ASSUMPTIONS AND EFFICIENCY IN GAUSSIAN PROCESS MODELLING

ANDERS KIRK UHRENHOLT

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING
UNIVERSITY OF GLASGOW

JULY 2021

Acknowledgements

First and foremost a big thank you to my supervisor, Bjørn Sand Jensen, for inviting me to the fair country of Scotland and encouraging me to pursuit this degree. I'm grateful for the excellent guidance and help in identifying the few good ideas amongst the numerous bad ones that I have burdened you with over the past four years. It's been a pleasure.

To my parents, Ane, Lars, and Jakob: Thank you for the support from afar. Writing the thesis in the midst of a pandemic meant that the connection to home was more important than ever. I'm eternally grateful that each of you were never more than a phone call away.

Thank you to Tristan, Antoine, Francesco, Natascha, Grimur, Valentin, Julia, Gözel, and Oisin for insightful discussions and constructive feedback. Your company made the PhD journey all the more enjoyable.

And lastly, thank you, Robyn, for unconditionally supporting, encouraging, and – most importantly – tolerating me throughout this endeavour. You have been, and still are, amazing and I couldn't have asked for more.

Table of Contents

List of Figures

*Figure also used in Uhrenholt et al. [2021]
†Figure also used in Uhrenholt and Jensen [2019]

Notation

Scalars, vectors, and matrices

We let scalar variables be denoted by non-bold, lowercase, italic characters, e.g. $x \in \mathbb{R}$. When referring to the tally of some quantity, such as the number of datapoints or the number of dimensions, we will use non-bold, uppercase, italic characters, e.g. $\{x_i\}_{i=1}^N$. Vectors of arbitrary dimensionality will be denoted with bold, lowercase, non-italic characters, e.g. $\mathbf{x} \in \mathbb{R}^D$. Unless otherwise specified, vectors are assumed to be column-oriented. Finally, we use bold, uppercase, non-italic characters for matrices, e.g. $\mathbf{X} \in \mathbb{R}^{N \times D}$.

Indexing

We use subscript to retrieve the scalar entry of vectors, e.g. $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$. For matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ we refer to the i'th row by $\mathbf{x}_i$ and the j'th column by $\mathbf{x}_{:,j}$, i.e.:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{:,1} & \mathbf{x}_{:,2} & \cdots & \mathbf{x}_{:,D} \end{bmatrix}.$$

The scalar entry in the i'th row and j'th column is denoted by $x_{i,j}$. We will on rare occasions have to diverge from these conventions to avoid notational clutter, but we will make it clear how the overloaded notation is to be understood in a specific context.

Chapter 1

Introduction

Probabilistic machine learning is an interaction between assumptions and data. A model encodes a set of assumptions through its specification of e.g. conditional dependencies, feature representation, and free parameters, which in conjunction define an a priori hypothesis space. Fitting the model is now equivalent to narrowing this space down to a (set of) viable explanation(s) that agree with the training data. The result is a calibrated model that may yield insights about some phenomenon of interest and provide predictions for novel observations. A popular and successful approach in contemporary machine learning is to specify a very large hypothesis space, constructed through millions of free parameters, and rely on vast amounts of data for arriving at a reasonable parameterisation. This approach, mainly exemplified by deep neural networks, has been instrumental for recent decades' advances in weak artificial intelligence, and it is made possible by the availability of abundant data and computing power.

This thesis takes the view that, despite the successes of big data methods, we should still be interested in progressing the field of parsimonious models that make efficient use of the available resources. Firstly, because technological developments in computing efficiency inevitably create a push towards packing more intelligence into smaller devices such as watches and sensors whose limited memory, power supply, and computational capacity prohibits excessively complex models (Chen et al. [2014], Gokhale et al. [2014]). And secondly, because data sources themselves become ever more esoteric. There is thus a hard bound on the amount of training data available when all data originates from e.g. one particular person (Ghahramani [2015]) or a novel, scientific experiment where new samples are costly to obtain (Radovic et al. [2018]). The focus of this thesis is modelling scenarios where resources – here meaning training data, computational power, or both – must be efficiently utilised. We are especially interested in dissecting how the underlying assumptions of a given model may affect such efficiency, and whether manipulating the assumptions can promote better resource utilisation.

To this end Gaussian processes (GP) provide a very popular methodology for learning functional relationships between input and target values in a data efficient manner. They allow the model designer to reason, not about the inner workings of the function, but about the relationship between different function evaluations. This enables a powerful paradigm for encoding interpretable assumptions such as smoothness, periodicity, and stationarity. For standard regression problems we can furthermore obtain closed-form expressions for two of the difficult quantities in Bayesian inference: the posterior and the marginal likelihood. However, the downside is a model with limited flexibility that scales poorly in the number of observed datapoints.

Much research has been devoted to the analysis and manipulation of the assumptions underpinning GP modelling and inference. For instance, we can obtain a more flexible model by composing multiple GP functions in a "deep" structure, by using a non-Gaussian likelihood, or by assuming that the inputs themselves are unknown. On the inference side, the majority of progress in GP research over the past two decades is arguably owed to the simple assumption that the true function posterior can be adequately estimated by conditioning on a small set of inducing points. This body of work has paved the way for GP models entering the world of big data and producing state-of-the-art results while conforming to a principled, Bayesian paradigm.

In this thesis we similarly focus on the assumptions that define a given GP model and its inference algorithm. However, in contrast to the above mentioned research we are instead interested in the degree to which such assumptions affect the resource utilisation w.r.t. learning and making predictions. We claim that state-of-the-art GP models inadvertently encode assumptions that, depending on the setting in which they are employed, may lead to inefficient consumption of training data and/or computational resources. In addition, we claim that revising those assumptions results in models that are demonstrably more efficient. We will focus on three important areas of GP modelling – sparse variational inference, Bayesian optimisation for target vector estimation, and preference learning with multiple users – and for each identify assumptions in the inference scheme and/or model specification that may be problematic in terms of making optimal use of the computational resources or training data. We will then propose minute changes to these assumptions that result in drastically more efficient models.

1.1 Outline of thesis

The research contributions of this thesis are contained in Chapters 3, 4, and 5. Each chapter targets a separate modelling problem that incorporates Gaussian processes to achieve state-of-the-art results. For each problem we examine i. what assumptions are implicitly and explicitly encoded in prevalent approaches, and ii. how we can update these assumptions in order to achieve more resource efficient models and/or inference schemes. The thesis is structured as follows:

- In **Chapter 2** we introduce the requisite background theory pertaining to probabilistic modelling, Bayesian reasoning, inference methods, and Gaussian processes.

- In **Chapter 3** we consider the assumption underpinning the de facto standard approach of dealing with big data and complex modelling architectures in the GP paradigm, namely the inducing points methodology. We show that by revising the model specification so that inducing points are stochastically generated rather than deterministically selected, we obtain a model that adapts its computational complexity to the requirements of the data.

  This chapter is based on the paper: A. K. Uhrenholt, V. Charvet, and B. S. Jensen. Probabilistic Selection of Inducing Points in Sparse Gaussian Processes. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2021.

- **Chapter 4** focuses on the method of Bayesian optimisation applied to target vector estimation, where we argue that the default approach of using a single-output GP as surrogate model results in pathological behaviour that severely impacts the downstream optimisation. By updating the surrogate model to better reflect the optimisation task at hand, and through the introduction of novel acquisition functions, we derive a new optimisation procedure that is much more data efficient than the default approach.

  The chapter expands upon the paper: A. K. Uhrenholt and B. S. Jensen. Efficient Bayesian Optimization for Target Vector Estimation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 2661–2670. PMLR, 2019.

- In **Chapter 5** we turn to preference learning for multiple users. Here we introduce a simple assumption, namely that observed preference responses rely not only on the items but also on some unobserved user features. We demonstrate that we can infer this latent information directly from the user responses, which allows us to simultaneously learn the user preferences from fewer data points compared to baseline methods whilst obtaining an abstract representation of each user that can be valuable for post-hoc analysis and downstream modelling tasks such as clustering.

- **Chapter 6** concludes the thesis by summarising the developed models and results, and setting a course for future research that this thesis may inspire.

Chapter 2

Background

2.1 Probabilistic modelling

A model is an idealised mathematical description of a system. Purposes may vary but we typically utilise models for gaining insight about some phenomenon of interest and predicting novel outcomes that may inform subsequent decision making. In machine learning the model constitutes only one part of the necessary ingredients, the other being a set of observations produced by the system that we want our model to explain. By endowing the model with the capability of choosing an explanation from some predefined *hypothesis space* and codifying the suitability of a given explanation in an objective function, $\mathcal{L}(\cdot)$, we have a way of letting the model "learn" from the observed data and evaluating the quality of any given model that has been provided with data.

Usually a given model will be unable to perfectly explain what has been observed. This is especially true when the observations are collected from the real world where the data have been corrupted by noise. One way of accounting for such inaccuracy while still obtaining meaningful models is to accept the system as being *uncertain* and explicitly incorporating stochasticity in the modelling framework. Specifically, we assume noisy measurements to be realisations of random variables which in turn are specified by probability distributions. Any data point – observed or predicted – is thus taken to be the outcome of such a distribution, and a *probabilistic model* is one that assigns a probability to any collection of outcomes.

This provides a formalism for expressing how *likely* we are to observe a collection of realisations under a given model. Different probabilistic models can thus be assessed in terms of the likelihood that they associate with our observations, or training data, $\mathcal{D}$. Furthermore, if a model is granted flexibility in how probability is assigned, typically through a set of free parameters, θ, we can search for the most appropriate model configuration by optimising the likelihood, $p(\mathcal{D}; \theta, \mathcal{M})$, i.e. the probability of observing $\mathcal{D}$ under model $\mathcal{M}$ with parameterisation θ. We refer to this search as "training" or "fitting" the model. Commonly we construct a model that specifies a conditionally independent distribution over N observations, $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$, in which case the likelihood factorises as a product. And to avoid numerical instability resulting from repeated multiplications, we apply the (monotonic) logarithmic transformation to arrive at the default objective function for probabilistic models:

$$\mathcal{L}(\theta) \triangleq \log p(\mathcal{D}; \theta, \mathcal{M}) = \sum_{i=1}^{N} \log p(x_i \mid \theta, \mathcal{M}).$$

One important aspect of probabilistic modelling is the differentiation between signal and noise. For example, the probabilistic linear regression model (Bishop [2006]) assumes a set of response variables, $\mathbf{y} \in \mathbb{R}^N$, to be linear transformations of explanatory variables, $\mathbf{X} \in \mathbb{R}^{N \times D}$, corrupted by additive, Gaussian noise:

$$p(\mathbf{y} \mid \mathbf{X}; \mathbf{w}, \sigma^2) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{X}\mathbf{w}, \mathbf{I}\sigma^2\right),$$

for $\mathbf{w} \in \mathbb{R}^D$. This yields the log likelihood:

$$\mathcal{L}(\theta; \mathbf{y}, \mathbf{X}) = \sum_{i=1}^{N} \log \mathcal{N}\left(y_i \mid \mathbf{x}_i^T \mathbf{w}, \sigma^2\right)$$

where $\theta = \{\mathbf{w}, \sigma^2\}$ are the free parameters of the model. Here, the *latent signal* is given by $\mathbf{X}\mathbf{w}$, and constitutes what we can hope to learn. Meanwhile, the residuals, $\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{X}\mathbf{w}$, are artefacts of the inherent noise of the system that can never be predicted regardless of the amount of training data. The best we can settle for is to learn the noise distribution, i.e. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}\sigma^2)$.

Note that the linear model is characterised as a conditional probability distribution, and that this distribution describes the generative process for $\mathbf{y}$, i.e. how we assume that $\mathbf{y}$ came to be. This is a general trait of probabilistic models, and we will throughout this thesis say that we *specify* a model by writing up the factorisation of the joint distribution over uncertain variables. This will interchangeably be referred to as the *generative* model.

2.2 Bayesian inference

Probabilistic models can be categorised as being either frequentist or Bayesian. Under the frequentist paradigm we perceive a given model parameterisation, θ, as the explanation of the observed data. This explanation is inherently deterministic and a question such as "what is the most probable value of θ?" is nonsensical since we cannot associate probabilities with deterministic quantities.

In Bayesian modelling we interpret the explanation itself as being uncertain. This allows us

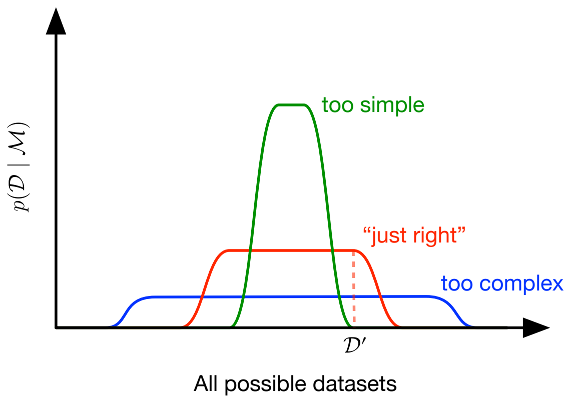

Figure 2.1: Illustration of the principle of Occam's Razor, as realised through Bayesian inference. The horizontal axis represents all possible datasets we may encounter while $\mathcal{D}'$ is the one particular instantiation that we have observed. The vertical axis shows the marginal evidence for three models, as a function of datasets. An overly complex model (blue) will have to spread its prior hypothesis space wide, thus assigning less probability to any particular instantiation. On the other hand, a too constrictive model (green) will not consider $\mathcal{D}'$ as a viable outcome. The best trade-off is the model that views $\mathcal{D}'$ as reasonable without entertaining an abundant number of alternative explanations (red). The illustration is adapted from Rasmussen and Ghahramani [2001].

to associate both $\mathcal{M}$ and θ with a probability distribution and reason about any model and its parameterisation as just a particular realisation. This fundamentally separates the Bayesian and frequentist paradigms in at least three respects. First, it requires us to specify our *prior beliefs* through the distributions, $p(\mathcal{M})$ and $p(\theta \mid \mathcal{M})$.[1] This is sometimes pointed to as a fallacy of Bayesian inference, since the model designer's subjective opinion may ultimately affect the conclusion. However, proponents will argue that all inference, Bayesian or otherwise, is impossible without making some initial assumptions; in frequentist models these assumptions just materialise implicitly through the construction of $\mathcal{M}$ (Gelman et al. [2013]). Second, we no longer seek a single canonical explanation for our data. Instead, we are interested in learning the entire distribution over possible explanations: $p(\theta \mid \mathcal{D}, \mathcal{M})p(\mathcal{M} \mid \mathcal{D})$. We refer to this as the *posterior* distribution. When making predictions about the output for a novel data point, $x_\star$, we are then required to marginalise over the posterior:

$$p(x_\star \mid \mathcal{M}, \mathcal{D}) = \int p(x_\star \mid \theta, \mathcal{M}) p(\theta \mid \mathcal{D}, \mathcal{M}) \, \mathrm{d}\theta.$$

Thirdly, we obtain a new measure for assessing a model, namely the model evidence:

$$\mathcal{L}(\mathcal{M}) \triangleq p(\mathcal{D} \mid \mathcal{M}) = \int p(\mathcal{D} \mid \theta) p(\theta \mid \mathcal{M}) \, \mathrm{d}\theta.$$

[1]Usually we restrict attention to a single model and refrain from explicitly conditioning on $\mathcal{M}$. We will do the same in this thesis but include it now for expositional purposes.

That is, we consider how well the data are explained by *any* parameterisation of our model, weighted by the prior beliefs we have for each parameterisation. This has the attractive consequence of penalising overly complex models. Here, we use the term “complex” to describe models with a wide a priori hypothesis space that through $p(\theta \mid \mathcal{M})$ regards a large set of parameterisations as reasonable. Such a model must assign less prior probability to any particular realisation of θ since $\int p(\theta \mid \mathcal{M})\, \mathrm{d}\theta = 1.$ In turn, the few “good” explanations will be outweighed by the many inappropriate ones when evaluating the evidence, as illustrated in Figure 2.1. A model is therefore rewarded for constraining its hypothesis space which reduces the risk of accidentally arriving at a parameterisation that supports only the training data without generalising. This is why Bayesian inference is often said to embody the principle of Occam’s Razor which states that for two hypotheses that explain a phenomenon equally well, we should favour the one that makes the fewest assumptions (Rasmussen and Ghahramani [2001]).

Essential to the main topic of this thesis is the dichotomy between assumptions and flexibility that the prior establishes in Bayesian inference. In general, when we encode stronger assumptions through the use of a narrow prior distribution, we require less data to arrive at a conclusion with a high degree of certainty. This immediately implies that if we specify a prior that either allows for a wide range of viable explanations or encourages a parameterisation that is incongruent with the system being modelled, we must obtain more data and typically spend more computation in order to infer the posterior. In turn, we can generally expect better resource utilisation by tailoring our prior to encode a set of assumptions that are both sensible and restrictive.

We may encounter free parameters in the prior, $p(\theta \mid \mathcal{M})$, or data likelihood, $p(\mathcal{D} \mid \theta, \mathcal{M})$, that are not viewed as part of the model parameterisation. These are denoted *hyperparameters* and examples include kernel lengthscales in Gaussian processes (introduced in Section 2.5) and noise variance in Gaussian likelihoods. Committing fully to a Bayesian approach would entail marginalising over these along with θ. While this is sometimes seen in practice, e.g. Shah et al. [2013], it is much more common to maximise the evidence as a function of the hyperparameters – so called type II maximum likelihood (ML-II) – and argue that we are more protected against the pitfalls of frequentist modelling because the flexibility is encoded higher in the Bayesian hierarchy (Gelman et al. [2013]). This will also be the default approach in this thesis, although we note that relying on an excessive amount of hyperparameters can introduce pathological behaviour in one’s model, as recently documented in Ober et al. [2021].

The three principal quantities of Bayesian inference – the prior, the posterior, and the evidence – are linked through Bayes rule:

$$p(\theta \mid \mathcal{D}, \mathcal{M}) = \frac{p(\mathcal{D} \mid \theta, \mathcal{M}) p(\theta \mid \mathcal{M})}{p(\mathcal{D} \mid \mathcal{M})},$$

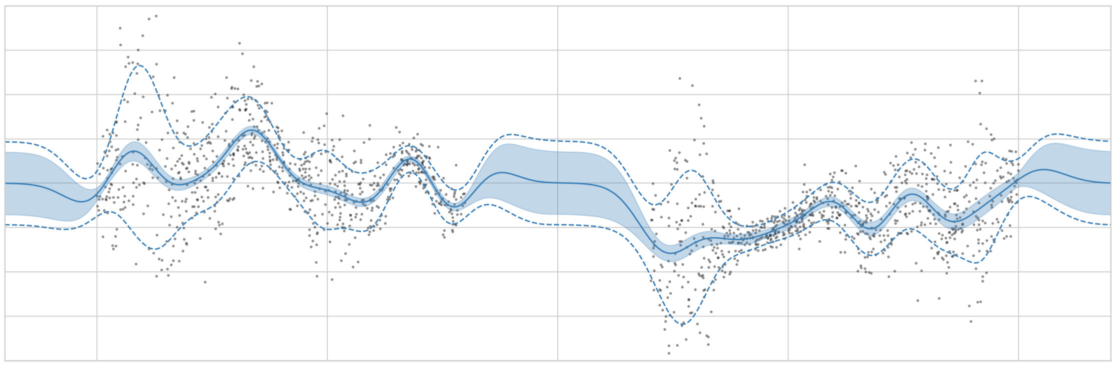

Figure 2.2: Demonstration of the distinction between epistemic and aleatoric uncertainty. The observed data (black dots) have observation noise that varies as a function of the input. The model has inferred a mean signal for the entire domain (solid blue line) along with the standard deviation (blue shading). This accounts for the epistemic uncertainty and is generally small in densily populated regions of the input space. In addition, the model has inferred the observation noise (blue stripped lines). This is the aleatoric uncertainty which, in contrast to its epistemic counterpart, will not be reduced by collecting more observations.

which tells us how to get from our prior beliefs and data likelihood to our posterior beliefs. This is most often applied in the context of going from no data observed, i.e. $p(\theta \mid \mathcal{M})$, to all data observed, i.e. $p(\theta \mid \mathcal{D}, \mathcal{M})$, but the rule is also applicable when data are collected sequentially over time and learning thus consists of a sequence of "Bayesian updates". We will encounter such a scenario in Section 2.4.

Bayes' rule can be extended hierarchically with a prior over multiple models in order to determine the posterior probability of each:

$$p(\mathcal{M} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \mathcal{M})p(\mathcal{M})}{p(\mathcal{D}) = \int p(\mathcal{D} \mid \mathcal{M})p(\mathcal{M})\,\mathrm{d}\mathcal{M}},$$

yielding a Bayesian approach to model selection. Most often, however, we just consider one model at a time which is equivalent to using a point distribution for $p(\mathcal{M})$. For the remainder of the thesis we will drop the dependency on $\mathcal{M}$ and assume that only one model is included in the inference.

Whilst frequentist models differentiate between signal and noise, we obtain under the Bayesian paradigm a further distinction, namely the separation of *aleatoric* and *epistemic* uncertainty. The former is identical to the noise term within a frequentist model and encompasses any stochasticity resulting from measurement inaccuracies or similar that we cannot hope to predict. Epistemic uncertainty, on the other hand, is the level of ignorance of our model. It accounts for the uncertainty that arises from not having obtained a sufficient amount of observations, and that could in principle be eliminated by collecting more data. This distinction is critical when assessing model predictions, and it plays a crucial role in fields such as optimal experimental design (Lindley [1956]) where we systematically collect data according to the maximal expected information gain. Figure 2.2 illustrates the distinction where we are modelling a set of 1D data points that have varying degrees of observation noise. By applying a Bayesian model (in this case a Gaussian process which will be introduced in Section 2.5) that assumes heteroscedasticity, i.e. that the noise is a function of the input, we can disentangle whether the predictive variance is due to measurement noise or lack of training data.

2.3 Handling intractability

While conceptually simple the Bayesian approach often leads to models for which both assessment and making predictions are impossible. This is due to the marginalisation of the latent parameters in the model evidence and predictive posterior:

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta) p(\theta) \, \mathrm{d}\theta, \qquad p(x_\star \mid \mathcal{D}) = \int p(x_\star \mid \theta) p(\theta \mid \mathcal{D}) \, \mathrm{d}\theta,$$

which are generally intractable. In this thesis we will rely on two approximation techniques for circumventing this problem, namely Monte Carlo sampling and variational inference.

2.3.1 Monte Carlo sampling

In the context of machine learning any method that relies on sampling repeatedly from a probability distribution is classified as a Monte Carlo method. Such methods are mainly applied for two purposes: obtaining samples from an intractable distribution, which will not be relevant in this thesis; and approximating an integral:

$$\mathbb{E}_{p(z)}\left[g(z)\right] = \int g(z) p(z) \, \mathrm{d}z \approx \frac{1}{S} \sum_{s=1}^{S} g(\tilde{z}^{(s)}), \qquad \tilde{z}^{(s)} \sim p(z).$$

This situation may arise when we have learnt a posterior over the model parameters, θ, and want to predict the expected value for a new point:

$$\hat{x}_\star = \int p(x_\star \mid \theta, \mathcal{D}) p(\theta \mid \mathcal{D}) \, \mathrm{d}\theta.$$

Somewhat more challenging is the scenario in which we associate parameters (λ, ϕ) with the distribution, $p_\lambda(z)$, and inner function, $g_\phi(z)$, and wish to optimise the integral with respect to these. As the number of parameters increases it will be necessary to leverage gradient information and apply first-order optimisation algorithms such as Stochastic Gradient Descent (Zhang [2004]) or Adam (Kingma and Ba [2014]). Estimating the gradients w.r.t. ϕ

is straightforward since the differentiation operator can be moved inside the expectation, leading to a new integral that may again be estimated with Monte Carlo sampling:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{p_\lambda(z)}\left[g_\phi(z)\right] &= \mathbb{E}_{p_\lambda(z)}\left[\nabla_\phi g_\phi(z)\right] \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\phi g_\phi(\tilde{z}^{(s)}), \\ \tilde{z}^{(s)} &\sim p_\lambda(z).\end{aligned}$$

For λ, however, this is not immediately applicable since:

$$\nabla_\lambda \mathbb{E}_{p_\lambda(z)}\left[g_\phi(z)\right] = \int g_\phi(z) \nabla_\lambda p_\lambda(z)\,\mathrm{d}z,$$

does not take the form of an expectation. We will rely on two methods for resolving this issue. When the random variable, z, can be written as a deterministic transformation, $z = h_\lambda(t)$, where $t \sim p_0$ for some unparameterised distribution p_0, we can apply the reparameterisation trick (Kingma and Welling [2014]) and rewrite the expectation as:

$$\mathbb{E}_{p_\lambda(z)}\left[g_\phi(z)\right] = \mathbb{E}_{p_0(t)}\left[g_\phi(h_\lambda(t))\right].$$

Since λ no longer defines the expectation we can approximate the gradient the same way as for ϕ. For instance, when $\mathbf{z} \in \mathbb{R}^D$ follows a multivariate normal, $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, we can make the reparameterisation $\mathbf{z} = h_{\boldsymbol{\mu},\mathbf{L}}(\mathbf{t}) = \mathbf{L}\mathbf{t} + \boldsymbol{\mu}$, where $\mathbf{t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{L}\mathbf{L}^T = \Sigma$. In the univariate case, $z \sim \mathcal{N}(\mu, \sigma^2)$, this simplifies to $t \sim \mathcal{N}(0,1)$ and $h_{\mu,\sigma}(t) = \sigma \cdot t + \mu$. The method has been generalised to also include discrete variables in Maddison et al. [2017] and Jang et al. [2016], subject to a continuous relaxation of the distribution.

When a deterministic transformation is not available, we can instead use a score function estimator (SFE) to approximate the gradient (Williams [1992], Fu [2006]). This method relies on the observation that $\nabla_\lambda p_\lambda(z) = p_\lambda(z)\nabla_\lambda \log p_\lambda(z)$. Making this substitution inside the expectation yields:

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{p_\lambda(z)}\left[g_\phi(z)\right] &= \int g_\phi(z)\left(\nabla_\lambda p_\lambda(z)\right)\mathrm{d}z \\ &= \int g_\phi(z)\left(p_\lambda(z)\nabla_\lambda \log p_\lambda(z)\right)\mathrm{d}z \\ &= \mathbb{E}_{p_\lambda(z)}\left[g_\phi(z)\nabla_\lambda \log p_\lambda(z)\right].\end{aligned}$$

We can then, yet again, apply Monte Carlo integration to approximate the gradients:

$$\nabla_\lambda \mathbb{E}_{p_\lambda(z)}\left[g_\phi(z)\right] \approx \frac{1}{S}\sum_{s=1}^{S} g_\phi(\tilde{z}^{(s)})\nabla_\lambda \log p_\lambda(\tilde{z}^{(s)})$$
$$\tilde{z}^{(s)} \sim p_\lambda(z).$$

These methods make it very easy to implement any objective function that takes the form of an expectation when relying on a framework that supports automatic differentiation, such as PyTorch (Paszke et al. [2019]) or Tensorflow (Abadi et al. [2016]). However, while the gradients provided by both methods are unbiased, they may be subject to a considerable amount of sampling noise, especially for the SFE method. We will return to this problem in Chapter 3.

2.3.2 Variational inference

The other approach for handling intractability, that we make use of in this thesis, is variational inference (VI) (Jordan et al. [1998], Blei et al. [2017]). Here we introduce a proposal distribution, $q_\phi(\theta)$, which relies on the *variational parameters*, ϕ, and is intended to approximate the true posterior, i.e. $q_\phi(\theta) \approx p(\theta \mid \mathcal{D})$. By applying Jensen's inequality we can now derive the evidence lower bound (ELBO):

$$\begin{aligned}
\log p(\mathcal{D}) &= \log \int \frac{q_\phi(\theta)}{q_\phi(\theta)} p(\mathcal{D} \mid \theta) p(\theta)\, \mathrm{d}\theta \\
&\geq \int q_\phi(\theta) \log \frac{p(\mathcal{D} \mid \theta) p(\theta)}{q_\phi(\theta)}\, \mathrm{d}\theta \\
&= \mathbb{E}_{q_\phi(\theta)}\left[\log p(\mathcal{D} \mid \theta)\right] - \mathrm{KL}\left[q_\phi(\theta) \parallel p(\theta)\right] \\
&\triangleq \mathcal{L}(\phi),
\end{aligned}$$

where $\mathrm{KL}\left[\cdot \parallel \cdot\right]$ denotes the Kullback-Leibler (KL) divergence. The ELBO serves as the objective to be maximised and has multiple purposes. Most importantly, increasing the ELBO has the effect of reducing the KL divergence from $q_\phi(\theta)$ to $p(\theta \mid \mathcal{D})$. This can be seen by substituting in $\log p(\mathcal{D} \mid \theta) = \log p(\theta \mid \mathcal{D}) + \log p(\mathcal{D}) - \log p(\theta)$ and rearranging:

$$\begin{aligned}
\mathcal{L}(\phi) &= \mathbb{E}_{q_\phi(\theta)}\left[\log p(\mathcal{D} \mid \theta) - \log q_\phi(\theta) + \log p(\theta)\right] \\
&= \mathbb{E}_{q_\phi(\theta)}\left[\log p(\theta \mid \mathcal{D}) + \log p(\mathcal{D}) - \cancel{\log p(\theta)} - \log q_\phi(\theta) + \cancel{\log p(\theta)}\right] \\
&= -\mathrm{KL}\left[q_\phi(\theta) \parallel p(\theta \mid \mathcal{D})\right] + \log p(\mathcal{D}).
\end{aligned}$$

Since $\log p(\mathcal{D})$ is constant we thus improve the approximate posterior, as measured under the KL divergence, by maximising $\mathcal{L}(\phi)$. And because the KL divergence is 0 if and only if

$q_\phi(\theta) = p(\theta \mid \mathcal{D})$ we a guaranteed to recover the true posterior when $\mathcal{L}(\phi) = \log p(\mathcal{D})$.

