Gaussian Processes for Data Scarcity Challenges

The University Of Sheffield

Fariba Yousefi

Department of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

March 2021

I would like to dedicate this thesis to my loving 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.

Fariba Yousefi
March 2021

Acknowledgements

I would like to thank my supervisors Dr. Mauricio A. Álvarez and Prof. Neil Lawrence for their invaluable guidance, support and patience during my studies. Neil provided an environment which felt like family in the early days of my PhD and I was lucky to meet many great researchers within his group and external visitors. Mauricio taught me how to be a better researcher and has always being a good friend to me. I feel privileged to have worked under his supervision for the last few years of my PhD.

I would like to thank Dr. Carl Henrik Ek for his invaluable advice, guidance and close collaborations in my first years of PhD.

I would like to thank Dr. Zhenwen Dai who thought me so much about Gaussian processes and who was always there to discuss ideas and mathematical details.

I would like to thank all my friends and colleagues for inspiring discussions over lunch. It has been a pleasure to get to know them. I would like to especially thank to those friends who helped in proofreading different sections of this thesis and discussions during my PhD; Ricardo Andrade, Andreas Damianou, Mike Smith, Alan Saul, Federico Tomasi, Zhenwen Dai, Javier Gonzalez, Max Zwiessele, James Hensman, Alessandra Tosi, Arif Rahman, Mike Croucher, Luisa Cutillo, Andrew Killer, Juan Jose Giraldo, Pablo Moreno Muñoz, Senee Kitimoon and Chunchao Ma.

I would like to thank the Department of Computer Science and Control Point Ltd for their scholarship and financial support during my PhD. Especially I would like to thank Prof. Guy Brown, Prof. Eleni Vasilaki, Prof. Jon Barker, Joanne Suter, Gillian Callaghan, Sarah Brown, Caroline Wyer and Karen Barker for their endless support at the department. Especially I would like to thank Eric Bridgstock (who is sadly no longer with us), Steve Hamshow, Nick Mark, Rob Eavis and Andrew Killer for their support and useful discussions.

I would like to thank my family for sending their love from miles away, especially my mother for her unconditional love and support.

Finally I would like to thank my partner, Morrie, for his love, support, patience and yummy dinners during my studies.

Abstract

This thesis focuses on Gaussian process models specifically designed for scarce data problems. Data scarcity or lack of data can be a weak spot for many machine learning algorithms. Nevertheless, both are commonly found in a diverse set of applications such as medicine, quality assurance, and remote sensing. Supervised classification algorithms can require large amounts of labeled data, and fulfilling this requirement is not straightforward.

In medicine, breast cancer datasets typically have few cancerous cells and many healthy cells due to the overall relative scarcity of cancerous cells versus non-cancerous ones. The lack of cancerous cells causes the dataset to be imbalanced, which makes it difficult for learning algorithms to learn the differences between cancerous and healthy cells. A similar imbalance exists in the quality assurance industry, in which the ratio of faulty to non-faulty cases is very low. In sensor networks, and in particular those which measure air pollution across cities, combining sensors of different qualities can help fill gaps in what is often a very data scarce landscape.

In data scarce scenarios, we present a probabilistic latent variable model that can cope with imbalanced data. By incorporating label information, we develop a kernel that can capture shared and private characteristics of data separately. On the other hand, in cases where no labels are available, an active learning based technique is proposed, based on a Gaussian process classifier with an oracle in the loop to annotate only the data about which the algorithm is uncertain. Finally, when disparate data types with different granularity levels are available, a transfer learning based approach is proposed. We show that jointly modeling data with various granularity helps improve prediction of rare data.

The developed methods are demonstrated in experiments with real and synthetic data. The results presented in this thesis show that the developed methods improve prediction for scarce data problems with various granularities.

Table of contents

List of figures

List of tables

Nomenclature

Gaussian process

$\mathcal{R}$	Real space
$\mathcal{O}$	Order
p	Dimension of the input space
f	Latent mapping function
q	Dimension of the latent space
N	Number of data points per output
D	Number of outputs
M	Number of inducing points
$\mathbf{X}$	Input data or latent data, $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$
Q	Number of latent functions
Y	Observed output data
$\mathbf{Z}$	Inducing inputs, $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^{M}$
ε	Noise model
σ^2	Gaussian likelihood variance
θ_K	Hyper-parameters of kernel
θ_L	Parameters of likelihood
υ	Support that refers to area or volume

$u_q(\mathbf{z})$ q-th latent function evaluated at $\mathbf{z}$

$k_q(\mathbf{z},\mathbf{z}')$ Covariance function for the Gaussian process of $u_q(\mathbf{z})$

$f_d(\upsilon)$ d-th output evaluated at support υ

$\delta_{\alpha,\beta}$ Kronecker delta between α and β

$k_{f_d,f_{d'}}(\upsilon,\upsilon')$ Cross-covariance between outputs $f_d(\upsilon)$ and $f'_d(\upsilon')$

$k_{f_d,u_q}(\upsilon,\mathbf{z}')$ Cross-covariance between output $f_d(\upsilon)$ and latent $u_q(\mathbf{z}')$

$\mathbf{u}$ Inducing points

$\mathbf{u}_q$ $u_q(\mathbf{z})$ evaluated at $\mathbf{Z}$, $\mathbf{u}_q = [u_q(\mathbf{z}_1), u_q(\mathbf{z}_2), ..., u_q(\mathbf{z}_M)]^\top$

$\mathbf{I}$ Identity matrix

$\mathbf{K}_{ff}$ Kernel function for f, evaluated $k(\mathbf{x},\mathbf{x})$ with $\mathbf{x} \in X$

$\mathbf{K}_{fu}$ Kernel function for f and u, evaluated $k(\mathbf{x},\mathbf{z})$ with $\mathbf{x} \in X$ and $\mathbf{z} \in Z$

Abbreviations

erf Error function

AMTGP Aggregated multi-task Gaussian process

ARD Automatic relevance determination

AUC Area under curve

BGPLVM Bayesian Gaussian process latent variable model

DPPCA Dual probablistic principle component analysis

ELBO Evidence lower bound

EM Expectation maximization

FN False negative

FP False positive

FPR False positive rate

GP Gaussian process

GPLVM Gaussian process latent variable model

i.i.d. Independent and identically distributed

KL Kullback-Leibler

LMC Linear model of coregionalization

MAP Maximum a posteriori

PCA Principle component analysis

PPCA Probablistic principle component analysis

RBF Radial basis function, also called exponentiated quadratic (EQ)

ROC Receiver operating characteristic

SCLVM Structured consolidation latent variable model

SMOTE Synthetic minority oversampling technique

SMSE Standardized mean squared error

SNLP Standardized negative log probability density

SVI Stochastic variational inference

TN True negative

TP True positive

TPR True positive rate or sensitivity

Operators

$\langle . \rangle$ Expected value

$\mathrm{tr}(.)$ Trace of a matrix

$E[.]$ Expected value

Chapter 1

Introduction

"All models are wrong; some are useful."

George Box

Machine learning is a subfield of artificial intelligence that focuses on the mathematical extraction of useful information from *data*. The rise of machine learning has been made possible through increased availability of data and computation power. Machine learning is also very dependent on the models being used. One of the ways to observe a machine learning process is through the relationship between data, model and inference: *data+model=prediction*.

Data scarcity — Although ***data*** is growing at an exponential rate, and large amounts of storage are available, this has not solved the ***data scarcity*** issues that arise in machine learning problems. Several real-life applications, in fields such as medicine, ecology, finance, and social media often suffer from data scarcity. For example, in medicine, diagnosing a rare disease or detecting cancerous cells when most of the cells in a body are healthy might be the main goal. In the rare disease case, acquiring data is very difficult since it only affects a small percentage of the population. In the cancer detection case, most tumors turn out to be benign, and only a small percentage of patients have malignant tumors. Furthermore, depending on the stage of the disease, cancer patients usually have more healthy cells than cancerous cells. Data scarcity, or lack of data, is one of the main bottlenecks for learning algorithms. In scarce data problems, the amount of data acquired is very small compared to what is needed.

Imbalanced data — Generally speaking, in the context of classification, any dataset with a different number of samples between classes is considered 'imbalanced'. Data scarcity often gives rise to such imbalanced (also known as skewed) problems. In skewed problems, the most interesting class is usually under-represented, which means that we do not have enough data for that class. Most available methods expect a balanced number of samples.

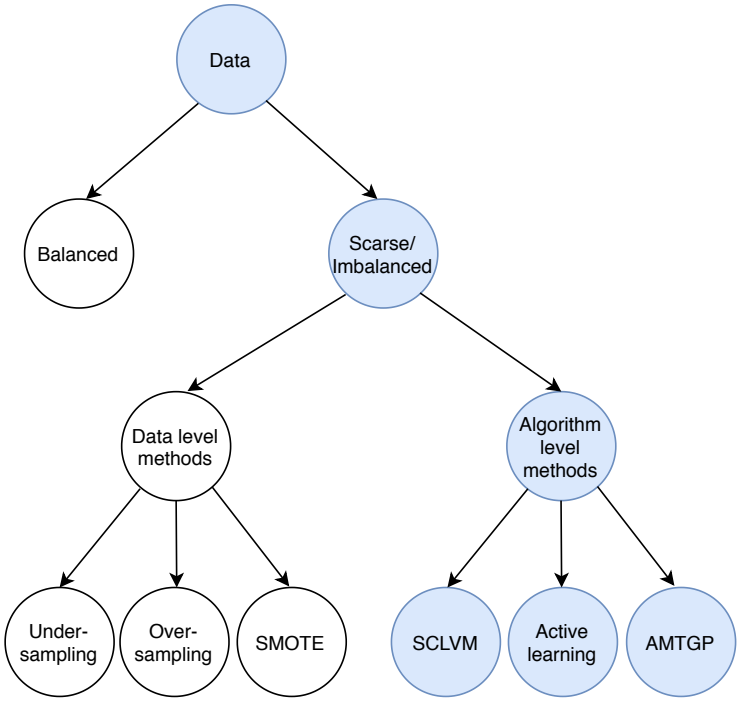


Fig. 1.1 Data roadmap of the thesis. In this figure SMOTE is short for Synthetic Minority Oversampling TEchnique. SCLVM is abbreviation for Structured Consolidation Latent Variable Model and AMTGP is short for Aggregated Multi-task Gaussian Process.

When using an imbalanced dataset, the result is often that minority samples are ignored and that the minority set performs poorly, which are often the most interesting to the user.

Figure 1.1 illustrates the data roadmap in this thesis to tackle the scarce/imbalanced data problem. Structured Consolidation Latent Variable Model (SCLVM), Active learning and Aggregated Multi-task Gaussian Process (AMTGP) will be explained in more details in the following chapters.

Data annotation — Most machine learning algorithms require large amounts of data, as well as their labels, in cases of supervised learning. In some cases the data at hand does not contain any labels; in this case, the data needs to be ***annotated*** by an oracle (human expert). Annotation is the process of labeling data by humans for machine learning algorithms. Annotation also helps these algorithms to identify patterns in data and understand which parts of the data humans are interested in. This helps models to recognize similar patterns in the future to predict results. Most algorithms depend on data labels to perform better. However, annotating data can be a tedious, time-consuming and hence expensive job. There are also different levels or granularity of annotation [Chen et al., 2018]. For example, in image classification problems, labels can be *low-level* or *high-level* (also known as fine-grained or coarse-grained, respectively). ***Low-level*** labels are those where the image has

pixel level labeling (also called image segmentation). For example, segmenting an area of interest is considered low-level annotation. ***High-level*** labels, on the other hand, are those where images have global labels per the whole image. In the medical domain, these are often called *patient-level* labels. For example, prostate cancer images can have labels that indicate whether a patient has cancer or not. However, such a label does not provide any information about which part of the image displays cancerous cells. This can happen when carefully labeling the data is not possible, or very expensive: it is usually very expensive for pathologists to carefully label cancerous cells. This distinction is not only relevant to medical images; related issues are common in other image datasets as well. In the popular ImageNet [Russakovsky et al., 2015] dataset, classes such as 'cat' or 'dog' might be of interest. However, there are no labels for the exact locations of those objects in the images in question and there might be other items in the background. Instead, there are only high-level labels which state there is a cat or a dog in the whole image, and this might confuse the learning algorithm. In contrast to ImageNet, in real-world problems, data collection and labeling can be difficult because they require specialized equipment and experts to collect and label the data.

Case study 1: histopathology images — As mentioned earlier, in the medical domain data scarcity is a big issue. This thesis will examine some challenges associated with histopathology images, and in particular breast cancer data. Breast cancer is one of the most common cancers among American women in recent years. It is estimated that about 1 in 38 women (2.6%) will die because of breast cancer in the United States [Society, 2020]. In these domains, negatively labeled data is often easy to obtain — e.g. healthy cells. Positive labels, on the other hand, can be difficult to acquire. Correct diagnosis is very important for early intervention. Mitosis detection is a stage in tumor assessment that involves determining whether individual cells are in the process of dividing to reproduce. Automatic mitosis (cell division) detection could aid pathologists in their work, and also could alleviate the tedious and time consuming process of manual mitosis counting. Pathologists sometimes need to repeat this counting process for different areas or sections that are borderline cases of cancer versus non-cancer diagnosis [Veta et al., 2015]. Cell divisions within a tumor can give an indication of the rate of growth of that tumor [Snell, 2013]. Figure 1.2 has a few different samples of mitosis image patches. This figure shows the difficulties of labelling data in the medical domain for non-experts. These images will look the same to non-specialists, which suggests that this type of data can only be annotated by trained experts. The Assessment of Mitosis Detection Algorithms 2013 (AMIDA13) dataset is publicly available [Veta et al., 2015]. This dataset has been annotated by pathologists. It does not have low-level labels, and each mitosis area is not segmented; instead, it has high-level labels. Similar to other

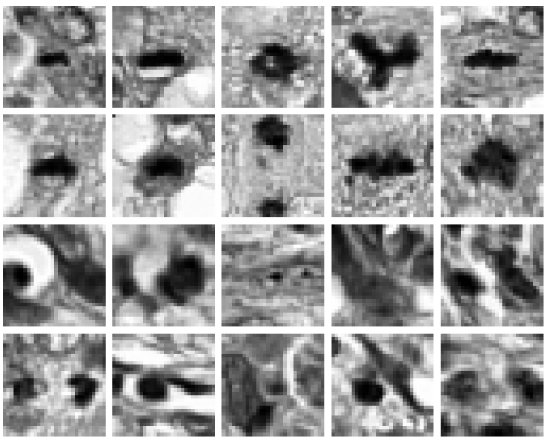


Fig. 1.2 Samples of the breast cancer dataset patches. The first two rows correspond to the positively labeled image patches, and the last two rows correspond to the negatively labeled image patches.

medical datasets, this dataset also suffers from data scarcity issues. It contains around 146000 negative cells but only 550 positive cells. Lack of positive data (cancerous cells) causes the dataset to be imbalanced and makes it difficult for a learning algorithm to learn the differences between cancerous cells and healthy ones.

Case study 2: electrofusion industry — Data scarcity affects many other real-world applications of machine learning as well, such as quality control and fault detection problems. In particular, this work will look in more detail at the data scarcity problem in the electrofusion industry. The electrofusion industry has a growing problem of polyethylene pipeline failure, and this not only costs millions of pounds but also gives a bad reputation to the polyethylene pipe industry. It has been proven that over 99% of joint failures are caused by poor practices before the welding process.[1] ControlPoint is a quality assurance company based in Chesterfield, UK[2] that remotely identifies faults, environmental factors, and poor installations in water and gas pipes. One important factor in proper installations of water and gas pipes is to have a clamp in place. It is essential that clamps are used when making electrofusion joints to ensure that the pipe and joint are correctly aligned and stabilized during the fusion process, which permanently welds pieces of pipe together to form a homogeneous joint. In the absence of clamps, there is a risk that pipes may be misaligned within the joint, with the possibility of gaps occurring, and therefore a potential source of weakness, resulting

[1] controlpoint.co.uk/about/ (last accessed 20.07.2020).
[2] controlpoint.co.uk/ (last accessed 20.07.2020)

in a higher risk of failure and leakage. Uncontrolled movement of the joint during welding may also occur in the absence of clamps, which is likely to result in a loss of intimate contact of the welded areas, and therefore result in a poorer weld. A few samples of these images are presented in Figure 1.3. These images come in a variety of colors, sizes, angles and lighting conditions. This dataset has high-level labeling — in other words, image-level labeling and clamps are not segmented. In this dataset, 99% of images contain a clamp, which makes detecting non-clamp images very difficult, due to data scarcity issues.

Fig. 1.3 Samples of the water and gas pipe images.

To ensure the quality of these joints, human experts check these images manually and score them based on how well the workers in the trenches used the guidelines for the joints. This process is very tiresome, since there are thousands of images coming in every day, and automating it would help the experts only be required to focus on the fewer cases that are on the borderline and which the algorithm is uncertain about. Automatically detecting non-clamps is very important because of the issues mentioned above, such as the possibility of leakages. So in this problem we have high-level labels and data scarcity issues, and as mentioned before, data scarcity in this case results in an imbalanced problem between clamp and non-clamp classes.

Case study 3: sensor networks — Data scarcity also affects the fields of ecology, epidemiology, remote sensing and demography. To give some examples, in sensor networks, correlated variables are measured at different resolutions, intervals or scales. In the near future, air pollution monitoring across cities and regions could be carried out using a combination of a few high-quality (high-cost) low time-resolution sensors and several low-quality

(low-cost) high time-resolution sensors. Since high-quality sensors can be very expensive, most developing countries, such as Uganda, cannot afford to use them. Instead, low-quality sensors are widely used. In this thesis we investigate air pollution dataset from two fine particulate matter (PM) sensors that are co-located in Kampala, Uganda. These sensors have a diameter of less than 2.5 micrometer (PM2.5). Particulate air pollution can be measured accurately with high temporal precision by using a β attenuation (BAM) sensor or similar. Unfortunately these are often quite expensive. Instead, one can combine the measurements from a low-cost optical particle counter (OPC) which gives good temporal resolution but is often badly biased — these are scaled to either be too large, e.g. due to humidity, or too small, e.g. due to dust contamination on the OPC sensor — with the results of a Cascade Impactors (CIs), which are a cheaper method for assessing the mass of particulate species, but which require the weighing of filters post-sampling. This allows for longer periods over which the measurements are integrated (e.g. 6 or 24 hours). In Figure 1.4, the upper plot refers to a cheap air pollution sensor with low-accuracy and high-resolution data, which is noisy and gives data for every 5 minute period. The lower plot refers to an expensive sensor with high-accuracy and low-resolution data, which would give data every 15 minutes.

This problem goes beyond single-task settings where only one task can be modeled at a time. In single-task learning, potential correlations among tasks are ignored, and it is assumed that the tasks are independent of each other [Alvarez et al., 2012]. If the problem at hand contains different tasks including, but not limited to, different intervals of data or different qualities of sensors, modeling these tasks jointly can be beneficial. This is called multi-task learning, and in this setting one can exploit the interaction between different tasks to improve individual predictions and also cope with data scarcity [Alvarez et al., 2012; Journel and Huijbregts, 1978]. For example, in the case of air pollution sensors, we have a few high-quality data and many more low-quality data, and using prior information between tasks can improve the prediction of high-quality and / or rare data.

Most of the current state of the art deep learning algorithms have a serious limitation: they cannot handle cases in which there is limited data. Deep learning methods are data-hungry and typically need a huge amount of data (millions or even billions of training examples) to be able to be trained effectively [Marcus, 2018]. Here we are more interested in methods that are data efficient and can handle cases where data is scarce.

Gaussian processes — Bayesian non-parametrics are a useful and efficient framework for modeling complex data. Bayesian non-parametrics combine large parameter spaces with probability density estimation over those parameter spaces [Hjort et al., 2010]. In non-parametric models, the size of the parameters can grow with the size of dataset, and so the information that parameters can capture grows with the growing size of data. This makes

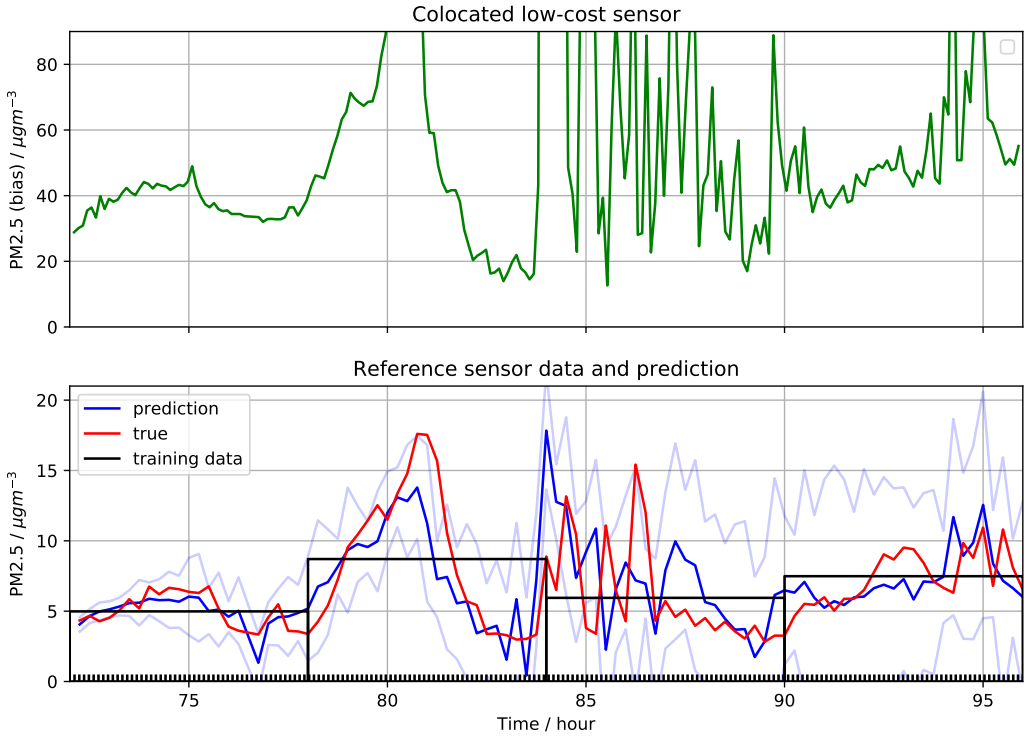


Fig. 1.4 Upper plot: a (biased) OPC low-accuracy high-frequency measurement of PM2.5 air pollution. Lower plot: the high-precision low-frequency data.

non-parametric models very flexible [Ghahramani, 2013]. Non-parametric models can be viewed as having infinitely many parameters, and their complexity grows with the number of data itself.

Gaussian processes are among the most frequently used non-parametric models when it comes to uncertainty quantification and probabilistic modeling. A GP allows for a principled way of handling the uncertainty of unknown functions. There are many different ways to learn functions, and probabilistic inference is a neat way of doing it. By combining prior knowledge about the function and information provided by observations, an inference is possible over all possible functions. A GP is defined by its two moments: mean and covariance function. GPs are simple to understand, easy to analyze, and interpretable (by allowing uncertainty quantification), which is vital in many real-world applications. This thesis attempts to tackle a number of challenges that arise from data scarcity and lack of labeled data using advances in Gaussian processes. To perform probabilistic inference, the prior information is used to express one's belief in the form of a prior distribution. Then the posterior distribution encodes the uncertainty in our predictions.

1.1 Outline of the thesis

- Chapter 2 contains a brief introduction to the imbalanced data problem and Gaussian processes. Gaussian process regression and classification is explained. Approximation techniques are discussed. Also, latent variable models are introduced, and the Gaussian process latent variable model is discussed. Finally, the sparse Gaussian process is explained.
- Chapter 3 is concerned with latent variable models for dimensionality reduction. It introduces the Structure Consolidation Latent Variable Model (SCLVM), where the input is latent to the model. This model, based on Yousefi et al. [2016], extends the Gaussian Process Latent Variable Model (GPLVM) by structuring the latent space to handle the imbalanced data problem due to data scarcity.
- Chapter 4 is about real applications of scarce data where an expert/oracle is needed. In this approach, the model decides which data would be the most informative to be labeled. By using Gaussian processes, the model's uncertainty is measured. Based on the potential informativeness of the new data point, the oracle/user is asked to annotate the data. This model is based on Yousefi et al. [2018].
- Chapter 5 is concerned with aggregated multi-task learning. A novel multi-task learning model based on Gaussian processes is developed to jointly learn the variables that are aggregated at different input scales. This model represents each task as a linear combination of the realizations of latent processes that are integrated at different scales per task, and usually the task with high-resolution samples has data scarcity issues. This work is based on Yousefi et al. [2019].
- Chapter 6 summarizes the contributions of the thesis and discusses ideas for future work.

1.2 Publications and software

This thesis is built on the work from the following publications:

(i) Fariba Yousefi, Zhenwen Dai, Carl Henrik Ek, Neil Lawrence (2016): Unsupervised Learning with Imbalanced Data via Structure Consolidation Latent Variable Model, *International Conference on Learning Representations (ICLR) workshop track.*

(ii) Fariba Yousefi, Mauricio A Álvarez, Neil Lawrence, Carl Henrik Ek (2018): Active Learning Using Gaussian Processes for Imbalanced Datasets, *Neural Information Processing Systems (NeurIPS) workshop on Bayesian NonParametrics (BNP)*.

(iii) Fariba Yousefi, Michael Thomas Smith, Mauricio A Álvarez (2019): Multi-task Learning for Aggregated Data using Gaussian Processes, *Advances in Neural Information Processing Systems (NeurIPS)*.

The softwares developed for the methods described in Chapters 3, 4 and 5 are publicly available under https://github.com/SheffieldML/GPy/ and https://github.com/frb-yousefi/aggregated-multitask-gp repositories.

Note: Part of my PhD was funded by ControlPoint. I had a chance to work with them. Some part of the work aligned with Chapter 4 for detecting clamps and is not included in this Thesis because of confidentiality reasons.
I also had a chance to collaborate with Zurich hospital on the problem of unsupervised prostate cancer detection. Current cancer diagnoses often involve manual visual inspection of biopsy by pathologists. Such manual inspection is very time consuming and limits the throughput of diagnoses in hospitals. Cancer diagnosis has different stages and in this work we focus on one of the indicators for detecting prostate cancer. In prostate cancer, PTEN (Phosphatase and Tensin homolog) gene is responsible for providing instructions for making protein that is found in almost all tissues in the body. The loss of PTEN gene has a relation with tumor progression. PTEN loss is used as an indicator for diagnosing prostate cancer. In this work, we aimed at automatically detecting PTEN loss from CISH (Chromogenic In Situ Hybridization) images. These images contain black and red stain, the black stain indicates the existence of PTEN gene in individual cells, while the red stain indicates the existence of a reference gene. This work was not compatible with the theme of this thesis and is not included.

Chapter 2

Background

Looking at some real-world applications has shown that data can come in very large quantities and yet the user can be interested in specific parts of the data that are usually under-represented. We are interested in solving data scarcity issues and the imbalanced data problem by taking into account the uncertainty involved. Scarce data problem and imbalanced data problem are very closely related. There are various strategies to cope with scarce data, such as data augmentation, semi-supervised learning, active learning, multi-task learning and transfer learning. We will introduce some of these concepts in the following chapters. In this chapter in Section 2.1, we define the imbalanced data problem, and explain the current state of the art. In section 2.2 we introduce Gaussian processes for regression, classification, and latent variable models.

2.1 Imbalanced data overview

Advanced development in science and technology together with large storage have enabled the growth of data at an exponential rate. This has empowered people working in different fields to learn from data, arriving at results which can have a great impact for day to day life. For example, using maps to find real-time traffic information and calculate the fastest routes to our destinations, finding the most relevant web search results, and fraud detection to distinguish between legitimate and fraudulent bank transactions, are just a few examples that could be mentioned [Elliott, 2019]. However, there can be situations where data is scarce, due to the nature of the problem, or for external reasons.

Imbalanced data has received more attention in recent years [He and Garcia, 2009]. Generally speaking, in a classification case, any dataset containing unequal quantity of samples between classes is considered imbalanced. The primary cause of imbalanced data problem is the lack of data in one of the classes in case of classification. The imbalanced

learning problem is challenging, and one of the major issues is that most algorithms expect a balanced number of samples between classes; in cases of imbalanced data they will ignore the classes which are poorly represented. A broad overview of the state of the art on imbalanced data is provided in the following sections.

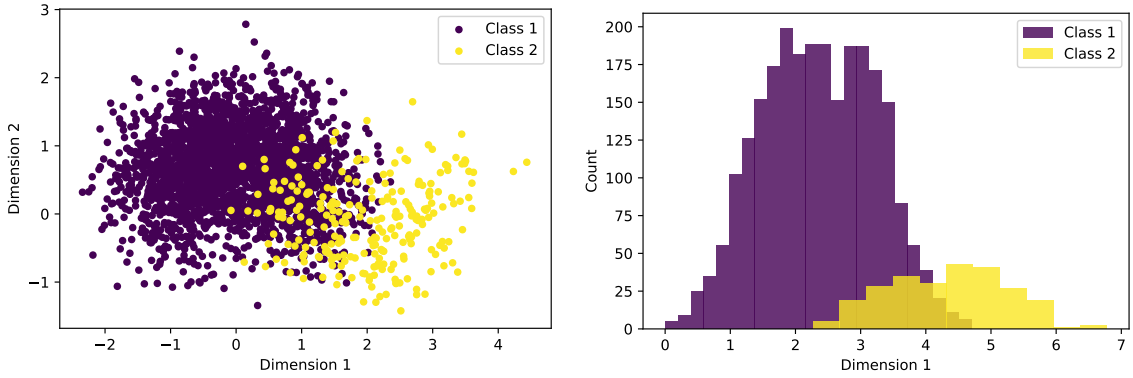
2.1.1 The imbalanced data problem in binary classification tasks

In classification tasks, imbalanced classes can have different imbalanced ratios. These imbalanced ratios can vary and can range from 1:4 up to 1:100 [He and Garcia, 2009]. However, there can also be an extreme imbalance between classes, and the imbalanced ratios can vary from 1:1000 to 1:10000 [He and Garcia, 2009; Krawczyk, 2016].

Imbalanced datasets affect various domains in the real world, such as detection of fraudulent telephone calls [Fawcett and Provost, 1996], fraudulent credit card transactions [Chan and Stolfo, 1998], oil spills in satellite radar images [Kubat et al., 1998], as well as text classification [Zheng et al., 2004] and many others.
A comprehensive overview of imbalanced datasets is available at He and Garcia [2009]. The imbalanced data problem is challenging for most algorithms because the negative class tends to dominate the objective function and the resulting model performs poorly. In the imbalanced domain we seek to have a high accuracy for the minority class without losing much accuracy of the majority class.

An imbalanced dataset can be divided into different categories, such as intrinsic or extrinsic. In the *intrinsic* case, the imbalance is the result of the nature of the data itself. In the *extrinsic* case, the imbalance is due to external reasons such as time and storage constraints, and *not* because of the nature of the data itself. For example, interruptions during a period of capturing continuous data from a balanced set might result in an imbalanced set [He and Garcia, 2009].

In this thesis, we are focusing on intrinsic imbalanced data problem and will address imbalanced data problem issues in Chapters 3 and 4. We present an example of a synthetically-generated imbalanced dataset in Figure 2.1 using the Scikit-learn imbalanced toolkit [Lemaître et al., 2017]. In Figure 2.1 (a) two classes illustrate the imbalanced data problem. The purple data refer to the majority class and the yellow data refer to the minority class. This dataset has an imbalanced ratio of 1:10, meaning that for every ten samples in the majority class there is only one sample in the minority class. In Figure 2.1 (b), a histogram representation of the data is shown. We can see the data distribution and hence the overlapping samples between two classes. In the real world, in most cases data between classes do overlap, and this may create further complexities to borderline cases for the learning process.



(a) Scatter plot of the data

(b) Histogram representation of the data

Fig. 2.1 Illustration of synthetic 2-dimensional binary, imbalanced data. The imbalanced ratio is 1:10.

2.1.2 Methods for addressing imbalanced learning

There are two approaches to addressing imbalanced datasets: data level methods and algorithm level methods [He and Ma, 2013]. The aim of data level methods is to balance the datasets by removing or adding data, and changing the class distribution so that standard classifiers can work. Algorithm level methods aim to apply a classifier that can handle the imbalance algorithmically without changing the data distribution. Data level and algorithm level methods can be combined.

2.1.2.1 Data level methods

Having a balanced dataset can produce better classification results, which motivates the use of sampling methods to balance the dataset. There are different methods for sampling such as random oversampling [Estabrooks et al., 2004; He and Ma, 2013] and random undersampling [Drummond et al., 2003; He and Ma, 2013].

Random *undersampling* randomly ignores samples from the majority class. In contrast, random *oversampling* inflates the minority class by randomly duplicating a certain quantity of samples from it [Drummond et al., 2003; Estabrooks et al., 2004; He and Ma, 2013]. These two methods are not equivalent and each has its disadvantages. For example, when undersampling we might miss some important information relating to the majority class, while in oversampling we might see overfitting when duplicates of the same data are created.

Other data level techniques include synthetic data generation approaches. Synthetic Minority Oversampling TEchnique (SMOTE) is one famous and advanced sampling method for creating synthetic datasets [Chawla et al., 2002]. In applying this technique, the dataset

is augmented and synthetic data is generated by using a sample and its k-nearest neighbor. However, in SMOTE the nearest neighbors are selected randomly. There are also variations of SMOTE that can generate data in more effective ways to overcome this limitation. Adaptive sampling techniques such as borderline-SMOTE [Han et al., 2005] generates data using samples from the minority class that are closer to its borderline with the majority class. However, by augmenting the data in this way we are increasing the size of the dataset, as well as computational complexity.

There are other techniques, such as hybrid methods, that combine both under- and over-sampling methods [Batista et al., 2004]. However, these combinations can be very complicated, since instead of optimizing a single under (over) sampling method, one has to optimize a combination of both. Data level methods have proven to be less effective than algorithm level classifiers [Cieslak et al., 2012]. We will have an overview of algorithm level classifiers in the next section.

2.1.2.2 Algorithm level methods

Commonly used approaches for algorithm level methods are cost sensitive methods. Cost sensitive methods aim to calculate the 'cost' of the misclassified samples [Elkan, 2001; He and Ma, 2013; Ting, 2002]. These methods have different weight matrices or costs for misclassifying samples of a particular class, and involve cost sensitive decision trees [Elkan, 2001; He and Ma, 2013; Maloof, 2003] and cost sensitive neural networks [He and Ma, 2013; Kukar et al., 1998; Zhou and Liu, 2006].

There are other methods such as one class classification [Chawla et al., 2004]; kernel based methods such as cost sensitive Support Vector Machines (SVM) [Fumera and Roli, 2002; Platt, 1998], and active learning [Abe, 2003; Ertekin et al., 2007a,b; Provost, 2000]. We will discuss active learning in more detail in Chapter 4. Ensemble methods are also popular since multiple weak classifiers can outperform a single strong classifier's performance [Domingos, 1999; Fan et al., 1999; Sun et al., 2007]. Algorithm level methods can also be combined with data level methods to improve performance [He and Garcia, 2009].

2.1.3 Assessment metrics

Assessment metrics help to evaluate how well an algorithm works for a given dataset. However, for imbalanced classes, assessment metrics need to be more carefully considered than in the balanced case, where it is often sufficient to consider accuracy alone. Accuracy is the sum of correctly detected samples in different classes divided by the total number of data, and is defined in Equation (2.1). Testing for accuracy is one of the most frequently used

		True class	
		Positive	Negative
Predicted class	Positive	TP	FP
	Negative	FN	TN

Table 2.1 Confusion matrix for binary classification problem.

techniques in classification. However, for imbalanced data it can be deceiving in situations where our class of interest is the minority class.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total number of data}}, \tag{2.1}$$

where True Positive (TP) is the number of positive examples that are correctly classified as positive. True Negative (TN) is the number of negative examples that are classified correctly as negative. For example, if we have 90% of the majority class and only 10% of the minority class, the naive approach would be to classify the entire data as belonging to the majority class, achieving a high accuracy of 90%. This might seems a great result for the majority class. However, this does not provide any information regarding the minority class — this is especially a problem when this is the more interesting and relevant class to the user and none of it is detected [He and Garcia, 2009], with everything misclassified as the majority class. So alternative techniques are needed for measuring the performance of the minority class.

For binary classification, a confusion matrix represents the result of correctly and incorrectly identifying each class. In Table 2.1, False Positives (FP) are also called type *I* error and False Negatives (FN) are also called type *II* error. The columns are true class and the rows are predicted class. In a confusion matrix, FP is the number of negative examples that are classified incorrectly as belonging to the positive class and FN is the number of positive examples that are classified incorrectly as belonging to the negative class.

