Enabling feature-level interpretability in non-linear latent variable models: a synthesis of statistical and machine learning techniques



Kaspar Märtens

St Peter's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2019

Acknowledgements

The last four years have gone very quickly: how time flies when you are having fun! I am fortunate to have spent these four years in a great environment, lucky to be surrounded by brilliant and supportive friends and colleagues.

First and foremost, I would like to thank Chris Yau, who has not only been an inspiring supervisor but also a mentor and a great companion on this journey through my DPhil. The path towards this thesis has been enjoyable and I have learnt a lot on the way. I am thankful for his advice, inspiration, and for supporting my development as an independent researcher. I also thank him for various motivational messages I have received throughout my DPhil, including reminders to wake up, and how "the paper is not going to write itself". I extend my gratitude to Chris Holmes for his continuous encouragement, support, and insights.

I am glad that I had the opportunity to spend three months on a placement in Prowler.io. I would like to thank Nicolas Durrande for pointing me to his earlier work which ended up being very useful for this thesis.

I have always been lucky to have excellent supervisors. It was my early BSc project with Raivo that got me excited about research, and later I was fortunate to work with Leopold from whom I had so much to learn.

I am thankful to many friends, both in Estonia and Oxford, for great company and good times. I want to thank Andi, Dominic, Emilien, Giuseppe, Jef, Leon, Luke, Paul, and many others for useful and enjoyable discussions, let it be related to science or anything else. I would like to thank Tanel for all the cool projects we have done together, and Kristel for all the advice. I am also thankful to Barbara, Cornelia and T. for being the best flatmates, and for making sure that I finish this thesis on time. Special thanks go to Kasia for enriching the final part of my DPhil, and for her support. Finally, I am indebted to my family who have supported my seemingly endless path of education: Aitäh emale, isale, vennale ja vanaemale.

Abstract

Gaining insights into complex high-dimensional data is challenging and typically requires the use of dimensionality reduction methods. These methods let us identify low-dimensional structures embedded within the data that may reveal patterns of interest. In probabilistic models, such low-dimensional structures are captured via *latent variables*.

In biomedical applications, e.g. in computational biology, it is common to use *linear* dimensionality reduction approaches like probabilistic PCA due to their interpretability. In machine learning, however, recently there has been substantial interest in non-linear "black box" latent variable models because of their improved predictive capabilities. In this thesis, we build upon and propose extensions to two non-linear dimensionality reduction frameworks: the Gaussian Process Latent Variable Model (GPLVM) and the Variational Autoencoder (VAE). The former is based on Gaussian Processes, whereas the latter utilises neural networks. It would be desirable if we could combine some interpretable aspects of linear models with the flexibility of modern non-linear probabilistic methods.

The goal of our work is to enable *feature-level* interpretability within non-linear latent variable models, while also incorporating covariate information. As opposed to post-hoc explainability, we aim to directly construct *decomposable* non-linear models. By combining ideas from classical statistics and embedding these within latent variable models, we investigate two different perspectives on how to aid feature-level transparency. First, we propose approaches to characterise and quantify the contribution of a particular latent variable (or a covariate) for every feature. We achieve this by building upon the notion of *functional ANOVA* decompositions from classical statistics, and embedding these decompositions within the GPLVM and the VAE frameworks. Second, we investigate another approach towards improving feature-level interpretability, by providing a mechanism to answer the question "which features exhibit similar patterns" in the VAE framework. We propose to do this by introducing probabilistic clustering structure as part of the decoder network. We demonstrate the utility of our proposed methods on various toy and high-dimensional genomics data sets.

Contents

1 Introduction and Background **1**

1.1 Supervised and Unsupervised Learning 3

1.2 Bayesian Machine Learning 5

1.3 Latent Variable Models 6

1.3.1 Motivation for Latent Variable Modelling 6

1.3.2 Discrete and continuous latent variables 8

1.3.3 Latent Variable Models for dimensionality reduction 9

1.4 Bayesian Inference 11

1.4.1 Monte Carlo methods 12

1.4.2 Variational Inference 14

1.4.2.1 Classical Variational Inference 15

1.4.2.2 Modern adaptations of Variational Inference 16

1.4.2.3 Amortised VI 17

1.5 Gaussian Processes 18

1.5.1 Distributions over functions with GPs 18

1.5.2 Gaussian Processes for Regression 19

1.5.3 Gaussian Processes for dimensionality reduction 20

1.5.4 Scalable inference for GPs 22

1.6 Variational Autoencoders 24

1.6.1 Example application: The Cancer Patient Experience Survey 27

1.7 Interpretability 29

1.8 Goals and contributions 30

2 Covariate Gaussian Process Latent Variable Models 35

2.1 Introduction . . . 35

2.2 Covariate-adjusted dimensionality reduction . . . 36

2.3 Covariate-GPLVM . . . 38

2.3.1 Kernel design . . . 40

2.3.2 Inference for c-GPLVM . . . 42

2.4 Partially observed and censored covariates . . . 43

2.4.1 Missing categorical covariates . . . 43

2.4.2 Censored covariates . . . 45

2.5 Experiments . . . 46

2.5.1 Synthetic data: Circles and Pinwheel . . . 47

2.5.2 MNIST . . . 49

2.5.3 Censored survival times: synthetic data . . . 51

2.5.4 Real-world data: The Cancer Genome Atlas . . . 52

2.5.4.1 Prediction of survival times . . . 53

2.5.4.2 Survival-adjusted cancer modelling . . . 53

2.6 Related work . . . 55

2.7 Discussion . . . 56

3 Decomposing feature-level variation with Covariate Gaussian Process Latent Variable Models 57

3.1 Introduction . . . 57

3.2 Background: Functional ANOVA decompositions . . . 60

3.2.1 Classical ANOVA . . . 60

3.2.2 Functional ANOVA . . . 61

3.2.3 Identifiable functional ANOVA . . . 62

3.3 Decomposable c-GPLVM . . . 64

3.3.1 Motivation . . . 64

3.3.2 Identifiable decomposition with Gaussian Processes . . . 64

3.3.3 Inference . . . 68

3.4 Experiments . . . 70

3.4.1 Synthetic data: Circles and Pinwheel 70
3.4.2 TCGA data 71
3.5 Related work 73
3.6 Discussion 73

4 Neural Decomposition: Functional ANOVA with Variational Autoencoders 75
4.1 Introduction 75
4.2 Identifiable Neural Decomposition 77
4.2.1 Decomposing Neural Network Regression 78
4.2.2 Inference under integral constraints 79
4.2.3 Decomposing CVAEs 82
4.2.4 Sparse Neural Decomposition 83
4.3 Experiments 84
4.3.1 Comparison with Decomposable c-GPLVM 85
4.3.2 Synthetic data 87
4.3.3 Decomposing CelebA 90
4.3.3.1 Pixel-level decomposition 90
4.3.3.2 Improved generalisation 91
4.3.4 Gene expression 92
4.4 Related work 96
4.5 Discussion 97

5 BasisVAE 99
5.1 Introduction 99
5.2 BasisVAE 102
5.2.1 Translation and scale invariance 103
5.2.2 Specifying the number of basis functions 105
5.2.3 Generative model and inference 106
5.2.3.1 Non-collapsed inference 107
5.2.3.2 Collapsed inference 107
5.3 Related work 110

5.4 Experiments 111
5.4.1 Toy data 111
5.4.2 Motion capture data 113
5.4.3 Single cell Spermatogenesis data 115
5.5 Discussion 118
5.5.1 On well-definedness of translation-invariant BasisVAE 119
5.5.2 Future work 121

6 Conclusions **122**
6.1 Summary of contributions 123
6.2 Future directions 124

Appendices

A Synthetic data generation mechanisms **127**
A.1 Synthetic data generative mechanism 127
A.2 Synthetic data generative mechanism for the batch effect 128

References **129**

1

Introduction and Background

Contents

1.1 Supervised and Unsupervised Learning 3
1.2 Bayesian Machine Learning 5
1.3 Latent Variable Models 6
1.3.1 Motivation for Latent Variable Modelling 6
1.3.2 Discrete and continuous latent variables 8
1.3.3 Latent Variable Models for dimensionality reduction . . 9
1.4 Bayesian Inference 11
1.4.1 Monte Carlo methods 12
1.4.2 Variational Inference 14
1.4.2.1 Classical Variational Inference 15
1.4.2.2 Modern adaptations of Variational Inference . 16
1.4.2.3 Amortised VI 17
1.5 Gaussian Processes 18
1.5.1 Distributions over functions with GPs 18
1.5.2 Gaussian Processes for Regression 19
1.5.3 Gaussian Processes for dimensionality reduction 20
1.5.4 Scalable inference for GPs 22
1.6 Variational Autoencoders 24
1.6.1 Example application: The Cancer Patient Experience Survey 27
1.7 Interpretability 29
1.8 Goals and contributions 30

Machine learning aims to create algorithms that *learn from data*. The idea is to build computer systems that utilise patterns in the observed data in order to make future predictions. For example, given electronic health records and genomics data

for a large cohort, we could aim to build a machine learning system to predict an individual's disease risk for a common disease (such as type 2 diabetes).

The idea to learn from data, however, is not new, and in fact is closely related to *statistical modelling*. While the fields of statistics and machine learning are closely related, they generally focus on different aspects of modelling and they have different goals: statistics generally focuses on careful modelling with the goal to draw conclusions about certain phenomena of interest, whereas machine learning is typically concerned with *prediction* (Breiman, 2001). For example, in the context of personalised medicine, answering the research question "which genes are associated with an increased disease risk" would fall under the former paradigm, whereas "prediction of future acute kidney injury using adverse events from electronic health records" has been formulated as a predictive problem. Studying the former type of questions would typically rely on relatively simple interpretable models (e.g. involving statistical hypothesis testing in the context of linear models), whereas when we are purely interested in predictive performance, it may sometimes be preferable to use flexible non-linear "black box" machine learning approaches such as neural networks.

While non-linear models (such as neural networks) can provide high predictive performance, they lack *interpretability*. However, being able to *explain* what the model has learnt is crucial for various high-stake application domains, but it is also important for scientific discovery applications. For example, in computational biology applications where researchers apply non-linear modelling techniques for exploratory data analysis, it is important to understand what the model has learnt. One strategy towards interpretable non-linear models is to rely on simpler (e.g. linear or tree-based) models for interpretation. In this thesis, however, we explore a different avenue. By borrowing ideas from classical statistics, we aim to build *decomposable* non-linear models that directly aid interpretability without the need for post-hoc explanations.

Another desirable property of any machine learning system is the ability to capture *uncertainty*, e.g. the system should know when it does not know. This is especially important for biomedical applications. For example, when a patient belongs to a subpopulation that is severely under-represented in the training data, the model's predictions are likely to be highly uncertain – this is important information especially in scenarios where we rely on this model for diagnosis or treatment. A natural way to introduce uncertainty into model predictions is via the notion of *Bayesian inference*, a principled and coherent framework to reason about uncertainty (Gelman et al., 2013). This is achieved by treating everything unknown (such as model parameters and unobserved quantities) as random variables, so they are represented via probability distributions. The practical need for uncertainty estimates, combined with the elegance and modularity of the Bayesian modelling framework have led to the sub-field of *probabilistic machine learning* (or *Bayesian machine learning*) becoming increasingly popular (Ghahramani, 2015). All models that we consider and develop in this thesis will fall under the paradigm of probabilistic machine learning, being either partially or fully Bayesian.

Next, we review the basics of probabilistic machine learning, with a focus on Bayesian latent variable models. This class of models is of particular interest to us, as it employs latent variables to capture hidden structure in the data. Then we cover two particular classes of latent variable models in detail: in section 1.5 we review Gaussian Processes for dimensionality reduction, and in section 1.6 we focus on Variational Autoencoders. In the last section of this introductory chapter, we discuss some motivating examples and outline the research questions that underpin the rest of this thesis.

1.1 Supervised and Unsupervised Learning

A common use-case for machine learning is prediction in the setting where we have observed input-output pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ with inputs $\mathbf{x}_i \in \mathcal{X}$ and outputs $\mathbf{y}_i \in \mathcal{Y}$ (typically class labels or a continuous-valued outcome), and our goal is to learn

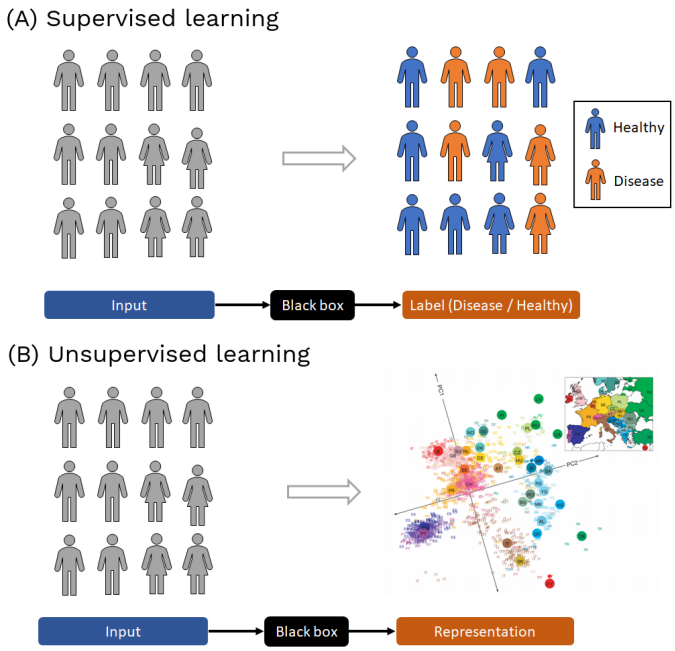


Figure 1.1: Illustration: (A) supervised learning is concerned with predicting an outcome (e.g. predicting disease status from molecular measurements), whereas (B) unsupervised learning aims to uncover low-dimensional structure in the data (e.g. genetic variation in Europeans exhibits structure that resembles the geographical map of Europe (Novembre et al., 2008)).

some mapping $f : \mathbf{x} \mapsto \mathbf{y}$ that we can use for out-of-sample prediction. Given a new data point with inputs $\mathbf{x}^*$, we would like to predict the corresponding output $\mathbf{y}^*$ value using the learnt mapping $\hat{\mathbf{y}^*} := f(\mathbf{x}^*)$. This task is known as *supervised learning* (Murphy, 2012). When $\mathcal{Y}$ is discrete, we refer to it as *classification*, for continuous-valued response we call it *regression* (or *multi-output regression* in case of multi-dimensional $\mathbf{y}_i \in \mathbb{R}^P$). The choice of the function class $f \in \mathcal{F}$ reflects (either explicitly or implicitly) our modelling assumptions, e.g. f could be represented using linear models, decision trees, neural networks etc. For example, if $\mathbf{x}_i$ are image data, then by choosing to use convolutional neural networks our model captures spatial structure between nearby pixels. In probabilistic models, supervised learning is formulated by specifying $p_\theta(\mathbf{y}|\mathbf{x})$.

Now we consider *unsupervised learning*, where we have (unlabelled) observed data $\{\mathbf{y}_i\}_{i=1}^N$ where often $\mathbf{y}_i \in \mathbb{R}^P$. Unsupervised learning can have various goals, such as

- density estimation, i.e. estimating $p_\theta(\mathbf{y})$ using some parametric or non-parametric model
- missing data imputation, i.e. generating pseudo-data $\mathbf{y}^* \sim p_\theta(\mathbf{y})$
- structure discovery, such as clustering and dimensionality reduction

In this thesis, we focus on *structure discovery* applications. Both, clustering and dimensionality reduction, play an important role in high-dimensional data analysis. For example, in genomics, the expression of more than 20,000 genes is measured, and getting insights into this data is challenging without clustering and/or dimensionality reduction techniques. In probabilistic machine learning, both these approaches rely on *latent variables*, which we cover in more detail in Section 1.3.

1.2 Bayesian Machine Learning

Bayesian interpretation of probability establishes a coherent framework to describe and quantify our *degrees of belief*, while relying on the rules of probability (see e.g. MacKay (2003)). As a result, within the Bayesian framework we can represent uncertainty about quantities that are unknown to us using probability distributions.

Let us denote all unknown quantities by $\boldsymbol{\theta}$ (these could include model parameters and/or latent variables that are specific to every data point) and observed data by $\mathbf{y}$. For example, in a regression setting, $\boldsymbol{\theta}$ could denote parameters of a linear model or a neural network, or in a non-parametric setting it could represent an infinite-dimensional function. In the Bayesian modelling workflow, we start by specifying the *prior* distribution over the latent variables $p(\boldsymbol{\theta})$ and the likelihood $p(\mathbf{y}|\boldsymbol{\theta})$. Note that as we have specified a joint probability model over $p(\mathbf{y}, \boldsymbol{\theta}) = p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})$, we have a *generative* model which lets us generate pseudo-observations from our model.

The prior distribution $p(\boldsymbol{\theta})$ represents our prior beliefs about these parameters *a priori*, before observing any data. Note that being a generative model, we can generate pseudo-data from the model even before observing any data (this could serve as a *prior predictive check*). Having observed some data $\mathbf{y}$, our prior beliefs

$p(\boldsymbol{\theta})$ are updated by conditioning on the observed data. The updated beliefs are represented by the *posterior* distribution $p(\boldsymbol{\theta}|\mathbf{y})$, which can be found via Bayes' rule

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})}{p(\mathbf{y})} = \frac{p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})}{\int p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})d\boldsymbol{\theta}}$$

Intuitively, conditioning on observed data typically reduces our uncertainty about $\boldsymbol{\theta}$. The normalising constant $p(\mathbf{y}) = \int p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})d\boldsymbol{\theta}$ is called the marginal likelihood, also known as model evidence. These two quantities, the posterior distribution $p(\boldsymbol{\theta}|\mathbf{y})$ and the marginal likelihood $p(\mathbf{y})$, are the key quantities of interest in Bayesian inference: the latter is useful e.g. for model comparison, whereas posterior distributions captures our knowledge about $\boldsymbol{\theta}$. The posterior is also needed to provide new predictions at test time. Predictions are made using the *posterior predictive* distribution

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{y}^*|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathbf{y})d\boldsymbol{\theta}$$

which involves averaging over $p(\boldsymbol{\theta}|\mathbf{y})$.

While the application of Bayes rule looks straightforward at first sight, it is quite the opposite in practice: the marginal likelihood is usually intractable, which makes it challenging to characterise the posterior. As a result, efficient posterior inference has been an active research topic in computational statistics and machine learning. In Section 1.4 we cover some of these techniques.

1.3 Latent Variable Models

Here, we first review the basics of latent variable models (LVMs), and then discuss how this class of models can let us identify low-dimensional structure in the data, i.e. perform dimensionality reduction.

1.3.1 Motivation for Latent Variable Modelling

A *hidden* or *latent* variable represents an *unobserved* quantity that we introduce as part of the model. In principle, these could directly represent missing data, but in

Bayesian models they are more frequently used as part of the generative model, to fill the role of underlying latent factors that have potentially given rise to the data.

We denote the observed (potentially high-dimensional) data vector by $\mathbf{y} \in \mathbb{R}^P$, and the associated latent variable by $\mathbf{z}$. The rationale behind introducing latent variables as part of the model is usually twofold:

- *Easier model specification:* Specifying the form of probability density $p(\mathbf{y})$ can be challenging, especially when the data $\mathbf{y}$ is high-dimensional. Instead, it is often desirable to introduce a latent variable $\mathbf{z}$ and specify a joint distribution $p(\mathbf{y}, \mathbf{z})$ that is tractable. This is usually formulated in the form

$$p(\mathbf{y}, \mathbf{z}) = p(\mathbf{y}|\mathbf{z})p(\mathbf{z}),$$

  where we need to specify a prior distribution over the latent variable $p(\mathbf{z})$ and characterise how the latent variables are mapped to the data via some likelihood $p(\mathbf{y}|\mathbf{z})$. Note that this way we have specified a generative mechanism for the data in terms of the latent variable $\mathbf{z}$.

- *Interpretation*: Often we would like the latent variables carry a meaningful interpretation. For example, for images of hand-written digits, a latent variable could capture a particular aspect of handwriting style, such as the angle or size. In cancer genomics, the latent variable could capture e.g. disease severity. In practice, the interpretation usually requires retrospective inspection of the fitted model, potentially relying on some held-out information.

The above two aspects are closely tied together: it is often a reasonable assumption that there exist a small number of underlying unobserved factors that characterise the variation in the data. That is, even though the data is very high-dimensional, it can be effectively represented in a much lower dimensional space without losing much information.

1.3.2 Discrete and continuous latent variables

We distinguish between *discrete* latent variables (with values $\mathbf{z} \in \{1, \ldots, K\}$, or equivalently represented in one-hot encoding) and *continuous* latent variables (with values $\mathbf{z} \in \mathbb{R}^D$). The former are typically used to cluster observations, whereas the latter are used to reveal continuous generative factors.

A common example of a discrete latent variable model would be to specify a prior $\mathbf{z} \sim \text{Categorical}(\pi_1, \ldots, \pi_K)$, where $\sum_k \pi_k = 1, \pi_k \geq 0$. This results in the marginal distribution $p(\mathbf{y})$ being a finite mixture of the following form

$$p(\mathbf{y}) = \sum_{k=1}^{K} \pi_k p(\mathbf{y}|\mathbf{z} = k).$$

Thus discrete latent variables are naturally suited to capture the presence of a finite number of sub-populations. This is useful for structure discovery applications, for example, in cancer genomics we could use discrete latent variables to represent the unknown tumour subtype of every patient. Common examples of discrete LVMs include the Gaussian mixture model (GMM), Dirichlet Process mixture model (Neal, 2000), Latent Dirichlet Allocation (Blei et al., 2003) etc.

A continuous latent variable can be seen as a generalisation of the latter, allowing us to capture changes over a continuous spectrum of values, resulting in the following marginal density

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{z})p(\mathbf{z})d\mathbf{z}.$$

A common choice would assume here a standard Gaussian prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. This framework is naturally suited for dimensionality reduction if we aim to learn continuous-valued factors of variation. For example, in cancer genomics a one-dimensional latent variable could be used to capture the underlying disease progression spectrum, representing the position of every patient along this axis. Examples of continuous LVMs include the probabilistic PCA (Tipping and Bishop, 1999b), Gaussian Process Latent Variable Models (Lawrence, 2005), Variational Autoencoders (Kingma and Welling, 2014) etc. In principle, discrete and continuous latent variables can be combined, resulting in hybrid models. For example, this can

be useful when we want to simultaneously perform dimensionality reduction and clustering. Such examples include Mixtures of Probabilistic Principal Component Analyzers (Tipping and Bishop, 1999a) and the Infinite Warped Mixture Model (Iwata et al., 2013).

In this thesis, we mostly focus on the use of continuous latent variables, but in Chapter 5 we will propose a hybrid model where we additionally make use of discrete latent variables. However, unlike most examples where they are introduced to cluster *observations*, we will employ them for the purpose of clustering *features*.

1.3.3 Latent Variable Models for dimensionality reduction

In this section, we focus on the setting where we would like to perform dimensionality reduction on an observed data matrix $\mathbf{Y} \in \mathbb{R}^{N \times P}$ with entries $y_i^{(j)}$ where $i \in \{1, \dots, N\}$ indexes observations and $j \in \{1, \dots, P\}$ features. We denote the vector of measurements for ith data point by $\mathbf{y}_i \in \mathbb{R}^P$ and the vector of values for jth feature by $\mathbf{y}^{(j)} \in \mathbb{R}^N$. As previously, we associate every observation with a latent variable $\mathbf{z}_i \in \mathbb{R}^D$, denoted collectively by $\mathbf{Z} \in \mathbb{R}^{N \times D}$.

So far, we have used a generic form of likelihood $p(\mathbf{y}_i|\mathbf{z}_i)$, i.e. we have not made any assumptions about how the latent variables $\mathbf{z}_i$ are mapped to the feature vector $\mathbf{y}_i \in \mathbb{R}^P$, as this depends on which class of models we consider. We now outline a general framework for dimensionality reduction with latent variable models, which depending on the choice of $p(\mathbf{y}_i|\mathbf{z}_i)$ can recover various methods.

As we primarily focus on tabular data where every feature has a distinct physical meaning, we index features individually, e.g. by writing $y_i^{(j)}$, whereas in the generative modelling literature (where image data is a common use case) it is common to simply treat all features collectively, e.g. by writing $\mathbf{y}_i$.

We now formulate a general framework for probabilistic dimensionality reduction. We assume a set of (parametric or non-parametric) mappings $f^{(j)} : \mathbf{z} \mapsto \mathbf{y}^{(j)}$ for every feature j, collectively denoted by $f^{(1:P)}$, and a likelihood that factorises over

features

$$p(\mathbf{Y} \mid \mathbf{Z}, f^{1:P}) = \prod_{j=1}^{P} \prod_{i=1}^{N} p(y_i^{(j)} \mid f^{(j)}(\mathbf{z}_i)).$$

For continuous-valued data, the likelihood could be chosen to be Gaussian, with mean $\mu = f^{(j)}(\mathbf{z})$ and either a shared or feature-specific variance σ_j^2, i.e.

$$p(\mathbf{Y} \mid \mathbf{Z}, f^{1:P}) = \prod_{j=1}^{P} \prod_{i=1}^{N} \mathcal{N}(y_i^{(j)} \mid f^{(j)}(\mathbf{z}_i), \sigma_j^2). \tag{1.1}$$

These functions f can either be specified by assuming a specific parametric form or non-parametrically, and can be treated in a frequentist or Bayesian framework. For example, parametric choices for f can include linear models, polynomial regression, and neural networks. Denoting these parameters by θ and the corresponding mapping $f_\theta^{(1:P)}$, the likelihood (1.1) would take the form

$$p(\mathbf{Y}|\mathbf{Z}, \theta) = \prod_{j=1}^{P} \prod_{i=1}^{N} \mathcal{N}(y_i^{(j)} \mid f_\theta^{(j)}(\mathbf{z}_i), \sigma_j^2).$$

These parameters θ can either be treated as hyper-parameters that are optimised (i.e. maximum likelihood estimation for θ), or alternatively in a fully Bayesian model where we would introduce a prior $p(\theta)$.

Both types of functional representations, parametric and non-parametric, can either be treated in a frequentist or Bayesian framework. The frequentist approach would find a single most likely function (i.e. the maximum likelihood estimate), whereas the Bayesian approach would infer a distribution over functions. Note that specifying a meaningful prior distribution over non-linear functions may be more challenging when using a parametric representations.

Depending on the choice of mappings f and whether they are treated in a Bayesian or non-Bayesian fashion, this framework will cover a range of approaches:

- **Linear mappings:** When mappings f are chosen to be linear, i.e. when we choose

  $$f^{(1:P)}(\mathbf{z}_i) := \boldsymbol{\mu} + \mathbf{W}\mathbf{z}_i$$

  with $\boldsymbol{\mu} \in \mathbb{R}^P, \mathbf{W} \in \mathbb{R}^{P \times D}$, we get the probabilistic PCA (PPCA) if we perform maximum likelihood estimation over $\{\boldsymbol{\mu}, \mathbf{W}\}$ (Tipping and Bishop, 1999b).

When additionally placing a prior over the intercept $\boldsymbol{\mu}$ and weights $\mathbf{W}$, i.e. when adopting a fully Bayesian treatment, we get the Bayesian PCA (Bishop, 1999).

- **Neural network mappings:** When mappings f are chosen to be parameterised via a neural network, we get the Variational Autoencoder (assuming a particular amortised inference model) (Kingma and Welling, 2014). We will cover this in more detail in Section 1.6.
- **Gaussian Process mappings:** When adopting a Bayesian non-parametric treatment for f, i.e. placing a Gaussian Process prior over $f \sim \mathcal{GP}(\cdot)$, we get the Bayesian Gaussian Process Latent Variable Model (Bayesian GPLVM) (Lawrence, 2005; Titsias and Lawrence, 2010). We will cover this in more detail in Section 1.5.3.

For the probabilistic PCA, the marginal likelihood with respect to $\mathbf{z}$ is tractable, whereas for the other two model classes a different approach for inference will be needed. Next, we give an overview of inference strategies in general, and in later sections discuss in detail how these apply to the aforementioned model classes (GPLVM and VAE).

1.4 Bayesian Inference

In Section 1.2 we introduced the Bayesian paradigm. We discussed how the Bayesian modelling framework lets us represent uncertainty about model parameters and/or latent variables $\boldsymbol{\theta}$ by treating them as random variables, and we showed how the prior $p(\boldsymbol{\theta})$ can be updated to incorporate observations $\mathbf{y}$ into a posterior $p(\boldsymbol{\theta}|\mathbf{y})$ using Bayes' rule.

In the Bayesian modelling framework, they key quantities of interest are *posterior* distributions. However, these are typically not available in closed form and they are challenging to characterise.

An exception to this is the case where the prior is *conjugate* to the likelihood, i.e. when the prior and the posterior belong to the same family of distributions. For

example, for Bernoulli likelihood the conjugate prior is the Beta distribution. In fact, all probability distributions that belong to an exponential family have natural conjugate prior distributions (Gelman et al., 2013). For example, this applies to the probabilistic PCA model considered in the previous section. However, closed form posteriors are generally not available any more in non-linear models.

One strategy to circumvent this would be to do point-estimation, i.e. to pick $\boldsymbol{\theta}$ value that maximises the posterior density, i.e.

$$\begin{aligned}\boldsymbol{\theta}^{\text{MAP}} :&= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathbf{y}) \\ &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \{\log p(\boldsymbol{\theta}) + \log p(\mathbf{y}|\boldsymbol{\theta})\}\end{aligned}$$

This $\boldsymbol{\theta}^{\text{MAP}}$ is known as the *maximum a posteriori* (MAP) estimate. While this approach does not require calculating the normalising constant, it does not capture any uncertainty about $\boldsymbol{\theta}$ and thus it loses various benefits of Bayesian inference. Note that MAP estimation can be seen as a penalised likelihood method, where the log-likelihood $\log p(\mathbf{y}|\boldsymbol{\theta})$ is maximised subject to a regularisation constraint that comes from the prior. Now, we turn to inference techniques that aim to capture posterior uncertainty.

Here we discuss two types of approaches for Bayesian inference for intractable posteriors: Monte Carlo methods and Variational Inference. The former is a sampling-based approach, whereas the latter class of techniques reformulates the inference task as an optimisation problem.

1.4.1 Monte Carlo methods

When obtaining a closed-form posterior is challenging, Monte Carlo methods instead aim to characterise this distribution by generating samples from it (Metropolis and Ulam, 1949; Gilks et al., 1995). These samples can then be used to estimate various characteristics of the posterior, such as the posterior mean and quantiles of the posterior distribution, but also e.g. the posterior predictive distribution.

Suppose we have an algorithm that provides samples from $p(\boldsymbol{\theta}|\mathbf{y})$, that is suppose we have $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T)} \sim p(\boldsymbol{\theta}|\mathbf{y})$, then for any quantity of interest $g(\boldsymbol{\theta})$ we can approximate $\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathbf{y})}[g(\boldsymbol{\theta})]$ with a Monte Carlo estimate as follows

$$\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathbf{y})}[g(\boldsymbol{\theta})] \approx \frac{1}{T}\sum_{t=1}^{T} g(\boldsymbol{\theta}^{(t)})$$

An entire sub-field of computational statistics is concerned with designing more efficient sampling methods for Bayesian inference. One common class of methods are Markov Chain Monte Carlo (MCMC) methods that aim to provide us with a sample from the posterior distribution. This is achieved by constructing a Markov chain whose stationary distribution is our posterior. Thus, (under certain assumptions) MCMC methods come with guarantees of producing asymptotically exact samples from the posterior density. Common examples of MCMC techniques include the Metropolis-Hastings algorithm (Hastings, 1970) and the gradient-based Hamiltonian Monte Carlo (HMC) (Neal, 2011; Betancourt, 2017) which has shown to be empirically successful for a wide range of problems. HMC also underlies as the key building block of a popular probabilistic programming framework Stan (Carpenter et al., 2017).

Despite the appealing theoretical guarantees, MCMC methods also have limitations, and inference for complex models with high-dimensional parameter spaces remains challenging. For example, efficient inference may require designing adaptive proposals, and it often involves running samplers for a large number of iterations (which cannot be trivially parallelised) followed by the need to perform chain convergence diagnostics. Furthermore, Monte Carlo samplers may get stuck in local modes. For example, in discrete state spaces which are inherently non-smooth and gradient-based methods (such as HMC) do not naturally apply, we have previously had to design ensemble MCMC techniques to overcome the problem of rapid convergence to local modes (Märtens et al., 2019a).

Next, we turn to a class of approximate inference methods.

1.4.2 Variational Inference

As opposed to sampling-based methods, the idea behind the class of methods known as *variational inference* (VI) is to rely on *optimisation* (Jordan et al., 1999; Blei et al., 2017). Specifically, instead of aiming to infer the *exact* the posterior distributions, we restrict ourselves to a family of densities $\mathcal{F}$. Then variational inference seeks to minimise the Kullback-Leibler (KL) divergence from the approximation $q(\boldsymbol{\theta}) \in \mathcal{F}$ to the true posterior $p(\boldsymbol{\theta}|\mathbf{y})$, i.e.

$$q^*(\boldsymbol{\theta}) := \arg\min_{q(\boldsymbol{\theta})\in\mathcal{F}} \mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathbf{y})) \tag{1.2}$$

where KL divergence is defined as

$$\mathrm{KL}(q||p) = \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} d\boldsymbol{\theta}$$

if $\mathbf{z}$ is a continuous random variable, and

$$\mathrm{KL}(q||p) = \sum_{\boldsymbol{\theta}_k\in\Theta} q(\boldsymbol{\theta}_k) \log \frac{q(\boldsymbol{\theta}_k)}{p(\boldsymbol{\theta}_k)}$$

if $\boldsymbol{\theta}$ is discrete and takes possible values in some set $\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$.

The resulting $q^*(\boldsymbol{\theta})$ is chosen to be our approximation of the posterior $p(\boldsymbol{\theta}|\mathbf{y})$.

Note that unless the family $\mathcal{F}$ includes the true posterior (which is typically not the case), VI is inherently an *approximate* method. This is in contrast to MCMC which can be seen as *exact in the limit* (i.e. as the number of MCMC iterations goes to infinity we get samples from the true posterior).

One possible (and in practice very commonly used) choice would be as follows

$$\mathcal{F} := \left\{ \mathcal{N}(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \;:\; \boldsymbol{\mu} \in \mathbb{R}^D,\; \boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_D),\; \sigma_1, \ldots, \sigma_D > 0 \right\}$$

which is known as the mean-field Gaussian approximation.

Note that KL divergence in (1.2) is not tractable. This can be seen when expanding according to the definition of KL divergence and applying Bayes rule

$$\begin{aligned} \mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathbf{y})) &= \mathbb{E}_{q(\boldsymbol{\theta})} \log q(\boldsymbol{\theta}) - \mathbb{E}_{q(\boldsymbol{\theta})} \log p(\boldsymbol{\theta}|\mathbf{y}) \\ &= \mathbb{E}_{q(\boldsymbol{\theta})} \log q(\boldsymbol{\theta}) - \mathbb{E}_{q(\boldsymbol{\theta})} \log p(\boldsymbol{\theta}, \mathbf{y}) + \log p(\mathbf{y}). \end{aligned}$$

This expression still involves calculation of the log marginal likelihood $\log p(\mathbf{y})$. However, note that the latter does not involve $q(\boldsymbol{\theta})$, so for the purpose of optimisation it can be treated as a constant and thus ignored. Subtracting $\log p(\mathbf{y})$ and negating the sign, we arrive at the following objective

$$\text{ELBO}(q) := \mathbb{E}_{q(\boldsymbol{\theta})} \log p(\boldsymbol{\theta}, \mathbf{y}) - \mathbb{E}_{q(\boldsymbol{\theta})} \log q(\boldsymbol{\theta}) \tag{1.3}$$

which is known as the evidence lower bound (ELBO). Note that

- Maximising the ELBO is equivalent to minimising $\text{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathbf{y}))$, i.e.

$$\arg\min_{q \in \mathcal{F}} \text{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathbf{y})) = \arg\max_{q \in \mathcal{F}} \text{ELBO}(q).$$

- ELBO lower bounds the log marginal likelihood (also known as log evidence), i.e. $\log p(\mathbf{y}) \geq \text{ELBO}(q)$, thus justifying its name. This holds because KL divergence is always non-negative, therefore indeed

$$\log p(\mathbf{y}) = \text{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta}|\mathbf{y})) + \text{ELBO}(q) \geq \text{ELBO}(q).$$

For example, common examples include the previously discussed Latent Variable Models (Section 1.3.3) where every observation $i = 1, \ldots, N$ is associated with a latent variable $\mathbf{z}_i$. In this case, we would be interested in performing variational inference over all latent variables $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ and potentially some other global parameters. Below, we particularly focus on variational inference for latent variable models where $\boldsymbol{\theta} := \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$.

1.4.2.1 Classical Variational Inference

Before the era of automatic differentiation software, variational inference was derived for specific class of models where conditional distributions had a convenient form, assuming a particular variational family (Ghahramani and Beal, 2001).

One commonly used algorithm was *coordinate ascent variational inference* (CAVI), which iteratively updates each coordinate of the variational distribution (Blei et al., 2017). This relies on coordinate-wise updates which are available in closed form

e.g. when all complete conditionals belong to the exponential family. These include models such as Gaussian mixture models (and mixtures of other exponential family distributions), Bayesian linear regression, Latent Dirichlet Allocation etc. However, it would be desirable to have a generic approach which would apply beyond a restricted class of models, without having to derive update rules.

1.4.2.2 Modern adaptations of Variational Inference

In the era of automatic differentation, deriving analytic solutions to optimisation problems or writing down gradients in closed form are not required any more. Not only has this made it possible to perform maximum likelihood estimation using stochastic gradient descent (SGD) without having to analytically derive gradients, this has been equally important for stochastic variational inference (SVI) (Hoffman et al., 2013).

Here we outline *desiderata* for modern adaptations of VI:

- Being generally applicable, without requiring model-specific derivations.
- Scalability to large data sets via subsampling.
- Sharing of variational parameters between data points, so that the number of variational parameters would not necessarily grow with the number of observations N.

The first property is closely related to Black Box Variational Inference (BBVI) (Ranganath et al., 2014), which outlines a generic framework for VI. For various strategies for Monte Carlo gradient estimation of (potentially black box) probabilistic objectives we refer to (Mohamed et al., 2019). In practice, it is very common to resort to using pathwise gradient estimators, i.e. the so-called reparameterisation trick (Rezende et al., 2014; Kingma and Welling, 2014). The second and third property can be achieved within the *amortised* variational inference framework.

1.4.2.3 Amortised VI

Imagine a latent variable model, where every observation $\mathbf{y}_i \in \mathbb{R}^P$ is associated with a latent variable $\mathbf{z}_i \in \mathbb{R}^Q$ (where i indexes data points $i = 1, \ldots, N$). When the number of observations N grows, inference becomes computationally more challenging due to a large number of latent variables. For example, when using standard (non-amortised) variational inference for $\boldsymbol{\theta} = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$ with the following approximating distribution

$$q(\mathbf{z}_1, \ldots, \mathbf{z}_N) = \prod_{i=1}^{N} q(\mathbf{z}_i) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i | \mathbf{m}_i, \text{diag}(\mathbf{s}_i)),$$

then the number of variational parameters grows linearly with N.

The idea behind *amortised* variational inference is the strategy of *sharing* variational parameters across data points (Gershman and Goodman, 2014; Kingma and Welling, 2014; Rezende et al., 2014). This is achieved in practice by introducing some *global* variational parameters ϕ, e.g. ϕ could be parameters of a neural network, and parameterising

$$q_\phi(\mathbf{z}_1, \ldots, \mathbf{z}_N) = \prod_{i=1}^{N} q_\phi(\mathbf{z}_i | \mathbf{y}_i),$$

where $q_\phi(\mathbf{z}_i|\mathbf{y}_i)$ is a parametric mapping that characterises the approximate posterior. For example, ϕ can parameterise the mappings $\mu_\phi : \mathbf{y}_i \mapsto \mathbf{m}_i$ and $\sigma^2_\phi : \mathbf{y}_i \mapsto \mathbf{s}_i$, where $\mathcal{N}(\mathbf{z}_i|\mathbf{m}_i, \text{diag}(\mathbf{s}_i))$ are the approximate posteriors. That is, $q_\phi(\mathbf{z}_i|\mathbf{y}_i) = \mathcal{N}(\mathbf{z}_i|\mu_\phi(\mathbf{y}_i), \text{diag}(\sigma^2_\phi(\mathbf{y}_i)))$. Now, in comparison to non-amortised VI where ELBO is optimised with respect to local parameters $\{\mathbf{m}_i, \mathbf{s}_i\}_{i=1}^N$, here instead the optimisation is carried out with respect to global parameters ϕ. When this mapping is parameterised by a neural network, it is referred to as the *inference network*. This is a key component to Variational Autoencoders (Kingma and Welling, 2014) that will be introduced in Section 1.6.

Amortisation lets us avoid data point specific variational parameters, which combined with stochastic gradient descent, makes it possible to scale up inference to large data sets. Note that additionally, we can use it at test time for a new observation $\mathbf{y}^*$ to approximate the posterior $p(\mathbf{z}^*|\mathbf{y}^*, \mathbf{y}_1, \ldots, \mathbf{y}_N) \approx q_\phi(\mathbf{z}^*|\mathbf{y}^*)$ without having to carry out any additional optimisation.