When the model holds parameters not encompassed by θ (e.g. the generator parameters in variational autoencoders (Kingma and Welling [2014]) or hyperparameters in sparse variational Gaussian processes (Titsias [2009])), the maximisation of the ELBO may also result in an increase of the true model evidence. This, of course, depends on the tightness of the bound and how changes to the variational posterior affect the remaining model parameters (Cremer et al. [2018]). I.e. under an inappropriate proposal distribution, the hyperparameters that maximise $\mathcal{L}(\phi)$ will not match those that maximise the true model evidence. For the same reason we cannot generally rely on the ELBO as a solid metric for model selection.

To highlight the main differences between MC methods and VI, we see that the former is stochastic while the latter is deterministic. The predictive capacity of VI depends on the proposal distribution's ability to emulate the true posterior, and there is thus a hard bound on the accuracy that can be achieved through VI estimation. MC sampling, on the other hand, is asymptotically exact. In terms of computational complexity, it may require a large amount of samples to arrive at an adequate approximation using sampling methods, especially when dealing with large dimensionality. Here, VI is typically the more efficient option.

2.3.3 Stochastic variational inference

Even though we are free to choose the proposal distribution so as to achieve a simpler optimisation task, we may still find the ELBO to be intractable, especially when working with a large number of observations. When the model can be written as:

$$p(\mathbf{x}, \mathbf{z}, \beta) = p(\beta) \prod_{i=1}^{N} p(x_i \mid z_i, \beta) p(z_i), \tag{2.1}$$

where $\mathbf{x} = \{x_i\}_{i=1}^N$ are our observations; $\mathbf{z} = \{z_i\}_{i=1}^N$ are local variables, each only influencing the distribution for a single observation; and β are global parameters, we can apply stochastic variational inference (SVI) to significantly speed up optimisation (Hoffman et al. [2013]). Here we construct a proposal distribution that follows a mean-field assumption, meaning that all parameters are independent:

$$q_\phi(\mathbf{z}, \beta) = q_\phi(\beta) \prod_{i=1}^{N} q_\phi(z_i).$$

This allows us to move the expectation inside the summation over log densities in the ELBO and decompose the KL divergences w.r.t. local variables:

$$\begin{aligned}\mathcal{L}(\phi) = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\beta)} \left[\mathbb{E}_{q_\phi(z_i)} \left[\log p(x_i \mid z_i, \beta)\right]\right] \\ - \sum_{i=1}^{N} \mathrm{KL}\left[q_\phi(z_i) \parallel p(z_i)\right] - \mathrm{KL}\left[q_\phi(\beta) \parallel p(\beta)\right].\end{aligned}$$

This does not entirely solve our problem since we need to evaluate the expectations for all datapoints w.r.t. $q_\theta(z_i)$ to obtain the gradient for β. However, we can now apply Monte Carlo methods by sampling from the data population and thus obtaining noisy, unbiased gradients. We can then optimise the variational parameters, ϕ, by appealing to the reparameterisation trick or score function estimation.

2.4 Bayesian optimisation

Often the predictions that we obtain from a model are just the means to an end, whilst the actual goal is to use those predictions for making informed decisions. The canonical way of making decisions in a probabilistic framework is to specify a cost function, $c : \mathcal{Y} \to \mathbb{R}$, that maps outcomes to some measure of risk, and then choose from an action space, $\mathcal{A}$, the action that minimises the expected risk (MacKay [2003]):

$$a^\star = \arg\min_{a \in \mathcal{A}} \int c(y) p(y \mid a) \, \mathrm{d}y, \tag{2.2}$$

where $p(y \mid a)$ is the predictive distribution over outcomes given action a. Here the quantification of epistemic uncertainty is critical. Any model that we rely on for making decisions should be acutely away of its own ignorance, which motivates the use of a Bayesian approach.

One popular application is Bayesian optimisation (Močkus [1975], Brochu et al. [2010], Shahriari et al. [2015]) where the goal is to minimise a blackbox function, $f : \mathcal{X} \to \mathcal{Y}$, that is unknown in the sense that we can only interact with it by making queries and retrieving its outputs. We usually also assume that evaluations are expensive in terms of e.g. time or financial cost, and that observations are corrupted by noise: $y_i = f(\mathbf{x}_i) + \epsilon$. On a limited querying budget (usually tens to hundreds of evaluations) we now want to retrieve the optimal input, $\mathbf{x}^\star$. Use cases include optimising the hyperparameters in machine learning models (Snoek et al. [2012]); policy search in reinforcement learning (Brochu et al. [2010]); sensor selection for meteorological modelling (Garnett et al. [2010]); and compiler configuration (Shah and Ghahramani [2016]).

We can cast the optimisation of f as a decision problem by defining a cost function such as simple regret:

$$c(\mathbf{y}_{1:N}) = \min \sum_{i=1}^{N} y_i - y_\star,$$

where

$$y_\star = \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}), \qquad \mathbf{y}_{1:N} = [y_1, y_2, \ldots, y_N]^T,$$

and then minimising (2.2) where the action space is sequences of inputs, $\mathcal{A} = \mathcal{X}^N$. This objective, however, requires us to propagate predictions through N steps of evaluations and is typically intractable.

In Bayesian optimisation we instead take a myopic approach where, in each iteration, we select the input that we believe to be most promising under some heuristic. This is formalised in an *acquisition function*, $\alpha : \mathcal{X} \to \mathbb{R}$, that maps inputs to some measure of querying utility under the predictive model. Formally we place a prior on f and derive the posterior conditioned on all samples collected up until iteration i:

$$p(f \mid \mathbf{y}_{1:i}, \mathbf{X}_{1:i}) = \frac{p(\mathbf{y}_{1:i} \mid f)p(f \mid \mathbf{X}_{1:i})}{p(\mathbf{y}_{1:i} \mid \mathbf{X}_{1:i})}.$$

Using this as our *surrogate model* for the true blackbox function, f, we then rely on the predictive posterior for finding the input that maximises α:

$$\mathbf{x}_{i+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \ \alpha(\mathbf{x}; \mathcal{D}_{1:i}),$$

where $\mathcal{D}_{1:i} = (\mathbf{X}_{1:i}, \mathbf{y}_{1:i})$. After having collected the new sample, $y_{i+1} = f(\mathbf{x}_{i+1}) + \epsilon$, we update our posterior belief over f and repeat until we meet some termination criterion.

This introduces a secondary optimisation task, namely that of optimising α in between queries to the blackbox function. We therefore require both our surrogate model and acquisition function to be relatively cheap to evaluate and easy to optimise. Ideally they are both differentiable w.r.t. the input so that we may obtain the partial derivative:

$$\alpha'(\mathbf{x}; \mathcal{D}) = \xi'(p(f \mid \mathbf{x}, \mathcal{D})) \frac{\partial p(f \mid \mathbf{x}, \mathcal{D})}{\partial \mathbf{x}},$$

where we define $\xi(p(f \mid \mathbf{x}, \mathcal{D})) = \alpha(\mathbf{x}; \mathcal{D})$.

In principle we can use any probabilistic model that yields a predictive distribution over f in the Bayesian optimisation framework, but the de facto standard is the Gaussian process (Rasmussen and Williams [2006]) due to its data efficiency, tractability, and robust uncertainty estimation. However, other models have also been explored, e.g. random forests

(Hutter et al. [2011]), Student-t processes (Shah et al. [2013]), and Bayesian neural networks (Springenberg et al. [2016]).

The catalogue of available acquisition functions, on the other hand, is more varied. Generally we will want the acquisition function to balance exploration, i.e. collecting inputs from unexplored regions to learn about f, and exploitation, i.e. using the limited querying budget to obtain good samples. Two of the most popular choices are Expected Improvement and Lower Confidence Bound, which we review in Chapter 4.

2.5 Gaussian processes

As mentioned in the introduction a popular approach in contemporary machine learning is to construct a very flexible parametric model and curb the behaviour by training on vast amounts of data. This relieves the model designer of the burden of encoding anything but the most vague assumptions about the behaviour of the latent function. The caveats are complex models that are difficult to interpret, use more training time and data than should arguably be necessary, and lack solid guarantees which makes them vulnerable to critical prediction errors (Szegedy et al. [2013], Lynch et al. [2017]).

Bayesian nonparametrics (Ghosh and Ramamoorthi [2003]) offer a complimentary approach wherein the complexity and capacity of the model grow as a function of the training data. Formally, these models define a prior over an infinite dimensional parameter space that in the posterior can be marginalised out to only contain dimensions corresponding to the observed data. The ability of the model to adapt its complexity means that the model designer does not have to worry about under- or overfitting; provided that the prior is sensible, the model will only be as flexible as what is supported and required by the data.

One prominent member of the Bayesian nonparametric family is the Gaussian process (GP) (Rasmussen and Williams [2006]) which specifies a prior over functions that may explain the mapping from a set of inputs, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathcal{X}$ to their corresponding, real outputs, $\mathbf{f} = \{f_i\}_{i=1}^N \in \mathbb{R}^N$ where we use the shorthand $f_i = f(\mathbf{x}_i)$. They can be viewed as the generalisation of a multivariate normal distribution when the dimensionality tends to infinity. Formally, a GP is defined as a collection of random variables, any finite subset of which is jointly normally distributed as such:

$$\begin{aligned} p(f_1, f_2, \dots, f_N \mid \mathbf{X}) &= \mathcal{N}\left(\boldsymbol{\mu}, \mathbf{K}\right), \\ \mu_i &= \mathbb{E}[f_i] = m(\mathbf{x}_i), \\ k_{ij} &= \text{Cov}[f_i, f_j] = \kappa(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \tag{2.3}$$

where the statistics are determined by the GP's mean function, $m : \mathcal{X} \to \mathbb{R}$, and kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Commonly we use a zero-mean function, i.e. $m(\mathbf{x}_i) = 0,\ \forall i$, which does not restrict the set of representable functions specified by the prior but simplifies both notation and analysis – we will take this approach throughout this thesis. The function, f, is then the infinite vector obtained by mapping from all elements in $\mathcal{X}$, which of course is not explicitly available. However, due to the marginalisation property of the multivariate normal distribution we can reason about any finite subset of evaluations of f, as given in (2.3). This is equivalent to first constructing the infinite vector and then marginalising out all values except those pertaining to $\mathbf{X}$, which is how the GP conforms to the nonparametric definition of specifying an infinite dimensional prior.

The GP effectively encodes the assumption that any pair of inputs, that are "close" in kernel space, should have similar function evaluations. This is arguably a more interpretable prior than e.g. stating that the weights of a Bayesian neural network should follow a standard Gaussian distribution (Gal and Ghahramani [2016]), and it affords us much agency in constricting the a priori hypothesis space given by $p(\mathbf{f} \mid \mathbf{X})$. If $\mathbf{X} \in \mathbb{R}^{N \times D}$ and we believe that a good similarity metric is the Euclidean distance, we may opt for e.g. the radial basis function (RBF) kernel:

$$\kappa_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(- \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2\ell_d^2} \right),$$

where the subscript denotes the input dimension. Here, the kernel variance, σ^2, and characteristic lengthscale, $\ell = \{\ell_d\}_{d=1}^{D}$, are hyperparameters that determine respectively the scaling of the function and the effective distance of influence between points. Note also that this kernel assumes stationarity, meaning that the covariance between any pair of points is determined solely by their distance. This is a fairly standard assumption but may prove too restrictive for certain problems – we will return to such a scenario in Chapter 5.

Any kernel function is required to produce valid covariance matrices. This entails that the matrix obtained by applying the kernel to any set of distinct inputs is positive definite (PD):

$$\mathbf{v}^T \kappa(\mathbf{X}) \mathbf{v} > 0, \qquad \forall\, \mathbf{v} \in \mathbb{R}^N, \mathbf{X} \subseteq \mathcal{X},$$

where we use the shorthand

$$\kappa(\mathbf{X}) = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \kappa(\mathbf{x}_N, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}.$$

When this requirement is upheld we say that κ is PD. Conveniently, positive definiteness is closed under addition and multiplication meaning that given two PD kernels, κ_1 and κ_2, we can construct new, valid kernels as such:

$$\kappa_+(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}') + \kappa_2(\mathbf{x}, \mathbf{x}'), \qquad \kappa_\times(\mathbf{x}, \mathbf{x}') = \kappa_1(\mathbf{x}, \mathbf{x}') \times \kappa_2(\mathbf{x}, \mathbf{x}').$$

To incorporate the Gaussian process prior into our Bayesian inference framework we need to specify a data likelihood for our observations, $p(\mathbf{y} \mid \mathbf{f})$, and derive the posterior, $p(\mathbf{f} \mid \mathbf{y}, \mathbf{X})$. The difficulty of this task depends solely on our choice of data likelihood. When learning a regression model, we obtain tractability by assuming that any output observation is simply a function evaluation corrupted by independent, Gaussian noise: $y_i = f_i + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$. This yields the expressions

$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \Sigma_y), \qquad p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K} + \Sigma_y),$$

where Σ_y is a diagonal matrix whose entries are given by σ_i^2. Importantly, this likelihood is conjugate to the prior so that the posterior is itself a multivariate Gaussian:

$$p(\mathbf{f} \mid \mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}(\mathbf{K} + \Sigma_y)^{-1}\mathbf{y}, \mathbf{K}(\mathbf{K} + \Sigma_y)^{-1}\Sigma_y\right).$$

Furthermore, for any new input, $\mathbf{x}_\star$, we have by definition that the joint distribution between observed outputs and the latent function evaluation is again a multivariate normal:

$$p(f_\star, \mathbf{y} \mid \mathbf{x}_\star, \mathbf{X}) = \mathcal{N}\left(\begin{bmatrix} f_\star \\ \mathbf{y} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} k_{\star\star} & \mathbf{k}_\star^T \\ \mathbf{k}_\star & \mathbf{K} + \Sigma_y \end{bmatrix}\right),$$

where $\mathbf{k}_\star = [\kappa(\mathbf{x}_\star, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}_\star, \mathbf{x}_N)]^T$ and $k_{\star\star} = \kappa(\mathbf{x}_\star, \mathbf{x}_\star)$. This yields a closed-form expression for the conditional distribution over $f_\star$:

$$p(f_\star \mid \mathbf{x}_\star, \mathbf{y}, \mathbf{X}) = \mathcal{N}\left(f_\star \mid \mathbf{k}_\star^T(\mathbf{K} + \Sigma_y)^{-1}\mathbf{y}, k_{\star\star} - \mathbf{k}_\star^T(\mathbf{K} + \Sigma_y)^{-1}\mathbf{k}_\star\right),$$

which serves as our prediction for new inputs. Typically we will simplify this further by using a homoscedastic noise model where the likelihood variance, σ_i^2, is identical for all observations. This will be the default choice in this thesis.

Note that the predictive variance increases with the magnitude of $\mathbf{k}_\star$. In effect, the GP has the attractive behaviour of assigning higher uncertainty to points that are dissimilar from all current observations. Additionally, the model naturally disentangles the aleatoric uncertainty, given by Σ_y, and the epistemic uncertainty, given by the variance of $p(f_\star \mid \mathbf{x}_\star, \mathbf{y}, \mathbf{X})$.

The evidence term, $p(\mathbf{y} \mid \mathbf{X})$, serves as a measure of plausibility of the GP prior and can be used as a model selection criterion for e.g. kernel design (Duvenaud et al. [2013]). Furthermore, when we have free hyperparameters in the kernel and/or data likelihood, we can use the evidence as the optimisation objective. This is amenable to numerical optimisation since all operations required for calculating $p(\mathbf{y} \mid \mathbf{X})$ are differentiable.

One of the main caveats of GP training and prediction is the complexity as a function of observed points. This stems from the calculation of the inverse and the determinant of the covariance matrix, both of which require $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory. However, recent progress has been made in alleviating these issues by speeding up the matrix calculations to enable exact inference for millions of points (Wang et al. [2019]) or applying asymptotically exact mini-batch optimisation (Chen et al. [2020]). Besides the number of datapoints, there are two other factors relevant to this thesis that may render the evidence and predictive posterior intractable. First is the use of non-conjugate data likelihoods, e.g. the Student-t distribution for regression or the Bernoulli distribution for binary classification. Exclusively relying on Gaussian likelihoods would entail that the model is only applicable for regression, and that we always assume normal posteriors for function evaluations which may prove too restrictive for noisy datasets (Jylänki et al. [2011]). Second is the setting where inputs are uncertain, e.g. when they are the result of noise corrupted measurements (Girard and Murray-Smith [2005]), outputs of another GP (Damianou and Lawrence [2013]), or latent variables to be inferred along with the function (Lawrence [2004]). The perhaps most popular approach to re-establish tractability in the face of large datasets, non-conjugate likelihoods, and/or uncertain inputs is the methodology of sparse GP's (Quiñonero-Candela and Rasmussen [2005]), which we review in Section 2.7.

2.6 Multitask Gaussian processes

By default the GP is a prior over scalar functions. This can, however, easily be generalised to multitask GP's (MTGP) where K function evaluations are associated with any particular input. This is useful when we expect one output dimension (or task) to be informative w.r.t. to another, and it may be necessary when we have insufficient data for modelling each dimension independently. The common approach for correlating tasks within the GP paradigm is to model all evaluations across dimensions with a single multivariate normal distribution. We will encounter MTGP's throughout the thesis, and so this section provides a brief overview over the models that may be applicable for a given multitask problem. We leave the data likelihood, pertaining to a given MTGP model, unspecified for now and focus solely on the derivation of the latent function.

The taxonomy of MTGP's can broadly be divided into those that assume isotopic data, where all outputs are evaluated for the same set of inputs, and heterotopic data, where different outputs are not restricted to sharing inputs. The former case will be relevant in e.g. deep Gaussian processes, where multiple "layers" of MTGP's are stacked to create a more flexible prior over functions (Damianou and Lawrence [2013]). Another example is global optimisation of an unknown function with multiple outputs, where a MTGP may be employed as surrogate model (Shah and Ghahramani [2016], Astudillo and Frazier [2019]). The need for a heterotopic model may arise in e.g. preference learning (Chu and Ghahramani [2005]), where we want to learn multiple, correlated functions pertaining to different users who have not necessarily assessed the same inputs.

A common approach for introducing correlation between dimensions is to view a particular output as the linear combination of Q latent, independent GP functions, $[u_1, \dots, u_Q]$, where $Q \leq K$. The perhaps most popular formulation is the linear coregionalisation model (LCM) (Journel and Huijbregts [1976]), originally developed within the field of geostatistics. Here we consider the i'th evaluation of the k'th output dimension to be given by:

$$f_i^{(k)} = \sum_{q=1}^{Q} \sum_{j=1}^{R_q} a_{q,k}^{(j)} u_q^{(j)}(\mathbf{x}_i).$$

That is, for each latent process, u_q, we draw R_q samples and let the k'th output dimension be their sum, weighted by free parameters $a_{k,q}^{(j)}$ for $1 \leq j \leq R_q$. Defining the $K \times R_q$ weight matrix $\mathbf{A}_q$ for process u_q:

$$\mathbf{A}_q = \begin{bmatrix} a_{q,1}^{(1)} & a_{q,1}^{(2)} & \cdots & a_{q,1}^{(R_q)} \\ a_{q,2}^{(1)} & a_{q,2}^{(2)} & \cdots & a_{q,2}^{(R_q)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{q,K}^{(1)} & a_{q,K}^{(2)} & \cdots & a_{q,K}^{(R_q)} \end{bmatrix},$$

we can write the entire K dimensional output vector for a given input, $\mathbf{x}_i$, succinctly as:

$$\mathbf{f}_i = \sum_{q=1}^{Q} \mathbf{A}_q \mathbf{u}_q(\mathbf{x}_i),$$

where $\mathbf{u}_q(\mathbf{x}_i) = [u_q^{(1)}(\mathbf{x}_i), \dots, u_q^{(R_q)}(\mathbf{x}_i)]^T$. Furthermore, the covariance matrix between two output vectors is given by:

$$\text{Cov}[\mathbf{f}_i, \mathbf{f}_j] = \sum_{q=1}^{Q} \mathbf{B}_q \kappa_q(\mathbf{x}_i, \mathbf{x}_j),$$

where $\mathbf{B}_q = \mathbf{A}_q \mathbf{A}_q^T$ is the *coregionalisation matrix* and $\kappa_q(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel function for u_q. Stacking all function vectors, $\mathbf{f} = [\mathbf{f}_1^T, \dots, \mathbf{f}_N^T]^T$, we then have:

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}\left(\mathbf{f} \,\middle|\, \mathbf{0}, \sum_{q=1}^{Q} \mathbf{K}_q \otimes \mathbf{B}_q\right), \tag{2.4}$$

where $\mathbf{K}_q$ is the kernel matrix for u_q and $\otimes$ is the Kronecker product.

From this model we can retrieve several simpler MTGP models. Using the same number of samples, R_q, for all latent functions, $[u_1, \ldots, u_Q]$, we have the semiparametric latent factor model (SLFM) (Seeger et al. [2005]). Meanwhile, if we only assume a single latent function from which we have drawn R samples, we obtain the intrinsic coregionalisation model (ICM) (Goovaerts et al. [1997], Williams et al. [2007]). And finally, by setting the coregionalisation matrix to identity in the ICM, we retrieve a MTGP that assumes independency between output dimensions. This is equivalent to just learning a separate GP for each dimension, all of which share the same kernel function.

The LCM and SFML are especially appropriate for isotopic data. When all dimensions are evaluated for the same N inputs, we only need to compute a single $N \times N$ covariance matrix for each latent function and then take advantage of the Kronecker structure in (2.4) to obtain the $NK \times NK$ covariance matrix across the entire function vector. The ICM model, on the other hand, is an obvious choice for heterotopic data. Here, we can equivalently write the covariance between any pair of evaluations as:

$$\text{Cov}[f^{(k)}(\mathbf{x}_i), f^{(\ell)}(\mathbf{x}_j)] = b_{k,l} \cdot \kappa(\mathbf{x}_i, \mathbf{x}_j),$$

where $b_{k,\ell}$ is the covariance between dimension k and ℓ. The main caveat is that all dimensions are forced to use the same kernel, meaning that we implicitly assume all tasks to adhere to the same function dynamics.

The above models all share some critical limitations. Firstly, the coregionalisation matrices may introduce a rather large number of free hyperparameters leading to a risk of overfitting when data are scarce. This can, as always, be alleviated in a Bayesian fashion as is done in e.g. Titsias and Lázaro-Gredilla [2011] where all matrix entries are associated with spike-and-slab priors and the posterior is approximated through VI. Secondly, they assume the presence of relatively few, salient processes. This is not a weakness in and of itself, but we may encounter settings where such an assumption is too restrictive. And thirdly, they always enforce a Gaussian predictive distribution. This, again, is often an appropriate characteristic, but in some modelling scenarios it may be preferable to allow for multimodal distributions when uncertainty is high. We will examine these limitations further in Chapter 5 in the context of MTGP's applied for preference learning.

2.7 Sparse Gaussian processes

The intractability of GP's when modelling large datasets was circumvented in early works by only conditioning on a relatively small subset of observations (Williams and Seeger [2001], Csató and Opper [2002], Lawrence et al. [2003]). This, in turn, lead to the combinatorial problem of identifying the most informative subset for representing the entire dataset. An elegant approach for solving this issue was proposed by Snelson and Ghahramani [2006] who lifted the restriction of having the conditioning points be part of the observed data, leading to the methodology of sparse Gaussian processes (SGP). Here we introduce a set of inducing points, or pseudo points, $(\mathbf{u}, \mathbf{Z})$, where $\mathbf{Z} = \{\mathbf{z}_j\}_{j=1}^M$ belong to the input domain, $\mathcal{X}$, and $\mathbf{u} = \{u_j\}_{j=1}^M$ are taken to be evaluations of the latent function, i.e. $u_j = f(\mathbf{z}_j)$. By definition these points are jointly Gaussian with the observed data:

$$p(\mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{f}\\ \mathbf{u}\end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix}\mathbf{K}_{NN} & \mathbf{K}_{NM}\\ \mathbf{K}_{MN} & \mathbf{K}_{MM}\end{bmatrix}\right),$$

where $\mathbf{K}_{NN}$ is the covariance pertaining to our N observed points, $\mathbf{K}_{MM}$ is the covariance between the M inducing points, and $\mathbf{K}_{NM} = \mathbf{K}_{NM}^T$ is the covariance between observed and inducing points. As for the predictive distribution given by the exact GP, we can now obtain the conditional for $\mathbf{f}$ in closed form:

$$p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{u}, \mathbf{K}_{NN} - \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}\right). \tag{2.5}$$

Crucially, the marginals of $\mathbf{f}$ in (2.5) only depend on the corresponding inputs from $\mathbf{X}$ as well as the inducing points:

$$\begin{aligned} p(f_i \mid \mathbf{X}, \mathbf{u}, \mathbf{Z}) = p(f_i \mid \mathbf{x}_i, \mathbf{u}, \mathbf{Z}) &= \mathcal{N}\left(\mathbf{f} \mid \kappa(\mathbf{x}_i, \mathbf{Z})^T\mathbf{K}_{MM}^{-1}\mathbf{u}, \lambda_i\right), \qquad (2.6)\\ \kappa(\mathbf{x}_i, \mathbf{Z}) &= [\kappa(\mathbf{x}_i, \mathbf{z}_1), \ldots, \kappa(\mathbf{x}_i, \mathbf{z}_M)]^T,\\ \lambda_i &= \kappa(\mathbf{x}_i, \mathbf{x}_i) - \kappa(\mathbf{x}_i, \mathbf{Z})^T\mathbf{K}_{MM}^{-1}\kappa(\mathbf{x}_i, \mathbf{Z}). \end{aligned}$$

This means that when we have $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{f}, \mathbf{I}\sigma_y^2\right)$, we can replace $p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{X})$ of the exact GP model with

$$p(\mathbf{y} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})p(\mathbf{u} \mid \mathbf{Z}) = \left[\int p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})\,\mathrm{d}\mathbf{f}\right] p(\mathbf{u} \mid \mathbf{Z}),$$

with the effect of obtaining a data likelihood that factorises across datapoints:

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) &= \prod_{i=1}^N \int p(y_i \mid f_i)p(f_i \mid \mathbf{u}, \mathbf{x}_i, \mathbf{Z})\,\mathrm{d}f_i \qquad (2.7)\\ &= \prod_{i=1}^N \mathcal{N}\left(y_i \mid \kappa(\mathbf{x}_i, \mathbf{Z})^T\mathbf{K}_{MM}^{-1}\mathbf{u}, \lambda_i + \sigma_y^2\right). \end{aligned}$$

Note that in order to calculate (2.7) we only need to compute the diagonal of the covariance in (2.5). This property lends the method its name: Fully Independent Training Conditionals

(FITS). The marginal likelihood is still available in closed form:

$$\begin{aligned}p(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}) &= \int p(\mathbf{y}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}) p(\mathbf{u} \mid \mathbf{Z}) \,\mathrm{d}\mathbf{u} \\ &= \mathcal{N}\left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN} + \Lambda + \mathbf{I}\sigma_y^2\right), \\ \Lambda &= \mathrm{diag}([\lambda_1, \ldots, \lambda_N]),\end{aligned}$$

which importantly only requires the inversion of rank M matrices and the diagonal, Λ, thus eliminating much of the complexity of the exact GP model. Furthermore, the marginal likelihood is differentiable w.r.t. both hyperparameters and inducing inputs, $\mathbf{Z}$, making it amenable to numerical optimisation. Compared with the exact GP model, we have now reduced the complexity to $\mathcal{O}(M^3N)$ time and $\mathcal{O}(M^2N)$ memory which for $M \ll N$ results in a substantial reduction. On the other hand we have introduced $M \cdot D$ new parameters, i.e. the inducing inputs, which may complicate optimisation and risk introducing overfitting unless they are associated with a prior (Rossi et al. [2021], Uhrenholt et al. [2021]).