In the case of imbalanced data sets, metrics such as precision, recall, F-measure, Receiver Operating Characteristic (ROC), and area under the ROC curve (AUC) are considered. Precision is defined as the number of true positives divided by the number of true positives plus false positives. Recall, which is also called True Positive Rate (TPR) or Sensitivity, is defined as the number of true positives divided by the number of true positives plus false negatives. Precision measures how many predicted positive data are actually positive, whereas recall calculates how many positive data are correctly predicted as positive. Precision and recall have an inverse relationship. Improving recall without losing much precision is the goal of imbalanced learning. F-measure combines precision and recall, and the output is a single measure that reflects the result of classification in the presence of the minority class

[Chawla, 2009; He and Garcia, 2009; He and Ma, 2013; Provost and Fawcett, 2001].
The expressions for *Precision*, *Recall*, *False Positive Rate* (FPR) and *F-measure* are as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \text{ Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \text{ FPR} = \frac{\text{FP}}{\text{TN}+\text{FP}}$$

$$F_\beta = (1+\beta^2) \times \frac{\text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}},$$

where β corresponds to the relative importance of *precision* vs *recall*; it is usually set to one [Chawla, 2009].

2.2 Gaussian process overview

In this section we will provide an overview of Gaussian Process (GP) models that will be used in the following chapters in combination with the imbalanced data. A Gaussian process can be defined as a distribution over functions. Formally it is defined as [Rasmussen and Williams, 2006a]:

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

There are numerous problems such as regression, classification, and dimensionality reduction that can be solved using GPs. In a GP there exists a mapping function called f which is unknown (latent), and the aim is to learn the distribution over it. Using Bayes' rule the posterior can be calculated that finds the function which best fits the data. Gaussian processes are non-parametric, which means they do not rely on a fixed number of parameters.

In the following sections, we will have a more formal introduction to GPs. In these sections, latent outputs are $\mathbf{f} = \{f_n\}_{n=1}^N$ and $f_n \in \{-\infty, +\infty\}$, where N is the size of dataset. The targets in the regression task are real and denoted as $\mathbf{y} = \{y_n\}_{n=1}^N$ and $y_n \in \mathcal{R}$. Targets in classification are binary, and denoted as $\mathbf{y} = \{y_n\}_{n=1}^N$ and $y_n \in \{0,1\}$. The training and test inputs for both regression and classification tasks are $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N, \mathbf{x}_n \in \mathcal{R}^p$ and $\mathbf{X}_* = \{\mathbf{x}_n\}_{n=1}^{N_*}, \mathbf{x}_n* \in \mathcal{R}^p$ respectively, and p is the dimensionality of the inputs. In Section 2.2.1, we define Gaussian process regression; in Section 2.2.3, we introduce Gaussian process classification; in Section 2.2.5.2, we review Gaussian process latent variable models (GPLVM), and in Section 2.2.6, we introduce sparse GP.

2.2.1 Gaussian process regression

Gaussian processes are popular in machine learning partially because they have properties that make computations tractable and easy. For example, the conjugate property means that we can calculate the posterior tractably using the Bayesian framework. A Gaussian process is defined with its mean function, $\mu(\boldsymbol{x})$ and covariance function, $k(\boldsymbol{x},\boldsymbol{x'})$, with $\mathbf{x} \in \mathcal{R}^p$ and $\mathbf{x}' \in \mathcal{R}^p$,

$$\begin{aligned} f &\sim \mathcal{GP}(\mu(\boldsymbol{x}),\, k(\boldsymbol{x},\boldsymbol{x'})) \\ \mu(\boldsymbol{x}) &= E[f(\boldsymbol{x})] \\ k(\boldsymbol{x},\boldsymbol{x'}) &= E[(f(\boldsymbol{x})-\mu(\boldsymbol{x}))(f(\boldsymbol{x'})-\mu(\boldsymbol{x'}))]. \end{aligned} \tag{2.2}$$

While modeling, we often select the GP's mean in Equation (2.2) as $\mu(\boldsymbol{x}) = 0$. The covariance function can be selected from a variety of functions that hold a series of specific properties, for example being positive semi-definite. The covariance function defines the similarity or relation between data points and is evaluated on a finite set of inputs. More information regarding covariance functions is given in the Section 2.2.2.

In the equations below, $\mathbf{X} \in \mathcal{R}^{N\times p}$, $\mathbf{y} \in \mathcal{R}^{N\times 1}$, $\mathbf{f} \in \mathcal{R}^{N\times 1}$ and $\mathbf{X}* \in \mathcal{R}^{N_*\times p}$. N and N_* are the respective numbers of the training and test points. f is the mapping function that maps input data to observed values. In real life applications, it is typically assumed that only noisy measurements, $\mathbf{y} = [y_1, y_2, \ldots, y_N]^\top$, are observed, rather than the true observations, $\mathbf{f} = f(\mathbf{x}) = [f(x_1), f(x_2), \ldots, f(x_N)]^\top$. The joint distribution of the training noisy outputs, $\mathbf{y}$, which are assumed to be independent and identically distributed (i.i.d.) corruptions of latent function $\mathbf{f}$ by a zero mean Gaussian noise and σ^2 variance, is defined as:

$$\mathbf{y} = \mathbf{f} + \boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}). \tag{2.3}$$

We assume this noise follows i.i.d Gaussian distribution. The GP prior is defined as:

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{ff}) \tag{2.4}$$

$$= (2\pi)^{-\frac{N}{2}} |\mathbf{K}_{ff}|^{-\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{f}^\top \mathbf{K}_{ff}^{-1}\mathbf{f}). \tag{2.5}$$

$\mathbf{K}_{ff} \in \mathbb{R}^{N\times N}$ is a covariance matrix, evaluated between all pairs of training data. GP prior encodes information about how probable each of the infinity many possible functions

are, before observing any data. The log of the prior is:

$$\log p(\mathbf{f}|\mathbf{X}) = -\frac{1}{2}\mathbf{f}^\top \mathbf{K}_{ff}^{-1}\mathbf{f} - \frac{1}{2}\log|\mathbf{K}_{ff}| - \frac{N}{2}\log 2\pi. \tag{2.6}$$

So, keeping in mind that $\mathbf{y}$ is the noisy version of $\mathbf{f}$, the likelihood function — that is, the probability of the observations given the parameters — is as follows:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I}). \tag{2.7}$$

Marginal likelihood is defined by integrating out the latent function $\mathbf{f}$ and the log marginal likelihood is correspondingly defined as follows:

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

where θ denotes hyper-parameters, for example parameters of the kernel function. The first term in the Equation (2.8) that involves targets $\mathbf{y}$ promotes a good fit to the data; the second term, containing the determinant, is independent of the targets and penalizes complex models. The third term is a normalization constant. The posterior distribution of $\mathbf{f}$ is defined as:

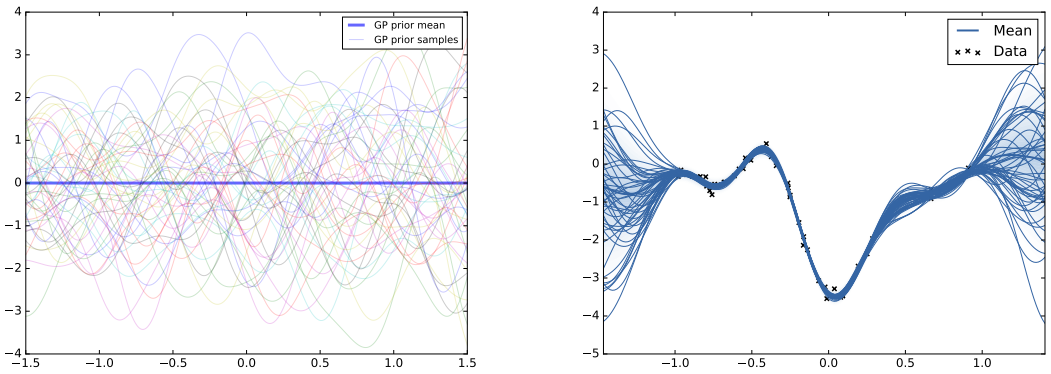
$$\begin{aligned}p(\mathbf{f}|\mathbf{X},\mathbf{y}) &= \frac{p(\mathbf{y}|\mathbf{f})\ p(\mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})} \\ &\propto \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I})\ \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{ff}).\end{aligned} \tag{2.9}$$

Calculating the marginal likelihood in most cases is not tractable. In GP regression since our likelihood and prior are both Gaussian, given the conjugacy property, calculating the posterior is tractable. Samples from the GP prior and the GP posterior are illustrated in Figure 2.2.

The joint distribution of the training noise-free outputs (latent function) $\mathbf{f}$ and the test noise-free outputs (latent function) $\mathbf{f}_*$, by assuming GP prior has zero-mean, is:

$$p\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix}\middle|\mathbf{X},\mathbf{X}_*\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{f}_*\end{bmatrix}\middle|\mathbf{0}, \begin{bmatrix}\mathbf{K}_{ff} & \mathbf{K}_{f*}\\ \mathbf{K}_{*f} & \mathbf{K}_{**}\end{bmatrix}\right). \tag{2.10}$$

Here, $\mathbf{K}_{**}$ is the covariance between the test inputs, $\mathbf{X}_*$, $\mathbf{K}_{*f}$ is the cross covariance between the test and train inputs, and $\mathbf{K}_{f*}$ is the covariance between the train and test inputs, which is the transpose of $\mathbf{K}_{*f}$.



(a) (b)

Fig. 2.2 (a) Samples from the GP prior. (b) Samples from the GP posterior after observing some data points.

Conditioning the joint Gaussian prior distribution to the latent function gives [Rasmussen and Williams, 2006a]:

$$p(\mathbf{f}_*|\mathbf{f},\mathbf{X},\mathbf{X}_*) = \mathcal{N}(\mathbf{f}_*|\mathbf{K}_{*f}\mathbf{K}_{ff}^{-1}\mathbf{f},\ \mathbf{K}_{**}-\mathbf{K}_{*f}\mathbf{K}_{ff}^{-1}\mathbf{K}_{f_*}). \tag{2.11}$$

The joint distribution of the training noisy outputs, $\mathbf{y}$, and the test noise-free outputs (latent function) $\mathbf{f}_*$, assuming GP prior has zero-mean, is:

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix}\middle|\mathbf{X},\mathbf{X}_*\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix}\middle|\mathbf{0},\begin{bmatrix}\mathbf{K}_{ff}+\sigma^2\mathbf{I} & \mathbf{K}_{f*}\\ \mathbf{K}_{*f} & \mathbf{K}_{**}\end{bmatrix}\right). \tag{2.12}$$

The prediction using noisy observations is:

$$\begin{aligned} p(\mathbf{f}_*|\mathbf{y},\mathbf{X},\mathbf{X}_*) &= \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*,\mathbf{K}_*) \\ \boldsymbol{\mu}_* &= \mathbf{K}_{*f}(\mathbf{K}_{ff}+\sigma^2\mathbf{I})^{-1}\mathbf{y} \\ \mathbf{K}_* &= \mathbf{K}_{**}-\mathbf{K}_{*f}(\mathbf{K}_{ff}+\sigma^2\mathbf{I})^{-1}\mathbf{K}_{f_*}, \end{aligned} \tag{2.13}$$

where $\boldsymbol{\mu}_*$ is the mean and $\mathbf{K}_*$ is the covariance of the predictive distribution. The graphical model representation of GP regression is illustrated in Figure 2.3. Latent functions f are fully connected to each other since they are sampled from the same GP. Labels y_n are conditionally independent from the other nodes given latent function f_n.

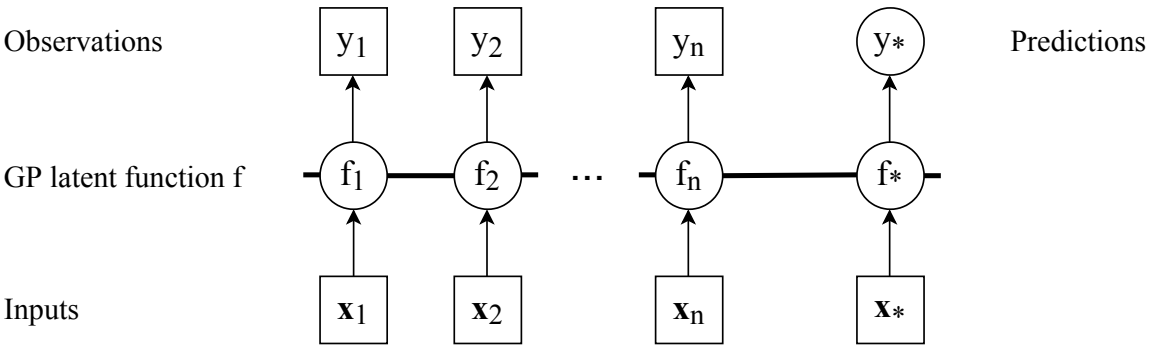


Fig. 2.3 Graphical model representation for Gaussian process regression. Squares represent observed data and circles refer to unknown (latent) functions. Horizontal bars represent fully connected nodes [Rasmussen and Williams, 2006a].

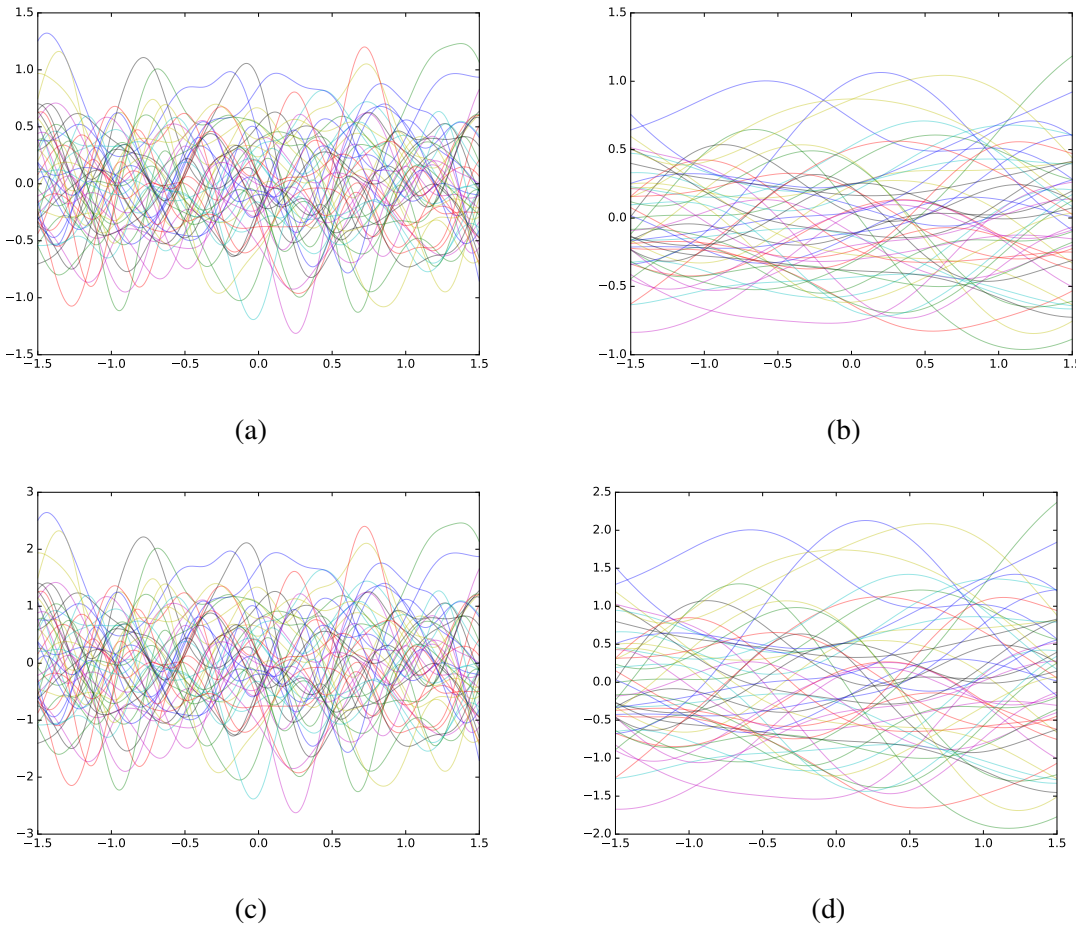
2.2.2 Covariance functions

A Covariance function (also called a kernel) plays an important role in Gaussian process modeling. It encapsulates the prior information we may have about the nature of the data. A kernel should be a *positive semi-definite* function over all the possible input pairs $(\mathbf{x}, \mathbf{x}')$. This means the eigenvalues of the covariance matrix should be non-negative. Kernels can be seen as a similarity measurement. The inputs that are closer to each other in the feature space are expected to have high correlation within the covariance function, and they are believed to have similar behavior in the output space. In contrast, those that are far from each other have very little correlation or no correlation at all.

The covariance function maps pair of inputs $\mathbf{x}$ and $\mathbf{x}'$ to the real value $k(\mathbf{x}, \mathbf{x}')$ function. There can be many different kernels, such as linear, exponentiated quadratic (EQ) — also known as Radial Basis Function (RBF) — polynomial, periodic, and Matérn class of kernels. Each of those kernels can have different attributes. For example, RBF has infinitely many derivatives and is appropriate for modeling smooth functions. Each kernel has some parameters which are called *hyper-parameters*. For instance, the RBF kernel is a popular choice within the literature and it has two hyper-parameters: length-scale (ℓ) and variance (σ^2). The length-scale defines how fast the output changes with changes in the input space. An RBF kernel is a stationary kernel: that is, it can be expressed as a function of the difference between its inputs.[1] As mentioned in the Section 2.2, $\mathbf{x}, \mathbf{x}' \in \mathcal{R}^p$ and p is the dimensionality of the inputs $\mathbf{x}$ and $\mathbf{x}'$.

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \sigma_{rbf}^2 \exp\Big(-\frac{1}{2\ell^2}\sum_{j=1}^{P}(x_j - x_j')^2\Big). \tag{2.14}$$

[1] transcendent-ai-labs.github.io/DynaML/core/core_kernel_stat, (last accessed 20.07.2020).



(a) (b)

(c) (d)

Fig. 2.4 (a) RBF kernel with $\sigma^2_{rbf} = 0.2$ and $\ell = 0.2$. (b) RBF kernel with $\sigma^2_{rbf} = 0.2$ and $\ell = 0.8$. (c) RBF kernel with $\sigma^2_{rbf} = 0.8$ and $\ell = 0.2$. (d) RBF kernel with $\sigma^2_{rbf} = 0.8$ and $\ell = 0.8$.

An illustration of the correlation of covariance function with its hyper-parameters is represented in Figure 2.4. In this figure, the variance is represented as σ^2_{rbf}. Figure 2.4 (a) and (c) have the same length-scale but different variance. Figure 2.4 (a) and (b) have the same variance but different length-scale. As can be seen in these figures, with a smaller length-scale the function changes much faster, while with a larger length-scale, the function becomes flat and it seems like outputs are highly correlated. Kernels can also be combined in many different ways, for example by addition or multiplication of the kernels. Combining the kernels may help to solve more sophisticated problems and to model complicated datasets. More information regarding different kernels and their combinations can be found at Duvenaud [2014]. In this thesis, we will use the RBF kernel as a general case, unless otherwise stated.

2.2.3 Gaussian process classification

As discussed in Section 2.2.1, in regression problems the target values are continuous. If our target values are discrete, it is called a classification problem. Classification problems can be categorized as either binary or multi-class. An example of a multi-class classification problem is hand-written digit recognition, where each of the hand-written digits needs to be classified as belonging to one of the classes $0, 1, ..., 9$. An example of a binary classification could be the recognition of "dog" versus "cat" in images. For example, the classes will be $+1$ for the "dog" class and 0 for the "cat" class. In Gaussian process classification, unlike Gaussian process regression, the likelihood function is no longer Gaussian, and the conjugate property does not apply, so the posterior cannot be calculated analytically, although it can be approximated. To achieve this, there are a variety of approximation techniques such as: Laplace approximation [Barber and Williams, 1997], Expectation Propagation (EP)[Minka, 2001], variational approximations [Gibbs and MacKay, 2000; Jaakkola and Jordan, 1996; Opper and Archambeau, 2009; Seeger, 2000] and Markov Chain Monte Carlo (MCMC) sampling [Neal, 1997]. Approximation techniques differ in how the density of the non-Gaussian posterior is approximated. A comparison of different methods with binary likelihood is available in Kuss and Rasmussen [2005]; Nickisch and Rasmussen [2008]. These approximation techniques with their respective likelihood functions pose trade-offs between computational complexity, simplicity and accuracy [Saul, 2016].

In binary classification the latent function f takes the values in the interval $(-\infty, +\infty)$, but since the target values are binary $\{0, 1\}$, we need a so-called 'squashing' function to shrink the range of latent values to $(0, 1)$. Two commonly used squashing functions are the logistic from the sigmoid family, and the cumulative Gaussian (probit) [Rasmussen and Williams,

2006a]. The squash function we use is defined as a logistic function, and the Bernoulli likelihood conditions the data on the squashed function values.

The prior is the same as explained in the regression case:

$$p(\mathbf{f}|\mathbf{X},\theta) = \mathcal{N}(\mathbf{f}|\mathbf{0},\ \mathbf{K}_{ff}). \tag{2.15}$$

The logistic function has the symmetry property $1 - f(x) = f(-x)$ and is defined as:

$$\sigma_{logit}(f(\mathbf{x})) = \frac{1}{1+\exp(-f(\mathbf{x}))}. \tag{2.16}$$

By using Equation (2.16), the class membership function for each case is defined as:

$$p(y=1|\mathbf{x}) = \sigma(f(\mathbf{x})) \tag{2.17}$$

$$p(y=-1|\mathbf{x}) = 1 - p(y=1|\mathbf{x}) = 1 - \sigma(f(\mathbf{x})) = \sigma(-(f(\mathbf{x}))) \tag{2.18}$$

Combining the two cases above, based on the symmetry property, will result in:

$$p(y|\mathbf{x}) = \sigma(y\, f(\mathbf{x})) = \sigma(y\, f). \tag{2.19}$$

The likelihood, which is the probability of the observations given the parameters, is factorized over the cases in the training set because of the conditional independence assumption.

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N} p(y_n|f_n) = \prod_{n=1}^{N} \sigma(y_n\, f_n). \tag{2.20}$$

Using Bayes' rule, the posterior distribution is computed as:

$$p(\mathbf{f}|\mathbf{X},\mathbf{y},\theta) = \frac{p(\mathbf{f}|\mathbf{X},\theta)\ \prod_{n=1}^{N} p(y_n|f_n)}{\int p(\mathbf{f}|\mathbf{X},\theta)\ \prod_{n=1}^{N} p(y_n|f_n)\, df} = \frac{\mathcal{N}(\mathbf{f}|\mathbf{0},\ \mathbf{K})}{p(\mathbf{y}|\mathbf{X},\theta)} \prod_{n=1}^{N} \sigma(y_n\, f_n). \tag{2.21}$$

The non-Gaussian likelihood makes this calculation intractable and approximation is needed to be able to calculate the posterior. The predictions at $\mathbf{x}_*$ are:

$$p(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*,\theta) = \int p(f_*|\mathbf{f},\mathbf{X},\mathbf{x}_*,\theta)\ p(\mathbf{f}|\mathbf{X},\mathbf{y},\theta)\, d\mathbf{f}, \tag{2.22}$$

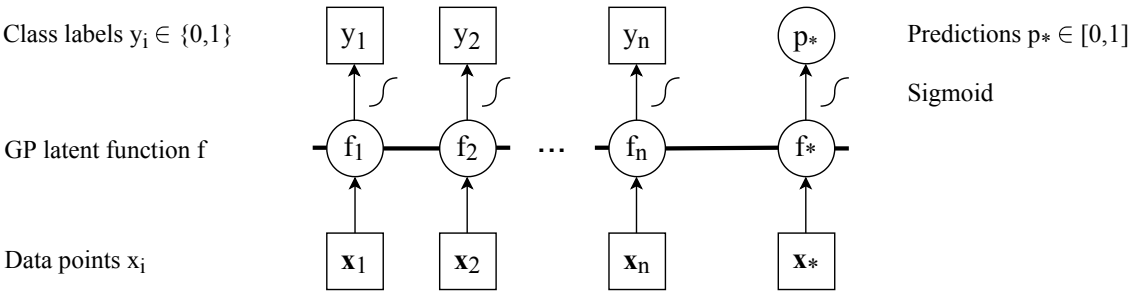


Fig. 2.5 Binary Gaussian process classification graphical model. Squares represent observed data and circles refer to unknown (latent) functions. The labels y_i are binary variables and predictions p_* are probabilities in the range of the interval $[0,1]$ [Nickisch and Rasmussen, 2008].

where the prediction is factorized into the product of the conditional prior and posterior. The predictive class probability is:

$$p(y_* = 1|\mathbf{X},\mathbf{y},\mathbf{x}_*,\theta) = \int p(f_*|\mathbf{X},\mathbf{y},\mathbf{x}_*,\theta)\ \sigma(f_*)\ df_*. \tag{2.23}$$

The integral in Equation (2.21) is intractable so we will look into approximation methods. Figure 2.5 represents a graphical model for GP classification. As presented in the figure, latent functions f are fully connected to each other since they are sampled from the same GP. Labels y_n and latent functions f_n are connected through the sigmoid function. Labels y_n are conditionally independent from the other nodes, given latent function f_n.

For further details on GP classification please refer to Bishop et al. [2006]; Nickisch and Rasmussen [2008]; Rasmussen and Williams [2006a].

2.2.4 Approximations

As mentioned in Section 2.2.3, an approximation is needed when the calculation of the posterior is no longer analytically tractable, for example due to the non-Gaussian likelihood. There are various approximation techniques, and they vary with respect to how they approximate a non-Gaussian distribution with a Gaussian distribution. In the following sections, we will describe Laplace and variational approximations. Laplace approximation will be used in chapter 4 and variational approximation will be used in Chapter 3 and Chapter 5.

2.2.4.1 Laplace approximation

Due to the non-Gaussian likelihood in classification case, the posterior is no longer tractable and should be approximated. The likelihood function will therefore be squashed through a logistic function.

The latent functions (f) should initially be in the range $(-\infty, +\infty)$. However, since the target values are now binary (for example $\{0,1\}$), the squash function will shrink the range of the latent values to $(0,1)$. We call the new function $\pi_n \in [0,1]$, and π_n is the same as $\sigma_{logit}(f(\mathbf{x}))$ in Equation (2.16):

$$\pi_n = \frac{1}{1+\exp(-f(\mathbf{x}_n))}. \tag{2.24}$$

The probability for the Bernoulli likelihood for binary observations $y_n \in \{0,1\}$ is defined as:

$$p(y_n|\pi_n) = \Pi_{n=1}^{N} \pi_n^{y_n}(1-\pi_n)^{1-y_n}. \tag{2.25}$$

Laplace approximation is based on Laplace's method, which fits the mean at the peak of the posterior and matches the curvature there. In Laplace approximation, we need to calculate the mean and covariance for the posterior. To approximate the posterior, we first need to find the modal point of the true log posterior using Newton's method and then use the second-order Taylor expansion around the modal value to calculate the curvature. We obtain a Gaussian approximation with mean equal to the modal point and curvature which is the negative inverse Hessian of log $p(\mathbf{f}|\mathbf{X},\mathbf{y})$ [Rasmussen and Williams, 2006a; Saul, 2016].
The Gaussian process prior is defined as:

$$p(\mathbf{f}|\mathbf{X}) = (2\pi)^{-\frac{N}{2}} \left|\mathbf{K}_{ff}\right|^{-\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{f}^\top \mathbf{K}_{ff}^{-1}\mathbf{f}). \tag{2.26}$$

The posterior is defined as:

$$p(\mathbf{f}|\mathbf{X},\mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}). \tag{2.27}$$

Gaussian process posterior is approximated with mean ($\hat{\mathbf{f}}$) and covariance ($\mathcal{A}$) and is defined as:

$$\begin{aligned} p(\mathbf{f}|\mathbf{X},\mathbf{y}) \approx q(\mathbf{f}|\mathbf{X},\mathbf{y}) &= N(\mathbf{f}|\hat{\mathbf{f}},\mathcal{A}^{-1}) \\ &\propto \exp(-\frac{1}{2}((\mathbf{f}-\hat{\mathbf{f}})^\top \mathcal{A}(\mathbf{f}-\hat{\mathbf{f}}))), \end{aligned} \tag{2.28}$$

where q refers to approximation, $\hat{\mathbf{f}} = \text{argmax}_{\mathbf{f}}\, p(\mathbf{f}|\mathbf{X},\mathbf{y})$, and $\mathcal{A}$ is equal to the Hessian of the negative log posterior at the point $\hat{\mathbf{f}}$. In other words, $\mathcal{A} = -\frac{\delta^2 \log(\mathbf{f}|\mathbf{X},\mathbf{y})}{\delta \mathbf{f}^2}$ [Rasmussen and Williams, 2006a]. For calculating $\hat{\mathbf{f}}$ and $\mathcal{A}$ the log posterior is defined as:

$$\begin{aligned} \log(\mathbf{f}|\mathbf{X},\mathbf{y}) &\propto \log(\mathbf{y}|\mathbf{f},\mathbf{X}) + \log(\mathbf{f}|\mathbf{X}) \\ &= \log(\mathbf{y}|\mathbf{f},\mathbf{X}) - \frac{N}{2}\log(2\pi) - \frac{1}{2}\log(\mathbf{K}_{ff}) - \frac{1}{2}\mathbf{f}^\top \mathbf{K}_{ff}^{-1}\mathbf{f}. \end{aligned} \tag{2.29}$$

If we take the first and the second derivative of Equation (2.29) with respect to $\mathbf{f}$ we get:

$$\frac{\delta \log(\mathbf{f}|\mathbf{y},\mathbf{X})}{\delta \mathbf{f}} = \frac{\delta \log(\mathbf{y}|\mathbf{f},\mathbf{X})}{\delta \mathbf{f}} - \mathbf{K}_{ff}^{-1}\mathbf{f}, \tag{2.30}$$

$$\frac{\delta^2 \log(\mathbf{f}|\mathbf{y},\mathbf{X})}{\delta \mathbf{f}^2} = \frac{\delta^2 \log(\mathbf{y}|\mathbf{f},\mathbf{X})}{\delta \mathbf{f}^2} - \mathbf{K}_{ff}^{-1} \tag{2.31}$$

$$= -\mathcal{W} - \mathbf{K}_{ff}^{-1}, \tag{2.32}$$

$\mathcal{W}$ is diagonal matrix because of the independence assumption over the cases. To find the maximum or the modal point,

$$\frac{\delta log(\mathbf{f}|\mathbf{y},\mathbf{X})}{\delta \mathbf{f}} = 0, \tag{2.33}$$

and the equation for $\hat{\mathbf{f}}$ is:

$$\hat{\mathbf{f}} = \mathbf{K}_{ff}\left(\frac{\delta \log(\mathbf{y}|\hat{\mathbf{f}},\mathbf{X})}{\delta \hat{\mathbf{f}}}\right). \tag{2.34}$$

Since $\nabla \log(\mathbf{y}|\hat{\mathbf{f}})$ is a non-linear function of $\hat{\mathbf{f}}$, we cannot solve it directly — we need to solve it using Newton's method, which is iterative; more details are provided in Appendix B.3. ∇ is a derivative operator. By calculating the maximum $\hat{\mathbf{f}}$, the Laplace approximation can be calculated with mean equal to $\hat{\mathbf{f}}$ and covariance matrix equal to $\mathcal{A}$. So the mean and the covariance of the approximate posterior is:

$$q(\mathbf{f}|\mathbf{X},\mathbf{y}) = \mathcal{N}(\hat{\mathbf{f}}, (\mathcal{W} + \mathbf{K}_{ff}^{-1})^{-1}). \tag{2.35}$$

2.2.4.2 Variational approximation

Variational inference aims to approximate difficult to compute probability densities, and is used to approximate posterior density for Bayesian models [Blei et al., 2017]. Variational inference is much faster than MCMC [Gelfand and Smith, 1990; Hastings, 1970; Neal,

1997] and can scale up to larger datasets. Unlike MCMC, which uses sampling, variational inference uses optimization to minimize the KL divergence [Kullback and Leibler, 1951] between the approximate posterior and the exact posterior. So variational inference turns the inference problem into an optimization problem, and approximates the posterior with the result of the optimization parameters [Blei et al., 2017]. The KL-divergence between the two distributions $q(\mathbf{f})$ and $p(\mathbf{f}|\mathbf{y})$, assuming f is latent and y is observed, is defined as:

$$\mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f}|\mathbf{y})) = \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f}|\mathbf{y})} d\mathbf{f} \tag{2.36}$$

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})\ p(\mathbf{f})}{p(\mathbf{y})}$$

$$\begin{aligned}\mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f}|\mathbf{y})) &= \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{y},\mathbf{f})} d\mathbf{f} + \log p(\mathbf{y}) \\ &= \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} - \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log p(\mathbf{y}) \\ &= \mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f})) - \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log p(\mathbf{y}).\end{aligned} \tag{2.37}$$

By re-arranging the equation above, we obtain the marginal log likelihood:

$$\log p(\mathbf{y}) = \mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f}|\mathbf{y})) - \mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f})) + \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}. \tag{2.38}$$

We assume $q(\mathbf{f})$ has a Gaussian distribution. The KL-divergence is asymmetric, which means $\mathrm{KL}(q\|p) \neq \mathrm{KL}(p\|q)$ and $\mathrm{KL}(q\|p) \geq 0$ [Blei et al., 2017]. In practice, since the true posterior is not tractable, we cannot solve the KL-divergence above — $\mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f}|\mathbf{y}))$. So we optimize an alternative objective called Evidence Lower Bound (ELBO). By removing the above KL, we get the bound that is larger or equal to the rest of the equation. Maximizing the ELBO is equivalent to minimizing the KL-divergence. Using the fact that KL-divergence is non-negative, the ELBO ($\mathcal{L}$) becomes:

$$\log p(\mathbf{y}) \geq \int q(\mathbf{f}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} - \mathrm{KL}(q(\mathbf{f})\|p(\mathbf{f})) = \mathcal{L}. \tag{2.39}$$

The first term in the equation above is the expected value of the likelihood under $q(\mathbf{f})$, and this equation encourages the parameters that best describe the latent function f that explain the observed data y. The second term encourages densities that are close to the prior, so the bound keeps the balance between the likelihood and the prior [Blei et al., 2017]. It is also possible to achieve the same bound using Jensen's inequality [Jordan et al., 1999] — this is explained in more detail in Section 2.2.6.1.

2.2.5 Latent variable models (LVM)

In this section, we discuss latent variable models. In latent variable models we could imagine inputs as actually unobserved, and that only outputs are observed. Latent variable models assume a mapping exists between a set of latent variables and a set of observed variables. There are various models that make this assumption, such as Principle Component Analysis (PCA), Probabilistic Principle Component Analysis (PPCA), Gaussian Process Latent Variable Model (GPLVM), and Factor analysis (FA). GPLVM is an instance of LVMs for unsupervised learning in general. Different latent variable methods do the mapping differently, and we will explain a few of them in the following sections. PCA [Jolliffe, 1986], is a well-known method of dimensionality reduction. PCA finds a lower dimensional representation of the data in the direction where the variance is maximized, and projects the data linearly in the direction of the eigenvectors corresponding to the highest eigenvalues. The probabilistic version of this method is called PPCA [Tipping and Bishop, 1999b]. PPCA assumes the relationship between observed and latent variables with the added noise,

$$\mathbf{y}_n = \mathbf{W}\mathbf{x}_n + \boldsymbol{\varepsilon}_n,$$

where $\mathbf{W} \in \mathcal{R}^{p\times q}$ is a matrix that maps between the latent space $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^\top$ where $\mathbf{X} \in \mathcal{R}^{N\times q}$ and the output space $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n]^\top$, where $\mathbf{Y} \in \mathcal{R}^{N\times p}$, linearly, and $\mathbf{x} \in \mathcal{R}^q$ and $\mathbf{y} \in \mathcal{R}^p$. The dimensionalities for $\mathbf{X}$ and $\mathbf{Y}$ are different from the ones mentioned in the regression and classification cases in Sections 2.2.1 and 2.2.3.
In the regression and classification cases input $\mathbf{X}$ and output $\mathbf{y}$ are both observed, however in the latent variable models $\mathbf{Y}$ is observed and is our input, and $\mathbf{X}$ is our latent variable that is not observed. The noise, $\boldsymbol{\varepsilon}_n \in \mathcal{R}^{p\times 1}$, is taken from an independent Gaussian with zero mean and covariance $\sigma^2\mathbf{I}$,

$$\boldsymbol{\varepsilon}_n \sim \mathcal{N}(\boldsymbol{\varepsilon}_n|\mathbf{0}, \sigma^2\mathbf{I}).$$

The likelihood on which we condition the data on the latent point, parameters, and other hyper-parameters is as follows,

$$p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{W}, \boldsymbol{\varepsilon}_n) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n,\ \sigma^2\mathbf{I}).$$

The marginal likelihood is defined by specifying prior distribution over latent variables and integrating them out. For PPCA, we assume our latent variables are i.i.d. with zero mean and

unit variance,

$$p(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{0},\ \mathbf{I}).$$

The marginal likelihood can be calculated analytically and is defined as below,

$$p(\mathbf{Y}|\mathbf{W},\varepsilon) = \prod_{n=1}^{N} \int \mathcal{N}(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n, \sigma^2\mathbf{I})\ \mathcal{N}(\mathbf{x}_n|\mathbf{0},\mathbf{I})\ d\mathbf{x}_n = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n|\mathbf{0},\ \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}).$$

We can find parameters $\mathbf{W}$ through the maximization of the marginal likelihood. Tipping and Bishop [1999b] showed that finding the maximum likelihood solution is equivalent to finding eigenvalue problem which makes the solution analytically tractable and computationally efficient.

