Compositionality, Stability and Robustness in Probabilistic Machine Learning

by

Ayman Boustati

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Mathematics for Real-World Systems CDT

January 2021

Declarations

I declare that the entirety of the thesis is my own work except for the analytical results in Appendix B.1, Appendix C.1, Appendix C.2 and Appendix D.2, where these results have been included for completion.

Three of the four content chapters in this thesis have been part of peer-reviewed published work and the remaining one is published as a pre-print:

- Chapter 3 is based on the work in **Boustati, A.**, Damoulas, T. and Savage, R. S. (2019). Non-linear Multitask Learning with Deep Gaussian Processes. arXiv e-prints, arXiv:1905.12407 [stat.ML], with code publicly available at https://github.com/aboustati/dgplib.

- Chapter 4 is based on the work in **Boustati, A.**, Vakili, S., Hensman, J. & John, ST (2020). Amortized variance reduction for doubly stochastic objective. Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI). It also appears in **Boustati, A.**, John, S., Vakili, S., & Hensman, J. (2021). U.S. Patent Application No. 16/984,824.

- Chapter 5 is based on the work in Richter, L.*, **Boustati, A.***, Nüsken, N., Ruiz, F. J. & Akyildiz, Ö. D. (2020). VarGrad: A Low-Variance Gradient Estimator for Variational Inference. Advances in Neural Information Processing Systems (NeurIPS) 33, with code publicly available at https://github.com/aboustati/vargrad.

- Chapter 6 is based on the work in **Boustati, A.***, Akyildiz, Ö. D.*, Damoulas, T. & Johansen, A., M. (2020). Generalised Bayesian Filtering via Sequential Monte Carlo. Advances in Neural Information Processing Systems (NeurIPS) 33, with code publicly available at https://github.com/aboustati/robust-smc.

*Equal contribution.

Acknowledgments

First and foremost, I would like to thank my supervisor Dr Theo Damoulas. He adopted me mid-way through my PhD and gave me the guidance and mentorship to see it through until the end. I learned a lot from him and, needless to say, this thesis would not have been possible without his guidance and support.

I am also very grateful to my previous supervisor Dr Rich Savage who started me off on this road and set my academic foundations. I also learned a lot from him during our time working together.

During my PhD, I had the good fortune to be in two research groups! I learned a lot from both sets of groupmates and developed friendships along the way. I would like to acknowledge everyone in the Warwick Machine Learning Group (WMLG) and in Team Savage. In particular, thanks to Iliana Peneva and Jim Skinner for their friendship and for the pizza nights, and also Ale Avalos for her friendship and support.

I was lucky to be part of a great group at the Mathematics for Real-World Systems CDT and the Centre for Complexity Science. I would like to express my gratitude to all the students and staff who made my time there a very rewarding experience. Specifically, I would like to thank Professor Magnus Richardson and Professor Colm Connaughton for managing a great programme, and also Heather Robson for her support and for helping me with obtaining my scholarship to study there. Many people have played a part in making my time there very special. I am thankful to my office-mates in D2.05 and D2.17: Sami Al-Izzi, Aditi Shenvi, Joe Hilton, Roger Hill, Bhavan Chahal, Ellen Webborn, Jon Skipp and Kutwulano Bashe, for making my days at the office very exciting. I have also met some great friends on this programme. Thanks to the "Mostly not London" machine learning crew: Michael Pearce, Jev Gamper, Janis Klaise and Matt Groves for the random discussions and complaints. Also thanks to the MathSys/Complexity friends: Nadia Jankovicova, Laura Guzman Rincon, Alvaro Cabrejas Egea, Rob Eyre, Giovanni Mizzi, Tim Pollington, Chris Davis, Peter De Ford, Max Smilovitskiy, Jeremy Reizenstein, Jason Lewis and Helen Kochkina for their friendship over the PhD years.

During my PhD, I had the privilege to be part of the Alan Turing Institute. I would like to thank everyone there for making this experience very fun and rewarding. In particular, I want to thank Tim Pearce and Nicolas Anastassacos (the Logistic Depression crew) for their friendship and for the blue sky discussions on machine learning and other random topics, and my desk buddy Jessie Liu for the fun escapades. I am also thankful to Alex Bird, Merve Alanyali and Ollie Hamelijnck

for the casual technical and non-technical chats.

I was very fortunate to spend three months at Secondmind.ai in Cambridge as an intern. I would like to express my gratitude to everyone there, especially to the probabilistic modelling team. I would like to single out Dr James Hensman who brought me on board and from whom I learned a lot about probabilistic modelling and research in general. I am also very thankful to Dr ST (Ti) John for her patience with me and her mentorship during my time there and afterwards. I am also grateful to Dr Sattar Vakili for his help and contribution to my project.

The work in this thesis could not have been done without the help of my collaborators. Thanks to Deniz Akyildiz for being a great friend and collaborator, and an excellent StarCraft II partner! I would also like to thank Professor Adam Johansen for his help and insights into our collaboration. It was also a great pleasure collaborating with Lorenz Richter, Nikolas Nüsken and Francisco Ruiz.

I am very grateful to Dr Maurizio Filippone for acting as my external examiner and initiating great discussions during my viva. I would also like to thank Professor Colm Connaughton again for organising a plain sailing viva, despite the challenges of the Covid-19 pandemic.

Finally, I would not be where I am today without the endless love, patience and support of my family. They believed in me from the beginning and supported me during the good and bad times. I am eternally grateful to my father, Mamoun Boustati, my mother, Nazli Alouch and my sister, Alma Boustati. Sadly, I have lost my loving grandmother, Majida Takieddine, during the final months of my PhD. She has always been a source of comfort and inspiration to me. To her memory, I dedicate this thesis.

Abstract

Probability theory plays an integral part in the field of machine learning. Its use has been advocated by many [MacKay, 2002; Jaynes, 2003] as it allows for the quantification of uncertainty and the incorporation of prior knowledge by simply applying the rules of probability [Kolmogorov, 1950]. While probabilistic machine learning has been originally restricted to simple models, the advent of new computational technologies, such as automatic differentiation, and advances in approximate inference, such as Variational Inference [Blei et al., 2017], has made it more viable in complex settings. Despite this progress, there remain many challenges to its application to real-world tasks. Among those are questions about the ability of probabilistic models to model complex tasks and their reliability both in training and in the face of unexpected data perturbation. These three issues can be addressed by examining the three properties of *compositionality*, *stability* and *robustness* in these models. Hence, this thesis explores these three key properties and their application to probabilistic models, while validating their importance on a range of applications.

The first contribution in this thesis studies compositionality. Compositionality enables the construction of complex and expressive probabilistic models from simple components. This increases the types of phenomena that one can model and provides the modeller with a wide array of modelling options. This thesis examines this property through the lens of Gaussian processes [Rasmussen and Williams, 2006]. It proposes a generic compositional Gaussian process model to address the problem of multi-task learning in the non-linear setting.

Additionally, this thesis contributes two methods addressing the issue of stability. Stability determines the reliability of inference algorithms in the presence of noise. More stable training procedures lead to faster, more reliable inferences, especially for complex models. The two proposed methods aim at stabilising stochastic gradient estimation in Variational Inference using the method of control variates [Owen, 2013].

Finally, the last contribution of this thesis considers robustness. Robust machine learning methods are unaffected by unaccounted-for phenomena in the data. This makes such methods essential in deploying machine learning on real-world datasets. This thesis examines the problem of robust inference in sequential probabilistic models by combining the ideas of Generalised Bayesian Inference [Bissiri et al., 2016] and Sequential Monte Carlo sampling [Doucet and Johansen, 2011].