To infer the optimal inducing outputs, $\mathbf{u}$, we can readily apply Bayes rule:

$$\begin{aligned}p(\mathbf{u} \mid \mathbf{y}, \mathbf{X}, \mathbf{Z}) &= \frac{p(\mathbf{y} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) p(\mathbf{u} \mid \mathbf{Z})}{p(\mathbf{y} \mid \mathbf{X}, \mathbf{Z})} \\ &= \mathcal{N}\left(\mathbf{u} \mid \mathbf{K}_{MM}\mathbf{Q}^{-1}\mathbf{K}_{MN}(\Lambda + \mathbf{I}\sigma_y^2)^{-1}\mathbf{y}, \mathbf{K}_{MM}\mathbf{Q}^{-1}\mathbf{K}_{MM}\right), \\ \mathbf{Q} &= \mathbf{K}_{MM} + \mathbf{K}_{MN}(\Lambda + \mathbf{I}\sigma_y^2)^{-1}\mathbf{K}_{NM}.\end{aligned} \tag{2.8}$$

New predictions are now made by using the posterior of (2.8) to integrate out the conditional distribution given by (2.6) for novel input, which only requires $\mathcal{O}(M)$ time and $\mathcal{O}(M^2)$ memory, providing that $\mathbf{Q}^{-1}$ has been cached.

The advantage of this approach is primarily the decoupling of data points in (2.7) and the reduction in time and space complexity at training and inference time. However, it comes at the price of replacing the joint distribution, $p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{X})$, with the sparse approximation, $p(\mathbf{y} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})p(\mathbf{u} \mid \mathbf{Z})$, which is simultaneously much less expressive and more heavily parameterised than the exact GP. While approaches have been proposed for increasing the predictive capacity by e.g. re-introducing correlation between individual function evaluations in (2.6) (Quiñonero-Candela and Rasmussen [2005]), the fundamental limitation remains that the generative model relies explicitly on the inducing points. This was done away with in the variational re-interpretation of Titsias [2009] which we turn to next.

2.7.1 Sparse variational Gaussian processes

The sparse variational Gaussian process (SVPG) method leaves the generative model unchanged – i.e. we still assume that the data are generated via an exact GP – and instead augments the collection of latent function evaluations with the, initially redundant, inducing points:

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}) = p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}).$$

We stress that this does not constitute a change from the exact GP model since $\mathbf{f}$ is a sufficient statistic for $\mathbf{y}$, and so:

$$p(\mathbf{f} \mid \mathbf{y}, \mathbf{u}, \mathbf{X}, \mathbf{Z}) = p(\mathbf{f} \mid \mathbf{y}, \mathbf{X}) = \frac{\int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}) \, \mathrm{d}\mathbf{u}}{p(\mathbf{y} \mid \mathbf{X})}.$$

The novelty of this method is to instead take a variational approach and introduce a proposal distribution over the latent variables, $q(\mathbf{f}, \mathbf{u})$, that should approximate the true posterior, $p(\mathbf{f}, \mathbf{u} \mid \mathbf{y}, \mathbf{X}, \mathbf{Z})$. Specifying

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u} \mid \mathbf{Z}),$$

where $p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})$ is defined as in (2.5), we now obtain a much simplified ELBO:

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{X}) &= \log \int \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}) \, \mathrm{d}[\mathbf{f}, \mathbf{u}] \\
&\geq \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z})}{q(\mathbf{f}, \mathbf{u})} \, \mathrm{d}[\mathbf{f}, \mathbf{u}] \\
&= \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) \cancel{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})} p(\mathbf{u} \mid \mathbf{Z})}{\cancel{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})} q(\mathbf{u} \mid \mathbf{Z})} \, \mathrm{d}[\mathbf{f}, \mathbf{u}] \\
&= \mathbb{E}_{q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z})} \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] - \mathrm{KL} \left[q(\mathbf{u} \mid \mathbf{Z}) \parallel p(\mathbf{u} \mid \mathbf{Z}) \right], \qquad (2.9)
\end{aligned}$$

where we define:

$$q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z}) = \int p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u} \mid \mathbf{Z}) \, \mathrm{d}\mathbf{u},$$

which is analytically tractable when $q(\mathbf{u} \mid \mathbf{Z})$ is normal. Predictions for new datapoints now proceed with the approximation:

$$\begin{aligned}
p(f_* \mid \mathbf{x}_*, \mathbf{y}, \mathbf{X}) &= \int p(f_* \mid \mathbf{x}_*, \mathbf{f}, \mathbf{X}) p(\mathbf{f} \mid \mathbf{y}, \mathbf{X}) \, \mathrm{d}\mathbf{f} \\
&\approx \int p(f_* \mid \mathbf{x}_*, \mathbf{f}, \mathbf{X}) q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z}) \, \mathrm{d}\mathbf{f}. \\
&= \int p(f_* \mid \mathbf{x}_*, \mathbf{u}, \mathbf{Z}) q(\mathbf{u} \mid \mathbf{Z}) \, \mathrm{d}\mathbf{u}.
\end{aligned}$$

Titsias [2009] derives the collapsed bound that maximises the ELBO when the data likelihood is an isotropic Gaussian, i.e. $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{f}, \mathbf{I}\sigma_y^2\right)$:

$$\begin{aligned} q(\mathbf{u} \mid \mathbf{Z}) &= \mathcal{N}\left(\mathbf{u} \mid \boldsymbol{\mu}_{\mathbf{u}}, \Sigma_{\mathbf{u}}\right), \\ \Sigma_{\mathbf{u}}^{-1} &= \sigma_y^{-2}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}\mathbf{K}_{NM}\mathbf{K}_{MM}^{-1} + \mathbf{K}_{MM}^{-1}, \\ \boldsymbol{\mu}_{\mathbf{u}} &= \sigma_y^{-2}\Sigma_{\mathbf{u}}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}\mathbf{y}. \end{aligned}$$

Besides only applying for one specific family of likelihoods, this formulation also suffers from a restrictive complexity for large N since $\boldsymbol{\mu}_{\mathbf{u}}$ and $\Sigma_{\mathbf{u}}$ rely on all observations. It is furthermore incompatible with SVI because the proposal distribution does not adhere to the required form of (2.1). To solve this, Hensman et al. [2013] introduce the more general proposal distribution:

$$q(\mathbf{u} \mid \mathbf{Z}) = \mathcal{N}\left(\mathbf{u} \mid \mathbf{m}, \mathbf{S}\right),$$

where $\mathbf{m}$ and $\mathbf{S}$ are free variational parameters. Since the likelihood for any individual datapoint, y_i, now only relies on the global parameter, $(\mathbf{m}, \mathbf{S})$, and the associated input, $\mathbf{x}_i$ (as can be seen from (2.6)), we can decompose the expectation $\mathbb{E}_{q(\mathbf{f}\mid\mathbf{X},\mathbf{Z})}\left[\log p(\mathbf{y} \mid \mathbf{f})\right]$ across datapoints and apply mini-batch optimisation. Furthermore, we no longer have any restrictions on type of likelihood for $p(\mathbf{y} \mid \mathbf{f})$, meaning that this formulation enables a wide range of modelling tasks such as classification (Hensman et al. [2015a]), ordinal regression (Sheth et al. [2015]), and Poisson regression for count data (Law et al. [2018]).

Here, the main limitation is arguably that the expressiveness of the proposal distribution is completely reliant on the inducing points, which bound both the ELBO and the accuracy of new predictions. We can thus only increase flexibility by adding more inducing points, which will eventually make the $\mathcal{O}(M^2)$ time complexity infeasible. In Chapter 3 we develop a method for instead learning M from the data by including the selection of inducing points in the inference process. Another limitation in terms of flexibility is the assertion that the approximate posterior distribution for all function evaluations must be Gaussian, which is decidedly false when using anything but a Gaussian likelihood. This assumption is treated in Chapter 5 in the context of sparse MTGP's.

2.7.2 Gaussian process latent variable modelling

Unsupervised learning aims at finding an informative, and often low-dimensional, representation of one's data. A popular approach is latent variable modelling where we assume that our high-dimensional observations are generated from some low-dimensional representation. This intuitively encourages the model to capture the salient factors of variation in the data, which hopefully correspond to high-semantic features that are useful for data analysis, visualisation, or downstream modelling tasks. Parametric approaches include Bayesian principal component analysis (Bishop [2006]), variational autoencoders (Kingma and Welling [2014]), and generative adversarial networks (Goodfellow et al. [2014]).

In the GP framework we can construct a latent variable model (GPLVM) by associating latent inputs, $\mathbf{X} \in \mathbb{R}^{N \times Q}$, with a prior (Lawrence [2004], Titsias and Lawrence [2010]). We thus assume that our observed outputs, $\mathbf{Y} \in \mathbb{R}^{N \times D}$, result from the generative process:

$$\begin{aligned} \mathbf{X} &\sim p(\mathbf{X}), \\ \mathbf{F} \mid \mathbf{X} &\sim \mathcal{GP}(\mathbf{0}, \kappa), \\ \mathbf{Y} \mid \mathbf{F} &\sim p(\mathbf{Y} \mid \mathbf{F}), \end{aligned}$$

where $\mathbf{F}$ and $\mathbf{Y}$ are capitalised to reflect that both have multiple outputs. A common choice is to set $p(\mathbf{X}) = \mathcal{N}(\mathbf{X} \mid \mathbf{0}, \mathbf{I})$ and use a MTGP with independent outputs and shared covariance for $p(\mathbf{F} \mid \mathbf{X})$. Inference now proceeds by deriving the posterior:

$$p(\mathbf{F}, \mathbf{X} \mid \mathbf{Y}) = \frac{p(\mathbf{Y} \mid \mathbf{F}) p(\mathbf{F} \mid \mathbf{X}) p(\mathbf{X})}{\int p(\mathbf{Y} \mid \mathbf{F}) p(\mathbf{F} \mid \mathbf{X}) p(\mathbf{X}) \, \mathrm{d}(\mathbf{F}, \mathbf{X})}.$$

In contrast to the supervised setting, where $\mathbf{X}$ is non-stochastic, we cannot evaluate the evidence in closed-form due to the intractability of marginalising over the random covariance matrices in $p(\mathbf{F} \mid \mathbf{X})$. The approach originally proposed in Lawrence [2004] simply identifies a point estimate:

$$\mathbf{X}^\star = \underset{\mathbf{X}}{\arg\max} \; \log p(\mathbf{Y} \mid \mathbf{X}) + \log p(\mathbf{X}).$$

Although conceptually simple and easy to implement, this method does not provide uncertainty estimates for $\mathbf{X}$ nor access to the posterior for $\mathbf{F}$. In addition, the model has $\mathcal{O}(NQ)$ free parameters, leaving it susceptible to overfitting.

We can solve these challenges under the variational framework by following the same procedure as described in the previous section, but including $\mathbf{X}$ in the proposal distribution:

$$q(\mathbf{U}, \mathbf{F}, \mathbf{X}) = p(\mathbf{F} \mid \mathbf{U}, \mathbf{X}, \mathbf{Z}) q(\mathbf{U} \mid \mathbf{Z}) q(\mathbf{X}),$$

where $\mathbf{Z} \in \mathbb{R}^{M \times Q}$ and $\mathbf{U} \in \mathbb{R}^{M \times D}$ are the inducing inputs and outputs. The new proposal distribution, $q(\mathbf{X})$, can be chosen arbitrarily, although an isotropic Gaussian is common. The ELBO from (2.9) is then updated to

$$\log p(\mathbf{Y}) \geq \mathbb{E}_{q(\mathbf{F} \mid \mathbf{Z})} \left[\log p(\mathbf{Y} \mid \mathbf{F}) \right] - \mathrm{KL} \left[q(\mathbf{U} \mid \mathbf{Z}) \parallel p(\mathbf{U} \mid \mathbf{Z}) \right] - \mathrm{KL} \left[q(\mathbf{X}) \parallel p(\mathbf{X}) \right].$$

The expectation term implicitly marginalises over $\mathbf{X}$ in the proposal distribution:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{F}|\mathbf{Z})}\left[\log p(\mathbf{Y} \mid \mathbf{F})\right] &= \int q(\mathbf{F} \mid \mathbf{Z}) \log p(\mathbf{Y} \mid \mathbf{F})\,\mathrm{d}\mathbf{F} \\ &= \int \left[\int q(\mathbf{F} \mid \mathbf{X}, \mathbf{Z}) q(\mathbf{X})\,\mathrm{d}\mathbf{X}\right] \log p(\mathbf{Y} \mid \mathbf{F})\,\mathrm{d}\mathbf{F},\end{aligned}$$

which remains intractable in the general case, although certain kernels allow for a closed-form expression when $q(\mathbf{X})$ is normal (Titsias and Lawrence [2010]). In this thesis we will instead rely on stochastic variational inference and estimate the entire integral with Monte Carlo sampling, as is done in Salimbeni and Deisenroth [2017]:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{F}|\mathbf{Z})}\left[\log p(\mathbf{Y} \mid \mathbf{F})\right] &\approx \frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{Y} \mid \tilde{\mathbf{F}}^{(s)}), \\ \tilde{\mathbf{F}}^{(s)} &\sim q(\mathbf{F} \mid \tilde{\mathbf{X}}^{(s)}, \mathbf{Z}), \\ \tilde{\mathbf{X}}^{(s)} &\sim q(\mathbf{X}).\end{aligned}$$

Note that when the variational distribution for the latent inputs adheres to a mean-field assumption, i.e. $q(\mathbf{X}) = \prod_{i=1}^{N} q(\mathbf{x}_i)$, the entire expression decomposes over individual datapoints, yielding a highly scalable model that is amenable to mini-batch optimisation.

2.7.3 Deep Gaussian processes

The SVGP and GPLVM can be further generalised to the composition of multiple GP's, yielding a deep Gaussian process (DGP) (Damianou and Lawrence [2013]). This is inspired by the tremendous success of deep neural networks (Goodfellow et al. [2016]) where intermediate feature representations in the hidden layers of a deep model are learnt in an autonomous fashion. For the DGP specifically it broadens the family of representable functions encoded in the prior when compared to a standard, "shallow" GP. An example is shown in Figure 2.3 where the aim is to learn a simple step function, $f(x) = \mathbb{I}(x > 0)$. The discontinuity at $x = 0$ is difficult to handle for a stationary GP due to the implicit assumption of smoothness enforced by the kernel. In addition, the GP imposes a normal posterior which seems unreasonable for a dataset where any outcome has one of two values. However, in the DGP the correlation between any pair of evaluations depends, not only on the kernel of the last layer, but on the transformations incurred by the layers preceding it. This effectively lifts the restriction of stationarity and enables more attenuated jumps in the posterior function. Furthermore, the composition of multiple Gaussian distributions, as obtained through the deep structure, leads to a non-Gaussian output. This accounts for the bimodal distribution outside of the observed data in Figure 2.3.

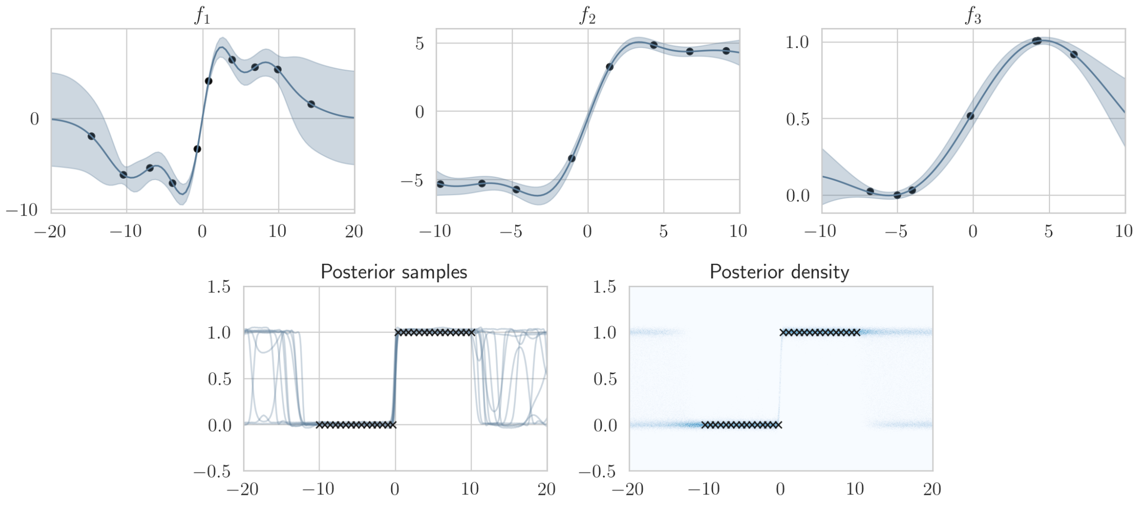

Figure 2.3: A three layer DGP employed for learning a step function. The three top plots show the approximate posterior of each SVGP layer, where the black dots are the inducing points. The lower left plot shows 10 samples drawn from the posterior along with the observations (black crosses). The lower right is the posterior density for 200 samples. Note that even though each layer by construction outputs Gaussian marginals, the propagation of uncertainty through the layers results in a bimodal distribution outside the observed region.

The probabilistic model for an L-layer DGP is specified as the joint distribution:

$$p(\mathbf{Y}, \mathbf{F}_1, \ldots, \mathbf{F}_L \mid \mathbf{X}) = p(\mathbf{Y} \mid \mathbf{F}_L) \prod_{\ell=1}^{L} p(\mathbf{F}_\ell \mid \mathbf{F}_{\ell-1}),$$
$$p(\mathbf{F}_\ell \mid \mathbf{F}_{\ell-1}) = \mathcal{N}\left(\mathbf{F}_\ell \mid \mathbf{0}, \kappa_\ell(\mathbf{F}_{\ell-1}, \mathbf{F}_{\ell-1})\right),$$

where we use subscript to denote layer index (rather than observation index) and define $\mathbf{F}_0 = \mathbf{X}$ for notational convenience. Each layer is thus a MTGP that takes the outputs of the previous layer as input. As was the case for the GPLVM, we cannot marginalise out the stochastic function evaluations in the intermediate layers. To solve this, we again rely on the sparse variational framework by augmenting each layer with inducing variables, $\{\mathbf{U}_\ell, \mathbf{Z}_\ell\}_{\ell=1}^{L}$, and introducing a proposal distribution:

$$q(\mathbf{F}_1, \ldots, \mathbf{F}_L, \mathbf{U}_1, \ldots, \mathbf{U}_L) = \prod_{\ell=1}^{L} p(\mathbf{F}_\ell \mid \mathbf{U}_\ell, \mathbf{F}_{\ell-1}, \mathbf{Z}_\ell) q(\mathbf{U}_\ell \mid \mathbf{Z}_\ell)$$
$$\triangleq \mathcal{Q}.$$

As in the standard SVGP derivation, this factorisation causes cancellations in the ELBO that decouples latent function evaluations across observations:

$$\begin{aligned}\log p(\mathbf{Y} \mid \mathbf{X}) &\geq \int \mathcal{Q} \log \left[p(\mathbf{Y} \mid \mathbf{F}_L) \prod_{\ell=1}^{L} \left(\frac{\cancel{p(\mathbf{F}_\ell \mid \mathbf{U}_\ell, \mathbf{F}_{\ell-1}, \mathbf{Z}_\ell)} p(\mathbf{U}_\ell \mid \mathbf{Z}_\ell)}{\cancel{p(\mathbf{F}_\ell \mid \mathbf{U}_\ell, \mathbf{F}_{\ell-1}, \mathbf{Z}_\ell)} q(\mathbf{U}_\ell \mid \mathbf{Z}_\ell)} \mathrm{d}\{\mathbf{F}_\ell, \mathbf{U}_\ell\} \right) \right] \\ &= \mathbb{E}_{\mathcal{Q}} \left[\log p(\mathbf{y} \mid \mathbf{F}_L) \right] - \sum_{\ell=1}^{L} \mathrm{KL} \left[q(\mathbf{U}_\ell \mid \mathbf{Z}_\ell) \parallel p(\mathbf{U}_\ell \mid \mathbf{Z}_\ell) \right].\end{aligned}$$

Here, we can optionally include a prior and variational distribution for $\mathbf{X}$ to obtain a deep version of the GPLVM. When $q(\mathbf{U}_\ell \mid \mathbf{Z}_\ell)$ are all Gaussian, the KL divergences become analytically tractable. This leaves the expectation term whose distribution includes the marginalisation of both inducing points and function evaluations for all L layers. Relying on the doubly stochastic formulation of Salimbeni and Deisenroth [2017] we propagate samples through the layers to jointly perform Monte Carlo integration for all nested expectations:

$$\begin{aligned}\mathbb{E}_{\mathcal{Q}} \left[\log p(\mathbf{Y} \mid \mathbf{F}_L) \right] &\approx \frac{1}{S} \sum_{s=1}^{S} \log p(\mathbf{Y} \mid \tilde{\mathbf{F}}_L^{(s)}, \mathbf{Z}_L), \\ \mathbf{F}_\ell^{(s)} &\sim q(\mathbf{F}_\ell \mid \tilde{\mathbf{F}}_{\ell-1}^{(s)}, \mathbf{Z}_L), \qquad \ell \in [1, \ldots, L], \\ q(\mathbf{F}_\ell \mid \tilde{\mathbf{F}}_{\ell-1}^{(s)}, \mathbf{Z}_L) &= \int p(\mathbf{F}_\ell \mid \mathbf{U}_\ell, \tilde{\mathbf{F}}_{\ell-1}^{(s)}, \mathbf{Z}_L) q(\mathbf{U}_\ell \mid \mathbf{Z}_\ell) \, \mathrm{d}\mathbf{U}_\ell.\end{aligned}$$

As was the case for the uncollapsed SVGP, the distribution for any function output only depends on its associated inputs as well as the inducing points of that layer. That is, letting $\mathbf{f}_{\ell,i}$ be the output vector for observation i in layer ℓ, we only need to consider $\mathbf{f}_{\ell-1,i}$, $\mathbf{Z}_\ell$, and $\mathbf{U}_\ell$ to find the variational statistics for $\mathbf{f}_{\ell,i}$:

$$\begin{aligned}q(\mathbf{F}_\ell \mid \mathbf{F}_{\ell-1}, \mathbf{Z}_\ell) &= \prod_{i=1}^{N} \mathcal{N} \left(\mathbf{f}_{\ell,i} \mid \boldsymbol{\mu}_{\ell,i}, \boldsymbol{\beta}_{\ell,i} \right) \\ \boldsymbol{\mu}_{\ell,i} &= \boldsymbol{\alpha}_{\ell,i}^T \mathbf{m}_\ell \\ \boldsymbol{\beta}_{\ell,i} &= \kappa_\ell(\mathbf{f}_{\ell-1,i}, \mathbf{f}_{\ell-1,i}) - \boldsymbol{\alpha}_{\ell,i}^T \left(\kappa_\ell(\mathbf{Z}_\ell, \mathbf{Z}_\ell) - \mathbf{S}_\ell \right) \boldsymbol{\alpha}_{\ell,i} \\ \boldsymbol{\alpha}_{\ell,i} &= \kappa_\ell(\mathbf{Z}_\ell, \mathbf{Z}_\ell)^{-1} \kappa_\ell(\mathbf{Z}_\ell, \mathbf{f}_{\ell-1,i}),\end{aligned}$$

where $\mathbf{m}_\ell$ and $\mathbf{S}_\ell$ are the global, variational parameters pertaining to layer ℓ. This crucially implies that the approximate posterior for any output vector, $\mathbf{y}_i$, only depends on the distributions pertaining to that point, $\mathcal{N}(\mathbf{f}_{\ell,i} \mid \boldsymbol{\mu}_{\ell,i}, \boldsymbol{\beta}_{\ell,i})$, for each layer ℓ (Salimbeni and Deisenroth

[2017]). In turn, the expectation over $\log p(\mathbf{Y} \mid \mathbf{X})$ decomposes into a sum of N integrals:

$$\begin{aligned}\mathbb{E}_{\mathcal{Q}}\left[\log p(\mathbf{Y} \mid \mathbf{F}_L)\right] &= \sum_{i=1}^{N} \mathbb{E}_{q_i}\left[\log p(\mathbf{y}_i \mid \mathbf{f}_{L,i})\right] \\ q_i &= \prod_{\ell=1}^{L} \mathcal{N}\left(\mathbf{f}_{\ell,i} \mid \boldsymbol{\mu}_{\ell,i}, \boldsymbol{\beta}_{\ell,i}\right),\end{aligned}$$

which can be easily approximated using Monte Carlo sampling.

Chapter 3

Complexity reduction in sparse Gaussian processes

In this chapter we consider the role of inducing points in sparse variational Gaussian processes (SVGP) and the modelling assumptions they represent. We argue that, although elegant, the sparse framework possesses certain characteristics that contradict the principles of Bayesian inference and prohibits flexibility. By expanding the hierarchical model to include the selection of inducing points, we develop a simple and theoretically well-founded method for alleviating these problems, and we demonstrate empirically that the method finds immediate applicability in the design of standard SVGP regression models, deep Gaussian processes, and latent variable modelling.

3.1 Introduction

As described in Section 2.5 the defining quality of nonparametric models is their ability to increase complexity and capacity with the amount of observed data. For the Gaussian process model in particular, we specify a function prior that incorporates the data directly through the mean and kernel functions thereby building an increasingly complex model as observations are collected. In conjunction, the predictive capacity grows since the model can rely on more observations to inform the conditional function distribution for novel data. When compared to parametric approaches such as neural networks, this alleviates the model designer of the responsibility of choosing a number of free parameters that offers sufficient flexibility while avoiding overfitting; provided that the GP prior is sensible, the model will only be as flexible as what is supported and required by the data. In relation to model design, this is perhaps the strongest advantage of the nonparametric paradigm when compared to parametric models, where substantial effort and expertise must be dedicated to identifying the correct amount flexibility.

Sparse Gaussian processes, and the later extension to sparse variational Gaussian processes (SVGP), were originally aimed at solving scalability issues but have since found relevance in enabling esoteric models such as deep Gaussian processes (Damianou and Lawrence [2013]), latent variable models (Titsias and Lawrence [2010]), and convolutional kernels (van der Wilk et al. [2017]). Here we make the assumption that our observed data can be adequately explained by a small set of prototypical data, denoted pseudo points or inducing points. The task of model training now reduces to quantifying and minimising the discrepancy between the exact GP, that incorporates all observed data, and the sparse GP that only conditions on the inducing points. When the number of inducing points, M, is significantly smaller than the number of datapoints, N, we obtain a much more efficient model.

Naturally, the posterior approximation of the SVGP improves with M. At the same time, M is the main bottleneck in terms of computational complexity in sparse GP's.[1] Much work has been dedicated to making the inducing points either more informative (Lázaro-Gredilla and Figueiras-Vidal [2009], Lázaro-Gredilla et al. [2010]) or less computationally demanding (Wilson and Nickisch [2015], Salimbeni et al. [2018], Dutordoir et al. [2020]), but the fact remains that the more inducing points we can afford, the more accurate a model we may hope to achieve. Identifying a suitable choice for M is thus ultimately a trade-off between complexity and capacity.

Given that the GP model belongs to the family of Bayesian nonparametrics, we argue that the sparse paradigm falls victim to two conceptual issues. Firstly, it requires us to make an a priori choice about a quantity, of which we may be uncertain, when specifying the model. However, a core principle of Bayesian inference is to make our uncertainty explicit and derive a posterior that balances prior beliefs and the support of the data. Secondly, we now find ourselves in the very dilemma of parametric models that the nonparametric approach promised to solve. That is, by fixing the number of inducing points we also fix the model complexity and (potential) capacity, and our model no longer adapts to the data without direct intervention from the model designer. This may then prompt an architecture search for identifying an appropriate number of points that allow for accurate predictions without being overly demanding in terms of computational requirements – something that is a frequent exercise in neural network design but blissfully absent when specifying non-sparse Gaussian process models.

In this chapter we propose an elegant and theoretically well-motivated method for solving both of these issues. Following a Bayesian approach we absorb M into the model inference by updating the model specification so that the inducing points assumed to be sampled rather than deterministically chosen. This is achieved by associating the inducing points with a

[1]Here we do not refer to the model complexity in terms of degrees of freedom, but rather the memory and time required for training the model and making new predictions.