2.2.5.1 Dual probabilistic principle component analysis

The standard approach for latent variable models is to marginalize out the latent input and optimize the parameters. An alternative approach is to marginalize out the parameters and to optimize the latent inputs. This approach is known as Dual PPCA (DPPCA) by Lawrence [2005]. If we optimize our latent inputs and marginalize out the parameters, we need to put the prior on the parameters $\mathbf{W}$:

$$p(\mathbf{W}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{w}_n|\mathbf{0},\mathbf{I}).$$

Now we need to marginalize out the parameters:

$$\begin{aligned} p(\mathbf{Y}|\mathbf{X},\varepsilon) &= \prod_{n=1}^{N} \int \mathcal{N}(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n,\ \sigma^2\mathbf{I})\ \mathcal{N}(\mathbf{w}_n|\mathbf{0},\ \mathbf{I}) d\mathbf{w}_n \\ &= \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n|\mathbf{0},\ \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I}). \end{aligned}$$

This is the same as PPCA except that instead of finding the optimal values for $\mathbf{W}$, we need to find optimal locations of latent points, $\mathbf{X}$, in the same way as PPCA by Tipping and Bishop [1999b].

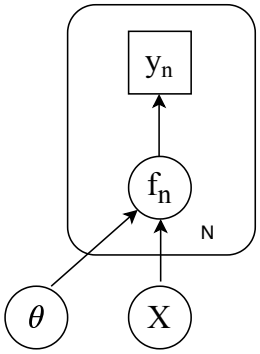


Fig. 2.6 Graphical model for GPLVM [Lawrence, 2005].

2.2.5.2 Gaussian process latent variable model (GPLVM)

The Gaussian Process Latent Variable Model (GPLVM) [Lawrence, 2005] is the non-linear version of dual PPCA. Lawrence [2005] recognized that the *kernel trick* [Schölkopf and Smola, 2002] can be applied to the inner product of $\mathbf{X}$ inside the model,

$$\begin{aligned}\hat{K} &= \mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I} \\ &= \mathbf{K} + \sigma^2\mathbf{I},\end{aligned}$$

where kernel $\mathbf{K}$ in the equation above, can be replaced by any valid linear or non-linear kernel.

The likelihood function for the GPLVM model is a product of p independent Gaussian processes, each associated with different dimensions of the data. So, we can rewrite the likelihood function as:

$$\begin{aligned}p(\mathbf{Y}|\mathbf{X}) &= \prod_{j=1}^{p} \int p(\mathbf{y}_j|\mathbf{f}_j)p(\mathbf{f}_j|\mathbf{X})d\mathbf{f} \\ &= \prod_{j=1}^{p} \mathcal{N}(\mathbf{y}_j|\mathbf{0}, \mathbf{K}_{ff} + \sigma^2\mathbf{I}),\end{aligned} \tag{2.40}$$

which is calculated using Equations (2.4) and (2.7).
The graphical model representation for the GPLVM is illustrated in Figure 2.6. Because of the non-linear kernel, the maximum likelihood solution to find the locations of latent inputs $\mathbf{X}$ through an eigenvalue problem is not possible. To find the optimal positions, the marginal likelihood should be maximized with respect to $\mathbf{X}$ and all the hyper-parameters. In GPLVM, we calculate the maximum likelihood estimate of $\mathbf{X}$ and, in case of having a prior on $\mathbf{X}$, we

can measure the maximum a posteriori (MAP),

$$\mathbf{X}_{\text{MAP}} = p(\mathbf{Y}|\mathbf{X})p(\mathbf{X}). \tag{2.41}$$

We will describe a fully Bayesian version of GPLVM in Section 3.1.

2.2.6 Sparse Gaussian process

As mentioned in Chapter 1, Gaussian processes are among one of the most used distributions. They have a principled way of handling uncertainty by integrating what you had previously thought with what you have learned. However, the computational complexity $\mathcal{O}(N^3)$, where N is the size of training data, is a burden, and makes for intractablility for larger size datasets. There are various works in the literature attempting to tackle this issue for GP regression. The most popular approach in the literature to overcome computational complexity falls under the category of *sparse approximations*. Changing the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ is the aim in most of the literature. M is the number of inducing variables and is selected by a user.

Quiñonero-Candela and Rasmussen [2005] have a comprehensive review on sparse approximation techniques. Most of the approximation techniques use low-rank approximation, where $M \ll N$ and the computational complexity can change from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ by introducing *inducing inputs* (or auxiliary variables) [Csató and Opper, 2002; Herbrich et al., 2003; Seeger et al., 2003; Snelson and Ghahramani, 2006a; Titsias, 2009a]. The inducing variables $\mathbf{Z} \in \mathcal{R}^{M\times q}$ and their corresponding outputs, that are called *inducing points* $\mathbf{u}$, live in the same space as $\mathbf{X}$ and $\mathbf{f}$, respectively. While the original covariance was $N \times N$ matrix, the new covariance is much smaller, and it is an $M \times M$ matrix. There are different strategies for selecting the inducing inputs: inducing inputs are typically chosen from the subset of data [Quiñonero-Candela and Rasmussen, 2005] or optimized [Hensman et al., 2013; Titsias, 2009a]. The joint Gaussian distribution is defined as:

$$p(\mathbf{f},\mathbf{u}|\mathbf{X},\mathbf{Z}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{u}\end{bmatrix}\middle|\mathbf{0}, \begin{bmatrix}\mathbf{K}_{ff} & \mathbf{K}_{fu}\\ \mathbf{K}_{uf} & \mathbf{K}_{uu}\end{bmatrix}\right), \tag{2.42}$$

where $\mathbf{K}_{fu}$ is a covariance function between inputs $\mathbf{X}$ and inducing variables $\mathbf{Z}$. $\mathbf{K}_{uu}$ is constructed by evaluating the covariance function on the inducing inputs. The prior for the inducing points is defined as:

$$p(\mathbf{u}|\mathbf{Z}) = \mathcal{N}(\mathbf{u}|\mathbf{0},\, \mathbf{K}_{uu}). \tag{2.43}$$

The conditional distribution of $p(\mathbf{f}|\mathbf{u})$ is defined as:

$$p(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \underbrace{\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}}_{\tilde{\mathbf{K}}}). \tag{2.44}$$

By using the Gaussian identity in Appendix A.1 and Equations (2.43), and (2.44), the conditional distribution can be written as:

$$\begin{aligned} p(\mathbf{f}|\mathbf{X}) &= \int p(\mathbf{f}|\mathbf{u},\mathbf{X})\; p(\mathbf{u}|\mathbf{Z})\; d\mathbf{u} \\ &= \mathcal{N}(\mathbf{K}_{fu}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{0},\; \mathbf{K}_{ff} - \cancel{\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{\mathbf{uf}}} + \cancel{\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}}) \\ &= \mathcal{N}(\mathbf{f}|\mathbf{0},\; \mathbf{K}_{ff}), \end{aligned} \tag{2.45}$$

where we can drop the conditioning on $\mathbf{Z}$ because of conditional independence. By integrating out $\mathbf{u}$, this equation is the same as Gaussian process prior in the regression model in Section 2.2.1. Since, in its current form, adding inducing variables $\mathbf{u}$ does not solve the computational complexity issue $\mathcal{O}(N^3)$, sparse methods apply approximation, $q(\mathbf{f}|\mathbf{u},\mathbf{X})$, to the true conditional $p(\mathbf{f}|\mathbf{u},\mathbf{X})$ in Equation (2.44). This is done by replacing $\tilde{\mathbf{K}}$ in Equation (2.44) with a different covariance matrix $\tilde{\mathbf{Q}}$, where $\tilde{\mathbf{Q}}$ does not require an inversion of $\mathcal{O}(N^3)$, but instead $\mathcal{O}(NM^2)$, where $\tilde{\mathbf{Q}} \neq \tilde{\mathbf{K}}$.

$$q(\mathbf{f}|\mathbf{u},\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \tilde{\mathbf{Q}}) \tag{2.46}$$

By using the Gaussian identity in Appendix A.1, we can write the approximate conditional distribution as:

$$\begin{aligned} q(\mathbf{f}|\mathbf{X}) &= \int q(\mathbf{f}|\mathbf{u},\mathbf{X})\; p(\mathbf{u}|\mathbf{Z})\; d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f}|\mathbf{0},\; \tilde{\mathbf{Q}} + \mathbf{Q}_{ff}), \end{aligned} \tag{2.47}$$

where $\mathbf{Q}_{ff} = \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}$. A few famous approximation techniques that have made suggestions on how $\tilde{\mathbf{Q}}$ should be approximated are deterministic training conditional (DTC) by Csató and Opper [2002]; Seeger et al. [2003], fully independent training conditional (FITC) by Snelson and Ghahramani [2006b], and the variational sparse GP by Titsias [2009a]. In the above literature, they use the sparse approximation technique by replacing the full covariance matrix with a low-rank form. In DTC approximation [Csató and Opper, 2002; Seeger et al., 2003], they assume $\tilde{\mathbf{Q}} = \mathbf{0}$. In FITC approximation [Snelson and Ghahramani, 2006b], they replace $\tilde{\mathbf{Q}}$'s diagonal with the exact $\mathbf{K}_{ff}$'s diagonal. A comprehensive overview of sparse

GPs and a unified view of variational sparse GP is available by Damianou [2015]. The literature mentioned above is for sparse GPs in the regression cases.

Titsias and Lawrence [2010] extended variational sparse GP for regression [Titsias, 2009a] to the latent variable models. We are interested in the variational approach and we will explain it in more detail in the following section.

2.2.6.1 Variational sparse Gaussian process

In variational sparse GP [Titsias, 2009a], the aim is to minimize the Kullback-Leibler (KL) divergence between the true posterior and the approximate posterior. This is the standard way of computing variational inference. In contrast, in Titsias [2009a] the bound on the true log marginal likelihood is maximized. The main difference between this method and other sparse approximate techniques is that the inducing inputs here are defined as variational parameters. The inducing inputs are selected by minimizing the KL divergence between variational GP and exact posterior. This is the same as maximizing the variational lower bound with respect to the exact posterior and approximate posterior [Titsias, 2009a]. The GP prior is augmented by introducing inducing points and the new GP prior is:

$$p(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})p(\mathbf{u}). \tag{2.48}$$

The augmented true posterior is defined as:

$$p(\mathbf{f},\mathbf{u}|\mathbf{y}) = p(\mathbf{f}|\mathbf{u})p(\mathbf{u}|\mathbf{y}). \tag{2.49}$$

The marginal likelihood is defined as:

$$\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{f}d\mathbf{u}. \tag{2.50}$$

The approximation to the exact marginal likelihood is introduced as $q(\mathbf{f},\mathbf{u})$ and is defined as:

$$q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u}), \tag{2.51}$$

where $q(\mathbf{u})$ is a variational distribution over $\mathbf{u}$. In the standard variational inference, the KL divergence is minimized, which is equal to maximizing the bound on the true log marginal

likelihood. By applying Jensen's inequality the bound is defined as:

$$
\begin{aligned}
\log p(\mathbf{y}|\mathbf{X}) &= \log \int\int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{f}d\mathbf{u} \\
&= \log \int\int q(\mathbf{f},\mathbf{u})\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})}{q(\mathbf{f},\mathbf{u})}d\mathbf{f}d\mathbf{u} \\
&\geq \int\int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})}{p(\mathbf{f}|\mathbf{u})q(\mathbf{u})}d\mathbf{f}d\mathbf{u} \\
&= \int\int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{f})\cancel{p(\mathbf{f}|\mathbf{u})}p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})}q(\mathbf{u})}d\mathbf{f}d\mathbf{u} \\
&= \int\int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\log\frac{p(\mathbf{y}|\mathbf{f})\ p(\mathbf{u})}{q(\mathbf{u})}d\mathbf{f}d\mathbf{u} \\
&= \int q(\mathbf{u})\Big[\int p(\mathbf{f}|\mathbf{u})\log p(\mathbf{y}|\mathbf{f})d\mathbf{f} + \log\frac{p(\mathbf{u})}{q(\mathbf{u})}\Big]d\mathbf{u} \\
&= \int q(\mathbf{u})\Big[\int p(\mathbf{f}|\mathbf{u})\log p(\mathbf{y}|\mathbf{f})d\mathbf{f}\Big]d\mathbf{u} + \int q(\mathbf{u})\Big[\log\frac{p(\mathbf{u})}{q(\mathbf{u})}\Big]d\mathbf{u} \\
&= \int q(\mathbf{u})\Big[\underbrace{\int p(\mathbf{f}|\mathbf{u})\log p(\mathbf{y}|\mathbf{f})d\mathbf{f}}_{\tilde{\mathcal{L}}}\Big]d\mathbf{u} - \mathrm{KL}(q(\mathbf{u})\|p(\mathbf{u})),
\end{aligned}
\tag{2.52}
$$

where $\log(\mathbf{y}|\mathbf{u})$ in Equation (2.52) is defined as:

$$
\begin{aligned}
\tilde{\mathcal{L}} = \log(\mathbf{y}|\mathbf{u}) &= \langle \log p(\mathbf{y}|\mathbf{f})\rangle_{p(\mathbf{f}|\mathbf{u})} \\
&= \langle \log\ \mathcal{N}(\mathbf{y}|\mathbf{f},\sigma^2\mathbf{I})\rangle \\
&= \log\left[\mathcal{N}(\mathbf{y}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u},\sigma^2\mathbf{I})\right] - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right).
\end{aligned}
\tag{2.53}
$$

Please note that $\langle . \rangle$ refers to an expectation calculation. We can factorize the likelihood due to the conditional independence, so the bound in Equation (2.52) becomes:

$$
\log p(\mathbf{y}|\mathbf{X}) \geq \int q(\mathbf{u})\Big[\sum_{i=1}^{N}\int p(f_i|\mathbf{u})\log p(y_i|f_i)df_i\Big]d\mathbf{u} - \mathrm{KL}(q(\mathbf{u})\|p(\mathbf{u})) \tag{2.54}
$$

If we take a derivative of this bound w.r.t. the variational distribution $q(\mathbf{u})$, we can find the optimal $q(\mathbf{u})$, which turns out to be a Gaussian distribution with mean and covariance as

below:

$$
\begin{aligned}
q(\mathbf{u}) &= \mathcal{N}(\mathbf{u}|\boldsymbol{\mu},\mathbf{S}) \\
\boldsymbol{\mu} &= \sigma^{-2}\mathbf{K}_{uu}(\mathbf{K}_{uu}+\sigma^{-2}\mathbf{K}_{uf}\mathbf{K}_{fu})^{-1}\mathbf{K}_{uf}\mathbf{y}, \\
\mathbf{S} &= \mathbf{K}_{uu}(\mathbf{K}_{uu}+\sigma^{-2}\mathbf{K}_{uf}\mathbf{K}_{fu})^{-1}\mathbf{K}_{uu}
\end{aligned} \tag{2.55}
$$

By computing the integral in Equation (2.52), the final bound becomes:

$$
\mathcal{L} = \log\left[\mathcal{N}(\mathbf{y}|\mathbf{0},\sigma^2\mathbf{I}+\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf})\right] - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff}-\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right). \tag{2.56}
$$

This bound is coupled between data and it requires all the data at once and not in batches. Very detailed derivations can be found in Damianou [2015]; Titsias [2009b]. Here the trace term is a regularization term and this is the main difference between Titsias [2009a] and the other methods mentioned earlier.

Stochastic variational inference In order to apply GPs to larger datasets, combining the idea of inducing variables with advances in variational inference [Hoffman et al., 2013] helps to develop a practical algorithm for applying GPs using Stochastic Variational Inference (SVI). In the variational inference in Titsias [2009a], we integrate out inducing points $\mathbf{u}$. However, when applying SVI to GPs we need to maintain the inducing variables. In the context of SVI, the variables $\mathbf{u}$ will perform the role of the global variables that are needed to apply SVI to GPs. Hensman et al. [2013] points out that, in the lower bound $\tilde{\mathcal{L}}$, it is possible to factorize the bound w.r.t. the number of data points and dimensionality. This is useful for very big data, since mini-batches can be used. We will go on to use SVI in Chapter 5. For SVI, we re-write Equation (2.52)

$$
\log p(\mathbf{y}|\mathbf{X}) \geq \int q(\mathbf{u})\Big[\underbrace{\sum_{i=1}^{N}\int p(f_i|\mathbf{u})\log p(y_i|f_i)df_i}_{\tilde{\mathcal{L}}} + \log\frac{p(\mathbf{u})}{q(\mathbf{u})}\Big]d\mathbf{u} \tag{2.57}
$$

$$
= \int q(\mathbf{u})\Big[\tilde{\mathcal{L}} + \log\frac{p(\mathbf{u})}{q(\mathbf{u})}\Big]d\mathbf{u} \tag{2.58}
$$

$$
= \langle\tilde{\mathcal{L}} + \log p(\mathbf{u}) - \log q(\mathbf{u})\rangle_{q(\mathbf{u})} \tag{2.59}
$$

$$
= \langle\tilde{\mathcal{L}}\rangle_{q(\mathbf{u})} - \mathrm{KL}(q(\mathbf{u})\|p(\mathbf{u})) \tag{2.60}
$$

The $\mathcal{L}$ above is factorized as a sum of N terms and each of the terms corresponding to one input/output pair (x_i, y_i), and mini-batches of data can be used to evaluate the bound.

Prediction The approximate predictive distribution for the sparse Gaussian process [Quinonero-Candela et al., 2007; Saul, 2016] is calculated by using the approximate posterior $q(\mathbf{u})$,

$$p(\mathbf{f}_*|\mathbf{y}) = \int p(\mathbf{f}_*|\mathbf{f})p(\mathbf{f}|\mathbf{u},\mathbf{y})p(\mathbf{u}|\mathbf{Z},\mathbf{y})d\mathbf{f}\,d\mathbf{u} \tag{2.61}$$

$$\simeq \int p(\mathbf{f}_*|\mathbf{f})p(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{f}\,d\mathbf{u} \tag{2.62}$$

$$= \int \mathcal{N}(\mathbf{f}_*|\mathbf{K}_{f_*u}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{f_*f_*} - \mathbf{K}_{f_*u}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf_*})q(\mathbf{u})d\mathbf{u} \tag{2.63}$$

$$= \mathcal{N}(\mathbf{f}_*|\mathbf{K}_{f_*u}\mathbf{K}_{uu}^{-1}\boldsymbol{\mu}, \mathbf{K}_{f_*f_*} - \mathbf{K}_{f_*u}(\mathbf{K}_{uu}^{-1} - \mathbf{K}_{uu}^{-1}\mathbf{S}\mathbf{K}_{uu}^{-1})\mathbf{K}_{uf_*}), \tag{2.64}$$

where $\boldsymbol{\mu}$ and $\mathbf{S}$ are the mean and covariance of the approximate posterior $q(\mathbf{u})$, and is calculated in Equation (2.55).

2.3 Conclusion

In this Chapter, we introduced the imbalanced data problem and reviewed the basics of the Gaussian process framework. Related work regarding the binary imbalanced data problem was discussed and assessment metrics were introduced to properly assess the performance of the imbalanced data. We introduced Gaussian process, such as GP for regression, GP for classification, and latent variable models such as GPLVM. The standard form of GP is not suitable for non-Gaussian likelihood and also does not scale well with the size of the data. For example, in GP classification, the posterior can not be computed analytically and approximation techniques are needed. We will consider these base models as tools to overcome challenges that arise from data scarcity issues, lack of labeled data, and the imbalanced data problem in the following chapters.

Chapter 3

Learning imbalanced data using structure consolidation latent variable model

In many medical applications (e.g. histopathology), negatively labeled data is extremely easy to obtain (e.g. healthy cells). Positive labels, on the other hand, can be harder to acquire (e.g. particular disease morphologies). The insufficiency of positive data (cancerous cells) raises data scarcity issues that result in an imbalanced data problem. These massively imbalanced problems are challenging for most algorithms because the negative class tends to dominate the objective function and the resulting model performs poorly. As discussed in Chapter 2, there are different ways of dealing with imbalanced data. In practice, the most common approach is often to throw away much of the negative data and re-balance the data set. However, as we mentioned in Chapter 2, this has its own disadvantages, and we might lose a lot of useful information by discarding some of the negative class.

In this chapter[1], we work with a breast cancer dataset. Cancer diagnosis is a challenging and important task in modern healthcare. Breast cancer is the most common type of cancer in the UK with 1 in 8 women diagnosed with breast cancer during their lifetime [Key et al., 2001]. Accurate diagnosis is important for early intervention and to provide patients with appropriate treatment.

Mitosis is the process of cell division, and this cell division within a tumor gives indications of the rate of growth of the tumor [Snell, 2013]. Automatic mitosis detection can aid pathologists in their work, and could also alleviate the tediousness of manual mitosis counting.

[1]This chapter is based on "Unsupervised Learning with Imbalanced Data via Structure Consolidation Latent Variable Model", which was published through the International Conference on Learning Representations (ICLR) workshop track, 2016 - a paper where I appear as a first author.

It usually takes between 5-10 minutes for a pathologist to perform mitosis counting, and this process sometimes needs to be repeated in different areas or sections for the borderline cases [Veta et al., 2015]. Detecting mitosis is one of the most challenging tasks since there are no separate cells with plain backgrounds; instead, there is a diversity of shapes and textures.

We tackle this problem with latent variable models, leveraging their generative capabilities. Latent variable models have attracted a lot of attention as they have the potential to serve as an underpinning technology for a range of challenges such as generative modeling, missing data imputation, and coping with multiple data modalities. Latent variable models, like GPLVM, introduced in Chapter 2, can also be applied to a wider range of data sets, because they do not rely on having carefully labeled data available.

In chapter 2, we introduced the imbalanced data problem and also Gaussian processes. As mentioned in Section 2.2.5.2, the GPLVM is an unsupervised, probabilistic dimensionality reduction technique. However, GPLVM uses a maximum likelihood solution to find the optimal positions for the latent inputs $\mathbf{X}$. The maximum likelihood approach seeks to find a single point estimate of $\mathbf{X}$ rather than a density estimation. On the other hand, in Bayesian methods, rather than finding a MAP solution we wish to integrate out all possible settings for the latent inputs and compute the marginal likelihood. For this reason we will introduce Bayesian Gaussian Process Latent Variable Models (Bayesian-GPLVM) in Section 3.1.

We build latent variable models using Bayesian-GPLVM methodology that can simultaneously accommodate a large number of negative examples while sharing their characteristics appropriately with the positive class. This allows the model to characterize the manner in which the positive and negative classes are differently characterized through preserved (or private) latent spaces that are separately learned for each class. The resulting model does not suffer from the standard challenges faced in this domain. For example, the model does not focus only on the majority class (negative examples) and does not ignore the minority class (positive examples). We call this method Structure Consolidation Latent Variable Model (SCLVM), and it is introduced in Section 3.2.

3.1 Bayesian Gaussian process latent variable model

A GPLVM [Lawrence, 2005] is defined as a mapping from the latent space to the observation space. As mentioned in Section 2.2.5.2, in a GPLVM model, the aim is maximizing a posteriori (MAP) estimates of latent variables $\mathbf{X}$ and jointly maximizing the kernel hyperparameters. In a Bayesian Gaussian Process Latent Variable Model (Bayesian-GPLVM), we get a density estimation, rather than a point estimation, of $\mathbf{X}$. Titsias and Lawrence [2010] uses a variational approach for approximating the posterior and marginalizes the

latent variables $\mathbf{X}$. Similar to Section 2.2.6.1, they optimize the lower bound on the marginal likelihood w.r.t. the hyper-parameters. In the equations below, $\mathbf{Y}$ is observed data and $\mathbf{Y} \in \mathcal{R}^{N \times p}$, where N is the size of the observed data and p is the dimensionality of the data. $\mathbf{X}$ is latent data and $\mathbf{X} \in \mathcal{R}^{N \times q}$, q is the dimensionality of the latent variables and it is assumed $q \ll p$. The likelihood can be factorised w.r.t. the dimensions, since GPs are taken to be independent across features:

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{j=1}^{p} p(\mathbf{y}_j|\mathbf{X}), \tag{3.1}$$

where $\mathbf{y}_j$ represents the j^{th} column of $\mathbf{Y}$. Since $\mathbf{X}$ is a latent variable, we assume a prior density given by the normal distribution over data points $\mathbf{x}_n$:

$$p(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{0}, \mathbf{I}). \tag{3.2}$$

We wish to compute the log marginal likelihood which is defined as:

$$p(\mathbf{Y}) = \int p(\mathbf{Y}|\mathbf{X})p(\mathbf{X})d\mathbf{X}. \tag{3.3}$$

We cannot calculate the marginal likelihood, since the integral above is intractable [Titsias and Lawrence, 2010]. Instead, we use approximate variational inference. The aim is to approximate the true posterior $p(\mathbf{X}|\mathbf{Y})$ with the approximate posterior $q(\mathbf{X})$ over the latent variables. The assumption is that $q(\mathbf{X})$ is also a Gaussian with mean μ and covariance $\mathcal{S}$:

$$q(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}|\mu_n, \mathcal{S}_n), \tag{3.4}$$

where μ_n and $\mathcal{S}_n$ are the variational parameters, and $\mathcal{S}_n$ is assumed to be a diagonal matrix for simplicity [Titsias and Lawrence, 2010].

The Bayesian-GPLVM by Titsias and Lawrence [2010] addresses the intractability problem in Equation (3.3) by introducing a variational lower bound on the marginal likelihood, using the sparse GP framework that was introduced in Section 2.2.6.1. Refer to Damianou [2015] for a detailed derivation. By applying Jensen's inequality on the marginal likelihood $p(\mathbf{Y})$:

$$
\begin{aligned}
\log p(\mathbf{Y}) &\geq \mathcal{L} \qquad (3.5)\\
&= \int q(\mathbf{X}) \log \frac{p(\mathbf{Y}|\mathbf{X})p(\mathbf{X})}{q(\mathbf{X})} d\mathbf{X} \\
&= \int q(\mathbf{X}) \Big[\log p(\mathbf{Y}|\mathbf{X}) + \log \frac{p(\mathbf{X})}{q(\mathbf{X})} \Big] d\mathbf{X} \\
&= \int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{X}) d\mathbf{X} + \underbrace{\int q(\mathbf{X}) \log \frac{p(\mathbf{X})}{q(\mathbf{X})} d\mathbf{X}}_{\text{-KL}} \\
&= \underbrace{\int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{X}) d\mathbf{X}}_{\tilde{\mathcal{L}}(q)} - \mathrm{KL}\left(q(\mathbf{X}) \,\|\, p(\mathbf{X})\right) \\
&= \tilde{\mathcal{L}}(q) - \mathrm{KL}\left(q(\mathbf{X}) \,\|\, p(\mathbf{X})\right),
\end{aligned}
$$

where $\mathcal{L}$ refers to the variational lower bound. The KL term is calculated analytically since both distributions are Gaussians. Calculating the first part of the equation is analytically intractable [Titsias and Lawrence, 2010]. Titsias and Lawrence [2010] proposed an approach for Bayesian-GPLVM similar to the variational sparse GP method [Titsias, 2009a] that is described in Section 2.2.6.1. The first part of the bound is difficult to estimate (because of the term $p(\mathbf{y}_j|\mathbf{X})$) but we can factorise it w.r.t. the features mentioned in Equation (3.1):

$$
\begin{aligned}
\tilde{\mathcal{L}}(q) &= \sum_{j=1}^{p} \int q(\mathbf{X}) \log p(\mathbf{y}_j|\mathbf{X}) d\mathbf{X} \qquad (3.6)\\
&= \sum_{j=1}^{p} \tilde{\mathcal{L}}_j(q).
\end{aligned}
$$

By introducing the concept of inducing inputs defined in Section 2.2.6.1, and by using Equation (3.1), $p(\mathbf{y}_j|\mathbf{X})$ can be written as:

$$
p(\mathbf{y}_j|\mathbf{X}) = p(\mathbf{y}_j|\mathbf{f}_j) p(\mathbf{f}_j|\mathbf{X}, \mathbf{u}_j, \mathbf{Z}) p(\mathbf{u}_j|\mathbf{Z}),
$$

where we have prior probabilities defined as

$$
p(\mathbf{u}_j|\mathbf{Z}) = \mathcal{N}(\mathbf{u}_j|\mathbf{0},\, \mathbf{K}_{uu}) \qquad (3.7)
$$

$$
p(\mathbf{f}_j|\mathbf{u}_j, \mathbf{X}, \mathbf{Z}) = \mathcal{N}(\mathbf{f}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}), \qquad (3.8)
$$

where $\mathbf{U} \in \mathcal{R}^{M \times p}$, $\mathbf{Z} \in \mathcal{R}^{M \times q}$ and vectors $\mathbf{u}_j, \mathbf{f}_j$ correspond to the j^{th} dimension of the data. We assume that $\mathbf{y}_j$ is a noise corrupted version of $\mathbf{f}_j$ given the independence assumption across features. The likelihood $p(\mathbf{y}_j|\mathbf{f}_j)$ is then defined as:

$$p(\mathbf{y}_j|\mathbf{f}_j) = \mathcal{N}(\mathbf{y}_j|\mathbf{f}_j, \sigma^2\mathbf{I}). \tag{3.9}$$

The marginal likelihood and the prior probabilities are defined in the Background chapter in Equations (2.52), (2.43), and (2.44), respectively.

We can approximate the true posterior with a sparse variational distribution [Titsias, 2009a]. The true posterior is defined as:

$$p(\mathbf{f}_j, \mathbf{u}_j|\mathbf{y}_j, \mathbf{X}) = p(\mathbf{f}_j|\mathbf{u}_j, \mathbf{y}_j, \mathbf{X})\ p(\mathbf{u}_j|\mathbf{y}_j, \mathbf{X}). \tag{3.10}$$

The sparse variational distribution takes the form:

$$q(\mathbf{f}_j, \mathbf{u}_j) = p(\mathbf{f}_j|\mathbf{u}_j, \mathbf{X})\ q(\mathbf{u}_j), \tag{3.11}$$

where $q(\mathbf{u}_j)$ is a variational distribution over the inducing points.

Thus the lower bound appears as:

$$\begin{aligned} \log p(\mathbf{y}_j|\mathbf{X}) \geq & \int q(\mathbf{u}_j)\ \log \left[\frac{p(\mathbf{u}_j)\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\ \sigma^2\mathbf{I})}{q(\mathbf{u}_j)}\right] d\mathbf{u}_j \\ & - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right), \end{aligned} \tag{3.12}$$

where we used the Equation (2.53) to calculate $\langle \log p(\mathbf{y}|\mathbf{f}) \rangle_{p(\mathbf{f}|\mathbf{u})}$.
For the Bayesian-GPLVM, we need to marginalize out $\mathbf{X}$ as well, so by combining Equations (3.6) and (3.12) we get:

$$\begin{aligned} \tilde{\mathcal{L}}_j(q) = & \int q(\mathbf{X}) \left[\int q(\mathbf{u}_j)\ \log \left[\frac{p(\mathbf{u}_j)\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\ \sigma^2\mathbf{I})}{q(\mathbf{u}_j)}\right] d\mathbf{u}_j \right. \\ & \left. - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right) \right] d\mathbf{X}. \end{aligned} \tag{3.13}$$

Since $q(\mathbf{u}_j)$ is independent from the random variable $\mathbf{X}$, swapping the integration between $\mathbf{u}_j$ and $\mathbf{X}$, and performing the integral on $\mathbf{X}$ first, we will have:

$$\begin{aligned}\tilde{\mathcal{L}}_j(q) &= \int q(\mathbf{u}_j)\Bigg[\int q(\mathbf{X})\ \log\ \Big[\frac{p(\mathbf{u}_j)\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\ \sigma^2\mathbf{I})}{q(\mathbf{u}_j)}\Big]d\mathbf{u}_j \\ &\quad - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right)\Bigg]d\mathbf{X} \\ &= \int q(\mathbf{u}_j)\Big[\langle\log\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\ \sigma^2\mathbf{I})\rangle_{q(\mathbf{X})} + \log\frac{p(\mathbf{u}_j)}{q(\mathbf{u}_j)}\Big]d\mathbf{u}_j \\ &\quad - \frac{1}{2\sigma^2}\mathrm{tr}\left(\langle\mathbf{K}_{ff}\rangle_{q(\mathbf{X})}\right) + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{uu}^{-1}\langle\mathbf{K}_{uf}\mathbf{K}_{fu}\rangle_{q(\mathbf{X})}\right).\end{aligned} \tag{3.14}$$

In the equation above, 'tr' denotes trace of matrix and '$\langle.\rangle_{q(\mathbf{X})}$' denotes expectation under the distribution $q(\mathbf{X})$. It is now possible to maximize the lower bound w.r.t. the distribution $q(\mathbf{u})$:

$$\begin{aligned}\tilde{\mathcal{L}}_j(q) &= \int q(\mathbf{u}_j)\log\frac{e^{\langle\log\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\sigma^2\mathbf{I})\rangle_{q(\mathbf{X})}}p(\mathbf{u}_j)}{q(\mathbf{u}_j)}d\mathbf{u}_j \\ &\quad - \frac{1}{2\sigma^2}\mathrm{tr}\left(\langle\mathbf{K}_{ff}\rangle_{q(\mathbf{X})}\right) + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{uu}^{-1}\langle\mathbf{K}_{uf}\mathbf{K}_{fu}\rangle_{q(\mathbf{X})}\right).\end{aligned} \tag{3.15}$$

This expression is a KL-like quantity and it will reach its optimal value when $q(\mathbf{u}_j)$ is set to be proportional to the numerator inside the logarithm of the above equation [Damianou et al., 2014]:

$$q(\mathbf{u}_j) \propto e^{\langle\log\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\sigma^2\mathbf{I})\rangle_{q(\mathbf{X})}}\ p(\mathbf{u}_j),$$

or

$$\log q(\mathbf{u}_j) \propto \langle\log\mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\sigma^2\mathbf{I})\rangle_{q(\mathbf{X})} + \log p(\mathbf{u}_j).$$

Now, by reversing Jensen's inequality and moving the log outside of the integral in Equation (3.15), the bound will be:

$$\tilde{\mathcal{L}}_j(q) = \log \int q(\mathbf{u}_j) \frac{e^{\langle \log \mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u},\sigma^2\mathbf{I})\rangle_{q(\mathbf{X})}}\, p(\mathbf{u}_j)}{q(\mathbf{u}_j)} d\mathbf{u}_j \tag{3.16}$$

$$- \frac{1}{2\sigma^2}\mathrm{tr}\left(\langle \mathbf{K}_{ff}\rangle_{q(\mathbf{X})}\right) + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{uu}^{-1}\langle \mathbf{K}_{uf}\mathbf{K}_{fu}\rangle_{q(\mathbf{X})}\right)$$

$$= \log \int \cancel{q(\mathbf{u}_j)} \frac{e^{\langle \log \mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\sigma^2\mathbf{I})\rangle_{q(\mathbf{X})}}\, p(\mathbf{u}_j)}{\cancel{q(\mathbf{u}_j)}} d\mathbf{u}_j \tag{3.17}$$

$$- \frac{1}{2\sigma^2}\mathrm{tr}\left(\langle \mathbf{K}_{ff}\rangle_{q(\mathbf{X})}\right) + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{uu}^{-1}\langle \mathbf{K}_{uf}\mathbf{K}_{fu}\rangle_{q(\mathbf{X})}\right)$$

$$= \log \int e^{\langle \log \mathcal{N}(\mathbf{y}_j|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}_j,\sigma^2\mathbf{I})\rangle_{q(\mathbf{X})}}\, p(\mathbf{u}_j) d\mathbf{u}_j$$

$$- \frac{1}{2\sigma^2}\mathrm{tr}\left(\langle \mathbf{K}_{ff}\rangle_{q(\mathbf{X})}\right) + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{uu}^{-1}\langle \mathbf{K}_{uf}\mathbf{K}_{fu}\rangle_{q(\mathbf{X})}\right).$$

This equation can be calculated in closed form since it is a Gaussian distribution [Titsias and Lawrence, 2010] and we need to compute the below statistics ψ_0, Ψ_1, Ψ_2. These equations are referred to as Ψ statistics.

$$\psi_0 = \mathrm{tr}\left(\langle \mathbf{K}_{ff}\rangle_{q(\mathbf{X})}\right) \tag{3.18}$$

$$\Psi_1 = \langle \mathbf{K}_{fu}\rangle_{q(\mathbf{X})} \tag{3.19}$$

$$\Psi_2 = \langle \mathbf{K}_{uf}\mathbf{K}_{fu}\rangle_{q(\mathbf{X})} \tag{3.20}$$

These statistics are analytically tractable for certain kernels, such as the RBF kernel and linear kernel [Damianou et al., 2014]. For the RBF kernel, it can be shown that the closed form of the lower bound on $\tilde{\mathcal{L}}$ is defined as:

$$\tilde{\mathcal{L}}_j(q) = \log\left[\frac{\sigma^{-N}\,|\mathbf{K}_{uu}|^{\frac{1}{2}}}{2\pi^{\frac{N}{2}}\,|\sigma^2\Psi_2 + \mathbf{K}_{uu}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{y}_j^\top \mathbf{W}\mathbf{y}_j}\right] - \frac{\psi_0}{2\sigma^2} + \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{uu}^{-1}\Psi_2\right), \tag{3.21}$$

where $\mathbf{W} = \sigma^{-2}\mathbf{I} - \sigma^{-4}\Psi_1(\sigma^{-2}\Psi_2 + \mathbf{K}_{uu})^{-1}\Psi_1^\top$ and instead of the inverse of $\mathbf{K}_{ff}$ matrix, we will compute the inverse of $\mathbf{K}_{uu}$, which is much cheaper. More details on calculation of Ψ statistics is available in Appendix B.1. In the next section we will build up on the Bayesian-GPLVM and describe our novel method called Structure Consolidation Latent Variable Model.