1.5 Gaussian Processes

Learning *functions* that map an input $\mathbf{x} \in \mathcal{X}$ to some output(s) $\mathbf{y} \in \mathcal{Y}$ constitutes an important part of statistics and machine learning. This includes regression and classification tasks, which on an abstract level, can be thought of as *curve fitting*. But learning (potentially non-linear) mappings is important not only for *predictive* models, but is often also used as part of *unsupervised* learning methods as well. For example, in latent variable models for dimensionality reduction, such mappings are used to map latent variables to the observed data (or vice versa).

In this thesis, we primarily focus on two types of non-linear mappings: neural networks and Gaussian Processes. These two model classes have different advantages and disadvantages: for example, Gaussian Processes are a Bayesian non-parametric approach to modelling functions (as opposed to neural networks, they do this directly in the function space). Next, we review the basics of Gaussian Processes and how they can be used for regression and dimensionality reduction.

1.5.1 Distributions over functions with GPs

Gaussian Processes (GPs) (Williams and Rasmussen, 2006) are a Bayesian non-parametric approach for inference directly over functions.

Let us consider functions f defined on inputs $\mathbf{x} \in \mathcal{X}$ (in the applications we consider, $\mathcal{X} = \mathbb{R}^D$). In principle, the construction of a probability distribution over functions f involves an infinite-dimensional random variable $\{f(\mathbf{x})\}_{\mathbf{x}\in\mathcal{X}}$, which raises the question whether this could be computationally feasible. The Gaussian Process framework circumvents this problem in an elegant way. It relies on defining joint distributions over *finite* subsets of $\mathcal{X}$ that are *marginally consistent*: A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution.

We say f is a Gaussian process with mean function μ and covariance function $k(\cdot)$, and we denote this as follows

$$f \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

if for every finite set of inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$, the corresponding vector of function values $\mathbf{f} := (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))$ follows a multivariate Gaussian distribution

$$p(\mathbf{f} \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_N, \theta) = \mathcal{N}(\mathbf{f} \,|\, \boldsymbol{\mu}, \mathbf{K})$$

where $\boldsymbol{\mu} \in \mathbb{R}^N$ is the mean vector $\boldsymbol{\mu} := (\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_N))$ and $\mathbf{K} \in \mathbb{R}^{N \times N}$ is the kernel matrix of all pairs of inputs, with elements $\mathbf{K}_{ik} := k(\mathbf{x}_i, \mathbf{x}_k)$. Often the mean function μ is chosen to be zero, i.e. $\mu(\mathbf{x}) \equiv 0$.

By *consistency* we mean that if we consider function values at any two sets of inputs $\mathbf{X}_A := (\mathbf{x}_1, \ldots, \mathbf{x}_r)$ and $\mathbf{X}_B := (\mathbf{x}_{r+1}, \ldots, \mathbf{x}_N)$, and define their joint distribution as

$$\begin{pmatrix} f(\mathbf{X}_A) \\ f(\mathbf{X}_B) \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{pmatrix}, \quad \begin{pmatrix} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \mathbf{K}_{BA} & \mathbf{K}_{BB} \end{pmatrix} \right]$$

then upon marginalisation

$$p(f(\mathbf{X}_A)) = \int p(f(\mathbf{X}_A), f(\mathbf{X}_B)) df(\mathbf{X}_B) = \mathcal{N}(\boldsymbol{\mu}_A, \mathbf{K}_{AA})$$
$$p(f(\mathbf{X}_B)) = \int p(f(\mathbf{X}_A), f(\mathbf{X}_B)) df(\mathbf{X}_A) = \mathcal{N}(\boldsymbol{\mu}_B, \mathbf{K}_{BB})$$

we get distributions that are consistent with definitions for $f(\mathbf{X}_A) \sim \mathcal{N}(\boldsymbol{\mu}_A, \mathbf{K}_{AA})$ and $f(\mathbf{X}_B) \sim \mathcal{N}(\boldsymbol{\mu}_B, \mathbf{K}_{BB})$.

Typically the kernel has some hyperparameters which we denote by θ. For example, one popular choice for $k(\cdot)$ is the squared exponential ARD kernel,

$$k(\mathbf{x}_i, \mathbf{x}_k) = \sigma^2 \left(-\frac{1}{2} \sum_{j=1}^{D} \frac{(x_{ij} - x_{kj})^2}{l_j^2} \right)$$

with hyperparameters $\theta := (\sigma^2, l_1, \ldots, l_D)$, where σ^2 is the kernel variance parameter and l_j are feature-specific length-scales.

1.5.2 Gaussian Processes for Regression

Here, we consider Gaussian Process regression (GPR). Suppose we have observed input-output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^D, y_i \in \mathbb{R}$, denoted collectively by $\mathbf{X} := (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and $\mathbf{y} = (y_1, \ldots, y_N)$. In GP regression, we are interested in inference over mappings $f : \mathbb{R}^D \to \mathbb{R}$, where we have placed a GP prior over f.

A standard approach to GP regression is to assume a Gaussian likelihood, i.e. $y_i = f(\mathbf{x}_i) + \varepsilon_i$ with Gaussian measurement noise $\varepsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$.

$$f \sim \mathcal{GP}(0, k(\cdot))$$
$$p(\mathbf{y}|f, \mathbf{X}) = \prod_{i=1}^{N} \mathcal{N}(y_i|f(\mathbf{x}_i), \sigma_\epsilon^2)$$

Denoting $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))$, we can integrate out $\mathbf{f}$ and obtain the *marginal* likelihood in closed form as follows

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{X}, \mathbf{f}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} &= \int \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_\epsilon^2 \mathbf{I}) \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}) d\mathbf{f} \\ &= \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma_\epsilon^2 \mathbf{I}). \end{aligned}$$

While this marginal likelihood can be calculated analytically, its calculation requires performing a Cholesky decomposition of $\mathbf{K}$ which has size $N \times N$, thus this computation has $O(N^3)$ complexity.

Predictions at new points $\mathbf{x}^*$, denoted by $\mathbf{f}^* := f(\mathbf{x}^*)$, can be obtained from the joint distribution

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}^* \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{xx} + \sigma_\epsilon^2 \mathbf{I} & \mathbf{K}_{xx^*} \\ \mathbf{K}_{x^*x} & \mathbf{K}_{x^*x^*} \end{pmatrix} \right]$$

by conditioning on observed outcomes $\mathbf{y}$ as follows

$$\mathbf{f}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^* \sim \mathcal{N}(\mathbf{f}^* \,|\, \mathbf{K}_{x^*x}(\mathbf{K}_{xx} + \sigma_\epsilon^2)^{-1}\mathbf{y},\ \mathbf{K}_{x^*x^*} - \mathbf{K}_{x^*x}(\mathbf{K}_{xx} + \sigma_\epsilon^2)^{-1}\mathbf{K}_{xx^*}).$$

Note that this posterior distribution $p(\mathbf{f}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*)$ is also a GP. The posterior predictive $\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^*$ can be obtained analogously, its variance has an additional σ_ϵ^2 term:

$$\mathbf{y}^*|\mathbf{y}, \mathbf{X}, \mathbf{x}^* \sim \mathcal{N}(\mathbf{y}^* \,|\, \mathbf{K}_{x^*x}(\mathbf{K}_{xx} + \sigma_\epsilon^2)^{-1}\mathbf{y},\ \mathbf{K}_{x^*x^*} - \mathbf{K}_{x^*x}(\mathbf{K}_{xx} + \sigma_\epsilon^2)^{-1}\mathbf{K}_{xx^*} + \sigma_\epsilon^2 \mathbf{I}).$$

1.5.3 Gaussian Processes for dimensionality reduction

We now return to the dimensionality reduction task discussed in section 1.3.3. Specifically, recalling the generative model in equation (1.1), we now consider the case where we place the Gaussian Process prior over the mappings $f^{(j)} \sim \mathcal{GP}(\cdot)$.

The resulting model is called the (Bayesian) Gaussian Process Latent Variable Model (GPLVM) (Lawrence, 2004; Lawrence, 2005).

Suppose we have an observed data matrix $\mathbf{Y} \in \mathbb{R}^{N \times P}$, consisting of P features $(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(P)})$ for N data points, and our goal is to perform dimensionality reduction, i.e. to learn a low-dimensional representation $\mathbf{Z} := (\mathbf{z}_1, \dots, \mathbf{z}_N)$ of the (potentially high-dimensional) data. When conditioning on the latent variables $\mathbf{z}$, the GPLVM is essentially a multi-output GP regression model as it specifies a GP prior for every latent mapping $f^{(j)} \sim \mathcal{GP}(\cdot)$ which relates to the observations via some likelihood $p(\mathbf{y}^{(j)}|f^{(j)}, \mathbf{Z})$ for every feature $j \in \{1, \dots, P\}$. Placing a prior over the low-dimensional $\mathbf{Z}$ our aim is to infer the respective posterior. Denoting $\mathbf{f}^{(j)} := (f^{(j)}(\mathbf{z}_1), \dots, f^{(j)}(\mathbf{z}_N))$ and the collection of function values $\mathbf{F} := (\mathbf{f}^{(1)}, \dots, \mathbf{f}^{(P)}) \in \mathbb{R}^{N \times P}$, the GPLVM is formulated as the following generative model

$$
\begin{aligned}
p(\mathbf{Z}) &= \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i|\mathbf{0}, \mathbf{I}) \\
p(\mathbf{F}|\mathbf{Z}, \theta) &= \prod_{j=1}^{P} \mathcal{N}(\mathbf{f}^{(j)}|\mathbf{0}, \mathbf{K}_{zz}^{(j)}) \\
p(\mathbf{Y}|\mathbf{F}) &= \prod_{j=1}^{P} p(\mathbf{y}^{(j)}|\mathbf{f}^{(j)}) = \prod_{j=1}^{P} \prod_{i=1}^{N} p(y_i^{(j)}|f^{(j)}(\mathbf{z}_i))
\end{aligned}
$$

where the kernel matrices $\mathbf{K}_{zz}^{(j)}$ are all evaluated at the same inputs, but they may potentially have different kernel hyperparameter configurations.

In the special case when the emission likelihood is Gaussian, i.e. when

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

one can analytically integrate out the GP mappings, resulting in the marginal

$$
p(\mathbf{Y}|\mathbf{Z}, \theta) = \prod_{j=1}^{P} p(\mathbf{y}^{(j)}|\mathbf{Z}, \theta) = \prod_{j=1}^{P} \mathcal{N}(\mathbf{y}^{(j)} \,|\, \mathbf{0}, \mathbf{K}_{zz}^{(j)} + \sigma^2\mathbf{I}). \tag{1.4}
$$

In this special case, inference needs to be carried out on latent variables $\mathbf{z}_1, \dots, \mathbf{z}_N$ and kernel hyperparameters θ only, whereas generally for non-Gaussian emissions we additionally need to infer $\mathbf{F}$ which can no longer be analytically marginalised.

Note that the likelihood in equation (1.4) involves latent variables $\mathbf{Z}$ that cannot be analytically marginalised. This can be addressed in various ways. For example, Lawrence (2005) proposed to perform MAP inference over $\mathbf{Z}$, i.e.

$$\mathbf{Z}^{\text{MAP}} := \arg\max_{\mathbf{Z}} p(\mathbf{Y}|\mathbf{Z}, \theta)p(\mathbf{Z})$$

however this ignores uncertainty about $\mathbf{Z}$. Alternatively, Titsias and Lawrence (2010); Damianou et al. (2016) propose a variational inference approach, where latent variables are approximately integrated out. That is, they introduce a variational approximation $q(\mathbf{Z}) := \prod_{i=1}^{N} q(\mathbf{z}_i) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}_i|\mathbf{m}_i, \mathbf{S}_i)$ and obtain the following lower bound

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\mathbf{Z})} \log p(\mathbf{Y}|\mathbf{Z}) - \sum_{i=1}^{N} \text{KL}(q(\mathbf{z}_i)||p(\mathbf{z}_i)) \\ &= \sum_{j=1}^{P} \mathbb{E}_{q(\mathbf{Z})} \log \mathcal{N}(\mathbf{y}^{(j)} \,|\, \mathbf{0}, \mathbf{K}_{zz}^{(j)} + \sigma_j^2 \mathbf{I}) - \sum_{i=1}^{N} \text{KL}(q(\mathbf{z}_i)||p(\mathbf{z}_i)) \end{aligned} \tag{1.5}$$

For specific kernels, such as the squared exponential kernel, the above expectation could be obtained in closed form, as outlined by Damianou et al. (2016).

In our implementation of the GPLVM and its extensions that we develop in this thesis, we decided to evaluate ELBO (1.5) using Monte-Carlo estimates of the expectation, i.e. using reparameterisation-based sampling for $\mathbf{Z} \sim \prod_{i=1}^{N} q(\mathbf{z}_i)$ to estimate $\mathbb{E}_{q(\mathbf{Z})} \log p(\mathbf{Y}|\mathbf{Z})$, which makes it easier to create a more modular implementation. Note that evaluating ELBO (1.5) has complexity $O(N^3 P)$, i.e. it scales cubically with N and linearly with the data dimension P.

1.5.4 Scalable inference for GPs

While GPs offer an elegant non-parametric probabilistic framework for inference over functions, the inference strategies we have discussed so far suffer from scalability issues. Exact GP inference has $O(N^3)$ computational complexity and $O(N^2)$ memory demands, which limit its applicability to large data sets. As a result, there has been substantial research interest in scaling up GPs in various ways.

The main line of research can be categorised as *inducing point* methods (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013) which aim to build a low rank approximation to the kernel matrix around the inducing points. These approaches are also referred to as *sparse* GPs.

Here, we outline scalable inference for the sparse GPLVM (Titsias and Lawrence, 2010; Damianou et al., 2016). The key idea behind this approach is to augment the model with additional variables, treated within a variational inference framework. Specifically, we introduce $M \ll N$ inducing variables, that is inducing inputs $\mathbf{Z}_u := (\mathbf{z}_1^u, \dots, \mathbf{z}_M^u)$ that live in the same space as our latent inputs $\mathbf{z}_i$ and the respective GP function values $\mathbf{u}^{(j)} = (u_1^{(j)}, \dots, u_M^{(j)})$, where $u_i = f^{(j)}(\mathbf{z}_i^u)$ for each feature $j \in \{1, \dots, P\}$ and inducing point $i \in \{1, \dots, M\}$. The variables $\mathbf{u}$ are referred to as the *inducing variables*.

Having augmented the model with $\mathbf{Z}_u$ and $\mathbf{U}$, we specify the joint generative model. We assume that inducing inputs and outputs $(\mathbf{Z}_u, \mathbf{u}^{(j)})$ are related in the same way as $(\mathbf{Z}, \mathbf{f}^{(j)})$, i.e.

$$\begin{pmatrix} \mathbf{f}^{(j)} \\ \mathbf{u}^{(j)} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fu} \\ \mathbf{K}_{uf} & \mathbf{K}_{uu} \end{pmatrix} \right]$$

for all features $j = 1, \dots, P$ independently. This implies

$$\begin{aligned} p(\mathbf{u}^{(j)}|\mathbf{Z}_u) &= \mathcal{N}(\mathbf{u}^{(j)}|\mathbf{0}, \mathbf{K}_{uu}) \\ p(\mathbf{f}^{(j)}|\mathbf{u}^{(j)}, \mathbf{Z}_u, \mathbf{Z}) &= \mathcal{N}(\mathbf{f}^{(j)}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}^{(j)}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}). \end{aligned}$$

Thus, the joint model factorises as follows

$$p(\mathbf{Y}, \mathbf{F}, \mathbf{U}, \mathbf{Z}|\mathbf{Z}_u) = p(\mathbf{Z}) \prod_{j=1}^{P} p(\mathbf{y}^{(j)}|\mathbf{f}^{(j)}) p(\mathbf{f}^{(j)}|\mathbf{u}^{(j)}, \mathbf{Z}, \mathbf{Z}_u) p(\mathbf{u}^{(j)}|\mathbf{Z}_u).$$

We define the following structured variational approximation

$$q(\mathbf{F}, \mathbf{U}, \mathbf{Z}) = q(\mathbf{Z}) \prod_{j=1}^{P} q(\mathbf{f}^{(j)}, \mathbf{u}^{(j)}|\mathbf{Z}) = \left[\prod_{i=1}^{N} q(\mathbf{z}_i) \right] \prod_{j=1}^{P} p(\mathbf{f}^{(j)}|\mathbf{u}^{(j)}, \mathbf{Z}_u, \mathbf{Z}) q(\mathbf{u}^{(j)}),$$

where $p(\mathbf{f}^{(j)}|\mathbf{u}^{(j)}, \mathbf{Z}_u, \mathbf{Z})$ is the prior predictive distribution. Note that in the variational formulation, inducing inputs $\mathbf{Z}_u$ become variational parameters. Regarding

the choice of $q(\mathbf{u}^{(j)})$, one common choice is to use a multivariate Gaussian as the variational family, i.e. $q(\mathbf{u}^{(j)}) = \mathcal{N}(\mathbf{m}_j, \mathbf{S}_j)$. This is a generic approach that works with any likelihood. In the special case when the likelihood is Gaussian, there exists another strategy. In this case the variational parameters $\mathbf{u}^{(j)}$ can be *optimally collapsed*, thus allowing more efficient inference.

The latter approach, where $\mathbf{u}^{(j)}$ has been collapsed, results in the following ELBO

$$\begin{aligned}\mathcal{L} = \sum_{j=1}^{P} \mathbb{E}_{q(\mathbf{Z})} \log \mathcal{N}(\mathbf{y}^{(j)} \,|\, \mathbf{0}, \mathbf{K}_{fu}^{(j)} \mathbf{K}_{uu}^{(j)^{-1}} \mathbf{K}_{uf}^{(j)} + \sigma_j^2 \mathbf{I}) \\ -\frac{1}{2\sigma_j^2} \mathbb{E}_{q(\mathbf{Z})} \mathrm{tr}\left(\mathbf{K}_{ff}^{(j)} - \mathbf{K}_{fu}^{(j)} \mathbf{K}_{uu}^{(j)^{-1}} \mathbf{K}_{uf}^{(j)}\right) \\ - \sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{z}_i) || p(\mathbf{z}_i)).\end{aligned} \tag{1.6}$$

Analogously to the lower bound in equation (1.5), this expression can either be obtained analytically for certain kernels, using so-called kernel expectations as in (Titsias and Lawrence, 2010; Damianou et al., 2016), or alternatively evaluated using the reparameterisation trick.

1.6 Variational Autoencoders

Now we return to the dimensionality reduction framework considered in section 1.3.3, and consider the special case of the latent variable model where the mappings are parameterised by neural networks. That is, recalling equation (1.1), we now choose to parameterise the mappings $f^{(1:P)} : \mathbb{R}^D \to \mathbb{R}^P$ jointly using a neural network. Combining this generative model with a particular amortised variational inference scheme, results in latent variable modelling framework known as the Variational Autoencoder (Kingma and Welling, 2014).

Let us denote our observed data matrix by $\mathbf{Y} \in \mathbb{R}^{N \times P}$, where $\mathbf{y}_i \in \mathbb{R}^P$ represents the feature vector for data point i. Then Variational Autoencoders (VAEs) assume the following generative model

$$\begin{aligned}\mathbf{z}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{y}_i | \mathbf{z}_i, \theta &\sim p(\mathbf{y}_i | f_\theta^{(1:P)}(\mathbf{z}_i))\end{aligned} \tag{1.7}$$

where $f_\theta^{(1:P)}$ is a neural network (NN) with parameters θ such that $f_\theta^{(1:P)} : \mathbf{z} \mapsto \mathbf{y}$ with input dimension $\dim(\mathbf{z}) = D$ and output dimension $\dim(\mathbf{y}) = P$. For example, under a Gaussian likelihood we would have

$$\begin{aligned} p(\mathbf{Z}) &= \prod_{i=1}^{N} N(\mathbf{z}_i|\mathbf{0}, \mathbf{I}) \\ p(\mathbf{Y}|\mathbf{Z}, \theta) &= \prod_{i=1}^{N} \prod_{j=1}^{P} \mathcal{N}(\mathbf{y}_i^{(j)}|f_\theta^{(j)}(\mathbf{z}_i), \sigma_j^2) \end{aligned} \tag{1.8}$$

where, notation $f^{(j)}$ refers to picking out the j-th output coordinate from the multivariate $f_\theta^{(1:P)}$.

The VAE generative model in equation (1.8) is similar to the one assumed by the GPLVM, but the two differ in how we specify the mappings $f^{(j)}$. In the GPLVM framework, we specify a GP prior for every mapping $f^{(j)} \sim \mathcal{GP}(\cdot)$, thus this is a fully Bayesian approach where the prior is specified directly in the function space. In contrast, in the VAE framework we represent functions in the weight space of some neural network, and we typically do not treat the parameters θ probabilistically (though in principle we can take a fully Bayesian approach by specifying a prior $p(\theta)$ in the weight space).

There is another subtle difference between the two models. In the GPLVM the prior over functions factorises over data dimensions $j = 1, \ldots, P$, so the mappings across the features are modelled independently and there where is no sharing mechanism. This differs from the VAE framework where we model all data dimensions jointly in a single neural network, thus potentially allowing for dependency between two output coordinates of $f_\theta^{(1:P)}$. As a result, neural architectures typically allow for *shared* representations.

The marginal likelihood of the VAE is given by

$$p(\mathbf{Y}) = \prod_{i=1}^{N} \int p_\theta(\mathbf{y}_i|\mathbf{z}_i) p(\mathbf{z}_i) d\mathbf{z}_i,$$

which is intractable and thus requires the use of approximate inference techniques. Specifically, the VAE employs amortised VI via the introduction of the inference network q_ϕ.

Application of standard variational inference methodology (as outlined in Section 1.4.2) leads to a lower bound on the log marginal likelihood, i.e. the ELBO of the form:

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i)}[\log p_\theta(\mathbf{y}_i|\mathbf{z}_i)] - \sum_{i=1}^{N} \mathrm{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i)||p(\mathbf{z}_i)) \tag{1.9}$$
$$\text{where } \; q_\phi(\mathbf{z}_i|\mathbf{y}_i) = \mathcal{N}(\mathbf{z}_i|\mu_\phi(\mathbf{y}_i), \mathrm{diag}(\sigma^2_\phi(\mathbf{y}_i)))$$

Training a VAE seeks to optimise both the model parameters θ and the variational parameters ϕ jointly using stochastic gradient ascent in combination with the reparameterisation trick to get Monte Carlo estimates of the expectation in the first term in equation (1.9).

In the VAE, the variational approximation $q_\phi(\mathbf{z}_i|\mathbf{y}_i)$ is referred to as the *encoder* since it encodes the data $\mathbf{y}$ into the latent variable $\mathbf{z}$, whilst the mapping as part of the generative model $p_\theta(\mathbf{y}|\mathbf{z})$ is referred to as the *decoder* since it decodes the latent variables into observations.

Conditional VAEs

The Conditional VAE (CVAE) (Kingma et al., 2014; Sohn et al., 2015) augments the VAE by conditioning the generative model on additional inputs $\mathbf{c}$, i.e. the VAE generative model $\mathbf{y}_i = f^\theta(\mathbf{z}_i) + \epsilon_i$ is now replaced by $\mathbf{y}_i = f^\theta(\mathbf{z}_i, \mathbf{c}_i) + \varepsilon_i$. Conditioning on the extra inputs can greatly increase the expressiveness of the VAE when applied to multimodal data where the indicator driving the multimodality is observable. For example, if the additional inputs correspond to class labels then we can obtain distinct output distributions for each class. For the genomics applications we consider, we use this mechanism to incorporate *known covariates* as part of the model.

Inference for the CVAE requires a minor alteration to the standard VAE objective, i.e. we modify equation (1.9) as follows

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i,\mathbf{c}_i)}[\log p_\theta(\mathbf{y}_i|\mathbf{z}_i, \mathbf{c}_i)] - \sum_{i=1}^{N} \mathrm{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i, \mathbf{c}_i)||p(\mathbf{z}_i)),$$

where both the encoder and the decoder now depend on additional inputs $\mathbf{c}_i$.

1.6.1 Example application: The Cancer Patient Experience Survey

To demonstrate the utility of latent variable modelling with VAEs, we will use publicly available data from the Cancer Patient Experience Survey, conducted on behalf of NHS England. The goal of this survey is to monitor cancer care. In the years 2015–2018 that we focus on, more than 250,000 individuals have filled in the questionnaire. There are 58 questions about various aspects of cancer treatment services, e.g. diagnostic tests, support for people diagnosed with cancer, deciding what is the best treatment, hospital care as an inpatient or outpatient etc. Thus the data contains 58 categorical variables, each variable having its own number of levels, which makes it challenging to analyse or summarise. There is an additional 59th question which asks the patient to rate their care overall from 0 to 10. For now, we will not include the overall score in our analysis, but we will retrospectively compare our dimensionality reduction to this score.

Here, we use a VAE with a Categorical likelihood (where the number of categories is specific for every feature) to learn a one-dimensional latent space $\mathbf{z} \in \mathbb{R}$. Our hope is that we will identify a meaningful latent representation, which acts as a continuous-valued summary statistic of an individual's answers to the questionnaire.

Figure 1.2 illustrates what we have learnt. Panel (A) demonstrates that our latent variable $\mathbf{z}$ correlates with the held-out overall happiness score. Thus we were able to extract such a summary statistic using a VAE in an unsupervised manner. Arguably, being based on all 58 questions, it is a better representation than a single score from 0 to 10. For example, there are a number of individuals who rated their care poorly by score 1, but their inferred $\mathbf{z}_i$ values are large which should reflect higher satisfaction. Upon manual inspection, we have confirmed that their answers indeed reflect much higher satisfaction than their assigned score[1]. Therefore the VAE analysis provides a means to potentially verify consistency.

[1]We note that there is also a possibility that these were data entry errors.

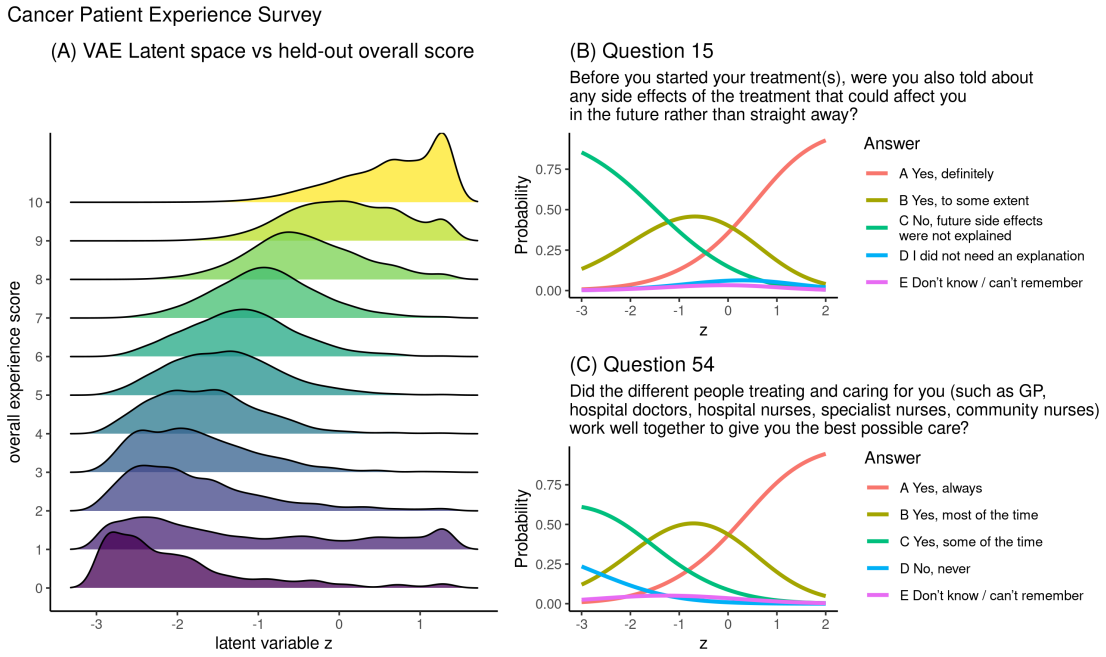


Figure 1.2: Application of latent variable modelling with the Variational Autoencoder on the National Cancer Patient Experience Survey (2015 – 2018). Using questionnaire data from more than 250,000 individuals, we fitted a VAE model with a one-dimensional latent $\mathbf{z}$. (A) The inferred latent coordinates (on x-axis) provide a continuous-valued summary statistic that is correlated with the held-out overall experience score (from 0 to 10 on y-axis). (B, C) Examples of two questions (Q15 and Q54) from the survey, and the learnt decoder mappings which indicate how likely an individual with a particular $\mathbf{z}$ value is to choose a specific answer.

Furthermore, we can inspect the decoder mappings, i.e. for every question we can visualise the probability of a certain answer across $\mathbf{z}$ values. We have visualised these for two selected questions, Q15 and Q54, in Figure 1.2(B, C). Their interpretation aligns with the hypothesis that larger $\mathbf{z}$ values correspond to being happier with the cancer care.

Thus, here we have demonstrated how the VAE, a particular non-linear latent variable model, has allowed us to analyse a large tabular data set and reveal latent structure that can retrospectively be inspected. Retrospective interpretation of the latent space can be aided by held-out information (like the overall experience score) or by inspecting how the latent space is mapped to the feature space, as we saw in Figure 1.2(B, C). In this particular case, the dimensionality of the data is moderate which makes the latter approach feasible, however, for high-dimensional data we

will need different strategies.

1.7 Interpretability

Black box machine learning systems have been shown to achieve highly accurate predictions in a variety of tasks. However, when we would like to deploy these systems in real-life applications where the decisions have a significant impact, prediction accuracy is not the only criterion that matters. Safety, robustness, fairness, and explainability become desirable in these scenarios. For example, regardless of whether we consider credit scoring, disease diagnosis, or self-driving cars, we do not want any sub-population to be discriminated against. Also, we might require an explanation which inputs were relevant for a particular prediction. In fact, a *right to an explanation* is implied by the European General Data Protection Regulation[2]. Note that all such aforementioned criteria are easier to ensure in non-black-box models, whereas in a black-box model we will need some form of built-in or post-hoc interpretability. This has led to substantial interest in developing *interpretable* machine learning models.

The precise meaning of the term *interpretability* can depend on the context, and there is no consensus how this term should exactly be defined. For example, Doshi-Velez and Kim (2017) define interpretability as "the ability to explain or to present in understandable terms to a human". Ribeiro et al. (2016) define an interpretable model as one that "can be readily presented to the user with visual or textual artifacts". Lipton (2016) outlines various desiderata for interpretability, and categorises interpretable models into two classes: one relating to *transparency*, and the other to *post-hoc interpretability*. The former relates to model design and typically *global* behaviour of the model, whereas the latter corresponds to approaches that extract information from learnt models and are typically *local*. Transparency can be present either at the level of the entire model (this includes

[2]Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)

e.g. linear models or decision trees) or at the level of individual components which is referred to as *decomposability*.

Various pre-existing techniques have focused on *local* explanations that are specific to a particular data point. For example, Ribeiro et al. (2016) present Local Interpretable Model-agnostic Explanations (LIME) which rely on locally simple, interpretable models (such as sparse linear models), and Lundberg and Lee (2017) propose a way to assign feature importance values to each input for a particular prediction. However, methods that are inherently *local* do not let us directly characterise *global* associations that the model has learnt. Arguably, the latter are important for scientific discovery applications, e.g. for pattern discovery in genomics data, where research questions such as "which genes are most strongly associated with this latent variable" or "which genes follow the same expression dynamics" can be naturally answered using global associations.

Therefore, in this thesis, our goal is to develop methodology to answer such research questions in the context of flexible non-linear models (such as Gaussian Processes or neural networks), and we focus on introducing *transparency* into these non-linear modelling frameworks. Instead of approximating complex non-linear models with simpler models, our work will focus on directly incorporating transparency within these black-box models via the notion of *decomposition*.

1.8 Goals and contributions

The methodological contributions in this thesis all share the same broad goal: our work aims to improve the interpretability of non-linear latent variable models, specifically focusing on explainability in the context of gaining insights into complex high-dimensional tabular data sets that arise e.g. in computational biology.

Motivating example (disease progression modelling): Studying disease progression (such as a particular subtype of cancer) using omics data can provide valuable insights into the underlying biological mechanisms which can potentially lead to improved diagnoses and therapies. As *progression* is a continuous-time

process by its nature, studying it would ideally require repeated measurements of the same patients as the disease progresses. In practice, such studies are scarce and have limited sample sizes. Most data sets available are *cross-sectional*, i.e. they capture a "snapshot" of the population. The scientific question of interest is *can we utilise cross-sectional data for disease progression modelling*. The approach we propose is to utilise latent variables to capture this underlying continuous process from cross-sectional data. Introducing a one-dimensional latent variable, we could map the molecular measurements to an axis which would hopefully correspond to the underlying molecular disease progression, as illustrated in Figure 1.3.

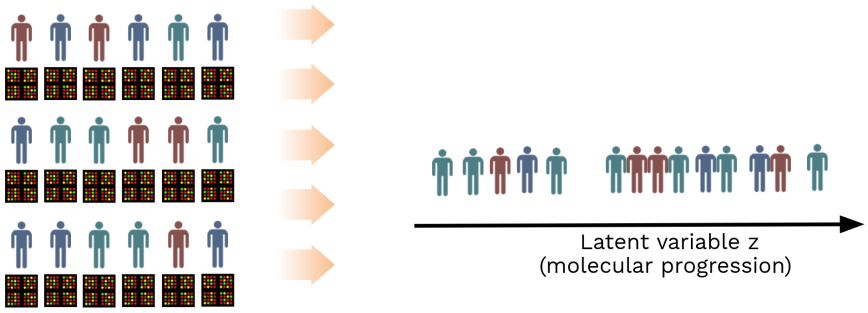


Figure 1.3: Hypothetical scenario: One-dimensional latent variable **z** has identified the underlying molecular disease progression from high-dimensional molecular measurements.

Generally, upon fitting a latent variable model, we would hope to retrospectively interpret what the latent variable captures. Next, we outline research questions that have motivated our contributions.

Research question 1: How can we incorporate covariate information in a Latent Variable Model?

This is important because often in practice there is prior knowledge on which covariates are important in terms of characterising variation in the data. For example, we might know about the presence of multiple subpopulations in our data set. For images of hand-written digits, the subpopulations could correspond to different digit classes, whereas in genomics applications, there might be different disease subtypes that have distinct gene expression patterns, as illustrated in Figure 1.3.

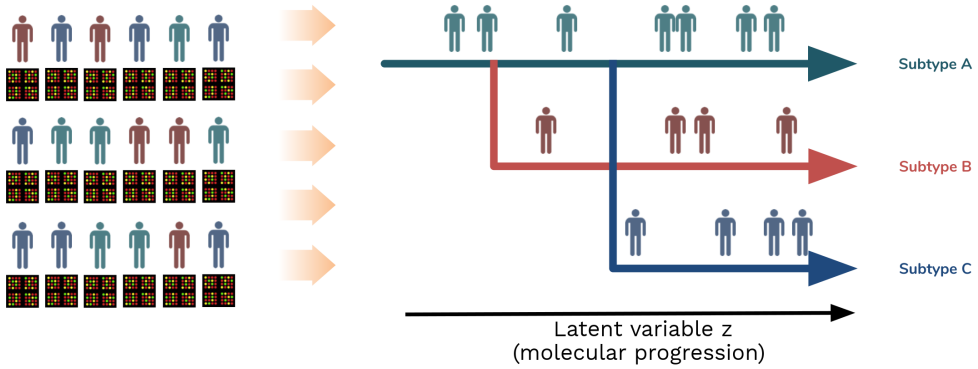


Figure 1.4: Illustration of a similar scenario as in Figure 1.3, but in the presence of multiple disease subtypes.

In chapter 2, we specifically focus on incoporating covariate information within the GPLVM. For this purpose, we develop a hybrid GP modelling framework which we call c-GPLVM, combining fixed inputs (as in the GP-regression model) and latent inputs (as in the GPLVM) (Märtens et al., 2019b). Furthermore, we explore how to do this when the covariate is censored, e.g. patient survival data.

Having incorporated both latent variables and covariates in a non-linear modelling framework, this brings us naturally to the next challenge: *interpretation*. That is, how to interpret what the model has learnt. For example, we would like to understand what do the inferred latent coordinates $\mathbf{z}$ correspond to. For high-dimensional tabular data, answering this question is not straightforward. The latent space would typically have to be interpreted in terms of the original features (e.g. genes) themselves. This requires characterising how the latent space is mapped to the feature space. While being relatively easy to interpret in a linear latent variable model (such as pPCA), this task is challenging in non-linear models, thus motivating the second broad research question:

Research question 2: How can we characterise feature-level variation in non-linear Latent Variable Models?

Specifically, we approach this question from the perspective of *variance decomposition*. That is, we aim to characterise the relative contribution of every input (i.e. either a latent coordinate or an observed covariate) across all features using

the *fraction of variance explained*. Note that we will have to distinguish between marginal effects (e.g. the effect of a particular latent variable) and interaction effects (e.g. the interaction between this latent variable and a covariate).

To achieve this, we adopt the notion of functional ANOVA decomposition, in order to aid interpretability in non-linear latent variable models. Specifically, we explore the adaptation of functional ANOVA separately in two classes of models. First, in Chapter 3, we focus on achieving this in Gaussian Process models, such as GPLVM and c-GPLVM (Märtens et al., 2019b). In Chapter 4, we investigate how to enable such decompositions in neural network models, such as VAE and CVAE (Märtens and Yau, 2020b).

We show how naive attempts to provide a variance decomposition can lead to misleading interpretations due to unidentifiability, and discuss how to obtain an identifiable functional ANOVA decomposition via functional constraints. For GP-based models such a decomposition can be elegantly achieved in closed form, whereas for VAEs we have to turn to constrained optimisation techniques. Despite the fact that in the latter case we have traded off analytic constraints with approximate ones, the VAE-based approach comes with significant gains in computational scalability.

Feature-level variance decompositions can serve two goals: First, it can aid us in finding an interpretation for the latent coordinate (e.g. by inspecting features that have the highest variance contribution by this latent variable). Second, it can indirectly help us identify similar features (e.g. we can identify groups of features that do not exhibit a covariate effect).

This is a common down-stream task in genomics data analysis, where e.g. it might be of interest to identify groups of genes with similar expression dynamics. However, as opposed to a two-stage data analysis workflow, where one would first apply dimensionality reduction and then cluster features, it would be desirable to have a joint model to do these two tasks simultaneously. This brings us to the third motivating question:

Research question 3: How can we construct a robust, scalable, and interpretable approach to clustering features within the VAE framework?

VAEs are an increasingly popular approach for latent variable modelling, but it is not obvious whether they can be extended for the purpose of clustering features. In Chapter 5, we investigate how to embed a clustering prior within the VAE framework, and how to additionally incorporate the notion of translation invariance (Märtens and Yau, 2020a).

2

Covariate Gaussian Process Latent Variable Models

Contents

2.1 Introduction 35
2.2 Covariate-adjusted dimensionality reduction 36
2.3 Covariate-GPLVM 38
2.3.1 Kernel design 40
2.3.2 Inference for c-GPLVM 42
2.4 Partially observed and censored covariates 43
2.4.1 Missing categorical covariates 43
2.4.2 Censored covariates 45
2.5 Experiments 46
2.5.1 Synthetic data: Circles and Pinwheel 47
2.5.2 MNIST 49
2.5.3 Censored survival times: synthetic data 51
2.5.4 Real-world data: The Cancer Genome Atlas 52
2.5.4.1 Prediction of survival times 53
2.5.4.2 Survival-adjusted cancer modelling 53
2.6 Related work 55
2.7 Discussion 56

2.1 Introduction

The goal of performing dimensionality reduction is to understand and characterise what are the patterns in our data set. However, often there is some *a priori* knowledge about the known factors that explain structure in the data. We refer

to this as *covariate* information. For example, for the widely used MNIST data set we know that there are ten classes of hand-written digits (even though this information is often disregarded when performing dimensionality reduction on this data). In biomedical applications, there is often some domain knowledge which covariates are relevant for a particular problem, e.g. in cancer genomics there might be known disease subtypes which exhibit different gene expression patterns. In these situations, it would be desirable to incorporate such covariate information as part of the latent variable model. In this chapter, we propose a way to achieve this in the context of Gaussian Process based dimensionality reduction, extending the Bayesian GPLVM to incorporate covariates.

2.2 Covariate-adjusted dimensionality reduction

More formally, suppose that in addition to observed data $\mathbf{y}_i \in \mathbb{R}^P$, each data point i is also associated with an additional C-dimensional covariate vector, $\mathbf{c}_i \in \mathbb{R}^C$ which may act to modulate the variation in the data. These covariates may either be continuous-valued measurements (e.g. biomarkers), categorical labels (e.g. denoting known disease subpopulations), or censored information (e.g. patient survival times).

In the presence of this additional covariate information, it can be used to guide dimensionality reduction in two different ways:

- with the goal to learn a latent representation that is *predictive* of the covariate, or
- with the goal to learn a representation that has been *adjusted* for the covariate.