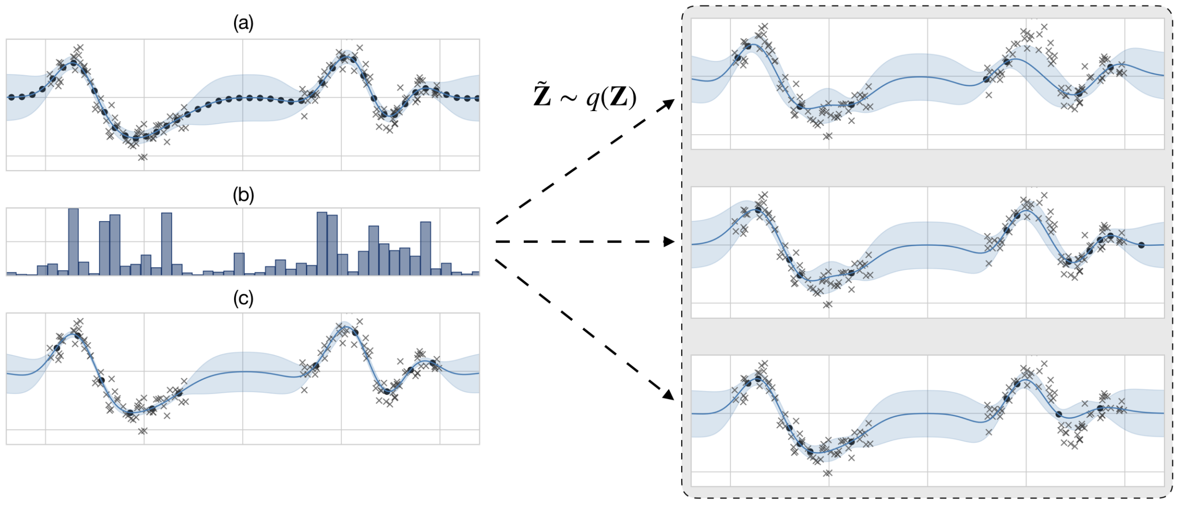

Figure 3.1: Illustration of the method. **(a)** The approximate posterior of a SVGP for a set of 1D datapoints (grey crosses), utilising 50 equidistant inducing points (black dots). **(b)** The marginal probability of inclusion for each inducing point midway through training, as assigned by our variational point process. Note that uninformative points are less likely to be included. The plots to the right show three samples from the point process and the conditional SVGP posteriors. **(c)** The final model after training where all but a small, informative subset of inducing points have been pruned away.

point process prior that encourages the model to be economical in its use of points. We then apply variational inference to derive an approximate posterior set of inducing points, thereby enabling the model to include only those points that it deems sufficiently informative in terms of explaining the observed data. The procedure is illustrated in Figure 3.1. As we will show experimentally, this approach reinstates the nonparametric ability of adapting M to the amount and characteristics of the data. Furthermore, we demonstrate how the approach can solve practical problems when specifying deep Gaussian processes and latent variable models.

The rest of the chapter is organised as follows: In Section 3.2 we analyse the role of inducing points in the sparse variational framework, define the notion of informativeness in the inducing set, and motivate the need for selecting one's inducing points with care. We then formulate our proposed solution in Section 3.3 and develop the necessary optimisation machinery for training the new model. Finally in Section 3.4 we provide empirical evidence of the efficacy of our approach for a variety of modelling tasks and datasets.

3.2 The role of inducing points

We pick up from the description of the variational free energy (VFE) framework from Section 2.7.1 where the optimisation objective is given by the evidence lower bound (ELBO):

$$\begin{aligned}\mathcal{L}(\mathbf{Z}) &\triangleq \mathbb{E}_{q(\mathbf{f}\mid\mathbf{X},\mathbf{Z})}\left[\log p(\mathbf{y}\mid\mathbf{f})\right] - \mathrm{KL}\left[q(\mathbf{u}\mid\mathbf{Z})\parallel p(\mathbf{u}\mid\mathbf{Z})\right], \\ q(\mathbf{f}\mid\mathbf{X},\mathbf{Z}) &= \int q(\mathbf{f},\mathbf{u}\mid\mathbf{X},\mathbf{Z})\,\mathrm{d}\mathbf{u} \\ q(\mathbf{f},\mathbf{u}\mid\mathbf{X},\mathbf{Z}) &= p(\mathbf{f}\mid\mathbf{u},\mathbf{X},\mathbf{Z})q(\mathbf{u}\mid\mathbf{Z}).\end{aligned}$$

Here we have made it explicit that the objective is a function of the inducing inputs, $\mathbf{Z}$, and we drop the dependency on all remaining model parameters to avoid notational clutter. The pivotal assumption of this construction is that we can adequately approximate the true function posterior at the observed inputs with a GP that is only conditioned on $\mathbf{u}$:

$$p(\mathbf{f}\mid\mathbf{X},\mathbf{y}) \approx \int p(\mathbf{f}\mid\mathbf{u},\mathbf{X},\mathbf{Z})q(\mathbf{u}\mid\mathbf{Z})\,\mathrm{d}\mathbf{u}.$$

This places a rather large amount of responsibility on the inducing points since all information about $\mathbf{y}$ must be communicated through $\mathbf{u}$. Ultimately, the inducing points act as an information bottleneck that upper bounds the potential capacity of our model. We can gauge this bottleneck through the posterior KL divergence:

$$\mathrm{KL}\left[q(\mathbf{f}\mid\mathbf{X},\mathbf{Z})\parallel p(\mathbf{f}\mid\mathbf{X},\mathbf{y})\right] = -\mathcal{L}(\mathbf{Z}) + \log p(\mathbf{y}\mid\mathbf{X}),$$

which is zero if and only if we manage to perfectly emulate the true function (i.e. the bottleneck is removed). However, this quantity is not available in most realistic scenarios since the marginal evidence will be intractable.

One important aspect for our analysis is the different factors that may influence this bottleneck. The most obvious factor, which is also the one that has attracted most attention in the literature, is the sheer number of points. However, just as crucial are various characteristics of the data being modelled and the latent function being approximated. As shown in Burt et al. [2019] we can communicate more information per inducing point when the data are clustered and the kernel is smooth. Another characteristic of interest, identified in Hensman and Lawrence [2014], is the aleatoric uncertainty. When the data are subject to more observation noise, there is less signal to infer for the exact GP, and we therefore require fewer inducing point to obtain a good approximation. The influence of these characteristics w.r.t. the posterior KL divergence is illustrated in Figure 3.2.

Common to these characteristics is that they are rarely known a priori. That is, when tasked

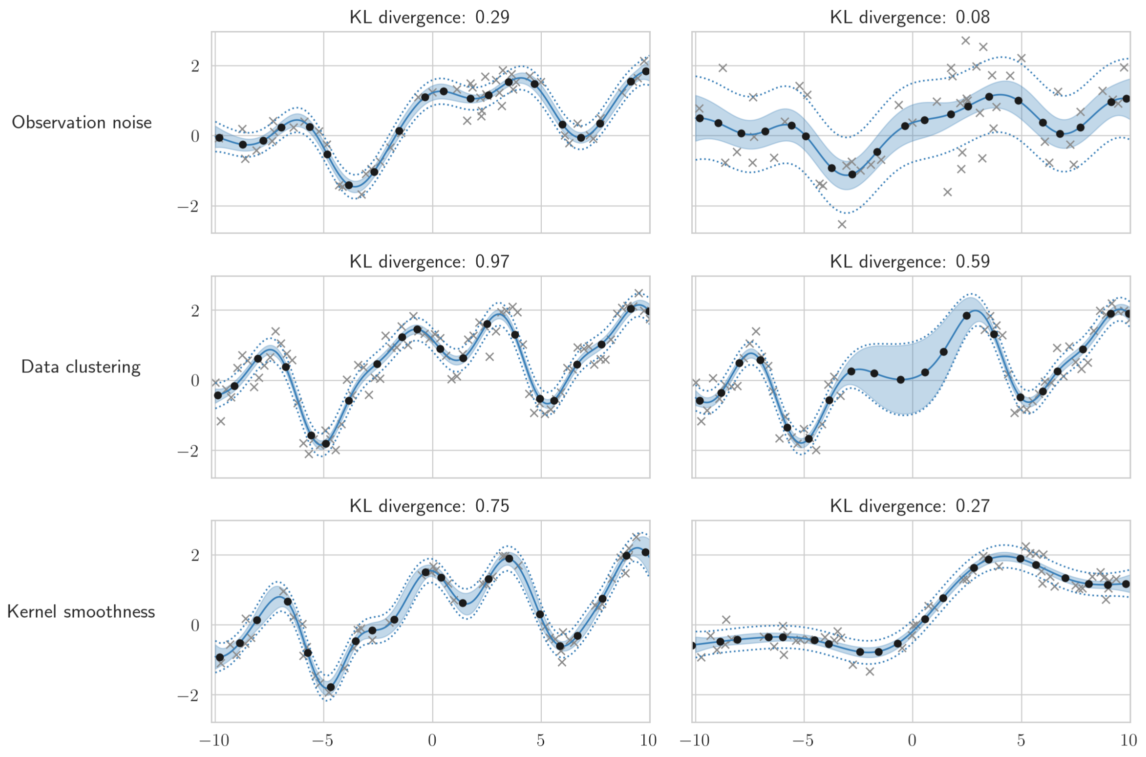

Figure 3.2: Illustration of the effect of three characteristics – observation noise, data clustering, and kernel smoothness – w.r.t. the posterior KL divergence. In each row the right example has a higher intensity of the given characteristic than the left. This results in a *lower* KL divergence when fitting a SVGP because the true function becomes easier to emulate.

with specifying a model for a new dataset, the model designer will generally not know how much observation noise is to be expected, how smooth the kernel should be, nor the amount of clustering within the data. Determining an adequate number of inducing points while being mindful of resource utilisation is therefore ultimately done on an uninformed basis. This lends weight to the argument that choosing the inducing points ought to be part of the inference rather than the model specification. We will return to this point in the experiments section, when we demonstrate that increasing the intensity of either of these characteristics causes our proposed model to adapt by pruning away inducing points and arriving at a more sparse approximation.

In some scenarios, the task of choosing the number of inducing points does not pose much of a dilemma. For instance, when fitting a standard SVGP we will usually opt for using as many inducing points as possible while keeping within the computational constraints at training and prediction time. In other scenarios, however, the choice between capacity and complexity is less obvious to the model designer. In an online setting where data are presented in batches and only available in memory for a limited time (see e.g. Bui et al. [2017b] or Titsias et al. [2019]) we will have to decide on the number of inducing points to dedicate

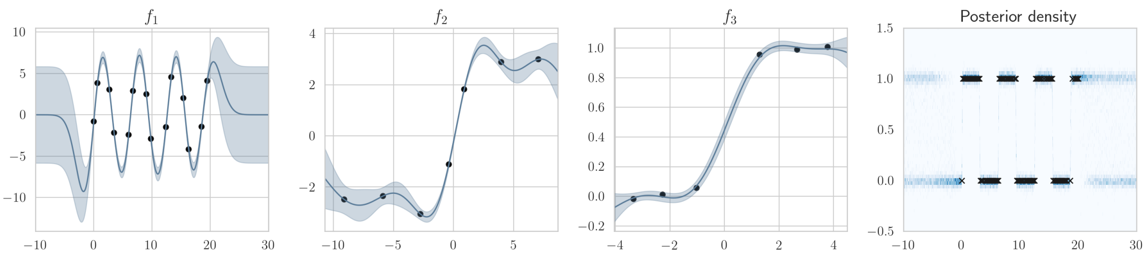

Figure 3.3: Illustration of different function fidelities in a 3 layer DGP. The three left plots show the learnt SVGP's for each layer, while the right plot shows the DGP's posterior density along with the target observations. We used our proposed method to jointly learn an adequate number of points across layers, resulting in the majority being assigned to the first and most complex one.

to a given batch in order to properly summarise it. This will in turn depend on various data characteristics for said batch, as described above, or the resemblance to already existing inducing points. Another relevant use case, that has recently seen a sharp increase in interest, is the design of models that consist of multiple GP compositions, e.g. DGP's (Damianou and Lawrence [2013]) and multi-fidelity modelling (Cutajar et al. [2019]). Here, the different functions may have varying degrees of fidelity and thus require different number of inducing points for achieving an acceptable approximation. This phenomenon is illustrated in Figure 3.3 where a square wave is modelled with a three layer DGP. The usual assumptions of smoothness and stationarity are quite poor in this case due to the discontinuous nature of the data, and a shallow GP with RBF or Matern kernel will therefore be ill-suited for this task. The deep model solves this by making a smooth approximation in the first layer while using the subsequent layers to attenuate the jumps through step functions. Note that because the first layer is more complex than the subsequent ones, it requires more inducing points to obtain an adequate function approximation. However, when working with high-dimensional data and more complex model architectures, the required per-layer fidelity may not be obvious, and finding a good allocation of inducing points will ultimately become a combinatorial problem. We will return to this scenario for real-world data in the experiments section.

Even when a suitable number of inducing points can be determined a priori, the model designer is still faced with the non-trivial task of choosing those points. This is relevant in methods where the inducing points are selected from the observed data, e.g. Titsias et al. [2019] and Cutajar et al. [2019]. Here, previous approaches have relied on measures such as the determinant of the inducing kernel matrix, $\mathbf{K}_{MM} = \kappa(\mathbf{Z}, \mathbf{Z})$, with the intuition being that a larger hypervolume indicates better coverage in the input space. However, it is important to note that the informativeness of the inducing points cannot be determined from the kernel matrix alone. A simple counter-example is shown in Figure 3.4 where a dataset with heteroscedastic noise is modelled. Since the observations to the left in each plot carry a stronger signal, the resources are better spent placing most of the inducing points here even

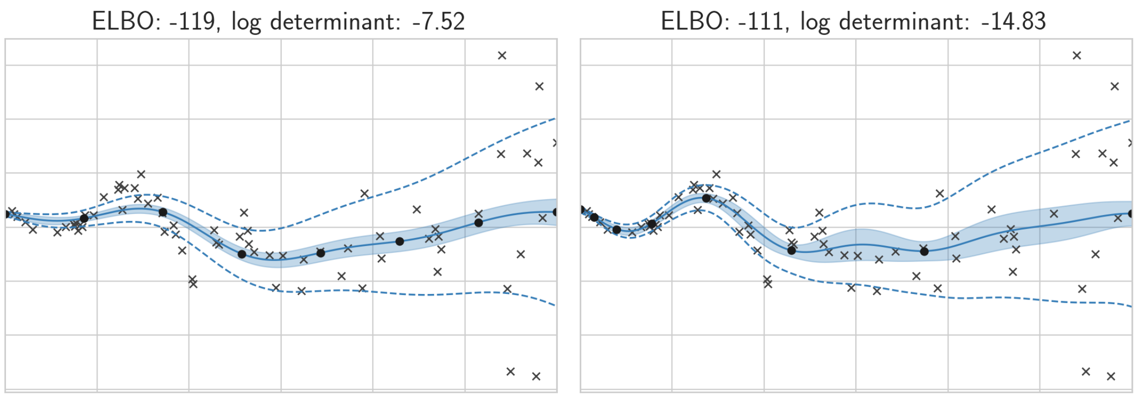

Figure 3.4: Demonstration of the notion that a larger hypervolume of the $\mathbf{K}_{MM}$ does not necessarily improve the ELBO. The SVGP to the left uses equidistant inducing inputs resulting in a maximal determinant of $\mathbf{K}_{MM}$, while the SVGP to the right clusters the inducing inputs where the signal is strongest.

if it results in a lower determinant of $\mathbf{K}_{MM}$.

Typically we will solve this by exploiting the fact that we have access to the gradient of any inducing input, $\frac{\partial \mathcal{L}}{\partial \mathbf{z}_j}$, which can thus be included in the model optimisation. However, here we encounter another subtle problem, namely that the gradient for a given point is proportional to the influence said point exerts over the approximate posterior. This has the unfortunate effect that as an inducing point moves away from the observed data, as measured under the kernel, our derivative information diminishes:

$$\lim_{\kappa(\mathbf{z}_j, \mathbf{X}) \to \mathbf{0}} \quad \frac{\partial}{\partial \mathbf{z}_j} \mathbb{E}_{q(\mathbf{f} | \mathbf{X}, \mathbf{Z})} \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] = \mathbf{0}.$$

We provide a formal argument for this in Appendix A. As a result the point may get “stuck” far away from the data without contributing to the modelling task but still allocating resources. This is illustrated for a 1D example in Figure 3.5 where an inducing point is located at a distance of 7 times the characteristic lengthscale from the nearest observation. Consequently, the gradient magnitude is less than 10^{-11} and the point will therefore not get close to the observed data in the immediate future for any sensible step size. Letting the selection of inducing points be part of the inference endows the model with the ability to simply remove points that have become redundant.

3.2.1 Relation to prior work

Since the sparse methodology was first introduced, much work has focused on reducing the information bottleneck between $\mathbf{y}$ and $\mathbf{u}$ by pursuing two, sometimes overlapping, strategies. In the first strategy, we seek to make the set of inducing points more informative by

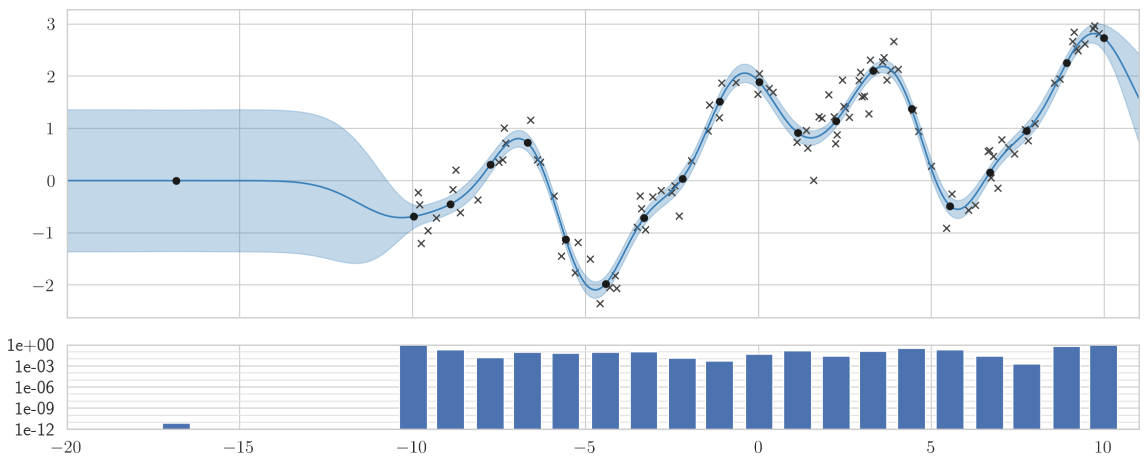

Figure 3.5: Illustration of the gradient diminishing as an inducing point is moved away from the observed data. The lower plot shows the magnitude of the partial derivative, $\frac{\partial \mathcal{L}}{\partial z_j}$, for each inducing input, z_j. The point to the left is effectively stuck, asserting no influence over the function approximation in the relevant part of the domain but still taking up computational resources.

optimising a compact representation that lies in a different domain than the observed data, either on the input side (Snelson and Ghahramani [2012], Wilson et al. [2016b]) or the output side (Lázaro-Gredilla and Figueiras-Vidal [2009], Hensman et al. [2018], van der Wilk et al. [2017]). In the second strategy, we aim at reducing the computational load of calculating the variational densities, thus allowing us to scale up the number of inducing points to many more than what would otherwise be feasible (Wilson and Nickisch [2015], Salimbeni et al. [2018], Dutordoir et al. [2020]).

The common goal for these methods is to make the most use of whatever computational resources are available. They are thus related but fundamentally orthogonal to the problem that we consider, which is to infer which and how many inducing points are required. It is important to note, however, that the approach that we develop in the following section can be applied in conjunction with any of these methods so long as they adhere to the general variational framework where $p(\mathbf{f} \mid \mathbf{X}, \mathbf{y})$ is approximated by $q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z}) = \int p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u} \mid \mathbf{Z}) \, \mathrm{d}\mathbf{u}$.

3.3 Probabilistic selection of inducing points

Our proposed solution to all issues listed in the previous section is to move the choice of inducing point set from being part of the model specification to instead being part of the model inference. We do this by simply expanding the hierarchical model with a point process prior over the inducing inputs, $p(\mathbf{Z})$, to explicitly encode our uncertainty as to which points should be used for summarising the data. To accommodate this expansion we introduce a variational point process, $q(\mathbf{Z})$, that approximates the posterior over inducing point sets. Note that $p(\mathbf{Z})$ and $q(\mathbf{Z})$ are *not* continuous distributions over the locations for fixed-sized sets of inputs; they are discrete distributions that assign probability mass to sets of points. Our joint distribution for the generative model is thus updated to:

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

while the variational distribution for the latent variables is given by:

$$q(\mathbf{f}, \mathbf{u}, \mathbf{Z} \mid \mathbf{X}) = p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u} \mid \mathbf{Z})q(\mathbf{Z}).$$

We note that within the SVGP literature, $\mathbf{Z}$ is often interpreted as a variational parameter rather than part of the generative model, the argument being that $\mathbf{Z}$ exclusively affects the proposal distribution (see e.g. Hensman et al. [2015b] and Hensman et al. [2018]). Following this line of reasoning it is not theoretically justified to associate $\mathbf{Z}$ with a prior. To counter this view, we argue that the inducing inputs are indeed present in the generative model as inputs to the infinite function vector of the GP – they are just redundant since $\mathbf{f}$, by assumption, is a sufficient statistic for $\mathbf{y}$. We are thus free to put a prior on $p(\mathbf{Z})$, causing $\mathbf{Z}$ to be marginalised out along with $\mathbf{u}$ without affecting $p(\mathbf{y} \mid \mathbf{f})$. However, once we make the variational approximation the inducing points cease to be redundant, and any prior associated with the inducing points will in turn cease to be redundant as well.

As will be specified in Section 3.3.1 we choose $p(\mathbf{Z})$ to encourage sparsity by assigning higher probability mass to sets of smaller cardinality. Through $q(\mathbf{Z})$ the model is now able to choose only those points that it deems sufficiently informative w.r.t. explaining $\mathbf{y}$. With $p(\mathbf{Z})$ and $q(\mathbf{Z})$ defined we obtain the new ELBO:

$$\begin{aligned}
\log p(\mathbf{y} \mid \mathbf{X}) &\geq \mathbb{E}_{q(\mathbf{f},\mathbf{u},\mathbf{Z}|\mathbf{X})}\left[\log \frac{p(\mathbf{y} \mid \mathbf{f})\cancel{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})}p(\mathbf{u} \mid \mathbf{Z})p(\mathbf{Z})}{\cancel{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})}q(\mathbf{u} \mid \mathbf{Z})q(\mathbf{Z})}\right] \\
&= \mathbb{E}_{q(\mathbf{Z})}\left[\mathbb{E}_{q(\mathbf{f},\mathbf{u}|\mathbf{X},\mathbf{Z})}\left[\log \frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{u} \mid \mathbf{Z})}{q(\mathbf{u} \mid \mathbf{Z})}\right] + \log \frac{p(\mathbf{Z})}{q(\mathbf{Z})}\right] \\
&= \mathbb{E}_{q(\mathbf{Z})}\left[\mathcal{L}(\mathbf{Z})\right] - \mathrm{KL}\left[q(\mathbf{Z}) \parallel p(\mathbf{Z})\right] \qquad (3.1) \\
&\triangleq \tilde{\mathcal{L}}.
\end{aligned}$$

Note that this objective encodes the capacity/complexity trade-off since the first term increases with the number of points drawn from $q(\mathbf{Z})$ while the latter decreases. Another point worth highlighting is that under this construction we never actually have to sample from our prior, $p(\mathbf{Z})$ – we only need to be able to evaluate the probability mass function. This affords us a lot of freedom in designing a prior that suits our purposes.

3.3.1 Specifying the point processes

Our overall objective is to enable the model to be selective w.r.t. which inducing points to include, based on their ability to explain the observed data. We therefore encode in our prior a preference towards smaller sets of inducing points:

$$p_\alpha(\mathbf{Z}) = C \cdot \exp\left(-\alpha|\mathbf{Z}|^2\right), \tag{3.2}$$

where α is a (fixed) hyperparameter determining the strength of the prior while C is the normalising constant. We use the squared cardinality of $\mathbf{Z}$ since the complexity of the standard SVGP model when making predictions is $\mathcal{O}(|\mathbf{Z}|^2)$ in time and memory – however, this is ultimately a design choice.

This prior only assigns probability according to the size of $\mathbf{Z}$ while being agnostic w.r.t. the actual locations in input space. Although we could also use the prior for other purposes, such as encouraging more dispersion of inducing points (as was recently done in Rossi et al. [2021]), we choose to focus solely on the objective of managing the size of $\mathbf{Z}$. Note that the location of points is still optimised during model training since they do affect the likelihood term, $\mathbb{E}_{q(\mathbf{Z})}\left[\mathcal{L}(\mathbf{Z})\right]$.

The hyperparameter α controls the sparsity level of the model, and it thus plays a regularisation role similar to the shrinkage penalty in Ridge or Lasso regression (Hoerl and Kennard [1970], Tibshirani [1996]). The reasoning behind including it as a configurable parameter is that the ideal sparsity level can never be determined from the data alone; it must necessarily be subject to resource considerations. I.e. different sparsity levels may be appropriate depending on whether a model is run on a GPU cluster or a laptop. This, of course, means that the model designer has not been completely relieved of all responsibility relating to the inducing points. However, any and all choices are now condensed into a single parameter residing on a higher level in the Bayesian hierarchy, and the model still maintains the ability to adapt the points according to function fidelity and data characteristics.

For the variational distribution, $q(\mathbf{Z})$, we choose the discrete Poisson Point Process (PPP) which simply associates each individual point from the input domain with an independent probability of inclusion (Streit [2010]). Given a set of candidate points, $\mathbf{Z}^\star$, the probability of observing $\mathbf{Z} \subseteq \mathbf{Z}^\star$ is:

$$q_\lambda(\mathbf{Z}) = \prod_{\mathbf{z}_k \in \mathbf{Z}} q(\mathbf{z}_k) \prod_{\mathbf{z}_k \notin \mathbf{Z}} (1 - q(\mathbf{z}_k)) = \prod_{\mathbf{z}_k \in \mathbf{Z}} \lambda_k \prod_{\mathbf{z}_k \notin \mathbf{Z}} (1 - \lambda_k),$$

where the vector $\lambda = [\lambda_1, \ldots, \lambda_K]^T$ is the variational parameter comprising the probabilities assigned to each of the K elements in $\mathbf{Z}^\star$. The domain of q_λ is thus the power set of $\mathbf{Z}^\star$ excluding the empty set, $\mathcal{P}(\mathbf{Z}^\star) \setminus \emptyset$, which yields $2^K - 1$ instances.

This choice of surrogate distribution implies two concessions: We restrict the domain to the finite set of points, $\mathbf{Z}^\star$, that is determined in advance, and we rely on a mean-field assumption rather than allowing for correlations between points. A simple remedy of restricting the domain is to simply overshoot by including more points in $\mathbf{Z}^\star$ than we believe to be necessary. This does not negatively impact optimisation complexity since only the sampled points enter the calculations of (3.1). The extra points and their associated probability will of course take up additional memory, but this will typically be negligible when compared to the actually observed data. The mean-field assumption, however, is more problematic. When inspecting Figure 3.1 it seems clear that neighbouring points should be assigned approximately the same marginal probability of inclusion while having high, negative correlation, such that when one is sampled, the other is left out. Under the PPP we instead force the model to choose a small set of favoured points and discard the rest. Learning the correlations could be achieved by instead using a determinantal point process (Kulesza and Taskar [2012]) for $q_\lambda(\mathbf{Z})$ which seems especially well-suited for this setting since it, like the GP, relies on a kernel to determine the degree of similarity between points. However, we will leave this avenue open for future research and focus on the PPP which has proven sufficiently robust for proving the efficacy of the proposed paradigm.

Under these choices of prior and variational distributions the KL divergence is analytically tractable. This can be seen by first decomposing into entropy and cross entropy:

$$\begin{aligned}\mathrm{KL}\left[q_\lambda(\mathbf{Z}) \parallel p_\alpha(\mathbf{Z})\right] &= -\mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\log p_\alpha(\mathbf{Z})\right] + \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\log q_\lambda(\mathbf{Z})\right] \\ &= CE(\alpha, \lambda) - H(\lambda).\end{aligned}$$

Since the elements of $\mathbf{Z}^\star$ are sampled independently, the joint entropy can be written as the sum of entropies pertaining to individual elements:

$$\begin{aligned}H(\lambda) &= -\mathbb{E}_{q(\mathbf{z}_1),\dots,q(\mathbf{z}_K)}\left[\sum_{k=1}^{K} \log q(\mathbf{z}_k)\right] \\ &= -\sum_{k=1}^{K}\left(\lambda_k \log \lambda + (1-\lambda_k)\log(1-\lambda_k)\right).\end{aligned}$$

For the cross-entropy, note that the cardinality of $\mathbf{Z}$ under $q_\lambda(\mathbf{Z})$ follows a Poisson binomial distribution. The two first moments of $|\mathbf{Z}|$ are thus:

$$\mathcal{E} \triangleq \mathbb{E}_{q_\lambda(\mathbf{Z})}[|\mathbf{Z}|] = \sum_{k=1}^{K} \lambda_k, \qquad \mathcal{V} \triangleq \mathrm{Var}_{q_\lambda(\mathbf{Z})}[|\mathbf{Z}|] = \sum_{k=1}^{K} \lambda_k \sum_{k=1}^{K}(1-\lambda_k).$$

We then have:

$$CE(\alpha, \lambda) = -\log C + \alpha \mathbb{E}_{q(\mathbf{Z})}\left[|\mathbf{Z}|^2\right] = -\log C + \alpha(\mathcal{V} + \mathcal{E}^2).$$

The normalising constant is given by

$$C = \frac{1}{K} \sum_{k=1}^{K} \binom{K}{k} e^{-\alpha \cdot k^2},$$

but can generally be ignored as it contains no free parameters.

