" indicates the number of inducing variables or inducing patches; "B" refers to the size of mini-batch. "Con-K" means convolutional kernel.*

Dataset	Model	Kernel	patch-size	M	B
Ojibwe & Blackfoot	MOGP-AR (1)	Con-K	3 * 3	100	50/70/90
Ojibwe & Blackfoot	G-A	Con-K	3 * 3	100	50/70/90
Ojibwe & Blackfoot	G-M	Con-K	3 * 3	100	50/70/90
Ojibwe & Blackfoot	MOGP-AR (1)	RBF-ARD	None	40	50/70/90
Ojibwe & Blackfoot	G-A	RBF-ARD	None	40	50/70/90
Ojibwe & Blackfoot	G-M	RBF-ARD	None	40	50/70/90
All alphabets	MOGP-AR (1)	Con-K	8 * 8	200	9
Background alphabets	MOGP-AR (1)	Con-K	8 * 8	200	9
Evaluation alphabets	MOGP-AR (1)	Con-K	8 * 8	200	9

MOCGPs-AR outperforms MOGPs-AR in both alphabets in terms of three performance metrics (see Figure 4.3). For example, compared to MOGPs-AR, MOCGPs-AR improves the recall-weighted from 0.468 to 0.714 on the Blackfoot alphabet. Moreover, we also combine G-M and G-A with the convolutional kernel and they also have stronger performance compared with RBF-ARD. In particular, G-M with the convolutional kernel obtains 0.5858 compared with 0.0857 using RBF-ARD in terms of recall-weighted on the Blackfoot alphabet. The performance of G-M with the convolutional kernel (0.5858) is better than MOGPs-AR (0.468) on the Blackfoot alphabet. The reason is that the convolutional kernel is captures image-level features more effectively than the RBF-ARD kernel.

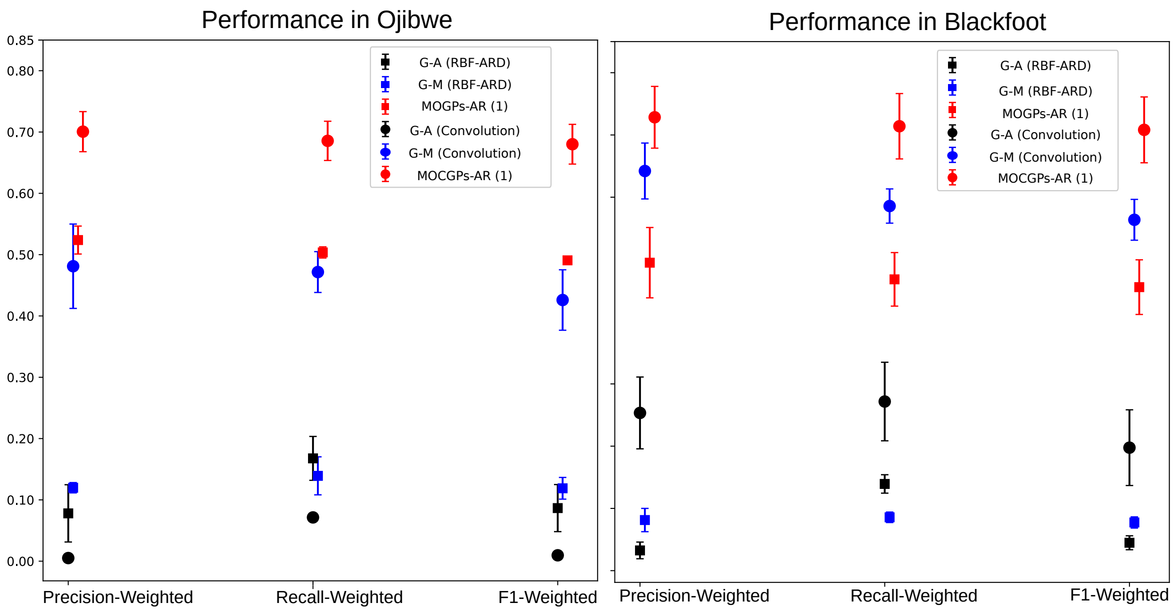


Fig. 4.3 Image: Performance in five folds cross-validation (mean ± standard deviation). We compare MOCGPs-AR with other models for all the models.

To investigate the effects of mini-batch size, we set up another experiment. We train the exact same models with the parameters initialised in the same way as the experiment above but using different mini-batch sizes (e.g., 50, 70, 90). Since the convolution kernel provided better results in the previous experiments, we only show the results using MOCGPs-AR, G-A and G-M with the convolution kernel and the recall-weighted performance measure for both alphabets. Figure 4.4 shows that the size of the mini-batch has more influence on single-output Gaussian processes than MOCGPs-AR. A small size number for the mini-batch, e.g., 50, has a negative impact on G-M and G-A. However, MOCGPs-AR has a slight increase in performance or maintains a similar result with the mini-batch size increasing. G-A and G-M improve the performance as the mini-batch size grows from 50 to 90. When the size of the mini-batch is 90, G-M has a similar performance with MOCGPs-AR. However, when we consider the mini-batch of size 50, MOCGPs-AR still can achieve good performance compared to single output GPs. Moreover, MOCGPs-AR as with a larger number of outputs, the performance would significantly improve (see Section 4.3.2).

To analyse the estimated hyperparameters and correlations between classes, we analyse the coregionalisation matrices through Hinton diagrams. We intend to obtain the values of a few different coregionalisation matrices $\mathbf{B}_q$ to investigate the correlation between the classes. Figure 4.5 shows Hinton diagrams of different coregionalisation matrices for Ojibwe and Blackfoot alphabets, with (a) and (b) indicating that the Ojibwe alphabet has a strong correlation with the Blackfoot alphabet.

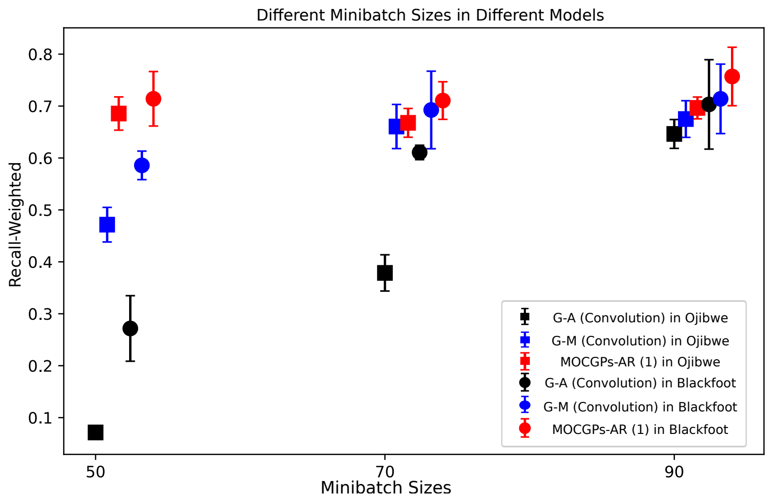


Fig. 4.4 Recall-Weighted Performance in cross-validation (mean ± standard deviation).

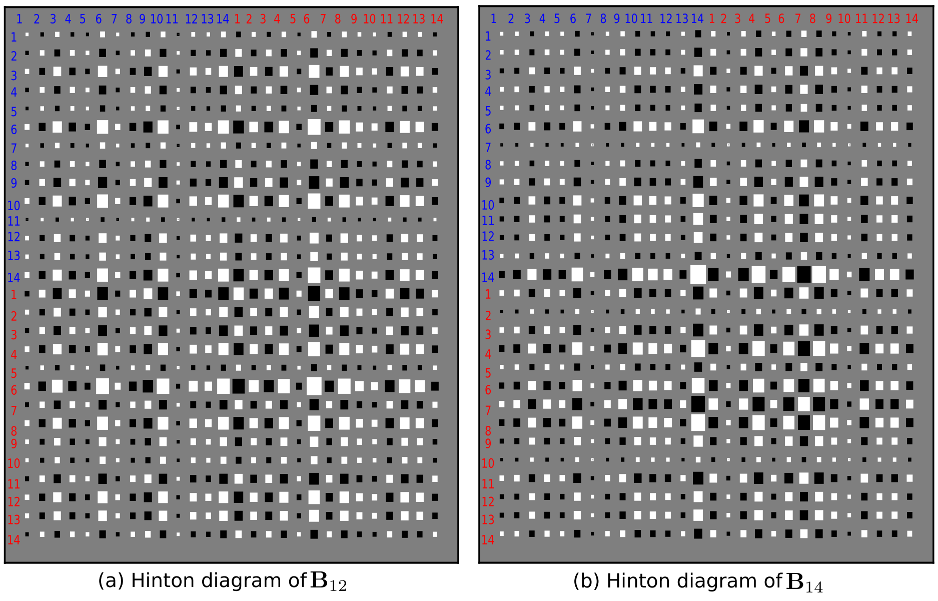


Fig. 4.5 Hinton diagrams of a few coregionalisation matrices. (a) and (b) are Hinton diagrams of $\mathbf{B}_{12}$ and $\mathbf{B}_{14}$ for both Ojibwe and Blackfoot alphabets during the first fold cross-validation where the red and blue colours are the index of 14 classes for Ojibwe and Blackfoot alphabets, respectively. The white and black squares represent the positive and negative correlation values respectively, where the magnitude of each value is represented by the size of each square.

We plot a global absolute coregionalisation matrix in Figure 4.6 to analyse the estimated hyperparameters and correlations between each output for Ojibwe and Blackfoot alphabets. Figure 4.6 indicates that our model captures the correlation between each alphabet. Since both alphabets are from Canadian Aboriginal syllabic we expect that they have a strong correlation. Figure 4.6 indeed shows there is a similar global correlation between intra- and inter- output for both alphabets, which indicates that our model has the capacity to capture the underlying correlation among those related datasets.

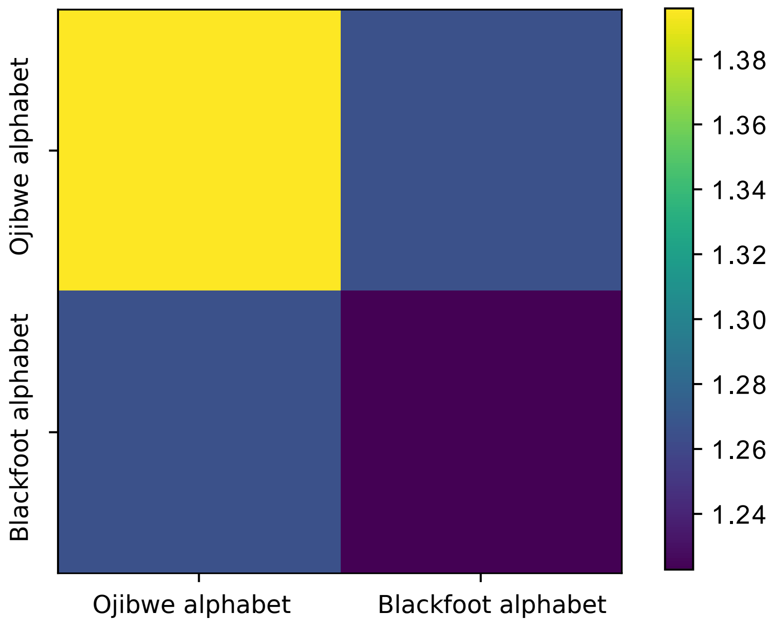


Fig. 4.6 Global absolute coregionalisation matrix of Ojibwe and Blackfoot alphabets.

4.3.2 All Alphabets

In our final experiment, we apply MOCGPs-AR in 50 alphabets in the original dataset. There are 50 outputs with a different number of classes in each output (for more detail on the number of classes in each output see Table 4.1). The total number of classes in the 50 outputs is 1623. We follow Lake et al. (2015) and split the 50 alphabets into two sets: a background set and an evaluation set, where the background set has 30 alphabets (with a total of 964 classes) and the evaluation set has 20 alphabets (with a total of 659 classes). In order to apply MOCGPs-AR in all 50 alphabets, we use a mini-batch size of nine data points for each output to train our model. The small mini-batch size has a negative impact on G-M and G-A so we only apply MOCGPs-AR in this experiment. We apply MOCGPs-AR for three different sets of alphabets: all alphabets, the background alphabets and the evaluation alphabets.

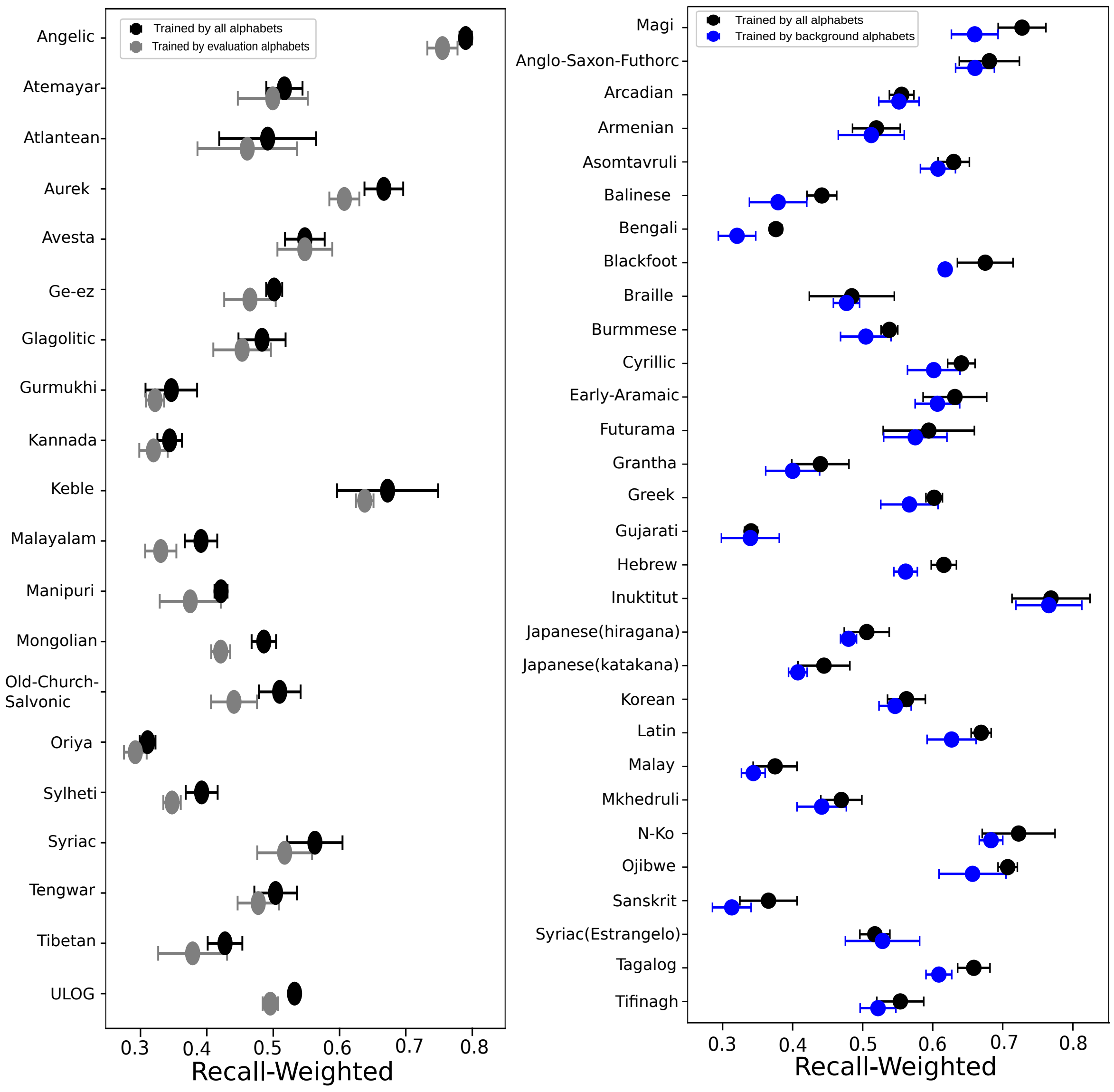


Fig. 4.7 Performance in cross-validation in evaluation alphabets (left) and background alphabets (right). In both diagrams, each circle is the mean of recall-weighted; the error bar is the standard deviation of recall-weighted.

In Figure 4.7, we empirically (50 different outputs and a total of 1623 classes of image data) show that MOCGPs-AR has better scalability than traditional multi-output Gaussian processes.

MOCGPs-AR has obtained the scalable property of MOGPs-AR so it can cope with large-scale classification through not only stochastic variational inference but also a softmax likelihood with Gumbel noise error in each output. Figure 4.7 also indicates that MOCGPs-AR obtains good performance even if we choose a small size of mini-batch (nine) and only a small number of classes (one) in each output since it captures both intra- and inter-output correlation.

In most predictions, our model trained with the data of all alphabets could outperform one trained with the data of part of the alphabets. For example, our model trained using all alphabets improves the recall-weighted from 0.6096 to 0.6692 for the Aurek alphabet, compared with one using evaluation alphabets for training. The extra alphabets can help our model improve its performance.

However, there are exceptions to the scenario in the last paragraph. For example, for the Syriac (Estrangelo) alphabet, the values of the recall-weighted 0.5174 is less than 0.5283 where only background alphabets are used to train our model. One likely reason is that our model assumes a correlation with all alphabets. However, the correlation with those alphabets may not exist or the correlation may hinder the predictive performance. Although there may be no correlation between the alphabets, our model assumes that the outputs share commonalities. If this assumption is not met, the model's performance could be worse than that of a single-output Gaussian process model. To avoid this scenario, we could increase the number of latent functions. This is because when the number of latent functions is greater than or equal to the number of outputs, MOGPs could automatically collapse into independent GPs in the absence of a correlation between each output (Li and Kontar, 2022).

4.4 Summary

In this chapter, we have introduced MOCGPs-AR, a novel stochastic scalable framework that allows the use of multi-output Gaussian processes for images. MOCGPs-AR is an extension of MOGPs-AR, replacing RBF-ARD with convolutional kernels (Van der Wilk et al., 2017) and substituting inducing points with inducing patches, while it can also tackle large-scale datasets and a large number of classes in each output due to inheriting the scalable property of MOGPs-AR. We have conducted experiments to show that MOCGPs-AR significantly improves the performance compared to MOGPs-AR and single output Gaussian processes in terms of image classification problems. In the next chapter, we focus on multi-output Gaussian processes for the hierarchically structured datasets. To this end, we drive a hierarchical framework of multi-output Gaussian processes where one Gaussian process is the mean of another and each has its own covariance functions. We conduct different experiments to display the property of our framework, which exploits the structure of the datasets, against other single- or multi-output Gaussian process models.

Chapter 5

Hierarchical Multi-output Gaussian Processes with Latent Information

Our main contribution in this chapter is to provide a new extension of the multi-output Gaussian process to cope with hierarchical datasets, known as *hierarchical multi-output Gaussian processes with latent variables (HMOGP-LV)*[1]. HMOGP-LV controls the correlation between each output through latent variables (Dai et al., 2017) and captures the hierarchical structure of the dataset through a hierarchical kernel (Hensman et al., 2013b). Since our inducing variables can keep the information of all replicas from outputs, our model can also predict missing replicas. When predicting a missing replica from one output, the inducing variables can use information from the corresponding replicas in other outputs.

In Bayesian statistics, hierarchical models take hierarchical data structure into consideration. The hierarchical dataset has the characteristics of top-down tree-like data architecture, i.e., Figure 5.1 (a) presents a hierarchical dataset that has two-layer tree-like data. We refer to the datasets in tree-leaf nodes on the same hierarchy as replicas since they inherit from the same parent node. In the real world, there are many datasets that have hierarchical structures, such as gene expression. Gene expression is the process in which information from a gene is used to effect a phenotype. In this dataset, each gene has eight replicas (Kalinka et al., 2010). In a hierarchical model, the prior distribution of some parameters depends on other parameters that also have their own prior distribution (Gelman et al., 2013). For example, Figure 5.1 (b) shows a hierarchical GP model that is the same as Hensman et al. (2013b), where R latent parameter functions in the second layer are drawn from a GP whose mean is a GP on the top layer. Simple non-hierarchical models often do not fit largely hierarchical data

[1]The model is implemented based on Python 3.7, mainly depending on the library GPflow 2.1.3 which was built on the library TensorFlow 2+. Our code is publicly available in the repository https://github.com/ChunchaoPeter/HMOGP-LV.

well with few parameters or are often overfitting with many parameters (Gelman et al., 2013). Whereas, hierarchical models can efficiently fit hierarchical data through enough parameters. Further, in hierarchical models, the hierarchical structure is considered, therefore, overfitting can be avoided (Gelman et al., 2013). Hierarchical models can model the correlation within the dataset, e.g., the correlation between each tree-leaf node data or replica. They can purposefully predict data in the corresponding replica (Gelman et al., 2013).