3.2 Structure consolidation latent variable model

Our probabilistic latent variable model, which we call Structure Consolidation Latent Variable Model (SCLVM), divides its latent space into a shared space of all of the categories in question and a private space for each category [Damianou et al., 2012]. The shared space captures the common regularities among categories (e.g. positive and negative classes), while the private space is dedicated to modeling the variance specific to individual categories. Because the modeling of the private space is category specific, there is no domination of the characteristics of the private space by the larger category. Thus the data in each category can be modeled appropriately while the common regularities are still exploited.

We implement the idea of a shared and private space in the framework of Bayesian Gaussian Process Latent Variable Models [Titsias and Lawrence, 2010] by deriving a particular covariance function (kernel) that enables such a separation. We exploit closed form variational lower bounds of the log marginal likelihood of the proposed model, which provides an efficient approximation inference method.

The performance of our model is to be evaluated with a real image dataset, in which the positive and negative data are extremely imbalanced. We show that our model can still learn from imbalanced data and perform well in both generative and discriminative tasks. We will explain the SCLVM model in detail below.

We assume that the dataset is represented as a set of fixed length vectors collected in a matrix $\mathbf{Y} \in \Re^{N \times p}$. Additionally, a label of category is associated with each data point, $\mathbf{c} = (c^{(1)}, \ldots, c^{(N)}), c^{(n)} \in \{1, \ldots, C\}$, where C indicates the number of categories in the dataset. Our aim is to build a probabilistic model $p(\mathbf{Y})$ that is robust when the numbers of data in different categories are highly imbalanced.

We assume that the data $\mathbf{Y}$ is associated with a set of latent representations $\mathbf{X} \in \Re^{N \times q}$, where q is the dimensionality of the latent space. The latent representations are related to the observed data through an unknown mapping function f, which follows a prior distribution that is defined as a Gaussian process,

$$y = f(\mathbf{x}) + \varepsilon, \quad f \sim \mathcal{GP}(0, k), \tag{3.22}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ denotes the observation noise and k is the kernel function. Given the observed data $\mathbf{Y}$, we wish to compute the marginal likelihood of the data using approximate variational inference. The aim is to approximate the true posterior distribution with a variational distribution $q(\mathbf{X})$ over the latent variables. In our model, we separate the latent space into a shared space with the dimensionality q_s and a private space with the dimensionality q_p. Therefore, a latent representation can be denoted as $\mathbf{X} = [\mathbf{x}_s^\top, \mathbf{x}_p^\top]^\top, \mathbf{x}_s \in \Re^{q_s}, \mathbf{x}_p \in \Re^{q_p}$,

where $\mathbf{x}_s$ and $\mathbf{x}_p$ are the latent representations in the shared and private spaces, respectively. Using this separated latent representation, we define the kernel function in our model as

$$k((\mathbf{x}, c_{\mathbf{x}}), (\mathbf{x}', c_{\mathbf{x}'})) = k_s(\mathbf{x}_s, \mathbf{x}'_s) + k_p((\mathbf{x}_p, c_{\mathbf{x}}), (\mathbf{x}'_p, c_{\mathbf{x}'})), \tag{3.23}$$

where k_s is the kernel function for the shared space and k_p is the kernel function for the private space. The shared kernel can be any kernel function built on a vector space from the literature. However, the private kernel is defined as taking the following form:

$$k_p((\mathbf{x}_p, c_{\mathbf{x}}), (\mathbf{x}'_p, c_{\mathbf{x}'})) = \begin{cases} k'(\mathbf{x}_p, \mathbf{x}'_p), & c_{\mathbf{x}} = c_{\mathbf{x}'}, \\ 0, & c_{\mathbf{x}} \neq c_{\mathbf{x}'}, \end{cases} \tag{3.24}$$

where k' is the kernel function chosen to calculate the covariance and $c_{\mathbf{x}}$ is the label of category for the data point $\mathbf{x}$. We give a unit Gaussian prior distribution to the latent representations $\mathbf{X} \sim \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|0, \mathbf{I})$. The log marginal likelihood for the proposed model can be derived as $\log p(\mathbf{Y}|\mathbf{c}) = \log \int p(\mathbf{Y}|\mathbf{X}, \mathbf{c}) p(\mathbf{X}) \mathrm{d}\mathbf{X}$.

There is no analytical solution for this marginal likelihood. We apply variational inference and derive a closed form lower bound of the log marginal likelihood, by following a sparse Gaussian process approximation [Titsias and Lawrence, 2010], which is explained in more detail in Section 3.1:

$$\log p(\mathbf{Y}) \geq \sum_{j=1}^{p} \tilde{F}_j(q) - \mathrm{KL}(q(\mathbf{X}) \| p(\mathbf{X})), \tag{3.25}$$

where,

$$\tilde{F}_j(q) = \log \left[\frac{|\mathbf{K}_{uu}|^{\frac{1}{2}}}{(2\pi\sigma^2)^{\frac{N}{2}} |\sigma^2 \Psi_2 + \mathbf{K}_{uu}|^{\frac{1}{2}}} e^{-\frac{1}{2} \mathbf{y}_j^\top \mathbf{W} \mathbf{y}_j} \right] - \frac{\psi_0}{2\sigma^2} + \frac{1}{2\sigma^2} \mathrm{Tr}(\mathbf{K}_{uu}^{-1} \Psi_2), \tag{3.26}$$

where $\mathbf{W} = \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^4}\Psi_1 (\frac{1}{\sigma^2}\Psi_2 + \mathbf{K}_{uu})^{-1} \Psi_1^\top$, and ψ_0, Ψ_1, Ψ_2 are the expectation of covariance matrices w.r.t. the variational posterior $q(\mathbf{X})$. These equations are described in Section 3.1 and Appendix B.1 in more details. In our model, these expectations are derived

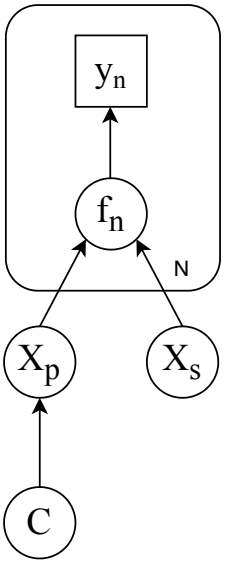


Fig. 3.1 Graphical representation of SCLVM. **y** represents observed data, **f** represents latent function, X_p is the latent input for the private space, X_s is the latent input for the shared space, and C refers to the class information.

as:

$$\psi_0 = \sum_{n=1}^{N} \left\langle k_s(\mathbf{x}_s^{(n)}, \mathbf{x}_s^{(n)}) \right\rangle_{q(\mathbf{x}_s^{(n)})} + \left\langle k_p((\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)}), (\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)})) \right\rangle_{q(\mathbf{x}_p^{(n)})} \tag{3.27}$$

$$(\Psi_1)_{nm} = \left\langle k_s(\mathbf{x}_s^{(n)}, \mathbf{z}_s^{(m)}) \right\rangle_{q(\mathbf{x}_s^{(n)})} + \left\langle k_p((\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)}), (\mathbf{z}_p^{(m)}, \mathbf{c}_\mathbf{z}^{(m)})) \right\rangle_{q(\mathbf{x}_p^{(n)})} \tag{3.28}$$

$$\begin{aligned}(\Psi_2)_{mm'} = &\sum_{n=1}^{N} \left\langle k_s(\mathbf{x}_s^{(n)}, \mathbf{z}_s^{(m)}) k_s(\mathbf{x}_s^{(n)}, \mathbf{z}_s^{(m')}) \right\rangle_{q(\mathbf{x}_s^{(n)})} \\ &+ \left\langle k_p((\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)}), (\mathbf{z}_p^{(m)}, \mathbf{c}_\mathbf{z}^{(m)})) k_p((\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)}), (\mathbf{z}_p^{(m')}, \mathbf{c}_\mathbf{z}^{(m')})) \right\rangle_{q(\mathbf{x}_p^{(n)})} \\ &+ \left\langle k_s(\mathbf{x}_s^{(n)}, \mathbf{z}_s^{(m)}) \right\rangle_{q(\mathbf{x}_s^{(n)})} \left\langle k_p((\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)}), (\mathbf{z}_p^{(m')}, \mathbf{c}_\mathbf{z}^{(m')})) \right\rangle_{q(\mathbf{x}_p^{(n)})} \\ &+ \left\langle k_p((\mathbf{x}_p^{(n)}, \mathbf{c}_\mathbf{x}^{(n)}), (\mathbf{z}_p^{(m)}, \mathbf{c}_\mathbf{z}^{(m)})) \right\rangle_{q(\mathbf{x}_p^{(n)})} \left\langle k_s(\mathbf{x}_s^{(n)}, \mathbf{z}_s^{(m')}) \right\rangle_{q(\mathbf{x}_s^{(n)})},\end{aligned} \tag{3.29}$$

where c_z are the variational parameters known as inducing labels. The graphical representation of the model is illustrated in Figure 3.1.

3.3 Experiments

Mitosis detection is a stage in tumor assessment that involves determining whether individual cells are in mitosis (dividing to reproduce). These cells are rare. We used the pre-processed

data from the assessment of mitosis detection algorithm (AMIDA) 2013 challenge by Snell [2013]. The original dataset is publicly available. The AMIDA dataset was released by the University Medical Center Utrecht. Some samples of original images are illustrated in Figure 3.2. These are breast biopsy tissue slices, and are stained to have better visibility. The standard staining process uses hematoxylin and eosin stains. The hematoxylin dyes the nuclei a dark purple color and the eosin dyes other structures a pink color [Veta et al., 2015]. Detecting mitosis is very challenging because the resulting images display a variety of shapes and textures. This dataset included low and high grade tumors. Two annotators labeled mitotic centroids in images and in case of disagreement, two additional experts labeled those images. The main goal of the AMIDA challenge was to find a mitosis detection method that can be automatic or semi-automatic.

We use the training set from the challenge, which consists of tissue images from 12 patients with various grades of disease, annotated by human experts. We used the pre-processed tissue

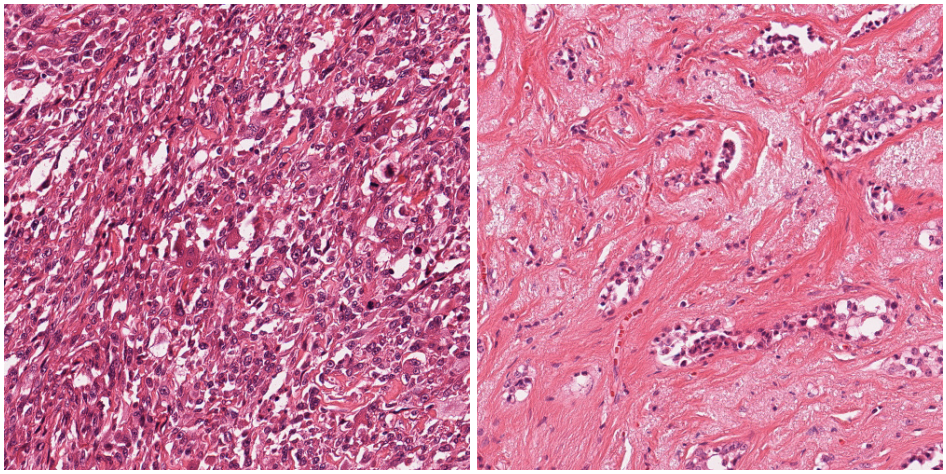


Fig. 3.2 Mitotic figures in Hematoxylin and Eosin(H&E) stained breast cancer.

images with the algorithm by Snell [2013] and focused on the generated candidate image patches. The pre-processing included histogram matching, detection of candidate locations, grey-scale conversion, and patch extraction. Histogram matching is the manipulation of pixels within an input image so that its histogram can match the histogram of a reference image. It is a way of normalizing an image and it is especially used when differences in conditions of image making, such as lighting, must be taken into account. The images are RGB images and have three channels, and histogram matching is applied to each channel. The source image is referred to each patient's images and target distribution is defined as the mean histogram from the whole training set. Histogram matching is performed by replacing the intensity values of source pixels with their corresponding values of the target histogram [Snell, 2013]. The annotations included only the centroid of the mitotic figure.

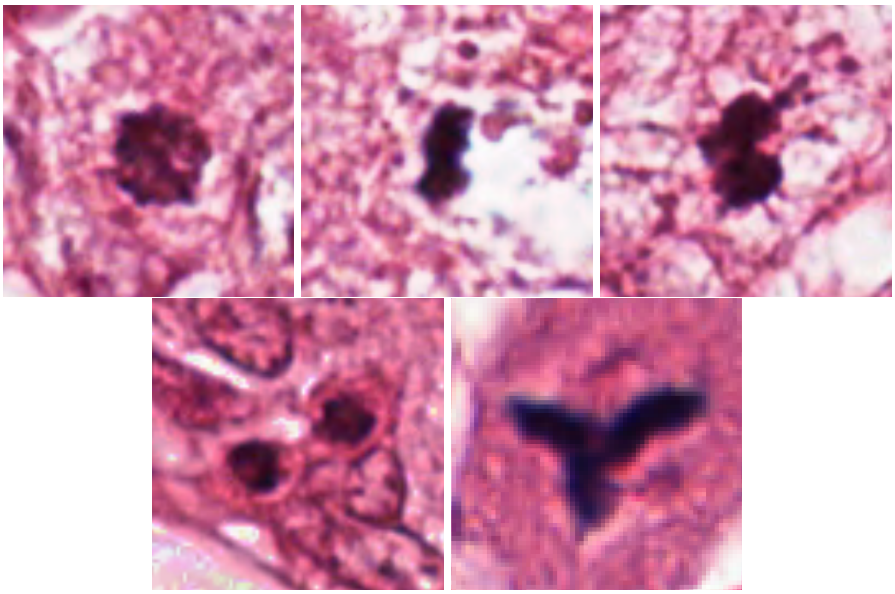
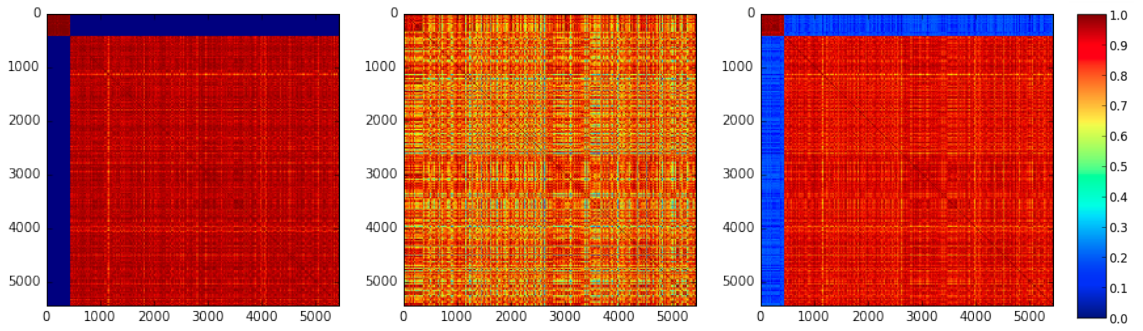


Fig. 3.3 Examples of different phases and variations of pre-processed 70×70 mitosis.

An approximations of 10 pixels radius around ground-truth marked locations by annotators (centroids) are assumed as candidate locations, since they are likely to be within the nucleus [Snell, 2013]. The 70×70 squares around each candidate location are used as patches. A few samples are illustrated in Figure 3.3. These patches are turned to grey-scale. Some grey-scale samples are illustrated in Figure 3.6a. For a detailed explanation regarding pre-processing, refer to Snell [2013]. The resulting image set contains $146,562$ grey-scale image patches, which we further resized to reduce the dimensionality to 30×30 pixels. Of the $146,562$ patches, manual annotation only identified 550 patches as positive (mitosis). We randomly take 80% of the positive images and $5,000$ negative images as training data. This gives in total $5,440$ images. Some examples of the training data are shown in Figure 3.6a. We applied SCLVM to this dataset and used an exponentiated quadratic kernel for both the shared and private space. The covariance matrices for the private space, public space, and the whole kernel (sum of private and public spaces) are illustrated in Figure 3.4. We set the dimensionality of both shared and private space to 5. Based on our experiments, increasing dimensions above 5 did not have much effect on the final score. We used 100 inducing points and allocated 50 inducing inputs to each class.

Both the latent representations and kernel parameters are optimized until convergence. The resulting latent space is visualized in Figure 3.5. The positive and negative images present similar structures in the shared space, which demonstrates the discovered common regularities, while their corresponding private spaces are significantly different from each other. To demonstrate the ability of SCLVM in balancing the modeling capabilities between



(a) (b) (c)

Fig. 3.4 (a) Covariance matrix for private space. (b) Covariance matrix for shared space. (c) Covariance matrix for the whole kernel. In these figures the color blue represents no correlation and the color red represents a high correlation between data points. The matrix for the private space is a block diagonal matrix between two classes and can capture private characteristics for each category. The matrix for the shared space captures common structures between categories. The whole kernel is the sum of private and shared covariances.

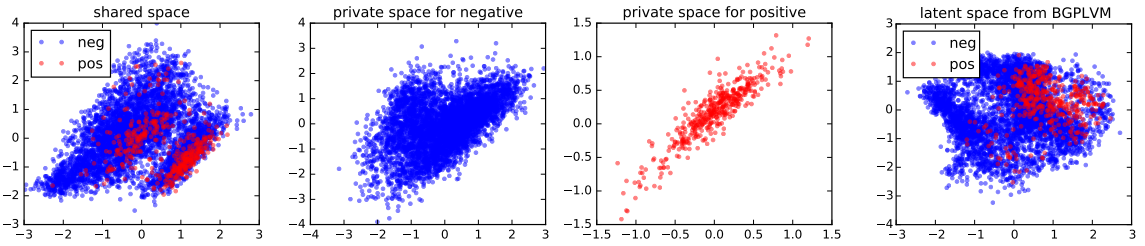
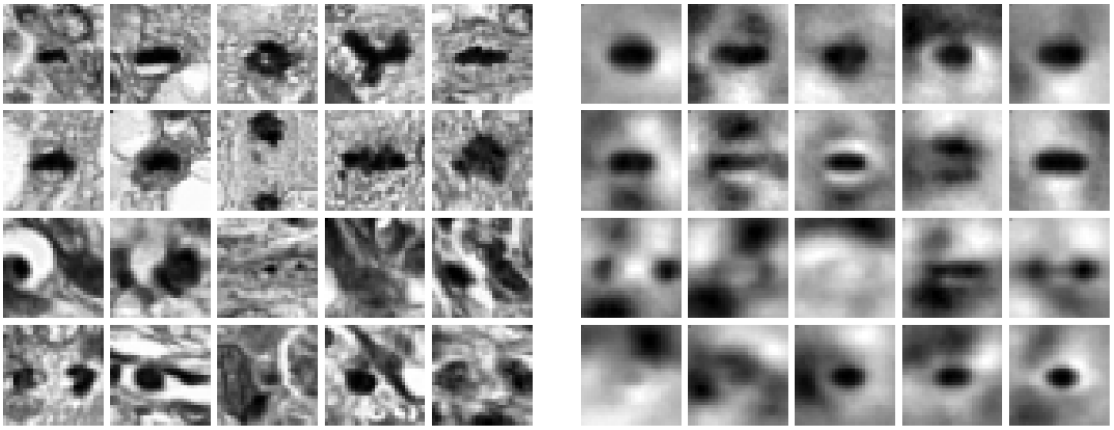


Fig. 3.5 The visualization of the training data in learned latent spaces. The first figure shows the positive and negative data in two of the shared dimensions. The second and third figures show two of the private dimensions for the negative and positive data respectively. The fourth figure shows the learned latent space from BGPLVM.

imbalanced categories, we draw samples from the learned latent space of SCLVM for both positive and negative categories (see Figure 3.6b). The generated samples from positive and negative categories are clearly different from each other, and they capture some characteristics of their own categories. We further evaluate the learned latent space by performing classification on a test set (the remaining positive examples, plus some randomly sampled negative examples, giving a total of 1000 images). We compare SCLVM with BGPLVM [Titsias and Lawrence, 2010] and Discriminative GPLVM[2] (DGPLVM) [Urtasun and Darrell, 2007] with ten test sets. DGPLVM uses an informative prior that encourages the latent position of the data points of the same class to be close to each other, and the ones from different classes to be far from each other. More information regarding DGPLVM is available in Appendix

[2]Due to its complexity $O(N^3)$, we only use 1000 images for training (440 positive and 560 negative).



(a) (b)

Fig. 3.6 (a) Some examples in the data sets. (b) Samples generated from the trained SCLVM. In both figures, the first two rows correspond to positive labeled image patches and the last two rows correspond to negative labeled image patches. The black points represent cell nuclei. Generated samples capture some characteristics of the corresponding training data. For example, some of them have more than one nuclei.

B.2. We apply a weighted Support Vector Machine (SVM) with an exponentiated quadratic kernel on the latent space from the BGPLVM and DGPLVM. The weighted SVM uses the values of labels to automatically adjust weights inversely proportional to class frequencies. We tuned hyper-parameters using grid search for SVM on a held out set from the training data. We optimized until the objective function converges. The results are shown in Table 3.1. Note that BGPLVM requires an additional classification model to be learned. This does not provide probabilities over the classes other than in the ad-hoc manner that an SVM will. Similarly, DGPLVM does learn a space that reflects the class information, but it does not provide means to get the posterior over the classes. Our model is the only one of the three that learns the classification jointly with the model, and provides a principled way of getting probabilities over the classes.

Based on the results in Table 3.1, BGPLVM and DGPLVM perform better on recall, however our model has better precision and overall F1 score. In imbalanced problems, the goal is to improve recall without losing precision. However, improving precision results in a cost to recall, since increasing the true positives for the minority class results in increasing false positives and so reduced precision [He and Ma, 2013]. It is often better to calculate an

F1-score that weights precision and recall equally [He and Ma, 2013], and SCLVM has a better balance between precision and recall so has better F1-score in total.

Table 3.1 Classification performance. The mean and standard deviation from ten test sets are shown.

	SCLVM	BGPLVM (SVM)	DGPLVM (SVM)
precision	0.426 ± 0.024	0.306 ± 0.013	0.242 ± 0.008
recall	0.555 ± 0.007	0.827 ± 0.000	1.000 ± 0.000
F1-score	0.482 ± 0.015	0.447 ± 0.014	0.390 ± 0.011

3.4 Conclusion

Learning imbalanced data is challenging because, when given imbalanced data, the current model is often dominated by the major category and ignores the categories with small amounts of data. We presented a probabilistic latent variable model that can cope with imbalanced data. We developed a kernel that separates the latent space into a shared space and a private space. One latent space is shared across all the data, while another latent space is used to model each category. Labels are used to encourage a specific structure of the latent space. An efficient variational inference method is proposed by deriving a closed form lower bound of marginal likelihood. The performance of our model is demonstrated with an imbalanced medical image dataset.

Chapter 4

Gaussian processes using active learning for scarce data problems

Classification is a supervised task and uses labeled training data to create a model that aims to classify unseen examples. Here, we restrict our attention to data from the image domain. Such image datasets are often considered to be *big data*, very high dimensional and noisy.

In this chapter[1] we introduce a general workflow for problems where we have a lack of labeled data. We apply this framework to a real-life and challenging problem in the electrofusion industry for detecting clamp versus non-clamp classes for water and gas pipes. As mentioned in Chapter 1, the electrofusion industry has a growing problem of polyethlylene pipeline failure. It has been shown that 99% of joint failures were caused by poor practices before the welding process. We have worked closely with ControlPoint, a quality assurance company based in England which remotely identifies faults and poor installations in water and gas pipelines. Usage of clamps is one of the important factors in distinguishing a good installation from a poor one. Clamps are used to correctly align and stabilize pipes and joints during the fusion process. In the absence of a clamp, there can be a risk of misalignment of pipes within the joint. This can cause gaps inside the joint and potential leakages. To ensure the quality of joints, human experts check thousands of images daily and manually label and score them based on the guidelines of good joint installation.

Labeling images is a tedious job and some datasets can be very expensive to label. In most cases, we have lots of data with no labels, and even if we have labels they are high-level rather than low-level labels. An image has a low-level label when it has pixel level labeling — the process of applying these labels is also called image segmentation. On the other hand,

[1] This chapter is based on "Active Learning Using Gaussian Processes for Imbalanced Datasets", which was presented at the Neural Information Processing Systems (NeurIPS) workshop track on Bayesian NonParametrics (BNP), 2018 - a paper where I appear as a first author.

images with high-level labels do not have a pixel level label, but rather a global label. For example, in the water and gas joints' case, carefully labeling images is not possible due to the high volume of images created daily, and instead we have a global label for the clamp vs non-clamp class.

The dataset is from a quality control company based in Chesterfield.[2] A few samples of clamp images are presented in Figure 4.1. There are various clamp types that are used within the industry. These images come in a variety of colours, sizes, angles and lighting. The dataset has high-level labeling and does not provide any pixel-level information about clamps.

We have so far considered GP models which address the imbalanced data problem. Here we take this one step further and consider a GP-based method which allows active label acquisition, referred to as 'active learning. This technique serves as a complement to the discussion in Chapter 3. In the case of clamp classification, the data of interest is scarce, since 99% of images show a clamp, and since there is no low-level annotation, one possible solution would be to apply *active learning*. In handling large amounts of imbalanced data with high-level labels, our goal will be to label less data to get the low-level labels and select data in a sensible manner. In this approach, the model decides which data would be the most informative, and asks a human expert/oracle to label the new data. Furthermore, we will combine active learning with GP based algorithms to build more sophisticated models in order to handle imbalanced and expensive to label datasets.

In active learning techniques, the process starts with a small set of data and proceeds by actively adding data from the pool of unlabeled data, labeling them and adding them to the training set. This process usually continues until the preferred stopping criteria is met. In Section 4.1, we provide a review of the literature surrounding active learning.

4.1 Active learning

Active learning improves machine learning systems by asking good questions [Settles, 2012]. The key idea behind active learning is that a model can perform well with less training data when it is allowed to choose the data from which it will learn [Settles, 2012]. This is motivated by many real-world applications where annotating labels is difficult, expensive, or time consuming.

The choice of new data points to be labeled is dependent on an acquisition function and the informativeness of the data points. There are various strategies for choosing the acquisition function. Acquisition functions include uncertainty sampling [Lewis and Catlett,

[2] https://www.controlpoint.co.uk/

Fig. 4.1 Some examples of different styles of clamp in water and gas pipe images. (a) Ring clamps. (b) Metal pinch clamps. (c) Plastic pinch clamps. (d) No clamp.

1994; Settles, 2012], query by committee [Iglesias et al., 2011; Seung et al., 1992], expected error or variance minimization [Roy and McCallum, 2001], information theory [Houlsby et al., 2011; Lawrence et al., 2003; MacKay, 1992], and decision theory [Kapoor et al., 2007].

Uncertainty sampling [Lewis and Catlett, 1994; Settles, 2012] is one of the most common, simple, and computationally efficient active learning techniques [Konyushkova et al., 2017].

The basic idea behind this technique is that the instances that the model is confident about can be ignored, while the attention is focused instead on the instances that the model finds the most confusing [Settles, 2012]. This can be a distance from a boundary for non-probabilistic methods such as SVM's margin [Tong and Koller, 2001], or in the case of probabilistic binary classifiers, the instances that are close to the decision boundary are the most confusing or the least confident.

Active learning has proven a useful method for handling a huge amount of data by actively choosing the next instance that the current classifier is the most uncertain about and labeling those in particular [Ertekin et al., 2007a,b; Provost, 2000]. One common method of making queries in active learning is 'pool-based' sampling. In the pool-based [Lewis and Gale, 1994] uncertainty sampling technique, the basic idea is to have a pool of unlabeled data, a set of initial labeled instances, and a learner. In this setting, the classifier has access to the pool of unlabeled data from which to choose the next sample.

For uncertainty sampling, besides the 'least confident' technique that queries the instance whose predicted output it is least confident in, entropy based uncertainty sampling is also very common. Entropy is a way to measure a variable's average information content. Entropy is more appropriate if our objective is to minimize the log-loss, while the least confident method is more appropriate if we aim to reduce classification error, because such a model can select instances that are more helpful to assign them to different classes [Settles, 2012].

Active learning also has similarities with semi-supervised learning [Zhu and Goldberg, 2009]. Both techniques aim to make the most of unlabeled data. A very basic semi-supervised learning technique is self-training [Yarowsky, 1995]. Similarly to the active learning approach, the learner is initially trained with a small labeled set. It then classifies the unlabeled data to re-train itself. This is usually an iterative approach, where the most confident unlabeled instances with their predicted labels are added to the training set. In a sense, then, this technique is the opposite of the active learning uncertainty sampling technique, where the least confident samples are selected instead.

Non-probabilistic active learning is usually performed using SVM [Ertekin et al., 2007a,b; Tong and Koller, 2001]. In active learning with SVM, intuitively the closest instance to the margin would offer the most information and the model will be least certain about this instance. In probabilistic techniques, within the GP framework, there are some works based on information gain in Bayesian methodology, such as Houlsby et al. [2011]; Lawrence et al. [2003]; MacKay [1992]. Informative Vector Machines (IVM) [Lawrence et al., 2003] are designed specifically for GPs in which you have to calculate the information gain from all of the data and pick the one that gives you the highest information. Houlsby et al. [2011] proposes an approach using information gain in terms of predictive entropies.

For the imbalanced data problem, active learning might not offer any help on its own. A few works tackle the imbalanced problem using active learning such as Attenberg and Provost [2010]; Ertekin et al. [2007a]; Wallace et al. [2010], to name a few. Attenberg and Provost [2010] tackles the imbalanced problem by proposing 'guided learning', which allows an oracle to search for a specific class. Wallace et al. [2010] built two models, one with fine granularity and the other with coarse granularity, and used query by disagreement between these two models to perform active learning on the new documents for querying. This technique overcomes the imbalanced problem better compared to many standard active learning techniques. Ertekin et al. [2007a] proposed active learning based on SVM to tackle the imbalanced data problem. In this method, they first choose $\mathcal{K}$ random samples from the pool, after which the instance that is closest to the margin is selected. This new instance is added to the training set, and the model is re-trained. This method is efficient when the dataset is very large, so there is no need to search the entire pool. They also show that the data within the overlapping margin between two classes is less imbalanced.

The different strategies for active learning have their own advantages and disadvantages. For example, the uncertainty sampling approach is simple and fast, easy to implement, and it can be used alongside probabilistic models [Settles, 2012]. However, there is a risk that the corresponding models become over confident about incorrect predictions [Settles, 2012].

The stopping criteria for active learning is when the accuracy has reached a plateau and labeling more data does not help to improve the accuracy, and so labeling more data is a waste of resources and time [Settles, 2012].

In this work we will use GPs as a probabilistic method to combine with the pool based least confident active learning technique. We review GP classification in the next section, and in Section 4.4 we will explain how we combine an active learning workflow with the GP classification.

4.2 Gaussian process classification

Gaussian process classification was introduced in detail in Section 2.2.3. As mentioned, when target values are discrete, we have a classification problem. In classification cases, since the likelihood is not Gaussian, we cannot calculate the posterior analytically. Instead we need to approximate the posterior. The Gaussian process prior f and likelihood are defined as:

$$p(f|\mathbf{x},\boldsymbol{\theta}) = \mathcal{N}(f|0,\, k_{ff})$$
$$p(y|\mathbf{x},f) = Bernoulli(\sigma(f(\mathbf{x}))).$$

We consider the logistic function σ where, given the value of f, the target value y takes the Bernoulli distribution with probability $\sigma(f(\mathbf{x}))$. The non-Gaussian likelihood σ introduces complications for calculating the posterior distribution, and we need to approximate it. In this chapter, we use Laplace approximation, which was explained in more detail in Section 2.2.4.1. The latent function f was originally between $(-\infty, +\infty)$, but since our target values are now binary and are, for example, between $(0,1)$, the logistic function will transform it; we call the new function $\pi_n \in [0,1]$. The mean and the covariance of the approximate posterior is:

$$q(\mathbf{f}|\mathbf{X},\mathbf{y}) = \mathcal{N}(\hat{\mathbf{f}}, (\mathcal{W} + \mathbf{K}_{ff}^{-1})^{-1}), \tag{4.1}$$

where, $\hat{\mathbf{f}} = \mathbf{K}_{ff}(\frac{\delta \log(\mathbf{y}|\hat{\mathbf{f}},\mathbf{X})}{\delta \hat{\mathbf{f}}})$ and $\mathcal{W} = -\frac{\delta^2 \log(\mathbf{y}|\hat{\mathbf{f}},\mathbf{X})}{\delta \mathbf{f}^2}$. These have to be found iteratively as they cannot be calculated in a closed form. More mathematical details can be found at Chapter 2.

GPs learn the structure of data using a covariance function. With kernels such as RBF, we can use the information in a dataset to learn about length-scale and variance hyper-parameters. The covariance function will tell us how quickly the correlation in our data changes with distance in the input space. To fully use the information, we therefore need to build kernels with great representation power. This will be achieved using Convolutional Neural Network (CNN) [Wilson et al., 2016b]. GPs achieve better performance when using features from CNN rather than raw pixels [Wilson et al., 2016a]. Furthermore, for handling the data scarcity problem, a combination of transfer learning from CNN with GP classification is proposed. In the next section we will briefly introduce CNNs.

4.3 Convolutional neural networks (CNN)

CNNs are widely used in computer vision tasks. These networks are composed of different layers: an input layer, an output layer, and several hidden layers, where some of these hidden layers are convolutional. Convolutional layers learn local patterns in small windows of two dimensions, such as visual futures like edges and lines. This is the main difference between convolutional layers and fully connected layers, because fully connected layers are useful for learning global patterns rather than local patterns. Convolutional layers have two hyper-parameters: filter size (also called kernel) and stride. Smaller filter sizes encourage the use of more local information, while larger filters allow the use of more global information. The number of steps the filter window moves on the input image is called stride. In different convolutional layers, using a variety of filters and strides allows for detecting specific characteristics or features in the image.

Convolutional layers are usually followed by pooling layers, which are a form of non-linear down-sampling method. Pooling layers provide condensed information from convolution layers, and so reduce the dimensionality on intermediate representation. There are a variety of pooling techniques, such as max-pooling and average-pooling, where max-pooling gets the maximum value of the window and average-pooling gets the average value of the window.

Image data are by nature usually very complex, and manual feature extraction is no longer needed with CNNs, as they use a different combination of filters and pooling which helps to have translation invariant features. There are various architectures such as LeNet [LeCun et al., 1998], AlexNet [Krizhevsky et al., 2012], VGGNet [Simonyan and Zisserman, 2014], and GoogLeNet (also known as Inception) [Szegedy et al., 2015]. We use InceptionV3 architecture [Szegedy et al., 2016] as the feature extractor with Imagenet pre-trained weights [Deng et al., 2009]. An InceptionV3 diagram is illustrated in Figure 4.2.