Besides being closed-form computable, the KL divergence is differentiable and thus easy to include in gradient-based optimisation. Its tractability is contingent on three statistical measures for the variational point process: the mean and variance of number of points and the entropy over the entire domain. Any other point process for which the same measures are available can thus be readily applied without requiring changes to the expressions above.

3.3.2 Optimising the new objective

Having specified a new objective and the relevant prior and variational distributions, we now turn to the practical matter of model training. In order to leverage efficient optimisation methods based on Stochastic Gradient Descent (Zhang [2004], Nemirovski et al. [2009]) we need to obtain the gradients for (3.1):

$$\nabla_\theta \tilde{\mathcal{L}} = \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\nabla_\theta \mathcal{L}_\theta(\mathbf{Z})\right] \tag{3.3}$$

$$\nabla_\lambda \tilde{\mathcal{L}} = \nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}_\theta(\mathbf{Z})\right] - \nabla_\lambda \mathrm{KL}\left[q_\lambda(\mathbf{Z}) \parallel p_\alpha(\mathbf{Z})\right] \tag{3.4}$$

where we let θ denote all free parameters for $\mathcal{L}_\theta(\mathbf{Z})$ excluding $\mathbf{Z}$, i.e. the variational parameters pertaining to the inducing points and the hyperparameters for the likelihood and GP prior. The expectations w.r.t. $q_\lambda(\mathbf{Z})$ constitute a summation over $2^K - 1$ subsets, K being the number of candidate points in $\mathbf{Z}^\star$, so calculating the above gradients directly is not tractable in the general case. Instead we will provide approximations for both (3.3) and (3.4) and show how they can be unified in a single objective that makes the optimisation straightforward to implement in automatic differentiation frameworks such as PyTorch (Paszke et al. [2019]) and Tensorflow (Abadi et al. [2016]).

For (3.3), note that $\nabla_\theta \mathcal{L}(\mathbf{Z})$ is the usual gradient we rely on when optimising the standard SVGP for a fixed set of inducing points. We can thus simply approximate the gradient of $\tilde{\mathcal{L}}$

with Monte Carlo sampling:

$$\nabla_\theta \tilde{\mathcal{L}} \approx \frac{1}{S}\sum_{s=1}^{S} \nabla_\theta \mathcal{L}_\theta(\tilde{\mathbf{Z}}^{(s)}), \qquad \tilde{\mathbf{Z}}^{(s)} \sim q_\lambda(\mathbf{Z}). \tag{3.5}$$

As for (3.3), the gradient pertaining to the KL divergence is readily available qua our choice of point processes. What remains is then $\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}_\theta(\mathbf{Z})\right]$ which is *not* directly amenable to sampling-based approximation since

$$\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}_\theta(\mathbf{Z})\right] = \sum_{\mathbf{Z}\in\mathcal{P}(\mathbf{Z}^\star)\setminus\emptyset} \mathcal{L}_\theta(\mathbf{Z})\nabla_\lambda q_\lambda(\mathbf{Z}),$$

does not have the form of an expectation as we would require. As described in Section 2.3 we can solve this with the re-parameterisation trick, provided that $q_\lambda(\mathbf{Z})$ can be rewritten as a deterministic function of some random variable that follows a non-parameterised distribution. Although we did develop a re-parameterisation scheme that makes this approach possible, it is contingent on $q_\lambda(\mathbf{Z})$ being a PPP and would prohibit generalising to other, more versatile point processes in the future. We therefore defer the detailed description of the re-parameterisation method to Appendix B.

Instead we will rely on score function estimation (SFE) to obtain an unbiased approximation of the gradient:

$$\begin{aligned}\nabla_\lambda \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}_\theta(\mathbf{Z})\right] &= \mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}_\theta(\mathbf{Z})\nabla_\lambda \log q_\lambda(\mathbf{Z})\right] \\ &\approx \frac{1}{S}\sum_{s=1}^{S} \mathcal{L}_\theta(\mathbf{Z})\nabla_\lambda \log q_\lambda(\tilde{\mathbf{Z}}^{(s)}) \\ \tilde{\mathbf{Z}}^{(s)} &\sim q_\lambda(\mathbf{Z}).\end{aligned} \tag{3.6}$$

When implementing the method in a framework that supports automatic differentiation, we can now combine (3.5) and (3.6) in the surrogate objective:

$$R(\lambda,\theta) = \frac{1}{S}\sum_{s=1}^{S}\left(\overline{\mathcal{L}_\theta(\mathbf{Z})}\log q_\lambda(\tilde{\mathbf{Z}}^{(s)}) + \mathcal{L}_\theta(\tilde{\mathbf{Z}}^{(s)})\right), \qquad \tilde{\mathbf{Z}}^{(s)} \sim q_\lambda(\mathbf{Z}),$$

where $\overline{(\cdot)}$ indicates that the expression is decoupled from the differentiation process. This objective allows us to acquire the noisy gradients w.r.t. both λ and θ in a single pass for each sample from $q_\lambda(\mathbf{Z})$, making it straightforward to implement on top of any existing SVGP method that formulates $\mathcal{L}_\theta(\mathbf{Z})$ as the optimisation objective.

One issue to be aware of for the SFE optimisation is the risk of high sampling noise in (3.6). There exist many methods for alleviating this problem but we have chosen the relatively simple approach of subtracting a baseline, b, from $\overline{\mathcal{L}_\theta(\mathbf{Z})}$ before differentiating $R(\lambda,\theta)$

(Glasserman [2013]). When b is constant w.r.t λ this operation does not affect the gradient in the limit of $S \to \infty$, but it will reduce the variance between gradient samples when if b is carefully chosen. We use the common heuristic of setting b to a decaying average of previous evaluations of $\mathcal{L}_\theta(\mathbf{Z})$, i.e. $b = a(\mathcal{L}_\theta^v(\mathbf{Z})_{-1} + \mathcal{L}_\theta^{v^2}(\mathbf{Z})_{-2} + \dots)$ where $\mathcal{L}_{\dot{\theta}}(\mathbf{Z})_{-i}$ is the i'th previous evaluation of $\mathcal{L}_\theta(\mathbf{Z})$, $v < 1$ is the decay parameter, and a is a normalising constant. This proved sufficient for obtaining stable optimisation in our experiments – however, if the method is to be applied for increasingly complex models and/or point processes in the future, it may be necessary to employ more sophisticated methods for ensuring stability.

3.4 Experiments

In this section we demonstrate the efficacy of our method in the context of both standard GP regression and the more esoteric settings of deep modelling and latent variable modelling. We wish to show that i. the model is able adapt to characteristics of the data and function dynamics that affect the informativeness of the inducing points, as described in Section 3.2; and ii. that our method has practical applicability when designing and training DGP's and GPLVM's.

3.4.1 Adapting to data characteristics

As described in Section 3.2 the informativeness of inducing points depends on the observation noise, kernel smoothness, and input clustering. When the intensity of either of these characteristics increases, the inducing points become more informative, and fewer points are required to reach the same level of approximation of the true posterior function. However, as these characteristics are typically not known a priori and need to be inferred, it only seems natural to also make the selection of inducing points part of model fitting.

In this experiment we show on synthetic and real-world data that our model does indeed adjust the number of inducing points as a function of the aforementioned characteristics. In the synthetic setting we first generate observations drawn from a GP with varying intensity of each characteristic. The specifics of the data generation process are described in Appendix C. For each set of observations we fit a standard SVGP with $[5, 10, \dots, 80]$ inducing points and measure the KL divergence between the true and approximate posterior, $\mathrm{KL}\left[q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z}) \parallel p(\mathbf{f} \mid \mathbf{X}, \mathbf{y})\right]$ (lower is better). These are the solid lines in Figure 3.6, each coloured by their intensity. As expected, we see that as the intensity increases, fewer points are required to emulate the true function.

Next we apply our method by associating a prior and variational point process for each condition and prune away points. The vertical, dashed lines in Figure 3.6 show the mean of

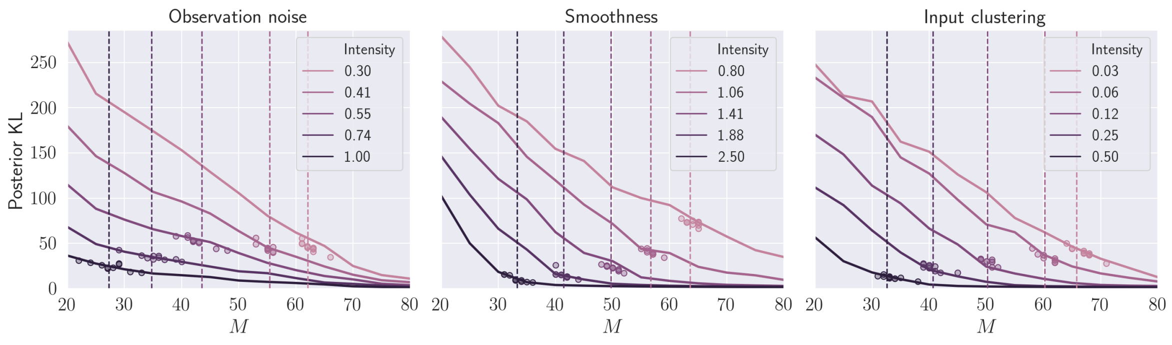

Figure 3.6: KL divergence between the approximate and exact posterior as a function of the number of inducing points under different data characteristics (lower is better). Solid lines are the baseline SVGP's with a fixed number of inducing points. As a given intensity increases, fewer points are required to obtain the same KL divergence. The vertical, dashed lines show the expected number of inducing points inferred by the point process under different characteristics, and the circles are 10 samples of subsets drawn from each. Our method adaptively reduces the number of points as informativeness decreases, and in certain cases it slightly improves upon the baseline.

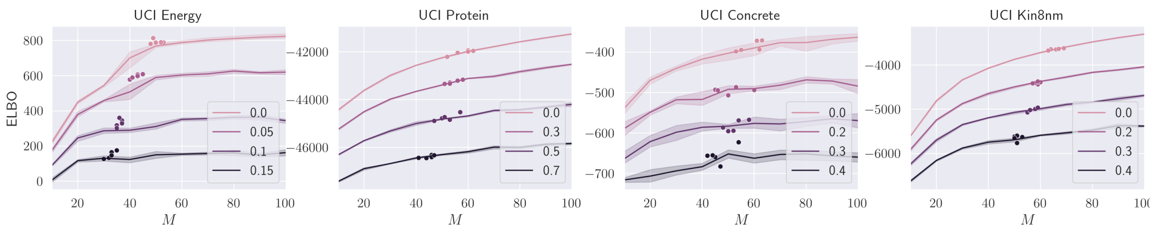

Figure 3.7: ELBO as a function of number of inducing points when adding Gaussian noise of varying intensity (higher is better). The setup is similar to that of Figure 3.6.

posterior number of inducing points which monotonically decreases with the intensities. The circles show the performance for 10 draws of inducing point sets from the variational point process. We see that we generally achieve similar or improved performance compared to the standard SVGP.

We repeat the experiment for 4 real-world datasets, constrained to the characteristic of observation noise. In this setting we measure the ELBO as a function of inducing points (higher is better) due to the intractability of calculating $\mathrm{KL}\left[q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z}) \parallel p(\mathbf{f} \mid \mathbf{X}, \mathbf{y})\right]$ for large datasets. The results are shown in Figure 3.7 and confirm that the point process adapts to the data as expected while maintaining or improving performance.

The two experiments demonstrate that the model is able to adapt the number of inducing points to the characteristics of the data. In settings where these characteristics are a priori unknown and resource utilisation is a priority, the method will relieve the model designer of having to pin down an adequate number of points over multiple iterations of re-fitting and evaluating for different model instantiations.

3.4.2 Deep Gaussian processes

In the sparse variational DGP formulation (e.g. Damianou and Lawrence [2013] and Salimbeni and Deisenroth [2017]) each layer of the model is associated with its own set of inducing points. Choosing the sizes of these sets are subject to the same considerations as for the "shallow" SVGP model, namely that more points will increase capacity but be more computationally demanding. However, as argued in Section 3.2, choosing the number of points for each layer is effectively a combinatorial problem that may constitute a time-consuming search where the model designer will have to re-fit the model for each configuration of points and compare the accuracy and efficiency. Our method provides a way of integrating this search into the model inference, allowing us to find an appropriate configuration in a single training run.

We consider a 2-layer DGP applied to the UCI Kin8nm regression dataset (Asuncion and Newman [2007]) with 8192 observations and 8 input dimensions. The DGP uses the doubly stochastic formulation of Salimbeni and Deisenroth [2017] that we reviewed in Section 2.7.3 where predictions are based on samples from the variational distributions, and optimisation relies on unbiased, noisy gradients obtained from those samples. Following the insights of Duvenaud et al. [2014] we add the observed input to each layer to avoid instability due to non-injective mappings between layers. We distinguish between the "fixed" model, where the number of inducing points per layer is decided before model fitting, and our "adaptive" model where the point sets of each layer are associated with their own prior and variational point process. The task we wish to solve is to find configurations of points between layers that result in high model accuracy while keeping prediction time low.

As a baseline, we train fixed models with $[10, 20, \dots, 100]$ inducing points in each layer for 5000 epochs, resulting in 100 distinct configurations. For each we measure the average test log likelihood and prediction time over 5 folds. The results are plotted as grey dots in Figure 3.8a, showing a large variation in prediction efficiency for models with similar performance. The large dots are those configurations that use the same number of points in each layer, which seems to be the default choice in the DGP literature (Damianou and Lawrence [2013], Salimbeni and Deisenroth [2017], Havasi et al. [2018b], Cutajar et al. [2019], Blomqvist et al. [2019]). For the adaptive model, we initialise each layer with 150 inducing points, pre-train for 2000 epochs, and learn the inducing point sets along with the remaining model parameters for another 3000 epochs. We repeat this for 4 settings of the prior – these are the coloured dots in Figure 3.8a.

From the results it is clear that all configurations found in the adaptive models are near the frontier of optimal trade-offs, and that increasing the prior weight steers the model towards more sparse configurations. This points to significant potential benefits of applying our method as DGP grows deeper and more complex. Additionally, the procedure may yield

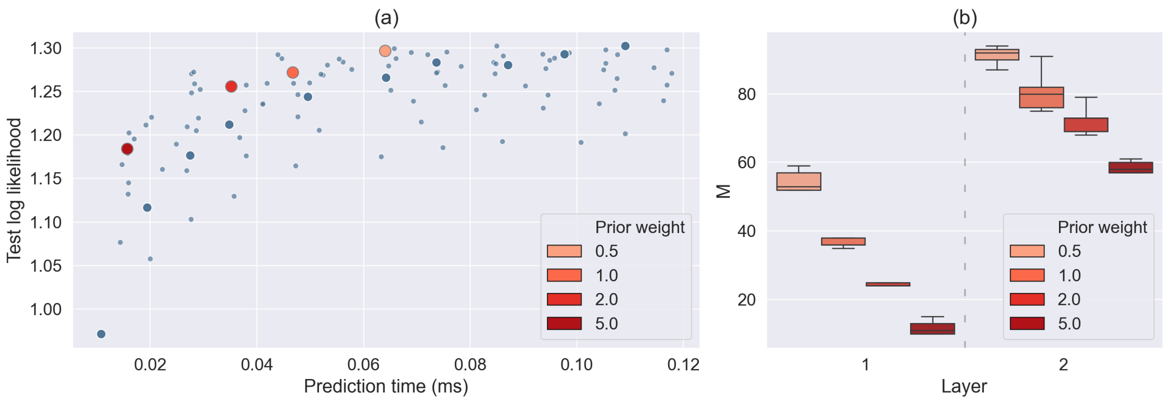

Figure 3.8: Dynamically allocating inducing points in a 2 layer DGP. **(a)** Each dot corresponds to the accuracy and performance of a specific configuration of inducing points. Grey dots are the outcomes of a comprehensive and tedious grid-search, with the 10 large dots representing those configurations that have equal number of inducing points in each layer. Red dots are the configurations found by our point process for four different settings of the prior weight, α. Note that the latter all fall on the frontier of optimal trade-offs between accuracy and performance. **(b)** The distribution of points across layers for over 5 folds. We see a clear preference towards placing more inducing points in the last layer.

post-hoc insights into the model in terms of layer fidelity and resource allocation. Figure 3.8b shows the distribution of points between layers for each of the prior settings, which indicates that for this specific architecture the second layer seems to be of higher fidelity than the first.

3.4.3 Latent variable modelling

Lastly we consider the Gaussian process Latent Variable Model (GP-LVM) which is identical to the standard GP regression model except for the inputs being unknown. By placing a prior on the inputs we can include these in the inference and derive a posterior over input locations which may be of interest for e.g. visualisation or features extraction for downstream tasks. However, the uncertainty of the inputs prohibits tractability of the marginal likelihood and posterior, and so again the variational sparse methodology can be applied to approximate the true posterior.

Choosing a suitable number of inducing points in this setting can prove especially difficult since the input space, that the points must represent, is an abstraction and will change over time as the variational input locations are inferred. The heuristic of choosing a set of inducing points that "covers" the input domain (Titsias et al. [2019]) thus proves difficult to apply when said domain is a moving target. Here our method provides a paradigm for including new points only when such is required by the current state of the latent representation.

To illustrate this we use a GP-LVM to learn a 2-dimensional representation of 48 single cell gene expressions for 437 samples.[2] The samples are taken at different stages of development

[2]qPCR dataset from `https://github.com/sods/ods`

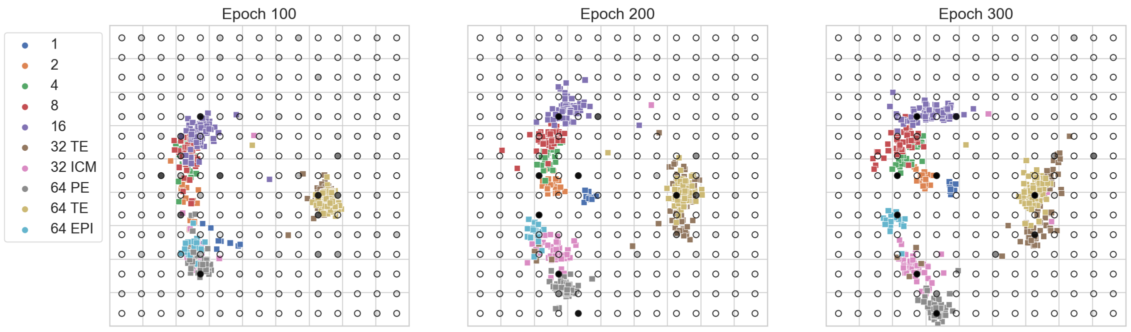

Figure 3.9: Demonstration of our method for a GPLVM applied to a single-cell qPCR dataset. Each coloured dot is the inferred location for a given cell in the latent space where the colour denotes the cell stage. We fix the inducing points to a 15×15 grid and let the filling indicate the marginal probability of inclusion. Here, filled means a probability of one while open means a probability of zero. As the latent inputs change throughout model training, different inducing points are considered relevant by the variational point process.

and the aim is to infer these stages from the gene expressions alone. We include 225 inducing points fixed to a 15×15 grid for illustrative purposes and jointly learn all model parameters (excluding inducing input locations) over 300 epochs. The training process is illustrated in Figure 3.9 where the latent coordinates are coloured according to cell stage. Notice that at any given time only a small subset of inducing points are assigned notable probability of inclusion, and that this subset changes over time as the inputs move around. In effect our method allows for the model to adapt to the change in function and inputs throughout model training, thus avoiding using more inducing points than what is necessary to accommodate the data.

3.5 Conclusion and future work

In this chapter we focused on the central assumption of the SVGP framework, namely that a deterministically chosen set of inducing points can adequately emulate the true posterior process. We argued that this assumption, in the context of Bayesian nonparametrics, is pathological in two respects: Firstly, because it compels the model designer to make a deterministic choice for a quantity, about which they may be quite uncertain; and secondly, because the number of inducing points bounds the potential capacity of the model and prevents the complexity to increase as a function of the amount and characteristics of the data. We also described various applications where the resource requirements are difficult or impossible to ascertain a priori, in which case the proper trade-off between capacity and complexity, as given by the number of inducing points, should be data dependent.

We showed that a very simple adjustment of the aforementioned assumption can help resolve these issues. By letting the inducing points be generated stochastically from a point process that assigns higher probability to smaller sets, we can use the Bayesian machinery to infer a posterior point process, thus learning which and how many inducing points to rely on for approximating the posterior function. In effect, the model can adapt its complexity requirements to the demands of the data.

For conducting inference under this model, we developed a VI scheme relying on an approximate point process that assigns independent probability of inclusion to all points from a finite candidate set. By the choice of the prior, this results in an analytically tractable KL divergence between point processes. The log likelihood expectation, however, requires approximation which we achieved through score function estimation. We demonstrated empirically that our proposed model adapts to various data characteristics that are known to affect complexity requirements; can be used for allocating inducing points between layers in a DGP; and adapts to the inferred representation for a GPLVM as the latent variables moves around in representation space during model training.

Future work may consider the integration of different point processes, both as prior and approximate posterior. The current prior, which penalises the quadratic cardinality of the inducing point set, was motivated by the $\mathcal{O}(M^2)$ time complexity when making predictions with a standard SVGP. However, other recent formulations (Cheng and Boots [2017], Havasi et al. [2018a], Salimbeni et al. [2018]) decouple the inducing points such that different sets are used for inferring respectively the predictive mean and variance. Since the mean calculation only imposes linear complexity in the number of inducing points, it would be sensible to update the point process prior to reflect this. Such an update would require minimal effort to implement since a linear variant of (3.2) would also result in a tractable KL divergence in the current framework. As mentioned in Section 3.3.1, another interesting avenue would be a more flexible variational posterior such as the determinantal point process (Kulesza and Taskar [2012]), which could rely on the same kernel as the GP prior for evaluating similarity between inducing points. Lastly, there may be great potential in integrating this method in the context of online learning where new data are continuously collected but the maximum number of inducing points is bounded by computational requirements, e.g. learning from streaming data (Bui et al. [2017a]) or reinforcement learning (Deisenroth and Rasmussen [2011]).

Chapter 4

Estimating a target vector through Bayesian optimisation

This chapter explores the problem of target vector estimation which is commonly addressed using Bayesian optimisation. However, we show that the default paradigm relies on erroneous assumptions that causes the method to waste valuable information while introducing pathological modelling deficiencies. By modifying the surrogate model to infer a more suitable predictive distribution, and by developing accompanying acquisition functions, we demonstrate a much improved search procedure on a variety of problems.

4.1 Introduction

The focus of this chapter is the task of matching the evaluations of a noisy, multi-output blackbox function to a pre-specified target vector. In other words, we wish to answer the question: “What inputs to our function may result in a given output vector?”. The premise remains largely identical to that of standard Bayesian optimisation: We assume that the blackbox function is expensive to evaluate, affording us a limited querying budget (typically 10’s to 100’s of evaluations) before deciding upon the most promising input. We also assume that we can only interact with the function through querying, allowing us no access to e.g. derivative information for guiding the search.

This problem pre-dates Bayesian optimisation and has been widely explored within the methodology of e.g. evolutionary algorithms (Schaffer [1985], Fourman [1985], Fonseca and Fleming [1998]). Example use cases span a diverse set of research areas, including modelling of blood flow in the human vasculature (Perdikaris and Karniadakis [2016]); estimating of the optimal hyperparameters of machine learning models (Snoek et al. [2012]); defining the structural design of helicopter rotor blades (Murugan et al. [2007]); estimating emission line intensities of alumina powder (Wienke et al. [1992]); and guiding the simulation parameterisation in Approximate Bayesian Computation (Järvenpää et al. [2018]). In the experimental section of this chapter we will furthermore consider two real-world scenarios: reverse engineering musical synthesizers (Horner et al. [1993], Garcia [2001], Lai et al. [2006], Heise et al. [2009]) and designing nano-topographies for stem cell development (Cutiongco et al. [2020]).

Multi-output functions in and of themselves are not a novelty in the context of Bayesian optimisation. The task that has received most attention is that of identifying the Pareto frontier, i.e. the set of non-inferior evaluations for which at least one output dimension is minimal/maximal (Knowles [2006], Zhang et al. [2010], Shah and Ghahramani [2016]). This is relevant in situations where we want to optimise over competing, correlated aspects of a given task, e.g. designing drugs that are simultaneously effective, safe, and cheap to develop (Negoescu et al. [2011]). Here the common approach is to model all outputs of the blackbox-function and apply an acquisition function that reasons about some relevant measure, such as the hypervolume of the Pareto frontier (Zitzler [1999]).

Less attention has been paid to our problem formulation where the quantity to be minimised is the distance between the function evaluations and a target vector. In contrast to the Pareto setting, we are here willing to accept the deterioration of one output dimension if it provides sufficient improvements in other dimensions such that the aggregated distance is reduced. Interestingly, this task is already perfectly amenable to standard Bayesian optimisation. Since the ultimate goal is to minimise a scalar (i.e. the distance to the target), we can simply place a Gaussian process prior on the function that maps input parameters to observed distances and use the predictive distribution in conjunction with a suitable acquisition function to search for promising querying locations that may reduce the objective value. This is indeed the approach taken in previous work (e.g. Perdikaris and Karniadakis [2016] and Järvenpää et al. [2018]) but in doing so we discard potentially valuable information since we are not observing the individual outputs of our blackbox function. Alternatively, we can follow the example of the Pareto-algorithms and model all output dimensions; however, the subsequent aggregation into a distance measure results in a quantity that is certainly not Gaussian, which breaks compatibility with the usual scalar-based acquisition functions.

In this chapter we solve these obstacles by updating the surrogate model to incorporate the individual function outputs, deriving an approximate distribution over the resulting distance, and developing accommodating acquisition functions that are both analytically tractable and differentiable. All combined this yields a principled and efficient framework for estimating a target vector under the Bayesian optimisation framework. The rest of the chapter is organised as follows: In Section 4.2 we formally specify the problem and our assumptions, consider the issues of using a naive approach, and review alternative approaches. In Section 4.3 we develop the surrogate model and acquisition functions for our method. Finally in 4.4 we

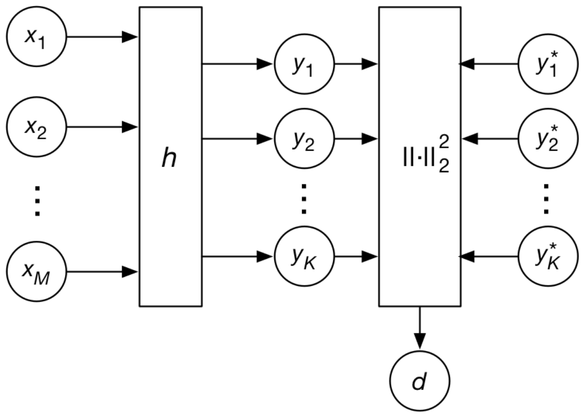

Figure 4.1: Pictorial depiction of the problem setup. We are given a stochastic blackbox function, h, that takes M-dimensional inputs and produces K-dimensional outputs. In addition, we know of a K-dimensional target vector, $\mathbf{y}^\star$. The task now is to identify the input that minimises the squared ℓ_2-norm between the function evaluation and the target vector.

show the superiority of our method on a wide range of benchmark functions and two real-world problems.

4.2 Problem specification

Given a blackbox, multi-out function $h : \mathcal{X} \to \mathcal{Y}$ we wish to identify the input that minimises the output distance to a known target vector, $\mathbf{y}^\star \in \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^M$ and $\mathcal{Y} \subseteq \mathbb{R}^K$. The appropriate distance metric is problem dependent but we will restrict attention to the widely used sum of squared residuals (or squared Euclidean distance or squared ℓ_2-norm):

$$d(\mathbf{y}) = \|\mathbf{y} - \mathbf{y}^\star\|_2^2 = \sum_{j=1}^{K} (y_j - y_j^\star)^2.$$

The optimisation task is depicted in Figure 4.1. We will use the notation from stochastic process theory and view the functions d and h as an infinite vector and matrix, respectively, each of which are indexed by a finite set of inputs. The distribution over the distance $d(\mathbf{x}) = \|h(\mathbf{x}) - \mathbf{y}^\star\|_2^2$ is simply written as $p(d \mid \mathbf{x})$.

To apply Bayesian optimisation for this minimisation task we place a prior on d and derive a posterior predictive distribution, $p(d \mid \mathbf{x}, \mathcal{D})$, for novel input $\mathbf{x}$ when conditioned on previously collected observations, $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$. The next input to query is then selected by maximising an acquisition function, $\alpha : \mathcal{X} \to \mathbb{R}$, that associates the predictive distribution with a measure of querying utility which should capture e.g. exploration/exploitation trade-off, level of greediness subject to the remaining budget, and other requirements of the model designer.