Hierarchical models in the area of GPs that have been studied in the literature (Damianou and Lawrence, 2013; Flaxman et al., 2015; Hensman et al., 2013b; Lawrence and Moore, 2007; Li and Chen, 2018). Lawrence and Moore (2007) introduced a hierarchical Gaussian process model for dimensionality reduction. Hensman et al. (2013b) built a hierarchical GP model through a new hierarchical kernel to handle gene expression (Kalinka et al., 2010). Damianou and Lawrence (2013) established a deep-layer model where each layer was based on a Gaussian process mapping. Flaxman et al. (2015) introduced a hierarchical model through a prior distribution over kernel hyperparameters and used MCMC for inference. Li and Chen (2018) proposed a hierarchical Gaussian process where they extracted latent features through the Gaussian process latent variable model from the input dataset and derived a Bayesian inference to generate output based on those latent features.

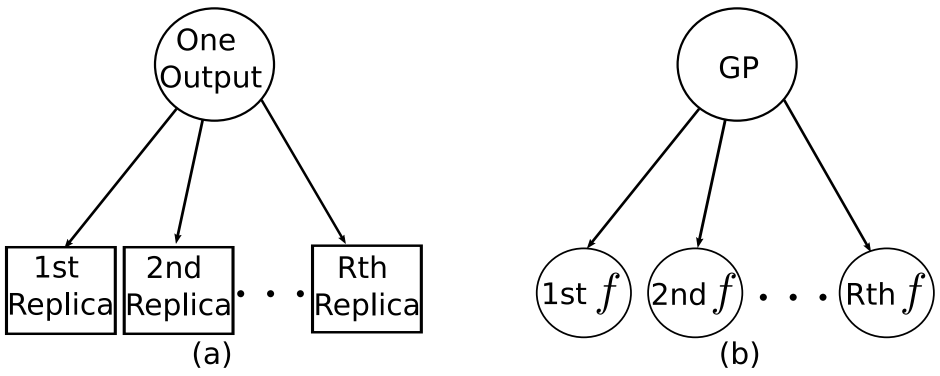


Fig. 5.1 (a): a hierarchical dataset has one output and the output has R replicas. (b): a hierarchical GP model

However, those GP models with the hierarchical structure mentioned above are not specific designs for a hierarchical dataset where there are multiple outputs and each output corresponds to a hierarchical structure. Therefore, they cannot sufficiently capture the correlation between each replica.

In this chapter, we introduce HMOGP-LV to deal with hierarchical datasets. Specifically, we show how HMOGP-LV can be used to handle two different types of hierarchical datasets: all different outputs have the same input data; all different outputs have different input data.

5.1 Related Work

As mentioned earlier, there is abundant research on hierarchical models using Gaussian processes (Damianou and Lawrence, 2013; Flaxman et al., 2015; Hensman et al., 2013b; Lawrence and Moore, 2007; Li and Chen, 2018). In this chapter, our model HMOGP-LV has a two-layer hierarchical kernel matrix. To construct this kernel matrix, we assume there is a latent parameter function drawn from a zero-mean GP in the first layer. The latent parameter function is the mean of another GP in the second layer, where other latent parameter functions follow the GP, which is the same as Hensman et al. (2013b). However, their model cannot predict missing replicas since they do not apply the inducing variables framework. In our model, the inducing variables have all information for all replicas so we can predict missing replicas from this information. Further, our model uses latent variables to capture the correlation between each output. Lawrence and Moore (2007) introduce a hierarchical model for unsupervised learning while our model is for supervised learning. The model proposed by Li and Chen (2018) assumes latent variables corresponding to each data point but our model assumes latent variables corresponding to each output. Deep Gaussian processes (Damianou and Lawrence, 2013) cannot capture the correlation between replicas since their kernel is not designed to support this. The hierarchical model derived by Flaxman et al. (2015) presents a prior distribution over kernel parameters. However, our hierarchical model builds through a prior distribution over latent parameter functions.

As mentioned in the introduction, multi-output Gaussian processes can improve prediction by exploiting the correlation between each output. Multi-output Gaussian processes mainly use the coregionalisation matrix B to control the correlation between each output, for example, the intrinsic coregionalisation model (ICM) in Section 2.7.1 and the linear model of coregionalisation (LMC) in Section 2.7.2 (Álvarez et al., 2012; Bonilla et al., 2008). The coregionalisation matrix B is a finite fixed matrix, which means that it can only represent the correlation between fixed outputs. However, our model uses a different method to exploit the correlation. To learn the correlation between each output, our model replaces the coregionalisation matrix B with a kernel matrix so that we can extend the dimension of our kernel matrix by adding latent variables. The methodology more relevant to ours is Dai et al. (2017) because we use the same method to build a kernel matrix to learn the correlation between each output; however, ours is different: HMOGP-LV introduces a hierarchical kernel matrix that captures the similarity between tree-like structures.

5.2 Methodology

In this section, we derive the hierarchical multi-output Gaussian processes with latent variables (HMOGP-LV). We first develop the HMOGP-LV to deal with different outputs where all outputs have the same input dataset, we then describe scalable variational inference for the model. We further define the predictive distribution for our model and finally we generalise HMOGP-LV to deal with different outputs where each output has its own input dataset.

We assume $\mathbf{y}(\mathbf{x})$ contains D different outputs:

$$\mathbf{y}(\mathbf{x}) = \left[\mathbf{y}_1^\top(\mathbf{x}), \mathbf{y}_2^\top(\mathbf{x}), \cdots, \mathbf{y}_D^\top(\mathbf{x})\right]^\top, \tag{5.1}$$

where $\mathbf{x} \in \mathbf{R}^v$ and each output represents a hierarchical structure. Since we have already considered the data points in each tree-leaf node as a replica, we assume each output has the same number of R replicas, for example,

$$\mathbf{y}_d(\mathbf{x}) = \left[y_d^1(\mathbf{x}), y_d^2(\mathbf{x}), \cdots, y_d^R(\mathbf{x})\right]^\top, \tag{5.2}$$

where $y_d^r(\mathbf{x})$ is the r-th replica in the d-th output evaluated at $\mathbf{x}$ where $r \in \{1, \cdots, R\}$ and $d \in \{1, \cdots, D\}$. For simplicity, we assume each replica has the same number of N data points. In fact, each replica can have a different number of data points. Each replica $y_d^r(\mathbf{x})$ can be modelled as a latent parameter function $f_d^r(\mathbf{x})$ corrupted with ε_d that follows a Gaussian noise with a zero mean and σ_d^2 variance:

$$y_d^r(\mathbf{x}) = f_d^r(\mathbf{x}) + \varepsilon_d, \quad f_d^r(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k_f\left(\mathbf{x}, \mathbf{x}'\right)\right), \quad \varepsilon_d \sim \mathcal{N}\left(\mathbf{0}, \sigma_d^2\right). \tag{5.3}$$

We take $\mathbf{X}_r = [\mathbf{x}_r^{(1)}, \cdots, \mathbf{x}_r^{(N)}]^\top \in \mathbf{R}^{N \times v}$ as a collection of all r-th input data points. $\mathbf{y}_d^r = [y_d^r\left(\mathbf{x}_r^{(1)}\right), \cdots, y_d^r\left(\mathbf{x}_r^{(N)}\right)]^\top \in \mathbf{R}^N$ denotes the vector of all data points in the r-th replica in the d-th output. The d-th input and output data are denoted as $\mathbf{X} = \{\mathbf{X}_r\}_{r=1}^R$ and $\mathbf{y}_d = \left\{\mathbf{y}_d^r\right\}_{r=1}^R$, respectively. The vector $\mathbf{y} = [\mathbf{y}_1^\top, \cdots, \mathbf{y}_D^\top]^\top$ refers to all outputs.

5.2.1 Hierarchical Multi-output Gaussian Processes with Latent Variables for the Same Input

In this section, we assume all outputs have the same input and our model is derived to deal with this dataset. To cope with all replicas in each output, we suppose there is an underlying function for the replicas. The underlying function $g(\mathbf{x})$ draws from a zero mean GP with covariance $k_g(\mathbf{x}, \mathbf{x}')$:

$$g(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k_g\left(\mathbf{x}, \mathbf{x}'\right)\right). \tag{5.4}$$

Then, we assume all latent parameter functions are drawn from a Gaussian process with mean $g(\mathbf{x})$ and covariance $k_f(\mathbf{x}, \mathbf{x}')$, where our model's hierarchical structure is the same as that of Hensman et al. (2013b). We thus obtain

$$g(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k_g\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.5}$$

$$f_d^r(\mathbf{x}) \sim \mathcal{GP}\left(g(\mathbf{x}), k_f\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.6}$$

$$y_d^r(\mathbf{x}) = f_d^r(\mathbf{x}) + \varepsilon_d. \tag{5.7}$$

The above functions show that all the parameter functions share the information for the input through kernel functions $k_g(\cdot,\cdot)$ and $k_f(\cdot,\cdot)$.

In order to replace the fixed coregionalisation matrix with a kernel matrix, we assume there is a continuous latent vector $\mathbf{h}_d$ that corresponds to an output $\mathbf{y}_d$ and the correlation between these latent vectors will be ultimately measured through a kernel matrix, where $\mathbf{h}_d \in \mathbf{R}^{Q_H}$ and the Q_H is pre-defined. The latent variable is extracted from observations via maximising marginal likelihood. Latent variables of all outputs are stacked in $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_D]^\top$ and each of them follows the same prior distribution, e.g. a normal distribution. Therefore, we obtain

$$g(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k_g\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.8}$$

$$f_d^r(\mathbf{x}) \sim \mathcal{GP}\left(g(\mathbf{x}), k_f\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.9}$$

$$y_d^r(\mathbf{x}) = f_d^r(\mathbf{x}, \mathbf{h}_d) + \varepsilon_d, \mathbf{h}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{5.10}$$

Figure 5.2 depicts the visualization of the plot of $\mathbf{h}_d$ in both 3-D and 2-D, assuming its dimension is two and it follows a bivariate normal distribution. The two axes at the bottom, ranging from -3 to 3, represent the two dimensions of $\mathbf{h}_d$ respectively. The vertical axis, ranging from 0 to 0.2, shows the probability density of the bivariate normal distribution.

There are many different ways to build our kernel based on the expression in Eq. (5.10). To explain the correlation between input and output, respectively, we build our kernel matrix through a Kronecker product as shown in Section 2.7.1.

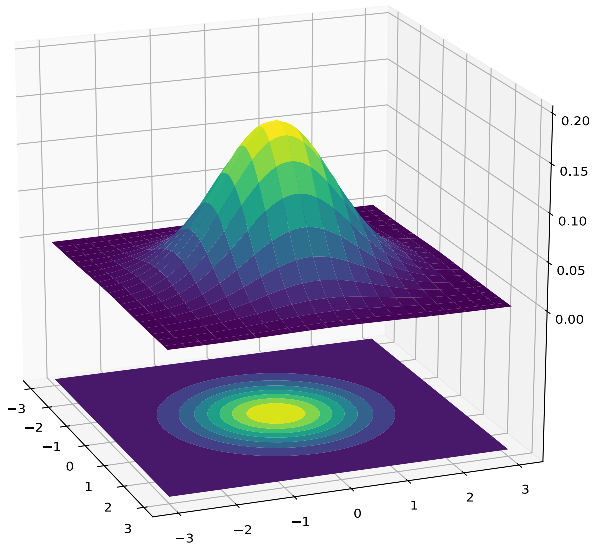


Fig. 5.2 Visualisation of a surface plot of $\mathbf{h}_d$ and projected a filled contour plot under it when $Q_H = 2$.

$$\begin{tikzcd}
\mathbf{H} \arrow[rr] & & \mathbf{K}_{\mathbf{ff}}^{H} \arrow[dr] & \\
& & & \mathbf{K}_{\mathbf{ff}} = \mathbf{K}_{\mathbf{ff}}^{H} \otimes \mathbf{K}_{\mathbf{ff}}^{X} \\
\mathbf{X} \arrow[r] & \mathrm{g}(\mathbf{X}) \arrow[r] & \mathbf{K}_{\mathbf{ff}}^{X} \arrow[ur] &
\end{tikzcd}$$

Fig. 5.3 How our kernel matrix is built. $\mathbf{K}_{\mathbf{ff}}^{X}$ contains the hierarchical structure of our model; $\mathbf{K}_{\mathbf{ff}}^{H}$ contains the correlation between each output

Figure 5.3 shows how we build the kernel matrix. We first build a kernel matrix for the outputs:

$$\mathbf{K}_{\mathbf{ff}}^{H} = \begin{pmatrix} K_{1,1}^{H} & \cdots & K_{1,D}^{H} \\ \vdots & \ddots & \vdots \\ K_{D,1}^{H} & \cdots & K_{D,D}^{H} \end{pmatrix}, \tag{5.11}$$

where $K_{i,j}^{H} = k_H\left(\mathbf{h}_i, \mathbf{h}_j\right)$ describes the correlation between i-th and j-th outputs and $k_H(\cdot,\cdot)$ is a kernel function. Now, we develop a kernel matrix for the inputs. Since there is a linearly hierarchical structure for our latent parameter functions, if two input points are from the same output with the same r-th replica, e.g., $\mathbf{x}_r^{(i)}$ and $\mathbf{x}_r^{(j)}$, then they follow a Gaussian distribution with a zero mean and covariance $k_f\left(\mathbf{x}_r^{(i)}, \mathbf{x}_r^{(j)}\right) + k_g\left(\mathbf{x}_r^{(i)}, \mathbf{x}_r^{(j)}\right)$. If two input points are from different replicas, e.g., $\mathbf{x}_r^{(i)}$ and $\mathbf{x}_{r'}^{(j)}$, they will follow a Gaussian distribution with a zero mean and covariance $k_g\left(\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)}\right)$. Then the kernel function for the input is

$$k_{\text{hierarchy}}\left(\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)}\right) = \begin{cases} k_g\left(\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)}\right) + k_f\left(\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)}\right) & r = r', \\ k_g\left(\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)}\right) & r \neq r', \end{cases} \tag{5.12}$$

where $\mathbf{x}_r^{(i)}, \mathbf{x}_{r'}^{(j)} \in \mathbf{X}$. We also consider $k_{\text{hierarchy}}(\cdot,\cdot)$ as a hierarchical kernel function that naturally fits the hierarchical data and captures dependencies within the data structure. Further, the hierarchical kernel matrix $\mathbf{K}_{\mathbf{ff}}^{X}$ is

$$\mathbf{K}_{\mathbf{ff}}^{X} = \begin{pmatrix} [k_g(\mathbf{X}_1,\mathbf{X}_1) + k_f(\mathbf{X}_1,\mathbf{X}_1)] & \dots & k_g(\mathbf{X}_1,\mathbf{X}_R) \\ \vdots & \ddots & \vdots \\ k_g(\mathbf{X}_R,\mathbf{X}_1) & \dots & [k_g(\mathbf{X}_R,\mathbf{X}_R) + k_f(\mathbf{X}_R,\mathbf{X}_R)] \end{pmatrix}. \tag{5.13}$$

Finally, the covariance matrix of our proposed model is defined as

$$\mathbf{K}_{\mathbf{ff}} = \mathbf{K}_{\mathbf{ff}}^{H} \otimes \mathbf{K}_{\mathbf{ff}}^{X}, \tag{5.14}$$

where $\otimes$ means the Kronecker product between matrices. Based on Eq. (5.14), we obtain our prior function and likelihood function:

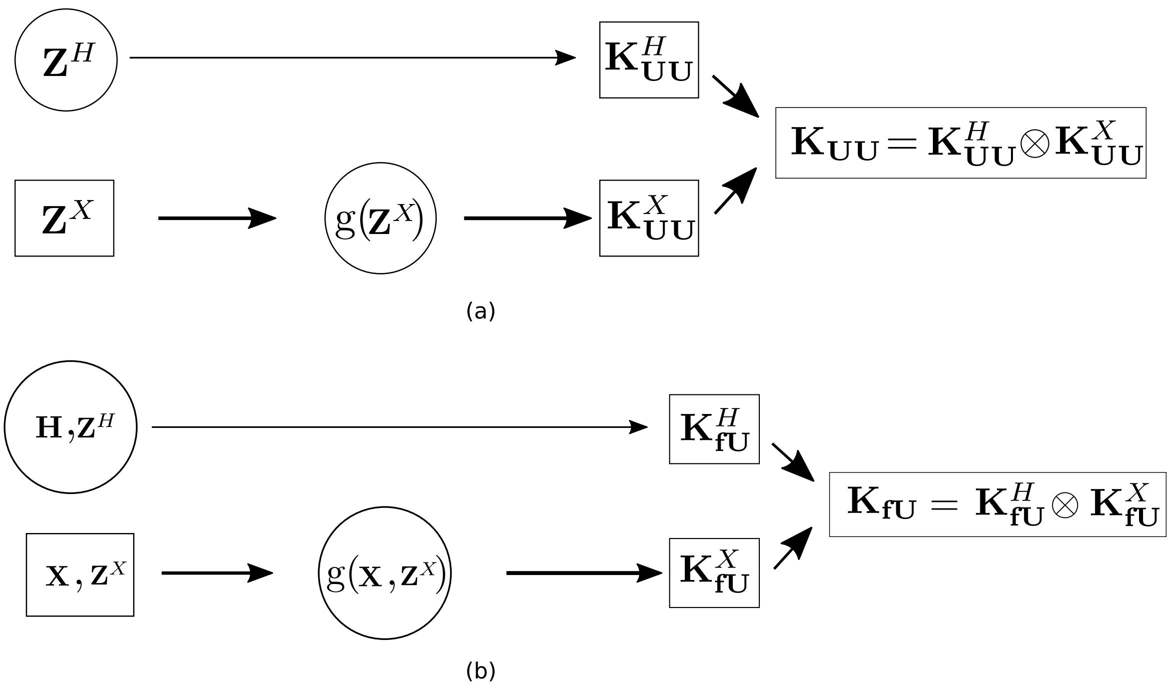
$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \mathbf{\Sigma}), \quad p(\mathbf{f} \mid \mathbf{X}, \mathbf{H}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{\mathbf{ff}}), \tag{5.15}$$

where $\mathbf{\Sigma} \in \mathbf{R}^{NRD \times NRD}$ is a diagonal matrix with $\sigma_d^2 = \sigma^2$ (here, we assume σ_d^2 is the same as σ^2 for all outputs for simplicity) and $\mathbf{f} = \left[\mathbf{f}_1^\top, \dots, \mathbf{f}_D^\top\right]^\top$ in which $\mathbf{f}_d = [f_d^1(\mathbf{X}_1), \dots, f_d^R(\mathbf{X}_R)]^\top \in \mathbf{R}^{NR}$. Thus, the corresponding marginal likelihood is

$$p(\mathbf{y} \mid \mathbf{X}) = \int p(\mathbf{y} \mid \mathbf{X}, \mathbf{f}, \mathbf{H})\, p(\mathbf{f})\, p(\mathbf{H})\, \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{H}. \tag{5.16}$$

5.2.2 Scalable Variational Inference

Since the integral of the marginal likelihood (5.16) of given latent variables is intractable, we derive the lower bound of the log marginal likelihood by inducing variables. Our method, which is based on Dai et al. (2017) using similar notations, can also deal with large-scale datasets.