List of Figures

List of Tables

Notation and Abbreviations

Symbols

x	A scalar-valued quantity
$\mathbf{x}$	A vector-valued quantity
$\mathbf{X}$	A matrix-valued quantity
$\mathcal{X}$	A vector space
$\mathbf{x}^\intercal$	The transpose of $\mathbf{x}$
$\mathbf{x}^\star$	The optimal value of $\mathbf{x}$
$\mathbf{x}^{(s)}$	The s^{th} Monte Carlo sample
$\hat{\mathbf{x}}$	The Monte Carlo estimate of $\mathbf{x}$, $\frac{1}{S}\sum_{s=1}^{S}\mathbf{x}^{(s)}$
$\tilde{\mathbf{x}}$	The Monte Carlo estimate of $\mathbf{x}$ with a scaled zero-mean control variate $\boldsymbol{\omega}$, $\hat{\mathbf{x}} - \hat{\boldsymbol{\omega}}$
$\bar{\mathbf{x}}$	The $-$ accent is reserved for contextual use
$\check{\mathbf{x}}$	The $\vee$ accent is reserved for contextual use
$\{\mathbf{x}_n\}_{n=1}^{N}$	A collection of N objects
$\mathbf{x}_{1:N}$	A collection of N objects
$\mathbf{x}_n$	The n^{th} member of a collection
$\mathbb{R}$	Real numbers
I_D	The $D \times D$ identity matrix

Functions

$f(\cdot\,;\theta)$	Explicit parameterisation of a function f with θ
$f_\theta(\cdot)$	Implicit parameterisation of a function f with θ
$p(\cdot)$	A probability density function or probability mass function

$\delta_y(\cdot)$	The Dirac delta measure concentrated at y
$\mathbb{1}_{\mathcal{Y}}(\cdot)$	The indicator function on the set $\mathcal{Y}$
$\|\cdot\|_2$	The Euclidean (Square) norm
$\|\cdot\|_p$	The p-norm
$\mathrm{KL}\left(p \,\|\, q\right)$	The Kullback-Leibler divergence from q to p
$\mathrm{ELBO}(\boldsymbol{\phi})$	The Evidence Lower BOund as a function of $\boldsymbol{\phi}$

Operators

$:=$	Equality by definition
$\equiv$	Equality by identity
$\approx$	Approximately equal to
$\propto$	Proportional to
$\odot$	The Hadamard (element-wise) product
$\sim$	Distributed as
$\setminus$	The set difference operator
$\|$	The conditioning operator
$\|\mathcal{S}\|$	The cardinality of the set $\mathcal{S}$
$\mathbb{E}[\cdot]$	The expectation operator
$\mathrm{Var}[\cdot]$	The variance operator
$\mathrm{Cov}[\cdot\,,\,\cdot]$	The covariance operator
$\mathrm{Tr}\,\cdot$	The trace operator
∇_θ	The gradient operator with respect to θ
∂_θ	The partial derivative operator with respect to θ
$\mathcal{O}(\cdot)$	Big Oh denoting computational or space complexity

Abbreviations

ANNl	Artificial Neural Network with l layers
APF	Auxiliary Particle Filter

ARD	Automatic Relevance Determination
BFP	Bootstrap Particle Filter
BNNl	Bayesian Neural Network with l layers
DGM	Data Generating Mechanism
DGP	Deep Gaussian Process
DVAE	Discrete Variational Auto-Encoder
EC	Empirical Coverage
ELBO	Evidence Lower BOund
ESS	Effective Sample Size
FFBS	Forward Filtering Backward Smoothing
GBI	Generalised Bayesian Inference
GP	Gaussian Process
HMM	Hidden Markov Model
ICM	Intrinsic Coregionalisation Model
i.i.d.	independently and identically distributed
IS	Importance Sampling
KL	Kullback-Leibler
LGSSM	Linear-Gaussian State-Space Model
MC	Monte Carlo
NELBO	Negative Evidence Lower Bound
NLPP	Negative Log Predictive Probability
NMSE	Normalised Mean-Squared Error
PF	Particle Filter
PSD	Positive Semi-Definite
RMSE	Root Mean-Squared Error
ROC-AUC	Area Under Receiver Operating Characteristic Curve
RTS	Rauch–Tung–Striebel smoother
SGD	Stochastic Gradient Descent
SIS	Sequential Importance Sampling
SMC	Sequential Monte Carlo

SVI	Stochastic Variational Inference
TAN	Terrain Aided Navigation
VAE	Variational Auto-Encoder
VI	Variational Inference
cGP	coregionalised Gaussian Process
cMDGP	coregionalised Multi-task DGP
iDGP	independent Deep Gaussian Process
iGP	independent Gaussian Process
mMDGP	multiprocess Multi-task Deep Gaussian Process
sMDGP	shared Multi-task Deep Gaussian Process
w.r.t.	with respect to

Contents

CHAPTER 1

Introduction

> *"Given for one instant an intelligence which could comprehend all the forces by which nature is animated and the respective situation of the beings who compose it –an intelligence sufficiently vast to submit these data to analysis– it would embrace in the same formula the movements of the greatest bodies of the universe and those of the lightest atom; for it, nothing would be uncertain and the future, as the past, would be present to its eyes."*
>
> — Laplace

> *"The curve described by a simple molecule of air or vapour is regulated in a manner just as certain as the planetary orbits; the only difference between them is that which comes from our ignorance."*
>
> — Laplace, two paragraphs later

In 1814, Pierre-Simon de Laplace articulated the notion of scientific determinism in what is commonly referred to as Laplace's Demon [Laplace, 1814]. This notion states that if an intelligent entity (the demon) knows the exact state of the universe and all the laws that govern it, then the intelligence can compute any past or future state provided it possesses the computational capacity to do so. Laplace argues that while humans thrive to be this intelligence, they "will always remain infinitely removed" from it.

All is not lost, however; Laplace advocates using the rules of probability [Kolmogorov, 1950] to reason about uncertain events and to propagate this uncertainty to consequent events.

> *"Probability is relative, in part to this ignorance, in part to our knowledge. We know that of three or a greater number of events a single one ought to occur; but nothing induces us to believe that one of them will*

> *occur rather than the others. In this state of indecision it is impossible for us to announce their occurrence with certainty. It is, however, probable that one of these events, chosen at will, will not occur because we see several cases equally possible which exclude its occurrence, while only a single one favours it."*
>
> — Laplace

One sticking point to this thought is the complexity of real-world phenomena. While it is both easy and desirable to model simple events using the language of probability, complex phenomena might prove elusive to such modelling due to the need to express intricate assumptions as probabilistic statements and the large computational burden associated with this modelling complexity.

This thesis examines the probabilistic view of modelling complex phenomena through the lens of *machine learning*. Machine learning incorporates tools that allow the modeller to abstract away some of the complexity in the target phenomena by specifying generic yet flexible models that learn the intricacies of the target systems through observed data. The probabilistic treatment of such methods can provide the quantification and propagation of uncertainty advocated by Laplace.

In particular, this thesis studies three desirable properties for probabilistic machine learning models: *compositionality*, *stability* and *robustness*. These three properties endow machine learning systems with the ability to deal with the various challenges in modelling real-world events through observed data.

1.1 The Three Key Themes

This thesis studies various models and computational methods in the field of probabilistic machine learning revolving around the three themes of compositionality, stability and robustness. An overview of these properties is given below.

1.1.1 Compositionality

Compositionality refers to the ability of simple models to compose with each other to create a more flexible model. Complex systems are characterised as a combination of smaller simpler systems. To model a complex system, one can create an expressive model by combining smaller simpler models that target the components of the complex system individually. Hence, compositional probabilistic models can model complex phenomena by combining probabilistic modules such as Gaussian process modules or probabilistic neural network layers [Tran et al., 2019]. The re-

sulting model has a higher learning capacity than its components while maintaining tractable inference procedures. The idea of compositionality is best embodied by probabilistic programming languages, which define modelling and inference primitives that allow various components to interact with each other [Tran et al., 2016].

1.1.2 Stability

In learning theory, the notion of stability relates to the sensitivity of a machine learning procedure to perturbations in the input data [Bousquet and Elisseeff, 2002]. This idea is inherently related to the variance of the algorithm, where high-variance algorithms have low stability and low-variance algorithms are said to be stable. Another related concept is that of *training stability* referring to the sensitivity of the algorithm to the noise in the training procedure [Anschel et al., 2017; Nikishin et al., 2018]. This is a common issue in modern machine learning methods that use stochastic training procedures (*e.g.*, stochastic optimisation or sampling) to tune the parameters of the model. The notion of training stability is also related to variance; more specifically, the variance of the objective function (and potentially its gradients).

While the two views of stability are conceptually related, this thesis studies the training stability view. This view is more relevant to the probabilistic setting as the input data in probabilistic models is considered to be fixed and generated from a known generative process, whereas the training procedure can depend on random quantities that induce instability.