The former class of approaches would aim to model both $\mathbf{y}_i$ and $\mathbf{c}_i$ as a function of $\mathbf{z}_i$, thus including methods such as Partial Least Squares (Geladi and Kowalski, 1986), supervised probabilistic PCA (Yu et al., 2006) and supervised-GPLVM (Gao et al., 2011). The latter approach, on the other hand, would seek to model $\mathbf{y}_i$ as a function of both $\mathbf{z}_i$ and $\mathbf{c}_i$. The conceptual difference between these approaches has been illustrated in Figure 2.2 panels (b) and (c). Note that the former is most useful in scenarios where class discrimination is the ultimate objective, whereas the

latter could be more useful for discovery applications. This is because being able to adjust for covariate effects can potentially reveal different latent structure in the data that was not revealed before. For example, in the presence of batch effects (in this context, batches can be treated as covariate information), it would be necessary to correct for these effects in order to identify meaningful latent structure in the data.

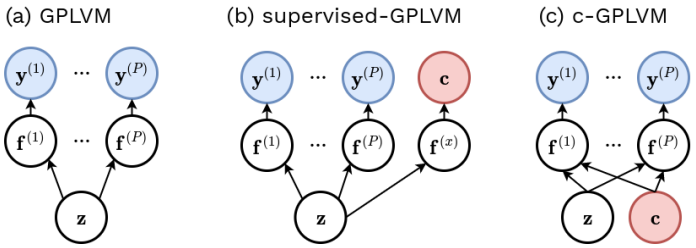


Figure 2.1: Graphical models for (a) GPLVM, (b) a particular implementation of supervised-GPLVM, and (c) c-GPLVM.

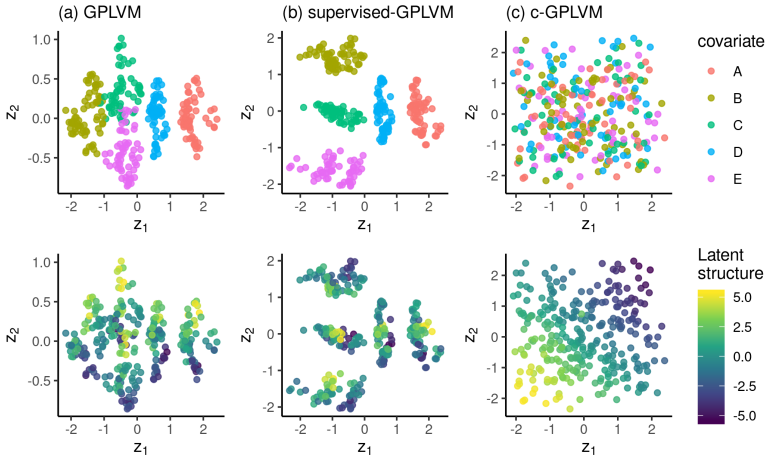


Figure 2.2: The latent $\mathbf{z}$ inferred by c-GPLVM differs fundamentally from what is inferred by GPLVM and supervised-GPLVM. For all three methods, their learned 2D latent space $\mathbf{z}$ has been shown twice, coloured according to the categorical covariate (top panel) and the underlying latent structure (bottom panel). In the presence of both between-cluster effects (highlighted in top row) and shared within-cluster latent structure (bottom row), the $\mathbf{z}$ inferred by GPLVM reflects the former but not the latter. This effect is even stronger for supervised-GPLVM which encourages $\mathbf{z}$ to *discriminate* between the clusters. Instead, c-GPLVM will *adjust* for the cluster effects and thus reveal the latent structure which is *shared* between all clusters.

In this work, we specifically focus on incorporating covariates within the GPLVM framework. As discussed, there are various ways how this can be done. We have illustrated both the *discriminative* and the *covariate-adjusted* approaches

as graphical models (see Figure 2.1). We refer to the former as the supervised-GPLVM, whereas the latter is the approach we propose (c-GPLVM). We have also demonstrated the implications of these choices in practice using a synthetic data set as shown in Figure 2.3. This Figure illustrates our setting of interest where features in the data exhibit dependence on both, an observed categorical covariate (5 discrete classes A-E) and a two-dimensional latent variable (in this case we have access to the true latent variable). The dominant effect of these 5 classes means that the GPLVM will learn a latent space that reflects the presence of these groups. This is further exaggerated by the supervised-GPLVM which acts to increase the separation between the classes in the latent space. In contrast, our proposed approach (c-GPLVM) would seek to find the common *shared* structure between the 5 classes, and infer the latent space which would be adjusted for the presence of the 5 classes. In linear modelling, this process would be referred to as "regressing out" the class-specific effects.

If the five different classes corresponded to five subtypes of a particular disease, both the GPLVM and the supervised-GPLVM would find a latent space that would enable a low-dimensional characterisation of the five subtypes. The covariate-adjusted model (c-GPLVM) on the other hand would seek to find what aspects do these five subtypes have in common.

2.3 Covariate-GPLVM

Figure 2.3 illustrates our scenario of interest from a different perspective, i.e. from the perspective of mappings $f^{(j)} : \mathbf{z} \mapsto \mathbf{y}^{(j)}$ for features $j = 1, \ldots, P$ that are confounded due to either a continuous covariate (in panel (A)) or a discrete one (in panel (B)).

This confounding due to $\mathbf{c}$ makes the standard GPLVM a mis-specified model. This is because the noise conditional on the mappings, i.e. $p(\mathbf{y}^{(j)}|f^{(j)}(\mathbf{z}))$, is not i.i.d. any more. The effects of this can be seen from the posterior predictive mean and

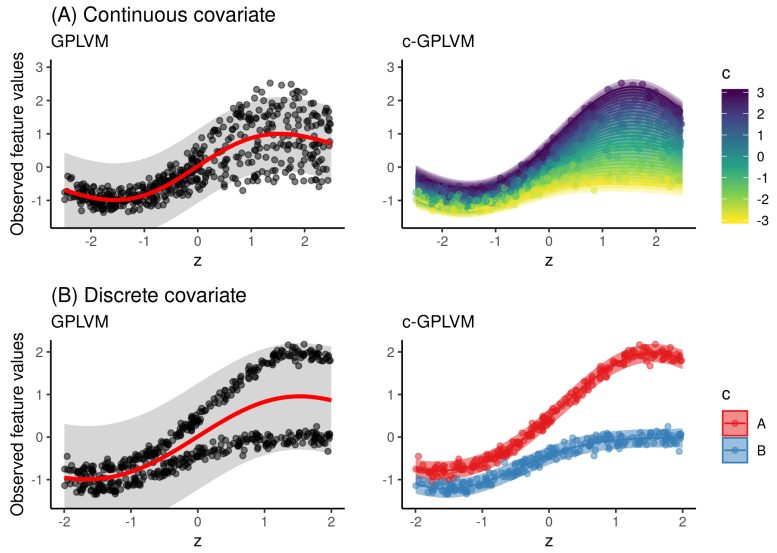


Figure 2.3: The presence of covariate effects confounds the mappings from the underlying latent $\mathbf{z}$ (on x-axis) to the feature space (y-axis). This has been illustrated on synthetic data for (a) a continuous covariate, and (b) a categorical one. Standard GPLVM (on the left) necessarily ignores covariate information, whereas its inclusion in c-GPLVM (on the right) allows it to capture a variety of nonlinear covariate effects. Red lines show the GP posterior mean and shaded areas the (95%) posterior predictive uncertainty. For c-GPLVM, both of these quantities additionally vary with $\mathbf{c}$.

uncertainty learnt by the GPLVM – in addition to being a mis-specified model, its predictive capabilities also suffer.

Our proposal, which we call the *covariate-GPLVM* (c-GPLVM), specifically focuses on learning a set of mappings $f^{(j)} : (\mathbf{z}, \mathbf{c}) \mapsto \mathbf{y}^{(j)}$ which are defined on the joint space of $\mathbf{z}$ and $\mathbf{c}$. This construction lets us capture non-linear covariate effects (potentially including non-linear interactions between $\mathbf{c}$ and $\mathbf{z}$), as illustrated in Figure 2.3. Now the predictive mean and uncertainty become the function of both $\mathbf{z}$ and $\mathbf{c}$. The c-GPLVM can be seen as a hybrid between the GP regression and the GP latent variable model, where the input space consists of two parts: fixed covariates $\mathbf{c}$ and unobserved latent coordinates $\mathbf{z}$. In contrast, in the former the inputs are fixed, whereas in the latter the inputs are random variables and they are inferred.

2.3.1 Kernel design

Different assumptions about the form of interaction between $\mathbf{z}$ and $\mathbf{c}$ can be made, and these can be encoded via different kernel structures. One approach would be to define the ARD kernel on this joint space:

$$k^{\texttt{int}}\left((\mathbf{c},\mathbf{z}),(\mathbf{c}',\mathbf{z}')\right) := \sigma_{cz}^2 \exp\left[-\frac{1}{2}\sum_{j=1}^{C}\left(\frac{c_j - c_j'}{l_j^{(c)}}\right)^2 - \frac{1}{2}\sum_{j=1}^{Q}\left(\frac{z_j - z_j'}{l_j^{(z)}}\right)^2\right].$$

Alternatively, one could define an additive kernel

$$k^{\texttt{add}}\left((\mathbf{c},\mathbf{z}),(\mathbf{c}',\mathbf{z}')\right) := k^c(\mathbf{c},\mathbf{c}') + k^z(\mathbf{z},\mathbf{z}')$$

where both k^c and k^z are ARD kernels.

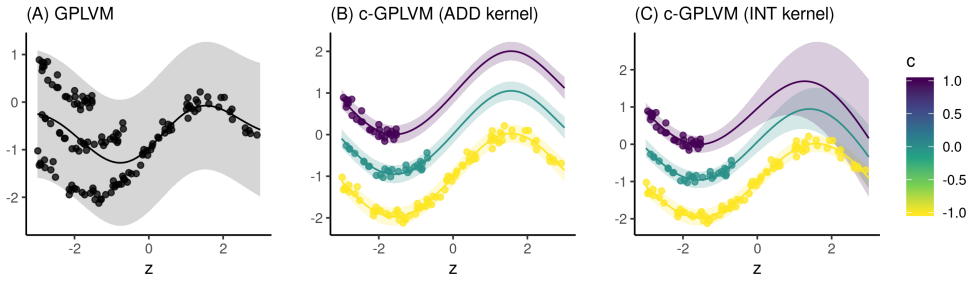


Figure 2.4: ADD and INT kernels have different behaviour when prediction requires extrapolation from training data. Synthetic observed data set is shown as dots ($\mathbf{z}$ on x-axis, selected feature on y-axis, coloured by covariate $\mathbf{c}$). Predictions from the c-GPLVM shown together with predictive uncertainty for $\mathbf{c} \in \{-1, 0, 1\}$ values.

Choosing to use an additive (ADD) kernel or an interaction (INT) kernel correspond to different modelling assumptions. We have illustrated the implications of this on a synthetic example in Figure 2.4 in the presence of sampling bias. Here the covariate $\mathbf{c}$ takes values $\mathbf{c} \in \{-1, 0, 1\}$ and the latent variable $\mathbf{z}$ values cover the entire range from -3 to 3. However, not all combinations of $(\mathbf{z}, \mathbf{c})$ are equally represented in the training data. In fact, only observations with $\mathbf{c} = -1$ cover the entire range of $\mathbf{z}$ values, whereas making predictions for covariate values $\mathbf{c} \in \{0, 1\}$ will require extrapolation. Figure 2.4 illustrates the difference between the additive (ADD) and interaction (INT) models: the former extrapolates with high confidence[1], whereas the latter becomes quickly uncertain.

[1]From the perspective of an additive model, our prediction task does not require extrapolation, as both $\mathbf{z} \in [-3, 3]$ and $\mathbf{c} \in \{-1, 0, 1\}$ values have been observed

Even though the ADD model is more restrictive, it is more data-efficient when the feature dependence w.r.t. to $\mathbf{z}$ is stationary, and can extrapolate well in scenarios when the non-stationarity of the INT model provides too much uncertainty. The INT model, on the other hand, provides increased flexibility than can be desirable for predictive applications.

The INT kernel, however, lacks straightforward interpretability. Even though the squared exponential ARD kernel is not completely a black box, as it has a length-scale parameter per input dimension, their interpretation is not straightforward. Despite the name *Automatic Relevance Determination* their values do not directly indicate the relevance of inputs, instead the length-scales rather measure the extent of non-linearity. Thus the inverse length-scale values cannot be generally used to determine the *relevance* of inputs, as has been shown before (Paananen et al., 2019). We have also shown it empirically for c-GPLVM in Figure 2.5. Note that even though the three scenarios in panels (A, B, C) exhibit different types of dependence on covariate $\mathbf{c}$, it is challenging to distinguish between them based on the INT kernel, as the inferred (inverse) lengthscales are similar across the three scenarios. However, in practice we would like to distinguish between these three scenarios: (A) does not exhibit $\mathbf{c}$ effects, whereas the second scenario (B) has an additive $\mathbf{c}$ effect and (C) exhibits an interaction between $\mathbf{z}$ and $\mathbf{c}$.

A natural question to ask would be whether the benefits of ADD and INT kernels can be combined, automatically separating out the additive effects from interactions. That is, one could hope to specify the following ADD+INT kernel

$$k^{\texttt{add+int}}\left((\mathbf{z},\mathbf{c}),(\mathbf{z}',\mathbf{c}')\right) := k^{c}(\mathbf{c},\mathbf{c}') + k^{z}(\mathbf{z},\mathbf{z}') + k^{\texttt{int}}\left((\mathbf{c},\mathbf{z}),(\mathbf{c}',\mathbf{z}')\right).$$

However, as we discuss in Chapter 3, this kernel decomposition suffers from unidentifiability, and thus further constraints need to be introduced for a meaningful decomposition. For the experiments in this chapter, we resort to using the INT kernel.

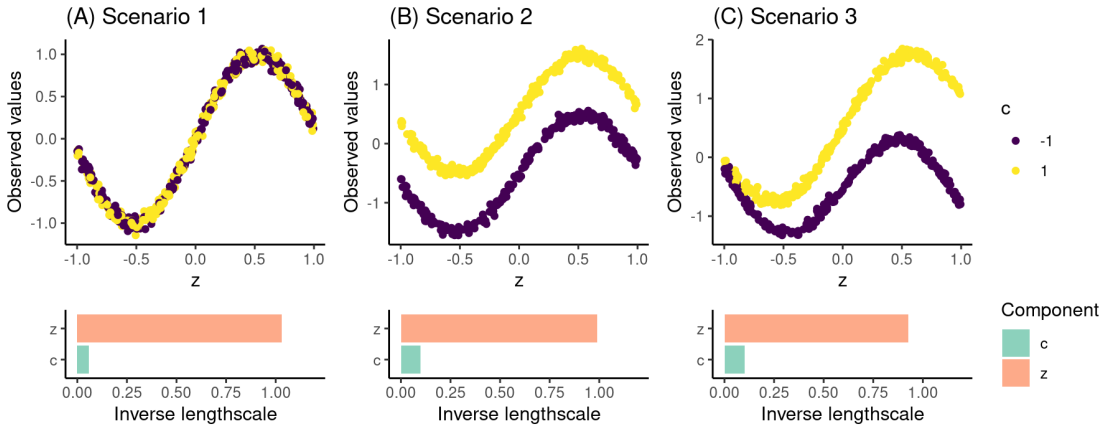


Figure 2.5: Distinguishing between the three different data generative processes is challenging when using the INT kernel. For three scenarios (panels A-C), we have shown the observed feature values (on y-axis) as a function of known $\mathbf{z}$ (on x-axis) and $\mathbf{c} \in \{-1, 1\}$ (colour coding), and the corresponding inferred inverse length-scales as part of the INT kernel.

2.3.2 Inference for c-GPLVM

As c-GPLVM can be seen as GPLVM with additional fixed inputs, inference for c-GPLVM is analogous to GPLVM inference.

Inference for c-GPLVM (without inducing points): We first discuss the scenario where we do not need an inducing point approximation. As the lower bound presented in Section 1.5.3 in equation (1.5), the ELBO for c-GPLVM will be closely related

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\mathbf{Z})} \log p(\mathbf{Y}|\mathbf{Z}, \mathbf{C}) - \sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{z}_i)||p(\mathbf{z}_i)) \\ &= \sum_{j=1}^{P} \mathbb{E}_{q(\mathbf{Z})} \mathcal{N}(\mathbf{y}^{(j)}|\mathbf{0}, \mathbf{K}^{(j)}_{(zc)(z'c')} + \sigma_j^2 \mathbf{I}) - \sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{z}_i)||p(\mathbf{z}_i)) \end{aligned} \tag{2.1}$$

where $\mathbf{K}^{(j)}$ now involves kernel evaluations on pairs $(\mathbf{z}, \mathbf{c}), (\mathbf{z}', \mathbf{c}')$ which involves both latent and fixed inputs. We evaluate this ELBO using the reparameterisation trick, sampling the latent variables from their respective approximate posteriors $\mathbf{z}_i \sim q(\mathbf{z}_i)$.

Inference for sparse c-GPLVM (with inducing points): Analogously to Section 1.5.4, in addition to latent variables $\mathbf{z}_i$ for every data point $i = 1, \ldots, N$, we introduce M inducing inputs. In contrast to the GPLVM, now the inducing

inputs will lie in the joint $(\mathbf{z}, \mathbf{c})$ space, i.e. we introduce $(\mathbf{z}_i^u, \mathbf{c}_i^u)$ for $i = 1, \ldots, M$, denoted collectively by $(\mathbf{Z}_u, \mathbf{C}_u) \in \mathbb{R}^{M \times (D+C)}$, and the corresponding inducing outputs $\mathbf{U} := \{\mathbf{u}^{(j)}\}$. Similarly to the GPLVM setup, we formulate the generative model as follows

$$p(\mathbf{Y}, \mathbf{F}, \mathbf{U}, \mathbf{Z} | \mathbf{Z}_u, \mathbf{C}_u, \mathbf{C}) = p(\mathbf{Z}) \prod_{j=1}^{P} p(\mathbf{y}^{(j)} | \mathbf{f}^{(j)}) p(\mathbf{f}^{(j)} | \mathbf{u}^{(j)}, \mathbf{Z}, \mathbf{C}, \mathbf{Z}_u, \mathbf{C}_u) p(\mathbf{u}^{(j)} | \mathbf{Z}_u, \mathbf{C}_u),$$

where now the GP prior $p(\mathbf{u}^{(j)} | \mathbf{Z}_u, \mathbf{C}_u) = \mathcal{N}(\mathbf{0}, \mathbf{K}^{(j)}_{(zc),(z'c')})$ is defined on pairs $(\mathbf{Z}_u, \mathbf{C}_u)$ and the prior predictive $p(\mathbf{f}^{(j)} | \mathbf{u}^{(j)}, \mathbf{Z}, \mathbf{C}, \mathbf{Z}_u, \mathbf{C}_u)$ now additionally includes fixed inputs $\mathbf{C}$ and inducing inputs $(\mathbf{Z}_u, \mathbf{C}_u)$.

Following the derivation of the collapsed bound as discussed in Section 1.5.4, we get the collapsed ELBO that has a very similar form to equation (1.6) with the difference that all kernels are now evaluated on pairs $(\mathbf{z}_i, \mathbf{c}_i)$.

2.4 Partially observed and censored covariates

Here we show how this construction will let us additionally handle partially missing or censored covariates. We specifically focus on two cases: missing categorical covariates (which we will use in the MNIST experiment), and censored covariates (which we will use for incorporating survival information).

We derive these inference schemes for the case where we are fitting the c-GPLVM via optimising the ELBO in equation (2.1). Adapting the inducing point version is analogous.

2.4.1 Missing categorical covariates

We denote the collection of missing covariates $\mathbf{C}^{\text{mis}}$ and fully observed set by $\mathbf{C}^{\text{obs}}$. Here, we focus on the case where the covariates are discrete. We treat missing covariates as latent variables with a Categorical prior, i.e.

$$p(\mathbf{C}^{\text{mis}}) = \prod_{i \in \text{mis}} p(\mathbf{c}_i) = \prod_{i \in \text{mis}} \text{Categorical}(\mathbf{c}_i | \pi_1, \ldots, \pi_K)$$

with $\pi_k = 1/K$.

For a modular implementation of scalable variational inference, it would be desirable to apply the sampling-based reparameterisation trick. For this purpose, we rely on the continuous relaxation of Categorical random variables (Maddison et al., 2017; Jang et al., 2017), also known as the Concrete random variables.

The Concrete distribution $\mathbf{x} \sim \text{Concrete}(\boldsymbol{\alpha}, T)$ with location parameters $\boldsymbol{\alpha} \in (0, \infty)^K$ and temperature $T \in (0, \infty)$ places probability mass over the simplex $\{\mathbf{x} \in \mathbb{R}^K : x_k \in [0, 1], \sum_k x_k = 1\}$, and it has the following useful properties:

- First, $\mathbf{x} = (x_1, \ldots, x_K) \sim \text{Concrete}(\boldsymbol{\alpha}, T)$ can be sampled using reparameterisation,
$$x_k = \frac{\exp((\log \alpha_k + G_k)/T)}{\sum_{i=1}^K \exp((\log \alpha_i + G_i)/T)}$$
where G_k are Gumbel random variables, $G_k := -\log(-\log U_k), U_k \sim U(0, 1)$.
- Second, in the zero-temperature limit we obtain the Categorical random variable, i.e.
$$\mathbb{P}\left(\lim_{T \to 0} x_k = 1\right) = \frac{\alpha_k}{\sum_{i=1}^K \alpha_i}$$

The temperature parameter T lets us control how peaked Concrete distribution is. A common strategy is to use an annealing schedule, i.e. to gradually decrease temperature during training (Balın et al., 2019).

For a coherent treatment, we use the Concrete relaxation in both the prior $p(\mathbf{C}^{\text{mis}})$ as well as the approximate posterior $q(\mathbf{C}^{\text{mis}})$. That is, we modify the generative model as follows

$$p(\mathbf{C}^{\text{mis}}) = \prod_{i \in \text{mis}} p(\mathbf{c}_i) = \prod_{i \in \text{mis}} \text{Concrete}(\mathbf{c}_i | \pi_1, \ldots, \pi_K)$$

and introduce a variational approximation

$$q(\mathbf{C}^{\text{mis}}) = \prod_{i \in \text{mis}} q(\mathbf{c}_i) = \prod_{i \in \text{mis}} \text{Concrete}(\mathbf{c}_i | \boldsymbol{\lambda}_i)$$

where $\boldsymbol{\lambda}_i$ are variational parameters.

As a result, we can modify the c-GPLVM lower bound in equation (2.1) accordingly:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{C}^{\text{mis}})q(\mathbf{Z})} \log p(\mathbf{Y} | \mathbf{Z}, \mathbf{C}^{\text{obs}}, \mathbf{C}^{\text{mis}}) - \sum_{i=1}^N \text{KL}(q(\mathbf{z}_i) || p(\mathbf{z}_i)) - \sum_{i \in \text{mis}} \text{KL}(q(\mathbf{c}_i) || p(\mathbf{c}_i)).$$

2.4.2 Censored covariates

Next, we discuss inference in the scenario when some of the inputs $\mathbf{c}$ have not been observed, but instead have been censored. In practice, this happens when we want to use patient survival times in the role of covariate, but for many individuals we only have access to their last follow-up time. We treat this as a lower bound on their true survival. In this section, we assume $\mathbf{c}_i$ are one-dimensional and represent survival times (denoted by c_i).

Suppose for some individuals $i \in \text{obs}$ we have observed their true survival c_i, whereas for the rest we will treat c_i as latent. For the latter, we additionally have observed a lower bound a_i which we want to incorporate into our model, as this information is informative of the posterior of c_i.

Following a common assumption in survival analysis, we assume *a priori* Weibull-distributed patient survival, i.e. for all individuals i we specify a prior

$$p(c_i) = \text{Weibull}(k_0, \lambda_0)$$

with hyperparameters k_0, λ_0. Next, we incorporate censoring information by conditioning on survival c_i exceeding a_i, and potentially also upper bounding by the assumed maximum lifespan b_i (if one does not want to make such an assumption, b_i can be set to ∞). The conditional prior takes the form

$$p(c_i \mid a_i \leq c_i \leq b_i) = \text{TruncatedWeibull}_{[a_i,b_i]}(k_0, \lambda_0)$$

which has a closed form density.

Next we construct a variational inference scheme which will allow us to fit c-GPLVM in the presence of censored covariates as well as to infer the posteriors over true survival times. We need to choose the family of approximating distributions $q(c_i)$ such that they would have the same support as $p(c_i)$. We choose it to be the truncated Gaussian distribution which has been restricted to the interval $[a_i, b_i]$, i.e. we choose $q(c_i) = \mathcal{TN}_{[a_i,b_i]}(\mu_i, \sigma_i^2)$ where μ_i, σ_i^2 are variational parameters. In addition to having appropriate support, this choice lets us perform efficient

reparameterisation-based inference (Kingma and Welling, 2014; Rezende et al., 2014).

This lets us sample from $q(c_i)$ via reparameterisation:

$$\begin{aligned} F(a_i) &:= \Phi((a_i - \mu_i)/\sigma_i) \\ F(b_i) &:= \Phi((b_i - \mu_i)/\sigma_i) \\ \varepsilon &:= F(a) + u(F(b) - F(a)) \text{ with } u \sim U(0,1) \\ c_i &\sim \Phi^{-1}(\varepsilon)\sigma_i + \mu_i \end{aligned}$$

where $\Phi(\cdot)$ is the cdf of the standard normal.

Denoting the censored set of survival times by $\mathbf{C}^{\text{cens}}$ and fully observed set by $\mathbf{C}^{\text{obs}}$, we can fit c-GPLVM by optimising the following ELBO:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{C}^{\text{cens}})q(\mathbf{Z})} \log p(\mathbf{Y}|\mathbf{Z}, \mathbf{C}^{\text{obs}}, \mathbf{C}^{\text{cens}}) - \sum_{i=1}^{N} \text{KL}(q(\mathbf{z}_i)||p(\mathbf{z}_i)) - \sum_{i \in \text{cens}} \text{KL}(q(c_i)||p(c_i)).$$

2.5 Experiments

Here we compare our proposed c-GPLVM with the standard GPLVM and with the supervised-GPLVM, as shown in Figure 2.1. We additionally compare with a *linear* baseline, i.e.

$$y_i^{(j)} = \mu_j + \sum_{d=1}^{D} \alpha_d^{(j)} z_{id} + \sum_{k=1}^{C} \beta_k^{(j)} c_{ik} + \sum_{d=1}^{D}\sum_{k=1}^{C} \gamma_{dk}^{(j)} z_{id} c_{ik} + \varepsilon_{ij}$$

which we refer to as Latent Variable Regression (LVR) (Burnham et al., 1996; Campbell and Yau, 2018b), which can be seen as a linear counterpart of c-GPLVM. Even though it is less flexible, it has an advantage over non-linear black-box methods in terms of interpretability. The potential interpretability of c-GPLVM will be discussed in detail in the next chapter.

First, we demonstrate the utility of c-GPLVM on two toy problems, then on MNIST images of hand-written digits, and finally we consider a real-life cancer genomics example where we investigate how c-GPLVM lets us incorporate survival information.

2.5.1 Synthetic data: Circles and Pinwheel

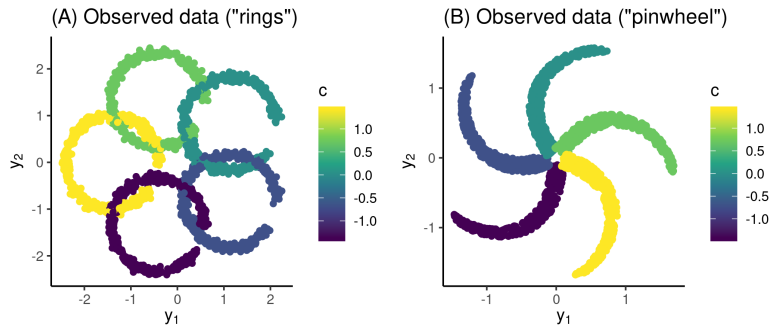


Figure 2.6: Observed 2D toy data with additional covariate information (continuous-valued $\mathbf{c} \in \mathbb{R}$) shown in colour.

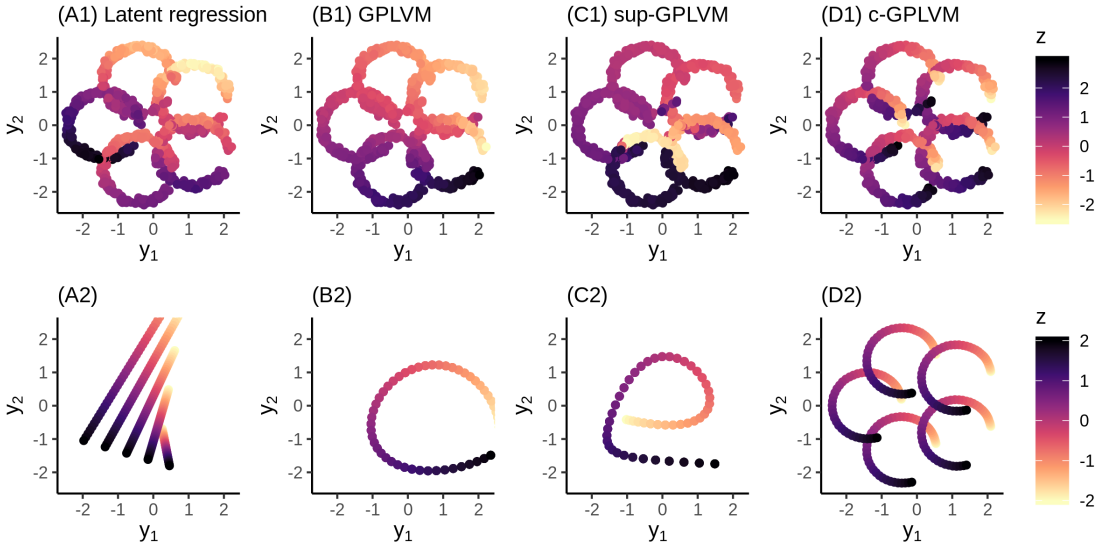


Figure 2.7: Comparison of models on the "rings" data. Top row: data coloured according to the inferred one-dimensional $\mathbf{z}$. Bottom row: Mean of the posterior predictive in the 2D observation space when varying $\mathbf{z} \in [-2, 2]$ and fixing $\mathbf{c}$ to the five observed values.

First, we consider synthetic two-dimensional data sets as shown as displayed in Figure 2.6 ("rings" and "pinwheel"). In the observation space, each circle (panel A) or spoke (panel B) corresponds to a *trajectory* and we would like to understand the shared properties of how features vary *along* each trajectory irrespective of the location of each circle or the *angle* of each spoke. If we had additional, *covariate* information on the clustering structure in the form of this angle (shown by colour coding in both panels), we could use this information *jointly* to achieve our goal. Note that the "rings" data has an additive data generating mechanism, whereas

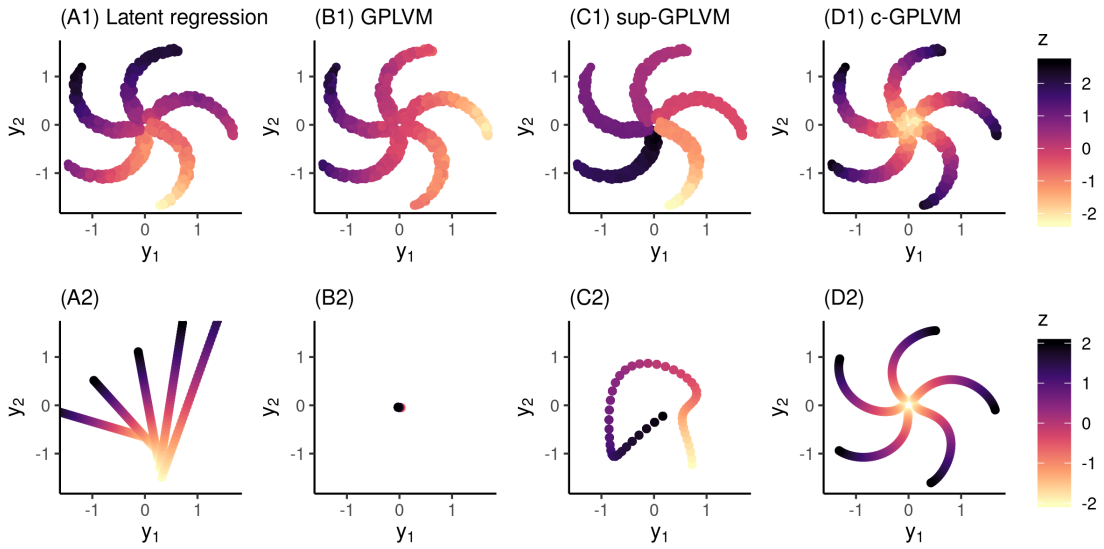


Figure 2.8: Comparison of models on the "pinwheel" data. Top row: data coloured according to the inferred one-dimensional $\mathbf{z}$. Bottom row: Mean of the posterior predictive in the 2D observation space when varying $\mathbf{z} \in [-2, 2]$ and fixing $\mathbf{c}$ to the five observed values.

the "pinwheel" features exhibit a complex non-linear interaction between the angle of the spoke and the position along the trajectory.

The inferred one-dimensional $\mathbf{z}$ values are shown by colour coding in the top row of Figures 2.7 and 2.8 (panels A1-D1 comparing four models). As expected, the $\mathbf{z}$ inferred by GPLVM captures global structure (ignoring covariate information) and $\mathbf{z}$ learned by supervised-GPLVM discriminates between the groups defined by the covariate. The Latent Variable Regression, which adjusts for the covariate in a linear model, struggles due to the highly non-linear signal present in the data.

To gain a better insight into what the models have *learned*, we have also visualised the mean of the posterior predictive distributions in the data space (bottom row of Figures 2.7 and 2.8, same models as in the top row). Here we have varied $\mathbf{z}$ from -2 to 2 and kept fixed $\mathbf{c}$ to its observed five values. We see that c-GPLVM is the only model which has identified the one-dimensional structure along the circular trajectories for the "rings" data and along the spokes for the "pinwheel" data.

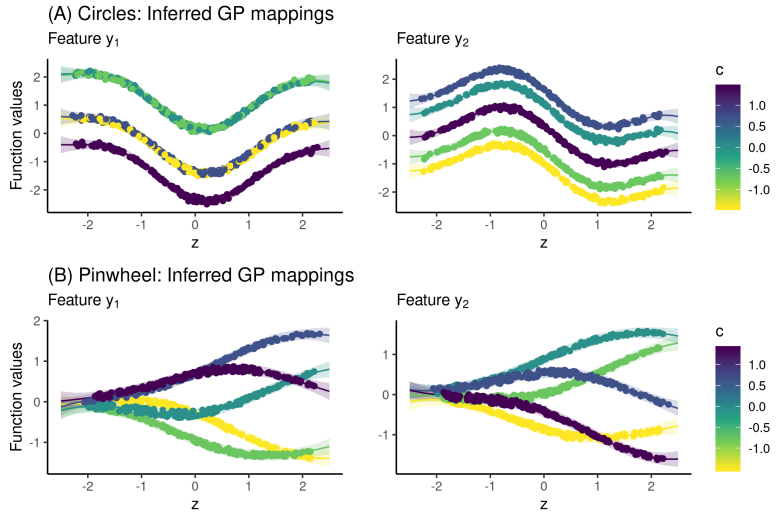


Figure 2.9: The inferred GP mappings shown for both coordinates ($\mathbf{y}_1$ and $\mathbf{y}_2$) of the (A) "circles" and (B) "pinwheel" toy examples, varying $\mathbf{z} \in [-2.5, 2.5]$ (shown on x-axis) and fixing $\mathbf{c}$ to the observed values (shown in colour coding). Lines indicate GP posterior mean, and shaded areas the 95% posterior predictive uncertainty for both features $\mathbf{y}_1$ and $\mathbf{y}_2$, overlaid with observed values.

2.5.2 MNIST

We next consider the MNIST dataset of hand-written digits (LeCun et al., 1998) as another demonstration of how c-GPLVM can reveal meaningful latent structure that is shared between all 10 digit classes. Here, our expectation is that the shared latent structure will reflect some aspect of handwriting style. Figure 2.10 illustrates the one-dimensional representations learned by various models: (a) all-class GPLVM, (b) separate GPLVMs for each class and (c) c-GPLVM by showing posterior draws from each model whilst varying $\mathbf{z}$ values in the interval $\mathbf{z} \in [-3, 3]$. Generating from the c-GPLVM is conditional on the class label, so we display results for all ten classes.

For GPLVM the single latent dimension does not have a straightforward interpretation as it captures both patterns associated with each digit class as well as shared patterns. The per-class approach (b) does reveal handwriting style but note that these are not aligned across classes. The c-GPLVM can be seen as a unified joint model combining class-specific GPLVMs: now the $\mathbf{z}$ space is shared between all classes. As a result, c-GPLVM has learned to capture handwriting style, underlying

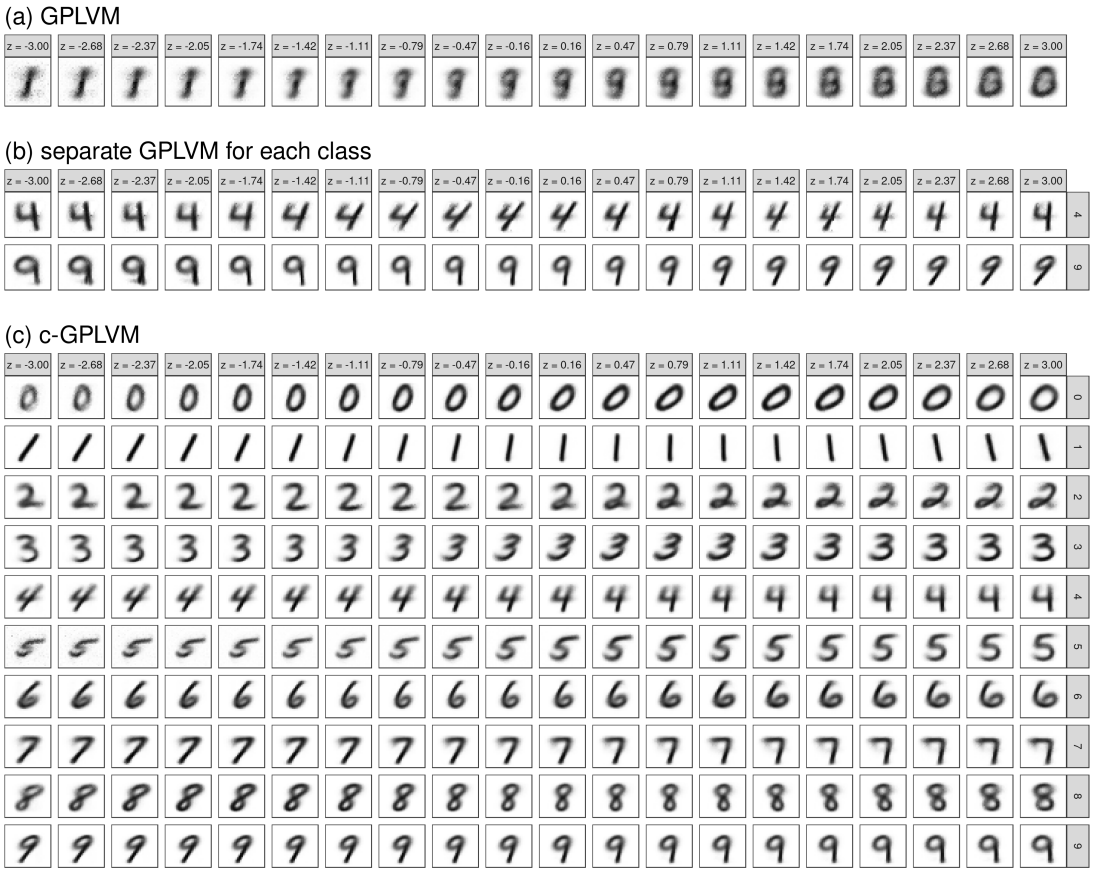


Figure 2.10: Illustration of the inferred one-dimensional $\mathbf{z}$ for (a) GPLVM, (b) separate GPLVMs for each class (shown for 4 and 9), and (c) c-GPLVM. We generated data from the model when varying $\mathbf{z}$ between -3 and 3 (panels from left to right). For data generation in (b) and (c) we also fixed a class label $\mathbf{c}$ (one-hot-encoding of digits 0 – 9 in rows).

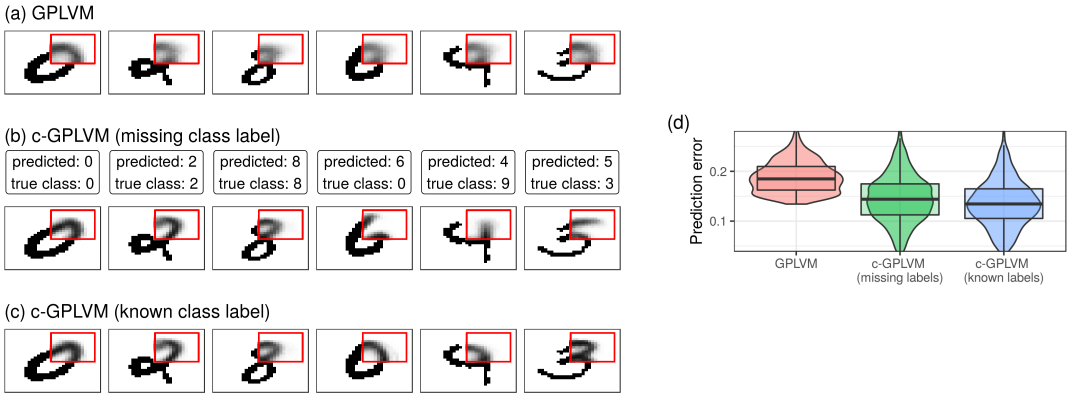


Figure 2.11: Missing data imputation on MNIST. For 5000 digits the pixel values in the upper corner (red rectangle) were set to be missing. Inferred posterior means shown for six selected examples: (a) GPLVM, (b) c-GPLVM with missing class labels $\mathbf{c}$, (c) c-GPLVM with known $\mathbf{c}$. (d) The distribution of mean absolute imputation error over all 5000 data points.

all 10 digits, over a continuum: from slanted and less round digits on the left to the opposite on the right.