$\mathbf{Z}^H$
$\mathbf{K}^H_{\mathbf{UU}}$
$\mathbf{K}_{\mathbf{UU}} = \mathbf{K}^H_{\mathbf{UU}} \otimes \mathbf{K}^X_{\mathbf{UU}}$
$\mathbf{Z}^X$
$g(\mathbf{Z}^X)$
$\mathbf{K}^X_{\mathbf{UU}}$
(a)
$\mathbf{H}, \mathbf{Z}^H$
$\mathbf{K}^H_{\mathbf{fU}}$
$\mathbf{K}_{\mathbf{fU}} = \mathbf{K}^H_{\mathbf{fU}} \otimes \mathbf{K}^X_{\mathbf{fU}}$
$\mathbf{X}, \mathbf{Z}^X$
$g(\mathbf{X}, \mathbf{Z}^X)$
$\mathbf{K}^X_{\mathbf{fU}}$
(b)

Fig. 5.4 (a): the way to build our kernel matrix for inducing variables, where $\mathbf{Z}^X$ and $\mathbf{Z}^H$ are associated with the inputs $\mathbf{X}$ and the latent variables $\mathbf{H}$, respectively; (b): the way to build our kernel matrix for between observations and inducing variables

We first introduce inducing variables $\mathbf{U} \in \mathbf{R}^{M_\mathbf{X} \times M_\mathbf{H}}$ and $\mathbf{U}_: = \text{vec}(\mathbf{U})$. The notation ":" represents the vectorisation of a matrix. We assume there is a prior distribution for $\mathbf{U}_:$: $p(\mathbf{U}_:) = \mathcal{N}(\mathbf{U}_: \mid \mathbf{0}, \mathbf{K}_{\mathbf{UU}})$. We also assume $\mathbf{K}_{\mathbf{UU}}$ has a similar format as Eq. (5.14): $\mathbf{K}_{\mathbf{UU}} = \mathbf{K}^H_{\mathbf{UU}} \otimes \mathbf{K}^X_{\mathbf{UU}}$ (see Figure 5.4 (a)). We obtain $\mathbf{K}^H_{\mathbf{UU}}$ using $k_H(\cdot,\cdot)$ evaluated at inducing outputs $\mathbf{Z}^H = \left[\mathbf{z}^H_1, \dots, \mathbf{z}^H_{M_\mathbf{H}}\right]^\top$, $\mathbf{z}^H_m \in \mathbf{R}^{Q_H}$. Similarly, we obtain $\mathbf{K}^X_{\mathbf{UU}}$ using kernel function $k_{\text{hierarchy}}(\cdot,\cdot)$ evaluated at inducing inputs $\mathbf{Z}^X$ where $\mathbf{Z}^X = \{\mathbf{Z}^X_r\}_{r=1}^R$. $\mathbf{Z}^X_r$ corresponds with the r-th replica and $\mathbf{Z}^X_r = \left[\mathbf{z}^X_{r,1}, \dots, \mathbf{z}^X_{r,M_r}\right]^\top$ in which $\mathbf{z}^X_{r,m} \in \mathbf{R}^v$, and M_r is the number of inducing input points in the r-th replica and $M_\mathbf{X} = M_r \times R$.

Like the inducing variables framework in Section 2.4, the conditional distribution of $\mathbf{f}$ here is:

$$p\left(\mathbf{f} \mid \mathbf{U}, \mathbf{Z}^X, \mathbf{Z}^H, \mathbf{X}, \mathbf{H}\right) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\mathbf{fU}}\mathbf{K}^{-1}_{\mathbf{UU}}\mathbf{U}_:, \mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fU}}\mathbf{K}^{-1}_{\mathbf{UU}}\mathbf{K}^\top_{\mathbf{fU}}\right), \tag{5.17}$$

where $\mathbf{K}_{\mathbf{fU}} = \mathbf{K}^H_{\mathbf{fU}} \otimes \mathbf{K}^X_{\mathbf{fU}}$ (see Figure 5.4 (b)). $\mathbf{K}^X_{\mathbf{fU}}$ is the cross-covariance computed between $\mathbf{X}$ and $\mathbf{Z}^X$ with $k_{\text{hierarchy}}(\cdot,\cdot)$; $\mathbf{K}^H_{\mathbf{fU}}$ is the cross-covariance computed between $\mathbf{H}$ and $\mathbf{Z}^H$ with

$k_H(\cdot,\cdot)$. Like the covariance matrix (5.13), we obtain

$$\mathbf{K}^X_{\mathbf{UU}} = \begin{pmatrix} \left[k_g\left(\mathbf{Z}^X_1,\mathbf{Z}^X_1\right)+k_f\left(\mathbf{Z}^X_1,\mathbf{Z}^X_1\right)\right] & \cdots & k_g\left(\mathbf{Z}^X_1,\mathbf{Z}^X_R\right) \\ \vdots & \ddots & \vdots \\ k_g\left(\mathbf{Z}^X_R,\mathbf{Z}^X_1\right) & \cdots & \left[k_g\left(\mathbf{Z}^X_R,\mathbf{Z}^X_R\right)+k_f\left(\mathbf{Z}^X_R,\mathbf{Z}^X_R\right)\right] \end{pmatrix}, \tag{5.18}$$

$$\mathbf{K}^X_{\mathbf{fU}} = \begin{pmatrix} \left[k_g\left(\mathbf{X}_1,\mathbf{Z}^X_1\right)+k_f\left(\mathbf{X}_1,\mathbf{Z}^X_1\right)\right] & \cdots & k_g\left(\mathbf{X}_1,\mathbf{Z}^X_R\right) \\ \vdots & \ddots & \vdots \\ k_g\left(\mathbf{X}_R,\mathbf{Z}^X_1\right) & \cdots & \left[k_g\left(\mathbf{X}_R,\mathbf{Z}^X_R\right)+k_f\left(\mathbf{X}_R,\mathbf{Z}^X_R\right)\right] \end{pmatrix}. \tag{5.19}$$

To approximate $\mathbf{f}$ and $\mathbf{H}$, we introduce variational posteriors $q(\mathbf{f} \mid \mathbf{U}_:, \mathbf{H}) = p(\mathbf{f} \mid \mathbf{U}_:, \mathbf{H})$, and $q(\mathbf{H})$. To obtain the optimal parameters and hyperparameters of our model, we can maximise the lower bound of $\log p(\mathbf{y})$. After calculation, we obtain (see appendix C.1 for more detail):

$$\mathcal{L} = \mathcal{F} - \mathrm{KL}(q(\mathbf{U}_:)\|p(\mathbf{U}_:)) - \mathrm{KL}(q(\mathbf{H})\|p(\mathbf{H})), \tag{5.20}$$

where we assume $q(\mathbf{U}_:) = \mathcal{N}\left(\mathbf{U}_: \mid \mathbf{M}_:, \mathbf{\Sigma}^{\mathbf{U}_:}\right)$ is another variational posterior distribution to approximate $p(\mathbf{U}_:)$, in which $\mathbf{M}_:$ and $\mathbf{\Sigma}^{\mathbf{U}_:}$ are variational parameters, and $\mathcal{F}$ (see appendix C.2 for more detail) is

$$\begin{aligned}\mathcal{F} = &-\frac{DRN}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\mathbf{y}^\top\mathbf{y} - \frac{1}{2\sigma^2}\mathrm{Tr}\left(\mathbf{K}^{-1}_{\mathbf{UU}}\Phi\mathbf{K}^{-1}_{\mathbf{UU}}\left(\mathbf{M}_:\mathbf{M}_:^\top + \mathbf{\Sigma}^{\mathbf{U}_:}\right)\right) \\ &+ \frac{1}{\sigma^2}\mathbf{y}^\top\Psi\mathbf{K}^{-1}_{\mathbf{UU}}\mathbf{M}_: - \frac{1}{2\sigma^2}\left(\psi - \mathrm{Tr}\left(\mathbf{K}^{-1}_{\mathbf{UU}}\Phi\right)\right),\end{aligned} \tag{5.21}$$

where $\mathrm{Tr}[\cdot]$ is a trace of a matrix, $\Phi = \left\langle\mathbf{K}^\top_{\mathbf{fU}}\mathbf{K}_{\mathbf{fU}}\right\rangle_{q(\mathbf{H})}$, $\Psi = \left\langle\mathbf{K}_{\mathbf{fU}}\right\rangle_{q(\mathbf{H})}$ and $\psi = \mathrm{Tr}\left\langle\mathbf{K}_{\mathbf{ff}}\right\rangle_{q(\mathbf{H})}$. Notice the computational complexity of the lower bound is dominated by the product $\mathbf{K}^\top_{\mathbf{fU}}\mathbf{K}_{\mathbf{fU}}$ that is $\mathcal{O}\left(NDRM^2_{\mathbf{X}}M^2_{\mathbf{H}}\right)$.

5.2.3 More Efficient Formulation

In this subsection, we reduce the computational complexity by exploiting the Kronecker product decomposition based on ideas, as in Dai et al. (2017). To fully utilise its properties, we assume there is a Kronecker product decomposition of the covariance matrix of $q(\mathbf{U}_:)$, $\mathbf{\Sigma}^{\mathbf{U}_:} = \mathbf{\Sigma}^{H_:} \otimes \mathbf{\Sigma}^{X_:}$ and this format can reduce variational parameters from $M^2_{\mathbf{X}}M^2_{\mathbf{H}}$ to $M^2_{\mathbf{X}} + M^2_{\mathbf{H}}$

in $q(\mathbf{U}_{:})$. We also reformulate Φ, Ψ, ψ as

$$\Phi = \mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}_{\mathbf{fU}}^{\top}\mathbf{K}_{\mathbf{fU}}\right] = \Phi^{H}\otimes\Phi^{X}, \tag{5.22}$$

$$\Phi^{H} = \mathbb{E}_{q(\mathbf{H})}\left[\left(\mathbf{K}_{\mathbf{fU}}^{H}\right)^{\top}\mathbf{K}_{\mathbf{fU}}^{H}\right], \tag{5.23}$$

$$\Phi^{X} = \left(\mathbf{K}_{\mathbf{fU}}^{X}\right)^{\top}\mathbf{K}_{\mathbf{fU}}^{X}, \tag{5.24}$$

$$\Psi = \mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}_{\mathbf{fU}}^{H}\otimes\mathbf{K}_{\mathbf{fU}}^{X}\right] = \mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}_{\mathbf{fU}}^{H}\right]\otimes\mathbf{K}_{\mathbf{fU}}^{X} = \Psi^{H}\otimes\mathbf{K}_{\mathbf{fU}}^{X}, \tag{5.25}$$

$$\psi = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}_{\mathbf{ff}}\right]\right) = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}_{\mathbf{ff}}^{H}\otimes\mathbf{K}_{\mathbf{ff}}^{X}\right]\right). \tag{5.26}$$

Using the property of the Kronecker product decomposition, we obtain a new format of the lower bound (for more detail see appendix C.3.1):

$$\begin{aligned}
\mathcal{F} = &-\frac{NDR}{2}\log 2\pi\sigma^{2} - \frac{1}{2\sigma^{2}}\mathbf{y}^{\top}\mathbf{y} \\
&-\frac{1}{2\sigma^{2}}\mathrm{Tr}\left(\mathbf{M}^{\top}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi^{X}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi^{H}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\right) \\
&-\frac{1}{2\sigma^{2}}\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi^{H}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\mathbf{\Sigma}^{H_{:}}\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi^{X}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{\Sigma}^{X_{:}}\right) \\
&+\frac{1}{\sigma^{2}}\mathbf{y}^{\top}\left(\mathbf{K}_{\mathbf{fU}}^{X}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\left(\Psi^{H}\right)^{\top}\right)_{:} - \frac{1}{2\sigma^{2}}\psi \\
&+\frac{1}{2\sigma^{2}}\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi^{H}\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi^{X}\right).
\end{aligned} \tag{5.27}$$

Similarly, the KL-divergence between $q(\mathbf{U}_{:})$ and $p(\mathbf{U}_{:})$ can also benefit from the above decomposition (see appendix C.3.1 for more detail):

$$\begin{aligned}
\mathrm{KL}\left\{q\left(\mathbf{U}_{:}\right)\mid p\left(\mathbf{U}_{:}\right)\right\} = &\frac{1}{2}\left(M_{\mathbf{X}}\log\frac{\left|\mathbf{K}_{\mathbf{UU}}^{H}\right|}{\left|\mathbf{\Sigma}^{H_{:}}\right|} + M_{\mathbf{H}}\log\frac{\left|\mathbf{K}_{\mathbf{UU}}^{X}\right|}{\left|\mathbf{\Sigma}^{X_{:}}\right|} + \mathrm{Tr}\left(\mathbf{M}^{\top}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\right)\right. \\
&\left.+\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\mathbf{\Sigma}^{H_{:}}\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{\Sigma}^{X_{:}}\right) - M_{\mathbf{H}}M_{\mathbf{X}}\right).
\end{aligned} \tag{5.28}$$

The computational complexity of $\mathcal{L}$ is led by the product $\left(\mathbf{K}_{\mathbf{fU}}^{H}\right)^{\top}\mathbf{K}_{\mathbf{fU}}^{H}$ and $\left(\mathbf{K}_{\mathbf{fU}}^{X}\right)^{\top}\mathbf{K}_{\mathbf{fU}}^{X}$ with a cost of $\mathcal{O}\left(DM_{\mathbf{H}}^{2}\right)$ and $\mathcal{O}\left(NRM_{\mathbf{X}}^{2}\right)$, respectively, which is more efficient than Eq. (5.21). Further, we can extend the lower bound with the mini-batch method to improve its scalability (Moreno-Muñoz et al., 2018).

5.2.4 Prediction

In this subsection, we derive the predictive distribution of HMOGP-LV. For existing outputs and a test set of inputs $\mathbf{X}^*$, we have $q(\mathbf{H})$ so we obtain:

$$q\left(\mathbf{f}^* \mid \mathbf{X}^*\right) = \int q\left(\mathbf{f}^* \mid \mathbf{X}^*, \mathbf{H}\right) q(\mathbf{H}) \mathrm{d}\mathbf{H}, \tag{5.29}$$

where

$$\begin{aligned} q\left(\mathbf{f}^* \mid \mathbf{X}^*, \mathbf{H}\right) &= \int p\left(\mathbf{f}^* \mid \mathbf{U}, \mathbf{X}^*, \mathbf{H}\right) q\left(\mathbf{U}_:\right) \mathrm{d}\mathbf{U}_: \\ &= \mathcal{N}\left(\mathbf{f}^* \mid \mathbf{K}_{\mathbf{f}^*\mathbf{U}} \mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1} \mathbf{M}_:, \mathbf{K}_{\mathbf{f}^*\mathbf{f}^*} - \mathbf{K}_{\mathbf{f}^*\mathbf{U}} \mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1} \mathbf{K}_{\mathbf{f}^*\mathbf{U}}^{\top} + \mathbf{K}_{\mathbf{f}^*\mathbf{U}} \mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1} \boldsymbol{\Sigma}^{\mathbf{U}_:} \mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1} \mathbf{K}_{\mathbf{f}^*\mathbf{U}}^{\top}\right), \end{aligned} \tag{5.30}$$

with $\mathbf{K}_{\mathbf{f}^*\mathbf{f}^*} = \mathbf{K}_{\mathbf{f}^*\mathbf{f}^*}^{H} \otimes \mathbf{K}_{\mathbf{f}^*\mathbf{f}^*}^{X}$ and $\mathbf{K}_{\mathbf{f}^*\mathbf{U}} = \mathbf{K}_{\mathbf{f}^*\mathbf{U}}^{H} \otimes \mathbf{K}_{\mathbf{f}^*\mathbf{U}}^{X}$. Eq. (5.29) is intractable, however, we can obtain the first and second moment of $\mathbf{f}^*$ in $q\left(\mathbf{f}^* \mid \mathbf{X}^*\right)$ (Titsias and Lawrence, 2010).

5.3 Hierarchical Multi-output Gaussian Processes with Latent Variables for Different Inputs

In the above section, we have derived a model that can deal with all different outputs with the same input dataset, however, in the real world, each output normally has its own input. In this section, we generalise our model to deal with a dataset where all outputs have different inputs.

We can also think that we have a large set of inputs and each output associated with a subset of the inputs is observed. For d-th input, the input data with replicated data is $\mathbf{X}_d = \left\{\mathbf{X}_{d,r}\right\}_{r=1}^{R}$, where $\mathbf{X}_{d,r} = [\mathbf{x}_{d,r}^{(1)}, \cdots, \mathbf{x}_{d,r}^{(N_d)}]^{\top}$. There is a different noise variance σ_d^2 for each output. Similarly, we obtain the lower bound of the log marginal likelihood (see appendix C.1 for more detail):

$$\mathcal{L} = \mathcal{F} - \mathrm{KL}(q(\mathbf{U}_:) \| p(\mathbf{U}_:)) - \mathrm{KL}(q(\mathbf{H}) \| p(\mathbf{H})), \tag{5.31}$$

where $\mathcal{F}$ is reformulated as (for more detail see appendix C.4):

$$\mathcal{F} = \sum_{d=1}^{D} -\frac{N_d R}{2}\log 2\pi\sigma_d^2 - \frac{1}{2\sigma_d^2}\mathbf{y}_d^\top \mathbf{y}_d + \frac{1}{\sigma_d^2}\mathbf{y}_d^\top \Psi_d \mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:}$$
$$-\frac{1}{2\sigma_d^2}\left(\psi_d - \mathrm{Tr}\left[\mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d\right]\right) - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d \mathbf{K}_{\mathbf{UU}}^{-1}\left(\mathbf{M}_{:}\mathbf{M}_{:}^\top + \boldsymbol{\Sigma}^{\mathbf{U}_{:}}\right)\right], \tag{5.32}$$

where $\Phi_d = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]$, $\Psi_d = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]$ and $\psi_d = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}\right]\right)$. The two KL divergence parts in the lower bound (5.31) are still the same since they do not depend on input and output datasets. The product $\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}$ with $\mathcal{O}(N_d R M_{\mathbf{X}}^2 M_{\mathbf{H}}^2)$ plays a key role in the computational complexity of the lower bound (5.31).

Eq. (5.32) is different from Eq. (5.21). First, Eq. (5.32) can use a different noise variance for each output. Further, in practice, it is computationally more expensive than Eq. (5.21) since we need to calculate the expectations of the covariance matrix Φ_d, Ψ_d, ψ_d for each output. Finally, Eq. (5.32) allows the use of different input datasets.

5.3.1 More Efficient Formulation

Similar to subsection 5.2.3, we reformulate Φ_d, Ψ_d, ψ_d as

$$\Phi_d = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right] = \mathbb{E}_{q(\mathbf{h}_d)}\left[\left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)^\top \left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)\right] = \Phi_d^H \otimes \Phi_d^X, \tag{5.33}$$
$$\Phi_d^H = \mathbb{E}_{q(\mathbf{h}_d)}\left[\left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H\right)^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H\right], \tag{5.34}$$
$$\Phi_d^X = \left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X, \tag{5.35}$$
$$\Psi_d = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right] = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H\right] \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X = \Psi_d^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X, \tag{5.36}$$
$$\psi_d = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}\right]\right) = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}^X\right]\right). \tag{5.37}$$

We also reduce the computational complexity by using the property of the Kronecker product decomposition (for more detail see appendix C.3.2):

$$
\begin{aligned}
\mathcal{F} = \sum_{d=1}^{D} & -\frac{N_d R}{2} \log 2\pi\sigma_d^2 - \frac{1}{2\sigma_d^2} \mathbf{y}_d^\top \mathbf{y}_d \\
& - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left(\mathbf{M}^\top \left(\mathbf{K}_{\mathbf{UU}}^X\right)^{-1} \Phi_d^X \left(\mathbf{K}_{\mathbf{UU}}^X\right)^{-1} \mathbf{M} \left(\mathbf{K}_{\mathbf{UU}}^H\right)^{-1} \Phi_d^H \left(\mathbf{K}_{\mathbf{UU}}^H\right)^{-1} \right) \\
& - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^H\right)^{-1} \Phi_d^H \left(\mathbf{K}_{\mathbf{UU}}^H\right)^{-1} \mathbf{\Sigma}^{H:} \right) \mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^X\right)^{-1} \Phi_d^X \left(\mathbf{K}_{\mathbf{UU}}^X\right)^{-1} \mathbf{\Sigma}^{X:} \right) \\
& + \frac{1}{\sigma_d^2} \mathbf{y}_d^\top \left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X \left(\mathbf{K}_{\mathbf{UU}}^X\right)^{-1} \mathbf{M} \left(\mathbf{K}_{\mathbf{UU}}^H\right)^{-1} \left(\Psi_d^H\right)^\top \right)_{:} - \frac{1}{2\sigma_d^2} \psi_d \\
& + \frac{1}{2\sigma_d^2} \mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^H\right)^{-1} \Phi_d^H \right) \mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^X\right)^{-1} \Phi_d^X \right).
\end{aligned}
\tag{5.38}
$$

The computational complexity of the lower bound (5.31) is mainly controlled by $\left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X$ with $\mathcal{O}\left(N_d R M_{\mathbf{X}}^2\right)$. Like Moreno-Muñoz et al. (2018), we can also extend $\mathcal{L}$ (5.31) by applying the mini-bath method to make our model more scalable.

5.4 Experiments

In this section, we evaluate HMOGP-LV in both synthetic and real datasets. We show that HMOGP-LV outperforms other single- and multi-output Gaussian process models with regard to two evaluation metrics to evaluate the predictive accuracy in regression problems: normalised mean square error (NMSE) and negative log predictive density (NLPD). In terms of both NMSE and NLPD, smaller values lead to a better model.