1.1.3 Robustness

Colloquially, robustness is the resilience property in objects or systems against unexpected shocks. Similarly, in statistics and machine learning, *robustness* refers to the ability of models to withstand small deviations from their assumptions [Huber, 1981]. In machine learning applications - particularly in probabilistic machine learning - explicit assumptions are made to model a dataset. These assumptions are usually simplistic and might not model all aspects of the data. There are many reasons why the deviation from assumptions can occur, *e.g.*, a) some instances in the dataset are corrupted and the model does not account for the corruption, b) the model is misspecified, *i.e.*, its assumptions are significantly different from the true data generating mechanism, or c) an adversary is manipulating the data that the model sees. Robust machine learning models can learn reliable hypotheses in these scenarios without explicitly modelling the source of deviation.

1.2 Motivation

Following the arguments of Laplace, modelling real-world phenomena requires the ability to assign probabilities to observed events and reason about the unknown by manipulating them with the rules of probability [Kolmogorov, 1950]. Naturally, these two tasks become more difficult as the complexity of the modelled phenomena increases. For instance, assigning probabilities to multiple observed events and relating them to unobserved variables becomes more challenging as the number of variables increases. Furthermore, a large number of variables with complex relationships between them causes computational difficulties when applying the rules of probability. Hence, to address complex real-world problems, probabilistic machine learning methods need to be both flexible and reliable. The choice of the three themes of compositionality, stability and robustness is motivated by these desiderata. Compositionality provides the needed flexibility by giving the modeller tools to build tailored models from simple building blocks. Stability and robustness ensure the reliability of these models both in training and in deployment.

It is important to note that there is some inherent trade-off between the considered properties. For instance, while the composition of multiple components increases a model's flexibility, it can also cause instability in training as the model's variance becomes higher than that of its components. Similarly, the added flexibility of compositional models can make them brittle against unexpected perturbations in the data. Thus, these properties cannot be studied in isolation as they influence one another. That is why this thesis attempts to study them holistically.

Modern machine learning methods incorporate many aspects of these properties. For example, the success of Deep Learning [LeCun et al., 2015] is a testament to the importance of compositionality in modelling. Furthermore, many algorithms have been designed to induce training stability, *e.g.*, Adam [Kingma and Ba, 2015], SVRG [Johnson and Zhang, 2013], *etc.* Robustness, on the other hand, has always been prevalent in statistics and machine learning going back to as far as the 1960s [Tukey, 1962; Huber, 1981]. However, in the probabilistic treatment of machine learning, these topics have been less explicitly emphasised and have also been under-explored. This thesis attempts to fill this gap by explicitly examining them in the context of probabilistic models and exploring ideas that make them applicable to real-world settings.

1.3 Contributions and Thesis Structure

This thesis starts with a general overview of the fundamentals of probabilistic machine learning in **Chapter 2**, focusing on topics in modelling, inference and variance reduction. Subsequently, it presents four main contributions in chapter 3–6 that revolve around the three key themes.

Chapter 3 studies the problem of compositionality in Gaussian process models. It presents a multi-task discriminative model defined through the non-linear composition of Gaussian process modules. The resulting formulation is a multi-task Deep Gaussian process with a latent space that is composed of private processes that capture within-task information and shared processes that capture across-task dependencies. Inference in this model is made possible due to its compositional nature, where standard inference procedures can be easily extended to the multi-task setting. Two different methods for segmenting the latent space are proposed: 1) through hard coding shared and task-specific processes, or 2) through soft sharing with Automatic Relevance Determination kernels. This formulation is able to improve the learning performance and transfer information between the tasks, outperforming other probabilistic multi-task learning models across real-world and benchmarking settings.

Chapter 4 & **Chapter 5** address the problem of stability, specifically in the context of Variational Inference. **Chapter 4** introduces a control variate scheme for reducing the variance of doubly stochastic objective functions. These types of objective functions appear as optimisation targets in Variational Inference for complex probabilistic models. They incorporate randomness both from mini-batch sub-sampling of the data and from Monte Carlo estimation of expectations inducing variance in their gradients. If the gradient variance is high, the optimisation problem becomes difficult with a slow rate of convergence. Control variates can be used to reduce the variance, but past approaches do not take into account how mini-batch stochasticity affects sampling stochasticity, resulting in sub-optimal variance reduction. This chapter proposes a new approach in which a recognition network is used to cheaply approximate the optimal control variate for each mini-batch, with no additional model gradient computations. The properties of this proposal are illustrated and its performance is tested on logistic regression and deep Gaussian processes.

Chapter 5 analyses the properties of an unbiased gradient estimator of the Evidence Lower Bound (ELBO) for Variational Inference, based on the score function method with leave-one-out control variates in Salimans and Knowles [2014]

and Kool et al. [2019]. The chapter shows that this gradient estimator can be obtained using a novel loss, defined as the variance of the log-ratio between the exact posterior and the variational approximation, which is called the log-variance loss. Under certain conditions, the gradient of the log-variance loss equals the gradient of the (negative) ELBO. It is shown that this gradient estimator, *VarGrad*, exhibits lower variance than the score function method in certain settings. Furthermore, the control variate coefficients that are induced by this estimator are proven to be close to the optimal ones. The utility of VarGrad is empirically demonstrated showing that it offers a favourable variance versus computation trade-off compared to other state-of-the-art estimators on a discrete Variational Auto-Encoder.

Chapter 6 is the final technical chapter and examines the problem of robustness in sequential models. It introduces a framework for robust inference in general state-space hidden Markov models under likelihood misspecification, leveraging the loss-theoretic perspective of Generalized Bayesian Inference to define generalised filtering recursions in hidden Markov models. The new framework can tackle the problem of inference under model misspecification, arriving at principled procedures for robust inference against observation contamination by utilising the β-divergence. Operationalising the proposed framework is made possible via sequential Monte Carlo methods, where most standard particle methods, and their associated convergence results, are readily adapted to the new setting. The new approach is applied to object tracking and Gaussian process regression problems, where improved performance over both standard filtering algorithms and other robust filters is observed.

Finally, **Chapter 7** summarises the contributions in this thesis, offering a general discussion of their placement within the wider field and outlooks for future research.

1.4 Publications

Some of the contents of this thesis appear in three published works and a pre-print. They are listed below in the chronological order of the thesis.

- Chapter 3 is based in the work in Boustati, Damoulas, and Savage [2019]. The code for this work is publicly available at `https://github.com/aboustati/dgplib`.
- Chapter 4 is based on the work in Boustati, Vakili, Hensman, and John [2020b]. It also appears in the following patent: Boustati, John, Vakili, and Hensman [2021].

- Chapter 5 is based on the work in Richter, Boustati, Nüsken, Ruiz, and Akyildiz [2020]. The code for this work is publicly available at `https://github.com/aboustati/vargrad`.
- Chapter 6 is based on the work in Boustati, Akyildiz, Damoulas, and Johansen [2020a]. The code for this work is publicly available at `https://github.com/aboustati/robust-smc`.

CHAPTER 2

The Fundamentals of Probabilistic Machine Learning

Traditionally, the field of machine learning has focused on inventing algorithms to solve learning tasks. The algorithm is all-encompassing and includes all aspects of the modelling such as incorporating the data, learning from the data (training) and prediction. This algorithmic view has been dominant in the study of machine learning until this day; however, one of the drawbacks of this type of thinking is its rigidity. Algorithms are designed to solve specific tasks and they implicitly incorporate assumptions and inductive biases. Hence, to address new tasks, one needs to come up with new algorithms since an algorithm's assumptions do not necessarily hold for all tasks. Probabilistic machine learning offers an alternative view: one which separates the assumptions from the learning mechanism and makes these assumptions explicit.

Probabilistic machine learning is model-based, *i.e.*, its focal point is a model that explicitly incorporates all the assumptions and the knowledge that the modeller has about the problem or task. More concretely, a *probabilistic model* describes an idealised mechanism that might have generated the data. This is done by introducing a set of random variables that interact in some way to generate the data. Probabilistic statements are added to determine the relationship between the random variables. The random variables that constitute the model can either be *observable* or *latent*. After seeing the observable variables, one can draw conclusions on the plausible values of the latent (unknown) variables[1]. This procedure is called

[1]In some literature, the term "latent variables" is reserved to a special type of unknown quantities that have one-to-one association with the observables, *i.e.*, each observable is associated with a separate unknown quantity. Other types of unknown quantities are usually referred to as unobserved or unknown variables (sometimes also model parameters). This thesis does not adopt this terminology. Instead, the term "latent variables" is used to denote any unknown quantity. The specific case of models containing one-to-one association is referred to as a *latent variable model*. The terminology of this thesis is adopted from Bishop [2006].