Next we demonstrate the benefits of c-GPLVM for inference in the presence of missing data. In principle, there can be missing values either in some of the features $\mathbf{y}^{(j)}$ or covariates $\mathbf{c}$. In c-GPLVM, we are able to handle both types of missingness. To demonstrate this on MNIST, we randomly selected 5,000 images and introduced artificial missingness in both. We removed the upper 1/4 corner of the pixels, and compared the predictive performance of c-GPLVM for the imputation of unobserved pixels under two scenarios: missing class labels and known class labels.

For variational inference with partially missing inputs, we follow the strategy outlined in Section 2.4.1. Figure 2.11 illustratates our experimental setup and displays imputation quality for selected examples (imputation by posterior mean). These were chosen to represent 3 correctly and 3 incorrectly classified data points, to illustrate how missing class labels $\mathbf{c}$ can lead to less accurate imputation in the feature space. The imputation quality for missing pixels has been quantified in Figure 2.11(d). Most accurate predictions are obtained by c-GPLVM with known $\mathbf{c}$ (average imputation error 0.137), but c-GPLVM with unknown inputs $\mathbf{c}$ is only moderately less accurate (average error 0.147). Both of these provide a higher imputation quality than GPLVM (0.186).

2.5.3 Censored survival times: synthetic data

To demonstrate the utility of our inference scheme for censored covariates $\mathbf{c}$, we carried out a synthetic experiment. We generated synthetic data as a function of known $\mathbf{z}$ and $\mathbf{c}$ for four features as shown in Figure 2.12(A), and we wanted to understand the behaviour of our posterior approximation. To mimic a realistic scenario, two of the features (panels “gene 1” and “gene 2”) are informative of survival, whereas the rest are not.

Specifically, in our simulation experiment, we picked two hypothetical observations with $\mathbf{z}_i \in \{-2, 1\}$ as indicated with red dashed lines in panel (A), both having the

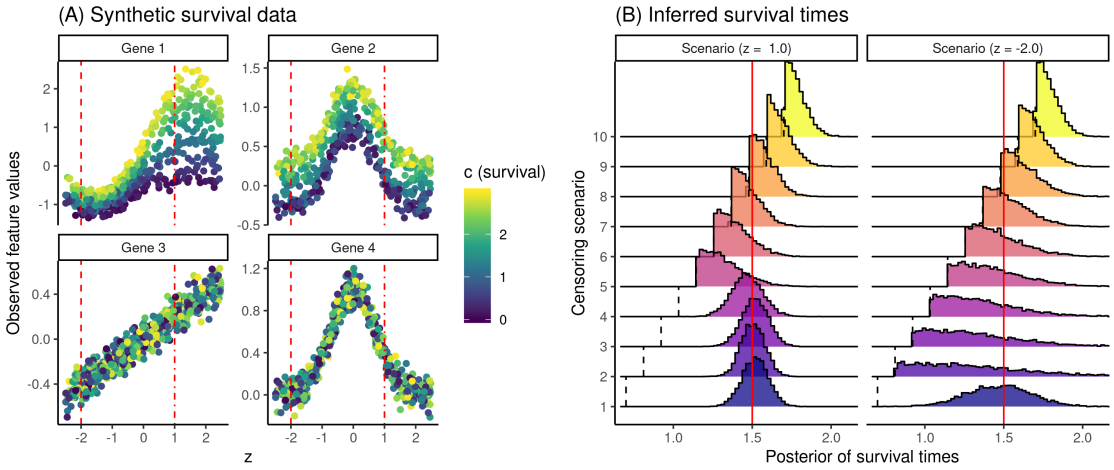


Figure 2.12: Posterior inference for censored survival times on synthetic data. (A) Synthetic four-dimensional data with known ground truth $\mathbf{z} \in \mathbb{R}$ and $\mathbf{c} \in \mathbb{R}^+$, where censoring is carried out for two selected individuals (red dashed lines indicating $\mathbf{z}_i \in \{-2, 1\}$) both with the true $\mathbf{c}_i = 1.5$. (B) We vary the censoring lower bound from 0.7 to 1.7 (ten censoring scenarios on y-axis) for both individuals, lower bound value denoted by small vertical dashed lines. The true $\mathbf{x}$ is always 1.5 (vertical red line).

true survival time $\mathbf{c}_i = 1.5$. We now treated this particular $\mathbf{c}_i$ to be censored, and we varied its lower bound on a grid from 0.7 to 1.7. The resulting approximate posteriors have been shown in Figure 2.12(B). The two subpanels (with $\mathbf{z}_i = -2$ and $\mathbf{z}_i = 1$ respectively) represent two observations which exhibit a different degree of uncertainty: note that the observations for $\mathbf{z}_i = 1$ are more informative of survival. Note that when the lower bound becomes larger than the true survival, the posterior starts to concentrate towards the lower bound.

2.5.4 Real-world data: The Cancer Genome Atlas

We now consider a real world data set consisting of $N = 770$ breast cancer patients from The Cancer Genome Atlas cohort (Weinstein et al., 2013). We use gene expression measurements across 500 most highly variable genes ($\mathbf{Y}$) and survival information for each patient as a covariate ($\mathbf{c}$) to explore the relationship between gene expression changes, survival and a unidimensional latent representation $\mathbf{z}$.

2.5.4.1 Prediction of survival times

To ensure that our modelling assumptions are aligned with the observed data, we first investigate the survival c-GPLVM in a controlled setting. Specifically we focus on its *predictive* ability. That is, we consider the subset of individuals whose death times have been observed ($N = 151$) and carry out artificial censoring in batches of size 5. For 5 patients at a time, we artificially censor their survival time by half a year, and fit c-GPLVM to infer the posterior of these survival times. Predictions have been shown against the true values in Figure 2.13(c).

When considering alternative methods for the prediction of survival times, we note that most models in the field of survival analysis are non- or semi-parametric, i.e. they do not model the baseline hazard function, and thus do not provide a straightforward approach for prediction. To obtain a parametric survival model, one common choice is to use the Weibull distribution. Thus, to compare c-GPLVM with a baseline survival model, we used the Weibull regression model with shrinkage priors as described in (Peltola et al., 2014), predicting patient survival as a function of the gene expression matrix $\mathbf{Y}$. The obtained predictions have been shown in Figure 2.13(a). As the posterior predictive from the Weibull regression does not take into account the fact that we have available censoring information, we have provided *conditional* predictions (i.e. conditional on patient survival up to the censoring time) in panel (b).

We note that not only is c-GPLVM more accurate (on repeated sampling from the posterior, the average MSE from the c-GPLVM is 9.2 whereas for the Weibull models it is 17.1 and 17.9 respectively), but also the c-GPLVM *knows when it does not know*, as its mis-predictions typically have higher uncertainty.

2.5.4.2 Survival-adjusted cancer modelling

Next, we returned to the original data set of all 770 breast cancers and fitted standard GPLVM and c-GPLVM to the entire cohort. Fig 2.14 compares a feature-level fit for standard GPLVM and c-GPLVM for three manually selected genes

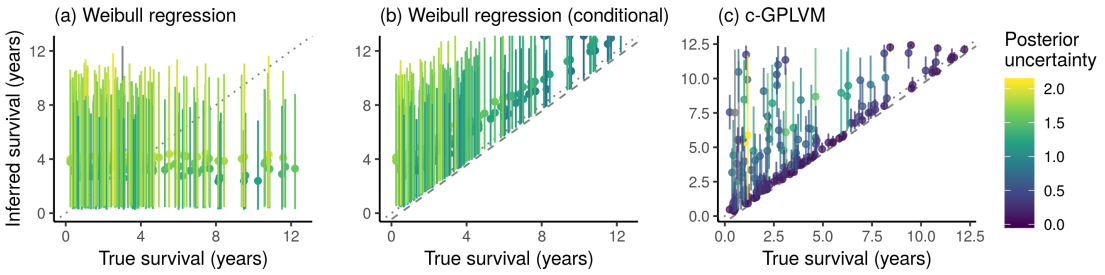


Figure 2.13: Predicted survival times for artificially censored individuals in TCGA data using c-GPLVM (true value on x-axis, inferred posterior mean and (5%, 95%) quantiles on y-axis, colour denotes posterior uncertainty).

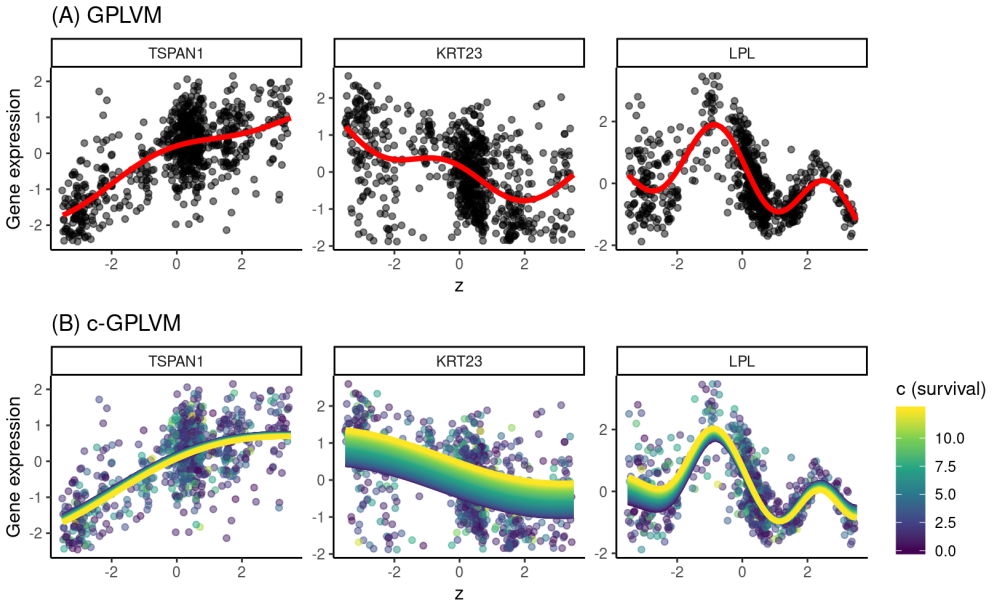


Figure 2.14: The GPLVM (panel A) does not capture the variation driven by the censored covariate (**c**=survival). As shown for three selected genes (*TSPAN1, KRT23, LPL*), GPLVM lets us characterise the effect of the latent **z**, but does not characterise whether some of the *unexplained* variance can be attributed to patient survival. The c-GPLVM fit on the other hand reveals for every gene the type of dependence: for *TSPAN1* there are almost no survival effects, whereas *KRT23* and *LPL* exhibit survival dependence (for *KRT23* it is constant across all **z**, whereas for *LPL* the dependence varies with **z**).

TSPAN1, *KRT23* and *LPL*.[2] We had to use manual selection of genes because even though the c-GPLVM with an INT kernel lets us capture a variety of covariate effects, it does not provide an explicit mechanism to choose genes with different modes of behaviour. Even though we have access to the input-specific lengthscales

[2]To be precise, these genes were selected based on the decomposition that will be introduced in Chapter 3, but for the purpose of this demonstration they can be treated as manually selected genes.

(i.e. the inferred l_z and l_c of the ARD kernel), despite the name *Automatic Relevance Determination* their values do not directly indicate the relevance of inputs, instead the length-scales have been shown to rather measure non-linearity (Paananen et al., 2019).

In the next chapter, we will re-visit this analysis with the aim to improve on the interpretability of the INT kernel. Specifically, we will explore the *identifiable* ADD+INT decomposition to achieve this. As a result, we would hope to avoid manual *ad hoc* selection of genes, and instead use the ADD+INT decomposition to automatically group genes based on the type of dependence (e.g. to find all genes like *TSPAN1* which do not exhibit covariate-dependence).

2.6 Related work

A number of alternative approaches have been previously developed which can be said to lie on the spectrum between GP regression and GPLVMs.

Latent GP regression models (LGPR) (Wang and Neal, 2012; Bodin et al., 2017) extend the input space of a GP regression model with additional latent variables that are used to modulate the covariance function. LGPR models often implicitly assume that the dimensionality of the latent variables is less than the number of regressors $\dim(\mathbf{c}_n) \gg \dim(\mathbf{z}_n)$, in this sense being closer to a GP regression model compared to a GPLVM. Approximate marginalisation of the latent variables allows these models to consider non-stationary, multimodal behaviour, and e.g. has recently[3] been used to construct such GP models for conditional density estimation (Dutordoir et al., 2018). In these hybrid GP models, the inputs $\mathbf{z}$ and $\mathbf{c}$ are usually concatenated, resulting in the INT kernel suited for predictive applications.

Semi-described and semi-supervised learning in GP regression is considered by (Damianou and Lawrence, 2015) when the inputs (or outputs) are partially observed or uncertain. This formulation also leads to a hybrid model with latent variables introduced in place of missing data.

[3] concurrent to our work

Supervised GPLVM encompass a family of related ideas that seek to decompose the joint distribution $p(\mathbf{y}, \mathbf{c}, \mathbf{z})$ in different ways. For example, in (Gadd et al., 2018), the joint distribution is factorised such that the latent variables are conditionally dependent on the covariates $p(\mathbf{z}|\mathbf{c})$. Whilst in the supervised-GPLVM formulation (Gao et al., 2011) and the shared-GPLVM (Shon et al., 2006) both observations and fixed inputs are conditionally dependent on the latent inputs, $p(\mathbf{y}|\mathbf{z})p(\mathbf{c}|\mathbf{z})$. In the discriminative-GPLVM (Urtasun and Darrell, 2007), class label information is used to maximise the discriminatory power between discrete classes in the latent space. This is fundamentally different from c-GPLVM which learns a *covariate-adjusted* $\mathbf{z}$.

2.7 Discussion

In this chapter, we have discussed the notion of confounding due to known covariates, and presented scenarios where learning a covariate-adjusted latent space is desirable. We proposed a construction how to incorporate covariates within GPLVM, and specifically focused on the case where the covariates are censored survival times.

However, for the high-dimensional genomics application that we considered, the approach considered in this section does not provide us sufficient insights into what the model has learnt. Without *ad hoc* post-processing, we are not able to characterise e.g. which genes are associated with survival. We will address enabling such capabilities in the next chapter.

3

Decomposing feature-level variation with Covariate Gaussian Process Latent Variable Models

Contents

3.1 Introduction 57
3.2 Background: Functional ANOVA decompositions . . . 60
3.2.1 Classical ANOVA 60
3.2.2 Functional ANOVA 61
3.2.3 Identifiable functional ANOVA 62
3.3 Decomposable c-GPLVM 64
3.3.1 Motivation 64
3.3.2 Identifiable decomposition with Gaussian Processes . . . 64
3.3.3 Inference 68
3.4 Experiments 70
3.4.1 Synthetic data: Circles and Pinwheel 70
3.4.2 TCGA data 71
3.5 Related work 73
3.6 Discussion 73

3.1 Introduction

Dimensionality reduction is often performed as part of exploratory data analysis, where the goal is to reveal, understand and characterise patterns in the data. This typically involves the interpretation of low-dimensional latent representations. However, these representations usually cannot be interpreted on their own, in

isolation from the features in the data. For example, in order to assign an interpretation to a particular latent coordinate, we might retrospectively inspect which features are most highly predictive of this latent coordinate. Having come up with such an interpretation, we would often like to identify which features are driven by this latent variable, and to what extent. While answering this question is relatively straightforward in linear models like probabilistic PCA, this is much more challenging in a non-linear setting. Our goal here is to develop methodology that would permit such feature-level interpretability in non-linear latent variable models.

In this chapter we specifically focus on a common setting faced in real-life biomedical data analysis where there is additional covariate information, i.e. the scenario discussed in Chapter 2. Note that whilst we previously focused only on the question how to incorporate covariates, now we want to take a step further and quantify the importance of the model inputs, i.e. the effect of covariates and the effect of latent variables, for every one of the variables in the data.

More formally, suppose we have learnt a collection of (potentially non-linear) mappings $f^{(j)} : (\mathbf{z}, \mathbf{c}) \mapsto \mathbf{y}^{(j)}$ where $\mathbf{z} \in \mathbb{R}^D$ are latent variables, $\mathbf{c} \in \mathbb{R}^C$ are covariates, and where j indexes features $j = 1, \dots, P$. We want to characterise the (individual or joint) effect of low-dimensional inputs $\mathbf{z}, \mathbf{c}$ on each of the outputs $\mathbf{y}^{(j)}$. One approach to answering this question is via the notion of *functional decomposition*. To illustrate, suppose both $\mathbf{z}$ and $\mathbf{c}$ are one-dimensional, i.e. $\mathbf{z}, \mathbf{c} \in \mathbb{R}$. In this case, we would like to decompose $f^{(j)}(\mathbf{z}, \mathbf{c})$ into the overall mean, the marginal $\mathbf{z}$ effect, the marginal $\mathbf{c}$ effect, and the interaction between $\mathbf{z}$ and $\mathbf{c}$, i.e.

$$f^{(j)}(\mathbf{z}, \mathbf{c}) = f_0^{(j)} + f_z^{(j)}(\mathbf{z}) + f_c^{(j)}(\mathbf{c}) + f_{zc}^{(j)}(\mathbf{z}, \mathbf{c}) \tag{3.1}$$

As we would like to obtain this functional decomposition for every feature in the data, we refer to it as *feature-level decomposition*. It turns out that without any additional constraints the above decomposition of $f^{(j)}$ is unidentifiable, i.e. there exist infinitely many choices of $(f_0^{(j)}, f_z^{(j)}, f_c^{(j)}, f_{zc}^{(j)})$ that represent the same overall $f^{(j)}$. Thus, to obtain a meaningful decomposition, further constraints need to be imposed.

In this thesis, we focus on a particular approach which is known as the *functional analysis of variance* (functional ANOVA, or fANOVA). The intuition behind functional ANOVA constraints is that the pairwise interaction $f_{zc}^{(j)}(\mathbf{z}, \mathbf{c})$ should only capture the effects that are left *unexplained* by the individual marginal effects $f_z^{(j)}(\mathbf{z})$ and $f_c^{(j)}(\mathbf{c})$. This is arguably highly desirable. For example, if the true data generative mechanism is purely additive in terms of $f_z^{(j)}(\mathbf{z})$ and $f_c^{(j)}(\mathbf{c})$, i.e. it does not contain an interaction, then we would like the inferred interaction term to be non-existent, i.e. $f_{zc}^{(j)} \equiv 0$.

In this chapter, we explore how to embed functional ANOVA constraints within the multi-output Gaussian Process based modelling frameworks (such as the GPLVM or our c-GPLVM), with the goal to aid interpretability on the level of individual features. Such decompositions have traditionally been performed in a regression setting, i.e. where the inputs are fixed and the output is typically univariate, whereas we generalise these decompositions to latent variable models where inputs are latent variables (potentially involving covariates). Specifically, here we focus on how to implement this within the c-GPLVM modelling framework (as introduced in Chapter 2). This has been illustrated in Figure 3.1, where it would be desirable to not only fit a non-linear model, as shown in panel (B), we would like to separate out various sources of variation as shown in subpanels (C).

First, we give an overview of functional ANOVA decompositions and the corresponding "integrate-to-zero" constraints that are imposed for identifiability. Then, we focus on the c-GPLVM framework, and discuss how to embed functional ANOVA decompositions within the c-GPLVM. Specifically, we show how identifiability constraints can elegantly be introduced in the Gaussian Process framework, resulting in a particular structured kernel decomposition. We demonstrate how this construction leads to improved feature-level interpretability.

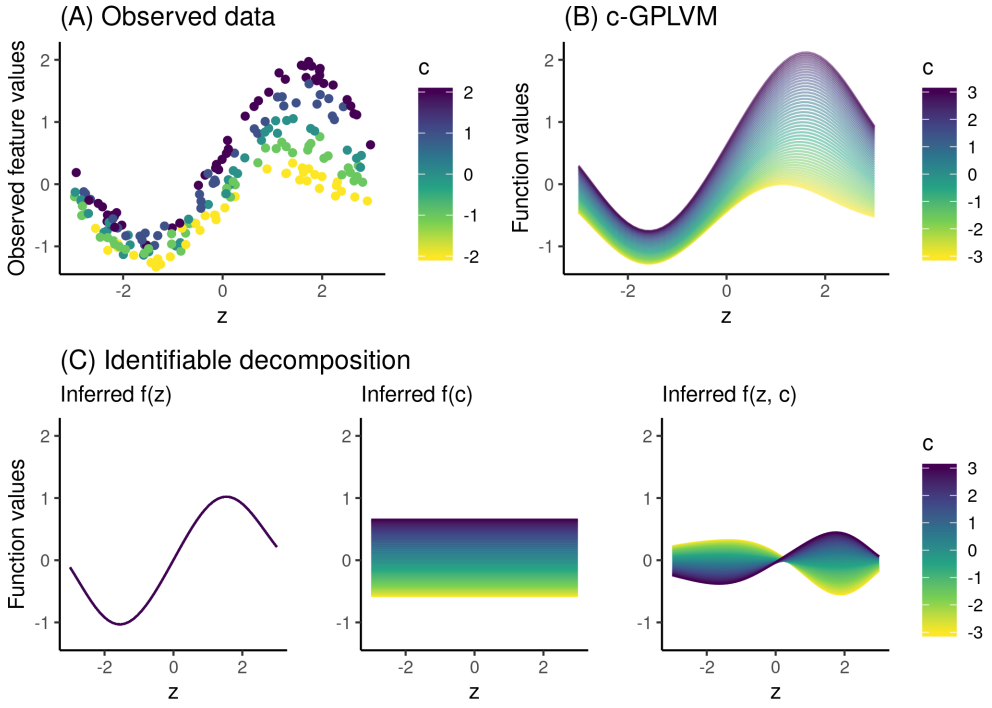


Figure 3.1: Illustration of identifiable decomposition within the c-GPLVM. Given observed data in panel (A) (selected feature values on y-axis, covariate $\mathbf{c}$ in colour coding, latent z on x-axis), the c-GPLVM learns a non-linear mapping $f(\mathbf{z}, \mathbf{c})$ as shown in panel (B). The identifiable feature-level decomposition in panel (C) lets us separate the marginal additive effects of the latent variable $f(\mathbf{z})$ and the covariates $f(\mathbf{c})$ from their non-linear interaction $f(\mathbf{z}, \mathbf{c})$, thus providing added interpretability.

3.2 Background: Functional ANOVA decompositions

First, we briefly discuss the concept of variance decompositions in the traditional ANOVA setting, and then give an overview of functional ANOVA decompositions.

3.2.1 Classical ANOVA

The history of *variance decomposition* goes back to Ronald Fisher's work in the late 1910s and 1920s, e.g. (Fisher, 1918). In the traditional Analysis of Variance (ANOVA) setup, usually the goal is to analyse the differences between groups, e.g. to characterise whether there is association between a grouping variable and the outcome of interest. Often this is carried out in the statistical hypothesis testing framework. The grouping variable can be seen as a categorical covariate (or

potentially multiple covariates) and the outcome is typically a continuous-valued univariate measurement. The quantity of interest in this ANOVA setup is the effect size, that is, what proportion of variance of the outcome variable is explained by the grouping variable(s). This calculation reduces to within-group and between-group variances, but it can be reformulated in terms of a linear model, where the outcome of interest is regressed on a categorical covariate.

While characterising how much variation do individual inputs explain in a linear model is relatively straightforward, the setting that is of interest to us in this thesis is concerned with variance decompositions in *non-linear* models. This is known as *functional* ANOVA.

3.2.2 Functional ANOVA

Suppose we have a function $f : \mathbb{R}^D \to \mathbb{R}$ with inputs $\mathbf{x} \in \mathbb{R}^D$. Functional analysis of variance (fANOVA) are a class of statistical models that decompose a functional response according to the main effects and interactions of various factors (Sobol, 1993; Ramsay and Silvermann, 1997), that is

$$f(x_1, \ldots, x_D) = f_0 + \sum_k f_k(x_k) + \sum_{k,l} f_{kl}(x_k, x_l) + \ldots + f_{1,\ldots,D}(x_1, \ldots, x_D), \quad (3.2)$$

where every $f_{\mathcal{I}}$ belongs to some family of functions $\mathcal{F}_{\mathcal{I}}$ for any index set $\mathcal{I} \subset \{1, ..., D\}$.

For example, smoothing spline ANOVA models (Gu, 2013) represent $f_{\mathcal{I}}$ via smoothing splines, i.e. as a linear combinations of some underlying basis functions, and the coefficients on these basis functions are chosen to minimise a loss function balancing goodness of fit with a measure of smoothness of the fitted functions. More recent treatments within a Bayesian nonparametric framework use Gaussian Processes as a prior over the function space, focusing on the setting where covariates are categorical (Kaufman and Sain, 2010).

Quantifying the importance of a subset of inputs is closely related to *global sensitivity analysis*: In fact, the Sobol sensitivity indices characterise the fraction of variance explained by a particular subset of inputs (Sobol, 2001).

3.2.3 Identifiable functional ANOVA

Here we do not make specific assumptions how the functions $f_{\mathcal{I}}$ in equation (3.2) are represented or parameterised, but later we will investigate two special cases:

- First, in this chapter we will focus on the case where the functions are samples from a Gaussian Process. In this case, the families $F_{\mathcal{I}}$ are implicitly defined via a kernel, i.e. $F_{\mathcal{I}} := \{f_{\mathcal{I}} : f_{\mathcal{I}} \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))\}$.
- Second, in chapter 4 we will investigate the case where the functions are parameterised by neural networks. In this case, the families $F_{\mathcal{I}}$ are defined in terms of the neural network architecture and its parameters θ, i.e. $F_{\mathcal{I}} := \{f_{\mathcal{I}}^{\theta} : \theta \in \mathbb{R}^{\dim(\theta)}\}$.

But first, we will formulate two general results regarding the uniqueness of the functional decomposition.

Proposition 1. *Let functions $f_i \in \mathcal{F}_i, f_{ij} \in \mathcal{F}_{ij}, \ldots, f_{1,\ldots,D} \in \mathcal{F}_{1,\ldots,D}$ be such that for any two sets of indices $\mathcal{I} \subsetneq \mathcal{J}$ the functional subspace $\mathcal{F}_{\mathcal{I}}$ is strictly contained within $\mathcal{F}_{\mathcal{J}}$, i.e. $\mathcal{F}_{\mathcal{I}} \subsetneq \mathcal{F}_{\mathcal{J}}$. Then the above decomposition (3.2) is unidentifiable.*

Proof. Consider the decomposition (3.2). Let $\mathcal{I}$ and $\mathcal{J}$ be two sets of indices such that $\mathcal{I} \subsetneq \mathcal{J}$. Now we are going to show that components $f_{\mathcal{I}}$ and $f_{\mathcal{J}}$ in (3.2) are not uniquely defined. Consider $g_{\mathcal{I}} \in \mathcal{F}_{\mathcal{I}} \subsetneq \mathcal{F}_{\mathcal{J}}$. Indeed, consider an alternative decomposition where the sum $f_{\mathcal{I}} + f_{\mathcal{J}}$ has been replaced by $(f_{\mathcal{I}} + g_{\mathcal{I}}) + (f_{\mathcal{J}} - g_{\mathcal{I}})$. Clearly $(f_{\mathcal{I}} + g_{\mathcal{I}}) \in \mathcal{F}_{\mathcal{I}}$ and $(f_{\mathcal{J}} - g_{\mathcal{I}}) \in \mathcal{F}_{\mathcal{J}}$, so the decomposition is not unique. □

As a result, without any additional constraints, the decomposition (3.2) is not meaningful. This unidentifiability does not affect the model's predictive performance, however, the interpretation regarding how important various components are is arbitrary, since higher-order interactions can *absorb* variability that could be explained by main effects or low-order interactions.

To turn this into an *identifiable* learning problem, we need to introduce functional constraints. Specifically, the functional ANOVA framework introduces constraints which enforce all marginal effects of $f_{\mathcal{I}}(\cdot)$ to be zero, i.e. for any $\mathcal{U} \subset \mathcal{I}$ the functions

$f_{\mathcal{I}}(\cdot)$ should be such that $\int f_{\mathcal{I}}(\mathbf{x}_{\mathcal{I}})d\mathbf{x}_{\mathcal{U}} = 0$. We refer to these as mean-zero integral constraints. This is formally summarised as follows.

Proposition 2. *Consider the decomposition in formula (3.2). If all its components $f_{\mathcal{I}}$ have zero marginals, i.e. for every index set $\mathcal{I} = \{i_1, \ldots, i_s\}$ the following holds*

$$\int f_{\mathcal{I}}(\mathbf{x}_{\mathcal{I}})dx_i = 0 \quad \text{for all } \; i \in \mathcal{I} \tag{3.3}$$

then for any two sets of indices $\mathcal{I} \neq \mathcal{J}$ the functional subspaces $\mathcal{F}_{\mathcal{I}}$ and $\mathcal{F}_{\mathcal{J}}$ do not overlap any more (apart from the constant zero function). Furthermore, these functional subspaces are orthogonal in L_2.

Note that the above integral constraints (3.3) imply that for any subset $\mathcal{U} \subset \mathcal{I}$ we also have $\int f_{\mathcal{I}}(\mathbf{x}_{\mathcal{I}})d\mathbf{x}_{\mathcal{U}} = 0$. Both of the properties (no overlap and orthogonality) are a direct consequence of the integral constraints. The general proof follows derivations as in previous works, e.g. (Sobol, 1993). Note that the former (no overlap) is sufficient for identifiability, but the latter (orthogonality) leads to an easily interpretable variance decomposition. Denoting the variance attributed to component $f_{\mathcal{I}}$ by $\mathrm{Var}(f_{\mathcal{I}}) := \int f_{\mathcal{I}}(\mathbf{x}_{\mathcal{I}})^2 d\mathbf{x}_{\mathcal{I}}$, where the integral is taken over some hypercube $[a, b]^{|\mathcal{I}|}$, we can indeed decompose the overall variance as follows

$$\mathrm{Var}(f) = \mathrm{Var}(f_0) + \sum_k \mathrm{Var}(f_k) + \sum_{k,l} \mathrm{Var}(f_{kl}) + \ldots + \mathrm{Var}(f_{1,\ldots,D}).$$

Sometimes it is desirable to characterise the *fraction* of variance explained by a specific component $f_{\mathcal{I}}$. These relative variances $\mathrm{Var}(f_{\mathcal{I}})/\mathrm{Var}(f)$ are also known as Sobol indices (Sobol, 2001).

Two-dimensional special case: For the special case of two-dimensional inputs, i.e. when the decomposition consists of f_0, f_1, f_2, f_{12}, we have the following integral constraints:

- $\int f_1(x_1)dx_1 = 0$
- $\int f_2(x_2)dx_2 = 0$
- $\int f_{12}(x_1, x_2)dx_1 = 0$ for all x_2
- $\int f_{12}(x_1, x_2)dx_2 = 0$ for all x_1.

where the integrals are taken over some hypercube $x_1 \in [a_1, b_1]$, $x_2 \in [a_2, b_2]$ which reflects the range of our observations. Note that such integration can be seen as averaging over a uniform distribution of inputs. In scenarios where this is undesirable (e.g. in the presence of strong correlation structure among the inputs), integration with respect to a different measure could potentially be considered (Hooker, 2007).

The functional ANOVA framework that we have introduced above has traditionally been used in a regression setting, whereas our interest in this thesis is to utilise this framework as part of latent variable models. In the next section, we will explore how functional ANOVA can be embedded within the c-GPLVM, and later, in Chapter 4, we will discuss how it can be embedded within the (C)VAE framework and how to ensure that the integral constraints would be fulfilled.

3.3 Decomposable c-GPLVM

3.3.1 Motivation

Here, we return to the c-GPLVM setup introduced in the previous chapter. That is, we have inputs $\mathbf{z} \in \mathbb{R}^D$ and $\mathbf{c} \in \mathbb{R}^C$ which are mapped to the feature space via GP mappings $f^{(j)}(\mathbf{z}, \mathbf{c})$. Previously, we focused on the setting where we used a GP prior with an interaction (INT) kernel, i.e. a squared exponential ARD kernel on the concatenation of inputs $[\mathbf{z}, \mathbf{c}]$. While it could be considered a good default choice for predictive applications, this kernel does not let us characterise the effects of individual inputs and separate these from interactions between two or more inputs. In order to obtain a feature-level decomposition, we would like to introduce a kernel decomposition. However, for this to be meaningful, we have to ensure identifiability via functional constraints. Next, we discuss in detail how to integrate functional ANOVA decomposition and c-GPLVM.

3.3.2 Identifiable decomposition with Gaussian Processes

We denote the inputs collectively $\mathbf{x} := [\mathbf{z}, \mathbf{c}] \in \mathbb{R}^D$. The first (naive) attempt to learn a functional decomposition within a multi-output GP model could be as

follows

$$f^{(j)}(\mathbf{x}) = f_0^{(j)} + \sum_k f_k^{(j)}(x_k) + \sum_{k,l} f_{kl}^{(j)}(x_k, x_l) + \ldots + f_{1,\ldots,D}^{(j)}(x_1, \ldots, x_D) \tag{3.4}$$

where all functions $f_{\mathcal{I}}^{(j)}$ have an independent GP prior $f_{\mathcal{I}}^{(j)} \sim \mathcal{GP}(0, k(\mathbf{x}_{\mathcal{I}}, \mathbf{x}'_{\mathcal{I}}))$ where $k(\cdot)$ is the squared exponential ARD kernel. However, such a decomposition is unidentifiable.

Below we drop index j to reduce notational clutter.

Proposition 3. *If all functions $f_{\mathcal{I}}$ in equation (3.4) are drawn from GP priors $f_{\mathcal{I}} \sim \mathcal{GP}(0, k(\mathbf{x}_{\mathcal{I}}, \mathbf{x}'_{\mathcal{I}}))$ where $k(\cdot)$ is the squared exponential ARD kernel, then the decomposition is not unique.*

Proof. Consider $\mathcal{U} \subsetneq \mathcal{I}$ and the respective functional subspaces $\mathcal{F}_{\mathcal{U}}$ and $\mathcal{F}_{\mathcal{I}}$, where $\mathcal{F}_{\mathcal{I}} = \{f_{\mathcal{I}} : f_{\mathcal{I}} \sim \mathcal{GP}(0, k(\cdot))\}$. Note that every $f_{\mathcal{U}}(\mathbf{x}_{\mathcal{U}})$ can be seen as a function defined on the extended input space $(\mathbf{x}_{\mathcal{U}}, \mathbf{x}_{\mathcal{I}\setminus\mathcal{U}})$ that is constant in the rest of the coordinates. But then every $f_{\mathcal{U}}$ has a positive probability under the GP prior $f_{\mathcal{I}} \sim \mathcal{GP}(0, k(\mathbf{x}_{\mathcal{I}}, \mathbf{x}'_{\mathcal{I}}))$. Proposition 1 directly applies, because $\mathcal{F}_{\mathcal{U}} \subsetneq \mathcal{F}_{\mathcal{I}}$. □

For uniqueness, we would like to enforce the integral constraints as outlined in Proposition 2. Note that the subtle difference is that here we operate in a Bayesian setting, i.e. we have distributions over functions. Ideally, for a coherent setup, we would like to specify a prior distribution that satisfies these integral constraints (i.e. functions that do not fulfill the constraints should have zero probability mass under the prior), then the same would hold for the posteriors. It turns out that these constraints can be elegantly introduced via *conditioning*.

Mean-zero GP for one-dimensional inputs: First we discuss the special case where we have one-dimensional inputs $x \in \mathbb{R}$ with the corresponding GP prior $f \sim \mathcal{GP}(0, k(x, x'))$. It turns out that we can construct a GP prior for f conditional on $\int_a^b f(t)dt = 0$ (Durrande et al., 2012; Durrande et al., 2013). The joint distribution of $f(x)$ and its integral $\int_a^b f(t)dt$ over some interval $[a, b]$ is as follows

$$\begin{pmatrix} f(x) \\ \int_a^b f(t)dt \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} k(x,x) & \int_a^b k(x,t)dt \\ \int_a^b k(t,x)dt & \int_a^b \int_a^b k(t,s)dtds \end{pmatrix}\right]$$

which lets us express the conditional distribution of $f(x)$ conditional on $\int_a^b f(t)dt = 0$, i.e. conditional on f being mean-zero. As a result,

$$f \,\Big|\, \left(\int_a^b f(t)dt = 0\right) \sim \mathcal{GP}(0, \tilde{k}(\cdot)),$$

which is still a GP, but with a modified kernel $\tilde{k}$

$$\tilde{k}(x, x') := k(x, x') - \frac{\int_a^b k(x, t)dt \int_a^b k(t, x')dt}{\int_a^b \int_a^b k(t, s)dtds} .$$

Note that the above holds for any choice of kernel, but for the special case when $k(x, x')$ is the squared exponential kernel, i.e. when

$$k(x, x') = \sigma^2 \exp\left(-\frac{1}{2}\frac{(x - x')^2}{l^2}\right)$$

these integrals have closed forms,

$$\int_a^b k(x, t)dt = \frac{1}{2}\sqrt{2\pi}l\sigma^2 \left(\text{erf}\left(\frac{b - x}{\sqrt{2}l}\right) - \text{erf}\left(\frac{a - x}{\sqrt{2}l}\right)\right)$$

and

$$\int_a^b \int_a^b k(t, s)dtds = \sqrt{2\pi}l\sigma^2 \left((a - b)\text{erf}\left(\frac{a - b}{\sqrt{2}l}\right) + \frac{\sqrt{2}}{\sqrt{\pi}}l\left(\exp\left(-\frac{(b - a)^2}{2l^2}\right) - 1\right)\right).$$

Mean-zero GP for multivariate inputs: Now, being able to evaluate $\tilde{k}$ for a one-dimensional input, we will generalise these mean-zero constraints to multivariate inputs $\mathbf{x} \in \mathbb{R}^D$.

First, we note that given some multivariate input $\mathbf{x}_\mathcal{I}$, the squared exponential ARD kernel can be written as a product of kernels

$$k_\mathcal{I}(\mathbf{x}_\mathcal{I}, \mathbf{x}'_\mathcal{I}) = \sigma_\mathcal{I}^2 \exp\left(-\frac{1}{2}\sum_{d\in\mathcal{I}} \frac{(x_d - x'_d)^2}{l_d^2}\right) = \sigma_\mathcal{I}^2 \prod_{d\in\mathcal{I}} k_0^d(x_d, x'_d),$$

where every $k_0^d(\cdot, \cdot)$ is a squared exponential kernel defined on one-dimensional inputs with length-scale l_d and unit kernel variance.

In order to enforce the mean-zero integral constraints as in Proposition 2, we replace the kernels k_0^d in the above kernel decomposition with their respective mean-zero counterparts $\tilde{k}_0^d$, i.e.

$$\tilde{k}_\mathcal{I}(\mathbf{x}, \mathbf{x}') := \sigma_\mathcal{I}^2 \prod_{d\in\mathcal{I}} \tilde{k}_0^d(x_d, x'_d), n$$

where $\tilde{k}_0$ denotes the mean-zero kernel as above with kernel variance set to 1 and $\sigma^2_{1,\dots,D}$ denotes the kernel variance parameter (Durrande et al., 2013).

Now we can specify the full kernel decomposition that includes all terms with increasing interaction up to order D,

$$\sigma_0^2 + \sum_{\mathcal{I} \subset \{1,\dots,D\}} \tilde{k}_{\mathcal{I}}(\mathbf{x}_{\mathcal{I}}, \mathbf{x}'_{\mathcal{I}}).$$

Application to the c-GPLVM: Now we return to the previously used c-GPLVM notation, where we have fixed inputs $\mathbf{c}$ and latent variables $\mathbf{z}$. Note that our goal is to achieve identifiability of the *functional decomposition* rather than the entire c-GPLVM (i.e. we assume fixed inputs $\mathbf{z}, \mathbf{c}$)

To illustrate on a specific example, suppose both $\mathbf{z}$ and $\mathbf{c}$ are one-dimensional, i.e. $\mathbf{z}, \mathbf{c} \in \mathbb{R}$. In this case, we would like to decompose $f^{(j)}(\mathbf{z}, \mathbf{c})$ as follows

$$f^{(j)}(z, c) = f_0^{(j)} + f_z^{(j)}(z) + f_c^{(j)}(c) + f_{zc}^{(j)}(z, c) \tag{3.5}$$

where we would interpret $f_0^{(j)}$ as the overall mean for feature j, $f_z^{(j)}$ as the marginal $\mathbf{z}$ effect, $f_c^{(j)}$ as the marginal $\mathbf{c}$ effect, and $f_{zc}^{(j)}$ as the interaction between $\mathbf{z}$ and $\mathbf{c}$. Note that this interpretation is only appropriate when we use the mean-zero GP priors, otherwise the interaction term $f_{zc}^{(j)}$ could absorb the marginal effects. This unidentifiability is illustrated in Figure 3.2 for the case where both $\mathbf{z}, \mathbf{c} \in \mathbb{R}$, where on synthetic data we have shown an alternate realisation of the inferred decomposition (using the same data as in Figure 3.1). This alternate decomposition (in bottom row) would suggest that there is no additive $\mathbf{c}$ effect and that almost all of the variation is explained by the interaction effect, which was not the generative mechanism used to synthesise the data (shown in top row).