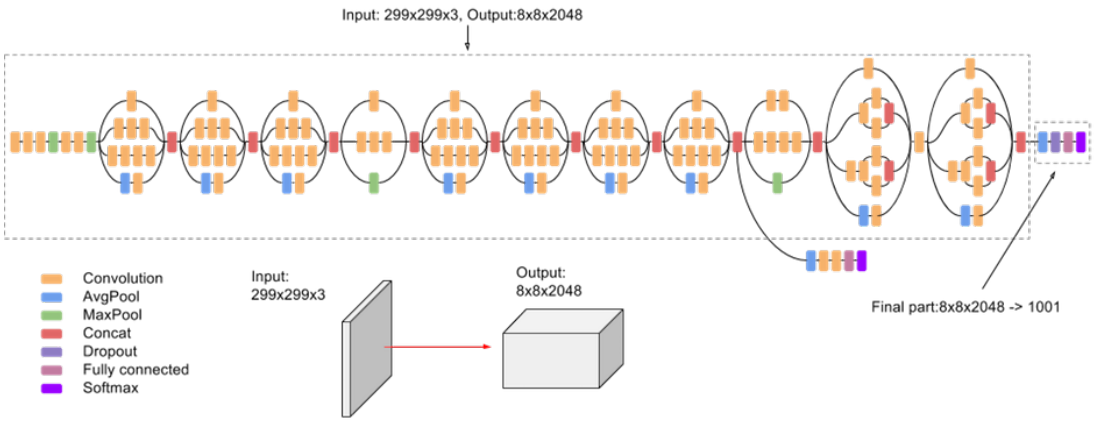


Fig. 4.2 An illustration of the high-level diagram of the model, taken from "cloud.google.com/tpu/docs/inception-v3-advanced".

In the next section, we will explain how we combine an active learning workflow with GP classification and CNN.

4.4 Active learning workflow

The active learning workflow that we are proposing is general and can be applied to any image problem that lacks labeled data or where labeling is very expensive. In such cases the model needs to learn the data in the most efficient way possible to reduce costs, for example. The active learning workflow is presented in Figure 4.3. Images are available in a large pool. We will use this pool to sample data for labeling and training purposes. Often there is a need

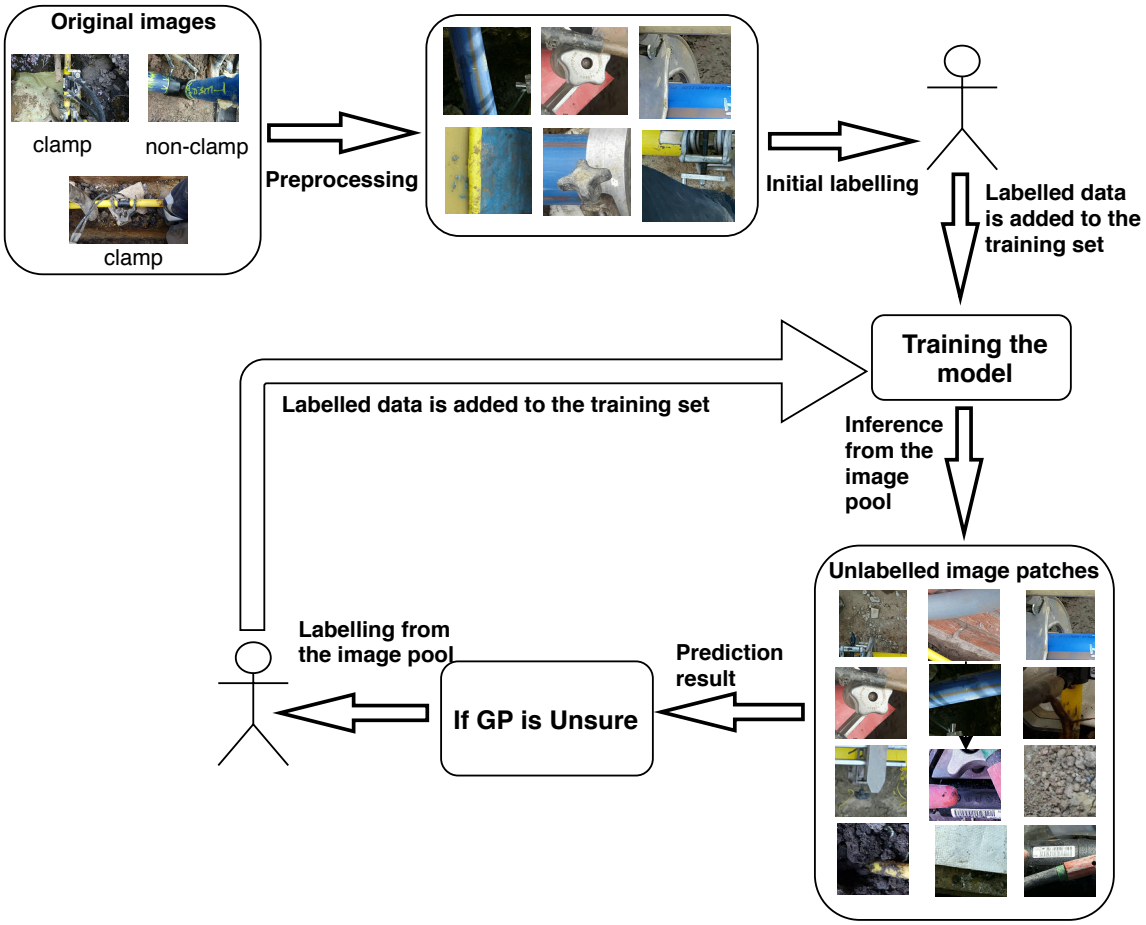


Fig. 4.3 The active learning workflow.

for pre-processing images. For example, images may differ in size or we might need to have a different granularity for labeling, to name a few. In the former case, we might need to resize images to a fixed size, and in the latter scenario we might need to cut images into small patches, depending on the size of the region of interest (ROI). The ROI might capture a very small section of the entire image and we are interested to label the ROI. Small patches should roughly capture the ROI. A small number of image patches are sent to human expert/oracle to label. We call this initial labeling. Initially labeled data will be used to train the model for the first time. To train the GP classifier we use features extracted using CNN from image patches. For feature extraction using CNNs, we need to resize images to have a fixed image size as input. For example, InceptionV3 needs images to be 299×299 pixels. The second step is that the model will select $\mathcal{K}$ samples from the pool of unlabeled image patches. Based on the prediction result of the model, the least confident samples are sent to an oracle for labeling. This process is continued until a stopping criteria is met.

In the next section, we will apply the active learning workflow to the water and gas pipe images. As mentioned earlier in this chapter, we have image-level labels for this dataset and we aim at classifying clamp versus non-clamp images.

4.5 Experiments

In this section, we demonstrate our model with a real-world dataset — our aim is to tackle the imbalanced data problem as it relates to detecting clamps. It is essential that clamps are used when making electrofusion joints to ensure that the pipe and joint are correctly aligned and stabilized during the fusion process. In the absence of clamps, there is a risk that pipes may be misaligned within the joint, which raises the possibility of gaps occurring and is a potential source of leakage. Uncontrolled movement of the joint during welding may also occur in the absence of clamps, which is likely to result in a loss of intimate contact of the welded areas and therefore a poorer weld. The flowchart of this work is represented in Figures 4.3 and 4.4. The images that are used are from an operational environment; each is taken from one of a few different angles with different phones under various lighting and weather conditions. The size of the images also varies. We first pre-process the images by fixing their size to 1280×720. A few of the original images are presented in Figure 4.1. The original images are pre-processed and cut into small patches. From this image pool, a small number of patches are labeled by an expert and an initial model is trained. In all our trainings we use 3000 iterations with a Stochastic Gradient Descent (SGD) algorithm for optimization. We use the RBF kernel with 20 dimensions. Then from the image patches' pool a few images are selected to test against the current model. If the model is unsure about

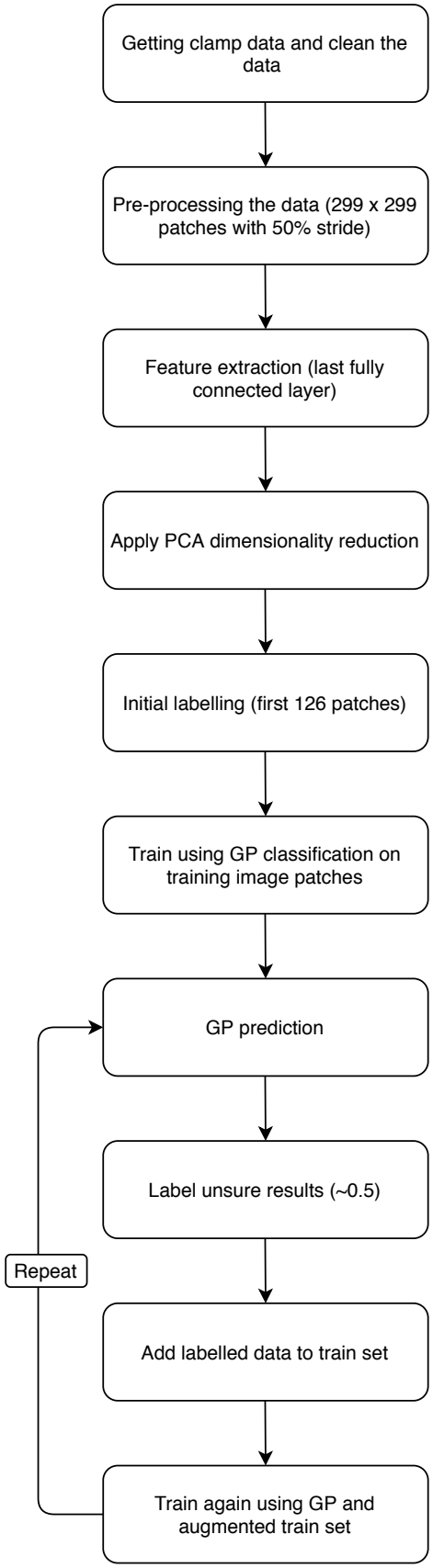


Fig. 4.4 The workflow of the whole process.

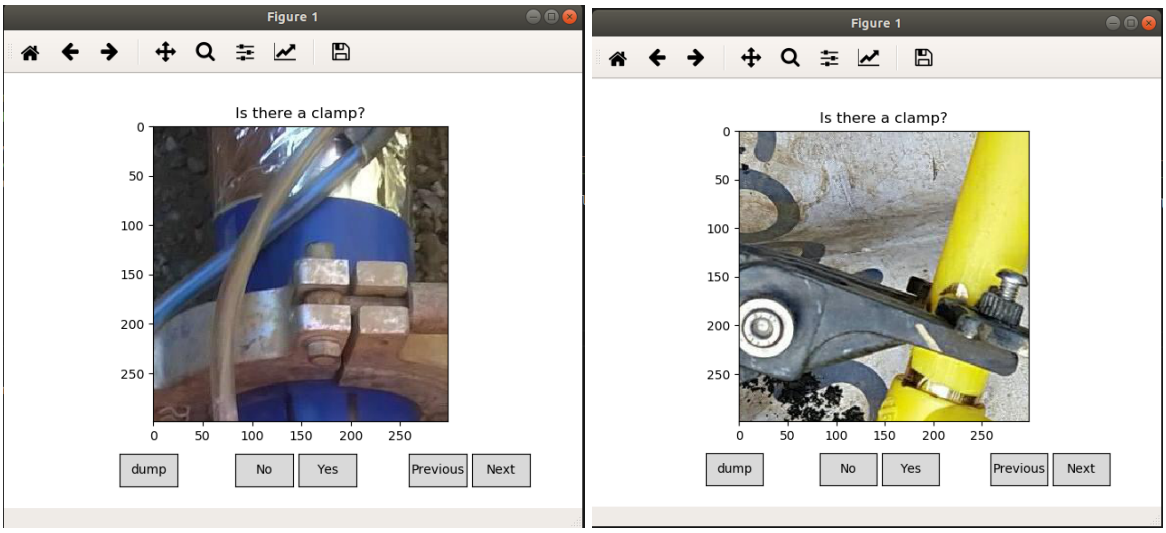


Fig. 4.5 Examples of the user interface of the software developed using Python for the labeling process. The dump button is for saving the labeled images.

any image patch, that patch will be labeled by the expert and added to the training set. This process is continued until the stopping criteria is met (no improvement in F-score).

Because of the imbalanced nature of this problem, and since we do not have any low-level labels, we are trying to model this problem in the most efficient way using active learning. This involves cutting the images into small patches (299×299) with 50% stride so that we can have a mixture of clamp and non-clamp image patches.

We used a pre-trained neural network as the feature extractor. We also used the features of the last fully connected layer from the InceptionV3 [Szegedy et al., 2016] architecture with Imagenet [Deng et al., 2009] pre-trained weights. These features have a dimensionality of $1 \times 1 \times 2048$, which were projected down to 20 dimensions using PCA [Tipping and Bishop, 1999a] which was introduced in Section 2.2.5. We chose 20 dimensions because the eigenvalues are negligible for the remaining dimensions. The result of the PCA is fed into the GP classification as input.

For the experiments, 300 images were used for training and validation purposes. Additionally, four different independent sets, each containing 100 images, were used as the test set, in order to calculate the standard deviation. The ratio of non-clamp to clamp images in the training and test sets was 1:4. Each image consisted of 21 patches.

For initialization, we first labeled 126 image patches (six images in total) using the software developed. An example of the software user interface is illustrated in Figure 4.5. These 126 image patches were split into the training and validation set at the patch level. GP classification was then used to train the model using the patch images. New image patches were added randomly from the pool of image patches, for prediction using GP classification.

The images that the GP is unsure about (0.5 ± 0.05) were added to the labeling system. The new unsure patches were labeled, and this labeled data was added to the training set for further training. This process was repeated until stopping criteria were met and there was not much improvement in the F1-score of the validation set. For the final result, GP prediction was used to predict the patches from four test sets. Only 433 image patches were labeled as the training set and 101 were labeled as the validation set. The total number of image patches that we used for the training and validation was equal to around 25 images, which shows the efficiency of the GP classification using active learning.

Since our data is imbalanced, and in this experiment we have a training set of only 300 images, training a competitive baseline of neural networks from scratch was not feasible. Instead, a pre-trained network was used in a transfer learning scenario. InceptionV3 pre-trained network using ImageNet images were used as the base model. The 300 training images that we had were augmented by a factor of 10 (rotation, mirroring, zoom, translation) and used to fine-tune the parameters of the model. During training, only parameters of the fully connected layers were updated — for all other layers, the parameters were frozen and were not updated during the error back propagation to avoid over-fitting.
Random sampling was used to compare against GP classification's selection criteria. The same amount of patches that were used for GP classification were randomly selected and added to the training set. The model was trained using these randomly selected patches.
To compare GP to another probabilistic method, we used Logistic Regression (LR). Like the GP method, in the LR case, image patches were added randomly from the pool in order to make predictions using LR, and the images that LR was unsure about (0.5 ± 0.05), were added to the labeling system. The new uncertain patches were labelled, and the labelled data was added to the training set for further training. This process was continued until a similar number of image patches that we used for our GP classification model were acquired.
In Table 4.1, we calculate precision, recall and F1-score. Our active learning technique is much more efficient both in terms of the quantity of data that is used and speed. It surpasses CNN, random patch selection, and LR in terms of assessment. Because the data is imbalanced and because the size of data is not big enough, CNN gets a very low F1-score when compared to other techniques. In random patch selection, we use the features extracted from CNN. We then train the random patches with a GP classifier that is not as data hungry as CNN, so we get better results in terms of recall and F1-score compared to CNN.

In this work we are interested in capturing all of the cases that are in fact positive (non-clamps) and are identified as belonging to the positive class. This is the definition of recall, and even though GP is comparable with baseline CNN in terms of precision, it surpasses the other techniques in term of recall and F1-score.

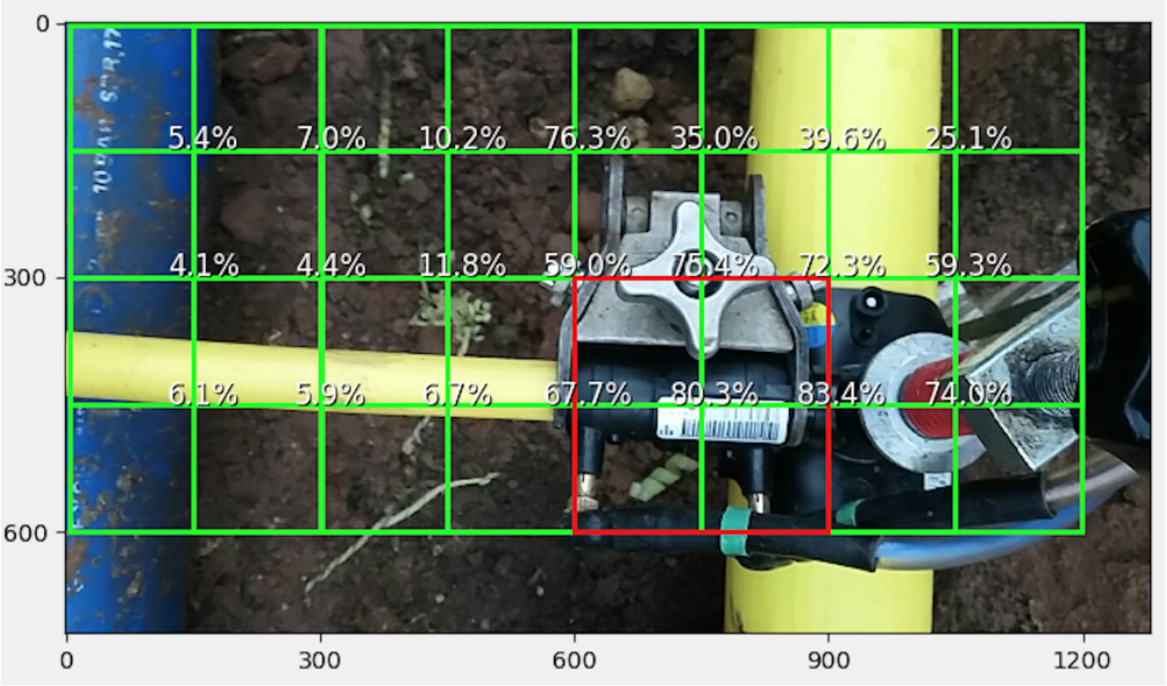


Fig. 4.6 Sample of predicted image patches. The big image is cut into small patches (299×299) with 50% stride. The patches are represented by green boxes. The parts of the image that are smaller than 299×299 are skipped. The numbers in the middle of the green boxes represent the prediction for that specific patch. The patches that have prediction closer to 100% are classified as clamps and the ones closer to 0% are classified as non-clamps.

We present a sample from our test set, by putting together all of the patches from one image and representing the probability of each patch belonging to the clamp or non-clamp classes in Figure 4.6. The threshold for the classifier is 50%.

Table 4.1 Results based on Precision, Recall and F1-score, with mean and standard deviation from four test sets.

Method	Precision	Recall	F1-score
Baseline CNN	**0.93** $\pm$ 0.09	0.41 $\pm$ 0.04	0.56 $\pm$ 0.04
LR	0.67 $\pm$ 0.005	0.70 $\pm$ 0.11	0.68 $\pm$ 0.06
Random patch	0.55 $\pm$ 0.08	0.63 $\pm$ 0.05	0.59 $\pm$ 0.06
CNN+GP classification	0.81 $\pm$ 0.08	**0.76** $\pm$ 0.08	**0.78** $\pm$ 0.03

4.6 Conclusion

The class imbalance problem is a challenging problem. Using high dimensional image data adds further complexity. In this chapter, active learning combined with GP classification was used as a selection criterion for the next round of labeling. Starting with a very small training set, additional samples were selected randomly from the pool of unlabeled data. These samples were labeled and the ones that the classifier is unsure about were added to the training set. The proposed technique requires fewer expert labels and as a result reduces the cost of labeling. We showed that our active learning method is able to use fewer data as compared to other supervised learning approaches. We were able to detect the minority class with an F1-score of 0.78 ± 0.03. This technique is applicable to a wide range of data, especially where there is a lack of labeled data or where labeling the data is very expensive since it is important to have good models with small labeled samples.

Chapter 5

Multi-task learning for aggregated data

Many datasets in fields such as ecology, epidemiology, remote sensing, sensor networks, and demography appear naturally aggregated; that is, variables in these datasets are measured or collected in intervals, areas, or supports of different shapes and sizes. For example, census data are usually sampled or collected in an aggregated form at different administrative divisions, e.g. borough, town, postcode, or city levels. Likewise, in sensor networks, correlated variables are measured at different resolutions or scales. In the near future, air pollution monitoring across cities and regions could be done using a combination of a few high-quality low time-resolution sensors and several low-quality (low-cost) high time-resolution sensors. Aggregating such data usually results in data scarcity issues. However, joint modelling of the variables registered in the census data or the variables measured using different sensor configurations at different scales can improve predictions of otherwise scarce data at the point or support levels.

In this chapter[1], we are interested in providing a general framework for multi-task learning on these types of datasets. Our motivation is to use multi-task learning to jointly learn models for different tasks, where each task is (potentially) defined at a different support of any shape or size, and (potentially) has a different nature, i.e. it is a continuous, binary, categorical, or count variable. We appeal to the flexibility of Gaussian processes (GPs) for developing a prior over this type of datasets, and provide a scalable approach for variational Bayesian inference.

Gaussian processes have previously been used for aggregated data [Law et al., 2018; Smith et al., 2018; Tanaka et al., 2019a], as well as in the related field of *multiple instance learning* [Haußmann et al., 2017; Kandemir et al., 2016; Kim and De la Torre, 2010]. In

[1]This chapter is based on "Multi-task Learning for Aggregated Data using Gaussian Processes" that was published at Advances in Neural Information Processing Systems (NeurIPS), 2019 - a paper where I appear as a first author.

multiple instance learning, each instance in the dataset consists of a set (or *bag*) of inputs, with only one output (or label) for that whole set, and the aim is to provide predictions at the level of individual inputs. Smith et al. [2018] provide a new kernel function to handle single regression tasks defined at different supports, using cross-validation for hyper-parameter selection. Law et al. [2018] use the weighted sum of a latent function, evaluated at different inputs, as the prior for the rate of a Poisson likelihood. The latent function follows a GP prior. The authors use stochastic variational inference (SVI) for approximating the posterior distributions. Tanaka et al. [2019a] mainly use GPs for creating data from different auxiliary sources. Furthermore, they only consider Gaussian regression and do not include inducing variables. While Smith et al. [2018] and Law et al. [2018] perform the aggregation at the latent prior stage, Kandemir et al. [2016]; Kim and De la Torre [2010] and Haußmann et al. [2017] perform the aggregation at the likelihood level. These three approaches target a binary classification problem. Both Kim and De la Torre [2010] and Haußmann et al. [2017] focus on the case in which the label of the bag corresponds to the maximum of the (unobserved) individual labels of each input. Kim and De la Torre [2010] approximate the maximum using a softmax function, computed using a latent GP prior, evaluated across the individual elements of the bag. They use Laplace approximation to compute the approximated posterior [Rasmussen and Williams, 2006b]. Haußmann et al. [2017], on the other hand, approximate the maximum using the so called *bag label likelihood*, introduced by the authors, which is similar to a Bernoulli likelihood, with soft labels given by a convex combination between the bag labels and the maximum of the (latent) individual labels. The latent individual labels in turn follow Bernoulli likelihoods, with parameters given by a GP. The authors provide a variational bound and include inducing inputs for scalable Bayesian inference. Kandemir et al. [2016] follow a similar approach to Law et al. [2018], equivalent to setting all the weights in Law et al.'s model to one. The sum is then used to modulate the parameter of a Bernoulli likelihood that models the bag labels. They use a Fully Independent Training Conditional approximation for the latent GP prior [Snelson and Ghahramani, 2006b].

In contrast to these works, we provide a multi-task learning model for aggregated data that scales to large datasets and allows for heterogeneous outputs. The idea of using multi-task learning for aggregated datasets was simultaneously proposed by Hamelijnck et al. [2019] and Tanaka et al. [2019b]; both are additional models to the one we propose in this chapter. In our work, we allow heterogenous likelihoods, in contrast with both Hamelijnck et al. [2019] and Tanaka et al. [2019b]. We also allow an exact solution to the integration of the latent function through the kernel in Smith et al. [2018], where Hamelijnck et al. [2019] does not. Furthermore, with respect to computational complexity, inducing inputs are used, which

also distinguishes our work from the work in Tanaka et al. [2019b]. Other relevant work is described in Section 5.2.

In building the multi-task learning model we appeal to the linear model of coregionalisation [Goovaerts, 1997; Journel and Huijbregts, 1978] which has gained popularity in the multi-task GP literature in recent years [Alvarez et al., 2012; Bonilla et al., 2008]. We also allow different likelihood functions [Moreno-Muñoz et al., 2018] and different input supports per individual task. Moreover, we introduce inducing inputs at the level of the underlying common set of latent functions, an idea initially proposed in Alvarez and Lawrence [2009]. We then use stochastic variational inference for GPs [Hensman et al., 2013], leading to an approximation similar to the one obtained in Moreno-Muñoz et al. [2018]. Empirical results show that the multi-task learning approach developed here provides accurate predictions in a number of challenging datasets where tasks have different supports.

5.1 Multi-task Gaussian process

Multi-task Gaussian processes deal with cases where there are multiple outputs available. Many applications of machine learning need to solve problems where there are multiple cases with dependencies between them. The challenge is usually that the task that we are interested in has little or no data but there are other tasks available that can possibly help to predict the task of interest. The simplest way of solving these kind of problems is to use a single model for each task, which ignores the correlation between tasks and makes predictions for each output individually. However, it has been proven that the joint prediction of these outputs, and using the interaction between them, improves individual predictions [Alvarez et al., 2012]. Within the machine learning community this type of modelling is generally referred to as multi-task learning. The core idea is that sharing information between tasks will result in better predictions than learning each task individually Alvarez et al. [2012]. In classical supervised single-task models there exist N input-output pairs $(\mathbf{X}, \mathbf{y})$, where each input $\mathbf{x}$ is a vector and each output y is a scalar. In multi-task learning each output is a vector. And, as mentioned, in multi-task learning the assumption is that the tasks are related to each other and each component has different inputs.

5.1.1 Multi-task learning for aggregated data at different scales

In this section we first define the basic model in the single-task setting. We then extend the model to the multi-task setting, and finally provide details for the stochastic variational formulation for approximate Bayesian inference.

Change of support using Gaussian processes

Change of support has previously been studied in geostatistics [Gotway and Young, 2002]. In this work, we use a formulation similar to Kyriakidis [2004]. We start by defining a stochastic process over the input interval (x_a, x_b) using

$$f(x_a, x_b) = \frac{1}{\Delta_x} \int_{x_a}^{x_b} u(z) dz,$$

where $u(z)$ is a latent stochastic process that we assume follows a Gaussian process with zero mean and covariance $k(z, z')$ and $\Delta_x = |x_b - x_a|$. Dividing by Δ_x helps to maintain proportionality between the length of the interval and the area under $u(z)$ in the interval. In other words, the process $f(\cdot)$ is modelled as a density, meaning that inputs with widely differing supports will behave in similar ways. The first two moments for $f(x_a, x_b)$ are given as:

$$\mathbb{E}[f(x_a, x_b)] = 0 \tag{5.1}$$

$$\mathbb{E}[f(x_a, x_b), f(x'_a, x'_b)] = \frac{1}{\Delta_x \Delta_{x'}} \int_{x_a}^{x_b} \int_{x'_a}^{x'_b} \mathbb{E}[u(z)u(z')] dz' dz. \tag{5.2}$$

The covariance for $f(x_a, x_b)$ follows as:

$$\text{cov}[f(x_a, x_b), f(x'_a, x'_b)] = \frac{1}{\Delta_x \Delta_{x'}} \int_{x_a}^{x_b} \int_{x'_a}^{x'_b} k(z, z') dz' dz, \tag{5.3}$$

since $\mathbb{E}[u(z)] = 0$. Let us use $k(x_a, x_b, x'_a, x'_b)$ to refer to $\text{cov}[f(x_a, x_b), f(x'_a, x'_b)]$. We can now use these mean and covariance functions for representing the Gaussian process prior:

$$f(x_a, x_b) \sim \mathcal{GP}(0,\ k(x_a, x_b, x'_a, x'_b)). \tag{5.4}$$

For some forms of $k(z, z')$ it is possible to obtain an analytical expression for $k(x_a, x_b, x'_a, x'_b)$. For example, if $k(z, z')$ follows an Exponentiated-Quadratic (EQ) covariance form:

$$k(z, z') = \sigma^2 \exp\{-\frac{(z - z')^2}{\ell^2}\}, \tag{5.5}$$

where σ^2 is the variance of the kernel and ℓ is the length-scale, it can be shown that $k(x_a, x_b, x'_a, x'_b)$ follows as:

$$k(x_a, x_b, x'_a, x'_b) = \frac{\sigma^2 \ell^2}{2\Delta_x \Delta_{x'}} \left[h\left(\frac{x_b - x'_a}{\ell}\right) + h\left(\frac{x_a - x'_b}{\ell}\right) - h\left(\frac{x_a - x'_a}{\ell}\right) - h\left(\frac{x_b - x'_b}{\ell}\right) \right],$$

where $h(z) = \sqrt{\pi} z \text{erf}(z) + e^{-z^2}$ with $\text{erf}(z)$, the error function, defined as $\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-r^2} dr$. Other kernels for $k(z, z')$ also lead to analytical expressions for $k(x_a, x_b, x'_a, x'_b)$. See for example Smith et al. [2018]. Detailed formulas regarding integrals and derivations are derived and available in Appendix C.1.

So far, we have restricted the exposition to one-dimensional intervals. However, we can define the stochastic process f over a general support υ, with measure $|\upsilon|$, using

$$f(\upsilon) = \frac{1}{|\upsilon|} \int_{\mathbf{z} \in v} u(\mathbf{z}) d\mathbf{z}. \tag{5.6}$$

The support υ generally refers to an area or volume of any shape or size. Following similar assumptions to the ones we used for $f(x_a, x_b)$, we can build a GP prior to represent $f(\upsilon)$ with covariance $k(\upsilon, \upsilon')$ defined as:

$$k(\upsilon, \upsilon') = \frac{1}{|\upsilon||\upsilon'|} \int_{\mathbf{z} \in \upsilon} \int_{\mathbf{z}' \in \upsilon'} k(\mathbf{z}, \mathbf{z}') d\mathbf{z}' d\mathbf{z}. \tag{5.7}$$

Let $\mathbf{z} \in \mathcal{R}^p$. If the support υ has a regular shape, e.g. a hyper-rectangle, then assumptions on $u(\mathbf{z})$ such as additivity or factorisation across input dimensions will lead to kernels that can be expressed as an addition of kernels or a product of kernels acting over a single dimension. For example, let $u(\mathbf{z}) = \prod_{i=1}^p u_i(z_i)$, where $\mathbf{z} = [z_1, \cdots, z_p]^\top$, and a GP over each $u_i(z_i) \sim \mathcal{GP}(0, k(z_i, z'_i))$. If each $k(z_i, z'_i)$ is an EQ kernel, then $k(\upsilon, \upsilon') = \prod_{i=1}^p k(x_{i,a}, x_{i,b}, x'_{i,a}, x'_{i,b})$, where $(x_{i,a}, x_{i,b})$ and $(x'_{i,a}, x'_{i,b})$ are the intervals across each input dimension. If the support υ does not follow a regular shape, i.e it is a polytope, then we can approximate the double integration by numerical integration inside the support.

5.1.2 Multi-task learning setting

Linear model of coregionalization Our inspiration for multi-task learning is the linear model of coregionalisation (LMC) [Journel and Huijbregts, 1978]. This model has connections with other multi-task learning approaches that use kernel methods. For example, Teh et al. [2005] and Bonilla et al. [2008] are particular cases of LMC. A detailed review is available in Alvarez et al. [2012]. In the LMC, each output (or task in our case) is represented

as a linear combination of a common set of latent Gaussian processes. Let $\{u_q(\mathbf{z})\}_{q=1}^{Q}$ be a set of Q GPs with zero means and covariance functions $k_q(\mathbf{z},\mathbf{z}')$. Each GP $u_q(\mathbf{z})$ is sampled independently and identically R_q times to produce $\{u_q^i(\mathbf{z})\}_{i=1,q=1}^{R_q,Q}$ realizations that are used to represent the outputs.

$$f_d(\mathbf{z}) = \sum_{q=1}^{Q}\sum_{i=1}^{R_q} a_{d,q}^i u_q^i(\mathbf{z}). \tag{5.8}$$

The cross-covariance between two functions f_d and $f_{d'}$ is given as:

$$k_{f_d,f_{d'}}(\mathbf{z},\mathbf{z}') = \sum_{q=1}^{Q} b_{d,d'}^q k_q(\mathbf{z},\mathbf{z}'),$$

where $b_{d,d'}^q = \sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i$. The kernel can now be expressed as:

$$\mathbf{K}(\mathbf{z},\mathbf{z}') = \sum_{q=1}^{Q} \mathbf{B}_q k_q(\mathbf{z},\mathbf{z}'),$$

where each $\mathbf{B}_q \in \mathbb{R}^{D\times D}$ is known as a coregionalisation matrix. The matrix $\mathbf{B}_q$ has elements $b_{d,d'}^q$. We will now integrate LMC with our multi-task learning. Let $\{f_d(\upsilon)\}_{d=1}^{D}$ be a set of tasks where each task is defined at a different support υ. We use the set of realizations $u_q^i(\mathbf{z})$ to represent each task $f_d(\upsilon)$ as

$$f_d(\upsilon) = \sum_{q=1}^{Q}\sum_{i=1}^{R_q} \frac{a_{d,q}^i}{|\upsilon|}\int_{\mathbf{z}\in v} u_q^i(\mathbf{z})d\mathbf{z}, \tag{5.9}$$

where the coefficients $a_{d,q}^i$ weight the contribution of each integral term to $f_d(\upsilon)$. Since $\text{cov}[u_q^i(\mathbf{z}), u_{q'}^{i'}(\mathbf{z}')] = k_q(\mathbf{z},\mathbf{z}')\delta_{q,q'}\delta_{i,i'}$, with $\delta_{\alpha,\beta}$ the Kronecker delta between α and β, the cross-covariance $k_{f_d,f_{d'}}(\upsilon,\upsilon')$ between $f_d(\upsilon)$ and $f_{d'}(\upsilon')$ is then given as

$$k_{f_d,f_{d'}}(\upsilon,\upsilon') = \sum_{q=1}^{Q} \frac{b_{d,d'}^q}{|\upsilon||\upsilon'|}\int_{\mathbf{z}\in\upsilon}\int_{\mathbf{z}'\in\upsilon'} k_q(\mathbf{z},\mathbf{z}')d\mathbf{z}'d\mathbf{z},$$

where $b_{d,d'}^q = \sum_{i=1}^{R_q} a_{d,q}^i a_{d',q}^i$. Following the discussion in Section 5.1.1, the double integral can be solved analytically for some options of υ, υ' and $k_q(\mathbf{z},\mathbf{z}')$, generally a numerical approximation can be obtained.

It is also worth mentioning at this point that the model does not require all tasks to be defined at the area level. Some tasks could also be defined at the point level. Say for example that

f_d is defined at the support level υ, $f_d(\upsilon)$, whereas $f_{d'}$ is defined at the point level, say $\mathbf{x} \in \mathbb{R}^p$, $f_{d'}(\mathbf{x})$. In this case, $f_{d'}(\mathbf{x}) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d',q}^i u_q^i(\mathbf{x})$. We can still compute the cross-covariance between $f_d(\upsilon)$ and $f_{d'}(\mathbf{x})$, $k_{f_d,f_{d'}}(\upsilon,\mathbf{x})$, leading to, $k_{f_d,f_{d'}}(\upsilon,\mathbf{x}) = \sum_{q=1}^{Q} \frac{b_{d,d'}^q}{|\upsilon|} \int_{\mathbf{z}\in\upsilon} k_q(\mathbf{z},\mathbf{x})d\mathbf{z}$. For the case $Q = 1$ and $p = 1$ (i.e. dimensionality of the input space), for $z, z', x \in \mathbb{R}$, $\upsilon = (x_a, x_b)$ and an EQ kernel for $k(z,z')$, we get

$$k_{f_d,f_{d'}}(\upsilon,x) = \frac{b_{d,d'}}{\Delta_x} \int_{x_a}^{x_b} k(z,x)dz = \frac{b_{d,d'}\ell}{2\Delta_x} \left[\operatorname{erf}\left(\frac{x_b - x}{\ell}\right) + \operatorname{erf}\left(\frac{x - x_a}{\ell}\right) \right].$$

We used $\sigma^2 = 1$ to avoid overparameterization for the variance. Again, if υ does not have a regular shape, we can approximate the integral numerically.