Inference.

One of the key distinctions between the algorithmic view of machine learning and the probabilistic modelling view is the separation of modelling and inference. In probabilistic machine learning, the model simply describes a mechanism that can generate data without specifying how to fit it to what has been observed. In the inference stage, the model is fit to the data using a specific inferential mechanism. The separation of these two concepts provides the modeller with flexibility since inference procedures are usually model agnostic and vice versa. This allows the modeller to mix-and-match models with inference procedures according to the problem specification.

Furthermore, the probabilistic machine learning framework allows for the explicit specification of the modeller's prior beliefs, through the choice of a prior belief distribution over the latent variables. These beliefs are chosen in addition to the structural priors that are usually present in algorithmic machine learning models, *e.g.*, Convolutional Neural Networks [LeCun et al., 1989]. The explicit specification of prior beliefs allows the modeller to update their beliefs after observing data, thus quantifying the uncertainty in the latent variables in relation to the specified prior.

This chapter introduces the fundamentals of probabilistic machine learning. It mainly covers an overview of popular families of probabilistic models in Section 2.1 and popular inference mechanisms in Section 2.2. Section 2.7, introduces some ideas on variance reduction that are essential for Chapter 4 and Chapter 5.

2.1 Probabilistic Models

In its most basic form, a probabilistic model is composed of two components: observable variables and latent variables. Observable variables constitute the data, $\mathbf{y} \in \mathcal{Y}$. These are the variables that can be observed reliably, *e.g.*, the body-mass index of a group of sixth-form students, the symptoms observed in participants of a clinical trial, readings from a Geiger counter, *etc.* Reliable observation means that one can measure the observable quantity; however, it does not relate to the quality of the measurements, *i.e.*, noisy measurements are still valid reliable observations.

The latent variables $\mathbf{x} \in \mathcal{X}$ are the unobserved quantities in the model. The term latent means that these quantities are computed through the observed data, *i.e.*, they come after the observables are considered. Examples of these include the athletic ability of the sixth-formers, whether the clinical trial participants have a certain disease, the underlying level of background radiation in the area where the Geiger counter is used, *etc.*

It is important to note that the relationship between the observable variables and the latent variables need not necessarily be causal. A mere correlation with the observable variables is enough to allow the user to make inferences about the latent variables.

In mathematical terms, one represents a model by a joint probability distribution on the shared space of latents and observables,

$$\mathcal{M} \equiv p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}), \tag{2.1}$$

where $\boldsymbol{\theta} \in \Theta$ is a set of parameters that index the set of models. These types of parameters are known as *hyper-parameters* or *nuisance parameters*. While hyper-parameters are latent in the sense that they are unobserved, one makes the distinction between them and the latent variables (known sometimes as model parameters in this context) since one does not wish to make inferences about them.

The probabilistic model representation in (2.1) can be factorised into two terms,

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) = \underbrace{p_{\boldsymbol{\theta}}(\mathbf{y} \,|\, \mathbf{x})}_{\text{Likelihood}} \underbrace{p_{\boldsymbol{\theta}}(\mathbf{x})}_{\text{Prior}}. \tag{2.2}$$

In (2.2), $p_{\boldsymbol{\theta}}(\mathbf{y} \,|\, \mathbf{x})$ is known as the *likelihood* and $p_{\boldsymbol{\theta}}(\mathbf{x})$ is known as the *prior*. The likelihood is a normalised probability distribution with respect to the data $\mathbf{y}$ conditioned on the latent variables $\mathbf{x}$. It can also be viewed as a function of the latents that indicates how likely is it to observe the given data for a specific value of $\mathbf{x}$. The prior is a normalised[2] probability distribution with respect to the latents, that encodes the modeller's prior beliefs about them. These beliefs can be either objective or subjective and there is a large body of literature advocating for each of these views [Efron and Hastie, 2016, Chapter 13]; however, this thesis does not make a distinction.

Finally, the problem of inference (discussed in detail in Section 2.2) is to simply compute the *posterior* probability distribution, *i.e.*, the probability of the latents given the data, $p_{\boldsymbol{\theta}}(\mathbf{x} \,|\, \mathbf{y})$.

The remainder of this section discusses some of the most popular families of probabilistic models and the tools used to construct them.

[2]This is not strictly necessary as unnormalised (improper) priors can also be defined; however, this thesis only considers normalised (proper) priors.

2.1.1 Parametric Models

Parametric models are a family of probabilistic models where the latent variables have a finite dimension. The model parameters (latent variable) are sufficient, in the sense that they summarise a given dataset and any future predictions become independent of the data given the parameters. This restriction on the dimensionality of the latent variables bounds the model complexity at the cost of its flexibility. There are many examples of parametric models in the machine learning literature, such as linear models, cubic splines [Hastie et al., 2009], neural networks [LeCun et al., 2015], mixture models with finite mixture components [Bishop, 2006], *etc.*

2.1.1.1 Linear Models

Linear models specify a linear relationship between the observable variables and the latent variables. This relationship gives rise to a very large class of models that address many machine learning tasks such as regression, classification, dimensionality reduction, *etc.*

Consider a set of three random variables $\mathbf{X} \in \mathcal{X}^N, \mathbf{Y} \in \mathcal{Y}^N$ and $\mathbf{W} \in \mathcal{W}$. These variables can be thought of as matrix valued without loss of generality, where $\mathbf{X}$ and $\mathbf{Y}$ have a design matrix structure, *i.e.*, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\intercal$ with $\mathbf{x}_n \in \mathcal{X}$ and $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)^\intercal$ with $\mathbf{y}_n \in \mathcal{Y}$. Now consider the linear relationship

$$\boldsymbol{\psi} = \mathbf{W}\mathbf{X}^\intercal. \tag{2.3}$$

A linear model for $\mathbf{Y}$ is specified by parameterising its distribution with $\boldsymbol{\psi}$,

$$\mathbf{Y} \sim p(\mathbf{Y}\,;\,\boldsymbol{\psi}), \tag{2.4}$$

where the semi-colon is used to denote the parameterisation explicitly. Using the construction in (2.4), one can arrive at many well-known machine learning models. For instance, consider the case of linear latent variable models [Bishop, 2006]. One can arrive at this class of models by identifying $\mathbf{X}$ as a latent variable and placing a prior $p(\mathbf{X})$ on it. Next, $\mathbf{W}$ can be considered as a nuisance parameter, giving the following generative model

$$\text{Prior:} \quad \mathbf{X} \sim p(\mathbf{X}), \tag{2.5}$$

$$\text{Likelihood:} \quad \mathbf{Y} \,|\, \mathbf{X} \sim p(\mathbf{Y}\,;\,\boldsymbol{\psi}). \tag{2.6}$$

The model in (2.5) & (2.6) specifies the general construction for linear latent

variable models. Depending on the specification of the prior and the likelihood, one can arrive at a specific model, *e.g.*, specifying a Gaussian prior and an isotropic Gaussian likelihood, recovers Probabilistic Principle Component Analysis [Tipping and Bishop, 1999].

Another possibility is to consider $\mathbf{W}$ in (2.3) as latent, and $\mathbf{X}$ as fixed and indexing $\mathbf{Y}$[3]. This leads to the well-known class of generalised linear models,

$$\text{Prior:} \quad \mathbf{W} \sim p(\mathbf{W}), \tag{2.7}$$

$$\text{Likelihood:} \quad \mathbf{Y} \,|\, \mathbf{W} \sim p(\mathbf{Y}\,;\, \boldsymbol{\psi}). \tag{2.8}$$

Specifying the prior in (2.7) as Gaussian and the likelihood in (2.8) also as Gaussian, recovers linear regression, while changing the likelihood to a Bernoulli with a logistic link-function recovers logistic regression.

Finally, one can consider both $\mathbf{X}$ and $\mathbf{W}$ as latent, placing priors on both

$$\text{Prior:} \quad \mathbf{W} \sim p(\mathbf{W}), \tag{2.9}$$

$$\text{Prior:} \quad \mathbf{X} \sim p(\mathbf{X}), \tag{2.10}$$

$$\text{Likelihood:} \quad \mathbf{Y} \,|\, \mathbf{W}, \mathbf{X} \sim p(\mathbf{Y}\,;\, \boldsymbol{\psi}). \tag{2.11}$$

The generative model in (2.9)–(2.11), leads to a family of matrix factorisation models, *e.g.*, probabilistic matrix factorisation [Mnih and Salakhutdinov, 2008].