The mean-zero GP priors, i.e. the GP priors specified in terms of the $\tilde{k}$ kernel, can be seen as a joint prior over the four functional subspaces (f_0, f_z, f_c, f_{zc}) with the following two properties: First, their supports are non-overlapping. Second, the respective functional subspaces are orthogonal in L^2. The former is sufficient for a unique decomposition, but the latter makes interpretation easier, because

$$\mathrm{Var}(f) = \mathrm{Var}(f_0) + \mathrm{Var}(f_z) + \mathrm{Var}(f_x) + \mathrm{Var}(f_{zx})$$

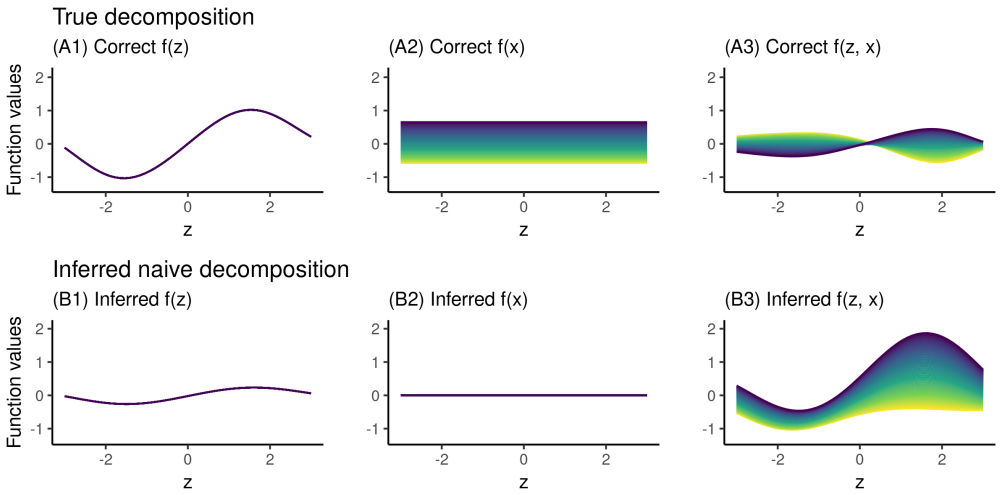


Figure 3.2: Example of how decomposition (3.4) without any further constraints is unidentifiable. The true generative mechanism (top row) exhibits strong additive $f_z(\mathbf{z})$ and $f_c(\mathbf{c})$ effects, whereas a realisation of the naive unidentifiable decomposition (bottom row) implies as if almost all of the variation was due to interactions between $\mathbf{z}$ and $\mathbf{c}$.

due to orthogonality. The corresponding kernel decomposition on the joint (z, c) space as follows

$$k((z,c),(z',c')) := \sigma_0^2 + \sigma_z^2\tilde{k}_0(z,z') + \sigma_c^2\tilde{k}_0(c,c') + \sigma_{zc}^2\tilde{k}_0(z,z')\tilde{k}_0(c,c').$$

We refer to the functional decomposition (3.5) as "ADD+INT" decomposition, emphasising that it lets us separate additive marginal effects from interactions.

Finally, we make use of Bayesian shrinkage priors on the kernel variances σ_z^2, σ_c^2, σ_{cz}^2 to encourage unnecessary components to be be shrunk to zero. For this purpose, we specify $\sigma_z^2, \sigma_c^2, \sigma_{cz}^2 \sim \Gamma(1,1)$.

3.3.3 Inference

As our main interest is developing the decomposable c-GPLVM methodology for real-life genomics data which is typically continuous-valued, here we focus on the Gaussian likelihood case only. Inference for the decomposable c-GPLVM follows the derivations outlined in Section 2.3.2, now using the "ADD+INT" kernel decomposition. For notational simplicity, here we outline inference details for the special case $\mathbf{z}, \mathbf{c} \in \mathbb{R}$, but all results are straightforward to extend for higher dimensional $\mathbf{z} \in \mathbb{R}^D, \mathbf{c} \in \mathbb{R}^C$.

Standard c-GPLVM inference (without inducing points): First, we consider the scenario when our sample size N is moderate and we do not require an inducing point approximation. For this case, we can directly adopt the bound from equation (2.1) for the decomposable kernel decomposition, i.e.

$$\mathcal{L} = \sum_{j=1}^{P} \mathbb{E}_{q(\mathbf{Z})} \mathcal{N}(\mathbf{y}^{(j)}|\mathbf{0}, \mathbf{K}_{zz'}^{(j)} + \mathbf{K}_{cc'}^{(j)} + \mathbf{K}_{(zc)(z'c')}^{(j)} + \sigma_j^2 \mathbf{I}) - \sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{z}_i)||p(\mathbf{z}_i))$$

where for every feature $j = 1, \ldots, P$ we learn the mean-zero kernel decomposition $\mathbf{K}_{zz'}^{(j)} + \mathbf{K}_{cc'}^{(j)} + \mathbf{K}_{(zc)(z'c')}^{(j)}$.

Inference for sparse c-GPLVM (with inducing points): Now we consider the case where the computational cost of inverting the $N \times N$ kernel matrix is prohibitive, thus we rely on the inducing point framework. In principle, we could approximate every one of the functions $f_z^{(j)}, f_c^{(j)}, f_{zc}^{(j)}$ with its own set of inducing inputs, but here we decide to view all these functions as taking values in the $(\mathbf{z}, \mathbf{c})$ space, and we assume *shared* inducing inputs, i.e. we introduce M inducing inputs $(\mathbf{z}_i^u, \mathbf{c}_i^u)$ for $i = 1, \ldots, M$, denoted collectively by $(\mathbf{Z}_u, \mathbf{C}_u) \in \mathbb{R}^{M \times (D+C)}$. We denote the respective function values at the inducing inputs $\mathbf{u}_z^{(j)} = f_z^{(j)}(\mathbf{Z}_u)$, $\mathbf{u}_c^{(j)} = f_c^{(j)}(\mathbf{C}_u)$ and $\mathbf{u}_{cz}^{(j)} = f_{cz}^{(j)}(\mathbf{Z}_u, \mathbf{C}_u)$. We will treat the function values across data dimensions independently, i.e. $q(\mathbf{U}_c, \mathbf{U}_z, \mathbf{U}_{zc}) = \prod_{j=1}^{P} q(\mathbf{u}_z^{(j)}, \mathbf{u}_c^{(j)}, \mathbf{u}_{cz}^{(j)})$, but the remaining question is how to construct the variational approximation $q(\mathbf{u}_z^{(j)}, \mathbf{u}_c^{(j)}, \mathbf{u}_{cz}^{(j)})$. For example, one could assume the following form $q(\mathbf{u}_z^{(j)})q(\mathbf{u}_c^{(j)})q(\mathbf{u}_{cz}^{(j)})$. Such an approach which factorises over additive functional components was e.g. used by Adam et al. (2016), but this is not the approach we take. Instead, as we are primarily interested in implementing the decomposable c-GPLVM assuming the Gaussian likelihood, we decide to follow the approach where we collapse all variables $\mathbf{u}_z^{(j)}, \mathbf{u}_c^{(j)}, \mathbf{u}_{cz}^{(j)}$ jointly without making any specific assumptions about the form of the variational approximation. Denoting the aggregate kernel matrix

$$\tilde{\mathbf{K}}^{(j)} := \mathbf{K}_{zz'}^{(j)} + \mathbf{K}_{cc'}^{(j)} + \mathbf{K}_{(zc)(z'c')}^{(j)}$$

analogously to equation (1.6) we obtain

$$\begin{aligned}\mathcal{L} = \sum_{j=1}^{P} \mathbb{E}_{q(\mathbf{Z})} \log \mathcal{N}(\mathbf{y}^{(j)} \,|\, \mathbf{0}, \tilde{\mathbf{K}}_{fu}^{(j)} \tilde{\mathbf{K}}_{uu}^{(j)^{-1}} \tilde{\mathbf{K}}_{uf}^{(j)} + \sigma_j^2 \mathbf{I}) \\ -\frac{1}{2\sigma_j^2} \mathbb{E}_{q(\mathbf{Z})} \mathrm{tr}\left(\tilde{\mathbf{K}}_{ff}^{(j)} - \tilde{\mathbf{K}}_{fu}^{(j)} \tilde{\mathbf{K}}_{uu}^{(j)^{-1}} \tilde{\mathbf{K}}_{uf}^{(j)}\right) - \sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{z}_i) || p(\mathbf{z}_i))\end{aligned} \tag{3.6}$$

which implicitly captures correlations between the inducing variables $\mathbf{u}_z^{(j)}, \mathbf{u}_c^{(j)}, \mathbf{u}_{cz}^{(j)}$.

3.4 Experiments

3.4.1 Synthetic data: Circles and Pinwheel

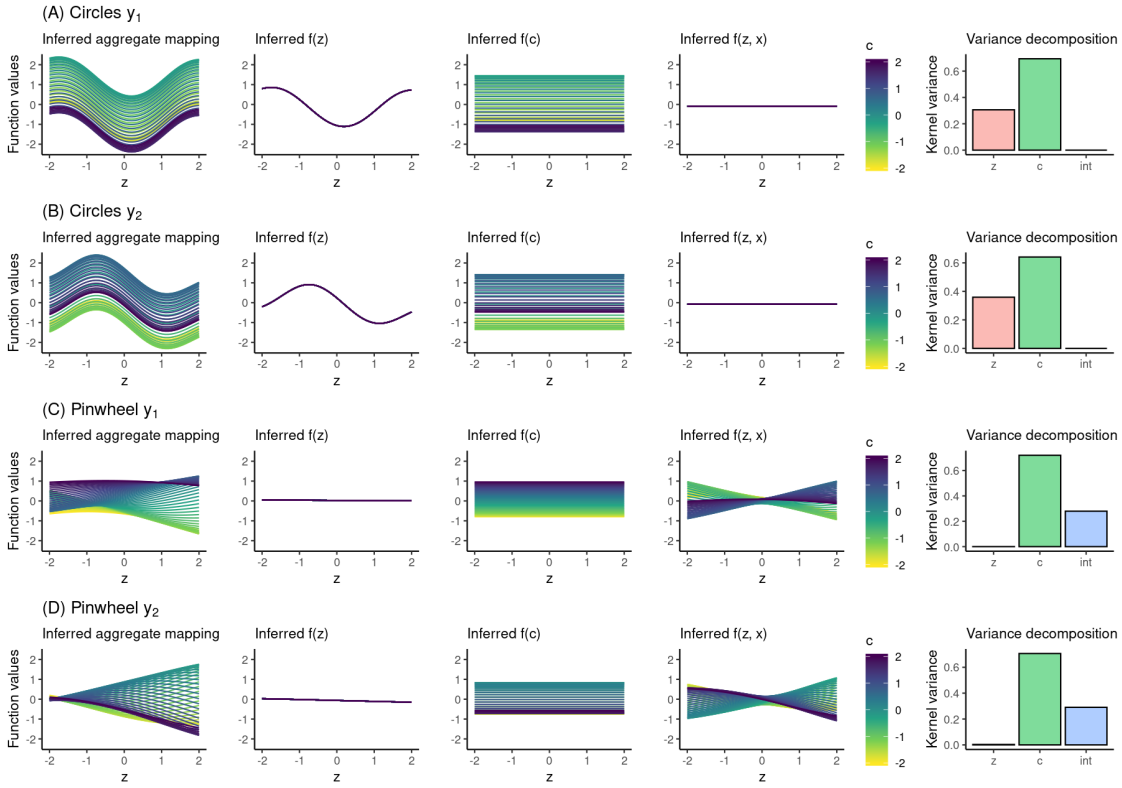


Figure 3.3: Functional decompositions with c-GPLVM on the two-dimensional (A, B) "circles" and (C, D) "pinwheel" toy examples as introduced in Section 2.5.1. For every feature ($\mathbf{y}^{(1)}$ or $\mathbf{y}^{(2)}$ for either "circles" or "pinwheel") we have shown the inferred aggregate GP mean and its functional decomposition into $f_z^{(j)}(\mathbf{z}) + f_c^{(j)}(\mathbf{c}) + f_{zc}^{(j)}(\mathbf{z}, \mathbf{c})$, all functions plotted over a grid of $\mathbf{c}$ values (shown in colour). The last column shows the normalised kernel variance parameters $\sigma_z^2, \sigma_c^2, \sigma_{zc}^2$, which constitute the inferred variance decomposition.

We now revisit synthetic "pinwheel" and "circles" examples that we explored in Section 2.5.1. Previously, in Figure 2.9, we explored the inferred GP mappings,

i.e. the posterior over $f^{(j)}(\mathbf{z}, \mathbf{c})$. While in these particular toy examples there are only two features that can be carefully visually inspected, this strategy will not scale to high-dimensional data sets where it is not feasible to manually inspect the mappings for all features any more. Instead, it would be desirable to explicitly characterise the type of dependence on inputs.

To demonstrate this capability within the decomposable c-GPLVM, in Figure 3.3 we have shown the inferred functional decomposition on the "Pinwheel" and "Circles" data sets. Note that the aggregate mappings (in the leftmost panels) match the ones we saw previously in Figure 2.9, but now we can additionally separate out and characterise the additive marginal effects, as shown in panels corresponding to f_z, f_c, and f_{zc}. The rightmost panels quantifies the contribution of each of the functions in terms of the inferred variance decomposition (the bar plot shows normalised kernel variances). For example, we see that the generative model for the Circles data does not involve interactions between $\mathbf{z}$ and $\mathbf{c}$, whereas the presence of non-linear interactions is crucial for generating a pinwheel-like structure.

3.4.2 TCGA data

We now revisit the TCGA breast cancer gene expression data that was initially analysed in Section 2.5.4. Now, having developed the framework for functional decompositions, we can explicitly characterise the presence (or absence) of the latent variable $\mathbf{z}$ effects, survival $\mathbf{c}$ effects, and the interactions between the two. For direct comparison with the previous Figure 2.14, we have now illustrated the functional and variance decompositions for the same three selected genes in Figure 3.4.

The identifiable functional decomposition and the associated variance decomposition let us identify three sets of genes: those without any covariate effect (e.g. *TSPAN1*), genes with an additive covariate effect (e.g. *KRT23*), and genes that exhibit a complex non-linear interaction (e.g. *LPL*).

First, *TSPAN1* is a gene that codes for a member of the protein family, Tetraspanins, also known as the transmembrane 4 superfamily. These are small transmembrane

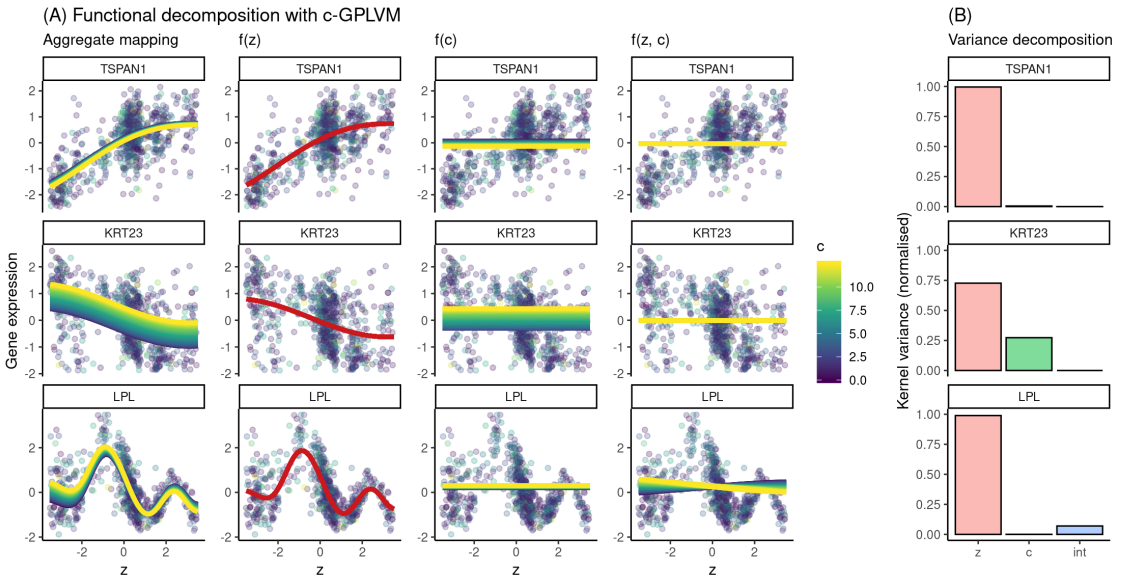


Figure 3.4: For three selected genes (*TSPAN1, KRT23, LPL*), (A) illustration of the c-GPLVM functional decompositions and (B) the respective variance decompositions. For *TSPAN1*, almost all of the variation is attributed to the latent variable **z**, whereas for *KRT23* we have detected the presence of a strong additive **c** contribution (lower *KRT23* expression corresponding to lower survival times) and for *LPL* we have detected a complex interaction between **z** and **c** in addition to the marginal additive non-linear effect of **z**.

glycoproteins which were first described in studies of tumour associated proteins and has been reported to regulate cancer progression in many human cancers (Munkley et al., 2017). In line with this, c-GPLVM identifies *TSPAN1* as a gene whose expression increases along the latent coordinate and not with survival suggesting that the identified latent dimension could correspond to disease progression. In contrast, Keratin-23 (*KRT23*) decreases with z but is also additively modulated by the survival covariate with patients who survive longer seemingly retaining a higher expression of *KRT23*. Lipoprotein lipase (LPL) plays a role in breaking down fat in the form of triglycerides, which are carried from various organs to the blood by molecules called lipoproteins. This gene is identified as having interaction effects and we see that there is a range of latent input values $-2 < z < 0$ where low *LPL* expression is associated with longer survival and high LPL expression is associated with reduced lifespan. This suggests that during certain periods of breast cancer development, high levels of *LPL* activity could enable aggressive disease progression possibly through acting to provide a supply of fatty acids to

fuel tumour growth (Kuemmerle et al., 2011). Our analysis here is only illustrative but serves to demonstrate that the kernel decomposition within c-GPLVM allowed us to "discover" three distinct gene behaviours. This could not have been readily achieved without further post-processing analysis or modification of other existing GPLVM extensions and implementations.

3.5 Related work

As discussed in Section 3.2, the use of functional decompositions in non-linear models is not new, and has been explored in a variety of modelling frameworks, including generalized additive models (Wood et al., 2017), splines (Gu, 2013), tree-based models (Lengerich et al., 2019), but also Gaussian Processes (Kaufman and Sain, 2010; Durrande et al., 2013). To our knowledge, all such pre-existing applications consider decompositions as part of regression models.

3.6 Discussion

By making use of GP mappings that are specified on the extended joint space of covariates and latent variables, we can model complex non-linear dependencies, while maintaining interpretable and decomposable mappings. The c-GPLVM is applicable for a wide range of applications where there is known structure accounting for variance attributable to this structure. This encompasses scenarios when we are interested in explicitly exploring the interactions between this covariate information and other features, as well as those where such covariates are "nuisance" variables and we would like to adjust for confounding factors (e.g. batch effects or the presence of different ancestral populations in population genetics). A natural extension of our work is to consider a deep multi-output Gaussian Process formulation (Damianou and Lawrence, 2013) in which multiple output dimensions can be coupled via shared Gaussian process mappings. Very recently, a hybrid deep GP model was explored in (Salimbeni et al., 2019), but there the authors focused on predictive applications as opposed to decomposability. Constructing interpretable decompositions in the deep

c-GPLVM framework would introduce many design choices with regards to how the shared structure could be embedded within our structured kernel decomposition framework. We will leave this as a future research direction.

A practical limitation of our c-GPLVM framework is related to its compute time and memory requirements for high-dimensional data, where we infer a separate kernel decomposition per feature. Even though the scalability of c-GPLVM w.r.t. sample size N can be handled via the variational inducing point framework, for high-dimensional data applications it would be desirable to avoid the linear scalability w.r.t. the number of features P. This could be achieved via a sharing mechanism within the multi-output GP framework, such as the linear model of coregionalisation (Alvarez et al., 2012), where the GPs would not be treated independently. We leave this for future work, and instead consider a different class of non-linear models in the forthcoming chapters.

4

Neural Decomposition: Functional ANOVA with Variational Autoencoders

Contents

4.1 Introduction 75
4.2 Identifiable Neural Decomposition 77
4.2.1 Decomposing Neural Network Regression 78
4.2.2 Inference under integral constraints 79
4.2.3 Decomposing CVAEs 82
4.2.4 Sparse Neural Decomposition 83
4.3 Experiments 84
4.3.1 Comparison with Decomposable c-GPLVM 85
4.3.2 Synthetic data 87
4.3.3 Decomposing CelebA 90
4.3.3.1 Pixel-level decomposition 90
4.3.3.2 Improved generalisation 91
4.3.4 Gene expression 92
4.4 Related work 96
4.5 Discussion 97

4.1 Introduction

Here, we focus on neural network models for dimensionality reduction, specifically on the Variational Autoencoder (VAE) framework (Kingma and Welling, 2014). Recently there has been particular interest in applications of VAEs since their

neural network underpinnings offers considerable modelling flexibility, while modern programming frameworks (e.g. TensorFlow and PyTorch) and computational techniques (e.g. stochastic variational inference) provide immense scalability to large data sets, including those in genomics (Lopez et al., 2018; Eraslan et al., 2019). The Conditional VAE (CVAE) (Sohn et al., 2015) extends this model to allow incorporating side information as additional fixed inputs.

The ability of (C)VAEs to extract latent, low-dimensional representations from high-dimensional inputs is a strength for predictive tasks. For example (C)VAEs have been particularly popular as generative models for images. However, when the objective is to understand the contributions of different inputs towards explaining variation in the data, the standard black box decoder is insufficient to permit this. This is particularly important for tabular data problems where individual features might correspond to actual physical quantities (e.g. expression of a gene, a physiological measurement, position of a joint on a moving body) and we are interested in how the variability of the multivariate output response is driven by specific changes in the inputs (where inputs may be latent variables and/or fixed covariates). To introduce such feature-level interpretability, we turn to the notion of functional analysis of variance that we introduced in Section 3.2.

While from the application perspective, the problem setup in this chapter is very similar to the one discussed previously in Chapter 3, in here we will focus on a different modelling framework. That is, instead of the Gaussian Process based c-GPLVM, we investigate feature-level interpretability in the Variational Autoencoder framework. In this chapter, we propose the notion of *neural decomposition* - the integration of functional ANOVA and deep neural networks for dimensionality reduction and variance decomposition. Specifically, we develop and explore a class of (C)VAE models where the *decoding* network is specifically implemented to explain feature-level variability in terms of additive and interaction effects between the latent variables and any additional covariate information.

Thus, neural decomposition (ND) can be seen from two perspectives:

- From the classical statistics perspective, it is a scalable *neural* adaptation of the functional ANOVA decomposition, where additional latent variables $\mathbf{z}$ have been introduced to capture low-dimensional structure within high-dimensional data and nonlinearities implemented via neural networks.
- From the deep learning perspective, it is an extension of the (C)VAE framework, where the decoder has a decomposable additive structure to aid interpretability on the level of individual features.

Crucially, in order for this class of models to be useful, identifiability must be enforced to ensure that the expressive power of neural networks that capture higher order interactions would not absorb variation that should be explained by others. Inference for our model relies on the imposition of functional constraints which we implement as *constrained* optimisation in the weight space of neural networks.

First, we discuss the functional ANOVA decomposition in the context of neural network regression (i.e. assuming fixed inputs) and its unidentifiability. Then we present a constrained optimisation approach to enforce identifiability constraints, which we apply to the CVAE. We further modify the CVAE to incorporate feature-level sparsity for improved interpretation. In our experiments, we demonstrate the benefits of neural decomposition in a series of experiments, including large-scale single cell gene expression data and CelebA image data, thus demonstrating the scalability of this modelling framework in comparison to the c-GPLVM approach.

4.2 Identifiable Neural Decomposition

As opposed to traditional functional ANOVA use cases where functions are e.g. represented by smoothing splines (Gu, 2013) or Gaussian Processes (as discussed in Chapter 3), here we propose to use deep neural networks instead. Specifically, our goal is to perform functional ANOVA decomposition as part of the (C)VAE, in order to simultaneously perform dimensionality reduction as well as obtain a feature-level variance decomposition which would aid interpretability. Our proposal, the *neural* decomposition (ND), would enable separating main effects from complex

interactions while otherwise maintaining the flexibility of deep generative models and fast, approximate inference via variational methods.

We focus on the *generative* part of the (C)VAE model, i.e. the decoder network which we denote by $f^\theta(\mathbf{z}, \mathbf{c})$. Note that as opposed to the c-GPLVM setup discussed in the previous chapter, the decoder in the (C)VAE framework is a parametric mapping, where θ represents the neural network weights (that are typically treated in a non-Bayesian way).

4.2.1 Decomposing Neural Network Regression

First, we discuss identifiability in a general neural network regression setting, where we are interested in decomposing $f^\theta : \mathbb{R}^D \to \mathbb{R}$ with some inputs $\mathbf{x} \in \mathbb{R}^D$ and output $y \in \mathbb{R}$. We would like to decompose as follows

$$f^\theta(\mathbf{x}) = f_0^\theta + \sum_k f_k^\theta(x_{ik}) + \sum_{k,l} f_{kl}^\theta(x_k, x_l) + \ldots + f_{1,\ldots,D}^\theta(x_1, \ldots, x_D) \tag{4.1}$$

where all functions $f_k^\theta, f_{k,l}^\theta, \ldots, f_{1,\ldots,D}^\theta$ are parameterised by neural networks.

We note that without any additional constraints on the neural networks this decomposition (4.1) is unidentifiable, as formalised below.

Proposition 4. *Let $f_i^\theta, f_{ij}^\theta, \ldots, f_{1,\ldots,D}^\theta$ be neural networks with the same architecture, i.e. assuming that the networks only differ in the number of inputs*[1]*. Then decomposition (4.1) is not unique.*

Proof. Let $\mathcal{U} \subsetneq \mathcal{I}$ be two sets of indeces, and let $f_\mathcal{U}^\theta$ and $f_\mathcal{I}^\theta$ be the respective neural networks. We separate the first fully-connected layer (affine transformation on the inputs $\mathbf{x}$) from the rest of the transformations, which we denote by $g(\cdot)$. Note that then, according to our assumption, both neural networks can be expressed as follows

$$f_\mathcal{U}^\theta(\mathbf{x}_\mathcal{U}) = g(\mathbf{b}^\mathcal{U} + \mathbf{W}^\mathcal{U}\mathbf{x}_\mathcal{U}) \qquad \text{where } \mathbf{b}^\mathcal{U} \in \mathbb{R}^H, \mathbf{W}^\mathcal{U} \in \mathbb{R}^{H \times \dim(\mathbf{x}_\mathcal{U})}$$

$$f_\mathcal{I}^\theta(\mathbf{x}_\mathcal{I}) = g(\mathbf{b}_\mathcal{I} + \mathbf{W}_\mathcal{I}\mathbf{x}_\mathcal{I}) \qquad \text{where } \mathbf{b}^\mathcal{I} \in \mathbb{R}^H, \mathbf{W}^\mathcal{I} \in \mathbb{R}^{H \times \dim(\mathbf{x}_\mathcal{I})}$$

[1] Assuming a fully connected first layer, i.e. that the first transformation applied to inputs $\mathbf{x}_\mathcal{I}$ is $\mathbf{W}\mathbf{x}_\mathcal{I} + \mathbf{b}$ where $\mathbf{b} \in \mathbb{R}^H, \mathbf{W} \in \mathbb{R}^{H \times \dim(\mathbf{x}_\mathcal{I})}$ where H is the dimensionality of the first hidden layer and $\dim(\mathbf{x}_\mathcal{I})$ is the input dimensionality. By "the same architecture" we mean that $\dim(\mathbf{x}_\mathcal{I})$ is the only element that is allowed to vary.

Note that the functional subspace $\{f^\theta_\mathcal{U}(\mathbf{x}_\mathcal{U}) : \theta \in \mathbb{R}^{\dim(\theta)}\}$ is strictly a subset of the functional subspace $\{f^\theta_\mathcal{I}(\mathbf{x}_\mathcal{I}) : \theta \in \mathbb{R}^{\dim(\theta)}\}$, because the elements in $\mathbf{W}_\mathcal{I}\mathbf{x}_\mathcal{I}$ can be seen as a special case of $\mathbf{W}_\mathcal{U}\mathbf{x}_\mathcal{U} + \mathbf{W}_{\mathcal{I}\setminus\mathcal{U}}\mathbf{x}_{\mathcal{I}\setminus\mathcal{U}}$ where $\mathbf{W}_{\mathcal{I}\setminus\mathcal{U}}$ values have been set to zero. Now Proposition 1 directly applies. □

As a result, without any additional constraints formula (4.1) is not meaningful: it can be used for predictive purposes, but the importance of different terms has no direct interpretation since higher-order interactions can *absorb* variability that could be explained by main effects or low-order interactions.

As in functional ANOVA, to turn this into an *identifiable* learning problem, the solution would be to introduce functional constraints. Specifically, we want to introduce mean-zero integral constraints to constrain the marginal effects of every neural network $f^\theta_\mathcal{I}(\mathbf{x})$ to be zero. Applying Proposition 2 to neural networks, we want to

$$\int f^\theta_\mathcal{I}(\mathbf{x}_\mathcal{I})dx_i = 0 \quad \text{for all } \; i \in \mathcal{I}$$

for all neural networks $f^\theta_\mathcal{I}$, $\mathcal{I} \subset \{1, \ldots, D\}$. However, enforcing these constraints is challenging. The elegant strategy that we used for Gaussian Processes is not transferrable to neural networks. In section 4.2.2 we will discuss a different strategy.

4.2.2 Inference under integral constraints

This is a constrained optimisation problem, which is non-trivial to solve in the context of deep learning, where state-of-the-art off-the-shelf optimisation techniques, such as Adam (Kingma and Ba, 2014), are typically implemented for unconstrained problems only. Thus alternative strategies have to be considered. We turn to the *Augmented Lagrangian* method, also known as the method of multipliers (Hestenes, 1969; Powell, 1969), to enforce such constraints. We refer to Platt and Barr (1988) for an adaptation to neural networks.

Augmented Lagrangian for a single constraint: For illustration, we first consider the case with a univariate input x, which corresponds to learning the decomposition $f^\theta_0 + f^\theta(x)$ subject to the constraint $\int f^\theta(x)dx = 0$. That is, to

separate out the intercept f_0^θ, we want to restrict the f^θ to a subspace such that $\int f^\theta(x)dx = 0$. To enforce this constraint, we will augment the objective function[2] with additional penalty term(s) which will be equal to zero when the integral constraints are fulfilled.

One such approach would be the *penalty method* which incorporates a penalty term $c \cdot \left(\int f^\theta(x)dx\right)^2$ where c is a fixed constant. This approach has the disadvantage that for a fixed value of c we do not have any guarantees that constraints would be fulfilled exactly.

Alternatively one could introduce a penalty $\lambda \int f^\theta(x)dx$ where now λ is treated as a parameter. This is analogous to the use of Lagrange multipliers, and following the terminology of (Platt and Barr, 1988), we refer to this as the Basic Differential Multiplier Method (BDMM). Instead of gradient updates $\lambda^{t+1} = \lambda^t - \eta\left(\int f^\theta(x)dx\right)$, BDMM would follow the opposite direction $\lambda^{t+1} = \lambda^t + \eta\left(\int f^\theta(x)dx\right)$ when optimising λ. Platt and Barr showed that this corresponds to optimisation behaviour where the system undergoes damped oscillation.

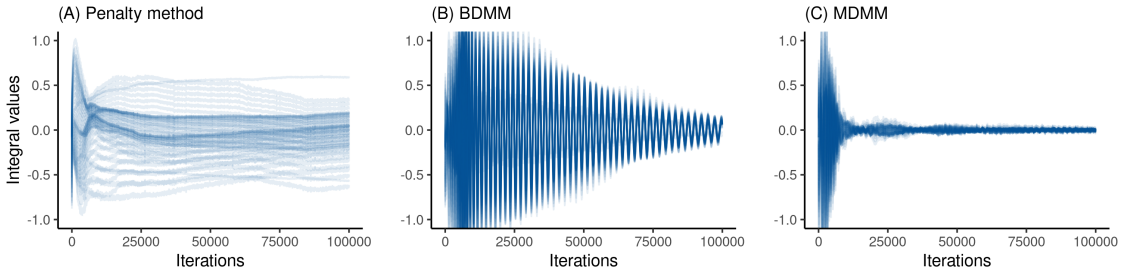


Figure 4.1: Different strategies for optimisation under constraints, showing the traces for $\int f^\theta(x_1, x_2)dx_2$ on a grid of x_1 values (each line corresponds to one grid point) over 100 000 iterations. (A) The constraints have not been fulfilled by the penalty method with a fixed c. (B) BDMM exhibits oscillating behaviour and integrals are slowly converging towards zero, whereas the hybrid (C) MDMM which combines the two penalties ina (A) and (B), using the same c and same learning rate η, leads to optimisation which results in ≈ 0 integral values much more quickly.

We have empirically compared the behaviour of the penalty method and the BDMM approach on a neural network regression problem with two-dimensional inputs, as

[2]e.g. the negative log-likelihood for NN regression or classification, or negative ELBO for variational methods

shown in Figure 4.1(A, B). A fixed penalty does not necessarily lead to fulfilled constraints, whereas the oscillating behaviour in BDMM leads to the integrals converging towards zero.

Finally, these two penalty terms can be combined, resulting in a hybrid constrained optimisation objective

$$\min_{\theta} \left\{ \mathcal{L}^{\theta} + \lambda \int f^{\theta}(x)dx + c \left(\int f^{\theta}(x)dx \right)^2 \right\},$$

where $\mathcal{L}^{\theta}$ is our initial objective, λ is optimised and c is a fixed constant. This scheme, the Modified Differential Multiplier Method (MDMM), results in more robust behaviour, both from a theoretical perspective (Platt and Barr, 1988) as well as supported by empirical evidence as shown in our Fig 4.1. MDMM is used throughout our numerical experiments.

Enforcing multiple identifiability constraints: Next we discuss how to handle multiple constraints, illustrating this on a special case with a two-dimensional input (x_1, x_2), which corresponds to learning a decomposition

$$f_0^{\theta} + f_1^{\theta}(x_1) + f_2^{\theta}(x_2) + f_{12}^{\theta}(x_1, x_2).$$

In order to satisfy the constraints $\int f_{12}^{\theta}(x_1, x_2)dx_2$ for every value $x_1 \in [a_1, b_1]$, we need to introduce a Lagrange multiplier $\lambda_{x_1}(x_1)$ which is now indexed by a continuous-valued x_1. The additional penalty corresponding to this term will be $\int \lambda_1(x_1) \left(\int f_{12}^{\theta}(x_1, x_2)dx_2 \right) dx_1$. Analogously, we will also introduce the term $\int \lambda_2(x_2) \left(\int f_{12}^{\theta}(x_1, x_2)dx_1 \right) dx_2$ in addition to $\lambda_1 \int f_1^{\theta}(x_1)dx_1$ and $\lambda_2 \int f_2^{\theta}(x_2)dx_2$. As a result, we end up with the following augmented objective (for brevity, we outline the penalty terms corresponding to BDMM only):

$$\begin{aligned} \min_{\theta} \Bigg\{ \mathcal{L}^{\theta} &+ \lambda_1 \int_{a_1}^{b_1} f_1^{\theta}(x_1)dx_1 + \lambda_2 \int_{a_2}^{b_2} f_2^{\theta}(x_2)dx_2 + \\ &+ \int_{a_1}^{b_1} \lambda_1(x_1) \left(\int_{a_2}^{b_2} f_{12}^{\theta}(x_1, x_2)dx_2 \right) dx_1 + \\ &+ \int_{a_2}^{b_2} \lambda_2(x_2) \left(\int_{a_1}^{b_1} f_{12}^{\theta}(x_1, x_2)dx_1 \right) dx_2 \Bigg\}. \end{aligned} \tag{4.2}$$

In practice, we can choose to estimate these integrals using either e.g. via quadrature or via Monte Carlo methods. In all our experiments, we rely on quadrature.

One might wonder how do we know whether the constraints have been (approximately) satisfied after a certain number of iterations. We propose to approach this as follows: we establish a desired tolerance threshold ε and evaluate the integrals after optimisation to make sure that all learned NNs have been constrained to the desired functional subspaces within the desired tolerance. We point out that all the additional penalty terms need to be evaluated during training time only. This is important, because once the model has been trained, we have obtained NNs which satisfy the constraints, and thus it can be used for prediction without any additional costs at test time.

4.2.3 Decomposing CVAEs

Here we discuss how to apply the previously introduced methodology to the CVAE setup, thus combining amortised variational inference and constrained optimisation techniques.

We now return to our original goal which is understanding the sources of variation in (C)VAEs. Our goal is to learn a decomposition as part of the *generative* model, that is we would like to decompose the *decoder* $f^\theta(\mathbf{z}, \mathbf{c})$ to extract additive marginal and interaction effects. Here we illustrate the special case with $\mathbf{z}, \mathbf{c} \in \mathbb{R}$, which corresponds to learning the following generative model

$$\begin{aligned} \mathbf{z}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{y}_i \,|\, \mathbf{z}_i, \theta &\sim p(\mathbf{y}_i \,|\, f_0^\theta + f_z^\theta(\mathbf{z}_i) + f_c^\theta(\mathbf{c}_i) + f_{zc}^\theta(\mathbf{z}_i, \mathbf{c}_i)). \end{aligned} \tag{4.3}$$

As previously discussed, in order to obtain an identifiable neural decomposition, we need to enforce the following integral constraints:

- $\int f_z^\theta(\mathbf{z})dz = 0$ and $\int f_c^\theta(\mathbf{c})dc = 0$
- $\int f_{zc}^\theta(\mathbf{z}, \mathbf{c})dz = 0$ for all $\mathbf{c}$
- $\int f_{zc}^\theta(\mathbf{z}, \mathbf{c})dc = 0$ for all $\mathbf{z}$.

Denoting the CVAE ELBO

$$\mathcal{L}^{\theta,\phi} = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i,\mathbf{c}_i)}[\log p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{c}_i)] - \mathrm{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i,\mathbf{c}_i)||p(\mathbf{z}_i))$$

where $\log p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{c}_i)$ has the specific form as given in (4.3), we augment this objective with the penalty terms analogously to equation (4.2) as follows

$$\begin{aligned}\min_{\theta,\phi}\Bigg\{ &- \mathcal{L}^{\theta,\phi} + \lambda_z \int_{a_1}^{b_1} f_z^\theta(z)dz + \lambda_c \int_{a_2}^{b_2} f_c^\theta(c)dc + \\ &+ \int_{a_1}^{b_1} \lambda_z^{zc}(z) \left(\int_{a_2}^{b_2} f_{zc}^\theta(z,c)dc \right) dz + \\ &+ \int_{a_2}^{b_2} \lambda_c^{zc}(c) \left(\int_{a_1}^{b_1} f_{zc}^\theta(z,c)dz \right) dc \Bigg\}.\end{aligned} \tag{4.4}$$

This objective function is not necessarily a lower bound any more, but it reduces to the ELBO once the constraints become fulfilled during optimisation. In practice, we recommend the strategy of introducing a desired tolerance threshold ε to make sure the integrals are sufficiently close to zero, in which case the loss function has its original interpretation as a lower bound.

4.2.4 Sparse Neural Decomposition

For interpreting what the decomposed (C)VAE has learnt, in addition to obtaining the variance decomposition it is also of practical interest to *detect* the presence or absence of dependence on certain input coordinates, e.g. detect groups of genes which depend purely on $\mathbf{z}$ and not on $\mathbf{c}$. This would make it easier to interpret.

The feature-level variance decomposition from ND does not explicitly *identify* such groups by default. One approach would be to apply an ad-hoc thresholding to decide which effects are zero. For a more principled probabilistic approach, in this section we introduce Bernoulli random variables for every decoder network and for every data dimension. For the special case when $\mathbf{z}, \mathbf{c} \in \mathbb{R}$, we would introduce $s_z^{(j)}, s_c^{(j)}, s_{zc}^{(j)} \sim \text{Bernoulli}(p)$ for every feature $j = 1, \ldots, P$, each indicating the presence of f_z^θ, f_c^θ and f_{zc}^θ non-zero effects for feature j. That is, we define the decoder to have the following structure

$$\mathbf{y}_i|\mathbf{z}_i,\mathbf{c}_i,\theta,\mathbf{s} = \mathbf{s}_c f_c^\theta(\mathbf{c}_i) + \mathbf{s}_z f_z^\theta(\mathbf{z}_i) + \mathbf{s}_{cz} f_{cz}^\theta(\mathbf{z}_i,\mathbf{c}_i) + \varepsilon_i$$

where $\mathbf{s}_c, \mathbf{s}_z, \mathbf{s}_{cz}$ have the same dimensionality as observations $\mathbf{y}$. For scalable variational inference, we use the continuous relaxation of Bernoulli random variables (Maddison et al., 2017) which allows us to implement a reparameterisation based inference scheme. We specify a prior $p(\mathbf{s}) = \text{RelaxedBernoulli}(p_0)$ for all $\mathbf{s}_c, \mathbf{s}_z, \mathbf{s}_{cz}$ and approximating distributions $q(\mathbf{s}) = \text{RelaxedBernoulli}(\mathbf{u})$ where $\mathbf{u}$ are variational parameters. While in principle we could introduce a hyperprior for the (Relaxed)Bernoulli parameter p_0, in all our experiments we found it sufficient to use the value $p_0 = 0.1$.