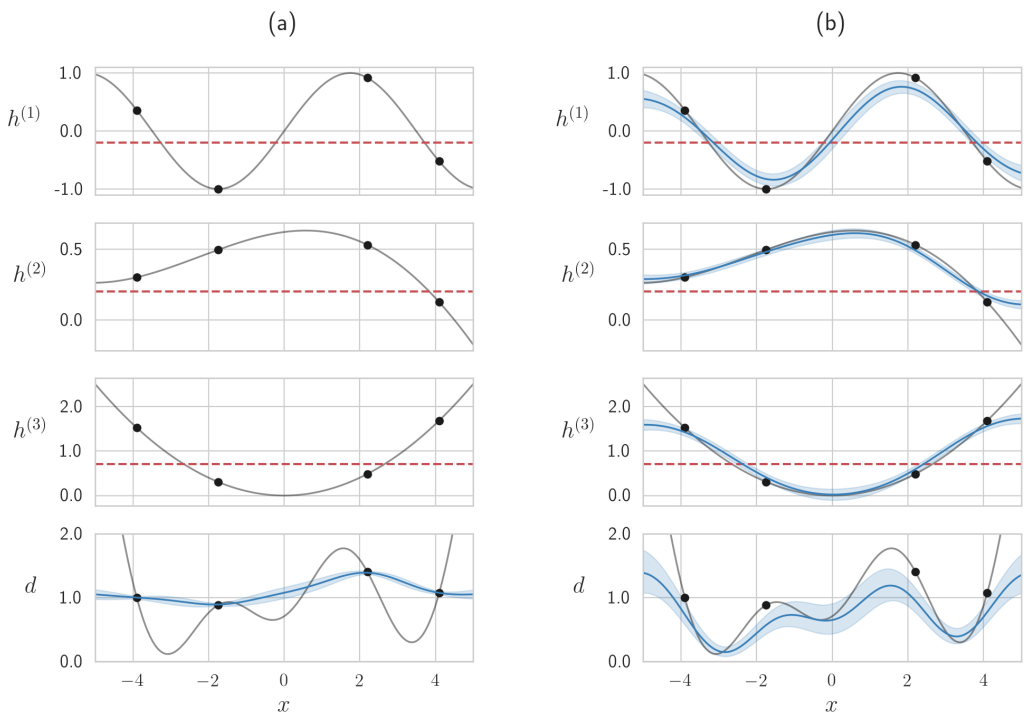

Figure 4.2: Inferring the distance between a target vector and a function with one input and three outputs. The three top rows show each dimension of the blackbox function while the horizontal red lines are the target values. The bottom row show the squared ℓ_2-norm, $d = \|h - \mathbf{y}^\star\|_2^2$. The left column illustrates a naive approach where we base our posterior on the distance measurements alone. Here, we will inevitably discard information about the individual outputs which may prove useful for selecting future querying locations. In the right column we model each output individually and use the posteriors to construct a more accurate distribution over d.

4.2.1 Applying standard Bayesian optimisation

The problem as currently specified is readily amenable to standard Bayesian optimisation. By placing a GP prior directly on d we get an analytically tractable, normal posterior for any unseen point, $p(d \mid \mathbf{x}, \mathcal{D})$. This distribution can now be used in conjunction with usual acquisition functions such as Expected Improvement (EI) or Lower Confidence Bound (LCB) to efficiently search for new inputs that minimise d.

Such an approach, however, has two major caveats. Firstly, we may discard potentially useful information about the individual outputs of h. This is because the surrogate model is only ever presented with the *aggregated* values, $d = \|h - \mathbf{y}^\star\|_2^2$. And since this aggregation is non-injective, information about h is inevitably lost. This is illustrated in Figure 4.2a where a number of different output vectors from h result in roughly the same distance to $\mathbf{y}^\star$. A surrogate model that only updates its posterior belief based on observations of d would therefore learn very little about h from such a set of training data.

The second caveat is that the normal distribution is quite ill-suited for modelling a distance.

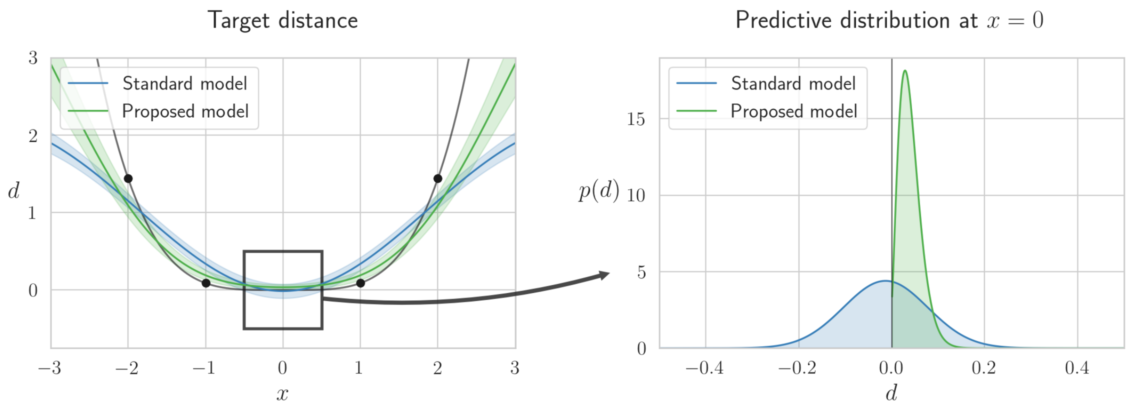

Figure 4.3: Modelling the distance $d = \|h - y^\star\|_2^2$ with $h(x) = 0.3x^2$ and $y^\star = 0$. The optimum is located at $x = 0$. Black dots are the observed samples. Even though both models have a fairly good fit and a reasonable idea of the true minimum, the standard model enforces a normal predictive distribution which results in an unrealistic estimate with both negative support and symmetrical density around the mean.

Any distribution over distances will generally be asymmetrical and without negative support, neither of which is true for the normal distribution. This is especially critical when $p(d \mid \mathbf{x}, \mathcal{D})$ is highly uncertain while being centered close to zero, since a large proportion of potential values are then assumed to be realised in the negative domain as shown in Figure 4.3. At the same time, such locations will be favoured by most acquisition functions as they satisfy both the exploration and exploitation criteria, potentially leading the procedure to spend most of the querying budget at regions where the surrogate model is least accurate.

4.2.2 Sampling based approaches

From the above it is clear that the main challenge of the optimisation problem arises from wanting to simultaneously i. incorporate the individual outputs of the blackbox function in the surrogate model, ii. obtain a reliable posterior of the distance, and iii. identifying an acquisition value for this posterior. The currently prevalent approach for accomplishing this is to rely on stochastic approximations. Astudillo and Frazier [2019] considers the more general problem of composite functions on the form, $v = t(r(\mathbf{x}))$, where $r : \mathcal{X} \to \mathbb{R}^K$ is an expensive blackbox function with multiple outputs while $t : \mathbb{R}^K \to \mathbb{R}$ is cheap, known, and differentiable. The authors propose modelling r with a MTGP and then consider the EI over current incumbent value, t_*:

$$\alpha_{\text{EI}}(\mathbf{x}; \mathcal{D}) = \mathbb{E}_{p(r|\mathbf{x},\mathcal{D})}\left[\max(t_* - t(r), 0)\right].$$

Since the $\max(\cdot)$ function is subdifferentiable (similar to the rectified linear operator commonly employed in neural networks), the gradients w.r.t. $\mathbf{x}$ can now be approximated with the re-parameterisation trick (Kingma et al. [2015]) given that t is known:

$$\begin{aligned}\nabla_{\mathbf{x}}\alpha_{\text{EI}}(\mathbf{x};\mathcal{D}) &= \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\nabla_{\mathbf{x}}\max(t_* - t(\boldsymbol{\mu}_r + \mathbf{L}_r\boldsymbol{\epsilon}), 0)\right] \\ &\approx \frac{1}{S}\sum_{s=1}^{S}\nabla_{\mathbf{x}}\max(t_* - t(\boldsymbol{\mu}_r + \mathbf{L}_r\widetilde{\boldsymbol{\epsilon}^{(s)}}), 0), \\ \tilde{\boldsymbol{\epsilon}}^{(s)} &\sim \mathcal{N}(\mathbf{0},\mathbf{I}),\end{aligned} \tag{4.1}$$

where $\boldsymbol{\mu}_r$ and $\Sigma_r = \mathbf{L}_r\mathbf{L}_r^T$ are the predictive mean and covariance of the surrogate model. We can obviously map this formulation to our problem by letting the outer composition be the distance to our target vector. This approach can be seen as a special case of the framework of Wilson et al. [2017] where the authors propose to use the re-parameterisation trick for any acquisition function that can be expressed as an integral. Prior work has also considered intractable acquisition functions, e.g. Hennig and Schuler [2012] and Henrández-Lobato et al. [2014] that use expectation maximisation to estimate the differential entropy at novel inputs, and Snoek et al. [2012] and Wang et al. [2020] that consider the EI in parallelised settings and use Monte Carlo sampling in a similar manner to (4.1). These works, however, presupposes one-dimensional predictive densities.

This family of methods are prone to two sources of added complexity: Firstly, the repeated evaluation of the acquisition function for different samples in each iteration when obtaining asymptotically unbiased gradient estimates; and secondly, the fact that we are limited to employing first-order stochastic optimisation methods rather than more powerful quasi-Newton based methods, such as BFGS (Nocedal and Wright [2006]).

These issues can be alleviated somewhat by instead using sample average approximation (SSA) as proposed in the context of Bayesian optimisation in Balandat et al. [2020]. Rather than drawing new base samples, $\{\tilde{\boldsymbol{\epsilon}}^{(s)} \sim \mathcal{N}(\mathbf{0},\mathbf{I})\}_{s=1}^{S}$, at every iteration, the SSA approach relies on a single set of samples that is maintained throughout optimisation of the acquisition function. This has the dual benefits of requiring fewer base samples in total and turning the maximisation of the acquisition into a deterministic optimisation problem. The caveat is that reusing the same set of base samples results in the gradient estimation no longer being unbiased, although theoretical convergence properties are provided in Balandat et al. [2020].

In this chapter we want to draw a distinction between our approach, that uses numerical calculations and approximations to derive special purpose, tractable acquisition functions, and the more general, sample based approaches reviewed above. We acknowledge that with sufficient time, memory, and parallelisation, these approaches should be able to perform on a par or better than the method we develop. However, they are not very compatible with the theme of this thesis, which is the development of inference methods that are mindful of resource utilisation. Another argument against the sample-based approaches is that they are only applicable when the acquisition function can be written as an integral. This is not the case for e.g. LCB[1] which would fall by the wayside when restricting attention to Monte Carlo methods. For these reasons, we will use the standard Bayesian optimisation approach discussed in the previous section as the baseline for comparisons against our method.

4.3 Efficient Bayesian optimisation for target vector estimation

Our approach is conceptually straightforward: As in Pareto and Monte Carlo methods we will use a surrogate model that learns the individual outputs of h. However, rather than a sample based approximation of the resulting distance, we develop an analytical one based on the noncentral χ^2. Given this new predictive distribution we then turn our attention to the acquisition functions and derive closed-form expressions for EI and LCB. We will refer to our proposed surrogate model as the ℓ_2 model.

4.3.1 Updating the distance distribution

Keeping with Gaussian processes as the main modelling tool, we assume that the observations, $\{\mathbf{y}_i = h(\mathbf{x}_i)\}_{i=1}^{n}$, are evaluations of a latent MTGP corrupted by independent Gaussian noise:

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}_\epsilon)\right).$$

Since the target vector, $\mathbf{y}^\star$, is constant, this effectively places a multivariate normal distribution on the difference vector, $\Delta = h - \mathbf{y}^\star$, i.e. $p(\Delta) = \mathcal{N}(\Delta \mid \mu, \Sigma)$ for some statistics given by the GP (we will drop the conditioning on $\mathbf{x}$ and $\mathcal{D}$ in the rest of this section to avoid notational clutter). Under this assumption we have a well-established distribution for the Euclidean distance, $d = \|\Delta\|_2^2$, namely the generalised χ^2 ($G\chi^2$). However, as will be shown in the following section we require closed-form expressions for, or approximations to, the CDF and truncated mean in order to derive the EI and LCB acquisition functions, neither of which are known for the $GC\chi^2$ distribution at the time of writing. In order to circumvent this problem we replace the covariance matrix, Σ, with a scaled identity matrix, $\mathbf{I}\gamma^2$, which turns $p(d \mid \mathbf{x}, \mathcal{D})$ into a scaled noncentral χ^2 ($NC\chi^2$) distribution.

The $NC\chi^2$ is defined as the distribution over a sum of independent, squared variables that each follows a normal distribution with unit variance. I.e. if $\mathbf{z} = [z_1, z_2, \ldots, z_k]^T$ is a vector of uncorrelated random variables s.t. $z_j \sim \mathcal{N}(\mu_j, 1)$, then $\|\mathbf{z}\|_2^2 \sim NC\chi^2(k, \lambda)$ where k is

[1]We observe that Wilson et al. [2017] does propose an integral formulation of LCB but this only applies when the predictive density is Gaussian. We do not know of a similar expression for distributions over distances.

the degrees of freedom (i.e. the size of $\mathbf{z}$) and $\lambda = \sum_{i=1}^{k} \mu_i^2$ is the noncentrality parameter. To map this to our problem, consider the distance scaled by γ^{-2}:

$$d\gamma^{-2} = \left(\sum_{j=1}^{k} \Delta_j^2\right) \gamma^{-2} = \sum_{j=1}^{k} (\Delta_j \gamma^{-1})^2.$$

Here Δ_j is the difference between the j'th entries of the output and target vector. Since $\Delta_j \gamma^{-1} \sim \mathcal{N}(\mu_j \gamma^{-1}, 1)$ it follows that $d\gamma^{-2} \sim NC\chi^2(k, \lambda)$ for $\lambda = \gamma^{-2} \sum_{j=1}^{k} \mu_j^2$. And since $p(d) = p(g(d)) \cdot |g'(d)|$ for monotonic function g, we have

$$p(d) = NC\chi^2(d\gamma^{-2} \mid k, \lambda)\gamma^{-2}. \tag{4.2}$$

We have yet to define the scaling parameter, γ^2, which was introduced to approximate the predictive distribution over the difference vector, $p_0(\Delta) = \mathcal{N}(\Delta \mid \mu, \Sigma)$, with an isotropic normal, $p_1(\Delta) = \mathcal{N}(\Delta \mid \mu, \mathbf{I}\gamma^2)$. To specify a notion of optimality for γ^2, we will rely on the KL divergence, $\mathrm{KL}[p_0 \parallel p_1]$, which quantifies the amount of information lost by replacing the "true" distribution, p_0, with an approximate distribution, p_1. When both distributions are multivariate normals this expression is analytically tractable, which allows us to find the minimum as a function of γ^2:

$$\begin{aligned}
\mathrm{KL}[p_0 \parallel p_1] &= \int p_0(\Delta) \log \frac{p_0(\Delta)}{p_1(\Delta)} \,\mathrm{d}\Delta \\
&= \frac{1}{2}\left(\mathrm{Tr}((\mathbf{I}\gamma^2)^{-1}\Sigma) - k + \log\left(\frac{\det \mathbf{I}\gamma^2}{\det \Sigma}\right) + (\mu - \mu)^T (\mathbf{I}\gamma^2)^{-1} (\mu - \mu)\right) \\
&= \frac{1}{2}\left(\gamma^{-2} \sum_{j=1}^{k} \sigma_j^2 - k + k \log \gamma^2 - \log \det \Sigma\right). \\
&\triangleq v(\gamma^2),
\end{aligned}$$

where $[\sigma_1^2, \ldots, \sigma_k^2]^T = \mathrm{diag}(\Sigma)$ are the marginal variances of p_0. Taking the derivative w.r.t. γ^2 we have:

$$v'(\gamma^2) = -\frac{1}{2\gamma^2}\left(\frac{1}{\gamma^2} \sum_{j=1}^{k} \sigma_j^2 + k\right),$$

from which we can identify the (only) stationary point at $\gamma^2 = \frac{1}{k}\sum_{j=1}^{k} \sigma_j^2$. And since v is convex and strictly positive it follows that this is the optimal setting of γ^2.

With γ^2 set as the average variance across output dimensions, it turns out that our distance distribution, $p(d) = NC\chi^2(d\gamma^{-2} \mid k, \lambda)\gamma^{-2}$, is unbiased. To see this, first note that the mean of a $NC\chi^2$ distributed variable is conveniently given by $k + \lambda$. The expectation of d under the scaled $NC\chi^2$ is therefore $\mathbb{E}[d] = \mathbb{E}[d\gamma^{-2}]\gamma^2 = (k + \lambda)\gamma^2$. If instead we let the expectation distribute over the sum of squared residuals we have:

$$\mathbb{E}[d] = \sum_{j=1}^{k} \mathbb{E}[\Delta_j^2] = \sum_{j=1}^{k} \mathbb{E}[\Delta_j^2 \sigma_j^{-2}]\sigma_j^2.$$

Since $\Delta_j^2 \sigma_j^{-2} \sim NC\chi^2(1, \mu_j \sigma_j^{-2})$, we have $\mathbb{E}[\Delta_j^2 \sigma_j^{-2}] = 1 + \mu_j \sigma_j^{-2}$, and so:

$$\mathbb{E}[d] = \sum_{j=1}^{k} (1 + \mu_j^2 \sigma_j^{-2})\sigma_j^2 = \sum_{j=1}^{k} \sigma_j^2 + \sum_{j=1}^{k} \mu_j^2 = (k + \lambda)\gamma^2.$$

In summary, we have constructed a new surrogate model that incorporates the individual outputs of h in order to provide a more realistic predictive distribution over the target distance. In doing so we have made the concession of approximating the multivariate predictive distribution over $\Delta = h - \mathbf{y}^\star$ with an isotropic normal in order to obtain a $NC\chi^2$ distribution for d. This simplification, however, has the benefits that it i. will allow us to derive efficient expressions for the EI and LCB acquisition functions; ii. is unbiased w.r.t. the true $G\chi^2$ distribution; and iii. is optimal as measured under the KL divergence. The advantage of incorporating individual outputs of h in the surrogate model is demonstrated in Figure 4.2b. Even though the distance measurements themselves are not very informative if taken in isolation, we manage to obtain a good estimation of the true distance due to the well-modelled individual outputs.

We also note that in the special case of h only having one output, our method is still applicable without any approximations being made, since in that case the $G\chi^2$ and $NC\chi^2$ distributions are identical.

4.3.2 Updating the acquisition functions

Once we have acquired a predictive distribution over the quantity that is being minimised, we need to consider how such a distribution should be mapped to a querying utility by means of an acquisition function. There are a plethora of interesting acquisition functions available (see Shahriari et al. [2015] for an overview), but the two most widely used are arguably the Expected Improvement (EI) and Lower Confidence Bound (LCB). We will restrict attention to these for the remainder of this chapter.

EI considers the improvement that is expected to be achieved over the current incumbent,

d_{min}:

$$\alpha_{\text{EI}}(\mathbf{x}; \mathcal{D}) = \mathbb{E}_{p(d|\mathbf{x},\mathcal{D})}[\max(d_{\text{min}} - d, 0)] = \int_c^{d_{\text{min}}} p(d \mid \mathbf{x}, \mathcal{D})(d_{\text{min}} - d)\,\mathrm{d}d,$$

where c is the minimum of the distribution domain. LCB is defined as the negative[2] minimum of a pre-specified confidence interval:

$$\alpha_{\text{LCB}}(\mathbf{x}; \mathcal{D}) = -F^{-1}(q \mid d), \tag{4.3}$$

where F^{-1} is the inverse CDF (or quantile function) associated with d, while q is the quantile of the confidence interval. This expression can be interpreted as mapping a given input location to the most optimistic outcome of said interval. Reducing the value of q results in a larger confidence interval which in turn yields a more exploratory search strategy. When d is normally distributed, the expression can be straightforwardly written as:

$$\alpha_{\text{LCB}}(\mathbf{x}; \mathcal{D}) = -\mu - \Phi^{-1}(q)\sigma,$$

where μ and σ are the mean and standard deviation for d and Φ is the standard normal CDF. Adopting the conventional notation we define $\beta := -\Phi^{-1}(q)$.

In order to derive these functions under the scaled $NC\chi^2$ distribution we will require three statistical quantities: The expected value for a truncated domain, $\mathbb{E}[d \mid d < d_{\text{min}}]$; the CDF, $F : [0, \infty] \to (0, 1)$, associated with d; and the quantile function, $F^{-1} : (0, 1) \to [0, \infty]$. Starting with the CDF, we use the results of Sankaran [1959] where it is shown that given $t \sim NC\chi^2(k, \lambda)$ we can make a transformation, $z = g(t)$, such that z is approximately normally distributed. This transformation is specified as follows:

$$z = g(t) = \left(\frac{t}{k + \lambda}\right)^{\ell}, \qquad \ell = 1 - \frac{r_1 r_3}{3 r_2^2}, \qquad r_s = 2^{s-1}(s-1)!(k + s\lambda),$$

resulting in z approximately following $\mathcal{N}\left(z \mid \nu, \rho^2\right)$ with the statistics given by:

$$\begin{aligned}
\nu &= 1 + \ell(\ell - 1)\left(\frac{r_2}{2r_1^2} - (2 - \ell)(1 - 3\ell)\frac{r_2^2}{8r_1^4}\right), \\
\rho &= \frac{\ell\sqrt{r_2}}{r_1}\left(1 - \frac{(1 - \ell)(1 - 3\ell)}{4r_1^2} r_2\right).
\end{aligned}$$

The approximation accuracy of this transformation is illustrated in Figure 4.4. Since the CDF is invariant for monotonic mappings we have $F_{\lambda,k}(b) \approx \Phi\left(\frac{g(b)-\nu}{\rho}\right)$, where we let $F_{\lambda,k}$ be the CDF for $NC\chi^2(k, \lambda)$. This immediately yields an approximation for the quantile function,

[2]The negation is applied to frame the acquisition function as an objective to be maximised.

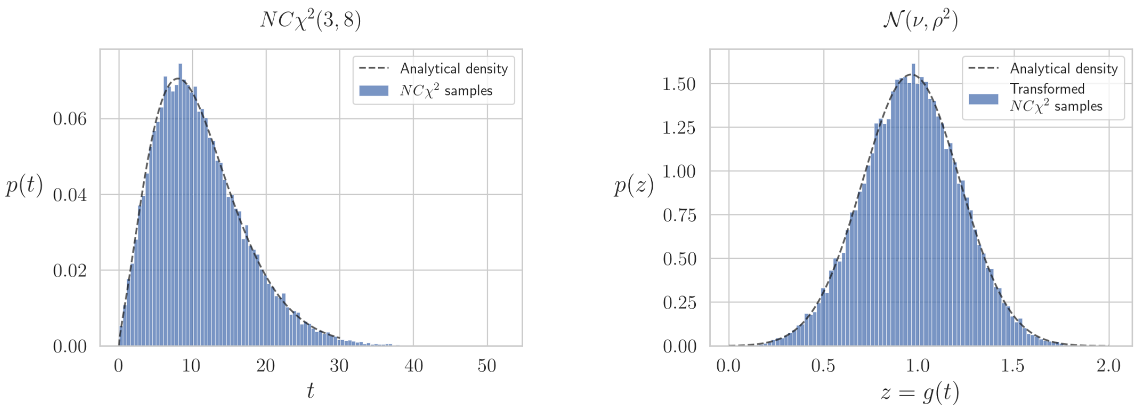

Figure 4.4: Example of the transformation $z = g(t)$, where $p(t) = NC\chi^2(t \mid 3, 8)$ and $p(z) \approx \mathcal{N}(z \mid \nu, \rho^2)$. The left histogram shows the empirical distribution based on 2^{15} samples from $NC\chi^2(t \mid 3, 8)$ while the right histogram is based on the transformation of those samples (and *not* samples from the true normal distribution). We see a very close correspondence between the approximate and the exact normal.

$F^{-1}_{\lambda,k}(q) \approx g^{-1}\left(\Phi^{-1}(q)\rho + \nu\right)$. Lastly, we will use the results of Marchand [1996] in which the following expression for the truncated mean is derived:

$$\mathbb{E}[t \mid t < b] = k \cdot F_{k+2,\lambda}(b) + \lambda F_{k+4,\lambda}(b).$$

We are now ready to specify α_{EI} and α_{LCB} under $p(d \mid \mathbf{x}, \mathcal{D}) \approx NC\chi^2(d\gamma^{-2} \mid k, \lambda)\gamma^{-2}$. Defining $t \triangleq d\gamma^{-2}$ such that $t \sim NC\chi^2(k, \lambda)$ and $\eta \triangleq d_{\text{min}}\gamma^{-2}$ we have:

$$\begin{aligned}\alpha_{\text{EI}}(\mathbf{x}; \mathcal{D}) &= \int_0^{d_{\text{min}}} p(d \mid \mathbf{x}, \mathcal{D})(d_{\text{min}} - d)\,\mathrm{d}d \\ &= \int_0^{d_{\text{min}}} NC\chi^2(d\gamma^{-2} \mid \mathbf{x}, \mathcal{D})\gamma^{-2}(d_{\text{min}} - d)\,\mathrm{d}d \\ &= \int_0^{\eta} NC\chi^2(t \mid \mathbf{x}, \mathcal{D})\gamma^{-2}(d_{\text{min}} - t\gamma^2)\,\mathrm{d}t \\ &= \eta F_{k,\lambda}(\eta) - \mathbb{E}[t \mid t < \eta]. \end{aligned} \tag{4.4}$$

For LCB we first note that $F(d) = F_{k,\lambda}(d\gamma^{-2}) \approx \Phi\left(\frac{g(d\gamma^{-2})-\nu}{\rho}\right)$. Solving for $F(d) = q$ we obtain

$$d = F^{-1}(q) \approx g^{-1}(\Phi^{-1}(q)\rho + \nu) = \sqrt[\ell]{\Phi^{-1}(q)\rho + \nu} \cdot (k + \lambda)\gamma^2.$$

And substituting in $\beta = -\Phi^{-1}(q)$ to match the signature of the Gaussian LCB, we arrive at

$$\alpha_{\text{LCB}}(\mathbf{x}; \mathcal{D}, \beta) = -\sqrt[\ell]{\nu - \beta\rho} \cdot (k + \lambda)\gamma^2. \tag{4.5}$$

Note that both (4.4) and (4.5) are closed-form computable and differentiable. The latter

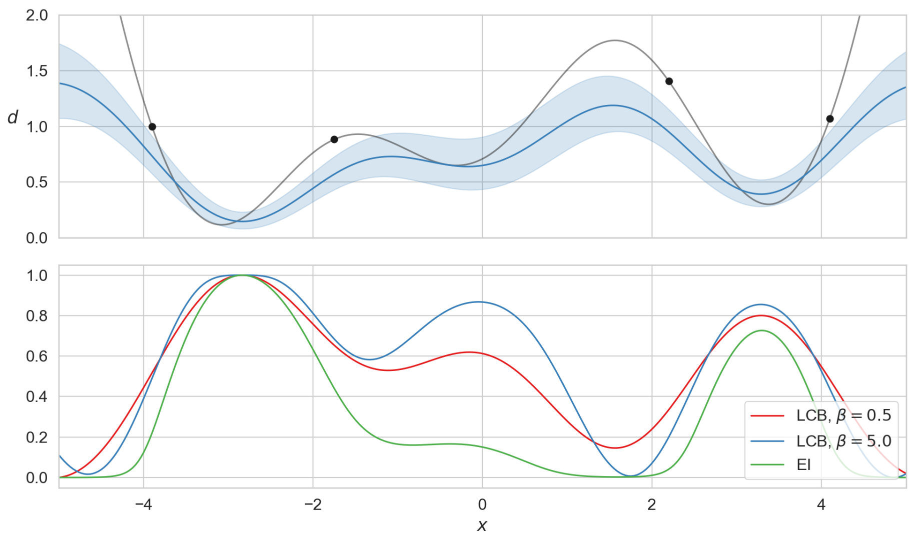

Figure 4.5: The acquisition functions calculated for the inferred distance distribution from Figure 4.2b, normalised for comparison. We evaluate α_{LCB} both for $\beta = 0.5$ (least explorative) and $\beta = 5.0$ (most explorative).

quality is especially important when searching over a large domain in $\mathcal{X}$ where we want to leverage quasi-Newton optimisation techniques to efficiently identify maxima.

Figure 4.5 shows the acquisition functions when applied to the example from Figure 4.2b. We see that a larger β-value for α_{LCB} results in more utility being assigned to the uncertain area around 0. Of the three acquisition functions, α_{EI} seems to be the least explorative. However, they all agree on $x = -2.8$ as the best location for the next query.

4.4 Experiments

In this section we evaluate our method on synthetic benchmark data as well as two real-world problems. We wish to determine whether the proposed ℓ_2 surrogate model, that incorporates individual outputs of the blackbox function and infers a more appropriate predictive distribution, leads to significantly better performance for the target vector estimation task.