2.1.1.2 Non-linear Models

The linear relationship in (2.3), can be easily extended to the non-linear case, in which, $\mathbf{W}$ and $\mathbf{X}$ are composed non-linearly,

$$\boldsymbol{\nu} = \Phi(\mathbf{W}, \mathbf{X}), \tag{2.12}$$

where Φ is a non-linear map. Typically, $\mathbf{W}$ are considered as the parameters of this map and $\mathbf{X}$ as its inputs, so one can write

$$\boldsymbol{\nu} = \Phi_{\mathbf{W}}(\mathbf{X}). \tag{2.13}$$

There are many choices for the functional form for $\Phi_{\mathbf{W}}(\cdot)$; however, by far the most popular choice is a neural network, *i.e.*, compositions of affine transformations with non-linear activation functions.

[3] In the context, indexing means a surjective correspondence between the elements of $\mathbf{X}$ and the elements of $\mathbf{Y}$, *i.e.*, $\mathbf{x} \in \mathcal{Y}^{\mathcal{X}}$.

As in the linear case, one can recover various models by fixing one of $\mathbf{X}$ or $\mathbf{W}$ and placing a prior on the other. For instance, if $\mathbf{X}$ is considered latent and $\mathbf{W}$ is fixed, one recovers auto-associative (auto-encoding) neural networks [Bishop, 2006]; whereas fixing $\mathbf{X}$ and considering $\mathbf{W}$ as latent, one recovers standard supervised neural networks.

2.1.1.3 Graphical and Sequential Models

In many cases, the latent variables in a probabilistic model can be partitioned into groups, where the groups have different probabilistic relationships with one another and with the observables. This type of model can be represented as a graph, where each node represents a unique group and the edges indicate probabilistic dependence. In the two most popular formulations, the edges can either be directed or undirected, corresponding to either a Bayesian Network (Bayes Net) and a Markov Random Field respectively.

In a Bayesian Network, the directed edges represent conditioning where the child nodes are conditioned on their parents. Conditional independence statements can be read by applying the d-separation criterion. Consequently, the joint model can be factorised according to the conditioning statements, where each variable only depends on its parents. In many cases, this allows for efficient inference.

Sequential models are a sub-family of graphical models with a chain structure, *i.e.*, the latent variables are sequentially connected to one another. General state-space (first-order) Hidden Markov Models (HMMs) are some of the most common sequential models, where the chain connecting the latent variables is formed by connecting each node to the next node via a directed link. The observables are connected to the latent variables by directed links as well; however, they are not connected to each other. The graphical depiction of a general state-space HMM is given in Fig. 2.1, and the joint distribution is factorised as

$$p(\{\mathbf{y}_t\}_{t=1}^T, \{\mathbf{x}_t\}_{t=0}^T) = f_0(\mathbf{x}_0) \prod_{t=1}^{T} \left(g_t(\mathbf{y}_t \,|\, \mathbf{x}_t) f_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})\right), \tag{2.14}$$

where the $\{\mathbf{y}_t\}_{t=1}^T$ is the collection of observed variables from time $t = 1$ to $t = T$ and $\{\mathbf{x}_t\}_{t=0}^T$ is the corresponding collection of latent variables. The factor $g_t(\mathbf{y}_t \,|\, \mathbf{x}_t)$ is known as the emission or observation density and represents the likelihood for observation t, and $f_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$ is the transition density or the Markov kernel, representing the prior on the latent $\mathbf{x}_t$.

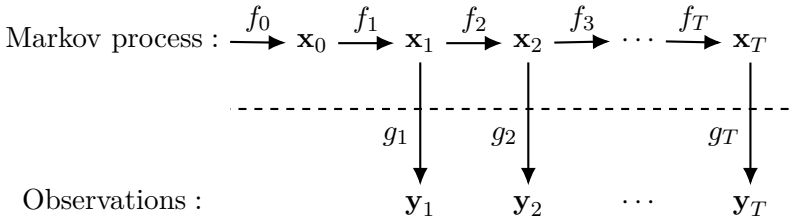


Figure 2.1: A graphical depiction of the general state-space HMM, where the latent variables are connected together in a directed chain and the observables are only connected to the corresponding latent variables.

2.1.2 Non-Parametric Models

In contrast to parametric models, non-parametric models do not assume finite-dimensional latent variables. In most cases, the latent variables are characterised as infinite-dimensional stochastic processes whose finite representation is realised upon observing the data. Non-parametric models are more flexible than their parametric counterparts since the complexity of the model scales with the complexity of the data. This property is appealing in terms of modelling quality, where one can model complex relationships in the data without being constrained by the structure of the parameters; however, this comes at the cost of computational efficiency since one needs to necessarily memorise all of the data in order to make inferences and predictions [Ghahramani, 2013].

This section explores a class of non-parametric models known as Gaussian processes. This is a flexible model family that can be used in many machine learning tasks.

2.1.2.1 Gaussian Processes

Gaussian processes (GPs) are an infinite collection of real-valued random variables any finite number of which are jointly Gaussian [Rasmussen and Williams, 2006]. As the name suggests, a GP, $\{f(\xi)\}_{\xi\in\Xi}$, is a real-valued stochastic process defined over the index set $\Xi \subseteq \mathbb{R}^D$. It is fully specified by its *mean function*, $m : \Xi \mapsto \mathbb{R}$, and its *covariance function* (or *kernel*), $k : \Xi \times \Xi \mapsto \mathbb{R}$. This means that for a finite collection of indices $\{\xi_1, \ldots, \xi_N\}$, one can compute the Gaussian marginals by evaluating these functions at the corresponding indices, *i.e.*, $\mathbf{f} = (f(\xi_1), \ldots, f(\xi_N)) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where,

$$\begin{aligned}\boldsymbol{\mu}_i &= m(\xi_i) = \mathbb{E}[f(\xi_i)],\\ \boldsymbol{\Sigma}_{ij} &= k(\xi_i, \xi_j) = \mathrm{Cov}[f(\xi_i), f(\xi_j)].\end{aligned}$$

While there are no restrictions on the form of the mean function, the covariance function needs to obey two properties: symmetry and positive semi-definiteness. Symmetry simply states that $k(\xi, \xi') = k(\xi', \xi)$ for all $\xi, \xi' \in \Xi$. Positive semi-definiteness is a more complex notion. A kernel is said to be positive semi-definite if it gives rise to a positive-semi definite (PSD) Gram matrix [Kanagawa et al., 2018]. A Gram matrix is a matrix arising from the pairwise evaluation of the kernel at a finite collection of indices, *i.e.*, $\mathbf{G}_{ij} = k(\xi_i, \xi_j)$. An $N \times N$ matrix, $\mathbf{G}$, is said to be PSD if its quadratic form is greater than or equal to zero, *i.e.*,

$$\mathbf{v}^\intercal \mathbf{G} \mathbf{v} \geq 0,$$

for all vectors $\mathbf{v} \in \mathbb{R}^N$.

For brevity, a Gaussian process can also be denoted as

$$f(\xi) \sim \mathcal{GP}(m(\cdot),\, k(\cdot, \cdot)),$$

which is the adopted notation in this thesis.