The integration of the sparsity priors and the neural decomposition model is summarised in Figure 4.2.

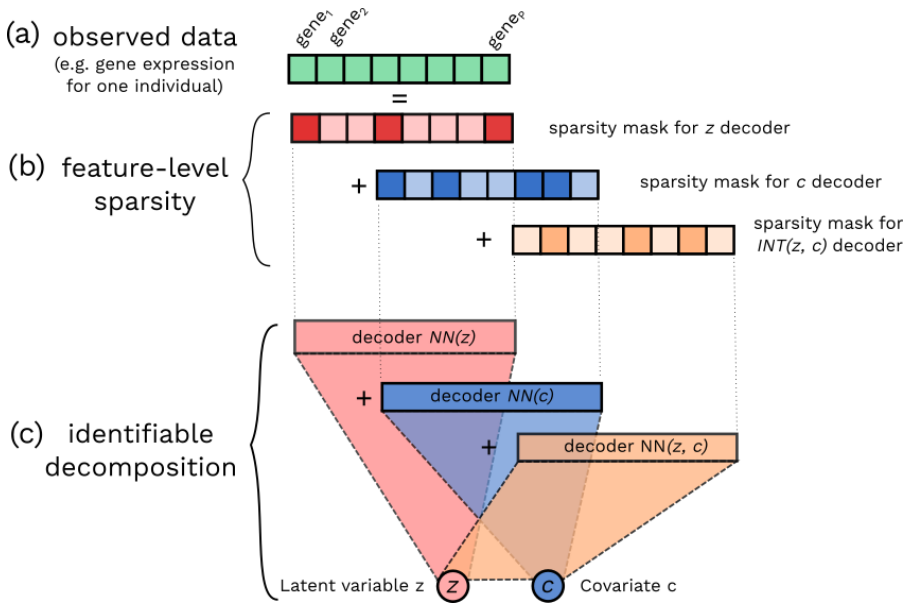


Figure 4.2: Diagram of ND decoding model. For every gene in panel (a), we learn an identifiable decomposition of decoder networks $f_z^\theta(\mathbf{z}_i)$, $f_c^\theta(\mathbf{c}_i)$ and $f_{zc}^\theta(\mathbf{z}_i, \mathbf{c}_i)$ as shown in panel (c), which are then elementwise multiplied with the respective Bernoulli sparsity masks (panel (b)) for easier interpretability.

4.3 Experiments

The value of ND lies in *scalable* and *identifiable* functional decompositions. Unfortunately in real data the true underlying functional decompositions are unknown, thus we are unable to quantify the correctness of the inferred decompositions on

real data. For this reason, we seek to carefully characterise the behaviour of ND in a controlled setting before considering the real-life genomics example.

4.3.1 Comparison with Decomposable c-GPLVM

We would like to characterise how the Neural Decomposition behaviour relates to a conceptually similar c-GPLVM decomposition. The latter, being a decomposition of Gaussian Processes, is better understood in the sense that for any configuration of kernel hyperparameters, the mean-zero decomposition can be performed in closed form. Thus, the mean-zero GP approach is *exact* in the sense that all the integral constraints are fulfilled exactly rather than approximately.

Thus, in some sense, one could consider a GP-decomposition as a golden standard for this purpose, however note that the set of functions that have positive probability under the GP prior (e.g. under the squared exponential kernel these are infinitely differentiable functions) does not necessarily overlap with the set of functions that are parameterised by a neural network. The former is determined by the kernel, whereas the latter is determined by various neural architectural choices.

In Figure 4.3 we have investigated how the behaviour of neural decomposition differs from the GP-based functional decomposition on synthetic example that involves inference over f_z, f_c, and f_{zc}. We have visualised the inferred GP mean and the inferred ND mappings (using a one hidden layer architecture, with 64 neurons, either with a ReLU or Softplus nonlinearity) for both f_z (on the left) and f_{zc} (on the right), highlighting the L^1 distance between the two functions. For the ND with identifiability constraints (top row), this distance is relatively small and there are only minor differences from the GP posterior means, whereas the mappings inferred by the unidentifiable ND (bottom row) differ significantly from the GP ones.

Despite its elegant theoretical underpinnings, the GP-decomposition suffers from scalability issues intrinsic to GP-based models. While the cubic complexity w.r.t. sample size N can be addressed via inducing-point methods, the scalability of

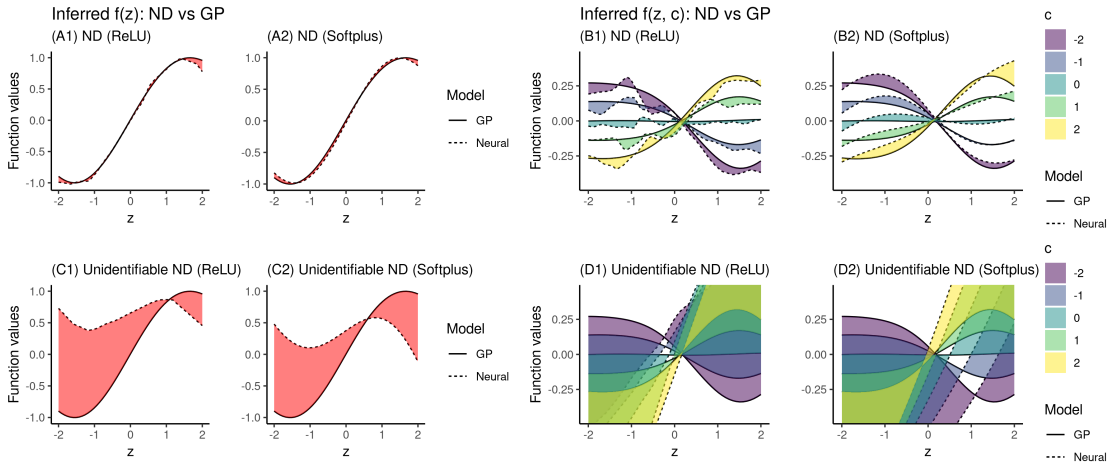


Figure 4.3: On a synthetic data set ($N = 500$) we compare the Neural Decomposition with the GP exact decomposition directly in the function space, visualising both the predicted mean for the inferred f_z (panels A) and f_{zc} (panels B, shown for $c \in \{-2, -1, 0, 1, 2\}$) as well as the L^1 distance (shaded area) between them. The functional subspace defined by ND depends on various architectural choices such as nonlinearity: we used ReLU in panels (A1, B1) and Softplus in (A2, B2). The ND without identifiability constraints (bottom row) learns a decomposition which lies far (in L^1 distance) from the GP one.

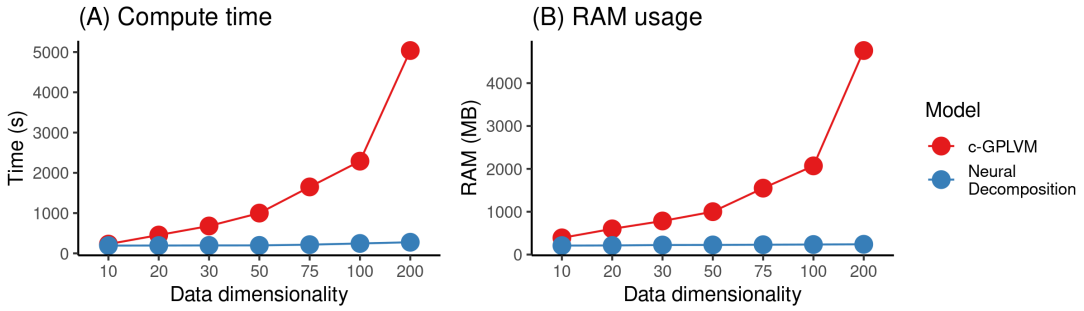


Figure 4.4: Neural decomposition is much more scalable w.r.t. data dimensionality than the GP decomposition (inducing-point implementation of the c-GPLVM), both in terms of compute time and memory requirements. For fixed sample size $N = 500$, we varied the dimensionality of data $P \in \{10, 20, \ldots, 100, 200\}$, and compared (A) the compute time, and (B) RAM usage for both methods when varying P (on x-axis). Experiments were made on a desktop with 8 Intel i7-6700 CPUs, both implementations in PyTorch.

c-GPLVM w.r.t. data dimensionality P can become the limiting factor for high-dimensional data. While the decomposable c-GPLVM (as formulated in section 3.3) scales linearly with P both in terms of compute and memory, even for moderate P it becomes prohibiting to fit c-GPLVM on a laptop because of the memory requirements, as shown in Figure 4.4.

4.3.2 Synthetic data

We first investigated the behaviour of ND in a controlled setting. We generated a variety of patterns, with a goal to mimic real gene expression data, thus including linear relationships, monotone warpings as well as non-monotone non-linear dependency structures (see Appendix A.1 for details). We applied three variants of neural decomposition: ND with a linear decoder, ND without identifiability constraints, and our full implementation.

Note that neural decomposition with a linear decoder (which we refer to as linear-CVAE) is closely related to the linear Latent Variable Regression model that we discussed before, i.e. it corresponds to the following decomposition

$$y_i^{(j)} = \mu_j + \sum_{d=1}^{D} \alpha_d^{(j)} z_{id} + \sum_{k=1}^{C} \beta_k^{(j)} c_{ik} + \sum_{d=1}^{D} \sum_{k=1}^{C} \gamma_{dk}^{(j)} z_{id} c_{ik} + \varepsilon_{ij}$$

which can be seen as a particular extension of probabilistic PCA, fitted using amortised inference. Using a linear decoder in the VAE framework was explored e.g. by Svensson and Pachter (2019) as a way to combine the scalability of VAEs with interpretability of linear models. However, the linearity assumption restricts the flexibility of the model.

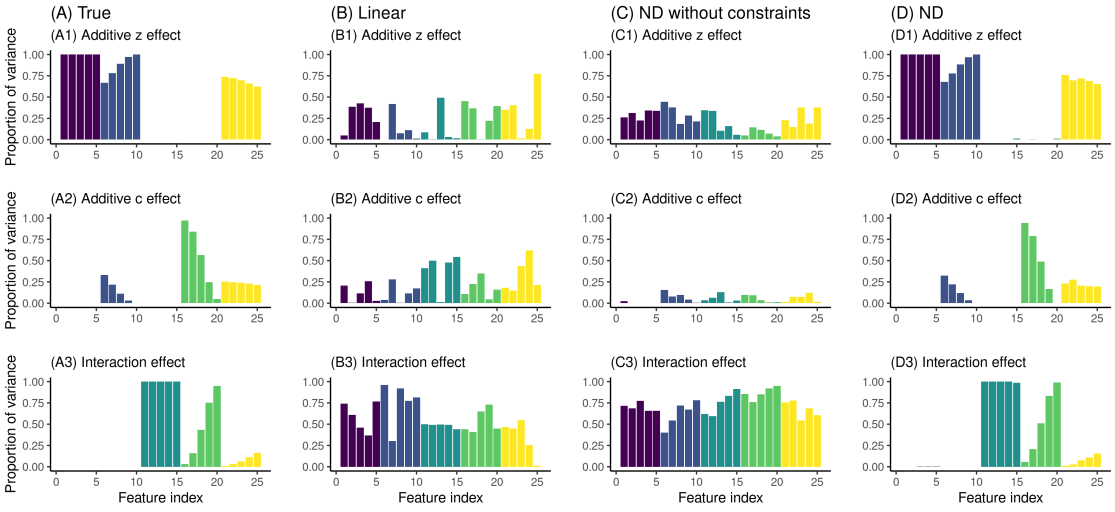


Figure 4.5: The true variance decomposition $\mathrm{Var}(f_z)$, $\mathrm{Var}(f_c)$, $\mathrm{Var}(f_{zc})$ on a synthetic data set shown in (A1-A3) for all 25 features (which have been grouped in blocks of 5 for visual aid), compared to the inferred decompositions by (B) linear-CVAE, (C) unidentifiable ND, and (D) identifiable ND.

Table 4.1 summarises results for a series of experiments, quantifying how accurately was the true underlying variance decomposition identified. We separate the (a) purely linear and (b) a more challenging mix of linear and non-linear data generative mechanisms. For linear dependencies, the linear-CVAE works well, but so does ND-CVAE despite the increased flexibility. However, when considering more challenging non-linear dependencies, ND-CVAE outperforms the linear-CVAE.

Figure 4.5 illustrates the inferred variance decompositions for one particular synthetic data set, containing 25 features with varying dependency structures (e.g. the first five features exhibit a pure additive **z** effect). Neither the linear-CVAE nor the ND without constraints have been able to identify the correct decomposition: the former is unable to capture non-linearities in the data due to its restrictive modelling assumptions, and the latter suffers from unidentifiability which makes its inferred variance decomposition arbitrary.

Table 4.1: Correlation between the true underlying variance component $\text{Var}(f)$ and the inferred value, measured on synthetic gene expression data, under (a) linear and (b) non-linear data generative mechanism.

	(a) linear		
	$\text{Var}(f_z)$	$\text{Var}(f_c)$	$\text{Var}(f_{zc})$
linear-cVAE	**0.99**	**0.87**	**0.92**
naive ND-cVAE	0.56	0.42	0.37
ND-cVAE	**0.97**	**0.88**	**0.92**
	(b) non-linear		
linear-cVAE	0.46	0.05	0.40
naive ND-cVAE	0.41	0.38	0.19
ND-cVAE	**0.98**	**0.45**	**0.56**

To demonstrate that our approach is not restricted to a univariate **z**, we consider a synthetic data set generated from a two-dimensional latent space in the presence of confounding batch effects. The data consists of two batches where each feature in the data is either unperturbed, differs by a constant by batch, or varies with z_1 by batch (see Appendix A.2 detailed data generation scheme). Our goal is two-fold: First, we would want to learn a 2D latent space where **z** would be adjusted for the confounding batch effect. Second, we would like to characterise the variance decomposition of every feature in terms of z_1, z_2 and c.

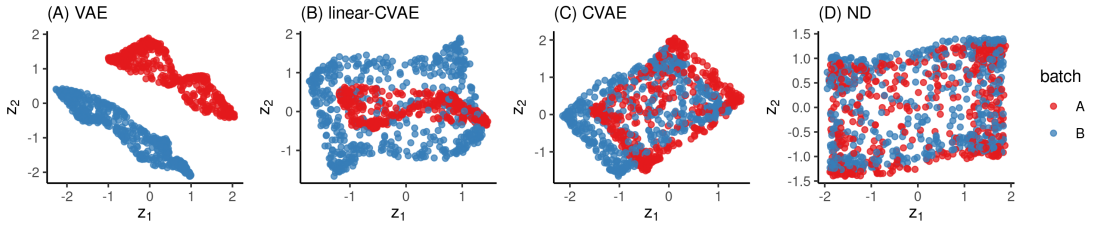


Figure 4.6: ND recovers a batch-adjusted two-dimensional **z** on a synthetic example whereas other approaches (VAE, CVAE, and linear-CVAE) struggle to appropriately adjust for known **c**.

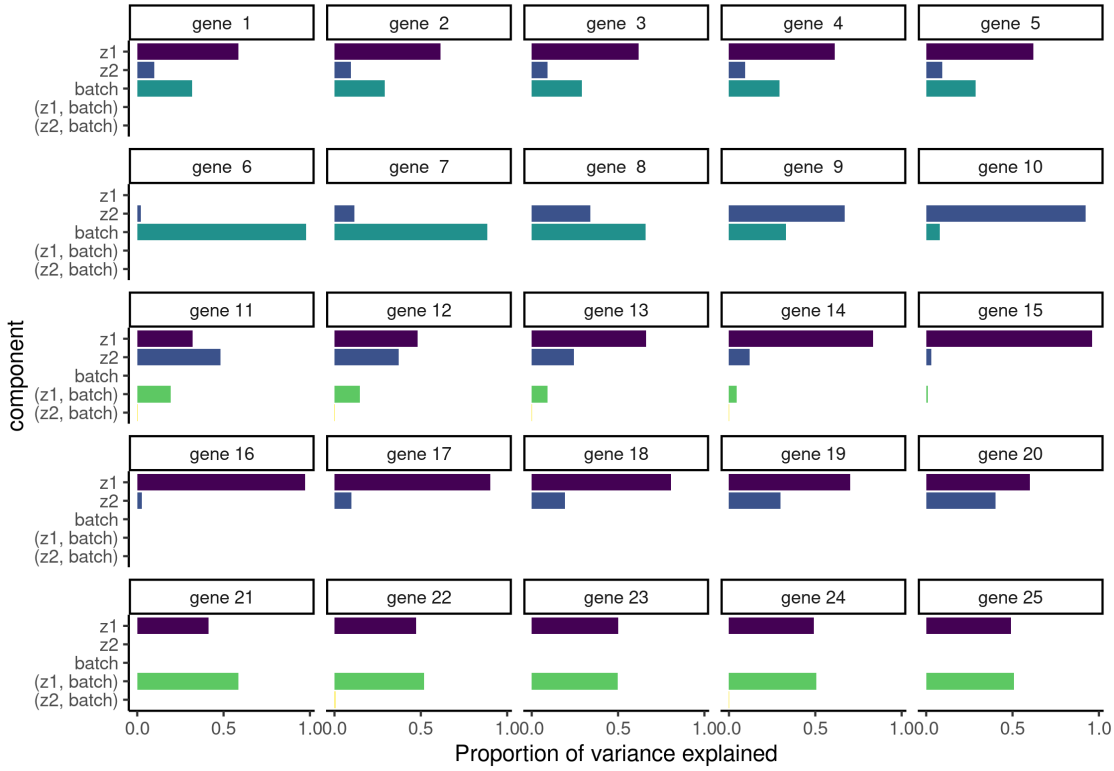


Figure 4.7: On the synthetic data set (see Appendix A.2), we characterised the neural decomposition for every gene as a function of z_1, z_2, batch c, and interactions (z_1, c) and (z_2, c).

We wanted to test if any VAE variant would be capable of achieving *batch correction* (using **c**=[batch]) in the latent space. With the restricted representation power of linear-CVAE, only translational shifts in the latent space could be corrected. Surprisingly, the standard CVAE did not entirely remove the batch effect in the latent space either. However, the sparse ND structure within the ND has correctly identified a (z_1, z_2) space in which the batches are now intermixed and the non-

linear batch effects removed (Figure 4.6). Furthermore, ND lets us characterise how features vary with latent z_1, z_2 and known c (Figure 4.7).

4.3.3 Decomposing CelebA

The motivation behind neural decomposition is understanding *tabular* (non-image) data. However, the ND methodology is generally applicable to a variety of problems, and we wanted to demonstrate how it enables scaling up functional ANOVA to large-scale problems. In this section we demonstrate ND on CelebA image data (Liu et al., 2015), treating the pixel space as if they were tabular features[3], thus using fully-connected non-convolutional networks.

4.3.3.1 Pixel-level decomposition

The CelebA data set contains more than $N > 200,000$ celebrity images together with some attribute annotations (e.g. whether the person is smiling, is wearing glasses, or whether they have a beard). To be able to capture relatively subtle effects like "glasses" and "beard" which are typically not captured in the inferred latent space of the VAE (e.g. as reflected by the empirical results in (Kim and Mnih, 2018)), we decided to include these as fixed covariates.

We consider a CVAE with four covariates c_1=Gender, c_2=Smiling, c_3=Glasses, c_4=Beard and a 1D latent variable $z \in \mathbb{R}$. Our goal is to perform a pixel-level decomposition and illustrate how the ND lets us interpret individual decoder networks. We fit a neural decomposition model with the decoder that has the following functional form

$$f^\theta_z(z) + \sum_{k=1}^{4} f^\theta_{c_k}(c_k) + \sum_{k=1}^{4} f^\theta_{zc_k}(z, c_k)$$

thus capturing the marginal additive effects of z and $\{c_k\}$ as well as their interactions as part of the decoding structure.

The inferred pixel-level sparsity masks in Fig 4.8 help us interpret what each of the decoder networks has learnt. Note that z captures how dark/light the background

[3]Our interest is *not* to tune ND to develop a state-of-the-art model for images, but instead to demonstrate ND on an easy-to-visualise high-dimensional use case

is, whereas the additive effects of covariates affect only relevant parts of the faces, e.g. relating to smiling, wearing glasses or having beard. The interaction effects between z and c_k are relatively small: they mostly affect pixels that interact with the overall background darkness, e.g. long hair for women, or the outer borders of glasses and beard.

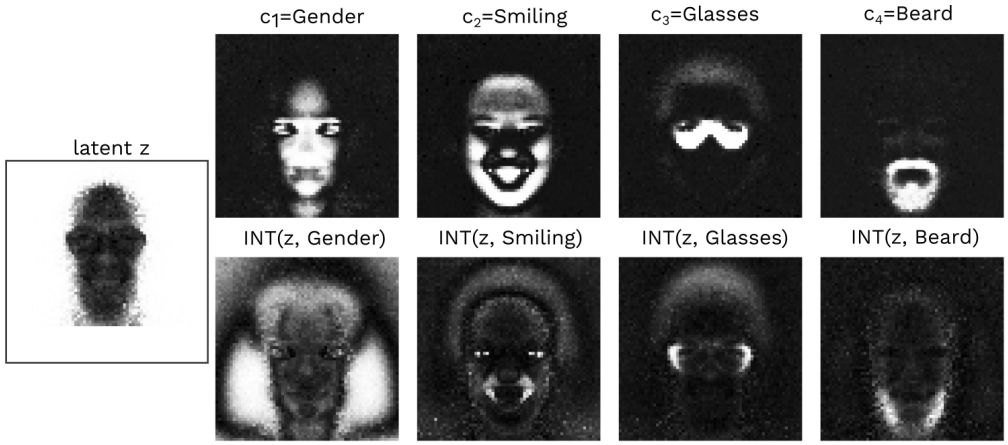


Figure 4.8: Inferred sparsity masks on CelebA data provide pixel-level insights into what each of the decoders in ND has learnt.

4.3.3.2 Improved generalisation

Despite the empirical success of deep generative models, their generalisation (e.g. to under-represented subpopulations) in the presence of sampling bias is still an active area of research (Zhao et al., 2018). Here, we demonstrate on a simple example, how additive structure in the decoder can sometimes improve the extrapolation properties of the CVAE.

We use CelebA data to mimic sampling bias as follows. We exclude all images of smiling men from the training set, and we explore how well the models (standard CVAE and ND) generalise at test time to these unobserved subgroups. We visualise predictions for the following subgroups:

- First, we visualise predictions for $z = 0$, Gender=Male, Smiling=False, Glasses=False, Beard=False (this a scenario that was observed during training, shown in the left-most panel of Figure 4.9).

- Then we consider three new scenarios that require extrapolation from training data:
  - First, setting Smiling=True (2nd panel).
  - Additionally, setting Glasses=True (3rd panel).
  - Additionally, setting Beard=True (4th panel).

as shown in Figure 4.9.

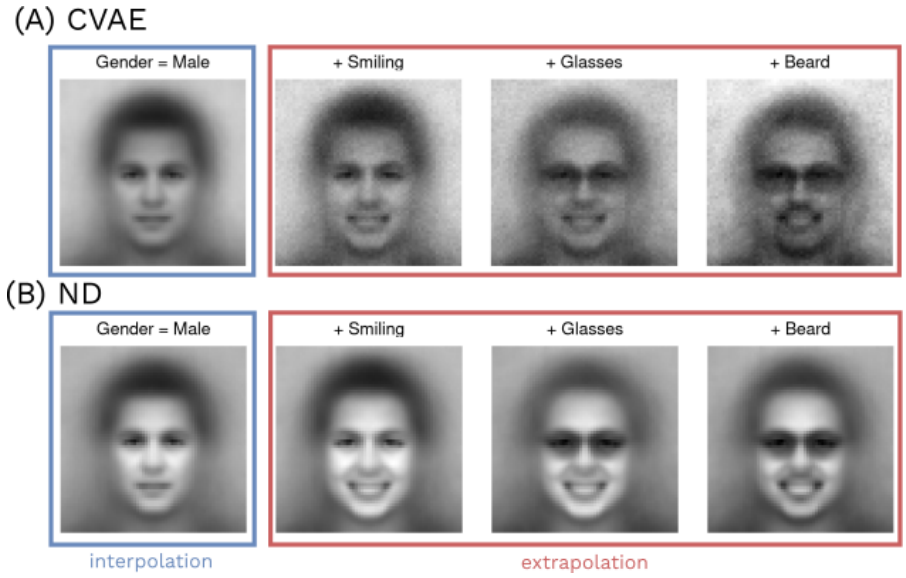


Figure 4.9: The additive structure within ND can aid extrapolation to unseen subgroups.

It is not surprising that CVAE has not been able to generate clear images for unseen subgroups: the decoder has not been trained on these particular combinations of inputs. However the additive structure in ND has led to visually better quality images. The same is indicated by the loglikelihood values on held-out images of smiling men (mean ± sd):

CVAE	-3181 (± 630)
ND-CVAE	**-2497 (± 319)**

4.3.4 Gene expression

We next examine a publicly available time-series single-cell RNA-seq (scRNA-seq) data set of bone marrow derived dendritic cells responding to particular stimuli (Shalek et al., 2014). In this experiment, cells were exposed to either LPS, a

component of Gram-negative bacteria, or PAM, a synthetic mimic of bacterial lipopeptides, and scRNA-seq performed at 1, 2, 4 and 6 h after stimulation. With the capture time information, the original study studied single-cell gene expression dynamics under the two exposures, however, the cells behave *asynchronously* and heterogeneity exists within the cellular populations at each time point (i.e. after 1 hour of stimulation, the cell do not reach exactly the same biochemical state). Previous analyses have suggested this data set is more suited to a latent variable (so-called *pseudotime*) analysis (Campbell and Yau, 2018b) where the capture times are treated as unreliable and the latent variable is used instead to capture a continuous measure of how the cells are continually changing in response to each stimuli. This has been illustrated in Figure 4.10(A).

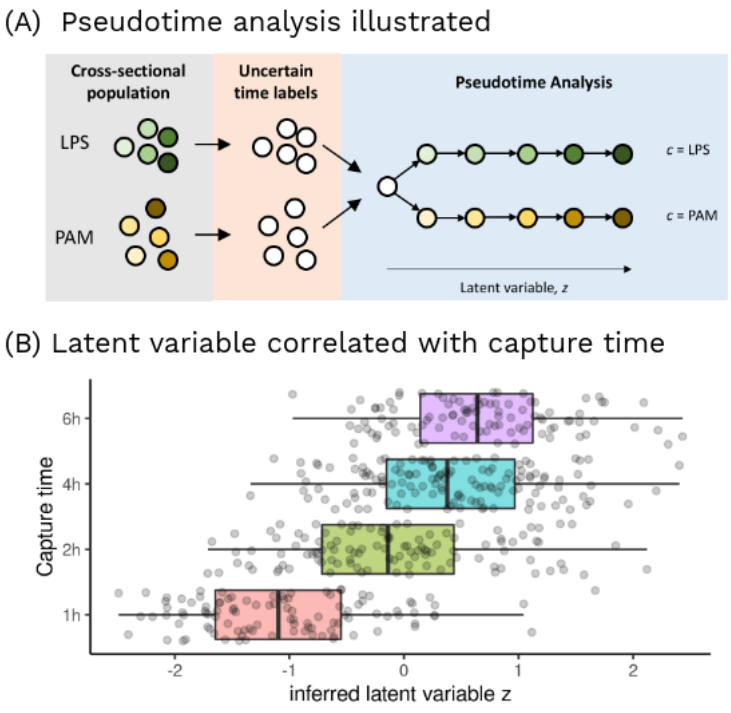


Figure 4.10: (A) Schema illustrating pseudotime analysis approach to mouse dendritic cells. (B) Inferred z (pseudotime) correlates with capture time.

We conducted an analysis using ND where we encoded the stimulant to which the cells were exposed as a binary covariate ($c_i \in \{\text{LPS}, \text{PAM}\}$) and used a single dimension for the latent variable $z_i \in \mathbb{R}$. Each gene was therefore modelled as a combination of main effects due to 1) LPS/PAM exposure, 2) temporal effects

(independent of stimulant type), and 3) temporal interaction effects that were modulated by the stimulant.

We applied ND to 820 cells using the 7,500 most variable genes[4], following the previously published *linear* analysis (Campbell and Yau, 2018b). In principle, the ND framework lets us easily make use of any likelihood, e.g. we could use the zero-inflated negative binomial likelihood that is commonly used for single-cell applications (Lopez et al., 2018). However, here we decided to use a Gaussian likelihood, so that we could directly compare against the results of Campbell and Yau (2018b). Furthermore, this lets us retrospectively compare the ND functional decomposition with the c-GPLVM one.

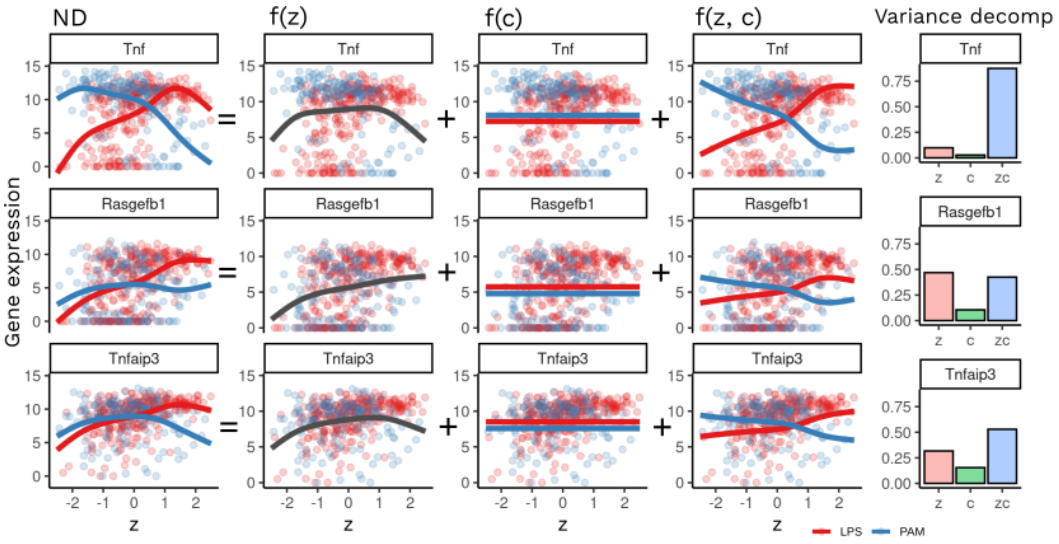


Figure 4.11: ND on the large-scale scRNA-seq data of dendritic cells, using $c \in \{\text{LPS}, \text{PAM}\}$, lets us decompose gene-level variation (shown for selected 3 genes) into the marginal z effect, the marginal c effect, and the non-linear interaction between the two, together with the fraction of variance attributed to each component. As opposed to earlier linear approaches (Campbell and Yau, 2018b), the decomposition provided by ND is non-linear.

Our inferred latent dimension was indeed correlated with capture time and therefore is indeed a measure of the continuous progression of these cells under stimulation, as shown in Figure 4.10(B). Furthermore, we were able to decompose the expression of each gene into components that were dependent on non-linear additive effects

[4]data from https://kieranrcampbell.github.io/phenopath/

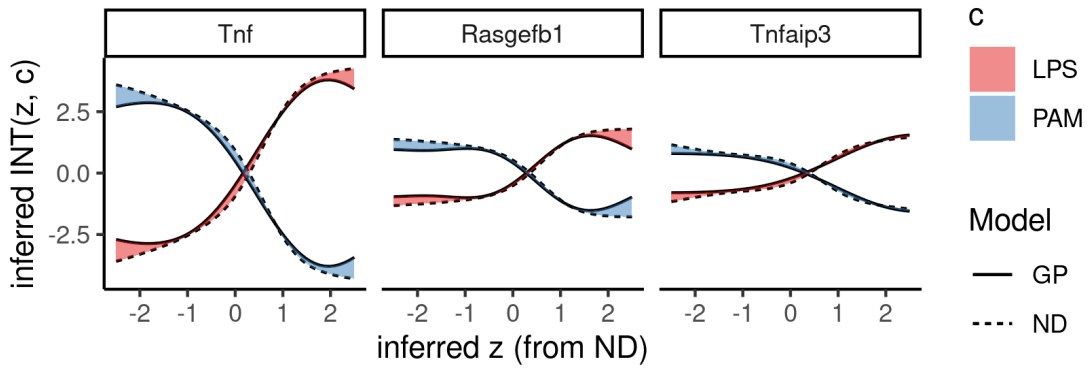


Figure 4.12: Using the inferred latent z from the ND as fixed input for c-GPLVM, we visually compare the ND and GP functional decomposition for three selected genes, plotting the inferred $f_{zc}(z, c)$ similarly to Figure 4.3.

of pseudotime z, covariate c, and interactions (z, c). In Figure 4.11, we highlight three genes previously identified to exhibit interaction effects (Campbell and Yau, 2018b). However, a limitation of the previous analysis was the reliance on linear models, which leaves open the possibility of model misspecification driving false positives in this very noisy single cell expression data. Here we have used a more flexible, non-linear CVAE-based approach. The genes *Tnf*, *Rasgefb1* and *Tnfaip3* all exhibit strong interaction effects and the dependence of the gene expression on (z, c) does follow a near-linear behaviour even under a flexible non-linear model, thus confirming the findings of (Campbell and Yau, 2018b) under a more flexible class of models, while providing a more accurate feature-level variance decomposition (rightmost panels in Figure 4.11).

We also wanted to retrospectively compare the ND functional decompositions with the c-GPLVM, to see if the conclusions are aligned between the two model classes. Fitting the latter on all genes would require the use of a compute server, but using the latent variable from ND as fixed inputs, we can compare the functional decompositions for a small number of selected genes. Visual comparison is shown in Figure 4.12.

We next wanted to examine the subsets of genes that exhibit the strongest additive and/or interaction effects. For this purpose, we identified the top 50 genes with the largest 1) additive z, 2) additive c, and 3) interaction effects as identified by

ND, and applied the popular dimensionality reduction and visualisation algorithm UMAP (McInnes et al., 2018), to examine these subsets via a two-dimensional representation (Figure 4.13). As expected, the set of genes with strong additive z effects but smaller additive c and interaction effects, showed considerable intermixing between cells stimulated under the two conditions. These genes exhibit behaviours that are largely independent of stimulus, so we would not expect segregation of the genes by stimulus type. In contrast, the UMAP visualisation of the gene sets with strong additive c or interaction effects shows considerable separation between the LPS and PAM stimulated cells. These are genes whose behaviour is heavily influenced by the type of stimulation as well as the latent variable.

Overall, this suggests that ND is identifying relevant structure in the single cell data and correctly attributing the appropriate feature behaviours to the relevant subsets of genes. Furthermore, unlike previous analyses, our flexible non-linear models permits greater robustness to model misspecification.

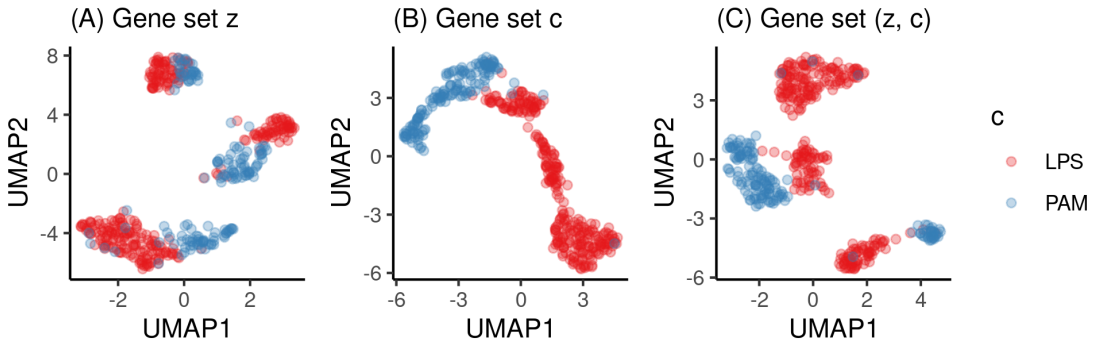


Figure 4.13: UMAP visualisations of gene sets with large (A) additive z, (B) additive c and (C) interaction effects from the mouse dendritic single cell data.

4.4 Related work

Decomposable c-GPLVM: As we discussed in Section 4.3.1, both the decomposable c-GPLVM and ND can be used to solve the same task from the application perspective, however the modelling frameworks are different and model fitting strategies employ different inference techniques. As a result, ND exhibits more desirable scalability properties which make it more suited for large-scale

data sets. Also, we note that replacing the Gaussian likelihood with a problem-dependent likelihood (such as Bernoulli, or zero-inflated negative binomial for single-cell applications) is straightforward in the ND framework, whereas non-Gaussian likelihoods require a different inference strategy in GP models.

Disentanglement with VAEs: In the context of VAEs, there has been substantial recent interest in disentanglement approaches (Bengio et al., 2013) that force the dimensions of the latent space $\mathbf{z}$ to be responsible for distinct generative factors in the data while being relatively invariant to other factors (Higgins et al., 2017; Kim and Mnih, 2018; Mathieu et al., 2019). Therefore learning disentangled representations implies learning which factors of variation are independent of the rest. Disentanglement approaches for VAEs therefore focus on the implementation of the encoder, $q_\phi(\mathbf{z}_i|\mathbf{y}_i)$. In this chapter, our focus is instead on a structured design of the *decoder*, thus our work has a fundamentally different goal. As a consequence, all the work we describe can in principle be combined alongside appropriate VAE disentanglement techniques, although throughout this work we only apply the standard encoding mechanism.

4.5 Discussion

We have proposed a VAE-framework where we embed functional ANOVA decompositions within the decoding structure. This specification allows us to associate variation in the latent space and other covariates to feature-level variability, leading to interpretability that is not present in existing (C)VAE models. Our work brings together ideas from classical statistics and constrained optimisation, while leveraging modern deep learning software to develop comparatively fast, scalable, and interpretable models for dimensionality reduction.

We note that the potential limitation of ND is the fact that our approach requires estimating integrals, which in general is challenging. As in this work we only considered one- and two-dimensional latent variables, we could effectively use quadrature, however, for larger $\dim(\mathbf{z})$ we might need to develop a different approach.

The ND construction makes the *functional decomposition* (with fixed inputs) identifiable. In principle, the entire latent variable model could be made identifiable, when combining ND with monotonicity constraints on the neural networks (Pierson et al., 2019), however this would considerably restrict the flexibility of the model.