Baselines: In terms of hierarchically structured datasets, we compare our model with three GP models that have a hierarchical kernel matrix like ours. We also compare our model with two multi-output GP models, one of which has a finite coregionalisation matrix; the other has a coregionalisation matrix with a kernel matrix like ours. The compared models are: 1) A Gaussian process with a hierarchical kernel matrix (Hensman et al., 2013b) (**HGP**); 2) A Gaussian process the same as HGP except that it has inducing variables like section 5.2.2 (**HGPInd**); 3) A Gaussian process with a deep hierarchical structure (three-layer) (Hensman et al., 2013b) (**DHGP**):

$$g(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k_g\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.39}$$

$$e_d(\mathbf{x}) \sim \mathcal{GP}\left(g(\mathbf{x}), k_e\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.40}$$

$$f_d^r(\mathbf{x}) \sim \mathcal{GP}\left(e_d(\mathbf{x}), k_f\left(\mathbf{x}, \mathbf{x}'\right)\right), \tag{5.41}$$

$$y_d^r(\mathbf{x}) = f_d^r(\mathbf{x}) + \varepsilon, \tag{5.42}$$

where $e_d(\mathbf{x})$ follows a Gaussian process with mean $g(\mathbf{x})$ and covariance $k_e(\mathbf{x}, \mathbf{x}')$, $\varepsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\right)$; 4) A standard linear model of coregionalisation that has a finite coregionalisation matrix (for more detail see Chapter 2.7.2) (**LMC**); and 5) A multi-output Gaussian process model that uses latent information to refer to the correlation between each output (Dai et al., 2017) (**LVMOGP**). We also compared our method to a Neural Network (NN), with 2 layers of 200 units and a ReLU activation, to handle a single output. HGP and HGPInd can only handle single outputs and each output has its own replicas. DHGP can handle multiple outputs, and each output has its own replicas, together. LMC and LVMOGP can also handle multiple outputs, where we stack all replicas in the same output in a vector as one output, i.e., the d-th output is $\mathbf{y}_d = \left[(\mathbf{y}_d^1)^\top, \cdots, (\mathbf{y}_d^R)^\top\right]^\top \in \mathbf{R}^{NR}$. We use the Adam optimiser for maximising the lower bound of log marginal likelihood (i.e., $\mathcal{L}$ in Eq. (5.31)) with a 0.01 learning rate (Kingma and Ba, 2014) to train HMOGPLV with 10,000 iterations. We also use the Adam optimiser with the same parameters setting to train LMC. Other models are trained with 10,000 iterations using the L-BFGS-B algorithm in SciPy (Virtanen et al., 2020).

Computational Complexity: We compare the computational complexity of our model with other models. For simplicity, we assume all the outputs have the same input, so the total number of data points is NR. HMOGPLV has the same computational complexity as LVMOGP since our model is derived from it, i.e., $\mathcal{O}\left(\max\left(NR, M_{\mathbf{H}}\right)\max\left(D, M_{\mathbf{X}}\right)\max\left(M_{\mathbf{H}}, M_{\mathbf{X}}\right)\right)$ (Dai et al., 2017). The computational complexity of LMC is $\mathcal{O}\left(QM^3 + DNRQM^2\right)$ (see section 2.7.2) when using mini-batch. We also could extend our model through mini-batch learning. The computational complexities of HGP (Hensman et al., 2013b) and HGPInd are $\mathcal{O}\left((NR)^3\right)$ and $\mathcal{O}\left(NR(M_{\mathbf{H}}M_{\mathbf{X}})^2\right)$, respectively. The DHGP has computational complexity $\mathcal{O}\left((DNR)^3\right)$ and can be reduced to $\mathcal{O}\left((ND)^3\right)$ by summing each kernel matrix corresponding to each replica together rather than a large matrix corresponding to all replicas (see Hensman et al. (2013b) for more detail).

Evaluation Metrics: To measure predictive accuracy, we choose two evaluation metrics in this chapter: one is the normalised mean square error (NMSE) which takes account of the predictive mean; the other is the negative log predictive density (NLPD) which takes both predictive mean and predictive variance into account. The following are the two metrics:

$$\text{NMSE} = \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_{test})^2}, \tag{5.43}$$

$$\text{NLPD} = \frac{1}{2}\frac{1}{N}\sum_{i=1}^{N}\left(\left(\frac{y_i - \hat{y}_i}{\hat{\sigma}_i}\right)^2 + \log\hat{\sigma}_i^2 + \log 2\pi\right), \tag{5.44}$$

where $\hat{y}_i$ and $\hat{\sigma}_i^2$ are the predictive mean and variance for the test data point i and y_i is the actual test value for that instance. The average output value for test data is $\bar{y}_{test}$.

All experiments were performed on a Dell PowerEdge C6320 with an Intel Xeon E5-2630 v3 at 2.40 GHz and 64GB of RAM. We take three repetitions for each experiment. Regarding no missing replicas' experiments, we take 50% of the data points in each replica for training data and the other 50% as test data for all experiments. HGP and DHGP do not have inducing variables. Q_H is set as two for HMOGP-LV and LVMOGP in all experiments.

5.4.1 Synthetic Data

In this subsection, we compare the performance of HMOGP-LV with other models on two synthetic hierarchical datasets: (a) the first dataset where for each replica in the output, part of the data points in each replica are used as training data and the task is to predict the remaining part per replica (see Figure 5.5 (a)); and (b) the second dataset where for each output, we predict a single missing replica using the remaining whole replicas as training data (see Figure 5.5 (b)). Regarding the parameter settings, HMOGP-LV, HGPInd, LVMOGP, and LMC use $M_{\mathbf{X}}$=3 three in each output for the first synthetic datasets. For the second dataset, we need to predict missing replicas so we choose $M_{\mathbf{X}}$ as six. HMOGP-LV and LVMOGP take $M_{\mathbf{H}}$ as five for those two datasets.

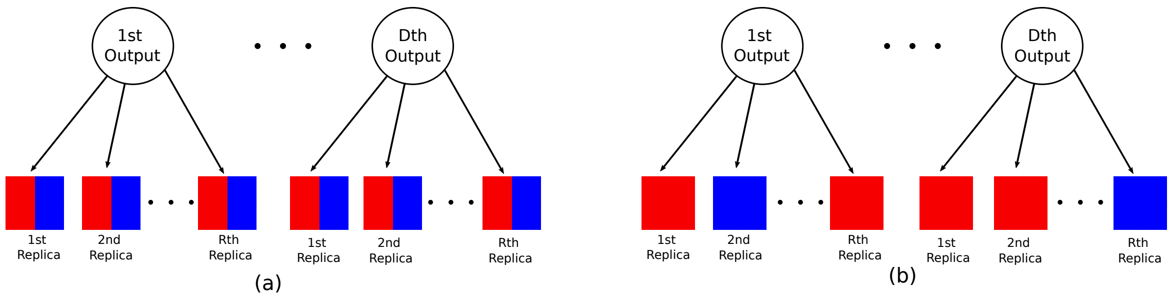


Fig. 5.5 (a): red represents training data points and blue represents test data points in the first dataset. (b): red represents training data points and blue represents test data points in the second dataset

5.4.1.1 First Synthetic Dataset

To show that our model can exploit the correlation in hierarchical data structures and correlation between each output, we generated our synthetic data by sampling from a Gaussian process with zero mean and covariance in Eq. (5.14). The kernel function is combined by two kernels: kernel $k_H(\cdot,\cdot)$ for outputs (two-dimensional space) Kronecker product with a hierarchical kernel. There are two kernels in the hierarchical kernel: $k_g(\cdot,\cdot)$ is a Matérn 32 kernel (2.16) having 1.0 length scale and 0.1 variance; $k_f(\cdot,\cdot)$ is another Matérn 32 kernel (2.16) having 1.0 length scale and 1.0 variance. Each output also has a different input dataset. Further, an observation noise with a variance of 0.02 is added to the dataset. The generated synthetic dataset has 50 outputs, each with three replicas where each replica has 10 data points. For example, in Figure 5.6, the first row shows the available data (training and test datasets) for the third output. In the first row, there are three graphs representing the first, second and third replicas in the third output, respectively.

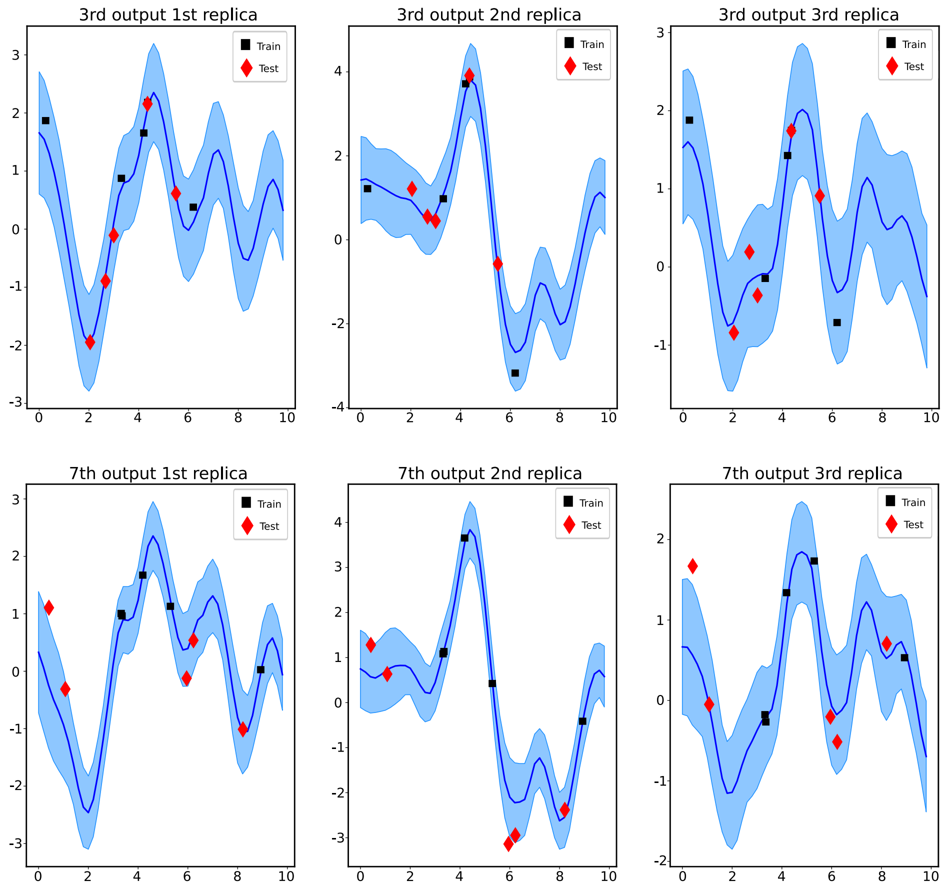


Fig. 5.6 Mean predictive curves associated with their 95% credible intervals for the third output (top row) and seventh output (bottom row) with three replicas each, coming from the synthetic dataset. Locations of training points (in black) and testing points (in red) are specific to each output.

Figure 5.6 shows that HMOGP-LV can make excellent predictions on the test dataset even if we only have a few training data points. For example, our model efficiently predicts the result of the test data points in the first replica in the third output and the second replica in the seventh output (see Figure 5.6). This is because HMOGP-LV can not only exploit the correlation intra- and inter-output by using latent variables but can also capture the hierarchical information from the dataset by using the hierarchical kernel matrix. The top and bottom rows show plots of the third and seventh output with three different replicas respectively, where the black and red data points are training and test data points.

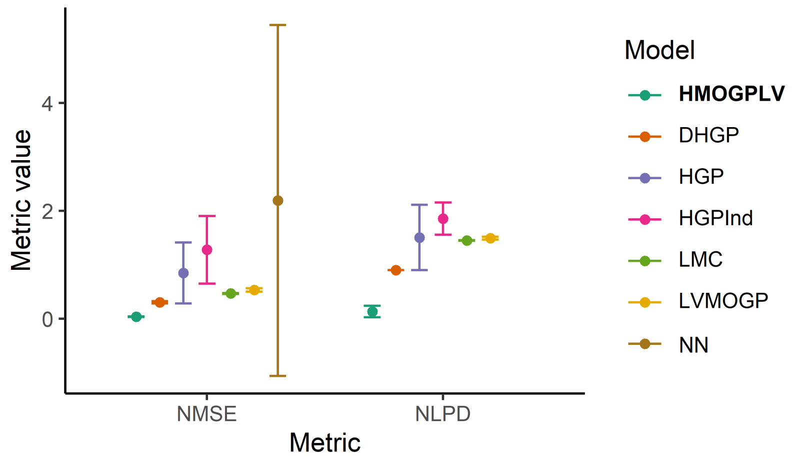


Fig. 5.7 Prediction performances (mean ± standard deviation) for the first synthetic dataset. For both NMSE and NLPD values, the lower the better.

HMOGP-LV outperforms single-output GPs (HGP and HGPInd). These single-output GPs can use their hierarchical kernel matrix to exploit the nature of the hierarchical dataset. Nevertheless, they cannot use other outputs to boost performance. In our model, we can use the latent variables to transfer knowledge between each output to improve performance. The HGP and HGPInd cannot capture any correlation between each output, leading to higher variability across different folds and resulting in large error bars (Figure 5.7). Regarding NN, it presents lower performances in terms of RMSE and noticeably high variability in results. Moreover, NN does not provide uncertainty quantification and cannot be evaluated in terms of NLPD.

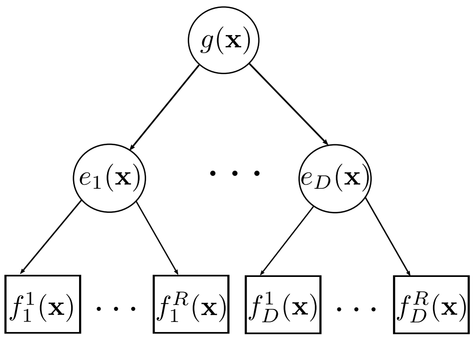


Fig. 5.8 A DHGP model

In terms of multi-output Gaussian processes, our model outperforms other models in both NMSE and NLPD in this synthetic dataset. Neither LVMOGP nor LMC performs well since they are non-hierarchical models that cannot capture the hierarchical structure from the dataset. Our model and DHGP are suitable for the hierarchical dataset. DHGP can deal with multi-output datasets where it assumes the multi-output datasets have three-layer tree-like data (see Figure 5.8). It has a three-layer hierarchical kernel, where the top layer (Eq. (5.39)) can capture the correlation between different outputs and the other two-layer (Eq. (5.40) and (5.41)) capture the correlation between replicas. However, the top layer (Eq. (5.39)) with a few hyper-parameters could not efficiently capture the correlation among outputs. By employing latent variables via a kernel function, our model uses more hyper-parameters and parameters than DHGP to exploit the correlation.

5.4.1.2 Second Synthetic Dataset

To show the ability of HMOGP-LV in the missing replicas' dataset, we designed a synthetic dataset with a missing replica in each output. Similar to the dataset above, we generate 50 outputs and each output has four replicas where each replica has 10 data points. In each output, we assume there is one missing replica, therefore, there are three replicas for training and we need to predict the missing replica for each output. For example, in Figure 5.9, the black points mean the training data points and the red points mean the missing replica data points. In the 14th output, the first, third and fourth replica are observed, whereas the second replica is missing. In this case, the aim of our model is to predict the second replica in the 14th output.

Figure 5.9 shows our model's performance in the 14th, 24th and 40th outputs with missing replicas. The first row shows our model has an excellent prediction for the missing replica in the 14th output. This is because the missing replica has a strong correlation with replicas in all outputs. Our model captures the correlation and can transfer it using our inducing variables when we predict missing replicas.

We compare our model against others in the two evaluation metrics, where the NMSE and NLPD are computed for all missing replicas in all the outputs and HMOGP-LV is superior to others (see Figure 5.10). HGP cannot make predictions for missing replicas since they cannot be trained without all replicas. HGPInd uses other replicas in the same output to obtain the information for the missing replica and all the information kept in the inducing points. However, it cannot use other outputs as extra information whereas our model can fully leverage the information between each output. Both LVMOGP and LMC can predict missing replicas since they do not distinguish replicas in each output that has a hierarchical structure. Nevertheless, HMOGP-LV can keep information of all replicas in inducing variables that can improve the performance of prediction.

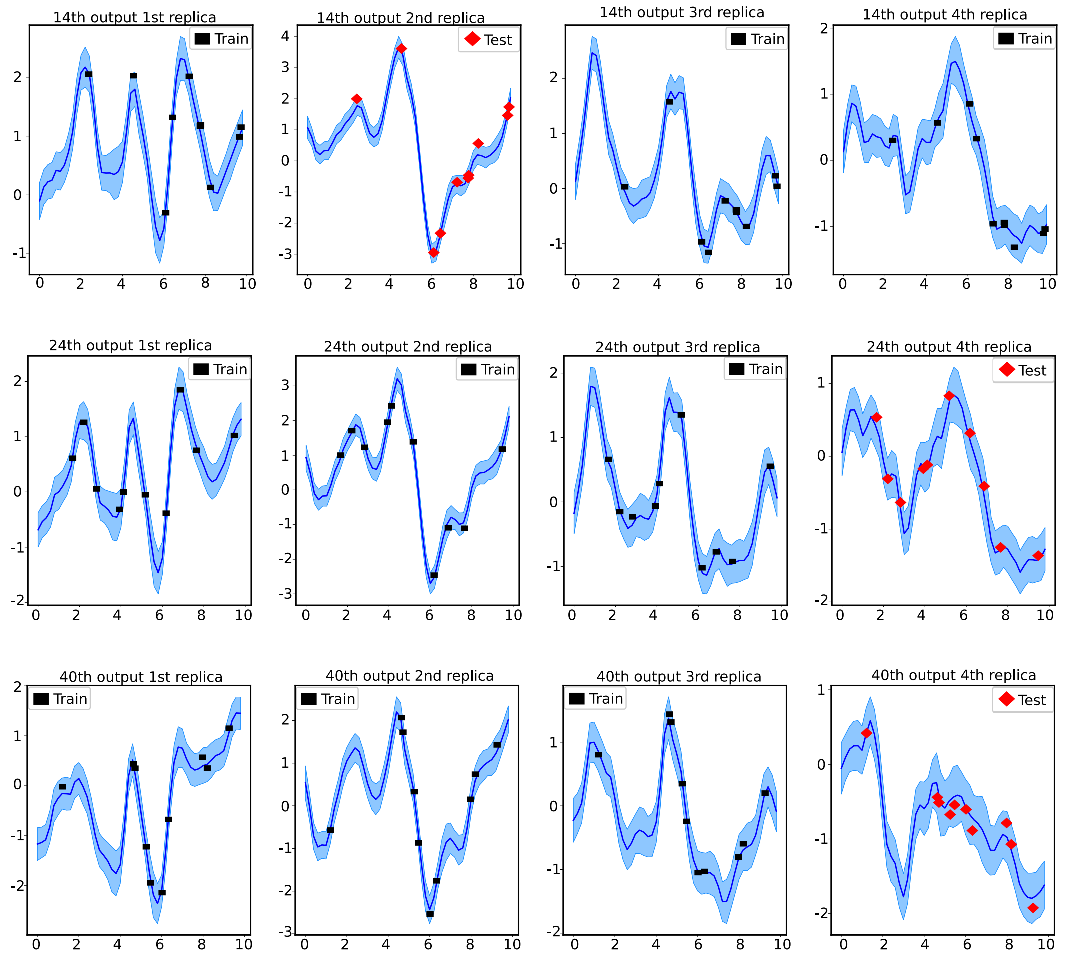


Fig. 5.9 Top row: the result of the 14th output with four replicas; Middle row: the result of the 24th output with four replicas; Bottom row: the result of the 40th output with four replicas. The black and red colour represents the train and test data points, respectively.

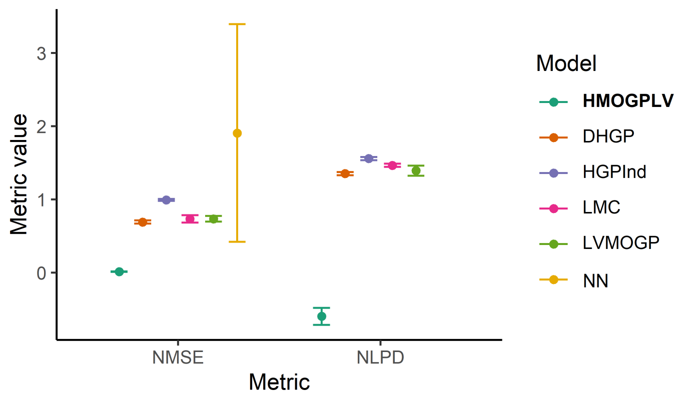


Fig. 5.10 Prediction performances (mean ± standard deviation) for the second synthetic data with one missing replica in each output. For both NMSE and NLPD values, the lower the better.

5.4.1.3 Evaluation the Dimension of latent variables

To explore the influence of Q_H, we generated synthetic datasets by sampling from a Gaussian process with zero mean and covariances. This covariance function is a combination of two kernels: $k_H(\cdot,\cdot)$ for outputs (two-dimensional space) Kronecker-times a hierarchical kernel. Two kernels are also involved in the hierarchical kernel design: $k_g(\cdot,\cdot)$, which is assumed to be Matérn(3/2) with 1.0 lengthscale and 0.1 variance; and $k_f(\cdot,\cdot)$ defined as another Matérn(3/2) kernel with 1.0 lengthscale and 1.0 variance. Each output is generated from a specific input set. In addition, a Gaussian noise term with a 0.02 variance is added to each data sample. One synthetic dataset consists of 10 outputs with five replicas each, while each replica comprises 10 data points.