An important property of GPs is their consistency [4]. This simply means that if a GP is specified on two sets of indices $\boldsymbol{\xi}_1$ and $\boldsymbol{\xi}_2$, such that

$$\begin{bmatrix} f(\boldsymbol{\xi}_1) \\ f(\boldsymbol{\xi}_2) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \right),$$

then it satisfies that $f(\boldsymbol{\xi}_1) \sim \mathcal{N}(\boldsymbol{\mu}_1,\, \boldsymbol{\Sigma}_{11})$. An implication of the consistency property is that one can easily condition on a finite subset of the GP. By the Gaussian conditional identity,

$$f(\boldsymbol{\xi}_2) \,|\, f(\boldsymbol{\xi}_1) \sim \mathcal{N}(\boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\left(f(\boldsymbol{\xi}_1) - \boldsymbol{\mu}_1\right),\, \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}). \tag{2.15}$$

The conditional implication is very important as it enables inference when using GPs as components in probabilistic models (see Section 2.4). Finally, by Kolmogorov's extension theorem [Matthews, 2017], one can use (2.15) to define the conditional Gaussian process. Let $f(\boldsymbol{\xi})$ be the conditioning set corresponding to elements $\boldsymbol{\xi}$ from the index set, one can define a GP conditioned on this set as

[4] This is a direct consequence of Kolmogorov's extension theorem applied to Gaussian processes [Matthews, 2017]

$\mathcal{GP}(m_{\boldsymbol{\xi}}(\cdot), k_{\boldsymbol{\xi}}(\cdot,\cdot))$, where

$$m_{\boldsymbol{\xi}}(\cdot) = m(\cdot) + k(\cdot, \boldsymbol{\xi}) k(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1} \left(f(\boldsymbol{\xi}) - m(\boldsymbol{\xi}) \right), \tag{2.16}$$

$$k_{\boldsymbol{\xi}}(\cdot,\cdot) = k(\cdot,\cdot) - k(\cdot, \boldsymbol{\xi}) k(\boldsymbol{\xi}, \boldsymbol{\xi})^{-1} k(\boldsymbol{\xi}, \cdot). \tag{2.17}$$

Intuition and Examples: One can think of Gaussian processes in two ways: a) a stochastic process indexed by some set, b) a probability distribution over functions. The second view is more relevant to machine learning applications, where one can model task outputs as an unknown (latent) function of some inputs and place a GP prior over this latent mapping.

The properties of a function mapping modelled as a GP depend on the choice of the mean function and the covariance function. In a sense, these two objects encode the behaviour and the likelihood of an array of candidate functions that the modelled mapping can take. The mean function is usually responsible for capturing trends and the covariance function captures global qualities such as smoothness, stationarity, *etc.* In many machine learning applications, the mean function is usually set to a constant zero function and the entirety of the prior specification is entrusted to the covariance function. Thus, the choice of covariance function is essential to the fit quality of a model utilising GP priors.

There are many choices of covariance functions encoding different behaviours. The most popular choice is the Squared Exponential covariance function [Rasmussen and Williams, 2006] given as

$$k_{\text{SE}}(\xi, \xi') = \sigma^2 \exp\left(\frac{\|\xi - \xi'\|_2^2}{2\ell^2} \right), \tag{2.18}$$

where σ^2 and ℓ are the kernel hyperparameters, known as the signal variance and the lengthscale respectively. The signal variance controls the amplitude of the functions and the lengthscale controls the volatility of the functions. Functions modelled with a Squared Exponential GP are infinitely differentiable.

Another popular class of covariance functions is the Matérn class [Rasmussen and Williams, 2006] given as

$$k_{\text{Matérn}}(\xi, \xi') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\xi - \xi'\|_2}{\ell} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\|\xi - \xi'\|_2}{\ell} \right), \tag{2.19}$$

where σ^2 is the signal variance, ℓ is the lengthscale, ν is a positive parameter, $\Gamma(\cdot)$ is the gamma function and $K_\nu(\cdot)$ is a modified Bessel function. The parameter ν is usually set to half integers with the most popular values being 5/2, 3/2 and

1/2. This parameter determines the roughness of the functions, with the number of continuous derivatives given as $\lfloor \nu \rfloor$.

The Squared Exponential and the Matérn kernels are examples of stationary covariance functions. Hence, one can encode non-stationary behaviour in the candidate function by specifying an alternative non-stationary covariance function. An example of such covariance functions is the ArcCosine [Cho and Saul, 2009] covariance function (sometimes also known as the neural network covariance function [Williams, 1997]), given as

$$k_{\text{ArcCosine}}(\xi, \xi') = \frac{1}{\pi} \|\xi\|_2^n \|\xi'\|_2^n J_n(\theta), \tag{2.20}$$

where $\theta = \arccos \frac{\xi \cdot \xi'}{\|\xi\|_2 \|\xi'\|_2}$ is the angle between the kernel's input and $J_n(\theta) = (-1)^n \sin^{2n+1}(\theta) \left(\frac{1}{\sin(\theta)} \frac{\partial}{\partial \theta}\right)^n \left(\frac{\pi - \theta}{\sin(\theta)}\right)$ is a family of functions the captures the angular dependence. The family of ArcCosine covariance functions defines non-stationary processes that mimic the behaviour of neural networks.

Fig. 2.2 shows 5 samples drawn from GPs with a zero mean function and three different kernels: the Squared Exponential, the Matérn-3/2 and ArcCosine with $n = 1$. The difference between the three kernels can be seen from the figure. The Squared Exponential covariance function produces smooth samples, in contrast to the rough samples produces by the Matérn-3/2 kernel. Both the Squared Exponential and the Matérn-3/2 GP draws exhibit stationarity, while the ArcCosine GP draws are non-stationary. Finally, Fig. 2.3 shows the effect of conditioning on a set of 5 points. The properties of the kernels are retained upon conditioning; however, all draws from the conditional GP pass through the elements of the conditioning set, restricting the space of possible samples.

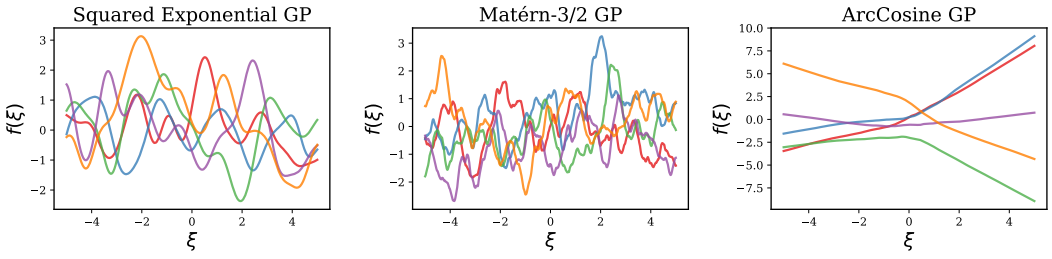


Figure 2.2: Samples from a GP for 3 kernels, the Squared Exponential, the Matérn-3/2 and ArcCosine with $n = 1$.

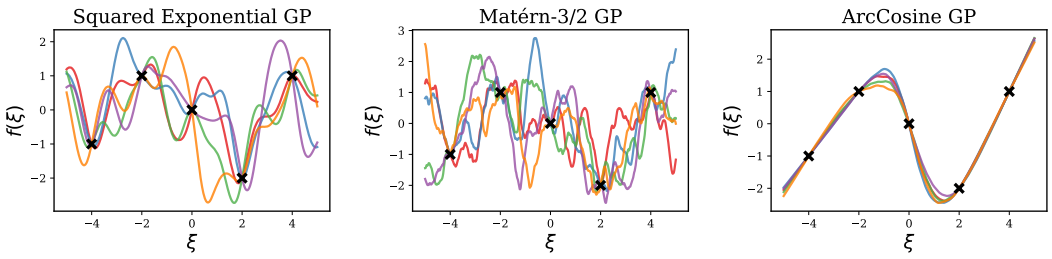


Figure 2.3: Samples from a conditional GP for 3 kernels, the Squared Exponential, the Matérn-3/2 and ArcCosine with $n = 1$. Elements of the conditioning set are marked with the black crosses.

2.1.2.2 Modelling with Gaussian Processes

GPs can be used in probabilistic modelling as priors on some of the model components. As mentioned earlier, GPs can be interpreted as distributions over functions; hence, they are most widely used as priors over functions, mapping the elements of their index set to random variables. This is very common in discriminative modelling, where one wishes to model observed outcomes based on some context (inputs). In this setting, the inputs are regarded as elements of the index set and the function, mapping the inputs to the outcomes, is modelled as a GP.