As a baseline we use the “standard” Bayesian optimisation approach where the distance is modelled with a single-output GP (see Figure 4.2 for an illustration of the difference between the two models). We consider both EI and LCB for each surrogate model, yielding 4 combinations in total. The metric we use for performance is simply the smallest distance achieved after a pre-determined number of iterations. For all models we use a RBF kernel with automatic relevance detection (ARD) and a Gaussian likelihood. The hyperparameters (likelihood noise, kernel lengthscales, and kernel variance) are optimised with type II maximum likelihood estimation between queries to the blackbox function. The exploration

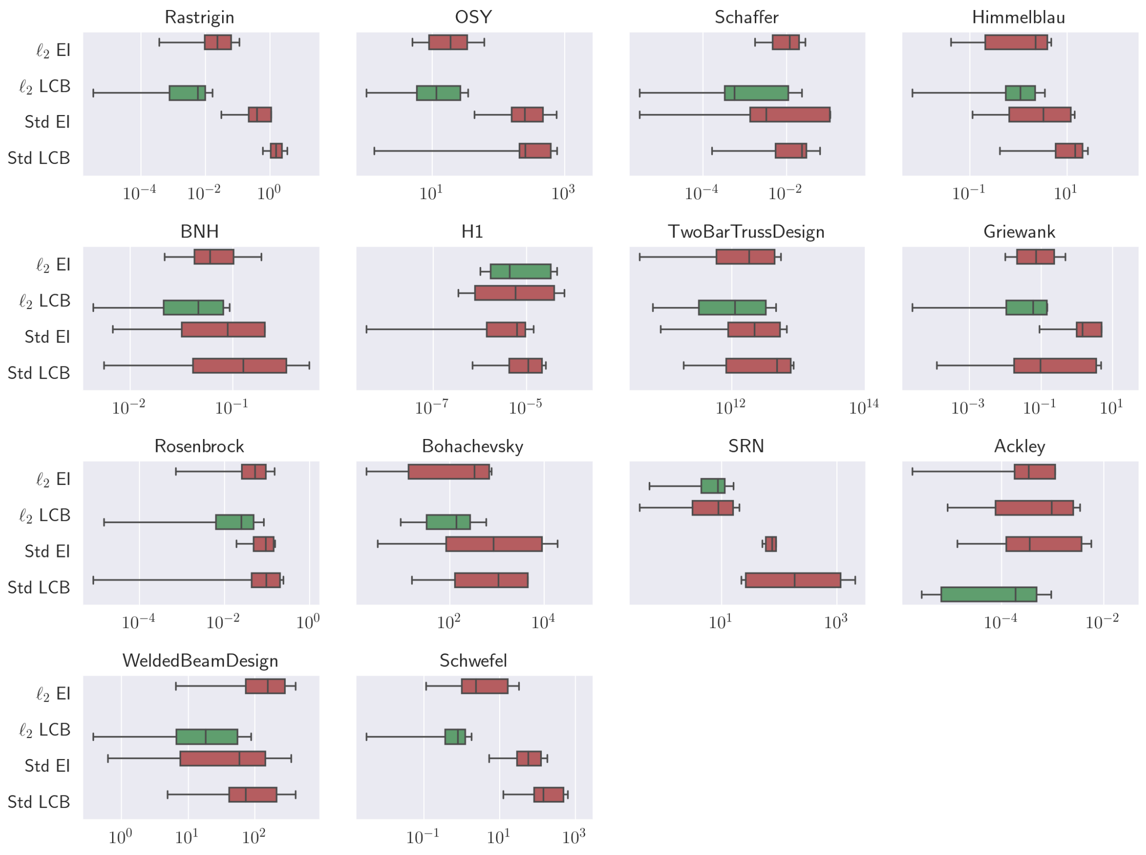

Figure 4.6: Distribution over smallest distances obtained after 30 iterations with 8 repetitions. We denote the proposed surrogate model as $\boldsymbol{\ell_2}$ and the baseline as **Std**. The body of the box is the interquartile range, the whiskers indicate the minimum and maximum, and the median is marked by the vertical line in each box. The green boxes are those for which the median is lowest for the given function.

parameter for the LCB acquisition function is fixed to $\beta = 2$ for both surrogate models.

Synthetic functions

We first test our approach on 14 synthetic functions, 9 of which have a single output while the remaining have between 2 and 6 output dimensions. Refer to Appendix C for further details. These functions are commonly used for benchmarking optimisation methods where the goal is to either locate a global minimum/maximum (when the function has a single output) or identify the Pareto frontier (when the function has multiple outputs). To cast the problem as target vector estimation we first sample an output, $\mathbf{y}^\star = h(\mathbf{x}^\star)$, at random from the given objective function, and then seek to minimise the distance to $\mathbf{y}^\star$. We initialise with 5 points selected by Latin hypercube sampling, run the optimisation procedure for 30 iterations, and report on the lowest distance obtained over the course of optimisation. This is repeated 8 times for each combination of surrogate model, acquisition function, and objective function while keeping $\mathbf{y}^\star$ fixed.

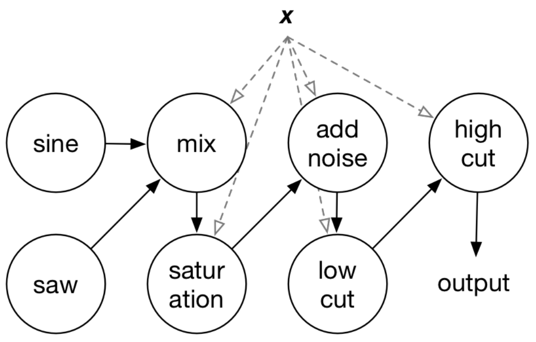

Figure 4.7: Sound processing pipeline for our synthesizer. Two oscillators, generating resp. sinusoidal and saw waves, are first mixed. We then saturate the signal, apply white noise, and filter out low and high frequencies. For each step of this pipeline we associate a free parameter that can be adjusted to shape the sound. The objective is to find the parameterisation that reproduces a target sound.

The benchmark functions are all deterministic so in order to induce stochasticity we add normally distributed noise to the output of all evaluations, scaled by the output range. Specifically we use

$$\mathbf{y} = h(\mathbf{x}) + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \text{diag}(\mathbf{v})\right), \qquad \mathbf{v} = [\delta(h_1), \delta(h_2), \ldots, \delta(h_k)]^T \cdot 10^{-2},$$

where $\delta(h_j)$ is the difference between the highest and the lowest value of dimension j from 10,000 random evaluations of h.

The results are summarised in Figure 4.6 where we show the distribution over minimum distances obtained for each method. Our surrogate model obtains the best median distance for all but one objective function, and we generally see best performance for the LCB acquisition function.

Audio target estimation

We next consider a problem from the field of music production where the aim is to reproduce a target sound with a musical synthesizer (Horner et al. [1993], Garcia [2001], Lai et al. [2006], Heise et al. [2009]). This might be relevant when a musician replaces their old synthesizer with a new model that has a different sound engine and/or user interface. The task of recreating all previously programmed sounds before the next concert is a process that can be both tedious and time-consuming when done manually.

We can frame this as target vector estimation by first projecting the sounds into some latent space in which Euclidean distances are representative of perceptual similarity. Although we will use a simple deterministic representation for our experiment, we note that more advanced methods have been considered for learning an appropriate representation space for sound perception using e.g. deep belief networks (Hamel and Eck [2010]), autoencoders

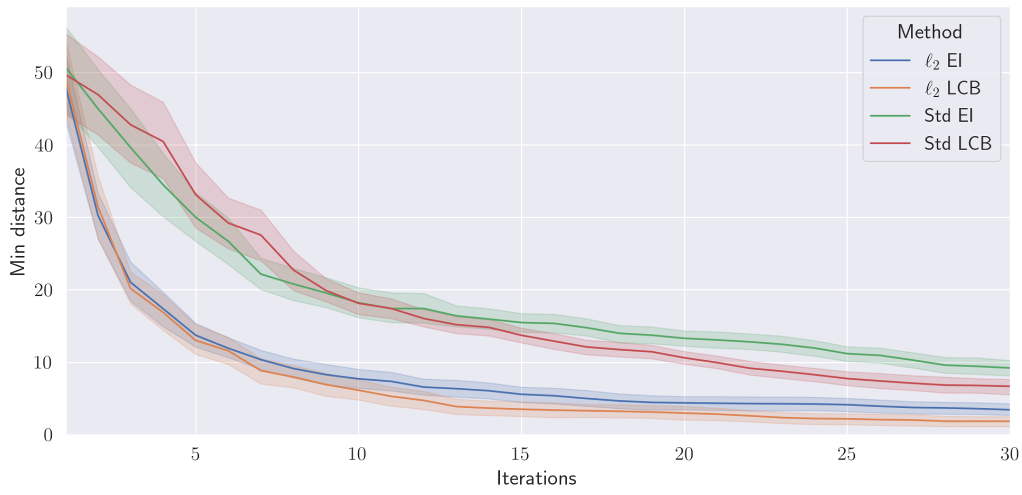

Figure 4.8: Optimisation trace for the audio estimation experiment. For each method we show the incumbent distance over 30 iterations, aggregated across 8 repetitions. The lines show the median distance while the shading is the interquartile range.

(Mor et al. [2018]), or siamese networks (Manocha et al. [2018]). In this setting, $\mathcal{X}$ is the set of configurable synthesizer parameters, $\mathcal{Y}$ is the set of projected sounds, and our blackbox function $h : \mathcal{X} \rightarrow \mathcal{Y}$ comprises the process of generating a sound with the synthesizer and projecting it into $\mathcal{Y}$.

As a simple use case we programmed a synthesizer containing typical audio processing operations, depicted in Figure 4.7. The output sound was projected from the discrete spectral domain into a 10-dimensional representation obtained by principal component analysis. We then followed the same procedure as for the synthetic experiments by first sampling a random configuration of the synthesizer and using the resulting sound as the target for each method to reproduce. The optimisation trace over 8 repetitions are showed in Figure 4.8. We see a significantly faster convergence for the ℓ_2 model and again a superior performance when relying on the LCB acquisition function.

Topography design for stem cell development

Lastly, we turn to the field of bioengineering where it has been widely demonstrated that morphological changes can be induced in stem cell material through the manipulation of nanotopographies. By placing cells on a surface with nanopits of varying sizes and geometric configurations it is possible to promote different gene expressions throughout cell development. This line of research is relevant both for gaining insights about the biological processes, that account for the stem cell development, and for generating cell tissue for medical applications, e.g. when producing biological implants or replacing cells after tumour removal.

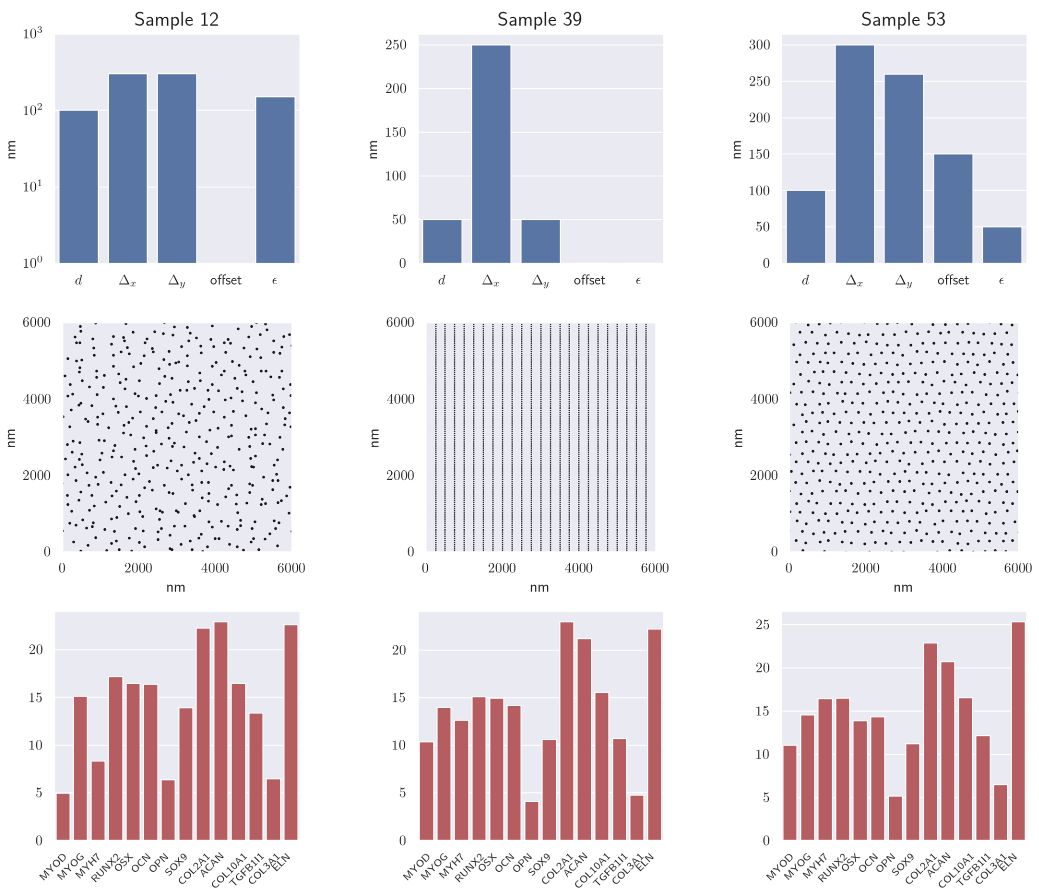

Figure 4.9: Topography and qPCR measurements for 3 random samples. The top row shows the 5 parameters that constitute the search domain: pit diameter, vertical spacing, horizontal space, row offset, and noise. The second row depicts the corresponding topography designs (these are not shown explicitly to the model). The third row are the qPCR measurements for 14 genes pertaining to the given topography.

We consider the task of identifying a nanotopography that reproduces some prespecified set of gene expressions. Following the approach of Cutiongco et al. [2020] we let the topography be determined by 5 parameters controlling the resp. vertical and horizontal spacing, the diameter of the pits, the offset between subsequent rows, and the amount of uniform noise added to the pit coordinates. The gene expressions are encoded as a 14-dimensional vector of quantitative Polymerase Chain Reaction (qPCR) values. See Figure 4.9 for example topographies and their corresponding qPCR measurements. As such, our blackbox function is the entire process of printing the nanotopography, letting the stem cells undergo morphological changes, and then measuring the qPCR values. This usually takes 2 to 4 weeks which motivates the need for an informed search in order to reduce the number of iterations required for identifying an appropriate topography.

Conducting the full experiment for a satisfactory number of iterations and repetitions would potentially take years, so we instead consider a post hoc simulation based on 76 samples col-

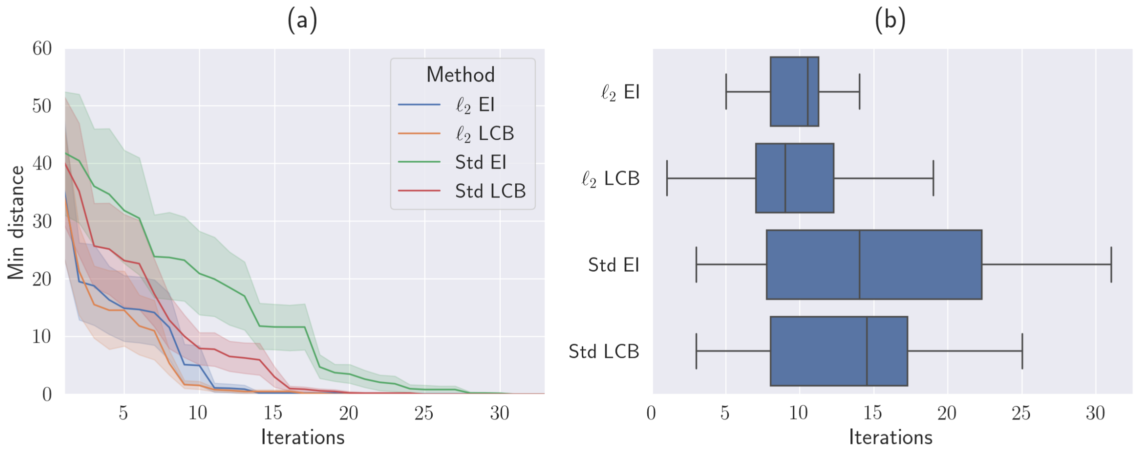

Figure 4.10: Results for the topography experiment. **(a)** The incumbent distance over iterations, aggregated across 20 repetitions. The solid lines are the medians and the shadings are the interquartile ranges. **(b)** The distribution of number of required iterations before the target topography was identified. The boxes are the interquartile ranges, the whiskers are the minimum and maximum values, and the vertical line within each box is the median.

lected for Cutiongco et al. [2020]. We first select a target qPCR vector, sampled at random from the 76 observations, and present the surrogate model with 5 initial topography/qPCR pairs, again chosen at random. In each iteration the surrogate model now selects one topography from the remaining, unseen observations, and we terminate once the target topography is found. The experiment was repeated 20 times for each condition.

Note that, in contrast to the previous experiments, we do not actually optimise over acquisition values in the input space in each iteration; instead we evaluate the acquisition function for all unobserved topographies and select the maximum. This discretisation is of course not ideal, but we argue that the benefits of a more effective search in the discrete domain should carry over to the realistic setting of searching over a continuous topography space.

The results are shown in Figure 4.10. To the left we show the aggregated optimisation traces which demonstrate consistently quicker convergence for the ℓ_2 surrogate model. To the right we show the distribution of the number of required iterations before the target topography was found. As for the other experiments, the ℓ_2 surrogate model significantly outperforms the standard GP, and we get slightly better results when using the LCB acquisition function.

4.5 Conclusion and future work

In this chapter we considered the task of estimating a target vector under the Bayesian optimisation framework. We showed that the problem is compatible with standard Bayesian optimisation by treating the distance between target and function output as an arbitrary cost to be minimised. However, this approach relies on overly simplistic assumptions about the system being modelled which in turn negatively impacts the overall optimisation procedure. Specifically, by employing a standard GP to model the distance we discard potentially valuable information about the multiout blackbox function whilst enforcing an inappropriate predictive distribution.

To correct these deficiencies we proposed to update the modelling assumptions of the surrogate model such that all outputs are explicitly incorporated. We developed approximations for the resulting predictive posterior and two of the most widely used acquisition functions. The result is a demonstrably more data efficient optimisation procedure that still allows for second-order numerical optimisation of the acquisition functions and does not rely on computationally demanding Monte Carlo methods.

Future work on this optimisation procedure may look at improving our approximation of $p(d \mid \mathbf{x}, \mathcal{D})$, which in its current form ignores correlations between outputs in the MTGP surrogate model. While the true distance distribution is known to be a generalised χ^2 its CDF, quantile function, and truncated expectation have yet to be derived at the time of writing. This prohibits the development of the EI and LCB acquisition functions, although alternatives could be considered. Our approach has already been applied for quantum dynamics (Deng et al. [2020]) and rainfall simulations (Hesslow [2020]), but there are still many use cases, some of which are listed in the introduction, that fit our problem formulation and could be interesting to revisit using our framework. In particular, we are currently testing our method in approximate Bayesian computation (Järvenpää et al. [2018]) to speed up the calibration process of costly simulations.

Chapter 6

Conclusion and discussion

This thesis has examined the role of assumptions in Gaussian processes with respect to resource utilisation. We have cast both the model specification and the accommodating inference scheme as a codification of a set of (sometimes unacknowledged) assumptions that bound the potential flexibility of the model and determine the resource efficiency in terms of training data and computational requirements. Much recent research has been devoted to adjusting these assumptions with the aim of obtaining more flexible models that scale well to large datasets and take full advantage of contemporary computing power. This thesis looks in the opposite direction and claims that identifying and revising one's assumptions can lead to models that perform drastically better when data and computational resources are scarce.

In Chapter 3 we treated the main assumption of the sparse variational Gaussian process (SVGP) framework, namely that the true posterior process can be sufficiently approximated by conditioning on a deterministically chosen set of inducing points. We argued that this assumption is not wholly compatible with the philosophy of Bayesian nonparameterics, since it bounds the potential capacity and complexity of the model rather than letting both grow with the amount of data. Furthermore, it forces the model designer to make a deterministic choice about a crucial variable – the number of inducing points – rather than associating it with uncertainty. We solved these issues by expanding the hierarchical model to include the sampling of inducing points from a prior point process. In effect, the selection of inducing points became part of the inference rather than the model design, yielding a new paradigm for managing computational complexity that is applicable for both standard SVGP regression and more exotic variants such as deep Gaussian process and latent variable models.

In Chapter 4 we focused on the problem of target vector estimation under the Bayesian optimisation framework with a Gaussian process acting as surrogate model. Here, a naive approach encodes a set of assumptions about the blackbox function that results in valuable information being lost, and a predictive distribution that is ill-suited for reasoning about distances. We showed that by letting the surrogate model align more closely with the blackbox function, and by making the necessary updates to the acquisition functions, we obtain a significantly more efficient optimisation procedure.

Finally, Chapter 5 treated a fundamental assumption of supervised learning in the Gaussian process framework, namely that the inputs are sufficient for explaining the outputs. We demonstrated that relaxing this assumption in the context of preference learning leads to a model that allows for transferring information between users through latent features, and for the derivation of complex posterior distributions pertaining to any given user's utility function. When compared to benchmark methods we then require less data to learn how individual users assign utility, and how different users relate to each other in terms of preference. The approach draws many parallels to various multitask Gaussian process models but also unifies ideas from the areas of data imputation and density estimation.

The main contribution of this thesis is the examination and modification of assumptions that are problematic in terms of efficient resource exploitation within the three research areas outlined above. In a broader perspective, the thesis pushes back against the blackbox approach towards machine learning where one's model is treated as an opaque oracle whose inner workings are poorly understood and of secondary interest. While this strategy is not without merit, we argue that many tasks, for which probabilistic modelling has the potential to prove very useful, will benefit from incorporating strong assumptions derived from expert knowledge, prior empirical evidence, or just plain common sense. The capacity for incorporating such insights in the modelling in an intuitive manner has always been a prime advantage of Gaussian processes. At the same time, it yields an impetus for actually dissecting one's model and considering, whether counterproductive assumptions have been inadvertently included. This is especially pertinent as ever more complex models are constantly being developed, for which it might be easy to lose sight of the underlying assumptions that precede any inference task. We hope to have shown in this thesis that identifying and revising such assumptions leads to Gaussian process models that can do more with fewer resources.

6.1 Future work

As argued in the introduction, we do not expect that advancements in computing power and the ever increasing amount of training data will render parsimonious models obsolete. Rather, the need for low capacity devices and learning from esoteric data sources should encourage more interest in models such as Gaussian processes where strong prior assumptions may enable small-scale learning. For the topics treated in this thesis in particular, we see the following potential paths of research:

- **Complexity reduction:** Within the field of neural networks the need for managing the computational complexity of one's model has long been acknowledged. This has prompted data-driven methods for deriving lean models, e.g. probabilistic pruning (Dikov and Bayer [2019]) and architecture search through Bayesian optimisation (Le Goff et al. [2020]). Concurrently, the interest in increasingly elaborate Gaussian processes modelling, e.g. deep Gaussian processes, is on the rise, and these rely primarily on the inducing point methodology. We therefore advocate a general purpose approach for adapting the complexity of such models through the concepts introduced in Chapter 3. For specific avenues of advancements we point to i. the revision or complete abandonment of score function estimation as the underlying optimisation method, due to the high variance in gradient estimation; and ii. the introduction of more flexible inference schemes, e.g. using a determinantal point process as the proposal distribution, or using Monte Carlo sampling for estimating the posterior.

- **Bayesian optimisation:** The interest surrounding Bayesian optimisation is not set to disappear soon due to its almost universal applicability. Especially its role in automated machine learning, or AutoML (He et al. [2021]), has linked the optimisation framework inextricably to the continued development of neural networks, meaning that research interest is bound to remain high in the foreseeable future. In terms of methodological advancements, there are multiple avenues currently being explored such as incorporating representation learning in the optimisation (Gómez-Bombarelli et al. [2018]) and using observations at multiple fidelities to inform a holistic surrogate model (Imani et al. [2019], Cutajar et al. [2019]). We believe that such advancements calls for a reexamination of the assumptions placed on the blackbox function to avoid the pathological behaviour similar to what we identified for the target vector estimation task.

- **Latent feature learning:** Finally, for the joint learning of a universal function and latent input features that we considered in Chapter 5, we believe there is much ground to be covered outside the field of preference learning. For instance, the stem cell experiment that we conducted in Chapter 4 relies on a surrogate model which represents the biological process that a cell undergoes from nanotopography to gene expressions. We are currently investigating whether datasets obtained from multiple cell types, which are otherwise incongruent, can be fused by including latent features that explain their discrepancy. As in the preference learning scenario, this would serve the orthogonal purposes of increasing the predictive capacity for each individual cell type, and yielding insights into which cell types react similarly to various topographies.

Bibliography

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation OSDI 16)*, pages 265–283, 2016.

A. M. Alaa, J. Yoon, S. Hu, and M. Van der Schaar. Personalized risk scoring for critical care prognosis using mixtures of Gaussian processes. *IEEE Transactions on Biomedical Engineering*, 65(1):207–218, 2017.

R. Astudillo and P. Frazier. Bayesian optimization of composite functions. In *International Conference on Machine Learning*, pages 354–363. PMLR, 2019.

A. Asuncion and D. Newman. UCI machine learning repository, 2007.

M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

A. Birlutiu, P. Groot, and T. Heskes. Multi-task preference learning with an application to hearing aid personalization. *Neurocomputing*, 73(7-9):1177–1185, 2010.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

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

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

E. V. Bonilla, S. Guo, and S. Sanner. Gaussian process preference elicitation. *Advances in Neural Information Processing Systems*, 23:262–270, 2010.

E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

T. D. Bui, C. Nguyen, and R. E. Turner. Streaming sparse Gaussian process approximations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 3299–3307. 2017a.

T. D. Bui, C. V. Nguyen, and R. E. Turner. Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 3300–3308, 2017b.

D. Burt, C. E. Rasmussen, and M. Van Der Wilk. Rates of convergence for sparse variational Gaussian process regression. In *International Conference on Machine Learning*, pages 862–871, 2019.

H. Chen, L. Zheng, R. Al Kontar, and G. Raskutti. Stochastic Gradient Descent in Correlated Settings: A Study on Gaussian Processes. *Advances in Neural Information Processing Systems*, 33, 2020.

T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Computer Architecture News*, 42(1):269–284, 2014.

C.-A. Cheng and B. Boots. Variational Inference for Gaussian Process Models with Linear Complexity. In *Advances in Neural Information Processing Systems*, page 5190–5200, 2017.

W. Chu and Z. Ghahramani. Preference learning with Gaussian processes. In *International Conference on Machine Learning*, pages 137–144, 2005.

C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

L. Csató and M. Opper. Sparse online Gaussian processes. *Neural Computation*, 14(3): 641–668, 2002.

K. Cutajar, M. Pullin, A. Damianou, N. Lawrence, and J. González. Deep Gaussian processes for multi-fidelity modeling. *arXiv preprint arXiv:1903.07320*, 2019.

M. F. Cutiongco, B. S. Jensen, P. M. Reynolds, and N. Gadegaard. Predicting gene expression using morphological cell responses to nanotopography. *Nature communications*, 11(1):1–13, 2020.

A. Damianou. *Deep Gaussian processes and variational propagation of uncertainty*. PhD thesis, University of Sheffield, 2015.

A. Damianou and N. Lawrence. Deep Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 207–215, 2013.

M. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472. Citeseer, 2011.

Z. Deng, I. Tutunnikov, I. S. Averbukh, M. Thachuk, and R. Krems. Bayesian optimization for inverse problems in time-dependent quantum dynamics. *The Journal of Chemical Physics*, 153(16):164111, 2020.

G. Dikov and J. Bayer. Bayesian learning of neural network architectures. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 730–738. PMLR, 2019.

E. A. Durant, G. H. Wakefield, D. J. Van Tasell, and M. E. Rickert. Efficient perceptual tuning of hearing aids with genetic algorithms. *IEEE Transactions on Speech and Audio Processing*, 12(2):144–155, 2004.

V. Dutordoir, H. Salimbeni, M. P. Deisenroth, and J. Hensman. Gaussian process conditional density estimation. In *Advances in Neural Information Processing Systems*, pages 2391—-2401, 2018.

V. Dutordoir, N. Durrande, and J. Hensman. Sparse Gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*, pages 2793–2802. PMLR, 2020.

D. Duvenaud, J. R. Lloyd, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search, 2013.

D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 202–210, 2014.

C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms I: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 28(1):26–37, 1998.

M. P. Fourman. Compaction of symbolic layout using genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 141–153, 1985.

M. C. Fu. Gradient estimation. *Handbooks in operations research and management science*, 13:575–616, 2006.

J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *European Conference on Machine Learning*, pages 145–156. Springer, 2003.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR, 2016.

R. Garcia. Growing sound synthesizers using evolutionary methods. In *ALMMA 2002: Artificial Life Models for Musical Applications Workshop*, 2001.

R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219, 2010.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*. CRC press, 2013.

Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459, 2015.

J. K. Ghosh and R. Ramamoorthi. *Bayesian nonparametrics*. Springer Science & Business Media, 2003.

M. N. Gibbs. *Bayesian Gaussian processes for regression and classification*. PhD thesis, Citeseer, 1998.

A. Girard and R. Murray-Smith. Gaussian processes: Prediction at a noisy input and application to iterative multiple-step ahead forecasting of time-series. In *Switching and Learning in Feedback Systems*, pages 158–184. Springer, 2005.

P. Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013.

V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello. A 240 g-ops/s mobile coprocessor for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 682–687, 2014.

R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

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

P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *ISMIR*, volume 10, pages 339–344. Citeseer, 2010.

O. Hamelijnck, T. Damoulas, K. Wang, and M. Girolami. Multi-resolution Multi-task Gaussian Processes. In *Advances in Neural Information Processing Systems*, pages 14025–14035. 2019.

O. Hau et al. Hearing-aid compression: Effects of channel bandwidth on perceived sound quality. In *Proceedings of the International Symposium on Auditory and Audiological Research*, volume 3, pages 449–455, 2011.

M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes. Deep Gaussian processes with decoupled inducing inputs. *arXiv:1801.02939*, 2018a.

M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes. Inference in Deep Gaussian Processes Using Stochastic Gradient Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, page 7517–7527, 2018b.

X. He, K. Zhao, and X. Chu. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems*, 212:106622, 2021.

M. Heinonen, H. Mannerström, J. Rousu, S. Kaski, and H. Lähdesmäki. Non-stationary Gaussian process regression with Hamiltonian Monte Carlo. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 732–740. PMLR, 2016.

S. Heise, M. Hlatky, and J. Loviscach. Automatic cloning of recorded sounds by software synthesizers. In *Audio Engineering Society Convention 127.* Audio Engineering Society, 2009.

P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

J. M. Henrández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, page 918–926. MIT Press, 2014.

J. Hensman and N. D. Lawrence. Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*, 2014.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2013.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2015a.

J. Hensman, A. G. Matthews, M. Filippone, and Z. Ghahramani. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1648–1656, 2015b.

J. Hensman, N. Durrande, and A. Solin. Variational Fourier Features for Gaussian Processes. *Journal of Machine Learning Research*, 18, 2018.

D. Hesslow. Fast rainfall runoff simulation and parameter optimization. 2020.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

A. Horner, J. Beauchamp, and L. Haken. Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis. *Computer Music Journal*, 17(4):17–29, 1993.

N. Houlsby, F. Huszar, Z. Ghahramani, and J. Hernández-lobato. Collaborative Gaussian processes for preference learning. *Advances in Neural Information Processing Systems*, 25:2096–2104, 2012.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

M. Imani, S. F. Ghoreishi, D. Allaire, and U. M. Braga-Neto. MFBO-SSM: Multi-fidelity Bayesian optimization for fast inference in state-space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7858–7865, 2019.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.

M. Järvenpää, M. U. Gutmann, A. Vehtari, and P. Marttinen. Gaussian process modelling in approximate Bayesian computation to estimate horizontal gene transfer in bacteria. *Annals of Applied Statistics*, 12(4):2228–2251, 2018.

J. M. Jerez, I. Molina, P. J. García-Laencina, E. Alba, N. Ribelles, M. Martín, and L. Franco. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial intelligence in medicine*, 50(2):105–115, 2010.

B. Johansen, M. K. Petersen, M. J. Korzepa, J. Larsen, N. H. Pontoppidan, and J. E. Larsen. Personalizing the fitting of hearing aids by learning contextual preferences from internet of things data. *Computers*, 7(1):1, 2018.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, pages 105–161. Springer, 1998.

A. G. Journel and C. J. Huijbregts. Mining geostatistics. 1976.

P. Jylänki, J. Vanhatalo, and A. Vehtari. Robust Gaussian Process Regression with a Student-t Likelihood. *Journal of Machine Learning Research*, 12(11), 2011.

G. Keidser, C. Brew, S. Brewer, H. Dillon, F. Grant, and L. Storey. The preferred response slopes and two-channel compression ratios in twenty listening conditions by hearing-impaired and normal-hearing listeners and their relationship to the acoustic input. *International Journal of Audiology*, 44(11):656–670, 2005.

M. C. Killion. New thinking on hearing in noise: A generalized articulation index. In *Seminars in Hearing*, volume 23, pages 47–62, 2002.

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

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, 2015.

J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

A. Kulesza and B. Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012. ISBN 1601986289.

Y. Lai, S. Jeng, D. Liu, and Y. Liu. Automated optimization of parameters for FM sound synthesis with genetic algorithms. In *International Workshop on Computer Music and Audio Technology*, 2006.

H. C. L. Law, D. Sejdinovic, E. Cameron, T. C. Lucas, S. Flaxman, K. Battle, and K. Fukumizu. Variational learning on aggregate outputs with Gaussian processes. *arXiv preprint arXiv:1805.08463*, 2018.

N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems*, pages 609–616, 2003.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*, pages 329–336, 2004.

N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with Gaussian processes. In *International Conference on Machine Learning*, pages 601–608, 2009.

M. Lázaro-Gredilla and A. Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, pages 1087–1095, 2009.

M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

L. K. Le Goff, E. Buchanan, E. Hart, A. E. Eiben, W. Li, M. de Carlo, M. F. Hale, M. Angus, R. Woolley, J. Timmis, et al. Sample and time efficient policy learning with CMA-ES and Bayesian optimisation. In *Artificial Life Conference Proceedings*, pages 432–440. MIT Press, 2020.

N. A. Lesica. Why do hearing aids fail to restore normal auditory perception? *Trends in Neurosciences*, 41(4):174–185, 2018.

D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

S. K. Lynch, A. Shah, J. C. Folk, X. Wu, and M. D. Abramoff. Catastrophic failure in image-based convolutional neural network algorithms for detecting diabetic retinopathy. *Investigative Ophthalmology & Visual Science*, 58(8):3776–3776, 2017.

D. J. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

C. J. Maddison, A. Mnih, and Y. W. Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *International Conference on Learning Representations*, 2017.

P. Manocha, R. Badlani, A. Kumar, A. Shah, B. Elizalde, and B. Raj. Content-based representations of audio using Siamese Neural Networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3136–3140. IEEE, 2018.

E. Marchand. Computing the moments of a truncated noncentral Chi-square distribution. *Journal of Statistical Computation and Simulation*, 54(4):387–391, 1996.

A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. *Advances in Neural Information Processing Systems*, 20:1257–1264, 2007.

J. Močkus. On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

N. Mor, L. Wolf, A. Polyak, and Y. Taigman. A universal music translation network. *arXiv preprint arXiv:1805.07848*, 2018.

M. S. Murugan, S. Suresh, R. Ganguli, and V. Mani. Target vector optimization of composite box beam using real-coded genetic algorithm: a decomposition approach. *Structural and Multidisciplinary Optimization*, 33(2):131–146, 2007.

D. M. Negoescu, P. I. Frazier, and W. B. Powell. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3):346–363, 2011.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

T. V. Nguyen, A. Karatzoglou, and L. Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 63–72, 2014.

J. B. Nielsen, B. S. Jensen, and J. Larsen. On sparse multi-task Gaussian process priors for music preference learning. In *NIPS Workshop on Choice Models and Preference Learning*. Citeseer, 2011.

J. B. B. Nielsen, J. Nielsen, and J. Larsen. Perception-based personalization of hearing aids using Gaussian processes and active learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1):162–173, 2014.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

S. W. Ober, C. E. Rasmussen, and M. van der Wilk. The promises and pitfalls of deep kernel learning. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2021.

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, 2016.

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

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

G. Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *CUIDADO Ist Project Report*, 54(0):1–25, 2004.

P. Perdikaris and G. E. Karniadakis. Model inversion via multi-fidelity Bayesian optimization: A new paradigm for parameter estimation in haemodynamics, and beyond. *Journal of The Royal Society Interface*, 13(118):20151107, 2016.

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

A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, and T. Wongjirad. Machine learning at the energy and intensity frontiers of particle physics. *Nature*, 560(7716):41–48, 2018.

C. E. Rasmussen and Z. Ghahramani. Occam's razor. *Advances in Neural Information Processing Systems*, pages 294–300, 2001.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, Jan. 2006.

S. Rossi, M. Heinonen, E. Bonilla, Z. Shen, and M. Filippone. Sparse Gaussian Processes Revisited: Bayesian Approaches to Inducing-Variable Approximations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 1837–1845. PMLR, 2021.

D. B. Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

T. N. Sainath, B. Kingsbury, and B. Ramabhadran. Auto-encoder bottleneck features using deep belief networks. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4153–4156. IEEE, 2012.

H. Salimbeni and M. P. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4589–4600, 2017.

H. Salimbeni, C.-A. Cheng, B. Boots, and M. Deisenroth. Orthogonally decoupled variational Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 8711–8720, 2018.

M. Sankaran. On the non-central chi-square distribution. *Biometrika*, 46(1/2):235–237, 1959.

J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the first international conference on genetic algorithms and their applications, 1985*. Lawrence Erlbaum Associates. Inc., Publishers, 1985.

M. Seeger, Y.-W. Teh, and M. Jordan. Semiparametric latent factor models. Technical report, 2005.

A. Shah and Z. Ghahramani. Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning*, pages 1919–1927. PMLR, 2016.

A. Shah, A. G. Wilson, and Z. Ghahramani. Bayesian optimization using Student-t processes. In *NIPS Workshop on Bayesian Optimization*, 2013.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

R. Sheth, Y. Wang, and R. Khardon. Sparse variational inference for generalized Gaussian process models. In *International Conference on Machine Learning*, pages 1302–1311. PMLR, 2015.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2006.

E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduction for sparse Gaussian processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2012.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. *Advances in Neural Information Processing Systems*, 29:4134–4142, 2016.

R. L. Streit. *Poisson point processes: imaging, tracking, and sensing*. Springer Science & Business Media, 2010.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2013.

R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

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

M. Titsias and M. Lázaro-Gredilla. Spike and slab variational inference for multi-task and multiple kernel learning. *Advances in Neural Information Processing Systems*, 24:2339–2347, 2011.

M. K. Titsias, J. Schwarz, A. G. d. G. Matthews, R. Pascanu, and Y. W. Teh. Functional regularisation for continual learning with Gaussian processes. In *International Conference on Learning Representations*, 2019.

A. K. Uhrenholt and B. S. Jensen. Efficient Bayesian Optimization for Target Vector Estimation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 2661–2670. PMLR, 2019.

A. K. Uhrenholt, V. Charvet, and B. S. Jensen. Probabilistic Selection of Inducing Points in Sparse Gaussian Processes. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2021.

M. van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional gaussian processes. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Parallel Bayesian global optimization of expensive functions. *Operations Research*, 68(6):1850–1865, 2020.

K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact Gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pages 14622–14632, 2019.

D. Wienke, C. Lucasius, and G. Kateman. Multicriteria target vector optimization of analytical procedures using a genetic algorithm: Part I. Theory, numerical simulations and application to atomic emission spectroscopy. *Analytica Chimica Acta*, 265(2):211–225, 1992.

C. Williams, E. V. Bonilla, and K. M. Chai. Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems*, pages 153–160, 2007.

C. K. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688, 2001.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

A. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on Machine Learning*, pages 1775–1784, 2015.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 370–378. PMLR, 2016a.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Stochastic variational deep kernel learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 2594–2602, 2016b.

J. T. Wilson, R. Moriconi, F. Hutter, and M. P. Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *International Conference on Machine Learning*, pages 1012–1019, 2005.

Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14, 2010.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine Learning*, page 116, 2004.

E. Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), 1999.

Appendix A

Diminishing gradients for inducing inputs

In Chapter 3, Section 3.2, we argued that when an inducing input, $\mathbf{z}_j$, moves away from the observed data, as measured through the kernel, the gradient for said input w.r.t. the expected log likelihood diminishes:

$$\lim_{\kappa(\mathbf{z}_j, \mathbf{X}) \to \mathbf{0}} \qquad \frac{\partial}{\partial \mathbf{z}_j} \mathbb{E}_{q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z})} \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] = \mathbf{0}.$$

To see this, first note that the function evaluations are conditionally independent given the inducing points, as shown in Section 2.7.1. Consequently, the partial derivative for $\mathbf{z}_j$ distributes as follows:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{z}_j} \mathbb{E}_{q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z})} \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] &= \frac{\partial}{\partial \mathbf{z}_j} \sum_{i=1}^{N} \mathbb{E}_{q(f_i \mid \mathbf{m}, \mathbf{S})} \left[\log p(y_i \mid f_i) \right] \\ &= \sum_{i=1}^{N} \int \left(\frac{\partial \mathcal{N}\left(f_i \mid \mu_i, \sigma_i^2\right)}{\partial(\mu_i, \sigma_i^2)} \cdot \frac{\partial(\mu_i, \sigma_i^2)}{\partial \mathbf{z}_j} \right) \log p(y_i \mid f_i) \, \mathrm{d} f_i, \end{aligned} \tag{A.1}$$

where μ_i and σ_i^2 are the variational statistics for the i'th function evaluation:

$$\begin{aligned} q(f_i \mid \mathbf{m}, \mathbf{S}) &= p(f_i \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u} \mid \mathbf{m}, \mathbf{S}, \mathbf{Z}) \, \mathrm{d}\mathbf{u} \\ &= \mathcal{N}\left(f_i \mid \mu_i, \sigma_i^2\right) \\ \mu_i = \kappa(\mathbf{x}_i, \mathbf{Z})^T \gamma, &\qquad \sigma_i^2 = \kappa(\mathbf{x}_i, \mathbf{x}_i) - \kappa(\mathbf{x}_i, \mathbf{Z})^T \beta \kappa(\mathbf{x}_i, \mathbf{Z}), \\ \gamma = \mathbf{K}_{MM}^{-1} \mathbf{m}^T, &\qquad \beta = \mathbf{K}_{MM}^{-1} (\mathbf{K}_{MM} - \mathbf{S}) \mathbf{K}_{MM}^{-1}. \end{aligned}$$

Considering these quantities as a function of a specific inducing input, $\mathbf{z}_j$, yields the relations:

$$\mu_i(\mathbf{z}_j) \propto \kappa(\mathbf{x}_i, \mathbf{z}_j)[\gamma]_j, \qquad\qquad \sigma_i^2(\mathbf{z}_j) \propto \kappa(\mathbf{x}_i, \mathbf{z}_j) \sum_{j'=1}^{M} \kappa(\mathbf{x}_i, \mathbf{z}_{j'})[\beta]_{j,j'}.$$

Now consider the case of moving away $\mathbf{z}_j$ from all training data such that $\kappa(\mathbf{x}_i, \mathbf{z}_j) \to 0$ for any $\mathbf{x}_i$. The inducing covariance matrix, $\mathbf{K}_{MM}$, is still guaranteed to be invertible so its condition number, α, stays bounded. This, in turn, upper bounds all entries of $\mathbf{K}_{MM}^{-1}$ since:

$$\|\mathbf{K}_{MM}^{-1}\|_\infty = \frac{\alpha}{\|\mathbf{K}_{MM}\|_\infty},$$

where $\|\mathbf{K}_{MM}\|_\infty = \max_{k,\ell}(\kappa(\mathbf{z}_k, \mathbf{z}_\ell)$ is not affected by moving $\mathbf{z}_j$. Consequently, $[\gamma]_j$ and $[\beta]_{j,j'}$ are bounded and so the mean and variance functions become constant in $\mathbf{z}_j$ for $\kappa(\mathbf{x}_i, \mathbf{z}_i) = 0$, implying that the partial derivative tends towards 0:

$$\lim_{\kappa(\mathbf{z}_j, \mathbf{x}_i) \to 0} \qquad \frac{\partial(\mu_i, \sigma_i^2)}{\partial \mathbf{z}_j} = \mathbf{0}.$$

This is then the case for all summands in (A.1), thereby proving the claim.

Appendix B

Re-parameterisation trick for the Poisson Point Process

This section develops a method for using the re-parameterisation trick for learning the parameters, λ, of the variational Poisson point process (PPP):

$$q_\lambda(\mathbf{Z}) = \prod_{\mathbf{z}_k \in \mathbf{Z}} q(\mathbf{z}_k) \prod_{\mathbf{z}_k \notin \mathbf{Z}} (1 - q(\mathbf{z}_k)) = \prod_{\mathbf{z}_k \in \mathbf{Z}} \lambda_k \prod_{\mathbf{z}_k \notin \mathbf{Z}} (1 - \lambda_k).$$

The method uses the framework of Maddison et al. [2017] that presents a continuous relaxation for multinomial variables. A special case is the binary variable, $b \sim \text{Bernoulli}(\phi)$ with probability $\phi \in (0, 1)$, which can be rewritten as

$$b \approx \sigma(\ell(\phi)), \;\; \ell(\phi) = \frac{\log\phi - \log(1-\phi) + \log u - \log(1-u)}{\lambda}, \;\; u \sim \text{Uniform}(0,1).$$

Here, σ is the logistic function and $\lambda > 0$ is a temperature parameter that determines the degree of the relaxation. We then have the approximation:

$$\mathbb{E}_{p_\phi(b)}[g_\theta(b)] \approx \mathbb{E}_{p(u)}[g_\theta(\sigma(\phi))],$$

rendering the optimisation of (ϕ, θ) amenable to the re-parameterisation trick.

To apply this method for our point process estimation, we first introduce a binary vector, $\mathbf{b} \in [0, 1]^K$, that indicates which inducing points are being sampled from $\mathbf{Z}^\star$, i.e. $b_k = 1 \Leftrightarrow \mathbf{z}_k \in \mathbf{Z}$. Next, we introduce a new objective function, $\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b})$, that relies on this binary vector as well as the entire candidate set. This function is developed in the following section and will ensure that $\hat{\mathcal{L}}(\mathbf{Z}^\star; \mathbf{b}) = \mathcal{L}(\mathbf{Z})$. Given a distribution over binary vectors, $q_\lambda(\mathbf{b})$, which awards the same probabilities to subsets as our point process, $q_\lambda(\mathbf{Z})$, it then follows that:

$$\mathbb{E}_{q_\lambda(\mathbf{Z})}\left[\mathcal{L}(\mathbf{Z})\right] = \mathbb{E}_{q_\lambda(\mathbf{b})}\left[\hat{\mathcal{L}}(\mathbf{Z}^\star;\mathbf{b})\right]. \tag{B.1}$$

When $q_\lambda(\mathbf{Z})$ is a PPP, the equivalent distribution over $\mathbf{b}$ is:

$$q_\lambda(\mathbf{b}) = \prod_{k=1}^{K} \lambda_k^{b_k}(1-\lambda_k)^{1-b_k}.$$

Substituting this into (B.1), we have:

$$\mathbb{E}_{q_\lambda(\mathbf{b})}[\hat{\mathcal{L}}(\mathbf{Z}^\star;\mathbf{b})] = \mathbb{E}_{q(b_1)}[\mathbb{E}_{q(b_2)}[\cdots[\mathbb{E}_{q(b_K)}[\hat{\mathcal{L}}(\mathbf{Z}^\star;\mathbf{b})]]\cdots]],$$

where $q(b_k) = \text{Bernoulli}(\lambda_k)$. This now allows for the re-parameterisation trick to be applied for each of the binary expectations.

Masking the bound

The masked bound, $\hat{\mathcal{L}}(\mathbf{Z}^\star;\mathbf{b})$, is obtained by updating the existing kernel function, κ, to the following:

$$\begin{aligned}
\hat{\kappa}_{\mathbf{b}}(\mathbf{x},\mathbf{x}') &= \kappa(\mathbf{x},\mathbf{x}')\cdot\Delta(\mathbf{x},\mathbf{b})\cdot\Delta(\mathbf{x}',\mathbf{b}), && \text{if } \mathbf{x}\neq\mathbf{x}'\\
\hat{\kappa}_{\mathbf{b}}(\mathbf{x},\mathbf{x}) &= \Delta(\mathbf{x},\mathbf{b})(\kappa(\mathbf{x},\mathbf{x})-1)+1, &&\\
\Delta(\mathbf{x},\mathbf{b}) &= \prod_{j=1}^{M} b_j^{\delta(\mathbf{x}-\mathbf{z}_j)}. &&
\end{aligned}$$

Despite the somewhat convoluted expressions, the above update is quite straightforward. It simply replaces the (i,j) entry in the covariance matrix with a value, v, if either b_i or b_j is 0. When the entry is on the diagonal, i.e. $i=j$, we set $v=1$; otherwise we set $v=0$. This operation can be carried out efficiently as a masking operation of the covariance matrix.

The effect of the above modifications is that the *masked* inducing points, for which $b_j = 0$, are factorised into a standard multivariate normal distribution. That is, if we denote the masked points as $\mathbf{u}_C$, and unmasked points as $\mathbf{u} = f(\mathbf{Z}^\star)\setminus\mathbf{u}_C$, we have:

$$p(\mathbf{y},\mathbf{f},\mathbf{u},\mathbf{u}_C \mid \mathbf{X},\mathbf{Z}^\star) = p(\mathbf{y},\mathbf{u},\mathbf{f}\mid\mathbf{X},\mathbf{Z})p(\mathbf{u}_C), \qquad p(\mathbf{u}_C) = \mathcal{N}\left(\mathbf{u}_C \mid \mathbf{0},\mathbf{I}\right).$$

The kernel remains PD under the modifications, meaning that the new prior, $\mathcal{GP}(0,\hat{\kappa}_{\mathbf{b}})$, is still a valid Gaussian process. Using the same masking operation for the variational distribution, we now have:

$$\begin{aligned}
\mathcal{L}(\mathbf{Z}^\star; \mathbf{b}) &= \int q(\mathbf{f}, \mathbf{u}, \mathbf{u}_C) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}, \mathbf{u}_C \mid \mathbf{X}, \mathbf{Z}^\star)}{q(\mathbf{f}, \mathbf{u}, \mathbf{u}_C)} \,\mathrm{d}[\mathbf{f}, \mathbf{u}, \mathbf{u}_C] \\
&= \int q(\mathbf{f}, \mathbf{u}) p(\mathbf{u}_C) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u} \mid \mathbf{X}, \mathbf{Z}) \cancel{p(\mathbf{u}_C)}}{q(\mathbf{f}, \mathbf{u}) \cancel{p(\mathbf{u}_C)}} \,\mathrm{d}[\mathbf{f}, \mathbf{u}, \mathbf{u}_C] \\
&= \mathcal{L}(\mathbf{Z}).
\end{aligned}$$

As such, we have an equivalent expression for $\mathcal{L}(\mathbf{Z})$ that enables the application of the re-parameterisation trick.

Appendix C

Experimental setups

C.1 Complexity reduction in sparse Gaussian processes

C.1.1 Procedure for synthetic data

In the experiment for synthetic data we considered three data characteristics:

1. observation noise,
2. kernel smoothness,
3. input clustering.

Each were evaluated with different "intensity" levels to ascertain the interaction between characteristic and informativeness of the inducing points. For each combination of characteristic and intensity, we sampled 500 observations from the following, generative model:

$$\begin{aligned}\mathbf{x} &\sim p(\mathbf{x}),\\ \mathbf{f} &\sim \mathcal{GP}(\mathbf{0}, \kappa(\mathbf{x}, \mathbf{x}'; \gamma)),\\ \mathbf{y} &\sim \mathcal{N}\left(\mathbf{f}, \mathbf{I}\sigma^2\right).\end{aligned}$$

Here, κ is a RBF kernel with variance 1.0 and lengthscale γ. To generate data for characteristics 1 and 2 we set $p(\mathbf{x})$ to a uniform distribution over $[0, 100]$. The intensities were then given by γ for characteristic 1 and σ for characteristic 2. To generate data for characteristic 3 we let $p(\mathbf{x})$ be a homogeneous mixture of 5 normal distributions, $\{\mathcal{N}\left(\mathbf{u}_j, \beta^{-1}\right)\}_{j=1}^{5}$, with equidistant means distributed over the input domain, $\mathcal{X} = [0, 100]$. The intensity was then given by the shared precision, β; i.e. higher values yields more clustering. As $\beta \to 0$ the mixture tends towards a uniform distribution. The default values for the intensity parameters were $\sigma = 0.1, \gamma = 1.0$.

C.1.2 Procedure for real-world data

For the real-world data we corrupted the original outputs, $\mathbf{y}$, with additive, standard Gaussian noise scaled by a constant factor:

$$\hat{\mathbf{y}} = \mathbf{y} + \epsilon \cdot \hat{\sigma}(\mathbf{y}) \cdot v, \qquad \epsilon \sim \mathcal{N}(0, 1),$$

where $\hat{\sigma}(\mathbf{y})$ is the empirical standard deviation of the observed outputs. The constant $v \in [0, 1)$ determines the level of corruption and is the value reported in Figure 3.7. The parameter, α, that yields the sparsity level imposed by the point process prior was manually chosen for each dataset. This is because the datasets vary widely in size and more observations will tend to diminish the influence of the prior. The prior configurations and noise levels are listed in Table C.1.

Dataset	N	D	Noise levels	α
UCI Concrete	1030	8	[0.0, 0.2, 0.3, 0.4]	0.01
UCI Energy	768	8	[0.0, 0.05, 0.1, 0.15]	0.05
UCI Kin8nm	8192	8	[0.0, 0.2, 0.3, 0.4]	0.1
UCI Protein	45730	9	[0.0, 0.3, 0.5, 0.7]	0.2

Table C.1: Specifications for the informativeness experiment carried out on real-world benchmark datasets.

C.2 Estimating a target vector through Bayesian optimisation

This section lists the 14 benchmark functions used in the experimental section.

Rosenbrock (2 inputs, 1 output)

$$y = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$
$$x_1 \in (0, 5)$$
$$x_2 \in (0, 3)$$

Ackley (2 inputs, 1 output):

$$y = 20 - 20\exp\left(-0.2\sqrt{0.5(x_1^2 + x_2^2}\right) + e - \exp\left(0.5(\cos 2\pi x_1 + \cos 2\pi x_2)\right)$$
$$x_1, x_2 \in (-30, 30)$$

Bohachevsky (2 inputs, 1 output):

$$y = x_1^2 + 2x_2^2 - 0.3\cos 3\pi x_1 - 0.4\cos 4\pi x_2 + 0.7$$
$x_1, x_2 \in (-100, 100)$

Griewank (2 inputs, 1 output):

$$y = 1 + \frac{1}{4000}(x_1^2 + x_2^2) - \cos x_1 \cdot \cos \frac{x_2}{\sqrt{2}}$$
$x_1, x_2 \in (-600, 600)$

H1 (2 inputs, 1 output):

$$y = \frac{\sin(x_1 - \frac{x_2}{8})^2 + \sin(x_2 + \frac{x_1}{8})^2}{\sqrt{(x_1 - 8.6998)^2 + (x_2 - 6.7665)^2} + 1}$$
$x_1, x_2 \in (-100, 100)$

Himmelblau (2 inputs, 1 output):

$$y = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$
$x_1, x_2 \in (-6, 6)$

Rastrigin (2 inputs, 1 output):

$$y = 20 + x_1^2 - 10\cos 2\pi x_1 + x_2^2 - 10\cos 2\pi x_2$$
$x_1, x_2 \in (-5.12, 5.12)$

Schaffer (2 inputs, 1 output):

$$y = \sqrt[4]{x_1^2 + x_2^2} \cdot \left[\sin^2\left(50\sqrt[10]{x_1^2 + x_2^2}\right) + 1\right]$$
$x_1, x_2 \in (-100, 100)$

Schwefel (2 inputs, 1 output):

$$y = 837.9658 - x_1 \sin\sqrt{|x_1|} - x_2 \sin\sqrt{|x_2|}$$
$x_1, x_2 \in (-500, 500)$

BNH (2 inputs, 2 output):

$$\begin{aligned} y_1 &= 4(x_1^2 + x_2^2) \\ y_2 &= (x_1 - 5)^2 + (x_2 - 5)^2 \\ x_1 &\in (0, 5) \\ x_2 &\in (0, 3) \end{aligned}$$

SRN (2 inputs, 2 output):

$$\begin{aligned} y_1 &= 2 + (x_1 - 2)^2 + (x_2 - 2)^2 \\ y_2 &= 9x_1 - (x_2 - 1)^2 \\ x_1, x_2 &\in (-20, 20) \end{aligned}$$

OSY (6 inputs, 2 output):

$$\begin{aligned} y_1 &= -25(x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2 \\ y_2 &= \sum_{i=1}^{6} x_i^2 \\ x_1, x_2, x_6 &\in (0, 10) \\ x_3, x_5 &\in (1, 5) \\ x_4 &\in (0, 6) \end{aligned}$$

TwoBarTrussDesign (3 inputs, 2 output):

$$\begin{aligned} y_1 &= x_1\sqrt{16 + x_3^2} + x_2\sqrt{1 + x_3^2} \\ y_2 &= \max\left(\frac{20\sqrt{16 + x_3^2}}{x_1 x_3}, \frac{80\sqrt{1 + x_3^2}}{x_2 x_3}\right) \\ x_1, x_2 &\in (0, 0.001) \\ x_3 &\in (1, 3) \end{aligned}$$

WeldedBeamDesign (4 inputs, 2 output):

$$y_1 = 1.10471 \cdot x_1^2 \cdot x_3 + 0.04811 \cdot x_4 \cdot x_2 \cdot (14 + x_3)$$
$$y_2 = \frac{2.1952 \cdot x_2}{x_4^3}$$

$x_1, x_2 \in (0.125, 5)$

$x_3, x_4 \in (0.1, 10)$