Figure 5.11 (Left) shows the performance of HMOGP-LV, in the synthetic data based on different dimensions of latent vectors in terms of both NMSE and NLPD. The performance in both NMSE and NLPD will get worse as the dimension of latent vectors increases. This may be because when the dimension of $\mathbf{h}_d$ increases, the term $k_H(\mathbf{h}_i,\mathbf{h}_j)$ cannot adequately capture the correlation between i-th and j-th output. Our model performs similarly when the dimensions lie between one and four. To decide which dimension to use, we followed the recommendation of Dai et al. (2017), whose research provided the basis for our model, and chose to use two dimensions.

In theory, increasing the number of inducing points in $\mathbf{Z}^H$ should improve the performance of HMOGP-LV. However, as shown in Figure 5.11 (Right), both the NMSE and NLPD performance of HMOGP-LV deteriorate as the number of inducing points increases. This may be due to difficulty in obtaining sufficiently optimized variational parameters and hyperparameters for our model when using a large number of inducing points in $\mathbf{Z}^H$. In this case, we choose a relatively small number of inducing points in $\mathbf{Z}^H$.

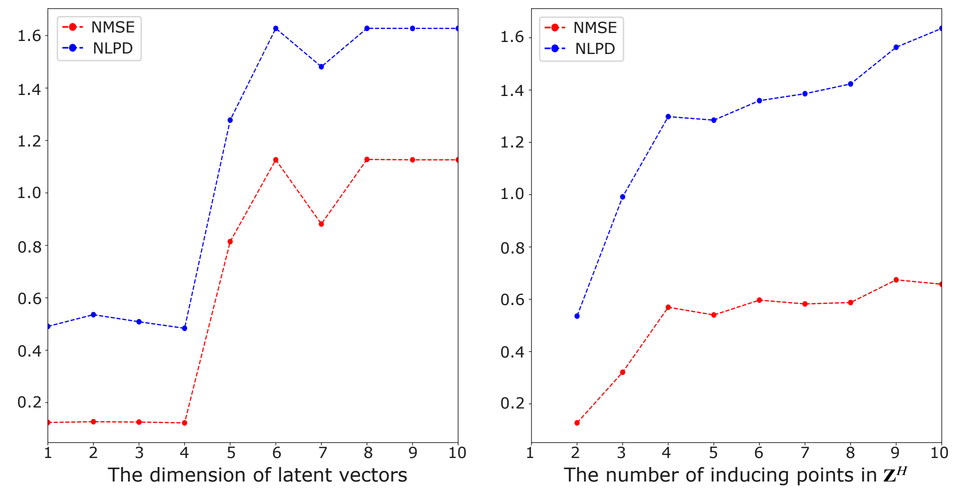


Fig. 5.11 Left: The performance of HMOGP-LV on synthetic data based on the dimensions of the latent vectors; Right: The performance of HMOGP-LV on synthetic data based on the number of inducing points in $\mathbf{Z}^H$.

5.4.2 Real Datasets

In this subsection, we compare the performance of HMOGP-LV against other GP models in two real datasets, gene and motion capture datasets, for multi-output regression problems.

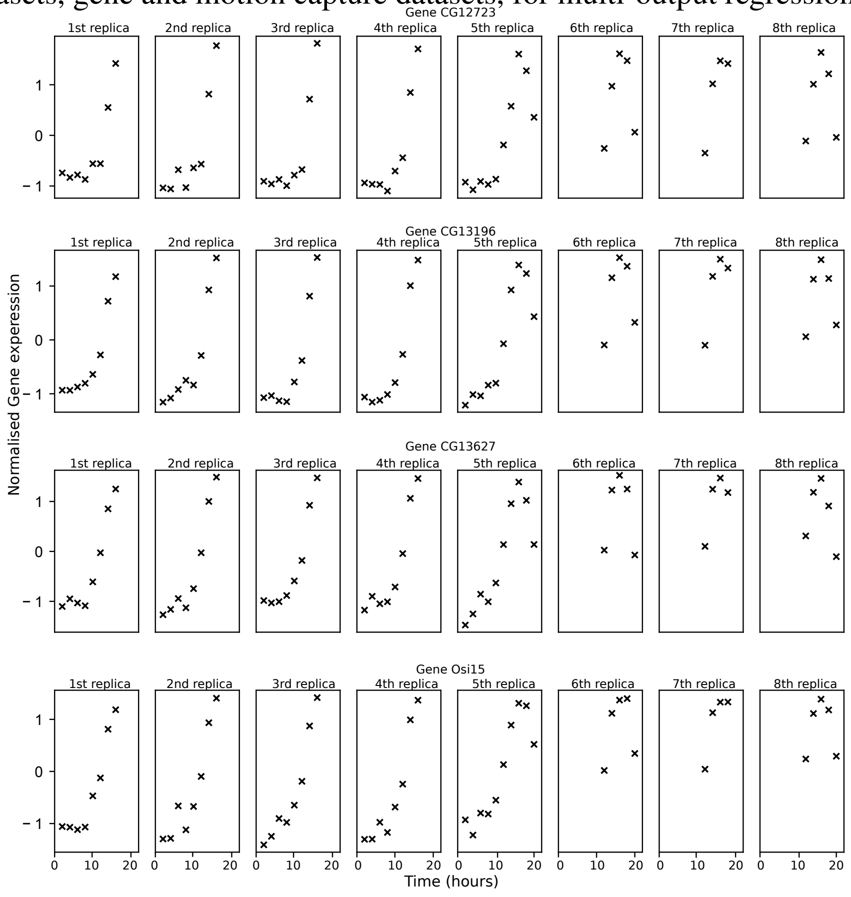


Fig. 5.12 Gene dataset with four gene expressions

5.4.2.1 Gene Dataset

To test the performance of different GP models, we predict temporal gene expression in a gene dataset on Drosophila development (Kalinka et al., 2010). There are six species and each species' gene expression has been measured in eight replicas at different time points. Kalinka et al. (2010) explore 3695 genes. Our experiments focus on one of six species' (melanogaster) development where there are eight replicas for each gene. We consider each gene as one output so each output has eight replicas in this dataset. Figure 5.12 shows some of the examples of this dataset, where replicas can contain a different number of data points. Following Hensman et al. (2013b), this chapter uses the data corresponding to the following genes: 'CG12723', 'CG13196', 'CG13627', 'Osi15' (see Figure 5.12). Therefore, we have four outputs, each with eight replicas, wherein the total number of data points for the eight replicas in each output is 56. Since the dataset is small, we use $M_{\mathbf{X}}$ as 14 for HMOGP-LV, HGPInd, LVMOGP, and LMC in the no missing replica case; we choose $M_{\mathbf{H}}$ as the number of half of all training datasets for those models in the missing replicas case. HMOGP-LV and LVMOGP use $M_{\mathbf{H}}$=2.

Figure 5.13 shows that DHGP, LVMOGP, and LMC have excellent performance since they can exploit the correlation between each output. Further, DHGP can also exploit the hierarchical structure of the dataset through a hierarchical kernel. However, HMOGP-LV has a slightly better performance in terms of NMSE and NLPD. HMOGP-LV not only exploits the hierarchical structure of the gene dataset but also leverages the correlation between each output through the latent information **H**.

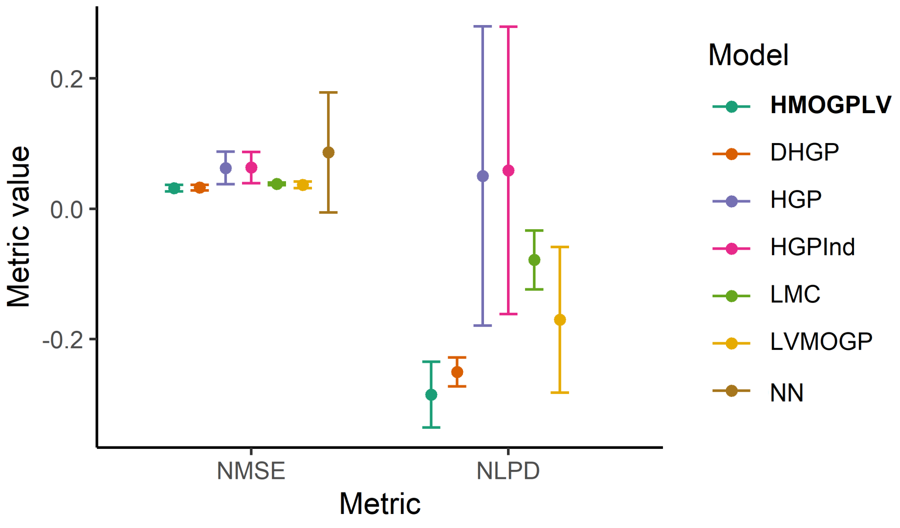


Fig. 5.13 Gene dataset with no missing replicas

To compare the performance of HMOGP-LV in the missing replicas problems, we apply HMOGP-LV in the gene dataset where we assume there is one missing replica in each output in the gene dataset. We randomly choose a missing replica per output so there are now seven replicas in each output as training datasets. We provide in Figure 5.14 the visual results of HMOGP-LV in this experiment where one can observe that the entirely missing replicas are remarkably reconstructed with high accuracy and confidence. From Figure 5.15, we can see that multi-output Gaussian processes approaches (e.g. LVMOGP and LMC) also provide excellent results, comparable to our method. In contrast, the performances of HGPInd appear notably poor in this context.

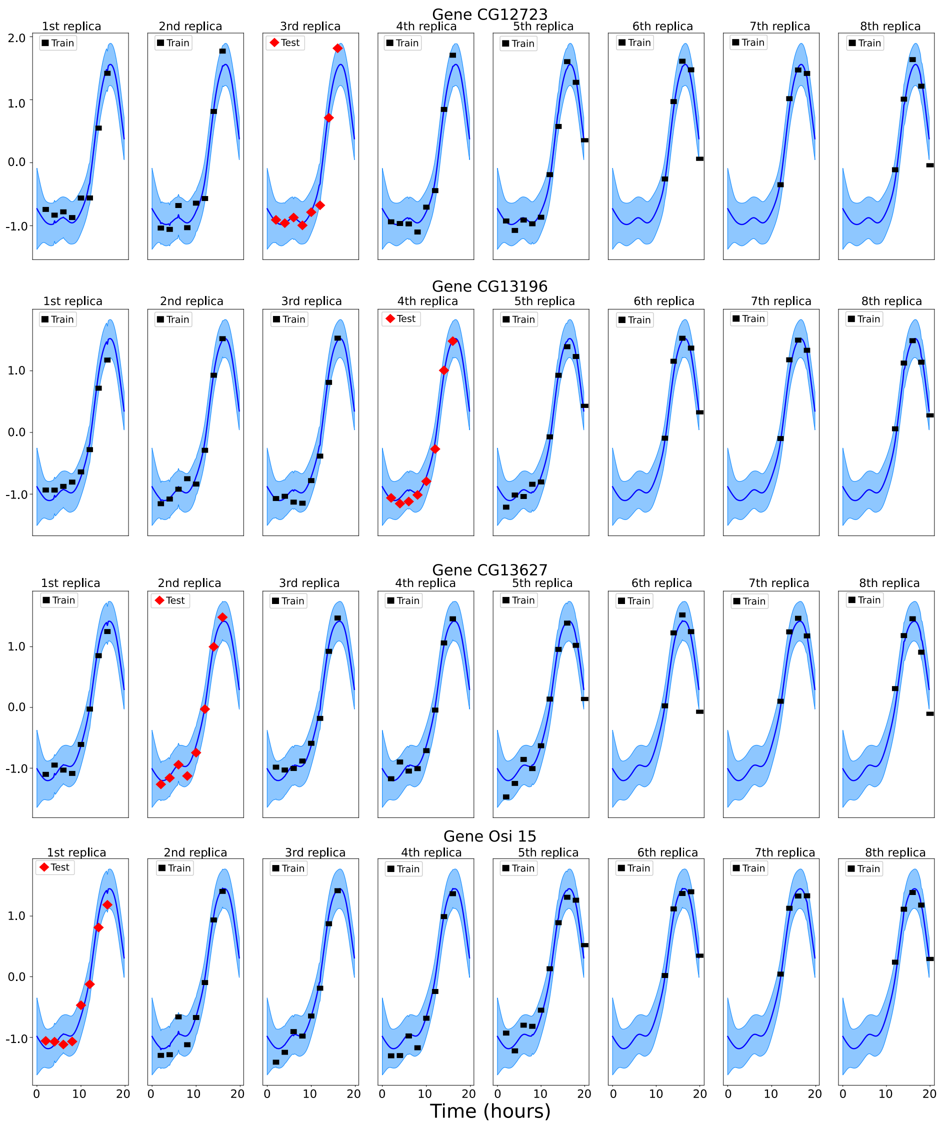


Fig. 5.14 Mean predictive curves associated with their 95% credible intervals for all outputs and replicas of the gene dataset. Locations of training points (in black) and testing points (in red) are specific to each output. Gene dataset with one missing replica in each output (**HMOGP-LV** performance)

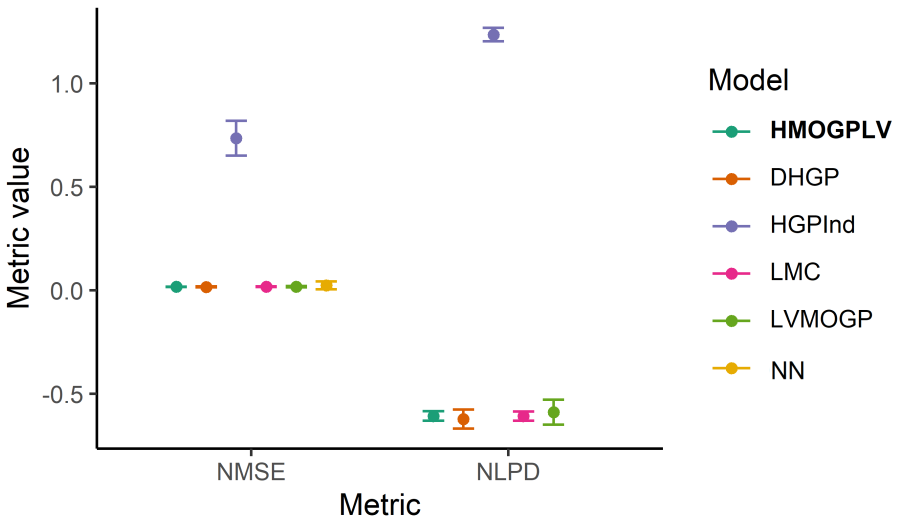


Fig. 5.15 Gene dataset with one missing replica in each output

5.4.2.2 Motion Capture Database

To test the performance of our model to predict missing replicas, we apply HMOGP-LV into the motion capture database (MOCAP) from the CMU motion capture database.[2] We consider four different categories of movement: walking, running, golf swing and jumping. We only consider part of the body. Regarding walking, we take subject 8 with trials 2, 3, 8 and 9, where we consider each trial as a replica. We take the right-hand (humerus, radius, wrist, femur and tibia) and there are 16 positions in total. Additionally, we normalise the input and output data points to zero mean and unit variance, respectively. We take each position as each output and take outputs whose signal-to-noise ratio is over 20 dB. Therefore, there are 16 outputs and each output has four replicas (MOCAP-8). For running, we take subject 9 with trials 1, 2, 3, 5, 6, and 11. We take the head and foot (lower-neck, upper-neck, head, femur, tibia and foot), then there are 16 outputs and each output has six replicas (MOCAP-9). With golf swing, we look at subject 64 with trials 3, 4, 5, 7, 8 and 9. We take the left and right-hand (humerus, radius and wrist). There are nine outputs and each output has six replicas (MOCAP-64). Regarding jumping, we choose subject 118 with trials 3, 4, 11 and 17. We choose the foot (femur, tibia and foot) so there are 12 outputs and each output has four replicas (MOCAP-118). Because each replica has many data points (e.g., around 1000 data points), we have not applied this dataset for training and test data points for each replica but for missing replicas experiments. In all settings, each replica is observed over 200 time points except MOCAP-9 (in MOCAP-9, each replica is observed over 100 time points

[2]The CMU Graphics Lab Motion Capture Database was created with funding from NSF EIA-0196217 and is available at http://mocap.cs.cmu.edu.

since its replica has around 140 times). We show the parameters settings in those datasets for GP models in Table 5.1.

Table 5.1 *Setting and parameters of different GP models in MOCAP dataset. $M_{\mathbf{X}}$ indicates the number of inducing points in $\mathbf{Z}^X$. $M_{\mathbf{H}}$ indicates the number of inducing points in $\mathbf{Z}^H$.* Neither DHGP or NN make use of inducing variables.

Dataset	Model	$M_{\mathbf{H}}$	$M_{\mathbf{X}}$
MOCAP-8	HMOGP-LV	2	6
	LVMOGP		
	HGPInd	None	
	LMC		
MOCAP-9	HMOGP-LV	5	5
	LVMOGP		
	HGPInd	None	
	LMC		
MOCAP-64	HMOGP-LV	5	5
	LVMOGP		
	HGPInd	None	
	LMC		
MOCAP-118	HMOGP-LV	3	6
	LVMOGP		
	HGPInd	None	
	LMC		

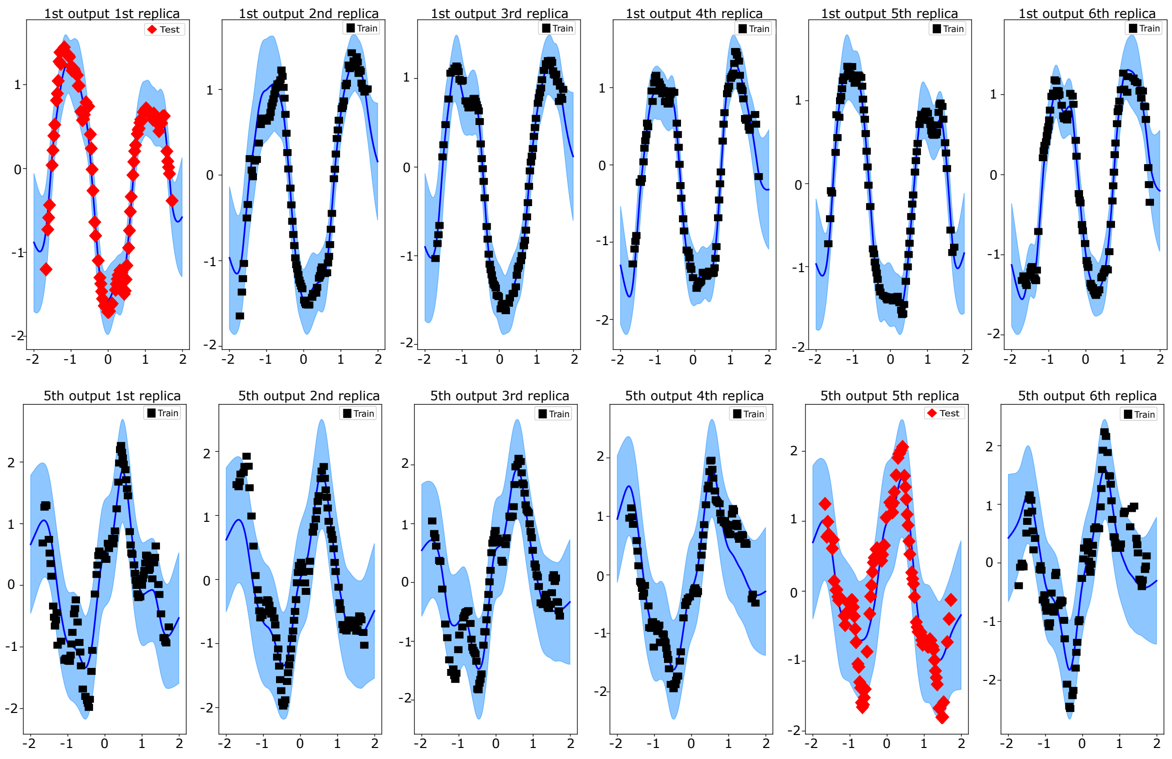


Fig. 5.16 Mean predictive curves associated with their 95% credible intervals for all outputs and replicas of the MOCAP-9 dataset. Locations of training points (in black) and testing points (in red) are specific to each output.

As highlighted in Figure 5.17, HMOGP-LV outperforms other methods in most situations, where DHGP and LVMOGP present comparable results. In particular, the results of the MOCAP-9 experiment, for which the improvement provided by our method is the most prominent, are illustrated in Figure 5.16. One can notice how our model retrieves adequately the overall pattern for the missing replica at no cost in terms of uncertainty. As displayed, it seems that sharing information at different levels, both among outputs and replicas, allows the prediction to remain accurate regardless of the sub-sample of data that is removed. It is worth mentioning that both multi-output methods (LMC and LVMOGP) also exhibit excellent performance in those task, although HMOGP-LV seems to remain the most sensible choice overall.