Let us define the vector-valued function $\mathbf{f}(\upsilon) = [f_1(\upsilon), \cdots, f_D(\upsilon)]^\top$. A GP prior over $\mathbf{f}(\upsilon)$ can use the kernel defined above so that

$$\mathbf{f}(\upsilon) \sim \mathcal{GP}\left(\mathbf{0}, \sum_{q=1}^{Q} \frac{1}{|\upsilon||\upsilon'|} \mathbf{B}_q \int_{\mathbf{z}\in\upsilon} \int_{\mathbf{z}'\in\upsilon'} k_q(\mathbf{z},\mathbf{z}')d\mathbf{z}'d\mathbf{z}\right),$$

where each $\mathbf{B}_q \in \mathbb{R}^{D\times D}$ is known as a coregionalisation matrix. The scalar term $\int_{\mathbf{z}\in\upsilon} \int_{\mathbf{z}'\in\upsilon'} k_q(\mathbf{z},\mathbf{z}')d\mathbf{z}'d\mathbf{z}$ modulates $\mathbf{B}_q$ as a function of υ and υ'.

The prior above can be used for modulating the parameters of likelihood functions that model the observed data. The most simple case corresponds to the multi-task regression problem, which can be modelled using a multivariate Gaussian distribution. Let $\mathbf{y}(\upsilon) = [y_1(\upsilon), \cdots, y_D(\upsilon)]^\top$ be a random vector modelling the observed data as a function of υ. In the multi-task regression problem $\mathbf{y}(\upsilon) \sim \mathcal{N}(\boldsymbol{\mu}(\upsilon), \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}(\upsilon) = [\mu_1(\upsilon), \cdots, \mu_D(\upsilon)]^\top$ is the mean vector and $\boldsymbol{\Sigma}$ is a diagonal matrix with entries $\{\sigma_{y_d}^2\}_{d=1}^D$. We can use the GP prior $\mathbf{f}(\upsilon)$ as the prior over the mean vector $\boldsymbol{\mu}(\upsilon) \sim \mathbf{f}(\upsilon)$. Since both the likelihood and the prior are Gaussian, both the marginal distribution for $\mathbf{y}(\upsilon)$ and the posterior distribution of $\mathbf{f}(\upsilon)$ given $\mathbf{y}(\upsilon)$ can be computed analytically. For example, the marginal distribution for $\mathbf{y}(\upsilon)$ is given as

$$\mathbf{y}(\upsilon) \sim \mathcal{N}(\mathbf{0}, \sum_{q=1}^{Q} \frac{1}{|\upsilon||\upsilon'|} \mathbf{B}_q \int_{\mathbf{z}\in\upsilon} \int_{\mathbf{z}'\in\upsilon'} k_q(\mathbf{z},\mathbf{z}')d\mathbf{z}'d\mathbf{z} + \boldsymbol{\Sigma}).$$

Moreno-Muñoz et al. [2018] introduced the idea of allowing each task to have a different likelihood function and modulated the parameters of that likelihood function using one or more elements in the vector-valued GP prior. For that general case, the marginal likelihood and the posterior distribution cannot be computed in closed form.

5.1.3 Stochastic variational inference

In Chapter 2 we introduced variational sparse Gaussian process (in Section 2.2.6.1) and stochastic variational inference (SVI) (in Section 2.2.6.1). Here we will introduce SVI for multi-task learning. Let $\mathcal{D} = \{\boldsymbol{\Upsilon}, \mathbf{y}\}$ be a dataset of multiple tasks with potentially different supports per task, where $\boldsymbol{\Upsilon} = \{\boldsymbol{\upsilon}_d\}_{d=1}^{D}$, with $\boldsymbol{\upsilon}_d = [\upsilon_{d,1}, \cdots, \upsilon_{d,N_d}]^\top$, and $\mathbf{y} = [\mathbf{y}_1, \cdots, \mathbf{y}_D]^\top$, with $\mathbf{y}_d = [y_{d,1}, \cdots, y_{d,N_d}]^\top$ and $y_{d,j} = y_d(\upsilon_{d,j})$. Notice that $\mathbf{y}$ without υ refers to the output vector for the dataset. We are interested in computing the posterior distribution $p(\mathbf{f}|\mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})/p(\mathbf{y})$, where $\mathbf{f} = [\mathbf{f}_1, \cdots, \mathbf{f}_D]^\top$, with $\mathbf{f}_d = [f_{d,1}, \cdots, f_{d,N_d}]^\top$ and $f_{d,j} = f_d(\upsilon_{d,j})$. In this work, we will use stochastic variational inference to compute a deterministic approximation of the posterior distribution $p(\mathbf{f}|\mathbf{y}) \approx q(\mathbf{f})$, by means of the the well known idea of *inducing variables*.

Even in the Gaussian likelihood case, for which the posterior distribution can be computed analytically, a variational treatment with inducing variables will lead to an inference procedure that is computationally efficient. In contrast to the use of SVI for traditional Gaussian processes, where the inducing variables are defined at the level of the process $\mathbf{f}$, we follow Álvarez et al. [2010] and Moreno-Muñoz et al. [2018], and define the inducing variables at the level of the latent processes $u_q(\mathbf{z})$. For simplicity in the notation, we assume $R_q = 1$. Let $\mathbf{u} = \{\mathbf{u}_q\}_{q=1}^{Q}$ be the set of inducing variables, where $\mathbf{u}_q = [u_q(\mathbf{z}_1), \cdots, u_q(\mathbf{z}_M)]^\top$, with $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^{M}$ the inducing inputs. Notice also that we have used a common set of inducing inputs $\mathbf{Z}$ for all latent functions but we can easily define a set $\mathbf{Z}_q$ per inducing vector $\mathbf{u}_q$.

For the multi-task regression case, it is possible to compute an analytical expression for the Gaussian posterior distribution over the inducing variables $\mathbf{u}$, $q(\mathbf{u})$, following a similar approach to Álvarez et al. [2010]. However, such approximation is only valid for the case in which the likelihood model $p(\mathbf{y}|\mathbf{f})$ is Gaussian and the variational bound obtained is not amenable for stochastic optimization. An alternative way of finding $q(\mathbf{u})$ also establishes a lower-bound for the log-marginal likelihood $\log p(\mathbf{y})$, but uses numerical optimization to maximise the bound with respect to the mean parameters, $\boldsymbol{\mu}$, and the covariance parameters, $\mathbf{S}$, for the Gaussian distribution $q(\mathbf{u}) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{S})$ [Moreno-Muñoz et al., 2018]. Such a numerical procedure can be used for any likelihood model $p(\mathbf{y}|\mathbf{f})$, and the optimization can be performed using mini-batches. We follow this approach.

Lower-bound The lower bound for the log-marginal likelihood follows as

$$\log p(\mathbf{y}) \geq \int\int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} = \mathcal{L},$$

where $q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$. By simplifying $\mathcal{L}$, as explained in Equation (2.52) the variational lower bound becomes:

$$\mathcal{L} = \int\int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\log p(\mathbf{y}|\mathbf{f})d\mathbf{f}d\mathbf{u} - \sum_{q=1}^{Q}\mathrm{KL}(q(\mathbf{u}_q)\|p(\mathbf{u}_q)),$$

where $p(\mathbf{f}|\mathbf{u}) \sim \mathcal{N}(\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{fu}^{\top}})$, and $p(\mathbf{u}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{uu}})$ is the prior over the inducing variables. Here $\mathbf{K_{fu}}$ is a blockwise matrix with matrices $\mathbf{K}_{\mathbf{f}_d,\mathbf{u}_q}$. In turn, each of these matrices have entries given by $k_{f_d,u_q}(\upsilon,\mathbf{z}') = \frac{a_{d,q}}{|\upsilon|}\int_{\mathbf{z}\in\upsilon} k_q(\mathbf{z},\mathbf{z}')d\mathbf{z}$. Similarly, $\mathbf{K_{uu}}$ is a block-diagonal matrix with blocks given by $\mathbf{K}_q$, with entries computed using $k_q(\mathbf{z},\mathbf{z}')$. The optimal $q(\mathbf{u})$ is chosen by numerically maximizing $\mathcal{L}$ with respect to the parameters $\boldsymbol{\mu}$ and $\mathbf{S}$. To ensure a valid covariance matrix $\mathbf{S}$ (positive definiteness), we optimize the Cholesky factor $\mathbf{L}$ for $\mathbf{S} = \mathbf{L}\mathbf{L}^{\top}$.

It can be shown [Moreno-Muñoz et al., 2018] that the bound is given as

$$\mathcal{L} = \sum_{d=1}^{D}\sum_{j=1}^{N_d}\mathbb{E}\left[\log p(y_d(\upsilon_{d,j})|f_d(\upsilon_{d,j}))\right] - \mathrm{KL}(q(\mathbf{u})\|p(\mathbf{u})), \tag{5.10}$$

where the expected value is taken with respect to the $q(\mathbf{f}) = \int q(\mathbf{f},\mathbf{u})d\mathbf{u}$ distribution, which is a Gaussian distribution. Moreover, the approximate marginal posterior for $\mathbf{f}$ is defined by $q(\mathbf{f})$ as:

$$q(\mathbf{f}) = \mathcal{N}\Big(\mathbf{f}|\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\boldsymbol{\mu}, \mathbf{K_{ff}} + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}(\mathbf{S} - \mathbf{K_{uu}})\mathbf{K_{uu}^{-1}}\mathbf{K_{fu}^{\top}}\Big). \tag{5.11}$$

The variational bound above is factorized w.r.t. the outputs according to the independence assumption between outputs:

$$\log p(\mathbf{y}|\mathbf{f}) = \sum_{d=1}^{D}\log p(\mathbf{y}_d|\mathbf{f}_d),$$

The conditional expectation is also independent across data observations:

$$\log p(\mathbf{y}|\mathbf{f}) = \sum_{j=1}^{N}\log p(\mathbf{y}_j|\mathbf{f}_j).$$

For Gaussian likelihoods,

$$p(y_d(\upsilon_{d,j})|f_d(\upsilon_{d,j})) = \mathcal{N}(y_d(\upsilon_{d,j})|f_d(\upsilon_{d,j}), \sigma^2_{y_d}),$$

we can compute the expected value in the bound in closed form. For other likelihoods, we can use numerical integration to approximate it, such as Gauss-Hermite quadratures as in Hensman et al. [2015] and Saul et al. [2016]. More details regarding Gauss-Hermite quadrature and various likelihoods can be found in Appendix C.2 and C.5, respectively. Instead of using the whole batch of data $N = \sum_{d=1}^{D} N_d$, we can use mini-batches to estimate the gradient.

The computational complexity is similar to that of the model in Moreno-Muñoz et al. [2018], $\mathcal{O}(QM^3 + JNQM^2)$, where J depends on the types of likelihoods used for the different tasks. For example, if we model all the outputs using Gaussian likelihoods, then $J = D$. The inversion cost of $\mathbf{K}_{uu}$ is $\mathcal{O}(QM^3)$, and the cost of products such as $\mathbf{K}_{fu}$ is $\mathcal{O}(JNQM^2)$. For details, see Moreno-Muñoz et al. [2018].

Hyper-parameter learning When using the multi-task learning method, we need to optimize the hyper-parameters associated with the LMC, namely: the coregionalisation matrices $\mathbf{B}_q$, the hyper-parameters of the kernels $k_q(\mathbf{z}, \mathbf{z}')$, and any other hyper-parameter associated with the likelihood functions $p(\mathbf{y}|\mathbf{f})$ that has not been considered as a member of the latent vector $\mathbf{f}(\upsilon)$. Hyper-parameter optimization is done using the lower bound $\mathcal{L}$ as the objective function. $\mathcal{L}$ is first maximised with respect to the variational distribution $q(\mathbf{u})$ and then with respect to the hyper-parameters. These two-steps are repeated one after the other until we reach convergence. This style of optimization is known as variational EM (Expectation-Maximization), when using the full dataset [Beal, 2003], or its stochastic version when employing mini-batches [Hoffman et al., 2013]. In the Expectation step we compute a variational posterior distribution, and in the Maximization step we use a variational lower bound to find point estimates of any hyper-parameters. To optimize the hyper-parameters in $\mathbf{B}_q$, we also use a Cholesky decomposition for each matrix to ensure positive definiteness. So instead of optimizing $\mathbf{B}_q$ directly, we optimize $\mathbf{L}_q$, where $\mathbf{B}_q = \mathbf{L}_q\mathbf{L}_q^\top$. For the experimental section, we use the EQ kernel for $k_q(\mathbf{z}, \mathbf{z})$, so we fix the variance for $k_q(\mathbf{z}, \mathbf{z})$ to one (the variance per output is already contained in the matrices $\mathbf{B}_q$) and optimize the length-scales ℓ_q. More mathematical details for derivatives of the variational bound with respect to variational parameters and hyper-parameters can be found at Appendix C.3 and C.4, respectively.

Predictive distribution Given a new set of test inputs $\boldsymbol{\Upsilon}_*$, the predictive distribution for $p(\mathbf{y}_*|\mathbf{y}, \boldsymbol{\Upsilon}_*)$ is computed using $p(\mathbf{y}_*|\mathbf{y}, \boldsymbol{\Upsilon}_*) = \int_{\mathbf{f}_*} p(\mathbf{y}_*|\mathbf{f}_*)q(\mathbf{f}_*)d\mathbf{f}_*$, where $\mathbf{y}_*$ and $\mathbf{f}_*$ refer to the vector-valued functions $\mathbf{y}$ and $\mathbf{f}$ evaluated at $\boldsymbol{\Upsilon}_*$. Notice that $q(\mathbf{f}_*) \approx p(\mathbf{f}_*|\mathbf{y})$. Even though $\mathbf{y}$ does not appear explicitly in the expression for $q(\mathbf{f}_*)$, it has been used to compute the

posterior for $q(\mathbf{u})$ through the optimization of $\mathcal{L}$, where $\mathbf{y}$ is explicitly taken into account. We are usually interested in the mean prediction $\mathbb{E}[\mathbf{y}_*]$ and the predictive variance $\text{var}[\mathbf{y}_*]$.

Both can be computed by exchanging integrals in the double integration over $\mathbf{y}_*$ and $\mathbf{f}_*$. For example, $\mathbb{E}[\mathbf{y}_*] = \int_{\mathbf{y}_*} \mathbf{y}_* p(\mathbf{y}_*|\mathbf{y}, \boldsymbol{\Upsilon}_*) d\mathbf{y}_* = \int_{\mathbf{f}_*} \int_{\mathbf{y}_*} \mathbf{y}_* p(\mathbf{y}_*|\mathbf{f}_*) d\mathbf{y}_* q(\mathbf{f}_*) d\mathbf{f}_*$. The inner integral in $\mathbb{E}[\mathbf{y}_*]$ is computed with the conditional distribution $p(\mathbf{y}_*|\mathbf{f}_*)$ and its form depends on the likelihood term per task. The outer integral can be approximated using numerical integration or Monte-Carlo sampling. A similar procedure can be followed to compute $\text{var}[\mathbf{y}_*]$.

5.2 Related work

Machine learning methods for different forms of aggregated datasets are also known as *multiple instance learning*, *learning from label proportions*, or *weakly supervised learning on aggregate outputs* [Bhowmik et al., 2015; Kotzias et al., 2015; Kück and de Freitas, 2005; Musicant et al., 2007; Patrini et al., 2014; Quadrianto et al., 2009]. Law et al. [2018] provided a summary of these different approaches. Typically these methods start with the following setting: each instance in the dataset is in the form of a set of inputs for which there is only one corresponding output (e.g. the proportion of class labels, an average, or a sample statistic). The prediction problem usually consists then in predicting the individual outputs for the individual inputs in the set. The setting we present in this chapter is slightly different in the sense that, in general, for each instance, the input corresponds to a support of any shape and size and the output corresponds to a vector-valued output. Moreover, each task can have its own support. Similarly, while most of these ML approaches have been developed for either regression or classification, our model is built on top of Moreno-Muñoz et al. [2018], allowing each task to have a potentially different likelihood.

As mentioned in the introduction, Gaussian processes have also been used for multiple instance learning or aggregated data [Hamelijnck et al., 2019; Haußmann et al., 2017; Kandemir et al., 2016; Kim and De la Torre, 2010; Law et al., 2018; Smith et al., 2018; Tanaka et al., 2019a,b]. Unlike most of these previous approaches, our model goes beyond the single task problem and allows learning multiple tasks simultaneously. Each task can have its own support at training and test time. Further, in contrast with other multi-task approaches, we allow for heterogeneous outputs. Although our model was formulated for a continuous support $\mathbf{x} \in \upsilon_{d,j}$, we can also define it in terms of a finite set of (previously defined) inputs in the support, e.g. a set $\{\mathbf{x}_{d,j,1}, \cdots, \mathbf{x}_{d,j,K_{d,j}}\} \in \upsilon_{d,j}$ which is more akin to the bag formulations in these previous works. This would require changing the integral $\frac{1}{|\upsilon_{d,j}|} \int_{\mathbf{z} \in \upsilon_{d,j}} u_q^i(\mathbf{z}) d\mathbf{z}$ in (5.9) for the sum $\frac{1}{K_{d,j}} \sum_{\forall \mathbf{x} \in \upsilon_{d,j}} u_q^i(\mathbf{x}_{d,j,k})$.

In geostatistics, a similar problem has been studied under the names *downscaling* and *spatial disaggregation* [Zhang et al., 2014], particularly using different forms of *kriging* [Goovaerts, 1997]. It is also closely related to the problem of *change of support* described in detail in Gotway and Young [2002]. Block-to-point kriging (or area-to-point kriging, if the support is defined in a surface) is a common method for downscaling; this is, it provides predictions at the point level provided data at the block level [Goovaerts, 2010; Kyriakidis, 2004]. We extend the approach introduced in Kyriakidis [2004] later revisited by Goovaerts [2010] for count data, to the multi-task setting, including also a stochastic variational EM algorithm for scalable inference.

If we consider the high-resolution outputs as high-fidelity outputs, and low-resolution outputs as low-fidelity outputs, our work also falls under the umbrella of *multi-fidelity models* where co-kriging using the linear model of coregionalisation has also been used as an alternative [Fernández-Godino et al., 2016; Peherstorfer et al., 2018].

5.3 Experiments

In this section, we apply the multi-task learning model for prediction to three different datasets: a synthetic example for two tasks that each have a Poisson likelihood, a two-dimensional input dataset of fertility rates aggregated by year of conception and ages in Canada, and an air-pollution sensor network, where we have two tasks. One task corresponds to a high-accuracy, low-frequency particulate matter sensor and another task corresponds to a low-cost, low-accuracy, high resolution sensor. In these examples, we use k-means clustering over the input data, with $k = M$, to initialise the values of the inducing inputs, $\mathbf{Z}$, which are also kept fixed during optimization. We assume the inducing inputs are points, but they could also be defined as intervals or supports. For standard optimization we used the LBFGS-B algorithm and, when SVI was needed, the Adam optimizer, included in *climin* library, was used for the optimization of the variational distribution (variational E-step) and the hyper-parameters (variational M-step). The implementation is based on the GPy framework and is available on Github: https://github.com/frb-yousefi/aggregated-multitask-gp.

Synthetic data In this section we evaluate our model with a synthetic dataset. For all of the experiments we use $Q = 1$ with an EQ covariance for the latent function $u_1(z)$. We set up a toy problem with $D = 2$ tasks, where both likelihood functions are Poisson. We sample from the latent vector-valued GP and use those samples to modulate the Poisson likelihoods using $\exp(f_1(\cdot))$ and $\exp(f_2(\cdot))$ as the respective rates. The first task is generated using intervals of $\upsilon_1 = 1$ units, whereas the second task is generated using intervals of $\upsilon_2 = 2$ units. All

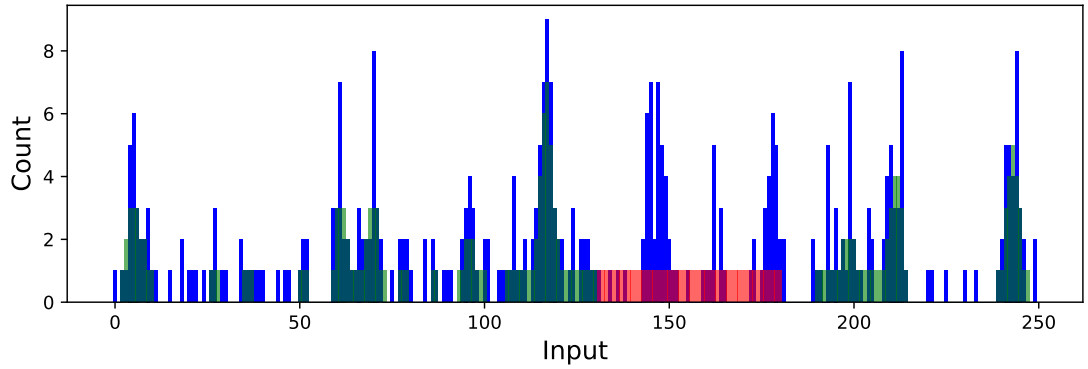
of the inputs are uniformly distributed in the range $[0, 250]$. We generated 250 observations for task 1 and 125 for task 2. To train the multi-task model, we select $N_1 = 200$ from the 250 observations for task 1 and use all $N_2 = 125$ for the second task. The other 50 data points for task 1 correspond to a gap in the interval $[130, 180]$ that we use as the test set. In this experiment, we evaluated our model's capability in predicting one task, sampled more frequently, using the training information from a second task with a larger support.

In Figure 5.1 we show that the data in the second task, with a larger support, helps in predicting the test data in the gap present in the first task, with a smaller support (b). Figure 5.1 (c) is the same as figure 5.1 (b), except that the green bars are removed for better visualisation purposes. However, this is not the case in the single task learning scenario, where the predictions are basically constant and equal to 1 (a). Both models predict the training data equally well. SMSE (Standardized Mean Squared Error) and SNLP (standardized negative log probability density) are calculated for five independent runs. For the multi-task scenario they are 0.464 ± 0.136 and -0.822 ± 0.109 and for the single task case they are 0.9699 ± 0.016 and -0.095 ± 0.036, respectively.

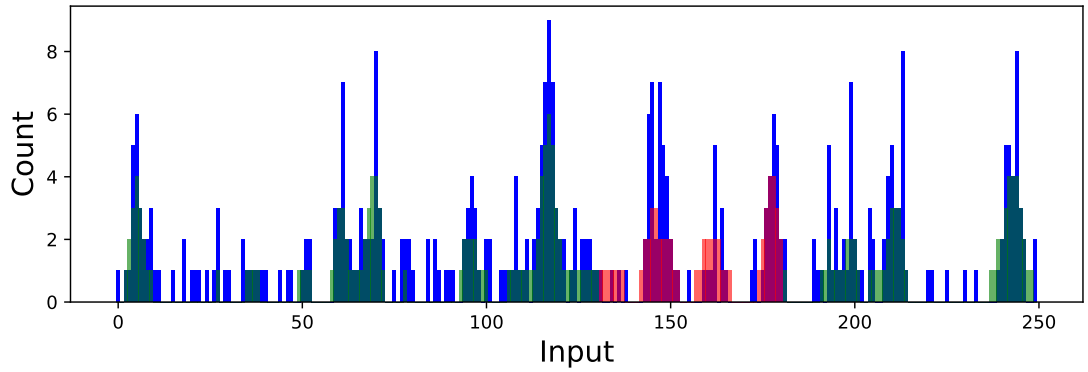
Fertility rates from a Canadian census In this experiment, a subset of the Canadian fertility dataset is used from the Human Fertility Database (HFD).[2] The dataset consists of live birth statistics by year, age of mother, and birth order. The mothers' ages fall between $[15, 54]$ and the years fall between $[1944, 2009]$. The dataset contains 2640 data points of fertility rate per birth order (the output variable), and there are four birth orders. We used the 2640 data points corresponding to the 1st birth only. The dataset was randomly split into 1640 training points and 1000 test points. We consider two tasks: the first task consists of a different number of data observations randomly taken from the 1640 training points. The second task consists of all the training data aggregated at two different resolutions: 5×5 and 2×2 (we wanted to test the predictive performance when the relation of high-resolution data to low-resolution data was 1^2 to 5^2 as well as 1^2 to 2^2). The aggregated data for the 5×5 case (a squared support of 5 years for the input `age` times 5 years for the input `years` of the study) is reduced to 104 data points and the aggregated data for the 2×2 case is reduced to 660 points.

In the experiments, we train this multi-task model by slowly increasing the original resolution training data, while maintaining a fixed amount of training points, as mentioned before for the aggregated second task. The output variable (fertility rate for the first birth) was assumed to be Gaussian, so both tasks follow a Gaussian likelihood. We use $Q = 1$ with an EQ kernel $k_1(\mathbf{z}, \mathbf{z}')$ with $\mathbf{z} \in \mathbb{R}^2$, where the two input variables are age of mother

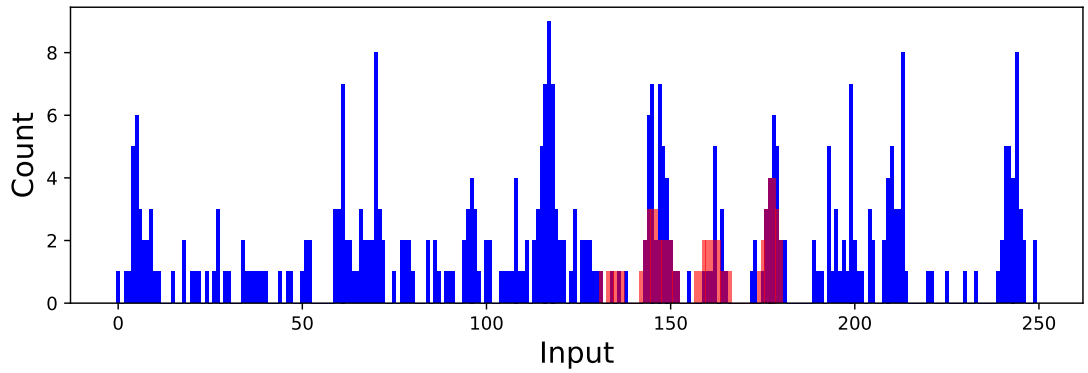
[2] https://www.humanfertility.org



(a) Single-task learning



(b) Multi-task learning



(c) Multi-task learning

Fig. 5.1 Counts for the Poisson likelihoods and predictions using the single-task vs multi-task models. Predictions are shown only for the first task (the one with support of $\upsilon_1 = 1$). The blue bars are the original one-unit support data, the green bars are the predicted training count data and the red bars are the predicted test results in the gap $[130, 180]$. We did not include the two-unit support data (the second task) for clarity in the visualisation. The multi-task figure (b) is illustrated again in figure (c) for better visualisation with the green bars removed.

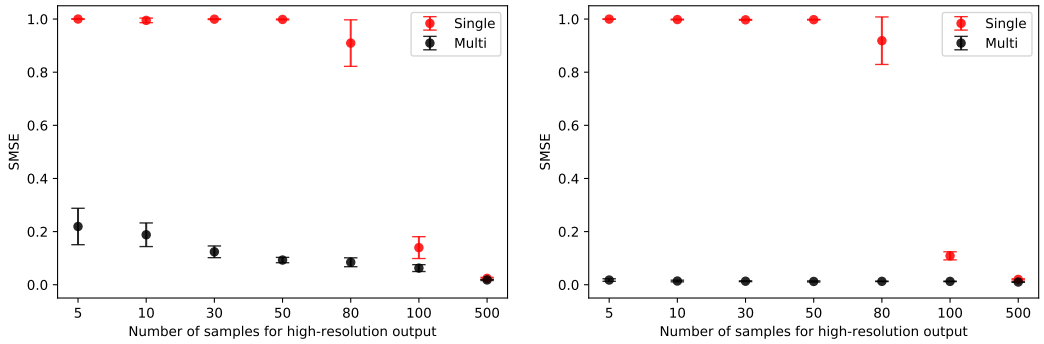


Fig. 5.2 SMSE plots for the fertility dataset for 5×5 (left panel) and 2×2 (right panel) aggregated data. The Figure shows the performance in terms of the number of training instances used for the data sampled at a higher resolution. The test set always contains 1000 instances. We plot the mean and standard deviation for five repetitions of the experiment with different sets of training and test data.

and birth year. We used 100 fixed inducing variables and mini-batches of size 50 samples. The prediction task consists of predicting the 1000 original resolution test data with the help of the second task, which consists of the aggregated data (5×5 or 2×2 for two separate experiments).

Figure 5.2 shows SMSE for five randomly selected data points in the training and test sets. We notice that the multi-task learning model outperforms the single-task GP when there are few observations in the task with the original resolution data. This pattern holds below 500 observations. At that point, both models perform equally well, since the single-task GP now has enough training data. With respect to the two different resolutions, the multi-task model performs better when the second task has a 2×2 resolution rather than 5×5 resolution, as one might also expect.

Figure 5.3 shows the results in terms of SNLP for the Fertility dataset. We can notice a similar pattern to the one observed for the SMSE in Figure 5.2.

In Figure 5.4, different baselines are compared to the proposed method. Dependent GPs (DGP) [Boyle and Frean, 2005] and Intrinsic Co-regionalisation Model (ICM) or Multi-task GPs [Bonilla et al., 2008] use the centroid of the area as input. MTGPA (the proposed method) performs better or similarly to these baselines as we increase the number of training points for the high-resolution output.

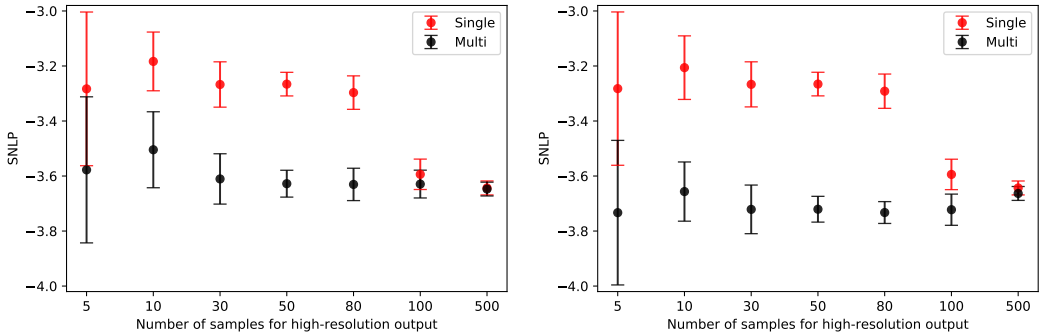


Fig. 5.3 SNLP plots for the fertility dataset for 5×5 (left panel) and 2×2 (right panel) aggregated data.

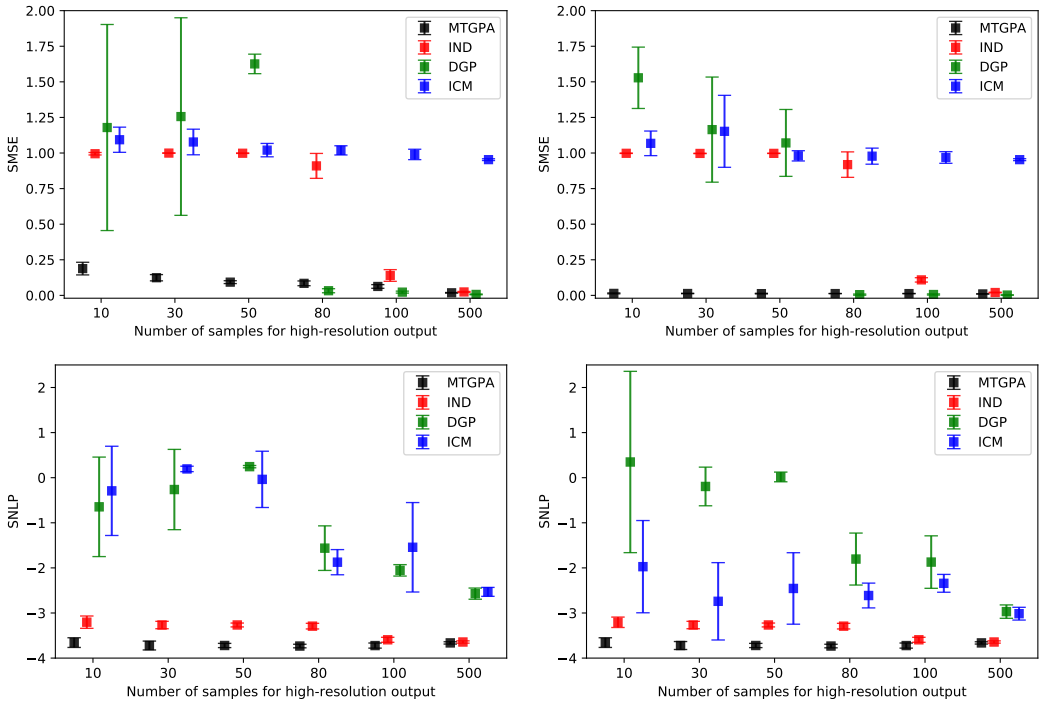


Fig. 5.4 SMSE and SNLP plots for the fertility dataset for 5×5 (left panel) and 2×2 (right panel) aggregated data for different baselines, MTGPA , Independent GP (IND), DGP and ICM.

In Figure 5.5, SNLP is calculated for four outputs (two outputs with high-resolution and a few data points, and two outputs with low-resolution and many more data points). The high-resolution data correspond to the fertility rates of the first and second birth orders.

The first task consists of a different number of data observations randomly taken from the training points of the fertility rate of the first birth. The second task consists of all the training data at the first task aggregated at two different resolutions, 5×5 and 2×2. The third task

consists of a different number of data observations randomly taken from the training points of the fertility rate of the second birth. The fourth task consists of all the training data at the third task aggregated at two different resolutions, 5×5 and 2×2.

We use two different versions of our model and compare their SNLPs. In one version, all of the outputs are considered as Gaussians (MTGPA), and in the second version all of the outputs are considered as heteroscedastic Gaussians (HetGPA).

Heteroscedastic Gaussians In the Gaussian case, only the mean parameter is modelled as a latent function, while the variance is a hyper-parameter. However, in the heteroscedastic case, both mean and variance are assumed to follow latent functions. The model with HetGPA outperforms the model with MTGPA, since it allows more flexibility toward the latent function that models the variance of the Gaussian.

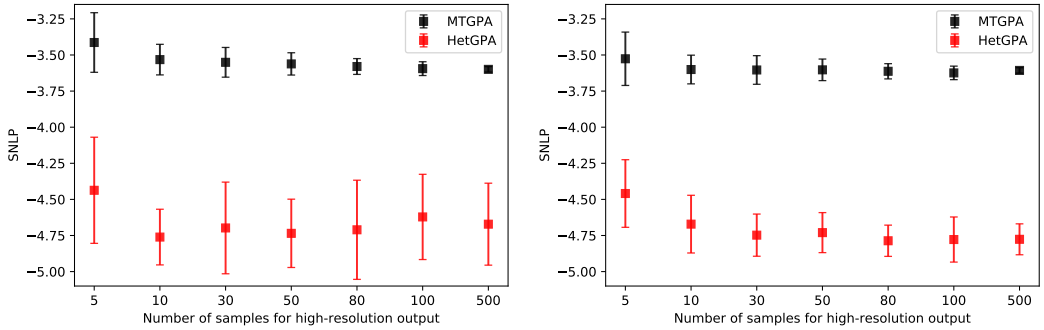


Fig. 5.5 SNLP plots for the fertility dataset for 5×5 (left panel) and 2×2 (right panel) aggregated data for four outputs (two fertility rates). All outputs are considered as Gaussian (MTGPA) and all outputs are considered as heteroscedastic Gaussian (HetGPA).

Air pollution monitoring network Particulate air pollution can be measured accurately with high temporal precision by using a β attenuation (BAM) sensor or similar. Unfortunately these are often prohibitively expensive. We propose instead that one can combine the measurements from a low-cost optical particle counter (OPC), which gives good temporal resolution but is often badly biased, with the results of Cascade Impactors (CIs), which are a cheaper method for assessing the mass of particulate species, but which integrate data over many hours (e.g. 6 or 24 hours).