Consider the non-linear setting in (2.13) in Section 2.1.1.2. Instead of parameterising $\Phi(\cdot)$ by $\mathbf{W}$, one can specify it non-parameterically by placing a GP prior on the mapping from $\mathbf{X}$ to $\boldsymbol{\nu}$ and considering $\mathbf{X}$ as the index, *i.e.*,

$$\boldsymbol{\nu} = \Phi(\mathbf{X}), \qquad \Phi(\mathbf{X}) \sim \mathcal{GP}(m(\cdot),\, k(\cdot,\cdot)). \tag{2.21}$$

Example: Gaussian process regression Let $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_N)^\intercal \in \mathbb{R}^{N \times D_{\text{out}}}$ be the matrix representing D_{out}-dimensional i.i.d Gaussian observations corresponding to the $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\intercal \in \mathbb{R}^{N \times D_{\text{in}}}$ inputs that are organised in a design matrix. One can model the relationship between $\mathbf{X}$ and $\mathbf{Y}$ as

$$\mathbf{Y} = f(\mathbf{X}) + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0,\, \sigma^2 \mathrm{I}_N). \tag{2.22}$$

Assuming a GP prior on the mapping f, *i.e.*, $f(\cdot) \sim \mathcal{GP}(m(\cdot),\, k(\cdot,\cdot))$, one can then write its realisation at $\mathbf{X}$ as a multivariate Gaussian, $\mathbf{F} := f(\mathbf{X}) \sim \mathcal{N}(m(\mathbf{X}),\, k(\mathbf{X}, \mathbf{X}))$ and the joint model can be written as

$$p(\mathbf{Y}, \mathbf{F}) = p(\mathbf{Y} \,|\, \mathbf{F}) p(\mathbf{F}), \tag{2.23}$$

with $P(\mathbf{Y} \mid \mathbf{F}) = \mathcal{N}(\mathbf{F}, \sigma^2 \mathrm{I}_N)$ and $p(\mathbf{F}) = \mathcal{N}(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X}))$.

More generally, GPs can be used in discriminative tasks to model input-varying parameters of the likelihood and applying the correct link function, *e.g.*, Gaussian process classification, Log-Gaussian Cox processes.

It is important to note that GPs are not only restricted to discriminative probabilistic models. For instance, one can easily place a prior on the index $\mathbf{X}$ in (2.21) to obtain a generative model known as the Gaussian Process Latent Variable Model (GPLVM) [Lawrence, 2005]. Moreover, it is also feasible to envision more complex probabilistic models where some of the variables are modelled as GPs, *e.g.*, a state-space model with GP components [Turner et al., 2010].

2.1.2.3 Deep Gaussian Processes

While GPs offer a flexible, non-parametric prior over functions, they are limited in expressivity by their marginal Gaussianity assumption and the choice of the kernel. There is a great host of work that derive more general stochastic processes, *e.g.*, Student's t processes [Shah et al., 2014] and Elliptical Processes [Bånkestad et al., 2020], and more flexible kernels, *e.g.*, spectral mixture kernels [Wilson and Adams, 2013] and non-separable kernels [Wang et al., 2020]. However, this type of prior specification is still somewhat restrictive and is based on rigid modelling assumptions. Another alternative is to specify a prior as a stochastic process made by composing multiple Gaussian processes. This is known as a Deep Gaussian Process (DGP) prior [Damianou and Lawrence, 2013]. Again, recall the non-linear setting in (2.13), one can set a DGP prior on Φ by assuming

$$\boldsymbol{\nu} = \Phi(\mathbf{X}) = \varphi_L(\varphi_{L-1}(\dots \varphi_1(\mathbf{X}))), \tag{2.24}$$

with

$$\varphi_l(\cdot) \sim \mathcal{GP}(m_l(\cdot), k_l(\cdot, \cdot)), \qquad \forall\, l \in \{1, \dots, L\}. \tag{2.25}$$

DGP priors can be used as a drop-in replacement to GP priors in any probabilistic model. Although, it is important to note that the result of the DGP composition is generally not a Gaussian process and forgoes the interpretability that comes with specifying a GP prior. DGPs retain other properties that make standard GPs appealing in probabilistic modelling such as their non-parametric construction but with added flexibility and expressiveness.

Intuition and Examples: As in the case of shallow GPs, DGPs can be seen as distributions over functions, albeit with different properties to GPs. An intu-

itive way to look at DGPs, is to consider the case of a shallow GP indexed by a deterministic transformation of the index set. This transformed process is not necessarily Gaussian with respect to the original index set; however, depending on the applied transformation, it might be able to capture input-to-output relationships that cannot be modelled with a GP on the original index set. This is similar to a change of basis in parametric discriminative models, *e.g.*, one can model a quadratic function in a linear regression setting by transforming the inputs with a quadratic transformation.

In all but the simplest problems, it is difficult to specify an input transformation that accurately captures the input-to-output relationship. A solution to this in the parametric case is to use a parameterised adaptive transformation whose parameters are inferred when fitting the model. While this can also be applied to the inputs of GPs and can yield expressive priors [Wilson et al., 2016], another method is to specify this transformation non-parametrically with another GP. The latter construction forms a two-layer DGP, where the original inputs are stochastically warped with the first GP and the outputs are modelled as a function of the transformed inputs with a GP prior. The same reasoning carries over for DGP constructions with L layers; however, in this case, the inputs are warped with a DGP with $L-1$ layers.

There are two advantages to this function prior construction. The first advantage is the additional expressivity it gives. For instance, non-stationary processes can be constructed by composing stationary GPs into a DGP. The second advantage is the propagation of uncertainty in the warping from the inputs to the outputs. In most machine learning problems, the input transformation is unknown; hence, deterministic transformations can overfit to a specific training dataset. DGPs circumvent that by transforming the inputs stochastically, thus allowing uncertainty in the transformation to propagate from one layer to the next.

Fig. 2.4, Fig. 2.5 and Fig. 2.6, show draws from zero-mean DGP priors for different kernels and different number of layers. Notice the qualitative difference in the draws from the shallow GP case in Fig. 2.2. For instance, the samples from the Squared Exponential DGP in Fig. 2.4 exhibit non-stationary behaviour. This is no present in the shallow Squared Exponential GP in Fig. 2.2.

An interesting property seen in Fig. 2.4, Fig. 2.5 and Fig. 2.6, is the degeneracy of the DGP samples with the number of layers. This is a known issue in these types of models [Duvenaud et al., 2014] that can be avoided by using the identity mean function [Salimbeni and Deisenroth, 2017], or by introducing skip connections

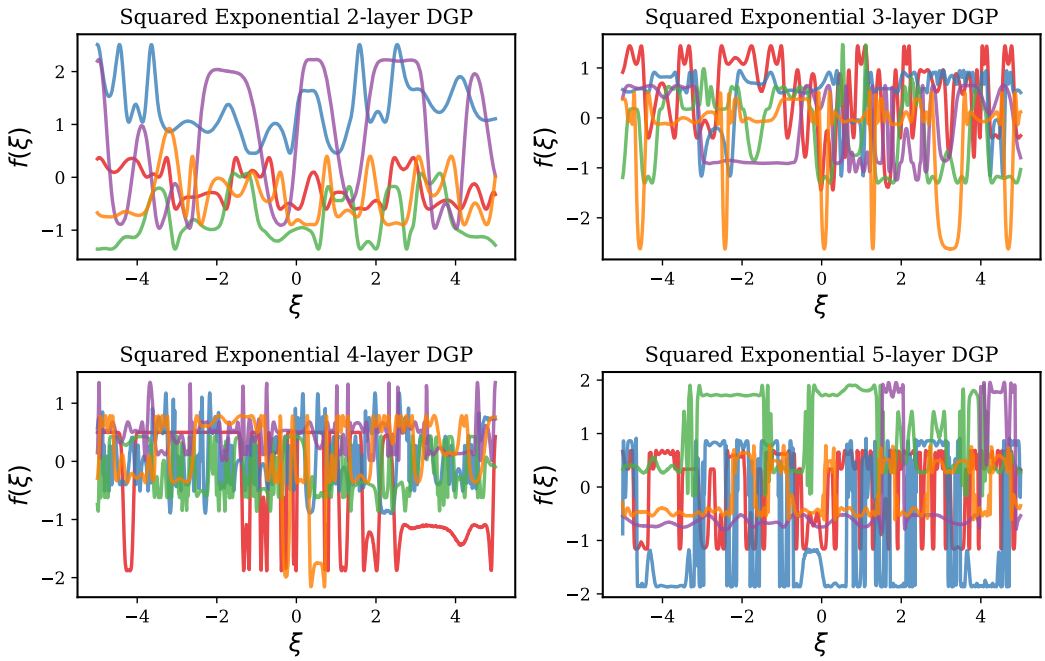


Figure 2.4: Samples from 4 Squared Exponential DGPs with different numbers of layers.

from the inputs to each layer [Neal, 1996; Duvenaud et al., 2014].