In some cases, the performance of HMOGP-LV is similar to other models. For example, regarding MOCAP-64 and MOCAP-118, HMOGP-LV has similar performance as LMC and LVMOGP in terms of NMSE (see Figure 5.17 for detail). One possible reason is that we assume there is a strong correlation between each replica and a hierarchical structure in the dataset. However, the hierarchical structure may not exist so our model cannot exploit it, or the correlation may decrease our model's predictive performance.

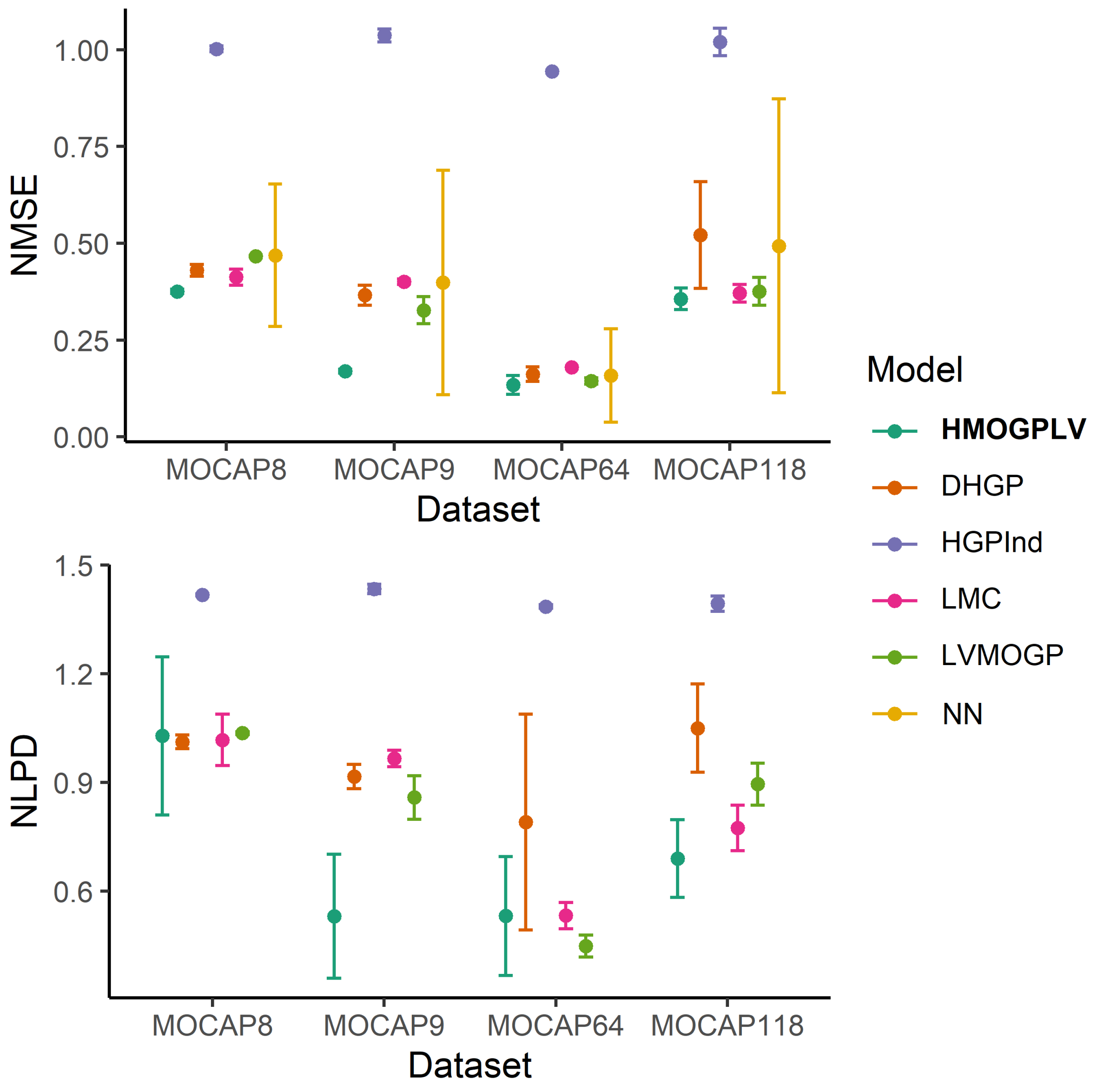


Fig. 5.17 Prediction performances (mean ± standard deviation) for the MOCAP-8, MOCAP-9, MOCAP-64 and MOCAP-118 datasets. For both NMSE and NLPD values, the lower the better.

5.5 Summary

In this chapter, we have introduced HMOGP-LV, a novel framework of multi-output Gaussian processes to deal with multiple regression problems in hierarchically structured datasets. HMOGP-LV uses latent variables to capture the correlation between multiple outputs and a hierarchical kernel matrix to capture the dependency in replicas for each output. If there are any missing replicas, HMOGP-LV can predict them using inducing variables which retain replicas information. We experimentally show that HMOGP-LV has better performance than other models in terms of NMSE and NLPD in both synthetic and real datasets.

Chapter 6

Conclusions and Future Work

This chapter summarises the work developed in the thesis, discusses its limitations, and provides some ideas for future work. In Section 6.1, we conclude the work carried out across the thesis. n Section 6.2, we examine the limitations of our work. In Section 6.3, we outline some insights for future work.

6.1 Summary of Contributions

In this thesis, we have extended multi-output Gaussian processes (MOGPs) to better cope with three different types of datasets: large-scale classification datasets, relatively high dimensional input datasets and datasets with a hierarchical structure in the context of GPs. To handle those datasets in the area of GPs, we proposed different generalisations of MOGPs. We chose a softmax likelihood function through Gumbel noise error in MOGPs to reduce the computational complexity in classification problems (Chapter 3). We integrated a convolutional kernel into MOGPs to deal with a relatively high-dimensional dataset (Chapter 4). Finally, we designed a novel MOGP for datasets with a hierarchical structure (Chapter 5).

- In Chapter 3, we strive to address a challenge for the scalability of MOGPs in the case of a large number of classes. We provided an extension of the MOGP model known as MOGPs-AR that can deal with large-scale classification by subsampling both training input data and classes in each output. Our model applies stochastic variational inference and chooses a softmax likelihood function via Gumbel noise error for all outputs so it can handle large-scale multi-output, multi-class classification problems through subsampling both training dataset and classes in each output. Through the experiments in both synthetic and real classification problems, we showed that MOGPs-

AR outperforms multi-output Gaussian processes regarding scalability and single-output Gaussian processes regarding different performance metrics.

- In Chapter 4, we presented a new MOGP model called MOCGPs-AR that can cope with relatively high-dimensional input data classification problems by integrating a convolutional kernel into MOGPs-AR. In the multi-class classification problems, image data are the most common type of data and MOGPs are not suitable for handling image data. Our model incorporated a convolutional kernel into MOGPs-AR so it can deal with downsized image input datasets. It then inherits the scalability of MOGPs-AR so it can also handle large-scale multi-class classification datasets. Thanks to the convolutional kernel, our model outperforms MOGPs-AR in terms of classification accuracy on the Omniglot dataset.

- In Chapter 5 we developed a novel MOGP model named HMOGP-LV that can handle hierarchically structured datasets. MOGPs are not specifically designed for the aforementioned datasets. We proposed a hierarchical structure of MOGPs, where we developed a hierarchical kernel through the structure. The hierarchical kernel can be divided into two parts: 1) one kernel that encapsulates the hierarchical structure for input data points; 2) another kernel that encodes the interaction between outputs through latent variables. We also derived a variational inference framework for HMOGP-LV. We empirically show that HMOGP-LV outperforms single-output and multi-output Gaussian processes in terms of normalised mean square error and negative log predictive density. By means of inducing variables, our model can make predictions for missing replicas in which we do not have any data at all.

By proposing three extensions of MOGPs in this thesis, we help MOGPs to handle the multi-class classification problem, alleviate the curse of dimensionality, and cope with hierarchically structured data.

6.2 Limitations

We analyse the limitations of our work in this section.

MOGPs-AR can not deal with a multi-label problem where each data point belongs to multiple classes because of using the softmax function which is only suited to each instance associated with a single class (Chapter 3). Then, MOCGP-AR only handles downsized image data, however, the practical application of Gaussian process models to realistic image recognition tasks is still an open research problem. For example, in terms of accuracy performance in a realistic RGB set CIFAR-10 (Krizhevsky et al., 2009), the accuracy performance of Gaussian processes (Blomqvist et al., 2019; Van der Wilk et al., 2017) is not as high as the state-of-the-art like deep learning (Chapter 4). Next, HMOGP-LV only addresses regression problems so far since the likelihood considered is Gaussian; HMOGP-LV is also limited to two layers of hierarchy when accounting for correlations where, in many cases, some datasets have deeper hierarchical structure (Hensman et al., 2013b) (Chapter 5). Therefore, there are various future directions to possibly solve current limitations.

6.3 Future Work

We envisage various directions for future work based on this thesis, as described in this section.

- In terms of the limitation MOGPs-AR, Panos et al. (2021) extend a semiparametric latent model to handle the multi-label problem by using sigmoidal/Bernoulli likelihood for each latent parameter function in single-output problems. It would be an interesting direction for future research to explore whether we can generalise MOGPs-AR for the multi-label problem, a problem that has a strong correlation with the extreme classification problem.
- Regarding realistic image recognition tasks in MOCGP-AR, a potential extension would be to consider integrating the structural properties of deep learning architectures into our model by using deep kernel learning (Wilson et al., 2016).
- Concerning the limitation of HMOGP-LV in regression problems and two-layer hierarchical structure for input space, a potential extension would be to generalise our model for heterogeneous multi-output prediction by combining different kinds of likelihoods and scalable variational inference (Moreno-Muñoz et al., 2018). Another possible extension would build a deep hierarchical structure for input space, e.g., a three-layer hierarchical structure (Hensman et al., 2013b).

References

Alaa, A. M. and Van Der Schaar, M. (2017). Bayesian inference of individualized treatment effects using multi-task Gaussian processes. Advances in neural information processing systems, 30.

Altamirano, M. and Tobar, F. (2022). Nonstationary multi-output Gaussian processes via harmonizable spectral mixtures. In International Conference on Artificial Intelligence and Statistics, pages 3204–3218. PMLR.

Alvarez, M., Luengo, D., and Lawrence, N. D. (2009). Latent force models. In Artificial Intelligence and Statistics, pages 9–16. PMLR.

Alvarez, M. A. (2011). Convolved Gaussian process priors for multivariate regression with applications to dynamical systems. PhD thesis, The University of Manchester (United Kingdom).

Álvarez, M. A., Rosasco, L., Lawrence, N. D., et al. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266.

Álvarez, M. A., Ward, W., and Guarnizo, C. (2019). Non-linear process convolutions for multi-output Gaussian processes. In The 22nd International Conference on Artificial Intelligence and Statistics, pages 1969–1977. PMLR.

Barber, D. and Williams, C. (1996). Gaussian processes for Bayesian classification via hybrid Monte Carlo. Advances in neural information processing systems, 9.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. Journal of the American statistical Association, 112(518):859–877.

Blomqvist, K., Kaski, S., and Heinonen, M. (2019). Deep convolutional Gaussian processes. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 582–597. Springer.

Bonilla, E. V., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160.

Chai, K. M. A. (2012). Variational multinomial logit Gaussian process. *The Journal of Machine Learning Research*, 13:1745–1808.

Cortez, P. and Silva, A. M. G. (2008). Using data mining to predict secondary school student performance. In *Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC)*, pages 5–12. EUROSIS-ETI.

Dahl, A. and Bonilla, E. V. (2019). Grouped Gaussian processes for solar power prediction. *Machine Learning*, 108(8-9):1287–1306.

Dai, Z., Álvarez, M. A., and Lawrence, N. D. (2017). Efficient modeling of latent information in supervised learning using Gaussian processes. arXiv preprint arXiv:1705.09862.

Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian processes. In Artificial intelligence and statistics, pages 207–215. PMLR.

Dezfouli, A. and Bonilla, E. V. (2015). Scalable inference for Gaussian process models with black-box likelihoods. *Advances in Neural Information Processing Systems*, 28:1414–1422.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Flaxman, S., Gelman, A., Neill, D., Smola, A., Vehtari, A., and Wilson, A. G. (2015). Fast hierarchical Gaussian processes. Manuscript in preparation.

Galy-Fajou, T., Wenzel, F., Donner, C., and Opper, M. (2020). Multi-class Gaussian process classification made conjugate: Efficient inference via data augmentation. In *Uncertainty in Artificial Intelligence*, pages 755–765. PMLR.

Gelman, A., Carlin, J. B., Stern, H. S., Vehtari, A., and Rubin, D. B. (2013). Bayesian data analysis, 3rd edition. Chapman and Hall/CRC.

Gibbs, M. N. and MacKay, D. J. (2000). Variational Gaussian process classifiers. IEEE Transactions on Neural Networks, 11(6):1458–1464.

Girolami, M. and Rogers, S. (2006). Variational Bayesian multinomial probit regression with Gaussian process priors. *Neural Computation*, 18(8):1790–1817.

Goovaerts, P. et al. (1997). Geostatistics for natural resources evaluation. Oxford University Press on Demand.

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

Hensman, J., Lawrence, N. D., and Rattray, M. (2013b). Hierarchical Bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. BMC bioinformatics, 14(1):252.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015a). Scalable variational Gaussian process classification. In Artificial Intelligence and Statistics, pages 351–360. PMLR.

Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. (2015b). MCMC for variationally sparse Gaussian processes. Advances in Neural Information Processing Systems, 28.

Hernández-Lobato, D., Hernández-lobato, J., and Dupont, P. (2011). Robust multi-class Gaussian process classification. *Advances in neural information processing systems*, 24:280–288.

Kalinka, A. T., Varga, K. M., Gerrard, D. T., Preibisch, S., Corcoran, D. L., Jarrells, J., Ohler, U., Bergman, C. M., and Tomancak, P. (2010). Gene expression divergence recapitulates the developmental hourglass model. Nature, 468(7325):811–814.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18, page 5. Citeseer.

Kim, H.-C. and Ghahramani, Z. (2006). Bayesian Gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1948–1959.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

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

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lawrence, N. and Hyvärinen, A. (2005). Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research*, 6(11).

Lawrence, N. D. and Moore, A. J. (2007). Hierarchical Gaussian process latent variable models. In Proceedings of the 24th international conference on Machine learning, pages 481–488.

Li, M. and Kontar, R. (2022). On negative transfer and structure of latent functions in multi-output gaussian processes. SIAM/ASA Journal on Uncertainty Quantification, 10(4):1714–1732.

Li, P. and Chen, S. (2018). Hierarchical Gaussian processes model for multi-task learning. Pattern Recognition, 74:134–144.

Liu, H., Ong, Y.-S., Yu, Z., Cai, J., and Shen, X. (2019). Scalable Gaussian Process Classification with Additive Noise for Various Likelihoods. arXiv preprint arXiv:1909.06541.

Minka, T. P. (2013). Expectation propagation for approximate Bayesian inference. arXiv preprint arXiv:1301.2294.

Moreno-Muñoz, P., Artés, A., and Álvarez, M. (2018). Heterogeneous multi-output Gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 6712–6721.

Neal, R. M. (1997). Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. arXiv preprint physics/9701026.

Nguyen, T. N. A., Bouzerdoum, A., and Phung, S. L. (2018). Stochastic variational hierarchical mixture of sparse Gaussian processes for regression. *Machine Learning*, 107(12):1947–1986.

Osborne, M. A., Roberts, S. J., Rogers, A., Ramchurn, S. D., and Jennings, N. R. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pages 109–120. IEEE.

Panos, A., Dellaportas, P., and Titsias, M. (2021). Large Scale Multi-Label Learning using Gaussian Processes. *Machine Learning*.

Parra, G. and Tobar, F. (2017). Spectral mixture kernels for multi-output Gaussian processes. Advances in Neural Information Processing Systems, 30.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rasmussen, C. E. and Williams, C. K. (2006). Gaussian processes for machine learning.

Ruiz, F. J., Titsias, M. K., Dieng, A. B., and Blei, D. M. (2018). Augment and reduce: Stochastic inference for large categorical distributions. arXiv preprint arXiv:1802.04220.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850.

Skolidis, G. and Sanguinetti, G. (2011). Bayesian multitask classification with Gaussian process priors. *IEEE Transactions on Neural Networks*, 22(12).

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264.

Snoek, C. G., Worring, M., Van Gemert, J. C., Geusebroek, J.-M., and Smeulders, A. W. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 421–430.

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

Titsias, M. and Lawrence, N. D. (2010). Bayesian Gaussian process latent variable model. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pages 844–851. JMLR Workshop and Conference Proceedings.

Torres-Sospedra, J., Montoliu, R., Martínez-Usó, A., Avariento, J. P., Arnau, T. J., Benedito-Bordonau, M., and Huerta, J. (2014). UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems. In 2014 international conference on indoor positioning and indoor navigation (IPIN), pages 261–270. IEEE.

Van der Wilk, M. (2019). Sparse Gaussian process approximations and applications. PhD thesis, University of Cambridge.

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

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3):261–272.

Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In Artificial intelligence and statistics, pages 370–378. PMLR.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2018). Scalable Gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78.

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning. arXiv preprint arXiv:1707.08114.

Appendix A

In this appendix, we provide some basic maths formulas about joint, marginal, conditional probability and Gaussian identities.

A.1 Joint, Marginal and Conditional Probability

The marginal probability of x given by:

$$p(x) = \int p(x,y)dy. \tag{A.1}$$

The conditional probability function is defined as:

$$p(x|y) = \frac{p(x,y)}{p(y)}. \tag{A.2}$$

The Joint probability is defined as :

$$\begin{aligned} p(x,y) &= P(x|y)p(y) \\ &= p(y|x)p(x). \end{aligned} \tag{A.3}$$

The Bayes' theorem is defined as:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{A.4}$$

A.2 Gaussian Identities

Based on a joint Gaussian distribution $\mathcal{N}(\mathbf{x}|\mu,\Sigma)$, $\mathbf{x}_1$ and $\mathbf{x}_2$ are two disjoint subsets of $\mathbf{x}$. There are

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}, \ \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \ \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}. \tag{A.5}$$

Then the conditional distribution is:

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}|\mu_1 - \Lambda_{11}^{-1}\Lambda_{12}(\mathbf{x}-\mu_2),\ \Lambda_{11}^{-1}), \tag{A.6}$$

where Λ is named as the precision matrix and $\Lambda = \Sigma^{-1}$:

$$\Lambda = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix} \tag{A.7}$$

The marginal distribution of $\mathbf{x}_1$ is:

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1|\mu_1,\Sigma_{11}). \tag{A.8}$$

We assume a marginal distribution for $\mathbf{x}$ and a conditional distribution for $p(\mathbf{y}|\mathbf{x})$:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu,\Sigma_x) \tag{A.9}$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|A\mathbf{x}+b,\Sigma_y). \tag{A.10}$$

We can obtain the marginal distribution of $\mathbf{y}$ and the conditional distribution of $p(\mathbf{x}|\mathbf{y})$:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|A\mu+b,\Sigma_y + A\Sigma_x A^\top) \tag{A.11}$$
$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\Sigma A^\top \Sigma_f^{-1}(\mathbf{y}-b) + \Sigma_x^{-1}\mu,\Sigma), \tag{A.12}$$

where $\Sigma = (\Sigma_x^{-1} + A^\top \Sigma_f^{-1} A)^{-1}$.