One can formulate the problem as observations of integrals of a latent function, with the CI integrating over 6 hour periods and the OPC sensor integrating over short 5 minute periods. We used data from two fine particulate matter (PM) sensors. The sensors are less than 2.5 micrometer diameter (PM2.5) and are colocated in Kampala, Uganda at 0.3073°N 32.6205°E. The data is taken between 2019-03-13 and 2019-03-22. We used the average

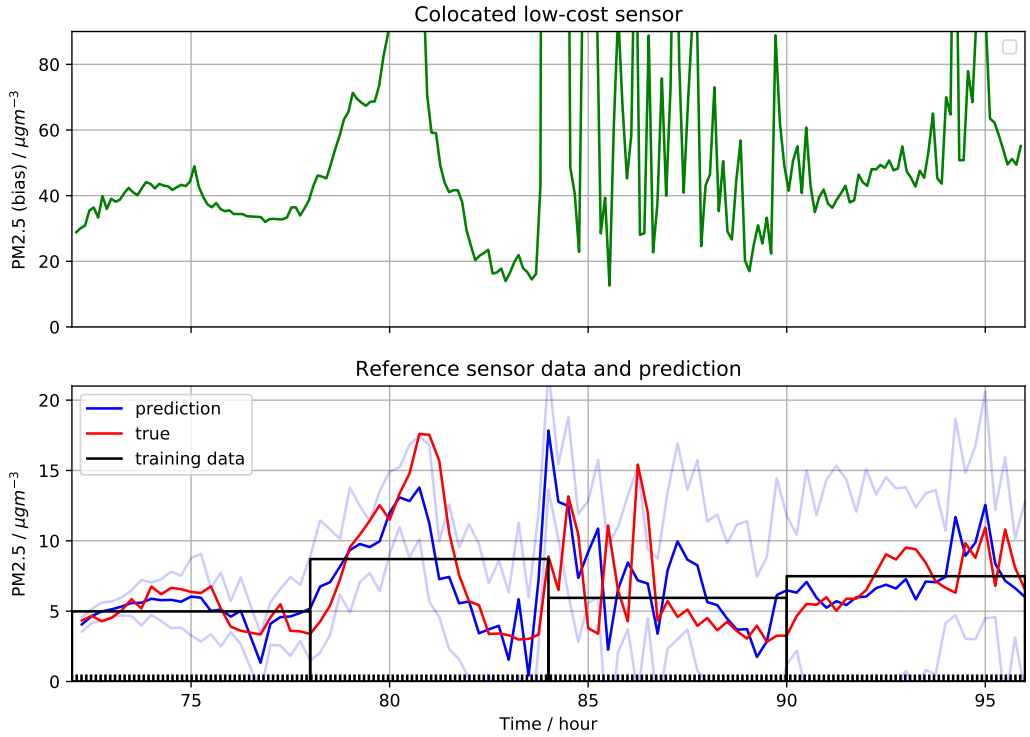


Fig. 5.6 Upper plot: a (biased) OPC low-accuracy high-frequency measurement of PM2.5 air pollution. Lower plot: the high-precision low-frequency training data (black rectangles), the test data from the same instrument (red), and the posterior prediction for this output variable, making predictions over the same 15-minute periods as the test data (blue, with pale blue indicating 95% confidence intervals). The ticks in the bottom of the lower plot indicate the position of the inducing inputs. We have also deliberately cut the higher peaks of the samples in the upper plot that can go as high as 500 $\mu g/m^3$, to visualise the samples in other parts of the plot better.

from six-hour periods from a calibrated mcerts-verified Osiris (Turnkey) particulate air pollution monitoring system as the low-resolution data, and then compared the prediction results to the original measurements (available at a 15 minute resolution). We used a PMS 5003 (Plantower) low-cost OPC to provide the high-resolution data. We typically found these values to be biased. We normalised (scaled) the data to emphasise that the *absolute* values of these variables are not of interest in this model.

Our multi-task model consists of a single latent function, $Q = 1$, with covariance $k_1(z, z')$, that follows an EQ form. We assume both outputs follow Gaussian likelihoods. In our model, one task represents the high accuracy low-resolution samples and the second task represents the low-accuracy high-resolution samples. The posterior GP aims both to fulfil the 6-hour long integrals of the high-accuracy data (from the Osiris instrument) while remaining correlated with the high-frequency bias data from the OPC. We used 2000 iterations of the

variational EM algorithm, with 200 evenly spaced inducing points, and a fixed length-scale of 0.75 hours. We only optimize the parameters of the coregionalisation matrix $\mathbf{B}_1 \in \mathbb{R}^{2\times 2}$ and the variance of the noise of each Gaussian likelihood.

Figure 5.6 illustrates the results for a 24 hour period. The training data consists of the high-resolution low-accuracy sensor and a low-frequency high accuracy sensor. The aim is to reconstruct the underlying level of pollution that both sensors are measuring. To test whether the additional high-frequency data improves accuracy, we ran the coregionalisation both with and without this additional training data.

We found that the SMSE for the predictions over the 9 days tested were substantially smaller with multi-task learning, as compared to using only the low-resolution samples, giving 0.439 ± 0.114 and 0.657 ± 0.100 respectively — this difference is statistically significant using a paired t-test with a p value of 0.0008. In summary, the model was able to incorporate this additional data and use it to improve estimates, while still ensuring the long integrals were largely satisfied.

5.4 Conclusion

In this chapter, we have introduced a powerful framework for working with aggregated datasets that allows the user to combine observations from disparate data types, with varied support. This allows us to produce both finely resolved and accurate predictions by using the accuracy of low-resolution data and the fidelity of high-resolution side-information. In all of the experiments, we showcased situations where high-resolution data can be rare, showing that jointly modeling the high-resolution data with the information from the other tasks helps to improve individual predictions, and so copes with lack of data or data scarcity issues. We chose our inducing points to lie in the latent space, a distinction which allows us to estimate multiple tasks with different likelihoods. SVI and variational-EM with mini-batches make the framework scalable and tractable for potentially very large problems. A potential extension would be to consider how the "mixing" achieved through coregionalisation could vary across the domain by extending, for example, the Gaussian Process Regression Network model [Wilson et al., 2012] to be able to deal with aggregated data. Such a model would allow latent functions of different length-scales to be relevant at different locations in the domain. In summary, this framework provides a vital toolkit, allowing a mixture of likelihoods, kernels, and tasks, and paves the way to the analysis of a very common and widely used data structure - that of values over a variety of supports on the domain.

Chapter 6

Conclusion and future work

This final chapter summarizes the research work and contributions carried out in this thesis and discusses possible directions and ideas for future research. In Section 6.1 we recapitulate the contributions presented across the thesis. In Section 6.2 we sketch possible directions for future work.

6.1 Thesis summary

In this thesis we have looked into data scarcity and the issues with lack of labeled data in different applications. The main objective of this thesis was to study various ways of tackling these issues using Gaussian processes. We proposed different approaches for various granularity of labels. We could use latent variable models for the cases where we have a limited patch-level labeled data (Chapter 3). We could use active learning when we have high-level labels and no patch-level labels are available (Chapter 4). Lastly, we could combine various granularity of data and use multi-task learning to jointly learn them in Chapter 5.

In Chapter 3, we developed a latent variable model known as SCLVM that can cope with imbalanced data by dividing the latent space into a shared space and a private space. The shared space captures the data characteristics shared across all of the data, regardless of category. The private space captures the data characteristics specific to each category. We proposed a new kernel formulation that enables the separation of the latent space by incorporating label information and which derives an efficient variational inference method. Since the modeling of the private space is category specific, the larger category does not dominate the minority class anymore.

In Chapter 4, we address data scarcity and lack of labeled data issues in image classification using a combination of transfer learning from Convolutional Neural Networks with Gaussian process classification. Manually labelling images is tedious and potentially

expensive. In many applications, there are lots of data with no labels, and even if labels are available, they are often high-level labels. To overcome this difficulty, our goal was to select data in a sensible manner so that we have efficient models which require fewer labeled images. This was achieved using active learning. The process started with a small set of labeled data and continued by actively adding data from a pool of unlabeled data by manually annotating them. The proposed technique requires fewer expert labels and reduces the cost of annotation, while being more data efficient compared to other supervised learning approaches such as CNNs. In order to tackle data scarcity, we proposed cutting images to small patches, which helped to increase the data for the region of interest, and subsequently alleviated the lack of minority class issue.

In Chapter 5, we presented a novel multi-task learning model based on Gaussian processes for joint learning of variables that have been aggregated at different input scales. Data rarity can arise due to lack of high-resolution data. Jointly modeling tasks using information from other tasks helps to improve predictions on rare cases. Our model represented each task as a linear combination of the realizations of latent processes that are integrated at a different scale per task. We were then able to compute the cross-covariance between different tasks either analytically or numerically. We also allowed each task to have a potentially different likelihood model and provide a variational lower bound that can be optimized in a stochastic fashion, making our model suitable for larger datasets.

By combining the ideas introduced throughout this thesis we believe we have gone some way toward addressing data scarcity and lack of labeled data issues with various granularities using Gaussian processes.

6.2 Future directions

There are various future directions which could be pursued based on the studies conducted in this thesis. A brief summary of possible future directions are provided in this section.

Deep GPs have been shown to be applicable when data is scarce [Damianou and Lawrence, 2013]. Deep GPs encapsulate the idea of stacking Gaussian Processes in an analogous way to nesting layers of a neural network. A single layer of the deep GP can be considered as a GPLVM, and variables in each layer of a deep hierarchy can be treated as latent variables [Damianou and Lawrence, 2013]. This enables a hierarchical setting for the variational methodology. One can extend the uncertainty propagation in the "shallow" GPs to multiple layers to allow for a more complex mapping function from input to output. This idea can be applied to the imbalanced data problem in Chapter 3 where very few labeled data examples are available for the class of interest.

In Chapter 3, we restricted our experiments to use RBF kernel since we could calculate the lower bound in closed form. One future direction is exploring other kernels or combining different kernels [Duvenaud, 2014]. Different kernels express different structures. We can explore combining various kernels with the proposed shared-private kernel structure to include as much structure as necessary into our model with the capacity necessary to represent more specific applications.

In Chapter 4, for addressing the lack of labeled data, we used active learning to determine which input the expert/oracle should label next based only on the GP classification predictive mean $p(y^*|\mathbf{y})$ of the transformed latent function. However, we have access to the full predictive posterior of f for any input point, as a result of modeling the latent function f with a Gaussian Process. One possible direction in which to take this would be to explore the predictive variance for the latent function f. In this setting there are a number of directions we could take. We could only choose the data with an uncertain mean of $p(y^*|\mathbf{y})$ and uncertain variance of $p(f^*|\mathbf{y})$, but we also could select the data with a certain mean and uncertain variance. Bayesian optimization (BO) can be used to search efficiently the space of parameters [GPyOpt, 2016; Osborne et al., 2009]. The parameters in our setting will have a predictive mean of $p(y^*|\mathbf{y})$ versus predictive variance of $p(f^*|\mathbf{y})$. In BO the aim is to find the optimum parameters by spending the lowest possible number of evaluations. This is linked to exploration versus exploitation trade-off. Exploration seeks to sample in locations where the uncertainty is high. Exploitation on the other hand, seeks to sample where the model predicts a good objective and continuously exploiting known information might give little to no yield.

A final possible direction in Chapter 5 would be to extend the method to apply to images. In Chapter 5 we only used the location information. Combining the features from images with the location information (x and y coordinates) could open new directions for applying multi-task learning to aggregated image datasets. There might be cases where there are many low-quality images and very few high quality images. Jointly modeling images with various resolutions will help improve the individual predictions. This could be achieved by combining the structural properties of deep learning architectures with the non-parametric flexibility of GPs [Wilson et al., 2016a].

Another potential extension that we mentioned in Chapter 5 would be to consider how the "mixing" achieved through coregionalisation could vary across the domain by extending, for example, the Gaussian Process Regression Network model [Wilson et al., 2012], which combines the structural properties of a Bayesian neural network with Gaussian processes to be able to deal with aggregated data. Such a model would allow latent functions of different length-scales to be relevant at different locations in the domain.

References

Abe, N. (2003). Invited talk: Sampling approaches to learning from imbalanced datasets: active learning, cost sensitive learning and beyond. In *Proc. of ICML Workshop: Learning from Imbalanced Data Sets*, volume 22.

Álvarez, M., Luengo, D., Titsias, M., and Lawrence, N. (2010). Efficient multioutput Gaussian processes through variational inducing kernels. In *AISTATS*, pages 25–32.

Alvarez, M. A. and Lawrence, N. D. (2009). Sparse convolved Gaussian processes for multi-output regression. In *NIPS*, pages 57–64.

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

Attenberg, J. and Provost, F. (2010). Why label when you can search? alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 423–432.

Barber, D. and Williams, C. K. (1997). Gaussian processes for bayesian classification via hybrid monte carlo. In *Advances in neural information processing systems*, pages 340–346.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, 6(1):20–29.

Beal, M. J. (2003). Variational algorithms for approximate Bayesian inference. *Ph. D. Thesis, University College London*.

Bhowmik, A., Ghosh, J., and Koyejo, O. (2015). Generalized linear models for aggregated data. In *AISTATS*, pages 93–101.

Bishop, C. M. et al. (2006). *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ.

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.

Bonilla, E. V., Chai, K. M., and Williams, C. (2008). Multi-task Gaussian process prediction. In *NIPS*, pages 153–160.

Boyle, P. and Frean, M. (2005). Dependent Gaussian processes. In *NIPS*, pages 217–224.

Chan, P. K. and Stolfo, S. J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *KDD*, volume 98, pages 164–168.

Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6.

Chen, Z., Ding, R., Chin, T.-W., and Marculescu, D. (2018). Understanding the impact of label granularity on cnn-based image classification. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 895–904. IEEE.

Cieslak, D. A., Hoens, T. R., Chawla, N. V., and Kegelmeyer, W. P. (2012). Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158.

Csató, L. and Opper, M. (2002). Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668.

Damianou, A. (2015). *Deep Gaussian Processes and Variational Propagation of Uncertainty*. PhD thesis, The University of Sheffield, Sheffield, UK.

Damianou, A., Ek, C., Titsias, M., and Lawrence, N. (2012). Manifold relevance determination. *arXiv preprint arXiv:1206.4610*.

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215.

Damianou, A. C., Titsias, M. K., and Lawrence, N. D. (2014). Variational inference for uncertainty on the inputs of gaussian process models. *arXiv preprint arXiv:1409.2287*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164.

Drummond, C., Holte, R. C., et al. (2003). C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

Elliott, A. (2019). *The culture of AI: Everyday life and the digital revolution*. Routledge.

Ertekin, S., Huang, J., Bottou, L., and Giles, L. (2007a). Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM.

Ertekin, S., Huang, J., and Giles, C. L. (2007b). Active learning for class imbalance problem. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 823–824. ACM.

Estabrooks, A., Jo, T., and Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). Adacost: misclassification cost-sensitive boosting. In *Icml*, volume 99, pages 97–105.

Fawcett, T. and Provost, F. J. (1996). Combining data mining and machine learning for effective user profiling. In *KDD*, pages 8–13.

Fernández-Godino, M. G., Park, C., Kim, N.-H., and Haftka, R. T. (2016). Review of multi-fidelity models.

Fumera, G. and Roli, F. (2002). Support vector machines with embedded reject option. In *Pattern recognition with support vector machines*, pages 68–82. Springer.

Gelfand, A. E. and Smith, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409.

Ghahramani, Z. (2013). Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110553.

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

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press.

Goovaerts, P. (2010). Combining areal and point data in geostatistical interpolation: Applications to soil science and medical geography. *Mathematical Geosciences*, pages 535–554.

Gotway, C. A. and Young, L. J. (2002). Combining incompatible spatial data. *Journal of the American Statistical Association*, pages 632–648.

GPyOpt (2016). GPyOpt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt.

Hamelijnck, O., Damoulas, T., Wang, K., and Girolami, M. (2019). Multi-resolution multi-task Gaussian processes.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

Haußmann, M., Hamprecht, F. A., and Kandemir, M. (2017). Variational Bayesian multiple instance learning with Gaussian processes. In *CVPR*, pages 810–819.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

He, H. and Ma, Y. (2013). *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *UAI*, pages 282–290.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable variational Gaussian process classification. In *AISTATS*, page 351–360.

Herbrich, R., Lawrence, N. D., and Seeger, M. (2003). Fast sparse gaussian process methods: The informative vector machine. In *Advances in neural information processing systems*, pages 625–632.

Hjort, N. L., Holmes, C., Müller, P., and Walker, S. G. (2010). *Bayesian nonparametrics*, volume 28. Cambridge University Press.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, pages 1303–1347.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Iglesias, J. E., Konukoglu, E., Montillo, A., Tu, Z., and Criminisi, A. (2011). Combining generative and discriminative models for semantic segmentation of ct scans via active learning. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 25–36. Springer.

Jaakkola, T. S. and Jordan, M. I. (1996). Computing upper and lower bounds on likelihoods in intractable networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 340–348. Morgan Kaufmann Publishers Inc.

Jolliffe, I. T. (1986). *Principal Component Analysis*. springer, New York.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Journel, A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*. Academic Press.

Kandemir, M., Haussmann, M., Diego, F., Rajamani, K. T., van der Laak, J., and Hamprecht, F. A. (2016). Variational weakly supervised Gaussian processes. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Kapoor, A., Grauman, K., Urtasun, R., and Darrell, T. (2007). Active learning with gaussian processes for object categorization. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Key, T. J., Verkasalo, P. K., and Banks, E. (2001). Epidemiology of breast cancer. *The lancet oncology*, 2(3):133–140.

Kim, M. and De la Torre, F. (2010). Gaussian processes multiple instance learning. In *ICML*, pages 535–542.

Konyushkova, K., Sznitman, R., and Fua, P. (2017). Learning active learning from data. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4225–4235. Curran Associates, Inc.

Kotzias, D., Denil, M., de Freitas, N., and Smyth, P. (2015). From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606.

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kubat, M., Holte, R. C., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215.

Kück, H. and de Freitas, N. (2005). Learning about individuals from group statistics. In *UAI*, pages 332–339.

Kukar, M., Kononenko, I., et al. (1998). Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary Gaussian process classification. *Journal of machine learning research*, 6(Oct):1679–1704.

Kyriakidis, P. C. (2004). A geostatistical framework for area-to-point spatial interpolation. *Geographical Analysis*, pages 259–289.

Law, H. C. L., Sejdinovic, D., Cameron, E., Lucas, T. C., Flaxman, S., Battle, K., and Fukumizu, K. (2018). Variational learning on aggregate outputs with Gaussian processes. In *NeurIPS*, pages 6084–6094.

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

Lawrence, N. D., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 15*. Citeseer.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

MacKay, D. J. (1992). Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604.

Maloof, M. A. (2003). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, volume 2, pages 2–1.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc.

Moreno-Muñoz, P., Artés, A., and Álvarez, M. A. (2018). Heterogeneous multi-output Gaussian process prediction. In *NeurIPS*, pages 6711–6720.

Musicant, D. R., Christensen, J. M., and Olson, J. F. (2007). Supervised learning by training on aggregate outputs. In *Seventh IEEE International Conference on Data Mining (ICDM)*, pages 252–261.

Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*.

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078.

Opper, M. and Archambeau, C. (2009). The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792.

Osborne, M. A., Garnett, R., and Roberts, S. J. (2009). Gaussian processes for global optimization. In *3rd international conference on learning and intelligent optimization (LION3)*, pages 1–15.

Patrini, G., Nock, R., Rivera, P., and Caetano, T. (2014). (almost) no label no cry. In *NIPS*, pages 190–198.

Peherstorfer, B., Willcox, K., and Gunzburger, M. (2018). Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization. *SIAM Review*, 60(3):550–591.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines.

Provost, F. (2000). Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 workshop on imbalanced data sets*, pages 1–3.

Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine learning*, 42(3):203–231.

Quadrianto, N., Smola, A. J., Caetano, T. S., and Le, Q. V. (2009). Estimating labels from label proportions. *J. Mach. Learn. Res.*, pages 2349–2374.

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

Quinonero-Candela, J., Rasmussen, C. E., and Williams, C. K. (2007). Approximation methods for gaussian process regression. In *Large-scale kernel machines*, pages 203–223. MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2006a). *Gaussian Processes for Machine Learning*. mit, Cambridge, MA.

Rasmussen, C. E. and Williams, C. K. I. (2006b). *Gaussian processes for machine learning*. MIT Press.

Roy, N. and McCallum, A. (2001). Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Saul, A. D. (2016). *Gaussian Process Based Approaches for Survival Analysis*. PhD thesis, University of Sheffield.

Saul, A. D., Hensman, J., Vehtari, A., and Lawrence, N. D. (2016). Chained Gaussian processes. In *AISTATS*, page 1431–1440.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Seeger, M. (2000). Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. In *Advances in neural information processing systems*, pages 603–609.

Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. Technical report.

Settles, B. (2012). *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning Series. Morgan & Claypool.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith, M. T., Álvarez, M. A., and Lawrence, N. D. (2018). Gaussian process regression for binned data.

Snell, V. (2013). *Shape and Texture Recognition for Automated Analysis of Pathology Images*. PhD thesis, Centre for Vision, Speech and Signal Processing, University of Surrey, Surrey, UK.

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

Snelson, E. and Ghahramani, Z. (2006b). Sparse Gaussian processes using pseudo-inputs. In *NIPS*, pages 1257–1264.

Society, T. A. C. (2020). How common is breast cancer?

Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Tanaka, Y., Iwata, T., Tanaka, T., Kurashima, T., Okawa, M., and Toda, H. (2019a). Refining coarse-grained spatial data using auxiliary spatial data sets with various granularities. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5091–5099.

Tanaka, Y., Tanaka, T., Iwata, T., Kurashima, T., Okawa, M., Akagi, Y., and Toda, H. (2019b). Spatially aggregated Gaussian processes with multivariate areal outputs.

Teh, Y.-W., Seeger, M., and Jordan, M. I. (2005). Semiparametric latent factor models. In *AISTATS*, page 333–340.

Ting, K. M. (2002). An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):659–665.

Tipping, M. E. and Bishop, C. M. (1999a). Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482.

Tipping, M. E. and Bishop, C. M. (1999b). Probabilistic principal component analysis. *JRSSb*, 6(3):611–622.

Titsias, M. (2009a). 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.

Titsias, M. K. (2009b). Variational model selection for sparse gaussian process regression. *Report, University of Manchester, UK*.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

Urtasun, R. and Darrell, T. (2007). Discriminative Gaussian process latent variable model for classification. In *Proceedings of the 24th international conference on Machine learning*, pages 927–934. ACM.

Veta, M., Van Diest, P. J., Willems, S. M., Wang, H., Madabhushi, A., Cruz-Roa, A., Gonzalez, F., Larsen, A. B., Vestergaard, J. S., Dahl, A. B., et al. (2015). Assessment of algorithms for mitosis detection in breast cancer histopathology images. *Medical image analysis*, 20(1):237–248.

Wallace, B. C., Small, K., Brodley, C. E., and Trikalinos, T. A. (2010). Active learning for biomedical citation screening. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 173–182.

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

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016b). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. In *ICML*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Yousefi, F., Álvarez, M., Lawrence, N., and Ek, C. H. (2018). Active learning using gaussian processes for imbalanced datasets. *Neural Information Processing Systems (NeurIPS) workshop on Bayesian NonParametrics (BNP)*.

Yousefi, F., Dai, Z., Ek, C. H., and Lawrence, N. (2016). Unsupervised learning with imbalanced data via structure consolidation latent variable model. *International Conference on Learning Representations-Workshop track*.

Yousefi, F., Smith, M. T., and Álvarez, M. (2019). Multi-task learning for aggregated data using gaussian processes. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 15076–15086. Curran Associates, Inc.

Zhang, J., Atkinson, P., and Goodchild, M. F. (2014). *Scale in Spatial Information and Analysis*. CRC Press.

Zheng, Z., Wu, X., and Srihari, R. (2004). Feature selection for text categorization on imbalanced data. *ACM Sigkdd Explorations Newsletter*, 6(1):80–89.

Zhou, Z.-H. and Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77.

Zhu, X. and Goldberg, A. B. (2009). Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130.

Appendix A

This appendix contains some basic maths knowledge that can be useful for calculating Gaussian distribution and matrix identities.

A.1 Gaussian identities

A.1.1 Conditional and marginal distributions of partitioned Gaussians

Given a joint Gaussian distribution $\mathcal{N}(\mathbf{x}|\mu,\Sigma)$, two dis-joint subsets of $\mathbf{x}$ are $\mathbf{x}_a$ and $\mathbf{x}_b$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix},\ \mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix},\ \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \tag{A.1}$$

The conditional distribution is:

$$p(\mathbf{x}_a|\mathbf{x}_b) = \mathcal{N}(\mathbf{x}|\mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(\mathbf{x}-\mu_b),\ \Lambda_{aa}^{-1}), \tag{A.2}$$

where, $\Lambda = \Sigma^{-1}$ and it is called the precision matrix:

$$\Lambda = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix} \tag{A.3}$$

The marginal distribution is:

$$p(\mathbf{x}_a) = \mathcal{N}(\mathbf{x}_a|\mu_a,\Sigma_{aa}). \tag{A.4}$$

A.1.2 Conditional and marginal distributions of Gaussians

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

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \Sigma_x) \tag{A.5}$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|A\mathbf{x}+b, \Sigma_y). \tag{A.6}$$

The marginal distribution of $\mathbf{y}$ and the conditional distribution of $p(\mathbf{x}|\mathbf{y})$ can be calculated as:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|A\mu+b, \Sigma_y + A\Sigma_x A^\top) \tag{A.7}$$
$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\Sigma A^\top \Sigma_f^{-1}(y-b) + \Sigma_x^{-1}\mu, \Sigma), \tag{A.8}$$

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

A.2 Matrix identities

Following notations are used in this section: B is a $n \times n$ matrix, U and V are $n \times m$ and W is $m \times m$.

The matrix inversion lemma:

$$(B + UWV^\top)^{-1} = B^{-1} - B^{-1}U(W^{-1} + V^\top Z^{-1}U)^{-1}V^\top B^{-1} \tag{A.9}$$

A is $n \times n$ invertible matrix and its inverse A^{-1} is partitioned into:

$$A = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}, A^{-1} = \begin{bmatrix} \tilde{P} & \tilde{Q} \\ \tilde{R} & \tilde{S} \end{bmatrix} \tag{A.10}$$

P and $\tilde{P}$ are $n_1 \times n_1$ matrices and S and $\tilde{S}$ are $n_2 \times n_2$ matrices where, $n = n_1 + n_2$ and sub-matrices are calculated as follows:

$$\begin{aligned} \tilde{P} &= P^{-1} + P^{-1}QMRP^{-1} \\ \tilde{Q} &= -P^{-1}QM \\ \tilde{R} &= -MRP^{-1} \\ \tilde{S} &= M, \end{aligned}$$

where,

$$M = (S - RP^{-1}Q)^{-1} \tag{A.11}$$

Appendix B

This appendix contains further details about calculating Ψ statistics, DGPLVM and Laplace approximation.

B.1 Calculating the ψ statistics

Here we explain how to calculate Ψ statistics for RBF kernel. We re-write the equations (3.18), (3.19), (3.20) from Chapter 3:

$$\begin{aligned}\psi_0 &= \mathrm{tr}\left(\langle \mathbf{K}_{ff} \rangle_{q(\mathbf{X})}\right)\\ \Psi_1 &= \langle \mathbf{K}_{fu} \rangle_{q(\mathbf{X})}\\ \Psi_2 &= \langle \mathbf{K}_{uf}\mathbf{K}_{fu} \rangle_{q(\mathbf{X})}\end{aligned}$$

The expectation of the covariance matrices above with respect to $q(\mathbf{X})$ can be computed for each $q(\mathbf{X}) = \sum_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mu_n, \mathcal{S}_n) = \sum_{n=1}^{N} q(\mathbf{x}_n)$. So we can write:

$$\psi_0 = \sum_{n=1}^{N} \psi_0^n, \tag{B.1}$$

where

$$\psi_0^n = \int k(\mathbf{x}_n, \mathbf{x}_n)\mathcal{N}(\mathbf{x}_n|\mu_n, \mathcal{S}_n)\, d\mathbf{x}_n. \tag{B.2}$$

Ψ_1 is an $N \times M$ matrix:

$$(\Psi_1)_{nm} = \int k(\mathbf{x}_n, \mathbf{z}_m)\mathcal{N}(\mathbf{x}_n|\mu_n, \mathcal{S}_n)\, d\mathbf{x}_n. \tag{B.3}$$

Finally, Ψ_2 is an $M \times M$ matrix and is written as:

$$\Psi_2 = \sum_{n=1}^{N} \Psi_2^n \tag{B.4}$$

where

$$(\Psi_2^n)_{mm'} = \int k(\mathbf{x}_n, \mathbf{z}_m) k(\mathbf{z}'_m, \mathbf{x}_n) \mathcal{N}(\mathbf{x}_n | \mu_n, \mathcal{S}_n)\, d\mathbf{x}_n. \tag{B.5}$$

Since the kernel is RBF, we can compute the convolution of the covariance function with a Gaussian density analytically in equations (B.2), (B.3) and (B.5), so we have:

$$\psi_0 = N\sigma^2 \tag{B.6}$$

$$(\Psi_1)_{nm} = \sigma^2 \prod_{j=1}^{q} \frac{\exp\left(-\frac{1}{2} \frac{w_j(\mu_{nj} - z_{mj})^2}{w_j \mathcal{S}_{nj} + 1} \right)}{(w_j \mathcal{S}_{nj} + 1)^{\frac{1}{2}}} \tag{B.7}$$

$$(\Psi_2^n)_{mm'} = \sigma^4 \prod_{j=1}^{q} \frac{\exp\left(-\frac{w_j(z_{mj} - z_{m'j})^2}{4} - \frac{w_j(\mu_{nj} - \frac{(z_{mj} + z_{m'j})}{2})^2}{2w_j \mathcal{S}_{nj} + 1} \right)}{(2w_j \mathcal{S}_{nj} + 1)^{\frac{1}{2}}}, \tag{B.8}$$

where, w_j is the hyper-parameter of the RBF kernel for every dimension of the data.

B.2 Discriminative Gaussian process latent variable model (DGPLVM)

We used DGPLVM in Chapter 3, Section 3.3. We will provide more details about it here. DGPLVM uses Generalized Discriminant Analysis (GDA) constraints and places informative prior that is derived from discriminative criterion. This prior encourages latent positions of the same class to be close to each other and latent position of different classes to be far. Linear discriminant analysis (LDA) and the kernelized version called Generalized discriminant analysis (GDA) are discriminative latent variable models. These methods are not probabilistic and might not generalize well when training data is little [Urtasun and Darrell, 2007].

$$J(\mathbf{X}) = tr(S_w^{-1} S_b), \tag{B.9}$$

where S_w is the within class matrix and S_b is the between class matrix and they are defined as:

$$S_w = \sum_c (\mu_c - \mu)(\mu_c - \mu) \tag{B.10}$$

$$S_b = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c^\top) \tag{B.11}$$

where c is the number of classes, μ is overall mean of the data of all the classes, μ_c is the mean of class c and x_i are the training points of class c. Equation (B.9), is a function of latent X and is defined as:

$$p(\mathbf{X}) = \frac{1}{\mathcal{Z}} \exp(-\frac{1}{\sigma^2} J^{-1}), \tag{B.12}$$

where $\mathcal{Z}$ is the normalization constant and σ^2 represents a global scaling of the prior, smaller σ means more discrimination and larger σ means more generalization. Inverse of Equation (B.9) is used because we are minimizing the log likelihood.

B.3 Laplace Approximation

This is more detailed derivations for the Laplace approximation that is defined in Chapter 2, Section 2.2.4.1 and can also be found in Rasmussen and Williams [2006a]. We re-write the posterior:

$$p(\mathbf{f}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X})}{p(\mathbf{y}|\mathbf{X})}. \tag{B.13}$$

We can only consider an un-normlized posterior when maximizing w.r.t. $\mathbf{f}$ [Rasmussen and Williams, 2006a].

$$p(\mathbf{f}|\mathbf{X},\mathbf{y}) \propto p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}|\mathbf{X}), \tag{B.14}$$

taking the logarithm on the right hand side of the above equation and using GP prior formula:

$$\begin{aligned} p(\mathbf{f}|\mathbf{X}) &= \mathcal{N}(\mathbf{f}|0,\mathbf{K}_{ff}) \\ \log p(\mathbf{f}|\mathbf{X}) &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} \\ \log p(\mathbf{f}|\mathbf{X},\mathbf{y}) &= \log(p(\mathbf{y}|\mathbf{f})) + \log(p(\mathbf{f}|\mathbf{X})) \\ &= \log p(\mathbf{y}|\mathbf{f}) - \frac{N}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} \end{aligned} \tag{B.15}$$

By assuming $\psi(\mathbf{f}) = \log p(\mathbf{f}|\mathbf{x},\mathbf{y})$ and by taking the first and second order derivatives of Equation (B.15) w.r.t. $\mathbf{f}$:

$$\frac{\partial \psi}{\partial \mathbf{f}} = \nabla \log p(\mathbf{y}|\mathbf{f}) - \mathbf{K}^{-1}\mathbf{f}$$
$$\frac{\partial^2 \psi}{\partial \mathbf{f}^2} = \nabla\nabla \log p(\mathbf{y}|\mathbf{f}) - \mathbf{K}^{-1}$$

It is assumed $\mathcal{W} = -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$ and at the maximum:

$$\nabla\psi(\mathbf{f}) = 0 \Rightarrow \hat{\mathbf{f}} = \mathbf{K}(\nabla \log p(\mathbf{y}|\hat{\mathbf{f}}))$$

This equation can not be solved directly, since $\nabla \log p(\mathbf{y}|\hat{\mathbf{f}})$ is a non-linear function of $\hat{\mathbf{f}}$, and Newton's method will be used to iteratively find the maximum:

$$\begin{aligned}\mathbf{f}^{\text{new}} &= \mathbf{f} - (\nabla\nabla\psi)^{-1}\nabla\psi \\ &= \mathbf{f} + (\mathbf{K}^{-1} + \mathcal{W})^{-1}(\nabla \log p(\mathbf{y}|\mathbf{f}) - \mathbf{K}^{-1}\mathbf{f}) \\ &= (\mathbf{K}^{-1} + \mathcal{W})^{-1}(\mathcal{W}\mathbf{f} + \nabla \log p(\mathbf{y}|\mathbf{f}))\end{aligned} \tag{B.16}$$

If we divide $\mathbf{f}$ to two vectors $\mathbf{f}_1$ and $\mathbf{f}_2$ and assume that $\mathbf{f}_1$ corresponds to the points that are not well explained and $\mathbf{f}_2$ corresponds to the points that are well explained. By well explained points we mean that $\partial \log p(\mathbf{y}_i|\mathbf{f}_i)/\partial \mathbf{f}_i$ and $\mathcal{W}_{ii}$ are close to zero for these points. We can show that

$$\mathbf{f}_1^{\text{new}} = \mathbf{K}_{11}(\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1}(\mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \tag{B.17}$$
$$\mathbf{f}_2^{\text{new}} = \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{f}_1^{\text{new}} \tag{B.18}$$

proof:

$$\begin{aligned}\begin{bmatrix}\mathbf{f}_1^{\text{new}} \\ \mathbf{f}_2^{\text{new}}\end{bmatrix} &= \begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix}\left(\begin{bmatrix}\mathbf{I}_{11} & 0 \\ 0 & \mathbf{I}_{22}\end{bmatrix} + \begin{bmatrix}\mathcal{W}_{11} & 0 \\ 0 & \mathcal{W}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix}\right)^{-1} \\ &\left(\begin{bmatrix}\mathcal{W}_{11} & 0 \\ 0 & \mathcal{W}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{f}_1 \\ \mathbf{f}_2\end{bmatrix} + \begin{bmatrix}\nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ \nabla \log p(\mathbf{y}_2|\mathbf{f}_2))\end{bmatrix}\right) \\ &= \begin{bmatrix}\mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22}\end{bmatrix}\begin{bmatrix}\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11} & \mathcal{W}_{11}\mathbf{K}_{12} \\ \mathcal{W}_{22}\mathbf{K}_{21} & \mathbf{I}_{22} + \mathcal{W}_{22}\mathbf{K}_{22}\end{bmatrix}^{-1}\begin{bmatrix}\mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ \mathcal{W}_{22}\mathbf{f}_2 + \nabla \log p(\mathbf{y}_2|\mathbf{f}_2))\end{bmatrix}\end{aligned}$$

Based on the assumption in Rasmussen and Williams [2006a], $\mathbf{f}_2$ corresponds to the points that are well-explained, so the $\mathcal{W}_{22} \simeq 0$ and $\nabla \log p(\mathbf{y}_2|\mathbf{f}_2) \simeq 0$ for these points and

$\mathbf{f}_1$ corresponds to the ones that are not well-explained. Applying this definition inside the matrices:

$$= \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11} & \mathcal{W}_{11}\mathbf{K}_{12} \\ 0 & \mathbf{I}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ 0 \end{bmatrix}$$

Using A.2 we have:

$$A^{-1} = \begin{bmatrix} \tilde{P} & \tilde{Q} \\ \tilde{R} & \tilde{S} \end{bmatrix}$$

$$\begin{aligned} \tilde{P} &= (\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1} \\ \tilde{R} &= 0 \\ \tilde{S} &= \mathbf{I}_{22} \end{aligned}$$

For $\tilde{Q}$ we will use itself rather than writing down the inverse to keep the equations simple.

$$\begin{aligned} \begin{bmatrix} \mathbf{f}_1^{\text{new}} \\ \mathbf{f}_2^{\text{new}} \end{bmatrix} &= \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} (\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1} & \tilde{Q} \\ 0 & \mathbf{I}_{22} \end{bmatrix} \begin{bmatrix} \mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_{11}(\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1} & \mathbf{K}_{11}\tilde{Q} + \mathbf{K}_{12}\mathbf{I}_{22} \\ \mathbf{K}_{21}(\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1} & \mathbf{K}_{21}\tilde{Q} + \mathbf{K}_{22}\mathbf{I}_{22} \end{bmatrix} \begin{bmatrix} \mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_{11}(\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1}\mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ \mathbf{K}_{21}(\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1}\mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \end{bmatrix} \end{aligned}$$

So

$$\begin{aligned} \mathbf{f}_1^{\text{new}} &= \mathbf{K}_{11}(\mathbf{I}_{11} + \mathcal{W}_{11}\mathbf{K}_{11})^{-1}\mathcal{W}_{11}\mathbf{f}_1 + \nabla \log p(\mathbf{y}_1|\mathbf{f}_1)) \\ \mathbf{f}_2^{\text{new}} &= \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{f}_1^{new} \end{aligned}$$

Appendix C

This appendix contains further details about detailed formulas and derivations of Chapter 5.