5

BasisVAE: Translation-invariant feature-level clustering with Variational Autoencoders

Contents

5.1 Introduction 99
5.2 BasisVAE 102
5.2.1 Translation and scale invariance 103
5.2.2 Specifying the number of basis functions 105
5.2.3 Generative model and inference 106
5.2.3.1 Non-collapsed inference 107
5.2.3.2 Collapsed inference 107
5.3 Related work 110
5.4 Experiments 111
5.4.1 Toy data 111
5.4.2 Motion capture data 113
5.4.3 Single cell Spermatogenesis data 115
5.5 Discussion 118
5.5.1 On well-definedness of translation-invariant BasisVAE . 119
5.5.2 Future work 121

5.1 Introduction

High-dimensional data analysis employs dimensionality reduction techniques to extract major components of variation, typically using non-linear techniques, that enable interpretation and visualisation. Clustering can additionally be utilised

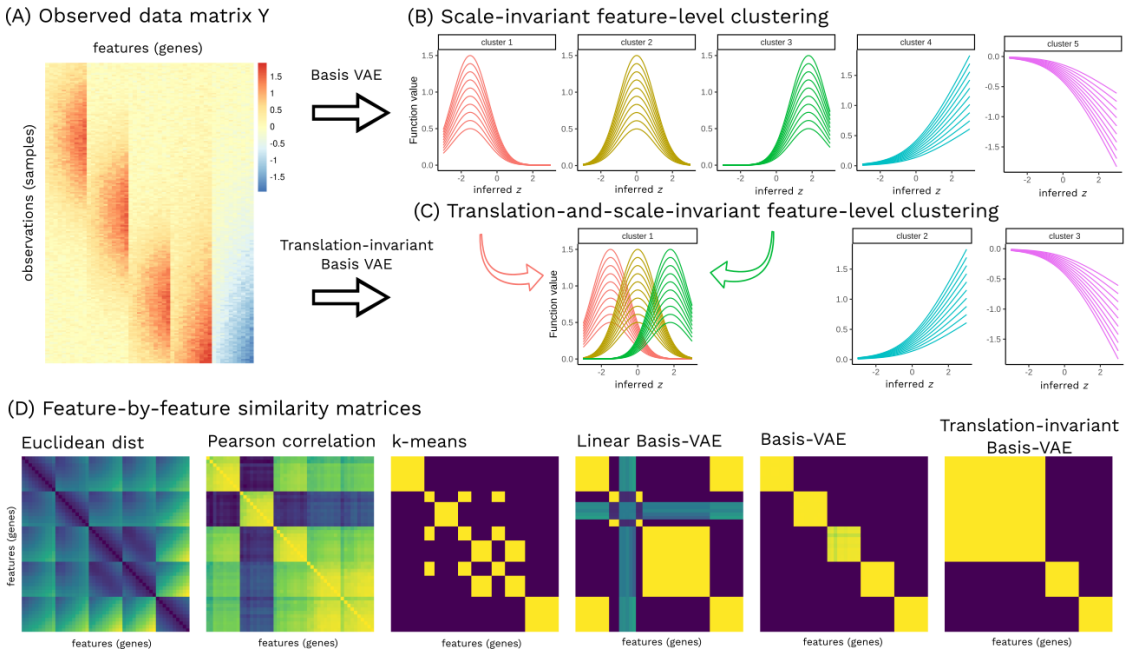


Figure 5.1: Illustration of our goal: Given a high-dimensional data matrix $\mathbf{Y}$ (heatmap in panel (A)), we want to learn a low-dimensional representation $\mathbf{z}$ and simultaneously cluster features based on the mappings $\mathbf{z} \mapsto \mathbf{Y}$ (to make it easier for the reader, we have ordered rows in panel (A) according to the inferred $\mathbf{z}$). Panels (B) and (C) show the inferred clusters: clustering (B) is scale-invariant whereas clustering (C) is additionally translation-invariant. For every feature we have plotted the inferred neural network mapping $\mathbf{z} \mapsto \mathbf{y}^{(j)}$. These have been grouped into subpanels which correspond to inferred clusters). (D) shows feature-by-feature similarity matrices either based on distances (Euclidean, Pearson correlation), or co-clustering metrics either for k-means (with $K = 5$) or co-occurrence posterior probabilities in the BasisVAE framework.

to reveal structure both within samples as well as across features. For example, common workflows in computational biology make use of clustering techniques to characterise the similarity between *features*, e.g. in order to identify groups of genes that behave similarly. Often, both dimensionality reduction and clustering are carried out consecutively in an *ad hoc* manner.

In this chapter, our goal is to develop a general purpose approach for joint dimensionality reduction and clustering of features within the Variational Autoencoder (VAE) framework. Our methodology particularly applies to any *tabular* data problem where individual features have a distinct physical meaning (e.g. a gene, a physiological measurement, the position of a joint on a moving body, etc) and there is an interest in understanding the relationship between these features. For

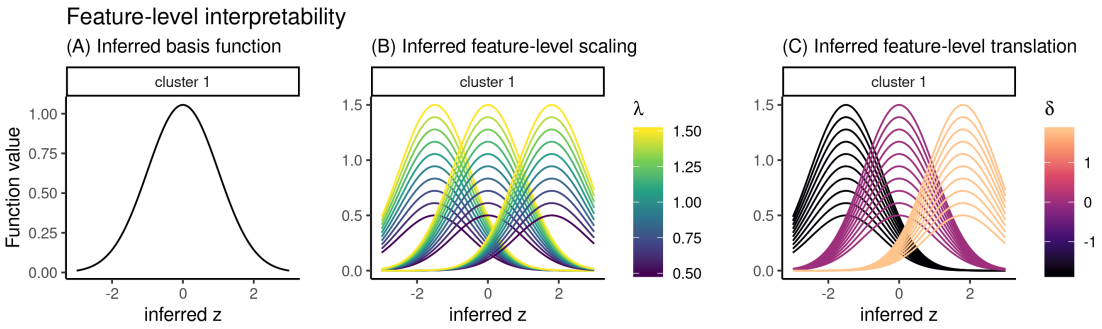


Figure 5.2: BasisVAE provides feature-level interpretability: Having identified the clusters as shown in Figure 5.1, not only can we visualise the inferred basis functions (panel (A)), but thanks to the *explicit* parameterisation we have access to the inferred (B) scaling and (C) translation parameters (λ and δ shown in colour).

example, in genomics there is interest in identifying groups of genes with similar behaviour, but the standard VAE-framework does provide a mechanism to cluster features without extensive *ad hoc* post-processing.

As a motivating example, consider the simulated gene expression data set shown in Figure 5.1(A). In the heatmap, the columns have been ordered to highlight the existence of five feature groups and the rows arranged to indicate that the observations have a certain ordering. In a real data set this ordering of rows and columns would be *unknown* and part of the statistical inference task would be to identify this latent structure, i.e. dimensionality reduction. On the other hand, there is also interest in identifying groups of similar features, as shown in Figure 5.1(B). Typically these two tasks are solved separately, whereas we propose a VAE extension to achieve this in a joint probabilistic model.

Specifically, our approach, which we refer to as BasisVAE, assumes that the features vary as functions of latent dimensions, $\mathbf{z}$, and that there exists a finite set of such (basis) functions. Hence, BasisVAE aims to discover the scale-invariant clustering of features shown in Figure 5.1(B) whilst simultaneously inferring the one-dimensional latent dimension $\mathbf{z} \in \mathbb{R}$ (in this case, $\mathbf{z}$ relates to an ordering of the observations, but in general $\mathbf{z}$ may be multi-dimensional). Moreover, in this example, three groups of features have similar shapes which are merely shifted versions of each other. So we introduce *translation-invariance* as part of the model so it would allow

these features to be further grouped (Figure 5.1C) whilst maintaining an explicit parameterisation that allows the unique properties of each feature to be maintained.

Feature-by-feature similarity measures shown in Figure 5.1(D) help us to further describe our desired output. Under Euclidean or Pearson correlation based distances, the structure of the features is not immediately obvious (note that the features displayed are ordered according to their known groupings which would ordinarily be unknown) whilst k-means clustering ($k = 5$) does not uncover this structure either. A linear model is also unsuitable due to the nonlinear relationship between the features and the latent dimensions. In comparison, BasisVAE reveals the correct structure through joint dimensionality reduction and clustering. Translation-invariance allows us to further determine whether to treat clusters 1-3 in Figure 5.1B as three distinct groups of features or as a single group where each feature has its own scale and translation parameters (Figure 5.2).

In this chapter, we endow VAEs with the capability to cluster features in the data. We propose to achieve this via introducing additional probabilistic structure within the decoder, thus combining the scalability of VAEs with a Bayesian mixture prior for the purpose of clustering features. The rest of the chapter is structured as follows. we then describe our generative model for BasisVAE and highlight a collapsed inference scheme that exhibits superior performance over simpler variational approximations. We can then demonstrate the utility of the method on a selection of synthetic and real data sets.

5.2 BasisVAE

The goal of our work is to extend the VAE to address the scientific problem of *identifying groups of features* with similar trends, i.e. clustering of features. We propose to achieve this via a mixture prior as part of the VAE decoding model. The idea is to assign every feature $\mathbf{y}^{(j)}$ to a "basis" function $f^{(k)}_{\text{basis}}$. We refer to this as a one-hot basis functional representation.

We do this by replacing the decoder network $f_{\text{decoder}} : \mathbb{R}^Q \to \mathbb{R}^P$ with a basis decoder network $f_{\text{basis}} : \mathbb{R}^Q \to \mathbb{R}^K$ (where the number of basis functions $K < P$) whose output is mapped to the data via Categorical random variables, that is

$$f^{(j)}_{\text{decoder}}(\mathbf{z}) := \sum_{k=1}^{K} w^{j,k} \cdot f^{(k)}_{\text{basis}}(\mathbf{z}) \tag{5.1}$$

where $(w^{j,1}, \ldots, w^{j,K}) \sim \text{Categorical}(\pi_1, \ldots, \pi_K)$ for every data dimension j. Note that this can be seen as a mixture model on the *features* rather than observations. The neural architecture of our proposed BasisVAE decoder has been illustrated in Figure 5.3.

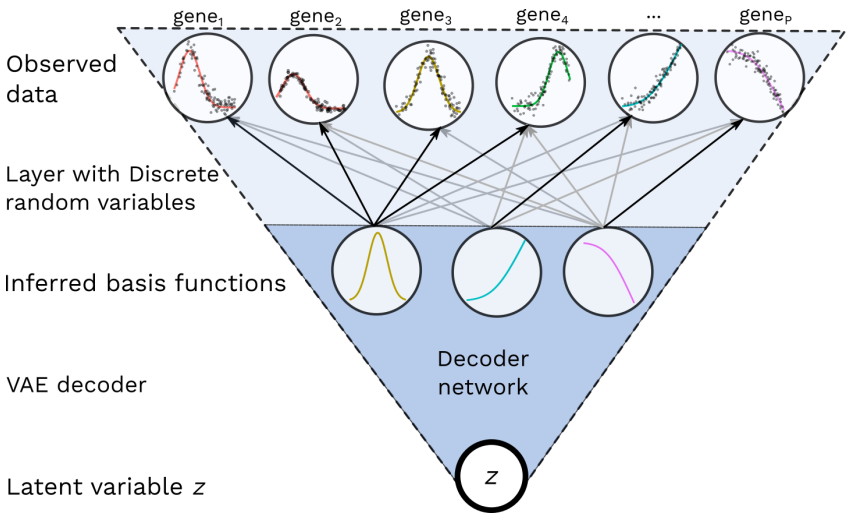


Figure 5.3: Diagram of the decoder architecture within BasisVAE. From top to bottom: observed data, the inferred basis functions, latent variable $\mathbf{z}$. It combines a typical decoder network, which is used to map $\mathbf{z}$ to the basis functions, with a specialised last layer consisting of Categorical random variables, which is used to map the basis functions to the features. The posterior distribution of these variables provides us with the implied clustering of features.

5.2.1 Translation and scale invariance

In the use of basis function representations, a question arises in how to represent two signals which have the same functional shape, but which may be shifted or scaled relative to one another. In a pure basis representation (as in equation (5.1)), each such signal would be associated with its own basis function, but this may not always be desirable. Sometimes, we would instead like the basis function to only capture the *shape* of the pattern. This can be thought of as an additional layer

of clustering in terms of functional similarity across features which is scale and translation invariant.

This behaviour could be achieved via introducing feature-specific scale and translation parameters to (5.1). For scale invariance, we introduce scaling parameters $\lambda_{j,k} \geq 0$ as follows

$$f^{(j)}_{\text{BasisVAE}}(\mathbf{z}) := \sum_{k=1}^{K} w^{j,k} \cdot \lambda_{j,k} \cdot f^{(k)}_{\text{basis}}(\mathbf{z}) \tag{5.2}$$

For additional translation invariance, we introduce $\delta_{j,k}$

$$f^{(j)}_{\substack{\text{translation-inv} \\ \text{BasisVAE}}}(\mathbf{z}) := \sum_{k=1}^{K} w^{j,k} \cdot \lambda_{j,k} \cdot f^{(k)}_{\text{basis}}(\mathbf{z} + \delta_{j,k}) \tag{5.3}$$

where for notational simplicity, we have assumed a uni-dimensional latent variable $\mathbf{z} \in \mathbb{R}$. While we have formulated a special case for $\dim(\mathbf{z}) = 1$, translation invariance can analogously be introduced for higher dimensional inputs.

Throughout the paper, we refer to (5.2) as BasisVAE (i.e. we default to scale invariance) and to (5.3) as translation-invariant BasisVAE. Note that we have restricted $\lambda_{j,k}$ to be positive. This is because in the genomics applications that are our main interest, we want to distinguish between gene upregulation and downregulation, as these correspond to different biological processes (for this reason, we want clusters 4 and 5 in Figure 5.1(B) not to be merged).

The added value from scale and translation invariance is two-fold:

- Interpretability: Having feature-specific scale and/or translation parameters lets us explicitly interpret how the features in one cluster relate to each other
- Increased model capacity: In model (5.1), a large number of basis functions K may be needed to capture the range of observed patterns, whereas (5.2) and (5.3) can obtain the same flexibility with a significantly smaller K

5.2.2 Specifying the number of basis functions

So far, we have presented BasisVAE as if the number of underlying basis functions was known. However, this is rarely the case in practice, i.e. we do not know K a priori. Ideally, we would like to over-specify the value of K, while having the property that the unnecessary extra components would be left empty.

In the mixture model framework outlined above with a Categorical$(\pi_1, \ldots, \pi_K)$ prior, this would require careful specification of the $\boldsymbol{\pi}$ values which would be problem dependent. Hierarchical Bayesian models offer a way to get around this by placing a distribution over $\boldsymbol{\pi}$. A possible construction would be as follows

$$\begin{aligned} \boldsymbol{\pi}|\boldsymbol{\alpha} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) \\ (w^{j,1}, \ldots, w^{j,K})|\boldsymbol{\pi} &\sim \text{Categorical}(\pi_1, \ldots, \pi_K) \end{aligned} \tag{5.4}$$

This hierarchical construction can be useful, because for $\alpha < 1$ values the draws from $p(\boldsymbol{\pi}|\boldsymbol{\alpha})$ will be sparse, thus providing a mechanism to *learn* an appropriately sparse $\boldsymbol{\pi}$ rather than pre-specify its value. This particular setup has been popular for finite-K Dirichlet mixture models as well as their Dirichlet Process counterparts, where conjugacy made it possible to develop efficient (collapsed) Gibbs samplers (see e.g. (Neal, 2000)). MCMC-based inference has perhaps been most popular for such mixture models, but also variational inference techniques have been derived (e.g. analytic update rules for exponential families (Blei and Jordan, 2006)).

However for non-linear latent variable models, such as VAEs, analytic updates are not available in closed form any more. Also it is not straightforward to exploit conjugacy in (5.4). This imposes a challenge: on the one hand we want to utilise the computational benefits (scalability, modularity, computational efficiency) of the VAE-framework, while on the other hand we would like to make use of the sparsity properties that are present in more classical ("pre automatic differentiation era") hierarchical Bayesian mixture models.

Existing work that has made use of mixture models in the context of VAEs[1] have either relied on using a fixed value of $\boldsymbol{\pi} = (1/K, \ldots, 1/K)$ (Dilokthanakul et al.,

[1]To our knowledge, existing work has only considered clustering observations rather than features

2016) or developed inference schemes where variational approximation is introduced on the joint space of $(\boldsymbol{\pi}, \mathbf{w})$ assuming factorisation $q(\mathbf{w}, \boldsymbol{\pi}) = q(\mathbf{w})q(\boldsymbol{\pi})$ (Nalisnick et al., 2016). The former approach is undesirable, as it has no mechanism to encourage sparsity. The latter could in principle work well, but its performance could suffer due to ignoring the (potentially strong) dependence between cluster allocations $\mathbf{w}$ and the $\boldsymbol{\pi}$.

Next, we show how our attempts to adapt these recent ideas within VAE frameworks (we refer to these as "non-collapsed inference") lead to *inefficient* inference schemes, before proposing a modern adaptation of the collapsed inference scheme within the VAE framework (which we call "collapsed").

5.2.3 Generative model and inference

The generative model for scale-and-translation-invariant BasisVAE is as follows:

$$\begin{aligned}
\mathbf{z}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
\boldsymbol{\pi}|\boldsymbol{\alpha} &\sim \text{Dirichlet}(\boldsymbol{\alpha}) \\
(w^{j,1}, \dots, w^{j,K})|\boldsymbol{\pi} &\sim \text{Categorical}(\pi_1, \dots, \pi_K) \\
y_i^{(j)}|\mathbf{w}^j, \mathbf{z}_i, \theta, \lambda, \delta &\sim \mathcal{N}\left(\sum_{k=1}^{K} w^{j,k}\lambda_{j,k} f_{\text{basis}}^{(k)}(\mathbf{z}_i + \delta_{jk}), \sigma_j^2\right)
\end{aligned} \tag{5.5}$$

where $i = 1, \dots, N$ indexes data points, $j = 1, \dots, P$ features, and k basis components, and where the decoder $f_{\text{basis}}^{(1:K)}(\cdot)$ is parameterised by a neural network with weights θ. We additionally place a prior over $\delta_{jk}, \lambda_{jk}$, i.e. $\delta_{jk} \sim \mathcal{N}(0, 1)$, $\lambda_{jk} \sim \Gamma(1, 1)$, but we will treat both λ_{jk} and δ_{jk} as point estimates and perform MAP inference over them.

We denote $\mathbf{Y} := \{\mathbf{y}_i^{(j)}\}$ and $\mathbf{Z} := \{\mathbf{z}_i\}$. Standard VAE methodology would lower bound

$$\log p(\mathbf{Y}) \geq \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i)}[\log p_\theta(\mathbf{y}_i|\mathbf{z}_i)] - \text{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i)||p(\mathbf{z}_i))$$

However, unlike for standard VAE, for our model $\log p_\theta(\mathbf{y}_i|\mathbf{z}_i)$ as given in (5.5) is intractable due to the need to integrate over $\mathbf{w}$ and $\boldsymbol{\pi}$.

5.2.3.1 Non-collapsed inference

One approach would be to introduce an approximate posterior $q(\mathbf{w}, \boldsymbol{\pi}) = \prod_{j=1}^{P} q(\mathbf{w}^j) q(\boldsymbol{\pi})$, for example this type of approximation used in (Nalisnick et al., 2016). E.g. one can choose variational families $q(\mathbf{w}^j) = \text{Categorical}(\phi_{j,1}, \ldots, \phi_{j,K})$ and $q(\boldsymbol{\pi}) = \text{Dirichlet}(\boldsymbol{\psi})$, where reparameterisation trick for the Dirichlet can for example be achieved as outlined in (Figurnov et al., 2018). Assuming mean-field across $\{\mathbf{z}_i\}, \{\mathbf{w}^j\}, \boldsymbol{\pi}$, we get the ELBO

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i)} \mathbb{E}_{q(\mathbf{w})} \log p_\theta(\mathbf{y}_i|\mathbf{z}_i, \mathbf{w}) - \text{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i)||p(\mathbf{z}_i)) \\ &- \text{KL}\left(\prod_{j=1}^{P} q(\mathbf{w}^j) q(\boldsymbol{\pi}) \;||\; \prod_{j=1}^{P} p(\mathbf{w}^j|\boldsymbol{\pi}) p(\boldsymbol{\pi})\right)\end{aligned}$$

We refer to this approach as *non-collapsed* inference, and later demonstrate how this inference scheme can get stuck in local modes, resulting in undesirable behaviour.

5.2.3.2 Collapsed inference

Given the success of earlier collapsed inference schemes, e.g. collapsed Gibbs samplers (Neal, 2000) or collapsed VB schemes (Teh et al., 2007; Kurihara et al., 2007; Hensman et al., 2012; Hensman et al., 2015), we hope to improve our previously decribed non-collapsed inference by marginalising out $\boldsymbol{\pi}$. If we adapted the approach from (Teh et al., 2007), further approximations for evaluating the ELBO would be needed. Instead, we follow the approach of (Hensman et al., 2012) where marginalisation is applied *after* (rather than before) making a variational approximation.

Now, following the strategy proposed by Hensman et al. (2012), we lower bound the term $\log p_\theta(\mathbf{y}|\mathbf{z})$, by first introducing an approximate posterior

$$q(\mathbf{w}) = \prod_{j=1}^{P} q(\mathbf{w}^j) = \prod_{j=1}^{P} \text{Categorical}(\phi_{j,1}, \ldots, \phi_{j,K})$$

and then collapsing $\boldsymbol{\pi}$ from the bound without any further approximations. We derive this as follows, by first investigating the term $\log p_\theta(\mathbf{y}_i|\mathbf{z}_i)$. Knowing that

$$\begin{aligned}\log p_\theta(\mathbf{y}_i|\mathbf{z}_i,\boldsymbol{\pi}) &= \log \int p_\theta(\mathbf{y}_i|\mathbf{z}_i,\boldsymbol{\pi},\mathbf{w})p(\mathbf{w})d\mathbf{w} \\ &\geq \int q(\mathbf{w}) \log \frac{p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{w})p(\mathbf{w}|\boldsymbol{\pi})}{q(\mathbf{w})} d\mathbf{w} \\ &= \mathbb{E}_{q(\mathbf{w})} \log \frac{p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{w})p(\mathbf{w}|\boldsymbol{\pi})}{q(\mathbf{w})}\end{aligned} \tag{5.6}$$

we can now lower bound

$$\begin{aligned}\log p_\theta(\mathbf{y}_i|\mathbf{z}_i) &= \log \int p_\theta(\mathbf{y}_i|\mathbf{z}_i,\boldsymbol{\pi})p(\boldsymbol{\pi})d\pi \\ &\geq \log \int \exp\left(\mathbb{E}_{q(\mathbf{w})} \log \frac{p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{w})p(\mathbf{w}|\boldsymbol{\pi})}{q(\mathbf{w})}\right) p(\boldsymbol{\pi})d\pi \\ &= \mathbb{E}_{q(\mathbf{w})} \log p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{w}) + \log \int \exp\left(\mathbb{E}_{q(\mathbf{w})} p(\mathbf{w}|\boldsymbol{\pi})\right) p(\boldsymbol{\pi})d\pi - \mathbb{E}_{q(\mathbf{w})} \log q(\mathbf{w})\end{aligned}$$

which results in the overall lower bound on the log marginal likelihood $\log p(\mathbf{Y})$ as follows

$$\mathcal{L} := \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i)} \mathbb{E}_{q(\mathbf{w})} \log p_\theta(\mathbf{y}_i|\mathbf{z}_i,\mathbf{w}) \tag{5.7}$$

$$+ \log \int \exp\left(\mathbb{E}_{q(\mathbf{w})} p(\mathbf{w}|\boldsymbol{\pi})\right) p(\boldsymbol{\pi})d\pi \tag{5.8}$$

$$- \mathbb{E}_{q(\mathbf{w})} \log q(\mathbf{w}) \tag{5.9}$$

$$- \sum_{i=1}^{N} \mathrm{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i)||p(\mathbf{z}_i)) \tag{5.10}$$

where now all terms can be analytically calculated. Indeed:

- For the first term (5.7), we use a combination of reparameterisation and analytic marginalisation, the former for sampling $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{y}_i)$ and the latter for empirically summing over $q(\mathbf{w})$, i.e.

$$\sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{z}_i|\mathbf{y}_i)} \sum_{j=1}^{P} \sum_{k=1}^{K} \phi_{j,k} \log \mathcal{N}(y_i^{(j)}|\lambda_{j,k} f_{\text{basis}}^{(k)}(\mathbf{z}_i + \delta_{jk}),\ \sigma_j^2).$$

- The second term (5.8) can be derived analytically, resulting in the expression

$$\log \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \int \prod_{k=1}^{K} \pi_k^{n_k+\alpha_k-1} d\boldsymbol{\pi}$$

where $n_k := \sum_{j=1}^{P} \phi_{j,k}$, and where the integral has a closed form, thus resulting in the term

$$\log \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \frac{\prod_k \Gamma(n_k + \alpha_k)}{\Gamma(\sum_k n_k + \alpha_k)}.$$

- Also the third term (5.9) has a simple form as follows

$$\sum_{j=1}^{P} \sum_{k=1}^{K} \phi_{j,k} \log \phi_{j,k}.$$

Finally, as previously mentioned, we propose to perform MAP inference over λ_{jk} and δ_{jk}, encouraging δ values to be centered around 0 and λ values centered around 1, so the final optimisation objective will incorporate additional penalties

$$\mathcal{L} + \sum_{j=1}^{P} \sum_{k=1}^{K} \log \mathcal{N}(\delta_{jk}|0, 1) + \log \Gamma(\lambda_{jk}|1, 1).$$

As a result, we have derived an inference scheme which can be easily implemented as a VAE extension, thus making it possible to leverage the advantages of deep learning software. From the computational efficiency perspective, the ELBO for the standard VAE involves summing over observations and features, whereas our ELBO includes an additional sum over cluster components $k = 1, \ldots, K$. Thus the additional scaling w.r.t. K is linear (which, we note, can be efficiently implemented by introducing an extra dimension to tensors).

Adaptation of ELBO for large data sets: For large high-dimensional data sets, the lower bound derived above will be dominated by the data log-likelihood in equation (5.7). For large N, also the KL-term (5.10) might be large. As a result, the clustering prior that we have introduced will implicitly become relatively less important when N and P increase. While the property that for large data sets the likelihood will dominate the prior is inherent to Bayesian models, it may not always be desirable, especially for mis-specified models.

In practice, one way to alleviate this problem where the likelihood starts to dominate is via introducing weights that either downweigh the likelihood or upweigh the prior. For example, β-VAE modifies the usual VAE lower bound by scaling the KL term by a constant $\beta > 0$ (Higgins et al., 2017). Even though the resulting

expression is not a lower bound on the original log marginal likelihood any more, it is closely connected to an ELBO on an alternative formulation with an annealed prior $p(\mathbf{z})^\beta / \int p(\mathbf{z})^\beta d\mathbf{z}$ (Hoffman et al., 2017; Mathieu et al., 2019). Analogously, we propose to achieve a similar effect, by introducing β as part of the following objective

$$\mathcal{L}_\beta := \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_i|\mathbf{y}_i)} \mathbb{E}_{q(\mathbf{w})} \log p_\theta(\mathbf{y}_i|\mathbf{z}_i, \mathbf{w}) + \\ + \beta \left(\log \int \exp\left(\mathbb{E}_{q(\mathbf{w})} p(\mathbf{w}|\boldsymbol{\pi}) \right) p(\boldsymbol{\pi}) d\pi - \mathbb{E}_{q(\mathbf{w})} \log q(\mathbf{w}) - \sum_{i=1}^{N} \mathrm{KL}(q_\phi(\mathbf{z}_i|\mathbf{y}_i)||p(\mathbf{z}_i)) \right)$$

where $\beta = 1$ corresponds to the original ELBO, but we may choose to select $\beta > 1$ in order to increase the relative importance of the sparse clustering prior for large-scale applications. In our experiments, for moderately sized synthetic data we used $\beta = 1$, whereas for single cell gene expression data we used $\beta = 20$.

5.3 Related work

To our knowledge, the problem of *clustering features* in the data has not been previously addressed in any existing VAE-framework.

Probabilistic methods for curve clustering in the context of regression models have been proposed before. E.g. Chudova et al. (2003) propose a method for translation-invariant curve clustering, however they assume fixed inputs on an equi-spaced time grid. Gaffney and Smyth (2005) derive an EM-algorithm for curve clustering and alignment, assuming that curves are parameterised using either polynomial regression or cubic splines (for the latter, the M-step cannot be carried out exactly). This approach assumes fixed inputs and does not scale to large high-dimensional data sets. Palla et al. (2012) propose a Bayesian non-parametric *linear* latent variable model. Thus, it could be seen as a special case of BasisVAE with a *linear* decoder and infinite K. However, the linearity assumption is too restrictive for the applications we consider in this paper.

Campbell and Yau (2018a) propose a specific translation-invariant framework for single-cell genomics, however it captures only two pre-defined types of parametric

mappings: sigmoidal and transient patterns. This is much more restrictive compared to BasisVAE that can *learn* the dynamics in an unsupervised way.

5.4 Experiments

It is challenging to evaluate the performance of clustering methods on real applications, as the ground truth cluster labels are rarely known (and even if labels exist, there may always be more fine-grained subgroups). For this reason, we first evaluate the performance of BasisVAE in a controlled setting, before considering the semi-synthetic clustering of motion capture data and finally demonstrating the utility of our approach for the challenging analysis of large-scale single-cell gene expression data.

5.4.1 Toy data

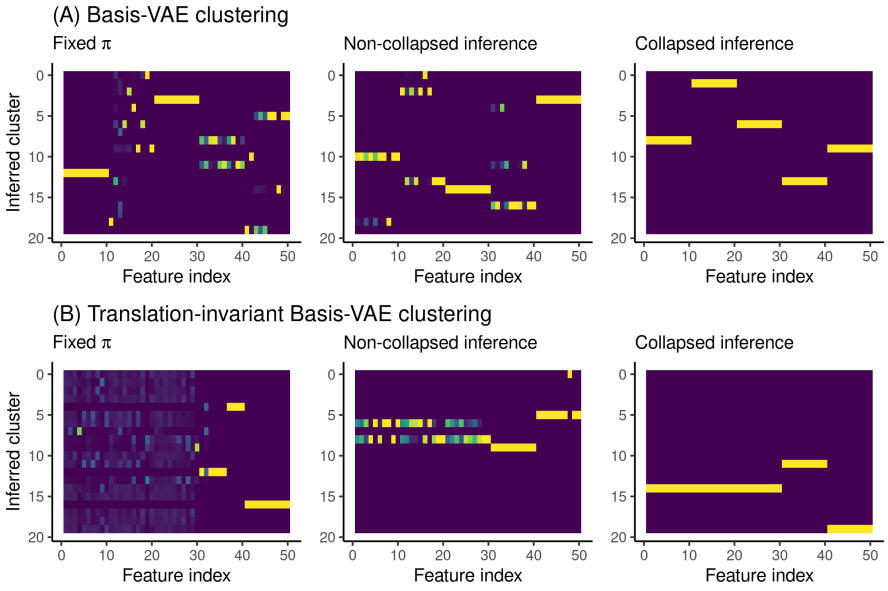


Figure 5.4: Comparison of inference techniques for $q(\mathbf{w}, \boldsymbol{\pi})$, for both (A) BasisVAE and (B) its translation-invariant version, using the synthetic data as in Figure 5.1 (consisting of 5 groups with 10 features each). The heatmaps show posterior probabilities for cluster allocations, i.e. elements of $q(\mathbf{w})$. Both "fixed $\boldsymbol{\pi}$" and "non-collapsed" inference (left and middle panels) result in suboptimal and non-sparse cluster assignments, whereas collapsed inference correctly identifies the underlying ((A) $K = 5$ and (B) $K = 3$) clusters.

We first consider the synthetic data shown in Figure 5.1 with 5 distinct groups of features (with 10 features each, $P = 5 \times 10$) as a function of latent $\mathbf{z} \in \mathbb{R}$. The data

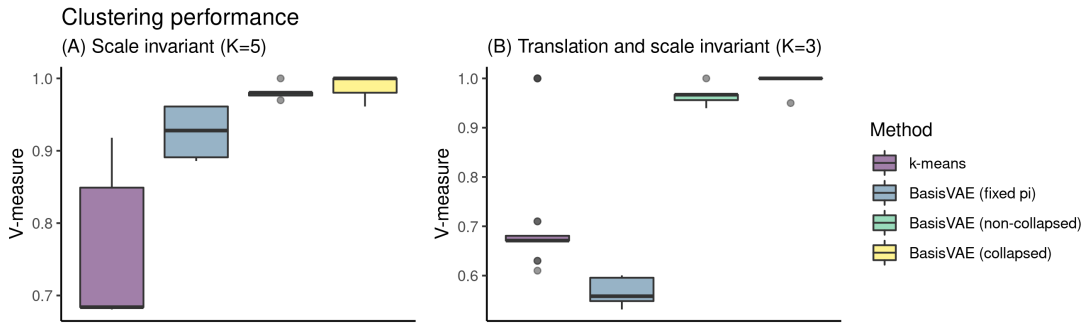


Figure 5.5: Using the V-measure to quantify the clustering performance of k-means and different versions of BasisVAE (with different inference techniques for $q(\mathbf{w}, \boldsymbol{\pi})$) on the synthetic data from Figure 5.1 . Results are shown for both (A) scale invariant and (B) translation and scale invariant setting. For k-means, we used the true number of clusters, i.e. (A) $K = 5$ and (B) $K = 3$. For BasisVAE, we used an overspecified $K = 20$.

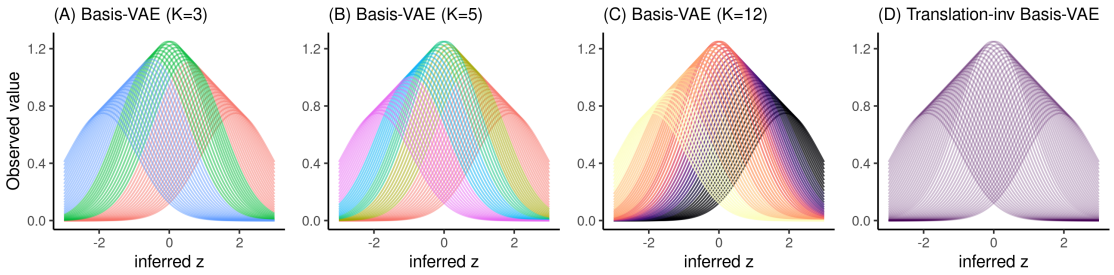


Figure 5.6: A large number of basis functions is required to capture the diverse range of mappings shown, whereas translation-invariant BasisVAE can represent this with a single basis function. BasisVAE with (A) K=3, (B) K=5, and (C) K=12 cluster components (shown in colour coding) indicate that an even larger K would be needed to capture subtle differences between the mappings, whereas translation-invariant BasisVAE (D) has inferred only a single cluster component.

generative process was designed in order to mimic the patterns we expect to see in real gene expression data: first three groups of features exhibit a transient trend (where gene expression spikes for a certain range of z values), whereas the fourth cluster exhibits gene up-regulation and fifth one down-regulation. The BasisVAE correctly infers five clusters (Figure 5.1B), whereas its translation-invariant version has merged the first three clusters together (Figure 5.1C).

Note that these results were obtained under our collapsed inference scheme. This would not have been possible with simpler approaches such as the "fixed $\boldsymbol{\pi}$" and "non-collapsed" inference, as shown in Figure 5.4, which result in cluster allocations that are less sparse than the one inferred by "collapsed" inference (all comparisons

were carried out with K fixed to $K = 20$ and $\alpha = 0.1$). "Fixed $\boldsymbol{\pi}$" has failed likely due to the reason that $\pi_k = 1/K$ does not encourage sparse solutions, whereas "non-collapsed" inference is more prone to getting stuck in local modes due to the dependence between $\mathbf{w}$ and $\boldsymbol{\pi}$. In Figure 5.5, we have provided a quantitative comparison of clustering performance using the V-measure (Rosenberg and Hirschberg, 2007) over 10 random restarts, comparing k-means (using the true number of clusters) and various inference schemes for BasisVAE.

Various widely used methods for clustering gene expression data rely on the feature-by-feature similarity matrix. As illustrated in Figure 5.1(D), these do not accurately capture the underlying cluster structure, and do not provide a way to naturally incorporate scale and translation invariance. Also the linear version of BasisVAE, which can be seen as a particular implementation of (Palla et al., 2012), does not provide the correct clustering either.

Next, we demonstrate the capabilities of BasisVAE to capture translation-invariance in a setting where we observe features spanning a range of δ values (Figure 5.6). The non-translation-invariant BasisVAE clustering behaves intuitively: when we use a small $K \in \{3, 5, 12\}$ (panels A-C), then features that have similar δ values, are clustered together (cluster allocations shown in colour). This may be sufficient for certain applications, but there is no way to quantify how all the clusters relate to one another. This capability is present in the translation-invariant BasisVAE, which identifies a single cluster (Figure 5.6(D)) and relates all features within the cluster to one another explicitly via inferred δ.

5.4.2 Motion capture data

Next we investigate the behaviour of our model in a semi-controlled setting. That is, we use CMU Motion Capture database, which consists of short segments of various activities over a repeated number of trials. As a result, even if we focus on the same activity performed by the same subject, the measurements (locations of joints) are not *synchronised* between repeated trials. We note that this alignment problem has a particular structure which is inherently simpler than the structure in

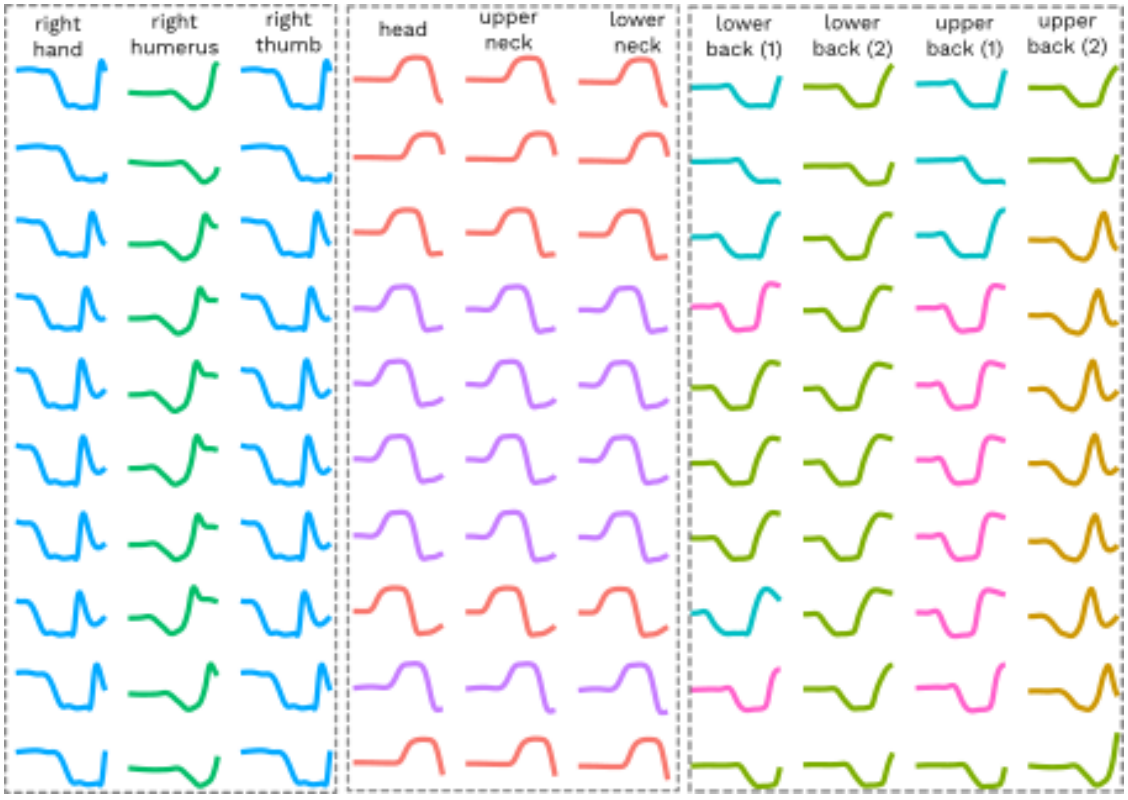


Figure 5.7: BasisVAE on motion capture data. The inferred mappings together with the clustering (shown in colour), shown over 10 repeated trials (in rows) and selected features (in columns). Features have been retrospectively grouped into 1) right-hand, 2) head-and-neck, and 3) lower-and-upper-back related features.

the genomics data in the next section. This is for various reasons: First, unlike for the genomics application, for motion capture time is observed. Second, we know that we need to align measurements only *between* trials, but not within a trial (if measurements for one joint are shifted by δ, it is likely the case for other joints as well).

Here, we focus on the 10 repeated trials of the activity "golf swing". For the purpose of demonstrating the capabilities of translation-invariant BasisVAE, we treat 10 trials of 59-dimensional measurements as a high-dimensional data set with $P = 590$ dimensions without *a priori* known structure. Our goal is three-fold: First, we hope to uncover clusters that are shared *across* trials, e.g. we expect the "right hand" movements to follow the same dynamics over all 10 trials. Second, we hope to explicitly align the movements across trials. Third, we would like to find *shared*

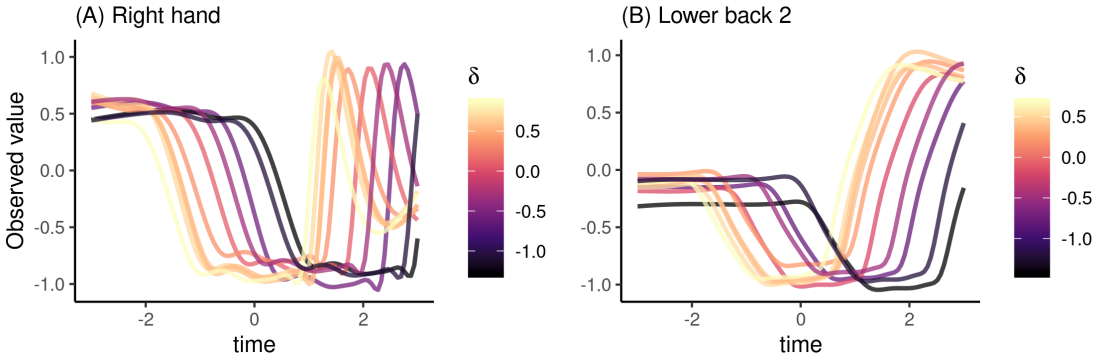


Figure 5.8: BasisVAE lets us explicitly align features in motion capture data, demonstrated on features "right hand" and "lower back (2)" (coloured based on the inferred δ values).

dynamics between the marker positions, e.g. two nearby joints may exhibit similar movement.

We have visualised results in Figure 5.7 for a subset of features (in columns) and all 10 trials (in rows), showing the inferred mappings (lines) and the respective cluster allocations (in colour). The features can be grouped into 1) right-hand , 2) head-and-neck, and 3) lower-and-upper-back related movement. Note that for many features, all 10 trials have been assigned to the same cluster, whereas sometimes there are subtle differences between the trials (e.g. for head and neck, the "bump" for some trials lasts slightly longer than for others).

Our model provides interpretability that is not present in alternative methods. We can align features between trials (inferred δ values shown in Figure 5.8), and identify groups of similar features. E.g. "right hand" and "right thumb" have been assigned the same cluster, whereas "right humerus" exhibits a different dynamic.