Appendix B

Complete Derivation of the Lower Bound $\mathcal{L}$

To compute the derivation of the lower bound $\mathcal{L}$ in Chapter 3, we begin with the following:

$$
\begin{aligned}
\mathcal{L} =& \mathbb{E}_{q(\mathbf{f},\mathbf{u},\varepsilon)}\left[log\frac{p(\mathbf{y},\mathbf{f},\mathbf{u},\varepsilon)}{q(\mathbf{f},\mathbf{u},\varepsilon)}\right] \\
=& \int\int\int q(\varepsilon|\mathbf{f})p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{f},\mathbf{u},\varepsilon)p(\mathbf{u})p(\varepsilon)}{q(\varepsilon|\mathbf{f})q(\mathbf{u})}\mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
=& \int\int\int \prod_{d=1}^{D}\prod_{i=1}^{N} q(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i))\prod_{c=1}^{C_d} p(\mathbf{f}_d^c|\mathbf{u})\prod_{q=1}^{Q} q(\mathbf{u}_q) \\
& \times\log\frac{\prod_{d=1}^{D}\prod_{i=1}^{N} p\left(y_d(\mathbf{x}_i)\mid\widetilde{\mathbf{f}}_d(\mathbf{x}_i),\varepsilon_{d,i}\right)\prod_{q=1}^{Q} p\left(\mathbf{u}_q\right)\prod_{d=1}^{D}\prod_{i=1}^{N} p(\varepsilon_{d,i})}{\prod_{d=1}^{D}\prod_{i=1}^{N} q(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i))\prod_{q=1}^{Q} q\left(\mathbf{u}_q\right)} \\
& \mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
=& \int\int\int \prod_{d=1}^{D}\prod_{i=1}^{N} q(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i))\prod_{c=1}^{C_d} p(\mathbf{f}_d^c|\mathbf{u})\prod_{q=1}^{Q} q(\mathbf{u}_q) \\
& \times\log\prod_{d=1}^{D}\prod_{i=1}^{N} p\left(y_d(\mathbf{x}_i)\mid\widetilde{\mathbf{f}}_d(\mathbf{x}_i),\varepsilon_{d,i}\right)\mathrm{d}\mathbf{f}\mathrm{d}\varepsilon\mathrm{d}\mathbf{u} \\
& -\sum_{q=1}^{Q}\mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right)-\sum_{i=d}^{D}\sum_{i=1}^{N}\mathrm{KL}\left(q\left(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)\|p\left(\varepsilon_{d,i}\right)\right),
\end{aligned}
\tag{B.1}
$$

where $q(\mathbf{f},\mathbf{u},\varepsilon) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})q(\varepsilon|\mathbf{f})$. We assume $q(\mathbf{f}) \approx p(\mathbf{f} \mid \mathbf{y})$ so we obtain

$$\begin{aligned} q(\mathbf{f}) &\approx p(\mathbf{f} \mid \mathbf{y}) \\ &= \int p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u}) \mathrm{d}\mathbf{u} \\ &= \int \prod_{d=1}^{D} \prod_{c=1}^{C_d} p\left(\mathbf{f}_d^c \mid \mathbf{u}\right) \prod_{q=1}^{Q} q\left(\mathbf{u}_q\right) \mathrm{d}\mathbf{u} \\ &= \prod_{d=1}^{D} \prod_{c=1}^{C_d} \mathbf{E}_{q(\mathbf{u})} \left\{ p\left(\mathbf{f}_d^c \mid \{\mathbf{u}_q\}_{q=1}^{Q}\right)\right\} \\ &= \prod_{d=1}^{D} q\left(\widetilde{\mathbf{f}}_d\right) = \prod_{d=1}^{D} \prod_{c=1}^{C_d} q\left(\mathbf{f}_d^c\right) = \prod_{i=1}^{N} \prod_{d=1}^{D} \prod_{c=1}^{C_d} q(f_d^c(\mathbf{x}_i)). \end{aligned} \tag{B.2}$$

The above function means the latent parameter functions are mutually independent in $q(\mathbf{f})$. Then, we obtain:

$$\begin{aligned} \mathcal{L} = & \int \int \prod_{d=1}^{D} \prod_{i=1}^{N} q(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)) q(\mathbf{f}) \log \prod_{d=1}^{D} \prod_{i=1}^{N} p\left(y_d(\mathbf{x}_i) \mid \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right) \mathrm{d}\mathbf{f}\mathrm{d}\varepsilon \\ & - \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right) \| p\left(\mathbf{u}_q\right)\right) - \sum_{i=d}^{D} \sum_{i=1}^{N} \mathrm{KL}\left(q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \| p\left(\varepsilon_{d,i}\right)\right) \\ = & \sum_{d}^{D} \sum_{i}^{N} \mathbb{E}_{q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)} \left[\log p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right)\right] \\ & - \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right) \| p\left(\mathbf{u}_q\right)\right) - \sum_{i=d}^{D} \sum_{i=1}^{N} \mathrm{KL}\left(q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \| p\left(\varepsilon_{d,i}\right)\right). \end{aligned} \tag{B.3}$$

The $q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)$ approximates the posterior $p\left(\varepsilon_{d,i} | y_d(\mathbf{x}_i), \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)$ (similar to Liu et al. (2019)):

$$\begin{aligned} p\left(\varepsilon_{d,i} | y_d(\mathbf{x}_i), \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) &\propto p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right) p\left(\varepsilon_{d,i}\right) \\ &= \phi_{\mathcal{G}}\left(\varepsilon_{d,i}\right) \prod_{c \neq y_d(\mathbf{x}_i)} \Phi_{\mathcal{G}}\left(\varepsilon_{d,i} + f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i) - f_d^c(\mathbf{x}_i)\right) \\ &= \exp\left(-\varepsilon_{d,i} - \left(1 + \sum_{c \neq y_d(\mathbf{x}_i)} e^{f_d^c(\mathbf{x}_i) - f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i)}\right) e^{-\varepsilon_{d,i}}\right) \\ &\overset{c}{=} \mathrm{Gumbel}\left(\varepsilon_{d,i} | \log \theta_{d,i}^*, 1\right), \end{aligned} \tag{B.4}$$

where $\theta^*_{d,i} = 1 + \sum_{c \neq y_i} e^{f_d^c(\mathbf{x}_i) - f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i)} = \sum_{c=1}^{C_d} e^{f_d^c(\mathbf{x}_i) - f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i)}$. The optimal $p\left(\varepsilon_{d,i} | y_d(\mathbf{x}_i), \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)$ does have exact analytic form. However, $\mathcal{L}$ will be intractable by using an analytic form. We thus take a more general distribution $q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) = \text{Gumbel}\left(\varepsilon_{d,i} | \log \theta_{d,i}, 1\right)$, which satisfies $\theta_{d,i} > 1$, also including the optimal distribution. Then the $\mathcal{L}$ is:

$$\begin{aligned}\mathcal{L} =& \sum_d^D \sum_i^N \mathbb{E}_{q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)} \left[\log p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right)\right] \\ & - \sum_{q=1}^{Q} \text{KL}\left(q\left(\mathbf{u}_q\right) \| p\left(\mathbf{u}_q\right)\right) - \sum_{i=d}^{D} \sum_{i=1}^{N} \text{KL}\left(q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \| p\left(\varepsilon_{d,i}\right)\right).\end{aligned} \tag{B.5}$$

We first consider the inner expectation in the double-expectation term in $\mathcal{L}$:

$$\begin{aligned}& \mathbb{E}_{q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)} \left[\log p\left(y_d(\mathbf{x}_i) | \widetilde{\mathbf{f}}_d(\mathbf{x}_i), \varepsilon_{d,i}\right)\right] \\ =& \sum_{c \neq y_d(\mathbf{x}_i)} \int_{-\infty}^{+\infty} q\left(\varepsilon_{d,i} | \widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \log \Phi_{\mathcal{G}}\left(\varepsilon_{d,i} + f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i) - f_d^c(\mathbf{x}_i)\right) \mathrm{d}\varepsilon_{d,i} \\ =& - \sum_{c \neq y_d(\mathbf{x}_i)} \int_{-\infty}^{+\infty} e^{\left(-\left(\varepsilon_{d,i} - \log \theta_{d,i}\right) - e^{-\left(\varepsilon_{d,i} - \log \theta_{d,i}\right)}\right)} \\ & e^{-\left(\varepsilon_{d,i} + f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i) - f_d^c(\mathbf{x}_i)\right)} \mathrm{d}\varepsilon_{d,i} \\ =& \sum_{c \neq y_d(\mathbf{x}_i)} \theta_{d,i} e^{f_d^c(\mathbf{x}_i) - f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i)} \left.\frac{(1 + \theta_{d,i} u) e^{-\theta_{d,i} u}}{\theta_{d,i}^2}\right|_0^{+\infty} \\ =& - \frac{1}{\theta_{d,i}} \sum_{c \neq y_i} e^{f_d^c(\mathbf{x}_i) - f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i)},\end{aligned} \tag{B.6}$$

where $u = e^{-\varepsilon_{d,i}}$. We second consider the outside expectation. Because of

$$q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) = \prod_{c=1}^{C_d} q\left(f_d^c(\mathbf{x}_i)\right) = \prod_{c=1}^{C_d} \mathcal{N}\left(f_d^c(\mathbf{x}_i) | \mu_{f_d^c}(\mathbf{x}_i), \nu_{f_d^c}(\mathbf{x}_i)\right), \tag{B.7}$$

we take expression B.7 and expression B.6 to obtain

$$
\begin{aligned}
&\mathbb{E}_{q\left(\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)q\left(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right)}\left[\langle \log p\left(y_d\left(\mathbf{x}_i\right)|\widetilde{\mathbf{f}}_d(\mathbf{x}_i),\varepsilon_{d,i}\right)\right] \\
&= -\frac{1}{\theta_{d,i}}\int e^{-f_d^{y_d(\mathbf{x}_i)}(\mathbf{x}_i)} q\left(f_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right)\right) \mathrm{d} f_d^{y_d(\mathbf{x}_i)}\left(\mathbf{x}_i\right) \\
&\quad \times \sum_{c \neq y_d(\mathbf{x}_i)} \int e^{f_d^c(\mathbf{x}_i)} q\left(f_d^c\left(\mathbf{x}_i\right)\right) \mathrm{d} f_d^c\left(\mathbf{x}_i\right) \\
&= -\frac{1}{\theta_{d,i}} \exp\left(\frac{\nu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)}{2} - \mu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)\right) \sum_{c \neq y_d(\mathbf{x}_i)} \exp\left(\frac{\nu_{f_d^c}(\mathbf{x}_i)}{2} + \mu_{f_d^c}(\mathbf{x}_i)\right).
\end{aligned} \tag{B.8}
$$

Calculating the KL divergence $\sum_{i=d}^{D}\sum_{i=1}^{N} \mathrm{KL}\left(q\left(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \| p\left(\varepsilon_{d,i}\right)\right)$ term, we have

$$
\sum_{i=d}^{D}\sum_{i=1}^{N} \mathrm{KL}\left(q\left(\varepsilon_{d,i}|\widetilde{\mathbf{f}}_d(\mathbf{x}_i)\right) \| p\left(\varepsilon_{d,i}\right)\right) = \sum_{i=d}^{D}\sum_{i=1}^{N}\left(\log\theta_{d,i} + \frac{1}{\theta_{d,i}} - 1\right). \tag{B.9}
$$

Then, the closed-form $\mathcal{L}$ is reorganized as

$$
\mathcal{L} = \sum_{d=1}^{D}\sum_{i=1}^{N}\left\{-\frac{1}{\theta_{d,i}}\mathcal{P}_{d,i} - \log\theta_{d,i} - \frac{1}{\theta_{d,i}} + 1\right\} - \sum_{q=1}^{Q} \mathrm{KL}\left(q\left(\mathbf{u}_q\right) \| p\left(\mathbf{u}_q\right)\right), \tag{B.10}
$$

where $\mathcal{P}_{d,i} = \exp\left(\frac{\nu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)}{2} - \mu_{f_d^{y_d(\mathbf{x}_i)}}(\mathbf{x}_i)\right)\sum_{c\neq y_d(\mathbf{x}_i)}\exp\left(\frac{\nu_{f_d^c}(\mathbf{x}_i)}{2} + \mu_{f_d^c}(\mathbf{x}_i)\right)$. To get a tight bound, we derivative $\mathcal{L}$ with respect to $\theta_{d,i}$,

$$
\frac{\partial\mathcal{L}}{\partial\theta_{d,i}} = \sum_{d=1}^{D}\sum_{i=1}^{N}\frac{1}{\theta_{d,i}^2}\left(\mathcal{P}_{d,i}+1\right) - \frac{1}{\theta_{d,i}} = 0. \tag{B.11}
$$

We thus obtain the optimal value $\theta_{d,i}^* = \mathcal{P}_{d,i}+1$. After substitution of $\theta_{d,i}$ by $\theta_{d,i}^*$, there is

$$
\mathcal{L} = -\sum_{d=1}^{D}\sum_{i=1}^{N}\log\left(\mathcal{P}_{d,i}+1\right) - \sum_{q=1}^{Q}\mathrm{KL}\left(q\left(\mathbf{u}_q\right)\|p\left(\mathbf{u}_q\right)\right). \tag{B.12}
$$

Appendix C

In this appendix, we present the detail about deriving the lower bound of the log marginal likelihood and exploiting kronecker product decomposition to obtain efficient formulations for $\mathcal{F}$ in Chapter 5.

C.1 The Lower Bound of The Log Marginal Likelihood

In this section, we derive the lower bound of the log marginal likelihood of our model. We assume variational posterior distributions are $q(\mathbf{H})$, $q(\mathbf{U}_{:})$ and $q(\mathbf{f} \mid \mathbf{U}_{:}, \mathbf{H}) = p(\mathbf{f} \mid \mathbf{U}_{:}, \mathbf{H})$. Then we obtain:

$$\begin{aligned}
\log p(\mathbf{y}) &= \log \int\int\int p(\mathbf{y}, \mathbf{f}, \mathbf{H}, \mathbf{U}_{:})\, \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{H}\mathrm{d}\mathbf{U}_{:} \\
&= \log \int\int\int \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{H}, \mathbf{U}_{:})\, q(\mathbf{f}, \mathbf{H}, \mathbf{U}_{:})}{q(\mathbf{f}, \mathbf{H}, \mathbf{U}_{:})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{H}\mathrm{d}\mathbf{U}_{:} \\
&\geq \int\int\int q(\mathbf{f}, \mathbf{H}, \mathbf{U}_{:}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{H}, \mathbf{U}_{:})}{q(\mathbf{f}, \mathbf{H}, \mathbf{U}_{:})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{H}\mathrm{d}\mathbf{U}_{:} \\
&= \mathcal{L}.
\end{aligned} \tag{C.1}$$

Then,

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{q(\mathbf{f}, \mathbf{H}, \mathbf{U}_{:})} \left[\log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{H}, \mathbf{U}_{:})}{q(\mathbf{f}, \mathbf{H}, \mathbf{U}_{:})} \right] \\
&= \int\int\int p(\mathbf{f} \mid \mathbf{U}_{:}, \mathbf{H})\, q(\mathbf{U}_{:}) q(\mathbf{H}) \log \frac{p(\mathbf{y} | \mathbf{f}, \mathbf{H}, \mathbf{U}_{:}) p(\mathbf{f} \mid \mathbf{U}_{:}, \mathbf{H})\, p(\mathbf{U}_{:}) p(\mathbf{H})}{p(\mathbf{f} \mid \mathbf{U}_{:}, \mathbf{H})\, q(\mathbf{U}_{:}) q(\mathbf{H})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{U}_{:}\mathrm{d}\mathbf{H} \\
&= \int\int\int p(\mathbf{f} \mid \mathbf{U}_{:}, \mathbf{H})\, q(\mathbf{U}_{:}) q(\mathbf{H}) \log \frac{p(\mathbf{y} | \mathbf{f}, \mathbf{H}, \mathbf{U}_{:}) p(\mathbf{U}_{:}) p(\mathbf{H})}{q(\mathbf{U}_{:}) q(\mathbf{H})} \mathrm{d}\mathbf{f}\mathrm{d}\mathbf{U}_{:}\mathrm{d}\mathbf{H}.
\end{aligned} \tag{C.2}$$

Finally,

$$\mathcal{L} = \int q(\mathbf{H}) \left[\int q(\mathbf{U}_:) \left[\mathbb{E}_{p(\mathbf{f}|\mathbf{U}_:,\mathbf{H})}[\log p(\mathbf{y} \mid \mathbf{f},\mathbf{H})] + \log \frac{p(\mathbf{U}_:)}{q(\mathbf{U}_:)} + \log \frac{p(\mathbf{H})}{q(\mathbf{H})} \right] \mathrm{d}\mathbf{U}_: \right] \mathrm{d}\mathbf{H}$$
$$= \overbrace{\mathbb{E}_{q(\mathbf{f},\mathbf{U}_:,\mathbf{H})}[\log p(\mathbf{y} \mid \mathbf{f},\mathbf{H})]}^{\mathcal{F}} - \mathrm{KL}(q(\mathbf{H}) \| p(\mathbf{H})) - \mathrm{KL}(q(\mathbf{U}_:) \| p(\mathbf{U}_:)). \tag{C.3}$$

C.2 Derivation of $\mathcal{F}$ Given the Same Input Datasets

In this section, we show details for deriving $\mathcal{F}$ using the same input datasets:

$$\begin{aligned}
\mathcal{F} &= \mathbb{E}_{p(\mathbf{f}|\mathbf{U}_:,\mathbf{H})q(\mathbf{U}_:)q(\mathbf{H})}\left[\log p(\mathbf{y} \mid \mathbf{f},\mathbf{H})\right] \\
&= \int q(\mathbf{H}) \int q(\mathbf{U}_:) \underbrace{\int p(\mathbf{f} \mid \mathbf{U}_:,\mathbf{H}) \log p(\mathbf{y} \mid \mathbf{f},\mathbf{H}) \, \mathrm{d}\mathbf{f}}_{\mathcal{L}_F} \mathrm{d}\mathbf{U}_: \mathrm{d}\mathbf{H} \\
&= \int q(\mathbf{H}) \underbrace{\int q(\mathbf{U}_:) \mathcal{L}_F \mathrm{d}\mathbf{U}_:}_{\mathcal{L}_U} \mathrm{d}\mathbf{H} \\
&= \underbrace{\int q(\mathbf{H}) \mathcal{L}_U \mathrm{d}\mathbf{H}}_{\mathcal{L}_H}.
\end{aligned} \tag{C.4}$$

First, we calculate $\mathcal{L}_F$:

$$\begin{aligned}
\mathcal{L}_F &= \int p(\mathbf{f} \mid \mathbf{U}_:,\mathbf{H}) \log p(\mathbf{y} \mid \mathbf{f},\mathbf{H}) \, \mathrm{d}\mathbf{f} \\
&= \log \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{U}_:, \sigma^2\right) - \frac{1}{2\sigma^2} \mathrm{Tr}\left[\mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{K}_{\mathbf{fU}}^{\top}\right],
\end{aligned} \tag{C.5}$$

where $p(\mathbf{f} \mid \mathbf{U}_:,\mathbf{H}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{U}_:, \mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{K}_{\mathbf{fU}}^{\top}\right)$ and $\mathrm{Tr}[\cdot]$ is a trace of a matrix. Second, we calculate $\mathcal{L}_U$:

$$\begin{aligned}
\mathcal{L}_U &= \int q(\mathbf{U}_:) \mathcal{L}_F \mathrm{d}\mathbf{U}_: \\
&= \log \mathcal{N}\left(\mathbf{y} \mid \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{M}_:, \sigma^2\right) - \frac{1}{2\sigma^2} \mathrm{Tr}\left[\mathbf{K}_{\mathbf{ff}} - \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{K}_{\mathbf{fU}}^{\top}\right] \\
&\quad - \frac{1}{2\sigma^2} \mathrm{Tr}\left[\mathbf{\Sigma}^{\mathbf{U}_:} \mathbf{K}_{\mathbf{UU}}^{-1} \mathbf{K}_{\mathbf{fU}}^{\top} \mathbf{K}_{\mathbf{fU}} \mathbf{K}_{\mathbf{UU}}^{-1}\right].
\end{aligned} \tag{C.6}$$

where $q(\mathbf{U}_:) = \mathcal{N}\left(\mathbf{U}_: \mid \mathbf{M}_:, \mathbf{\Sigma}^{\mathbf{U}_:}\right)$ in which $\mathbf{U}_:$ and $\mathbf{M}_:$ are variational parameters. Finally, we consider $\mathcal{L}_H$:

$$
\begin{aligned}
\mathcal{L}_H =& \int q(\mathbf{H})\mathcal{L}_U \mathrm{d}\mathbf{H} \\
=& \mathbb{E}_{q(\mathbf{H})}\left[\log \mathcal{N}\left(\mathbf{y} \mid \mathbf{K_{fU}}\mathbf{K_{UU}}^{-1}\mathbf{M}_{:}, \sigma^2\right)\right] - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\langle \mathbf{K_{ff}}\rangle_{q(\mathbf{H})} - \mathbf{K_{UU}}^{-1}\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}^\top \mathbf{K_{fU}}\right]\right] \\
& - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{\Sigma}^{\mathbf{U}_:}\mathbf{K_{UU}}^{-1}\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}^\top \mathbf{K_{fU}}\right]\mathbf{K_{UU}}^{-1}\right] \\
=& \underbrace{-\frac{DNR}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\mathbf{y}^\top \mathbf{y}}_{\mathbf{C}} + \frac{1}{\sigma^2}\mathbf{y}^\top \underbrace{\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}\right]}_{\Psi}\mathbf{K_{UU}}^{-1}\mathbf{M}_{:} \\
& - \frac{1}{2\sigma^2}\mathbf{M}_{:}^\top \mathbf{K_{UU}}^{-1}\underbrace{\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}^\top \mathbf{K_{fU}}\right]}_{\Phi}\mathbf{K_{UU}}^{-1}\mathbf{M}_{:} - \frac{1}{2\sigma^2}\underbrace{\mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{ff}}\right]\right)}_{\psi} \\
& + \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{K_{UU}}^{-1}\underbrace{\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}^\top \mathbf{K_{fU}}\right]}_{\Phi}\right] - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\mathbf{\Sigma}^{\mathbf{U}_:}\mathbf{K_{UU}}^{-1}\underbrace{\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}^\top \mathbf{K_{fU}}\right]}_{\Phi}\mathbf{K_{UU}}^{-1}\right] \\
=& \mathbf{C} + \frac{1}{\sigma^2}\mathbf{y}^\top \Psi \mathbf{K_{UU}}^{-1}\mathbf{M}_{:} - \frac{1}{2\sigma^2}\left(\psi - \mathrm{Tr}\left[\mathbf{K_{UU}}^{-1}\Phi\right]\right) \\
& - \frac{1}{2\sigma^2}\mathrm{Tr}\left[\mathbf{K_{UU}}^{-1}\Phi\mathbf{K_{UU}}^{-1}\left(\mathbf{M}_{:}\mathbf{M}_{:}^\top + \mathbf{\Sigma}^{\mathbf{U}_:}\right)\right],
\end{aligned}
\tag{C.7}
$$

where

$$
\Psi = \mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}^H_{\mathbf{fU}} \otimes \mathbf{K}^X_{\mathbf{fU}}\right] = \mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}^H_{\mathbf{fU}}\right] \otimes \mathbf{K}^X_{\mathbf{fU}} = \Psi^H \otimes \mathbf{K}^X_{\mathbf{fU}}, \tag{C.8}
$$

$$
\psi = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{ff}}\right]\right) = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K}^H_{\mathbf{ff}} \otimes \mathbf{K}^X_{\mathbf{ff}}\right]\right), \tag{C.9}
$$

$$
\Phi = \mathbb{E}_{q(\mathbf{H})}\left[\mathbf{K_{fU}}^\top \mathbf{K_{fU}}\right] = \mathbb{E}_{q(\mathbf{H})}\left[\left(\mathbf{K}^H_{\mathbf{fU}} \otimes \mathbf{K}^X_{\mathbf{fU}}\right)^\top \left(\mathbf{K}^H_{\mathbf{fU}} \otimes \mathbf{K}^X_{\mathbf{fU}}\right)\right] = \Phi^H \otimes \left(\mathbf{K}^X_{\mathbf{fU}}\right)^\top \mathbf{K}^X_{\mathbf{fU}}. \tag{C.10}
$$

C.3 More Efficient Formulation

We can reduce the computational complexity in $\mathcal{F}$ and Kullback–Leibler divergence by taking the advantage of the Kronecker product decomposition.