C.1 Change of support using Gaussian processes

This section contains more detailed formulas for the change of support using Gaussian processes that is defined in Chapter 5, Section 5.1.1.

Basics

$$\int \operatorname{erf}(x)dx = x\operatorname{erf}(x) + \frac{1}{\sqrt{\pi}}e^{-x^2} + C \tag{C.1}$$

$$\int \sigma^2 e^{-\frac{(x-x')^2}{\ell^2}}dx = \frac{\sigma^2\ell\sqrt{\pi}}{2}\operatorname{erf}\Big(\frac{(x-x')}{\ell}\Big) \tag{C.2}$$

$$\int_s^t \sigma^2 e^{-\frac{(x-x')^2}{\ell^2}}dx = \frac{\sigma^2\ell\sqrt{\pi}}{2}\operatorname{erf}\Big(\frac{(t-x')}{\ell}\Big) - \frac{\sigma^2\ell\sqrt{\pi}}{2}\operatorname{erf}\Big(\frac{(s-x')}{\ell}\Big) \tag{C.3}$$

$$\frac{\partial \operatorname{erf}(x)}{\partial x} = \frac{2}{\sqrt{\pi}}e^{-x^2} \tag{C.4}$$

$$\frac{\partial\big(\sigma^2 e^{\frac{-(x-x')^2}{\ell^2}}\big)}{\partial \ell} = \frac{2\sigma^2}{l^3}(x-x')^2 e^{-\frac{(x-x')^2}{\ell^2}} \tag{C.5}$$

Please note that the derivatives are w.r.t. the variance, that is why σ^2 is written in partial derivatives.

General formulas

We are using change of support because it finds the underlying function for block of data better than let's say RBF Smith et al. [2018]. For example in RBF for a block of data the average between the block is used, however, in change of support our aim is to find the

locations of the block more accurately and not only the estimated average. We use general notation to take integrals.

$$f(x_{b,1}, x_{b,2}) = \frac{1}{\Delta_x} \int_{x_{b,1}}^{x_{b,2}} u(x)dx, \tag{C.6}$$

where $u(x)$ is a latent stochastic process that we assume follows a Gaussian process with zero mean and covariance $k(x,x')$. This construction is usually known in the Geostatistic literature as the area-to-area interpolation problem or the *modified areal unit problem* (MAUP).
All the formulas and derivatives below are for the one dimensional case. The covariance is calculated as follows:

$$\begin{aligned}
&\operatorname{cov}[f(x_{a,1}, x_{a,2}), f(x_{b,1}, x_{b,2})] \qquad (C.7)\\
&= E[f(x_{a,1}, x_{a,2}) f(x_{b,1}, x_{b,2})] - E(f(x_{a,1}, x_{a,2}) E(f(x_{b,1}, x_{b,2}))\\
&= E[f(x_{a,1}, x_{a,2}) f(x_{b,1}, x_{b,2})]\\
&= E\Big[\int_{x_{b,1}}^{x_{b,2}} u(x)dx \int_{x_{a,1}}^{x_{a,2}} u(x')dx' \Big]\\
&= \int_{x_{b,1}}^{x_{b,2}} \int_{x_{a,1}}^{x_{a,2}} E[u(x)u(x')]dxdx'\\
K_{ff} &= \frac{1}{\Delta_x \Delta_{x'}} \int_{x_{b,1}}^{x_{b,2}} \int_{x_{a,1}}^{x_{a,2}} k(x,x')dxdx'.
\end{aligned}$$

For some forms of $k(x,x')$ it is possible to obtain an analytical expression for $k(x_a, x_b, x'_a, x'_b)$. For example, if $k(z,z')$ follows an Exponentiated-Quadratic (EQ) covariance form:

$$k(x,x') = \sigma^2 \exp\{-\frac{(x-x')^2}{2\ell^2}\}, \tag{C.8}$$

where σ^2 is the variance of the kernel and ℓ is the length-scale. Each of the double integrals take the following form:

$$\int_{x_{b,1}}^{x_{b,2}} \int_{x_{a,1}}^{x_{a,2}} \sigma^2 \exp\{-\frac{(x-x')^2}{2\ell^2}\}dxdx'. \tag{C.9}$$

The innermost integral in the indefinite form is:

$$\begin{aligned}\int e^{-\left(\frac{x^2}{2\ell}-\frac{x'}{\ell}x+\frac{(x')^2}{2\ell}\right)}dx &= \frac{1}{2}\sqrt{\frac{\pi}{\frac{1}{2\ell}}}e^{\frac{\left(\frac{x'}{2\ell}\right)^2-\frac{1}{2\ell}\frac{(x')^2}{2\ell}}{\frac{1}{2\ell}}}\operatorname{erf}\left(\sqrt{\frac{1}{2\ell}}x+\frac{-\frac{x'}{2\ell}}{\sqrt{\frac{1}{2\ell}}}\right) \\ &= \frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left(\sqrt{\frac{1}{2\ell}}x-\sqrt{2\ell}\frac{x'}{2\ell}\right) \\ &= \frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x-x')\right] \\ &= -\frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x'-x)\right].\end{aligned} \tag{C.10}$$

Using the above result, for the definite integral we get:

$$\begin{aligned}&\int_{x_{a,1}}^{x_{a,2}} e^{-\left(\frac{x^2}{2\ell}-\frac{x'}{\ell}x+\frac{(x')^2}{2\ell}\right)}dx \\ &\qquad= -\frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x'-x_{a,2})\right] \\ &\qquad\quad+\frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x'-x_{a,1})\right] \\ &\qquad= \frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x'-x_{a,1})\right] \\ &\qquad\quad-\frac{1}{2}\sqrt{2\ell\pi}\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x'-x_{a,2})\right].\end{aligned} \tag{C.11}$$

The outermost integral contains terms of the form:

$$\frac{1}{2}\sqrt{2\ell\pi}\int \operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x'-x_{a,1})\right]dx' = \frac{1}{2}\sqrt{2\ell\pi}\sqrt{2\ell}\int \operatorname{erf}(u)\,du, \tag{C.12}$$

where we have used $u = \sqrt{\frac{1}{2\ell}}(x' - x_{a,1})$. The indefinite integral above is equal to:

$$\begin{aligned}\frac{1}{2}\sqrt{2\ell\pi}\sqrt{2\ell}\int \operatorname{erf}(u)\,du &= \frac{1}{2}\sqrt{2\ell\pi}\sqrt{2\ell}\left[u\operatorname{erf}(u) + \frac{1}{\sqrt{\pi}}e^{-u^2}\right] \\ &= \frac{1}{2}\sqrt{2\ell\pi}\sqrt{2\ell}\left[\sqrt{\frac{1}{2\ell}}(x' - x_{a,1})\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x' - x_{a,1})\right]\right. \\ &\quad \left. + \frac{1}{\sqrt{\pi}}e^{-\left(\sqrt{\frac{1}{2\ell}}(x' - x_{a,1})\right)^2}\right] \\ &= \frac{1}{2}\sqrt{2\ell\pi}\left\{(x' - x_{a,1})\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x' - x_{a,1})\right]\right. \\ &\quad \left. + \sqrt{\frac{2\ell}{\pi}}e^{-\frac{(x' - x_{a,1})^2}{2\ell}}\right\}.\end{aligned} \tag{C.13}$$

By replacing u we get:

$$\begin{aligned}&\frac{1}{2}\sqrt{2\ell\pi}\int_{x_{b,1}}^{x_{b,2}} \operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x' - x_{a,1})\right]dx' \\ &\quad = \frac{1}{2}\sqrt{2\ell\pi}\left\{(x_{b,2} - x_{a,1})\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x_{b,2} - x_{a,1})\right]\right. \\ &\qquad \left. + \sqrt{\frac{2\ell}{\pi}}e^{-\frac{(x_{b,2} - x_{a,1})^2}{2\ell}}\right\} \\ &\quad - \frac{1}{2}\sqrt{2\ell\pi}\left\{(x_{b,1} - x_{a,1})\operatorname{erf}\left[\sqrt{\frac{1}{2\ell}}(x_{b,1} - x_{a,1})\right]\right. \\ &\qquad \left. + \sqrt{\frac{2\ell}{\pi}}e^{-\frac{(x_{b,1} - x_{a,1})^2}{2\ell}}\right\}.\end{aligned} \tag{C.14}$$

The double integral is now calculated as:

$$\int_{x_{b,1}}^{x_{b,2}} \int_{x_{a,1}}^{x_{a,2}} \exp\Big[-\frac{(x-x')^2}{2\ell}\Big] dx dx' \tag{C.15}$$

$$\begin{aligned}
&= \frac{1}{2}\sqrt{2\ell\pi}\left\{(x_{b,2}-x_{a,1})\text{erf}\left[\sqrt{\frac{1}{2\ell}}(x_{b,2}-x_{a,1})\right]\right. \\
&\qquad \left. + \sqrt{\frac{2\ell}{\pi}} e^{-\frac{(x_{b,2}-x_{a,1})^2}{2\ell}}\right\} \\
&- \frac{1}{2}\sqrt{2\ell\pi}\left\{(x_{b,1}-x_{a,1})\text{erf}\left[\sqrt{\frac{1}{2\ell}}(x_{b,1}-x_{a,1})\right]\right. \\
&\left. + \sqrt{\frac{2\ell}{\pi}} e^{-\frac{(x_{b,1}-x_{a,1})^2}{2\ell}}\right\} \\
&- \frac{1}{2}\sqrt{2\ell\pi}\left\{(x_{b,2}-x_{a,2})\text{erf}\left[\sqrt{\frac{1}{2\ell}}(x_{b,2}-x_{a,2})\right]\right. \\
&\qquad \left. + \sqrt{\frac{2\ell}{\pi}} e^{-\frac{(x_{b,2}-x_{a,2})^2}{2\ell}}\right\} \\
&+ \frac{1}{2}\sqrt{2\ell\pi}\left\{(x_{b,1}-x_{a,2})\text{erf}\left[\sqrt{\frac{1}{2\ell}}(x_{b,1}-x_{a,2})\right]\right. \\
&\qquad \left. + \sqrt{\frac{2\ell}{\pi}} e^{-\frac{(x_{b,1}-x_{a,2})^2}{2\ell}}\right\}.
\end{aligned}$$

For convenience, let us express $2\ell = \ell^2$ in the Equation (C.15). We then get:

$$\int_{x_{b,1}}^{x_{b,2}} \int_{x_{a,1}}^{x_{a,2}} \exp\Big[-\frac{(x-x')^2}{\ell^2}\Big] dx dx' \tag{C.16}$$

$$= \frac{\ell^2}{2}\Bigg\{\sqrt{\pi}\frac{(x_{b,2}-x_{a,1})}{\ell}\text{erf}\left[\frac{1}{\ell}(x_{b,2}-x_{a,1})\right] + e^{-\frac{(x_{b,2}-x_{a,1})^2}{\ell^2}}\Bigg\}$$

$$-\frac{\ell^2}{2}\Bigg\{\sqrt{\pi}\frac{(x_{b,1}-x_{a,1})}{\ell}\text{erf}\left[\frac{1}{\ell}(x_{b,1}-x_{a,1})\right] + e^{-\frac{(x_{b,1}-x_{a,1})^2}{\ell^2}}\Bigg\}$$

$$-\frac{\ell^2}{2}\Bigg\{\sqrt{\pi}\frac{(x_{b,2}-x_{a,2})}{\ell}\text{erf}\left[\frac{1}{\ell}(x_{b,2}-x_{a,2})\right] + e^{-\frac{(x_{b,2}-x_{a,2})^2}{\ell^2}}\Bigg\}$$

$$+\frac{\ell^2}{2}\Bigg\{\sqrt{\pi}\frac{(x_{b,1}-x_{a,2})}{\ell}\text{erf}\left[\frac{1}{\ell}(x_{b,1}-x_{a,2})\right] + e^{-\frac{(x_{b,1}-x_{a,2})^2}{\ell^2}}\Bigg\}.$$

We define $g(x)$ as $g(x) = x\sqrt{\pi}\text{erf}(x) + e^{-x^2}$.
Please note that $g(x)$ is an even function, if $g(x) = g(-x)$. We can then express the double integral above as:

$$\int_{x_{b,1}}^{x_{b,2}} \int_{x_{a,1}}^{x_{a,2}} \exp\Big[-\frac{(x-x')^2}{\ell^2}\Big] dx dx' \tag{C.17}$$

$$= \frac{\ell^2}{2}\Bigg\{g\left[\frac{(x_{b,2}-x_{a,1})}{\ell}\right] - g\left[\frac{(x_{b,1}-x_{a,1})}{\ell}\right] - g\left[\frac{(x_{b,2}-x_{a,2})}{\ell}\right] + g\left[\frac{(x_{b,1}-x_{a,2})}{\ell}\right]\Bigg\}.$$

The covariance is defined as:

$$k(x_a, x_b, x'_a, x'_b) \tag{C.18}$$
$$= \frac{\sigma^2 \ell^2}{2\Delta_x \Delta_{x'}} \times \left[g\Big(\frac{(x_{b,2} - x_{a,1})}{\ell}\Big) + g\Big(\frac{(x_{b,1} - x_{a,2})}{\ell}\Big) \right.$$
$$\left. - g\Big(\frac{(x_{b,1} - x_{a,1})}{\ell}\Big) - g\Big(\frac{(x_{b,2} - x_{a,2})}{\ell}\Big) \right].$$

The cross covariance is calculated as follows:

$$\begin{aligned} \text{cov}[f(s,t), u(t')] &= E[f(s,t)u(t')] - E(f(s,t)E(u(t')) \\ &= E[f(s,t)u(t')] \\ &= E\Big[\int_s^t u(z)\, dz\, u(t')\Big] \\ &= \int_s^t E[u(z)u(t')]dz \\ K_{fu} &= \int_s^t k(z,t')dz. \end{aligned} \tag{C.19}$$

By replacing k in the equation above we get:

$$\begin{aligned} K_{fu}((s,t),(t')) &= \sigma^2 \frac{\sqrt{\pi}\ell}{2} \Big[\text{erf}(\frac{(t'-s)}{\ell}) - \text{erf}(\frac{(t'-t)}{\ell}) \Big] \\ &= \sigma^2 \frac{\sqrt{\pi}\ell}{2} \Big[\text{erf}(\frac{(t-t')}{\ell}) + \text{erf}(\frac{(t'-s)}{\ell}) \Big]. \end{aligned} \tag{C.20}$$

We calculate the gradient of the objective function (negative log marginal likelihood) w.r.t. hyper-parameters to find the optimized parameters.

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial K} \frac{\partial K}{\partial \theta} \tag{C.21}$$

The RBF kernel has two hyper-parameters, so we will take derivative of the output kernel w.r.t. the lenghscale and variance. We define $h(z) = \frac{z\sqrt{\pi}}{2}\text{erf}(z) + e^{-z^2}$,

$$\frac{\partial K_{ff}((s,t),(s',t'))}{\partial \ell} = \sigma^2 \ell \times \left[h\left(\frac{(t-s')}{\ell}\right) + h\left(\frac{(t'-s)}{\ell}\right) \right.$$
$$\left. - h\left(\frac{(t-t')}{\ell}\right) - h\left(\frac{(s-s')}{\ell}\right) \right], \tag{C.22}$$

$$\frac{\partial K_{ff}((s,t),(s',t'))}{\partial \sigma^2} = \frac{\ell^2}{2} \times \left[g\Big(\frac{(t-s')}{\ell}\Big) + g\Big(\frac{(s-t')}{\ell}\Big) - g\Big(\frac{(t-t')}{\ell}\Big) - g\Big(\frac{(s-s')}{\ell}\Big) \right]. \tag{C.23}$$

The covariance between f and u and corresponding derivatives w.r.t length-scale and variance are defined as:

$$K_{fu} = \sigma^2 \frac{\sqrt{\pi}\ell}{2} \Big(\text{erf}(\frac{(t-t')}{\ell}) + \text{erf}(\frac{(t'-s)}{\ell}) \Big) \tag{C.24}$$

$$\frac{\partial K_{fu}}{\partial \ell} = \frac{\partial}{\partial \ell} \Big(\sigma^2 \frac{\sqrt{\pi}\ell}{2} \Big[\text{erf}(\frac{(t-t')}{\ell}) \Big] + \sigma^2 \frac{\sqrt{\pi}\ell}{2} \Big[\text{erf}(\frac{(t'-s)}{\ell}) \Big] \Big) \tag{C.25}$$

$$\begin{aligned}
\frac{\partial K_{fu}}{\partial \ell_a} &= \sigma^2 \frac{\sqrt{\pi}}{2} \Big[\text{erf}(\frac{(t-t')}{\ell}) \Big] + \sigma^2 \frac{\sqrt{\pi}\ell}{2} \times -(\frac{(t-t')}{\ell^2}) \times \frac{2}{\sqrt{\pi}} e^{-(\frac{(t-t')}{\ell})^2} \\
&= \sigma^2 \frac{\sqrt{\pi}}{2} \Big[\text{erf}(\frac{(t-t')}{\ell}) \Big] - \sigma^2 \frac{(t-t')}{\ell} e^{-(\frac{(t-t')}{\ell})^2} \\
\frac{\partial K_{fu}}{\partial \ell_b} &= \sigma^2 \frac{\sqrt{\pi}}{2} \Big[\text{erf}(\frac{(t'-s)}{\ell}) \Big] + \sigma^2 \frac{\sqrt{\pi}\ell}{2} \times -(\frac{(t'-s)}{\ell^2}) \times \frac{2}{\sqrt{\pi}} e^{-(\frac{(t'-s)}{\ell})^2} \\
&= \sigma^2 \frac{\sqrt{\pi}}{2} \Big[\text{erf}(\frac{(t'-s)}{\ell}) \Big] - \sigma^2 \frac{(t'-s)}{\ell} e^{-(\frac{(t'-s)}{\ell})^2}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial K_{fu}}{\partial \ell} &= \sigma^2 \Big[\frac{\sqrt{\pi}}{2} \text{erf}(z_1) - z_1 e^{-(z_1)^2} + \frac{\sqrt{\pi}}{2} \text{erf}(z_2) - z_2 e^{-(z_2)^2} \Big] \\
&= \sigma^2 \Big[h'(\frac{t-t'}{\ell}) + h'(\frac{t'-s}{\ell}) \Big],
\end{aligned} \tag{C.26}$$

where $z_1 = \frac{(t-t')}{\ell}$ and $z_2 = \frac{(t'-s)}{\ell}$ and $h'(z) = \frac{\sqrt{\pi}}{2}\text{erf}(z) - z e^{-z^2}$.

$$\frac{\partial K_{fu}}{\partial \sigma^2} = \frac{\sqrt{\pi}\ell}{2} \Big(\text{erf}(\frac{(t-t')}{\ell}) + \text{erf}(\frac{(t'-s)}{\ell}) \Big). \tag{C.27}$$

The derivatives w.r.t length-scale and variance for K_{uu} are defined as:

$$\frac{\partial K_{uu}(x,x')}{\partial \ell} = \frac{2\sigma^2}{\ell} \frac{(x-x')^2}{\ell^2} e^{-\frac{(x-x')^2}{\ell^2}} \tag{C.28}$$

$$\frac{\partial K_{uu}(x,x')}{\partial \sigma^2} = e^{-\frac{(x-x')^2}{\ell^2}}. \tag{C.29}$$

The derivatives of K_{fu} and K_{uu} w.r.t inducing inputs z are defined as:

$$\frac{\partial K_{fu}((s,t),(z))}{\partial z} = \sigma^2\Big(-e^{-\frac{(t-z)^2}{\ell^2}} + e^{-\frac{(z-s)^2}{\ell^2}}\Big) \tag{C.30}$$

$$\frac{\partial K_{uu}(t,t'(z))}{\partial z} = 2\sigma^2\frac{(t-z)}{\ell^2}e^{-\frac{(t-z)^2}{\ell^2}}. \tag{C.31}$$

C.2 Gauss-Hermite quadrature

Gauss-Hermite quadrature is a form of Gaussian quadrature for approximating the value of univariate or bivariate integrals over Gaussian distributed variables. For example, in a general case we have:

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)dx \;\approx\; \sum_{s=1}^{S} w_s f(x_s), \tag{C.32}$$

where S is the number of sample points used and w_s are associated weights defined as:

$$w_s = \frac{2^{s-1}s!\sqrt{\pi}}{s^2[H_{s-1}(x_s)]^2}, \tag{C.33}$$

x_s are the roots of Hermitian polynomial $H_S(x)(s = 1,2,\ldots,S)$, $H_S(x)$ is defined as:

$$H_S(x) = (-1)^S e^{x^2}\frac{d^S}{dx^S}e^{-x^2}. \tag{C.34}$$

For solving integrals such as $\int p(x)f(x)dx$ with $p(x) = \mathcal{N}(x|\mu,\sigma)$, the expectation of a function w.r.t a Gaussian distribution is defined as:

$$\mathbb{E}_{p(x)}\Big[f(x)\Big] = \int_{-\infty}^{+\infty} p(x)f(x)dx \tag{C.35}$$

$$= \int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)f(x)dx, \tag{C.36}$$

since the equation above does not correspond to the Hermite polynomial, we need change of variables:

$$y = \frac{x-\mu}{\sigma\sqrt{2}} \rightarrow x = \sqrt{2}\sigma y + \mu. \tag{C.37}$$

$$\mathbb{E}_{p(x)}\Big[f(x)\Big] = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{\pi}} \exp(-y_s^2) f(\sqrt{2}\sigma y_s + \mu) dy \tag{C.38}$$

$$\approx \frac{1}{\sqrt{\pi}} \sum_{s=1}^{S} w_s f(\sqrt{2}\sigma y_s + \mu). \tag{C.39}$$

As number of samples S increases, the approximation gets more accurate. For the Equation (5.10), the above calculation becomes:

$$\mathbb{E}_{q(\mathbf{f}_d)}\Big[\log p(\mathbf{y}_d|\mathbf{f}_d)\Big] \approx \frac{1}{\sqrt{\pi}} \sum_{s=1}^{S} w_s \log p(\mathbf{y}_d|\sqrt{2}\sigma_{q(\mathbf{f})}\mathbf{f}_s + \mu_{q(\mathbf{f})}), \tag{C.40}$$

where the mean $\mu_{q(\mathbf{f})}$ and variance $\sigma_{q(\mathbf{f})}$ of the variational distribution $q(\mathbf{f})$ are introduced in Equation (5.11).

C.3 Derivatives w.r.t variational parameters

We need to take the derivative of the bound w.r.t. mean ($\boldsymbol{\mu}$) and covariance ($\mathbf{S}$). We re-write the variational bound in Equation (5.10):

$$\mathcal{L} = \sum_{d=1}^{D} \sum_{j=1}^{N_d} \mathbb{E}_{q(\mathbf{f})}\left[\log p(y_d(\upsilon_{d,j})|\mathbf{f}_d(\upsilon_{d,j}))\right] - \sum_{q=1}^{Q} \mathrm{KL}(q(\mathbf{u}_q)\|p(\mathbf{u}_q)). \tag{C.41}$$

The derivative of the log marginal likelihood w.r.t mean ($\boldsymbol{\mu}$) The mean μ is composed of the concatenation of all the μ_q. So we need only to compute the derivatives w.r.t. μ_q.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}_q} \log p(\mathbf{Y}) &= \frac{\partial}{\partial \boldsymbol{\mu}_q} \sum_{d=1}^{D} \sum_{j=1}^{N_d} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_d(\upsilon_{d,j})|\mathbf{f}_d(\upsilon_{d,j}))\right] - \sum_{q=1}^{Q} \mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q)) \\ &= \sum_{d=1}^{D} \sum_{j=1}^{N_d} \underbrace{\frac{\partial}{\partial \boldsymbol{\mu}_q} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_d(\upsilon_{d,j})|\mathbf{f}_d(\upsilon_{d,j}))\right]}_{\text{VE}} - \sum_{q=1}^{Q} \underbrace{\frac{\partial}{\partial \boldsymbol{\mu}_q} \mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q))}_{\text{KL}}.\end{aligned}$$

The derivative of KL part in equation above w.r.t mean ($\boldsymbol{\mu}$):

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}_q}\mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q)) &= \frac{\partial}{\partial \boldsymbol{\mu}_q}\frac{1}{2}\Big\{\mathrm{tr}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q) + (\mathbf{0}-\boldsymbol{\mu}_q)^T\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}(\mathbf{0}-\boldsymbol{\mu}_q) - M \\ &\qquad + \log\left|\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}\right| - \log\left|\mathbf{S}_q\right|\Big\} \\ &= \frac{\partial}{\partial \boldsymbol{\mu}_q}\frac{1}{2}\{\boldsymbol{\mu}_q^T\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\boldsymbol{\mu}_q\} = \frac{1}{2}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1} + \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1^T})\boldsymbol{\mu}_q \\ &= \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\boldsymbol{\mu}_q.\end{aligned}$$

Please note that we use the notation $p(y|f)$ instead of $p(y(\upsilon)|f(\upsilon))$ in the below equations. The derivative of VE part in equation above w.r.t mean ($\boldsymbol{\mu}$) is as follows:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}_q}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] &= \frac{\partial}{\partial \boldsymbol{\mu}_q}\mathbb{E}_{\mathcal{N}(\mathbf{f}_{d,j}|\mathbf{m}_{d,j},\mathbf{v}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] \\ &= \frac{\partial}{\partial \mathbf{m}_{d,j}}\mathbb{E}_{\mathcal{N}(\mathbf{f}_{d,j}|\mathbf{m}_{d,j},\mathbf{v}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]\frac{\partial \mathbf{m}_{d,j}}{\partial \boldsymbol{\mu}_q} \\ &= \mathbb{E}_{q(\mathbf{f}_{d,j})}\Bigg[\underbrace{\frac{\partial}{\partial \mathbf{f}_{d,j}}\log p(y_{d,j}|\mathbf{f}_{d,j})}_{\text{See likelihood section C.5}}\Bigg]\frac{\partial \mathbf{m}_{d,j}}{\partial \boldsymbol{\mu}_q}.\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathbf{m}_{d,j}}{\partial \boldsymbol{\mu}_q} &= \frac{\partial}{\partial \boldsymbol{\mu}_q}\sum_{q=1}^{Q}\mathbf{K}_{\mathbf{f}_{d,j}\mathbf{u}_q}\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\boldsymbol{\mu}_q = \frac{\partial}{\partial \boldsymbol{\mu}_q}\mathbf{K}_{\mathbf{f}_{d,j}\mathbf{u}_q}\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\boldsymbol{\mu}_q \\ &= \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{K}_{\mathbf{u}_q\mathbf{f}_{d,j}}.\end{aligned} \tag{C.42}$$

The derivative of the log marginal likelihood w.r.t covariance (S)

$$\begin{aligned}\frac{\partial}{\partial \mathbf{S}_q}\log p(\mathbf{Y}) &= \frac{\partial}{\partial \mathbf{S}_q}\sum_{d=1}^{D}\sum_{j=1}^{N_d}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_d(\upsilon_{d,j})|f_d(\upsilon_{d,j}))\right] - \sum_{q=1}^{Q}\mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q)) \\ &= \sum_{d=1}^{D}\sum_{j=1}^{N_d}\underbrace{\frac{\partial}{\partial \mathbf{S}_q}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_d(\upsilon_{d,j})|f_d(\upsilon_{d,j}))\right]}_{\text{VE}} - \sum_{q=1}^{Q}\underbrace{\frac{\partial}{\partial \mathbf{S}_q}\mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q))}_{\text{KL}}.\end{aligned}$$

The derivative of KL part in equation above w.r.t covariance ($\mathbf{S}$):

$$\begin{aligned}\frac{\partial}{\partial \mathbf{S}_q}\mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q)) &= \frac{\partial}{\partial \mathbf{S}_q}\frac{1}{2}\Big\{\mathrm{tr}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q) + (\mathbf{0}-\boldsymbol{\mu}_q)^T\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}(\mathbf{0}-\boldsymbol{\mu}_q) - M \\ &\qquad + \log\left|\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}\right| - \log\left|\mathbf{S}_q\right|\Big\} \\ &= \frac{\partial}{\partial \mathbf{S}_q}\frac{1}{2}\Big(\mathrm{tr}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q) - \log(\left|\mathbf{S}_q\right|\Big) \\ &= \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1} - \frac{1}{2}\mathrm{diag}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}) - \frac{1}{2}\mathbf{S}_q^{-1}.\end{aligned}$$

The derivative of VE part in equation above w.r.t covariance ($\mathbf{S}$):

$$\begin{aligned}\frac{\partial}{\partial \mathbf{S}_q}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] &= \frac{\partial}{\partial \mathbf{S}_q}\mathbb{E}_{\mathcal{N}(\mathbf{f}_{d,j}|\mathbf{m}_{d,j},\mathbf{v}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] \\ &= \frac{\partial}{\partial \mathbf{v}_{d,j}}\mathbb{E}_{\mathcal{N}(\mathbf{f}_{d,j}|\mathbf{m}_{d,j},\mathbf{v}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{S}_q} \\ &= \frac{1}{2}\mathbb{E}_{q(\mathbf{f}_{d,j})}\Bigg[\underbrace{\frac{\partial^2}{\partial \mathbf{f}_{d,j}^2}\log p(y_{d,j}|\mathbf{f}_{d,j})}_{\text{See likelihood section C.5}}\Bigg]\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{S}_q},\end{aligned} \tag{C.43}$$

where,

$$\begin{aligned}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{S}_q} &= \frac{\partial}{\partial \mathbf{S}_q}\Big(\text{constant} - \sum_{q=1}^{Q}\mathbf{K}_{\mathbf{f}_{d,j}\mathbf{u}_q}\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{K}_{\mathbf{u}_q\mathbf{f}_{d,j}}\Big) \\ &= -\frac{\partial}{\partial \mathbf{S}_q}\mathbf{K}_{\mathbf{f}_{d,j}\mathbf{u}_q}\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{K}_{\mathbf{u}_q\mathbf{f}_{d,j}} \\ &= -\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{K}_{\mathbf{u}_q\mathbf{f}_{d,j}}\mathbf{K}_{\mathbf{f}_{d,j}\mathbf{u}_q}\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}.\end{aligned} \tag{C.44}$$

C.4 Derivatives w.r.t. hyper-parameters

Derivatives w.r.t. hyper-parameters: $\{\mathbf{Z}, \theta\}$

We need to compute derivatives of the variational bound with respect to inducing inputs ($\mathbf{Z}$) and kernel hyper-parameters (θ):

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}}, \frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} \text{ and } \frac{\partial \mathcal{L}}{\partial \mathrm{diag}(\mathbf{K}_{\mathbf{f}_d\mathbf{f}_d})}. \tag{C.45}$$

The derivative of variational bound w.r.t $\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} = \sum_{d=1}^{D}\sum_{j=1}^{N_d} \underbrace{\frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{VE}} - \sum_{q=1}^{Q} \underbrace{\frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} \mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q))}_{\text{KL}}.$$

The derivative of KL part w.r.t $\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} \mathrm{KL}(q(\mathbf{u}_q)|p(\mathbf{u}_q)) &= \frac{1}{2}\Bigg[\frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}}\mathrm{tr}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q) + \frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}}(\boldsymbol{\mu}_q\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\boldsymbol{\mu}_q) \\ &\quad + \frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}}\log|\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}|\Bigg] \\ &= \frac{1}{2}\Bigg[-(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1}\mathbf{S}_q\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1})^T - (\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1})^T\boldsymbol{\mu}_q\boldsymbol{\mu}_q^T(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1})^T \\ &\quad + (\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}^{-1})^T\Bigg].\end{aligned} \tag{C.46}$$

The derivative of VE part w.r.t $\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] &= \frac{\partial}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} f(\mathbf{m}_{d,j}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}), \mathbf{v}_{d,j}(\mathbf{K}_{\mathbf{u}_q\mathbf{u}_q})) \\ &= \frac{\partial f}{\partial \mathbf{m}_{d,j}}\frac{\partial \mathbf{m}_{d,j}}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} + \frac{\partial f}{\partial \mathbf{v}_{d,j}}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} \\ &= \underbrace{\frac{\partial}{\partial \mathbf{m}_{d,j}} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{see likelihood section C.5}}\frac{\partial \mathbf{m}_{d,j}}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}} \\ &\quad + \underbrace{\frac{\partial}{\partial \mathbf{v}_{d,j}} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{see likelihood section C.5}}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{K}_{\mathbf{u}_q\mathbf{u}_q}}.\end{aligned} \tag{C.47}$$

The derivative of variational bound w.r.t $\mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} = \sum_{d=1}^{D}\sum_{j=1}^{N_d} \underbrace{\frac{\partial}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} \mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{VE}}. \tag{C.48}$$

The derivative of VE part w.r.t $\mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] &= \frac{\partial}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} f\left(\mathbf{m}_{d,j}(\mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}), \mathbf{v}_{d,j}(\mathbf{K}_{\mathbf{f}_d\mathbf{u}_q})\right) \\ &= \frac{\partial f}{\partial \mathbf{m}_{d,j}}\frac{\partial \mathbf{m}_{d,j}}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} + \frac{\partial f}{\partial \mathbf{v}_{d,j}}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} \\ &= \underbrace{\frac{\partial}{\partial \mathbf{m}_{d,j}}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{see likelihood section C.5}}\frac{\partial \mathbf{m}_{d,j}}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}} \\ &\quad + \underbrace{\frac{\partial}{\partial \mathbf{v}_{d,j}}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{see likelihood section C.5}}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{K}_{\mathbf{f}_d\mathbf{u}_q}}.\end{aligned} \tag{C.49}$$

The derivative of variational bound w.r.t $\mathbf{K}_{\text{diag}}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}_{ff}} = \sum_{d=1}^{D}\sum_{j=1}^{n_d}\frac{\partial}{\partial \mathbf{K}_{\text{diag}}}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right].$$

The derivative of VE part w.r.t $\mathbf{K}_{\text{diag}}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{K}_{\text{diag}}}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right] &= \frac{\partial}{\partial \mathbf{K}_{\text{diag}}} f(\mathbf{v}_{d,j}(\mathbf{K}_{\text{diag}})) = \frac{\partial f}{\partial \mathbf{v}_{d,j}}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{K}_{\text{diag}}} \\ &= \underbrace{\frac{\partial}{\partial \mathbf{v}_{d,j}}\mathbb{E}_{q(\mathbf{f}_{d,j})}\left[\log p(y_{d,j}|\mathbf{f}_{d,j})\right]}_{\text{see likelihood section C.5}}\frac{\partial \mathbf{v}_{d,j}}{\partial \mathbf{K}_{\text{diag}}}.\end{aligned}$$

C.5 Likelihoods

In the experiments' section in Chapter 5, we used different likelihoods such as Poisson, Gaussian and heteroscedastic Gaussian. for more information on various likelihoods please refer to Moreno-Muñoz et al. [2018]. We can calculate the expected value in the bound in closed form for the Gaussian likelihood, however, for other likelihoods a numerical approximation such as Gauss-Hermite quadrature is needed and is explained in C.2. For allowing a heterogenous likelihood $p(y_d|f_d)$, we need to compute the first and second order derivatives for the VE part of the log likelihood.

Poisson likelihood: The Poisson likelihood is for modeling the number of times an event occurs in an interval of time or space. A discrete random variable y is said to have a Poisson distribution with parameter $\lambda > 0$, if for different number of occurrences k, the probability

mass function of y is defined as:

$$P(y=k) = \frac{\lambda^k e^{-\lambda}}{k!},$$

where e is the Euler's number ($e = 2.71828\ldots$) and λ is equal:

$$\lambda(x) = \exp(f(x)).$$

The equation above is the link transformation between latent parameters f and Poisson likelihood.

Heteroscedastic Gaussian likelihood has a link transformation between latent parameters f and the likelihood with mean and variance defined as:

$$\mu(x) = f_1, \sigma(x) = \exp(f_2),$$

where f_1 and f_2 both follow a GP. **Gaussian likelihood** has a link transformation between latent parameters function f and the likelihood with mean and variance defined as:

$$\mu(x) = f, \sigma,$$

where the latent parameter function only models the mean, while the variance is a hyper-parameter that does not follow a GP.