5.4.3 Single cell Spermatogenesis data

Recently, there has been a lot of interest in single-cell genomics, where gene expression is measured on the level of individual cells. This results in large data sets, both in the number of samples (i.e. cells) as well as the number of features (i.e. genes). The ability to quantify individual cells has led to a number of studies where the aim is to characterise *cellular differentiation* - the process by which a

cell changes from one cell type to another. As the technology lets us measure every cell only once, we do not have access to the true underlying time, how far along the differentiation trajectory every cell is. Those quantities are referred to as *pseudotimes*, and are treated as one-dimensional latent variables $\mathbf{z}$.

The main scientific interest in these studies lies in getting insights into the biological processes that drive this latent dimension, i.e. identifying groups of genes whose expression changes with $\mathbf{z}$, and characterising those changes. In this section, we demonstrate the utility of BasisVAE for such purposes, i.e. for joint dimensionality reduction and clustering of genes. Here, we re-analyse a publicly available data set from a recently published study (Ernst et al., 2019), where the authors used single-cell RNA-Sequencing to profile mouse spermatogenesis. However, our BasisVAE methodology applies directly to a variety of single-cell gene expression analyses such as (Furuyama et al., 2019; McFaline-Figueroa et al., 2019; Trapnell et al., 2014), letting us re-analyse the data using a principled joint probabilistic approach.

Here, we replace the Gaussian likelihood in the decoder with a zero-inflated negative binomial (ZINB) likelihood, which is a common approach due to the data being non-negative-valued and exhibiting zero-inflation (Lopez et al., 2018). Introducing notation for the negative binomial distribution as follows

$$y_i^{(j)} \sim \text{NB}(\mu_{ij}, \phi_{ij}) \iff p_{NB}(y_i^{(j)}) = \frac{\Gamma(y_i^{(j)} + \phi_{ij})}{\Gamma(y_i^{(j)} + 1)\Gamma(\phi_{ij})} \left(\frac{\phi_{ij}}{\phi_{ij} + \mu_{ij}}\right)^{\phi_{ij}} \left(\frac{\mu_{ij}}{\phi_{ij} + \mu_{ij}}\right)^{y_i^{(j)}}$$

we now outline the introduce the ZINB likelihood, a mixture of a negative binomial distribution and a point mass at zero:

$$y_i^{1:P} | l_i, z_i, \phi \sim \pi_{i,j}\delta_0 + (1 - \pi_{i,j})\text{NB}(l_i \cdot \text{softmax}(f_\theta^{1:P}(z_i)), \phi^{1:P}).$$

Here $\phi^{1:P}$ are the inverse dispersion parameters of the negative binomial distribution, l_i are sample-specific scaling constants, $f_\theta^{1:P}$ is a neural network (analogously to the Gaussian likelihood), and $\pi_{i,j}$ represent the so-called dropout probabilities, which we parameterise via another neural network as in (Lopez et al., 2018). For clustering purposes, our main interest lies in the mean function, therefore we apply the functional decomposition only across $f_\theta^{1:P}$.

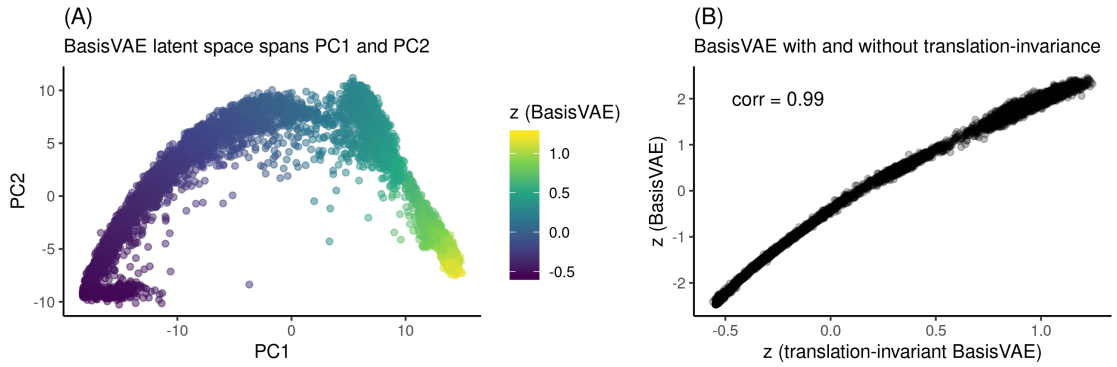


Figure 5.9: (A) BasisVAE has inferred a latent $\mathbf{z} \in \mathbb{R}$ that captures a trajectory in the (PC1, PC2) space. (B) BasisVAE and its translation-invariant version infer a highly similar latent space (the respective inferred latent coordinates are highly correlated).

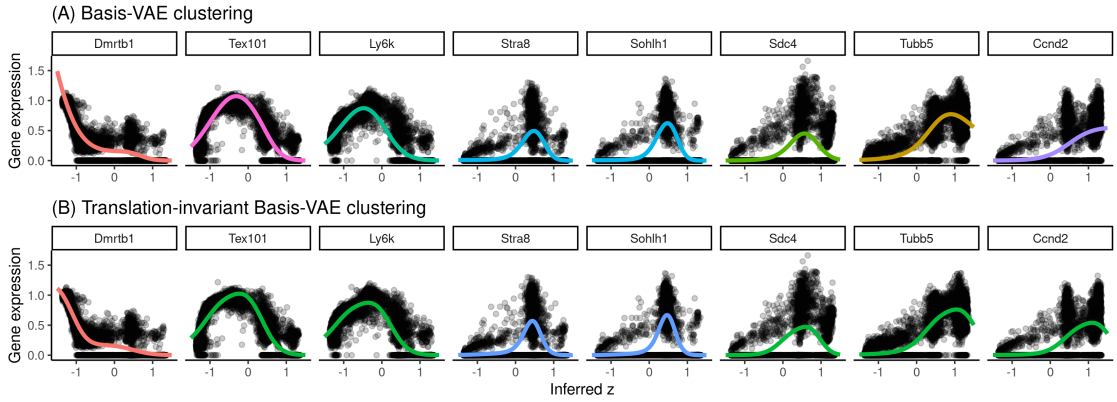


Figure 5.10: BasisVAE has discovered structure in the large-scale single-cell spermatogenesis data. For selected genes (whose peak expression along the z-axis varies from left to right, i.e. from Dmrtb1 to Ccnd2), we have shown how (A) BasisVAE has identified a number of clusters with distinct gene expression behaviour (clusters shown in colour), whereas (B) its translation-invariant version has grouped genes with similar profiles together (clusters shown in colour), thus aiding further interpretability.

Following (Ernst et al., 2019), we pick a subset of samples ($N = 8,509$) and $P = 7,000$ most variable genes, and fit BasisVAE without and with translation-invariance (we used $K = 50$ clusters). In Figure 5.9(A), we illustrate how the inferred one-dimensional latent $\mathbf{z}$ relates to the first two principal components (PC1, PC2) of the data. This $\mathbf{z}$ constitutes a trajectory in the (PC1, PC2) space, capturing the developmental trajectory of differentiation.

Manually interpreting individual genes in such a large-scale setting is infeasible, however the inferred BasisVAE clusters makes interpretation easier. In Figure 5.10 we display the genes that were highlighted in the analysis of (Ernst et al., 2019)

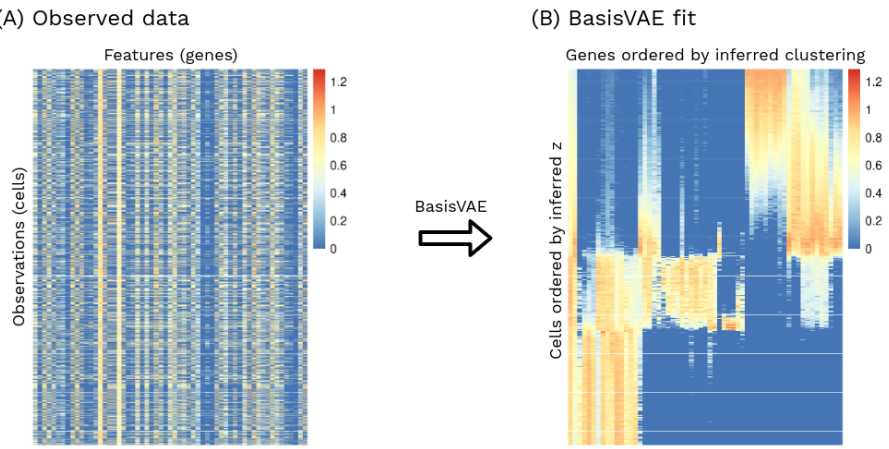


Figure 5.11: BasisVAE has uncovered structure in the spermatogenesis data. (A) Heatmap of the observed data (for a subset of genes), and (B) the same data with reordered rows (now ordered by the inferred z) and reordered columns (according to the inferred the clusters).

together with the inferred mappings and cluster allocations. Note that in the scale-invariant version, genes exhibiting the same shape but different shift were allocated different clusters, whereas e.g. genes *Stra8* and *Sohlh1* were identified as exhibiting the same behaviour. This is different for the translation-invariant model, where e.g. genes *Tex101, Ly6k, Sdc4, Tubb5, Ccnd2* were all assigned to the same cluster, so not only do we know that they have the same shape, but we can also interpret their relative shift by inspecting the respective δ values.

While the inferred clustering structure (e.g. where the behaviour of gene might be delayed relative to another, but exhibiting the same shape) does not imply any biological functional relatedness, BasisVAE methodology can add to the exploratory toolkit for investigators to explore and rank such potential relationships between genes explicitly rather than retrospectively through *ad hoc* approaches that are common in genomics.

5.5 Discussion

In this chapter, have proposed a joint model for dimensionality reduction and clustering of features, implemented in the VAE framework. Specifically, we have embedded a hierarchical Bayesian clustering prior within the decoder, and demonstrated how

collapsed variational inference can identify sparse solutions, thus making it possible to over-specify the number of clusters K. For improved interpretability, we have introduced explicit parameters for scale and translation invariance on the level of individual features. As a result, with a relatively small modification to the VAE objective function, we have obtained an approach for clustering features, while maintaining the *scalability* of the VAE framework.

Next, we discuss whether the added flexibility due to translation invariance can cause any potential problems w.r.t. being a well-defined model. We conclude with discussing future work.

5.5.1 On well-definedness of translation-invariant BasisVAE

As a non-linear latent variable model, even the standard (C)VAE exhibits multiple modes in the sense that different latent configurations $\{\mathbf{z}_i\}_{i=1}^N$ can give rise to the same likelihood. Even in the probabilistic PCA which is a linear model, different rotations of the latent space are indistinguishable in terms of the likelihood. Having introduced additional parameters δ_{jk}, it is natural to ask whether the BasisVAE model is well-defined and whether it is identifiable with respect to δ_{jk} parameters.

Here we assume $\mathbf{z} \in \mathbb{R}$. First, we note that without any constraints or priors on delta, our model is unidentifiable in two aspects:

- The inputs $\mathbf{z} + \delta_{jk}$ can both be shifted by an arbitrary constant, i.e. $\mathbf{z} + \delta_{jk} = (\mathbf{z} + a_k) + (\delta_{jk} - a_k)$ for any $a_k \in \mathbb{R}$
- Suppose latent variables z are fixed, then for any $f^{(j)}$ and δ_{jk}, there exist $f^{*(j)}$ and δ^*_{jk} such that $f^{(j)}(\mathbf{z} + \delta_{jk}) = f^{*(j)}(\mathbf{z} + \delta^*_{jk})$

However, having placed a prior over delta, arbitrary reparameterisations as outlined above are not equally likely any more. For example, let us consider a thought experiment where every feature is assigned to its own basis function: even though for every feature j there exist multiple pairs $(f^{(j)}, \delta_{jk})$ of functional representations that are indistinguishable $f^{(j)}(\mathbf{z} + \delta_{jk})$ in terms of their outputs, in this particular scenario the solution $f^{*(j)}$ where $\delta^*_{jk} = 0$ is preferred over any other combination.

This is because indistinguishable functional representations result all have the same log-likelihood, and thus the configuration with $\delta_{jk} = 0$ achieves the highest ELBO in this scenario. Thus, the prior $p(\delta_{jk})$ alleviates the problem with unidentifiability.

We also note that the existence of these multiple modes would not be a problem from the interpretability perspective, as for this purpose it is the relative difference between δ_{jk} values that is of interest, not their absolute values.

Potential pathological scenario: Here we discuss a corner case where the translation-invariant BasisVAE formulation could potentially be ill-defined due to the added flexibility by δ_{jk}. Even though we have not experienced this in practice, we first describe this scenario, and then propose a solution how this pathological case can be avoided by a simple modification.

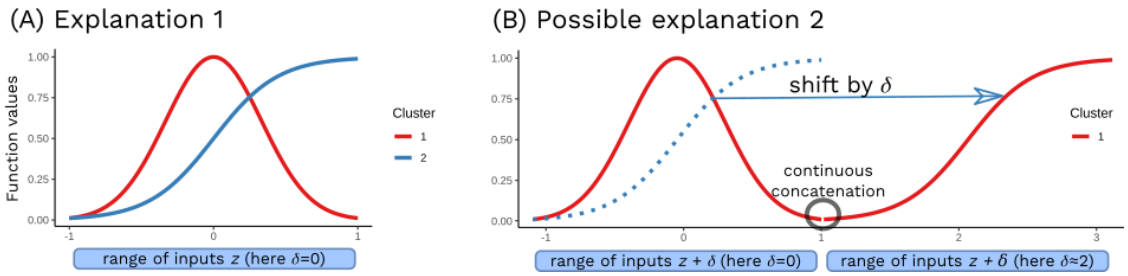


Figure 5.12: Illustration of an imaginary pathological scenario, where we would expect the two observed features (as shown in (A)) to be assigned to two separate clusters. Panel (B) describes a pathological case, where a large δ value for one feature allows the two curves to be joined and thus the two features to be assigned to the same cluster.

The idea behind the pathological scenario has been illustrated in Figure 5.12(A, B): when inputs $\mathbf{z}$ are in some interval $[a, b]$, then one can e.g. pick $\delta_{jk} = 0$ for one feature, $\delta_{jk} > (b - a)$ for another feature, and concatenate the two into one basis function (we note that the concatenation of the two functions must be continuous).

While we do not see this scenario as a probable risk in reality, we are able to avoid this potential behaviour altogether by introducing a constraint $|\delta_{jk} - \delta_{j'k'}| < (b - a)$, which can e.g. be easily implemented by explicitly restricting the range of all δ_{jk} values.

5.5.2 Future work

While BasisVAE methodology applies for any dimensionality of $\mathbf{z}$, the number of clusters needed to cover all different modes of behaviour will grow quickly when the input dimension $\dim(\mathbf{z})$ increases. Thus, in our future work, we would like to consider a *decomposable* extension to the BasisVAE modelling framework for multivariate inputs $\mathbf{z}$ and potentially covariates $\mathbf{c}$, where e.g. we would separate out the additive components f_z and f_c and obtain multiple sets of clusters, based on every input coordinate separately (and potentially higher order interactions). Formulating and implementing this would require combining ideas from ND and BasisVAE.

6

Conclusions

Contents

6.1 Summary of contributions 123
6.2 Future directions 124

In this thesis, we have developed probabilistic machine learning methods for dimensionality reduction with the goal to enable interpretability on the level of observed features. Having synthesised ideas from classical statistics and recent advances in machine learning, we have constructed a set of latent variable models that combine the flexibility of black box non-linear modelling with feature-level explainability.

We hope that this will open up the potential of modern non-linear models for biomedical applications. For example, in genomics currently most analyses rely on either linear methods for dimensionality reduction or *ad hoc* visualisation approaches such as t-SNE or UMAP. While the former are easier to interpret, the linearity assumption can be too restrictive. The latter methods, on the other hand, do not provide us with an explicit mapping between the low-dimensional representations and the high-dimensional data.

6.1 Summary of contributions

In this thesis, we have built upon and extended two classes of non-linear latent variable models, Gaussian Process Latent Variable Models (GPLVMs) and Variational Autoencoders (VAEs). The motivation behind our work has been to enable the use of such non-linear models for tabular data applications, e.g. for exploratory data analysis in genomics, where we want to be able to interpret how the low-dimensional representations relate to the feature space. We review our main contributions below.

The main goal of latent variable modelling is to reveal meaningful low-dimensional structure. In Chapter 2, we investigated the scenario where these low-dimensional representations in the data are confounded by known (covariate) information. Specifically, we proposed an extension of the GPLVM, called the c-GPLVM, which would provide a mechanism to incorporate covariates as part of the dimensionality reduction, letting us learn covariate-adjusted latent representations.

However, the learnt low-dimensional representations do not usually provide sufficient insight on their own, when explored in isolation from the observed data itself. For this reason, linear dimensionality reduction methods (such as probabilistic PCA) are often preferred, as they provide an *explanation* how the latent variables relate to the original features. In Chapter 3, we investigated how to embed feature-level interpretability within the c-GPLVM framework. Specifically, we proposed to enable this through a construction that combines Gaussian Processes with the notion of functional ANOVA decomposition from classical statistics. Using a structured kernel decomposition, it let us attribute feature-level variation to either latent variables, fixed covariates, or their interactions.

The motivation behind Chapter 4 was to overcome the scalability limitations of Gaussian Process based models which struggle to scale to large high-dimensional data sets. This is the case even when employing the variational inducing point framework which alleviates the scalability w.r.t. the number of observations, but not w.r.t. data dimensionality. Thus, here we instead focused on the VAE framework which employs neural networks for scalable latent variable modelling. As opposed

to standard predictive applications where it is sufficient to treat the VAE as a black box, here we constructed a decomposable decoder network to enable feature-level explainability. Similarly to Chapter 3, we chose to rely on the functional ANOVA decompositions, but here we had to take a different approach to enforcing integral constraints that are crucial for the identifiability of the decomposition. Specifically, we proposed to combine quadrature and a constrained optimisation approach to achieve this. The resulting *Neural Decomposition* modelling framework maintains the scalability of the VAE framework, whilst providing feature-level insights into the otherwise black box model.

In Chapter 5, we investigated a different perspective to feature-level interpretability. There our goal was to introduce a mechanism that would explicitly identify groups of similar features as part of the VAE. For this purpose, we developed BasisVAE: a modelling framework for joint dimensionality reduction and clustering of features, additionally allowing for scale and translation invariance. We achieved this by embedding a Bayesian clustering prior within the decoding structure of the VAE.

6.2 Future directions

The methodology that we have developed in this thesis implicitly assumes that we are modelling a single data set, potentially with some external covariate information. Here, we discuss two important scenarios when this is not the case: first, when we have access to multiple data modalities (such as multi-omics data), and second, sharing information across studies. We see both of these scenarios as important future research directions.

In genomics, it is becoming increasingly popular to perform multi-omics studies, letting us characterise multiple biological layers jointly. For example, this can include gene expression measurements, methylation data, somatic mutations etc. Joint integrative analysis approaches are desirable, as they can reveal latent variables that are shared between modalities as well as those that are specific to a particular data type. However, typical modelling approaches used for multi-omics data are linear

(Argelaguet et al., 2018). While VAEs have been extended to the multimodal setting (Wu and Goodman, 2018; Ainsworth et al., 2018), we believe it would be a valuable research direction to extend our methodology on feature-level interpretability from Chapters 4 and 5 to multiple data modalities. For example, a multimodal extension of neural decomposition could aid interpretability by associating latent variables with a crossmodal feature-level variance decomposition.

In the biomedical domain, most data sets have relatively small sample sizes, often focusing on studying a specific experimental condition or a particular disease subtype. However, even though every subtype is unique in some sense, we expect many of the underlying biological processes to be shared. Thus, an interesting research question is how to share information across data sets, e.g. how to *transfer* knowledge from one experimental condition to another, or how to utilise patterns from closely related (potentially larger) studies to the data set of interest. We are specifically interested in achieving this in the context of latent variable models. Thus, it would be interesting to investigate how to extend the VAE frameworks developed in this thesis to capture patterns across data sets. For example, one could combine these ideas with Neural Processes (Garnelo et al., 2018; Kim et al., 2019) that amortise inference over multiple data sets.

Appendices

A

Synthetic data generation mechanisms

A.1 Synthetic data generative mechanism

- $z_i \sim U(-2, 2)$ for $i = 1, ..., N$
- $c_i \sim U(-2, 2)$ for $i = 1, ..., N$
- for features $j = 1, \dots 5$
  - $y^{(j)} := w_j \cdot \cos(z) + \varepsilon$ where $w_j \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
- for features $j = 6, \dots 10$
  - $y^{(j)} := 0.5z + w_j c + \varepsilon$ where $w_j \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$
- for features $j = 11, \dots 15$
  - $y^{(j)} := w_j \tanh(z)c + \varepsilon$ where $w_j \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$
- for features $j = 16, \dots 20$
  - $y^{(j)} := w_j c + 0.01(0.12 - w_j) z \tanh(c) + \varepsilon$ where $w_j \in \{0.02, 0.04, 0.06, 0.08, 0.1\}$
- for features $j = 21, \dots 25$
  - $y_i^{(j)} := 0.1 \tanh(z) + 0.2 \tanh(c) + w_j \sin(z)x + \varepsilon$ where $w_j \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$

  with noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.05$

A.2 Synthetic data generative mechanism for the batch effect

To simulate a batch effect (here for two known batches), we generated a latent (z_1, z_2) space together with batch indicators c for $i = 1, ..., N$ $(N = 1000)$

- $z_{1,i} \sim U(-2, 2)$ for $i = 1, ..., N$
- $z_{2,i} \sim U(-2, 2)$ for $i = 1, ..., N$
- $c_i \sim \text{Bernoulli}(0.5)$ for $i = 1, ..., N$
- for features $j = 1, \dots 5$
  - $y^{(j)} := 0.3w_j \tanh(z_1) + 0.2w_j \exp(-0.5z_2^2) + 0.3w_j c + \varepsilon$ where $w_j \in \{1, 2, 3, 4, 5\}$
- for features $j = 6, \dots 10$
  - $y^{(j)} := 0.2w_j \tanh(z_2) + 0.4(6 - w_j)c + \varepsilon$ where $w_j \in \{1, 2, 3, 4, 5\}$
- for features $j = 11, \dots 15$
  - $y^{(j)} := w_j z_1 + (0.6 - w_j)z_2 + (0.6 - w_j) \tanh(z_1)c + \varepsilon$ where $w_j \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
- for features $j = 16, \dots 20$
  - $y^{(j)} := \tanh(2z_1) + w_j \tanh(z_2) + \varepsilon$ where $w_j \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$
- for features $j = 21, \dots 25$
  - $y_i^{(j)} := 0.1c + w_j \tanh(z_1)c + \varepsilon$ where $w_j \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$

  with noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = 0.05$

References

Vincent Adam, James Hensman, and Maneesh Sahani. "Scalable transformed additive signal decomposition by non-conjugate Gaussian process inference". In: *26th international workshop on machine learning for signal processing (MLSP)*. IEEE, 2016, pp. 1–6.

Samuel K. Ainsworth, Nicholas J. Foti, Adrian K. C. Lee, and Emily B. Fox. "oi-VAE: Output Interpretable VAEs for Nonlinear Group Factor Analysis". en. In: *International Conference on Machine Learning*. 2018, pp. 119–128.

Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. "Kernels for vector-valued functions: A review". In: *Foundations and Trends® in Machine Learning* 4.3 (2012), pp. 195–266.

Ricard Argelaguet, Britta Velten, Damien Arnol, Sascha Dietrich, Thorsten Zenz, John C. Marioni, Florian Buettner, Wolfgang Huber, and Oliver Stegle. "Multi-Omics Factor Analysis—a framework for unsupervised integration of multi-omics data sets". In: *Molecular systems biology* 14.6 (2018).

Muhammed Fatih Balın, Abubakar Abid, and James Zou. "Concrete Autoencoders: Differentiable Feature Selection and Reconstruction". In: *International Conference on Machine Learning*. 2019, pp. 444–453.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

Michael Betancourt. "A conceptual introduction to Hamiltonian Monte Carlo". In: *arXiv preprint arXiv:1701.02434* (2017).

Christopher M. Bishop. "Bayesian PCA". In: *Advances in neural information processing systems*. 1999, pp. 382–388.

David M. Blei and Michael I. Jordan. "Variational inference for Dirichlet process mixtures". In: *Bayesian analysis* 1.1 (2006), pp. 121–143.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877.

Erik Bodin, Neill DF Campbell, and Carl Henrik Ek. "Latent gaussian process regression". In: *arXiv preprint arXiv:1707.05534* (2017).

Leo Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.

Alison J. Burnham, Roman Viveros, and John F. MacGregor. "Frameworks for latent variable multivariate regression". In: *Journal of chemometrics* 10.1 (1996), pp. 31–45.

Kieran R. Campbell and Christopher Yau. "A descriptive marker gene approach to single-cell pseudotime inference". In: *Bioinformatics* 35.1 (2018), pp. 28–35.

Kieran R. Campbell and Christopher Yau. "Uncovering pseudotemporal trajectories with covariates from single cell and bulk expression data". In: *Nature communications* 9.1 (2018), p. 2442.

Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. "Stan: A probabilistic programming language". In: *Journal of statistical software* 76.1 (2017).

Darya Chudova, Scott Gaffney, Eric Mjolsness, and Padhraic Smyth. "Translation-invariant mixture models for curve clustering". In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 79–88.

Andreas Damianou and Neil Lawrence. "Deep gaussian processes". In: *Artificial Intelligence and Statistics*. 2013, pp. 207–215.

Andreas Damianou and Neil D. Lawrence. "Semi-described and Semi-supervised Learning with Gaussian Processes". In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. UAI'15. AUAI Press, 2015, pp. 228–237.

Andreas C. Damianou, Michalis K. Titsias, and Neil D. Lawrence. "Variational inference for latent variables and uncertain inputs in Gaussian processes". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1425–1486.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. "Deep unsupervised clustering with gaussian mixture variational autoencoders". In: *arXiv preprint arXiv:1611.02648* (2016).

Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

Nicolas Durrande, David Ginsbourger, and Olivier Roustant. "Additive covariance kernels for high-dimensional Gaussian process modeling". In: *Annales de la Faculté des sciences de Toulouse: Mathématiques*. Vol. 21. 2012, pp. 481–499.

Nicolas Durrande, David Ginsbourger, Olivier Roustant, and Laurent Carraro. "ANOVA kernels and RKHS of zero mean functions for model-based sensitivity analysis". In: *Journal of Multivariate Analysis* 115 (2013), pp. 57–67.

Vincent Dutordoir, Hugh Salimbeni, James Hensman, and Marc Deisenroth. "Gaussian process conditional density estimation". In: *Advances in Neural Information Processing Systems*. 2018, pp. 2385–2395.

Gökcen Eraslan, Lukas M. Simon, Maria Mircea, Nikola S. Mueller, and Fabian J. Theis. "Single-cell RNA-seq denoising using a deep count autoencoder". In: *Nature communications* 10.1 (2019), p. 390.

Christina Ernst, Nils Eling, Celia P. Martinez-Jimenez, John C. Marioni, and Duncan T. Odom. "Staged developmental mapping and X chromosome transcriptional dynamics during mouse spermatogenesis". In: *Nature communications* 10.1 (2019), p. 1251.

Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. "Implicit Reparameterization Gradients". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 441–452.

Ronald Fisher. "The Correlation between Relatives on the Supposition of Mendelian Inheritance". In: (1918).

Kenichiro Furuyama, Simona Chera, Léon Van Gurp, Daniel Oropeza, Luiza Ghila, Nicolas Damond, Heidrun Vethe, Joao A. Paulo, Antoinette M. Joosten, and Thierry Berney. "Diabetes relief in mice by glucose-sensing insulin-secreting human -cells". In: *Nature* 567.7746 (2019), p. 43.

Charles Gadd, Sara Wade, Akeel Shah, and Dimitris Grammatopoulos. "Pseudo-marginal Bayesian inference for supervised Gaussian process latent variable models". In: *arXiv preprint arXiv:1803.10746* (2018).

Scott J. Gaffney and Padhraic Smyth. "Joint probabilistic curve clustering and alignment". In: *Advances in neural information processing systems*. 2005, pp. 473–480.

Xinbo Gao, Xiumei Wang, Dacheng Tao, and Xuelong Li. "Supervised Gaussian Process Latent Variable Model for Dimensionality Reduction". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.2 (Apr. 2011), pp. 425–434.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Eslami, and Yee Whye Teh. "Neural Processes". In: *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*. 2018.

Paul Geladi and Bruce R. Kowalski. "Partial least-squares regression: a tutorial". In: *Analytica chimica acta* 185 (1986), pp. 1–17.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

Samuel Gershman and Noah Goodman. "Amortized inference in probabilistic reasoning". In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 36. 2014.

Zoubin Ghahramani. "Probabilistic machine learning and artificial intelligence". In: *Nature* 521.7553 (2015), pp. 452–459.

Zoubin Ghahramani and Matthew J. Beal. "Propagation algorithms for variational Bayesian learning". In: *Advances in neural information processing systems*. 2001, pp. 507–513.

Walter R. Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.

Chong Gu. *Smoothing spline ANOVA models*. Vol. 297. Springer Science & Business Media, 2013.

W. Keith Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. In: (1970).

J. Hensman, M. Rattray, and N. D. Lawrence. “Fast Nonparametric Clustering of Structured Time-Series”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (Feb. 2015), pp. 383–393.

James Hensman, Magnus Rattray, and Neil D. Lawrence. “Fast variational inference in the conjugate exponential family”. In: *Advances in neural information processing systems*. 2012, pp. 2888–2896.

James Hensman, Nicolo Fusi, and Neil D. Lawrence. “Gaussian processes for big data”. In: *arXiv preprint arXiv:1309.6835* (2013).

Magnus R. Hestenes. “Multiplier and gradient methods”. In: *Journal of optimization theory and applications* 4.5 (1969), pp. 303–320.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.” In: *International Conference on Learning Representations* 2.5 (2017), p. 6.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. “Stochastic variational inference”. In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 1303–1347.

Matthew D. Hoffman, Carlos Riquelme, and Matthew J. Johnson. “The -VAE’s Implicit Prior”. In: *Workshop on Bayesian Deep Learning, NIPS*. 2017, pp. 1–5.

Giles Hooker. “Generalized functional anova diagnostics for high-dimensional functions of dependent variables”. In: *Journal of Computational and Graphical Statistics* 16.3 (2007), pp. 709–732.

Tomoharu Iwata, David Duvenaud, and Zoubin Ghahramani. “Warped mixtures for nonparametric cluster shapes”. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. UAI’13 (2013).

Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: *International Conference on Learning Representations* (2017).

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. “An introduction to variational methods for graphical models”. In: *Machine learning* 37.2 (1999), pp. 183–233.

Cari G. Kaufman and Stephan R. Sain. “Bayesian functional {ANOVA} modeling using Gaussian process prior distributions”. In: *Bayesian Analysis* 5.1 (2010), pp. 123–149.

Hyunjik Kim and Andriy Mnih. “Disentangling by Factorising”. en. In: *International Conference on Machine Learning*. July 2018, pp. 2649–2658.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. “Attentive neural processes”. In: *International Conference on Learning Representations*. 2019.

Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)* (2014).

Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes". In: *Proceedings of the International Conference on Learning Representations (ICLR)* (2014).

Durk P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. "Semi-supervised learning with deep generative models". In: *Advances in neural information processing systems*. 2014, pp. 3581–3589.

Nancy B. Kuemmerle, Evelien Rysman, Portia S. Lombardo, Alison J. Flanagan, Brea C. Lipe, Wendy A. Wells, Jason R. Pettus, Heather M. Froehlich, Vincent A. Memoli, and Peter M. Morganelli. "Lipoprotein lipase links dietary fat to solid tumor cell proliferation". In: *Molecular cancer therapeutics* 10.3 (2011), pp. 427–436.

Kenichi Kurihara, Max Welling, and Yee Whye Teh. "Collapsed Variational Dirichlet Process Mixture Models." In: *IJCAI*. Vol. 7. 2007, pp. 2796–2801.

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

Neil D. Lawrence. "Gaussian process latent variable models for visualisation of high dimensional data". In: *Advances in neural information processing systems*. 2004, pp. 329–336.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits, 1998". In: *URL http://yann. lecun. com/exdb/mnist* 10 (1998), p. 34.

Benjamin Lengerich, Sarah Tan, Chun-Hao Chang, Giles Hooker, and Rich Caruana. "Purifying Interaction Effects with the Functional ANOVA: An Efficient Algorithm for Recovering Identifiable Additive Models". In: *arXiv preprint arXiv:1911.04974* (2019).

Zachary C. Lipton. "The mythos of model interpretability". In: *arXiv preprint arXiv:1606.03490* (2016).

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep learning face attributes in the wild". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3730–3738.

Romain Lopez, Jeffrey Regier, Michael B. Cole, Michael I. Jordan, and Nir Yosef. "Deep generative modeling for single-cell transcriptomics". In: *Nature methods* 15.12 (2018), p. 1053.

Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4765–4774.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *International Conference on Learning Representations*. 2017.

Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. "Disentangling Disentanglement in Variational Autoencoders". en. In: *International Conference on Machine Learning*. 2019, pp. 4402–4412.

José L. McFaline-Figueroa, Andrew J. Hill, Xiaojie Qiu, Dana Jackson, Jay Shendure, and Cole Trapnell. "A pooled single-cell genetic screen identifies regulatory checkpoints in the continuum of the epithelial-to-mesenchymal transition". In: *Nature genetics* 51.9 (2019), pp. 1389–1398.

Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

Nicholas Metropolis and Stanislaw Ulam. "The monte carlo method". In: *Journal of the American statistical association* 44.247 (1949), pp. 335–341.

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. "Monte Carlo Gradient Estimation in Machine Learning". In: *arXiv preprint arXiv:1906.10652* (2019).

Jennifer Munkley, Urszula L. McClurg, Karen E. Livermore, Ingrid Ehrmann, Bridget Knight, Paul Mccullagh, John Mcgrath, Malcolm Crundwell, Lorna W. Harries, and Hing Y. Leung. "The cancer-associated cell migration protein TSPAN1 is under control of androgens and its upregulation increases prostate cancer cell migration". In: *Scientific reports* 7.1 (2017), p. 5249.

Kevin P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Kaspar Märtens and Christopher Yau. "BasisVAE: Translation-invariant feature-level clustering with Variational Autoencoders". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

Kaspar Märtens and Christopher Yau. "Neural Decomposition: Functional ANOVA with Variational Autoencoders". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

Kaspar Märtens, Michalis Titsias, and Christopher Yau. "Augmented Ensemble MCMC sampling in Factorial Hidden Markov Models". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019, pp. 2359–2367.

Kaspar Märtens, Kieran Campbell, and Christopher Yau. "Decomposing feature-level variation with Covariate Gaussian Process Latent Variable Models". In: *International Conference on Machine Learning*. 2019, pp. 4372–4381.

Eric Nalisnick, Lars Hertel, and Padhraic Smyth. "Approximate inference for deep latent gaussian mixtures". In: *NIPS Workshop on Bayesian Deep Learning*. Vol. 2. 2016.

Radford M. Neal. "Markov Chain Sampling Methods for Dirichlet Process Mixture Models". In: *Journal of Computational and Graphical Statistics* 9.2 (2000), pp. 249–265.

Radford M. Neal. "MCMC using Hamiltonian dynamics". In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

John Novembre, Toby Johnson, Katarzyna Bryc, Zoltán Kutalik, Adam R. Boyko, Adam Auton, Amit Indap, Karen S. King, Sven Bergmann, and Matthew R. Nelson. "Genes mirror geography within Europe". In: *Nature* 456.7218 (2008), p. 98.

Topi Paananen, Juho Piironen, Michael Riis Andersen, and Aki Vehtari. "Variable selection for Gaussian processes via sensitivity analysis of the posterior predictive distribution". en. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 1743–1752.

Konstantina Palla, Zoubin Ghahramani, and David A. Knowles. "A nonparametric variable clustering model". In: *Advances in Neural Information Processing Systems*. 2012, pp. 2987–2995.

Tomi Peltola, Aki S. Havulinna, Veikko Salomaa, and Aki Vehtari. "Hierarchical Bayesian Survival Analysis and Projective Covariate Selection in Cardiovascular Event Risk Prediction." In: *Proceedings of the Eleventh UAI Conference on Bayesian Modeling Applications Workshop-Volume 1218*. Citeseer, 2014, pp. 79–88.

Emma Pierson, Pang Wei Koh, Tatsunori Hashimoto, Daphne Koller, Jure Leskovec, Nicholas Eriksson, and Percy Liang. "Inferring Multidimensional Rates of Aging from Cross-Sectional Data". In: *Proceedings of machine learning research* 89 (2019), p. 97.

John C. Platt and Alan H. Barr. "Constrained differential optimization". In: *Neural Information Processing Systems*. 1988, pp. 612–621.

Michael JD Powell. "A method for nonlinear constraints in minimization problems". In: *Optimization* (1969), pp. 283–298.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *Journal of Machine Learning Research* 6 (2005), pp. 1939–1959.

J. O. Ramsay and B. W. Silvermann. *Functional Data Analysis*. Springer Series in Statistics. 1997.

Rajesh Ranganath, Sean Gerrish, and David Blei. "Black box variational inference". In: *Artificial Intelligence and Statistics*. 2014, pp. 814–822.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *arXiv preprint arXiv:1401.4082* (2014).

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

Andrew Rosenberg and Julia Hirschberg. "V-measure: A conditional entropy-based external cluster evaluation measure". In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007, pp. 410–420.

Hugh Salimbeni, Vincent Dutordoir, James Hensman, and Marc Deisenroth. “Deep Gaussian Processes with Importance-Weighted Variational Inference”. en. In: *International Conference on Machine Learning*. May 2019, pp. 5589–5598.

Alex K. Shalek, Rahul Satija, Joe Shuga, John J. Trombetta, Dave Gennert, Diana Lu, Peilin Chen, Rona S. Gertner, Jellert T. Gaublomme, and Nir Yosef. “Single-cell RNA-seq reveals dynamic paracrine control of cellular variation”. In: *Nature* 510.7505 (2014), p. 363.

Aaron Shon, Keith Grochow, Aaron Hertzmann, and Rajesh P. Rao. “Learning shared latent structure for image synthesis and robotic imitation”. In: *Advances in neural information processing systems*. 2006, pp. 1233–1240.

Edward Snelson and Zoubin Ghahramani. “Sparse Gaussian processes using pseudo-inputs”. In: *Advances in neural information processing systems*. 2006, pp. 1257–1264.

Ilya M. Sobol. “Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates”. In: *Mathematics and computers in simulation* 55.1-3 (2001), pp. 271–280.

Ilya M. Sobol. “Sensitivity estimates for nonlinear mathematical models”. In: *Mathematical modelling and computational experiments* 1.4 (1993), pp. 407–414.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. “Learning structured output representation using deep conditional generative models”. In: *Advances in neural information processing systems*. 2015, pp. 3483–3491.

Valentine Svensson and Lior Pachter. “Interpretable factor models of single-cell RNA-seq via variational autoencoders”. In: *bioRxiv* (2019), p. 737601.

Yee Whye Teh, David Newman, and Max Welling. “A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation”. In: *Advances in neural information processing systems*. 2007, pp. 1353–1360.

Michael E. Tipping and Christopher M. Bishop. “Mixtures of Probabilistic Principal Component Analyzers”. In: *Neural computation* 11.2 (1999), pp. 443–482.

Michael E. Tipping and Christopher M. Bishop. “Probabilistic principal component analysis”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622.

Michalis Titsias. “Variational Learning of Inducing Variables in Sparse Gaussian Processes”. en. In: *Artificial Intelligence and Statistics*. Apr. 2009, pp. 567–574.

Michalis Titsias and Neil D. Lawrence. “Bayesian Gaussian Process Latent Variable Model”. en. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Mar. 2010, pp. 844–851.

Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J. Lennon, Kenneth J. Livak, Tarjei S. Mikkelsen, and John L. Rinn. “The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells”. In: *Nature biotechnology* 32.4 (2014), p. 381.

Raquel Urtasun and Trevor Darrell. "Discriminative Gaussian process latent variable model for classification". In: *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 927–934.

Chunyi Wang and Radford M. Neal. "Gaussian process regression with heteroscedastic or non-gaussian residuals". In: *arXiv preprint arXiv:1212.6246* (2012).

John N. Weinstein, Eric A. Collisson, Gordon B. Mills, Kenna R. Mills Shaw, Brad A. Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M. Stuart, and Cancer Genome Atlas Research Network. "The cancer genome atlas pan-cancer analysis project". In: *Nature genetics* 45.10 (2013), p. 1113.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

Simon N. Wood, Zheyuan Li, Gavin Shaddick, and Nicole H. Augustin. "Generalized additive models for gigadata: modeling the UK black smoke network daily data". In: *Journal of the American Statistical Association* 112.519 (2017), pp. 1199–1210.

Mike Wu and Noah Goodman. "Multimodal Generative Models for Scalable Weakly-Supervised Learning". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 5575–5585.

Shipeng Yu, Kai Yu, Volker Tresp, Hans-Peter Kriegel, and Mingrui Wu. "Supervised probabilistic principal component analysis". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 464–473.

Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. "Bias and generalization in deep generative models: An empirical study". In: *Advances in Neural Information Processing Systems*. 2018, pp. 10792–10801.