C.3.1 More Efficient Formulation Given the Same Input Datasets

In this section, given the same input datasets, we re-define $\mathcal{F}$ and Kullback–Leibler divergence by using the Kronecker product decomposition.

$$\begin{aligned}
\mathcal{F} =& -\frac{NDR}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\mathbf{y}^\top\mathbf{y} \\
&- \frac{1}{2\sigma^2}\mathrm{Tr}\left(\left(\mathbf{K}^H_{\mathbf{UU}}\otimes\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\left(\Phi^H\otimes\Phi^X\right)\left(\mathbf{K}^H_{\mathbf{UU}}\otimes\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\mathbf{M}_:\mathbf{M}_:^\top\right) \\
&- \frac{1}{2\sigma^2}\mathrm{Tr}\left(\left(\mathbf{K}^H_{\mathbf{UU}}\otimes\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\left(\Phi^H\otimes\Phi^X\right)\left(\mathbf{K}^H_{\mathbf{UU}}\otimes\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\left(\mathbf{\Sigma}^{H_:}\otimes\mathbf{\Sigma}^{X_:}\right)\right) \\
&+ \frac{1}{\sigma^2}\mathbf{y}^\top\left(\Psi^H\otimes\mathbf{K}^X_{\mathbf{fU}}\right)\left(\mathbf{K}^H_{\mathbf{UU}}\otimes\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\mathbf{M}_: - \frac{1}{2\sigma^2}\psi \\
&+ \frac{1}{2\sigma^2}\left(\mathrm{Tr}\left(\left(\mathbf{K}^H_{\mathbf{UU}}\otimes\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\left(\Phi^H\otimes\Phi^X\right)\right)\right) \\
=& -\frac{NDR}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\mathbf{y}^\top\mathbf{y} \\
&- \frac{1}{2\sigma^2}\mathrm{Tr}\left(\left(\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\Phi^H\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\right)\otimes\left(\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\Phi^X\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\right)\mathbf{M}_:\mathbf{M}_:^\top\right) \\
&- \frac{1}{2\sigma^2}\mathrm{Tr}\left(\left(\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\Phi^H\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\mathbf{\Sigma}^{H_:}\right)\otimes\left(\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\Phi^X\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\mathbf{\Sigma}^{X_:}\right)\right) \\
&+ \frac{1}{\sigma^2}\mathbf{y}^\top\left(\Psi^H\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\otimes\mathbf{K}^X_{\mathbf{fU}}\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\right)\mathbf{M}_: - \frac{1}{2\sigma^2}\psi \\
&+ \frac{1}{2\sigma^2}\left(\mathrm{Tr}\left(\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\Phi^H\otimes\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\Phi^X\right)\right) \\
=& -\frac{NDR}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\mathbf{y}^\top\mathbf{y} \\
&- \frac{1}{2\sigma^2}\mathrm{Tr}\left(\mathbf{M}^\top\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\Phi^X\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\mathbf{M}\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\Phi^H\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\right) \\
&- \frac{1}{2\sigma^2}\mathrm{Tr}\left(\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\Phi^H\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\mathbf{\Sigma}^{H_:}\right)\mathrm{Tr}\left(\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\Phi^X\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\mathbf{\Sigma}^{X_:}\right) \\
&+ \frac{1}{\sigma^2}\mathbf{y}^\top\left(\mathbf{K}^X_{\mathbf{fU}}\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\mathbf{M}\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\left(\Psi^H\right)^\top\right)_: - \frac{1}{2\sigma^2}\psi \\
&+ \frac{1}{2\sigma^2}\mathrm{Tr}\left(\left(\mathbf{K}^H_{\mathbf{UU}}\right)^{-1}\Phi^H\right)\mathrm{Tr}\left(\left(\mathbf{K}^X_{\mathbf{UU}}\right)^{-1}\Phi^X\right).
\end{aligned} \tag{C.11}$$

We also assume there is a Kronecker product decomposition of the covariance matrix of $q(\mathbf{U}_:)$, $\mathbf{\Sigma}^{\mathbf{U}_:} = \mathbf{\Sigma}^{H_:}\otimes\mathbf{\Sigma}^{X_:}$ so the KL-divergence between $q(\mathbf{U}_:)$ and $p(\mathbf{U}_:)$ can also take advantage of the decomposition:

$$\begin{aligned}
\mathrm{KL}\left\{q\left(\mathbf{U}_{:}\right) \mid p\left(\mathbf{U}_{:}\right)\right\} =& \frac{1}{2}\Bigg(\log\left|\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\left(\boldsymbol{\Sigma}^{H_{:}}\otimes\boldsymbol{\Sigma}^{X_{:}}\right)^{-1}\right| \\
&+\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\left(\mathbf{M}_{:}\mathbf{M}_{:}^{\top}+\boldsymbol{\Sigma}^{H_{:}}\otimes\boldsymbol{\Sigma}^{X_{:}}-\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)\right)\right)\Bigg) \\
=& \frac{1}{2}\Bigg(\log\left|\mathbf{K}_{\mathbf{UU}}^{H}\left(\boldsymbol{\Sigma}^{H_{:}}\right)^{-1}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\left(\boldsymbol{\Sigma}^{X_{:}}\right)^{-1}\right| \\
&+\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\left(\mathbf{M}_{:}\mathbf{M}_{:}^{\top}+\boldsymbol{\Sigma}^{H_{:}}\otimes\boldsymbol{\Sigma}^{X_{:}}\right)\right)-M_{H}M_{X}\Bigg) \\
=& \frac{1}{2}\Bigg(M_{X}\log\left|\mathbf{K}_{\mathbf{UU}}^{H}\left(\boldsymbol{\Sigma}^{H_{:}}\right)^{-1}\right|+M_{H}\log\left|\mathbf{K}_{\mathbf{UU}}^{X}\left(\boldsymbol{\Sigma}^{X_{:}}\right)^{-1}\right| \\
&+\mathrm{Tr}\left(\mathbf{M}^{\top}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\right) \\
&+\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\boldsymbol{\Sigma}^{H_{:}}\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\boldsymbol{\Sigma}^{X_{:}}\right)-M_{H}M_{X}\Bigg) \\
=& \frac{1}{2}\Bigg(M_{X}\log\frac{\left|\mathbf{K}_{\mathbf{UU}}^{H}\right|}{\left|\boldsymbol{\Sigma}^{H_{:}}\right|}+M_{H}\log\frac{\left|\mathbf{K}_{\mathbf{UU}}^{X}\right|}{\left|\boldsymbol{\Sigma}^{X_{:}}\right|}+\mathrm{Tr}\left(\mathbf{M}^{\top}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\right) \\
&+\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\boldsymbol{\Sigma}^{H_{:}}\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\boldsymbol{\Sigma}^{X_{:}}\right)-M_{H}M_{X}\Bigg). \qquad \text{(C.12)}
\end{aligned}$$

C.3.2 More Efficient Formulation Given the Different Input Datasets

In this section, given the different input datasets, we re-define $\mathcal{F}$ using the Kronecker product decomposition.

$$\begin{aligned}
\mathcal{F}=\sum_{d=1}^{D} & -\frac{N_{d}R}{2}\log 2\pi\sigma_{d}^{2}-\frac{1}{2\sigma_{d}^{2}}\mathbf{y}_{d}^{\top}\mathbf{y}_{d} \\
&-\frac{1}{2\sigma_{d}^{2}}\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\left(\Phi_{d}^{H}\otimes\Phi_{d}^{X}\right)\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}_{:}\mathbf{M}_{:}^{\top}\right) \\
&-\frac{1}{2\sigma_{d}^{2}}\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\left(\Phi_{d}^{H}\otimes\Phi_{d}^{X}\right)\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\left(\boldsymbol{\Sigma}^{H_{:}}\otimes\boldsymbol{\Sigma}^{X_{:}}\right)\right) \\
&+\frac{1}{\sigma_{d}^{2}}\mathbf{y}_{d}^{\top}\left(\Psi_{d}^{H}\otimes\mathbf{K}_{\mathbf{f}_{d}\mathbf{U}}^{X}\right)\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}_{:}-\frac{1}{2\sigma_{d}^{2}}\psi_{d} \\
&+\frac{1}{2\sigma_{d}^{2}}\left(\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\otimes\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\left(\Phi_{d}^{H}\otimes\Phi_{d}^{X}\right)\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{d=1}^{D} -\frac{N_d R}{2}\log 2\pi\sigma_d^2 - \frac{1}{2\sigma_d^2}\mathbf{y}_d^\top \mathbf{y}_d \\
&\quad - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left(\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi_d^H\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\right)\otimes\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi_d^X\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\right)\mathbf{M}_{:}\mathbf{M}_{:}^{\top}\right) \\
&\quad - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left(\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi_d^H\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\mathbf{\Sigma}^{H_:}\right)\otimes\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi_d^X\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{\Sigma}^{X_:}\right)\right) \\
&\quad + \frac{1}{\sigma_d^2}\mathbf{y}_d^\top\left(\Psi_d^H\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\otimes\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{X}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\right)\mathbf{M}_{:} - \frac{1}{2\sigma_d^2}\psi_d \\
&\quad + \frac{1}{2\sigma_d^2}\left(\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi_d^H\otimes\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi_d^X\right)\right) \\
&= \sum_{d=1}^{D} -\frac{N_d R}{2}\log 2\pi\sigma_d^2 - \frac{1}{2\sigma_d^2}\mathbf{y}_d^\top \mathbf{y}_d \\
&\quad - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left(\mathbf{M}^\top\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi_d^X\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi_d^H\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\right) \\
&\quad - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi_d^H\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\mathbf{\Sigma}^{H_:}\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi_d^X\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{\Sigma}^{X_:}\right) \\
&\quad + \frac{1}{\sigma_d^2}\mathbf{y}_d^\top\left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{X}\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\mathbf{M}\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\left(\Psi_d^H\right)^\top\right)_{:} - \frac{1}{2\sigma_d^2}\psi_d \\
&\quad + \frac{1}{2\sigma_d^2}\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{H}\right)^{-1}\Phi_d^H\right)\mathrm{Tr}\left(\left(\mathbf{K}_{\mathbf{UU}}^{X}\right)^{-1}\Phi_d^X\right).
\end{aligned}
\tag{C.13}
$$

C.4 Derivation of $\mathcal{F}$ Given Different Input Datasets

In this section, we show details for deriving $\mathcal{F}$ using different input datasets:

$$
\begin{aligned}
\mathcal{F} &= \mathbb{E}_{p(\mathbf{f}\mid\mathbf{U}_:,\mathbf{H})q(\mathbf{U}_:)q(\mathbf{H})}\left[\log p\left(\mathbf{y}\mid\mathbf{f},\mathbf{H}\right)\right] \\
&= \int q(\mathbf{H})\int q\left(\mathbf{U}_:\right)\underbrace{\int p\left(\mathbf{f}\mid\mathbf{U}_:,\mathbf{H}\right)\log p\left(\mathbf{y}\mid\mathbf{f},\mathbf{H}\right)\mathrm{d}\mathbf{f}}_{\mathcal{L}_F}\mathrm{d}\mathbf{U}_:\mathrm{d}\mathbf{H} \\
&= \int q(\mathbf{H})\underbrace{\int q\left(\mathbf{U}_:\right)\mathcal{L}_F\mathrm{d}\mathbf{U}_:}_{\mathcal{L}_U}\mathrm{d}\mathbf{H} \\
&= \underbrace{\int q(\mathbf{H})\mathcal{L}_U\mathrm{d}\mathbf{H}}_{\mathcal{L}_H}.
\end{aligned}
\tag{C.14}
$$

Now, we calculate $\mathcal{L}_F$:

$$\begin{aligned}\mathcal{L}_F &= \int \prod_{d=1}^{D} p\left(\mathbf{f}_d \mid \mathbf{U}_{:}, \mathbf{H}\right) \log \prod_{d=1}^{D} p\left(\mathbf{y}_d \mid \mathbf{f}_d, \mathbf{H}\right) \mathrm{d}\mathbf{f}_d \\ &= \sum_{d=1}^{D} \int p\left(\mathbf{f}_d \mid \mathbf{U}_{:}, \mathbf{H}\right) \log p\left(\mathbf{y}_d \mid \mathbf{f}_d, \mathbf{H}\right) \mathrm{d}\mathbf{f}_d \\ &= \sum_{d=1}^{D} \left(\log \mathcal{N}\left(\mathbf{y}_d \mid \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{U}_{:}, \sigma_d^2\right) - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d} - \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\right] \right), \qquad \text{(C.15)}\end{aligned}$$

where $p\left(\mathbf{f}_d \mid \mathbf{U}_{:}, \mathbf{H}\right) = \mathcal{N}\left(\mathbf{f}_d \mid \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{U}_{:}, \mathbf{K}_{\mathbf{f}_d\mathbf{f}_d} - \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\right)$. Then, we consider the $\mathcal{L}_U$:

$$\begin{aligned}\mathcal{L}_U &= \int q\left(\mathbf{U}_{:}\right) \mathcal{L}_F \mathrm{d}\mathbf{U}_{:} \\ &= \int q\left(\mathbf{U}_{:}\right) \sum_{d=1}^{D} \left(\log \mathcal{N}\left(\mathbf{y}_d \mid \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{U}_{:}, \sigma_d^2\right) - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d} - \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\right] \right) \mathrm{d}\mathbf{U}_{:} \\ &= \sum_{d=1}^{D} \left(\log \mathcal{N}\left(\mathbf{y}_d \mid \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:}, \sigma_d^2\right) - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d} - \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\right] \right. \\ &\quad \left. - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left[\boldsymbol{\Sigma}^{\mathbf{U}_{:}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\right] \right), \qquad \text{(C.16)}\end{aligned}$$

where $q(\mathbf{U}_{:}) = \mathcal{N}\left(\mathbf{U}_{:} \mid \mathbf{M}_{:}, \boldsymbol{\Sigma}^{\mathbf{U}_{:}}\right)$. Further, we obtain $\mathcal{L}_H$:

$$\begin{aligned}\mathcal{L}_H &= \int q(\mathbf{H}) \mathcal{L}_U \mathrm{d}\mathbf{H} \\ &= \sum_{d=1}^{D} \mathbb{E}_{q(\mathbf{h}_d)}\left[\log \mathcal{N}\left(\mathbf{y}_d \mid \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:}, \sigma_d^2\right)\right] \\ &\quad - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left[\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}\right] - \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\right]\right] \\ &\quad - \frac{1}{2\sigma_d^2} \mathrm{Tr}\left[\boldsymbol{\Sigma}^{\mathbf{U}_{:}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^{\top}\mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]\mathbf{K}_{\mathbf{UU}}^{-1}\right].\end{aligned}$$

Then,

$$
\begin{aligned}
\mathcal{L}_H = \sum_{d=1}^{D} & \underbrace{-\frac{N_d R}{2}\log 2\pi\sigma_d^2 - \frac{1}{2\sigma_d^2}\mathbf{y}_d^\top \mathbf{y}_d}_{\mathbf{C}_d} + \frac{1}{\sigma_d^2}\mathbf{y}_d^\top \underbrace{\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]}_{\Psi_d} \mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:} \\
& - \frac{1}{2\sigma_d^2}\mathbf{M}_{:}^\top \mathbf{K}_{\mathbf{UU}}^{-1} \underbrace{\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]}_{\Phi_d} \mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:} - \frac{1}{2\sigma_d^2}\underbrace{\mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}\right]\right)}_{\psi_d} \\
& + \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\mathbf{K}_{\mathbf{UU}}^{-1}\underbrace{\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]}_{\Phi_d}\right] - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\boldsymbol{\Sigma}^{\mathbf{U}_:}\mathbf{K}_{\mathbf{UU}}^{-1}\underbrace{\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right]}_{\Phi_d}\mathbf{K}_{\mathbf{UU}}^{-1}\right] \\
= \sum_{d=1}^{D} & \mathbf{C}_d + \frac{1}{\sigma_d^2}\mathbf{y}_d^\top \Psi_d \mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:} - \frac{1}{2\sigma_d^2}\mathbf{M}_{:}^\top \mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:} - \frac{1}{2\sigma_d^2}\psi_d \\
& + \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d\right] - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\boldsymbol{\Sigma}^{\mathbf{U}_:}\mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d\mathbf{K}_{\mathbf{UU}}^{-1}\right] \\
= \sum_{d=1}^{D} & \mathbf{C}_d + \frac{1}{\sigma_d^2}\mathbf{y}_d^\top \Psi_d \mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{M}_{:} - \frac{1}{2\sigma_d^2}\left(\psi_d - \mathrm{Tr}\left[\mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d\right]\right) \\
& - \frac{1}{2\sigma_d^2}\mathrm{Tr}\left[\mathbf{K}_{\mathbf{UU}}^{-1}\Phi_d\mathbf{K}_{\mathbf{UU}}^{-1}\left(\mathbf{M}_{:}\mathbf{M}_{:}^\top + \boldsymbol{\Sigma}^{\mathbf{U}_:}\right)\right], \qquad \text{(C.17)}
\end{aligned}
$$

where $q(\mathbf{H}) = \prod_{d=1}^{D} q(\mathbf{h}_d)$ and

$$
\Psi_d = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right] = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H\right] \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X = \Psi_d^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X \qquad \text{(C.18)}
$$

$$
\psi_d = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}\right]\right) = \mathrm{Tr}\left(\mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{f}_d}^X\right]\right), \qquad \text{(C.19)}
$$

$$
\begin{aligned}
\Phi_d = \mathbb{E}_{q(\mathbf{h}_d)}\left[\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}\right] &= \mathbb{E}_{q(\mathbf{h}_d)}\left[\left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)^\top \left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H \otimes \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)\right] \\
&= \mathbb{E}_{q(\mathbf{h}_d)}\left[\left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H\right)^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^H\right] \otimes \left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X \\
&= \Phi_d^H \otimes \left(\mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X\right)^\top \mathbf{K}_{\mathbf{f}_d\mathbf{U}}^X. \qquad \text{(C.20)}
\end{aligned}
$$