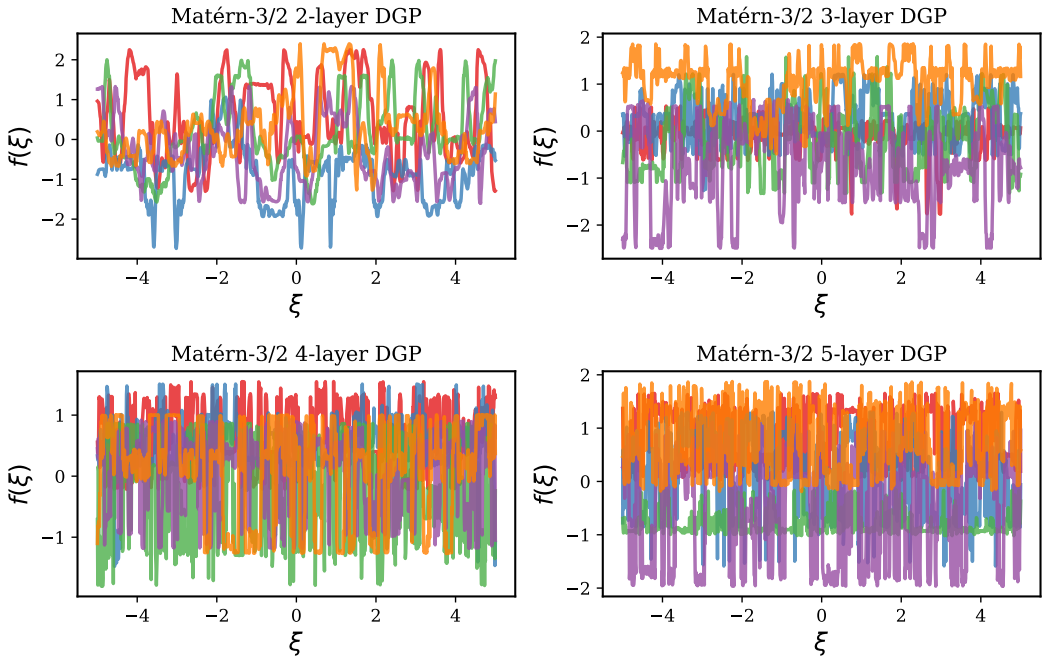


Figure 2.5: Samples from 4 Matérn-3/2 DGPs with different numbers of layers.

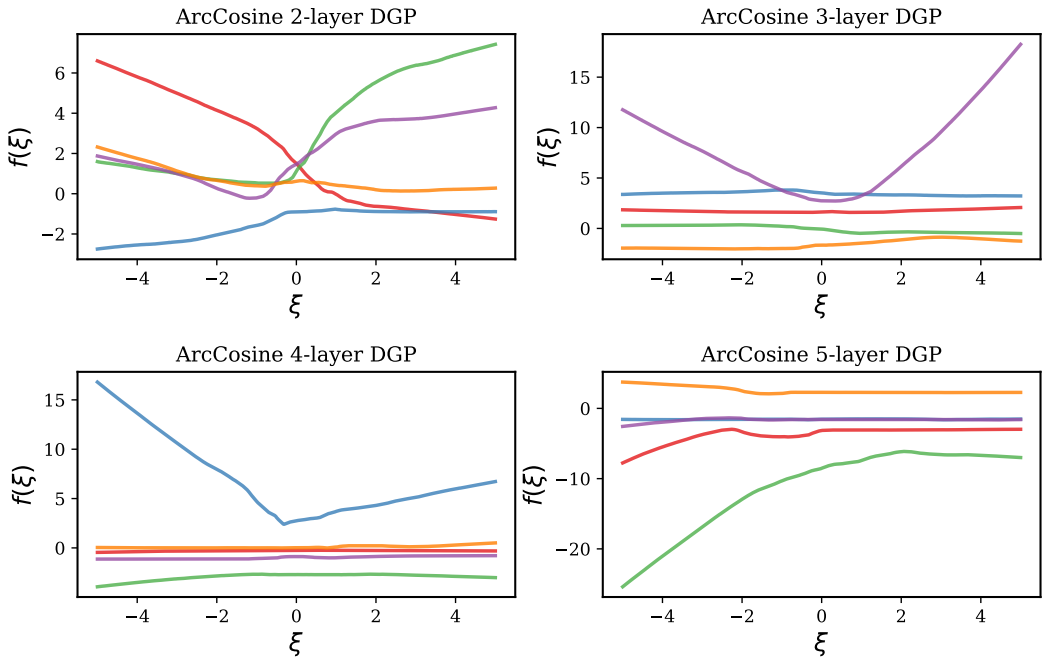


Figure 2.6: Samples from 4 ArcCosine DGPs with different numbers of layers.

2.2 Inference

Inference is the mechanism in which a modeller reasons about the plausible values of the latent variables in a probabilistic model. Computationally, this manifests in calculating the posterior distribution, $p(\mathbf{x} \mid \mathbf{y})$. The full posterior distribution describes the uncertainty in the values of the latent variables and allows the modeller to make decisions based on those values. This section delves into the topic of inference in probabilistic models. In particular, it focuses on Bayesian inference and some of its approximations. It introduces inference tools that are necessary for the rest of this thesis.

2.2.1 Bayes's Rule and the Difficulty of Bayesian Inference

The posterior can be easily derived by using by Bayes's Rule,

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}. \tag{2.26}$$

As the name suggests, Bayes's rule was first conceived by the Reverend Thomas Bayes and was published postmortem by Richard Price in 1763 [Bayes, 1763]. However, its current mathematical form is attributed to Pierre-Simon Laplace [Laplace, 1820].

Bayes's rule is a simple application of the sum, product and conditioning rules of probability; however, its interpretation is far-reaching. Bayes's rule provides a consistent method with which an agent can update its beliefs: an agent *updates* its prior beliefs about some proposition, by observing data that relate to this proposition. It is important to emphasise that the agent updates its prior beliefs after seeing evidence; it does not discard them or replace them with other beliefs.

Mathematically, the agent's prior beliefs are encoded in the prior, $p(\mathbf{x})$, and its observations in the likelihood, $p(\mathbf{y} \mid \mathbf{x})$. The two are combined into the posterior $p(\mathbf{x} \mid \mathbf{y})$ which represents the agent's updated beliefs.

Mechanistically, one can also view Bayes's rule as a filtering process. In this context, the likelihood acts as a filter which only allows propositions that are consistent with the observed data. The prior is filtered through the likelihood to obtain the posterior.

While the above views provide an intuitive understanding of Bayes's rule in (2.26), they omit one technical ingredient. The normaliser $p(\mathbf{y})$ ensures that the posterior $p(\mathbf{x} \mid \mathbf{y})$ is a normalised probability distribution, *i.e.*, it integrates to one. This is an essential ingredient to the application of Bayes's rule and is, unfortunately,

the source of its computational difficulty.

Note that one can write

$$p(\mathbf{y}) = \int p(\mathbf{y} \,|\, \mathbf{x}) p(\mathbf{x}) \, \mathrm{d}\mathbf{x}, \tag{2.27}$$

for continuous latent variables or $p(\mathbf{y}) = \sum_i p(\mathbf{y} \,|\, \mathbf{x}_i) p(\mathbf{x}_i)$ for discrete latent variables. This provides a conceptually easy way to computing the normalising constant; however, computationally this can be prohibitive since the integral in (2.27) is seldom tractable except for the simplest models[5]. For this reason, one usually evaluates Bayes's rule approximately, either analytically, *e.g.*, with Laplace's approximation, or numerically, *e.g.*, with Monte Carlo methods or Variational Inference.

Finally, an important property of Bayes's rule is that it is continuously applicable. Imagine the agent has access to two conditionally independent pieces of information (*i.e.*, observed data) $\mathbf{y}_1$ and $\mathbf{y}_2$. These data are observed sequentially, where the agent observes $\mathbf{y}_1$ first, and consequently $\mathbf{y}_2$ at a later point in time. Bayes's rule is applicable at any time period and for any prior information; hence, the agent can either wait until it observes all the information and then updates its prior beliefs, or it can update its beliefs after observing $\mathbf{y}_1$ and then use the posterior $p(\mathbf{x} \,|\, \mathbf{y}_1)$ as a prior when observing $\mathbf{y}_2$. This is shown below

$$\begin{aligned} p(\mathbf{x} \,|\, \mathbf{y}_1, \mathbf{y}_2) &= \frac{p(\mathbf{y}_1, \mathbf{y}_2 \,|\, \mathbf{x}) p(\mathbf{x})}{p(\mathbf{y}_1, \mathbf{y}_2)} \\ &= \frac{p(\mathbf{y}_2 \,|\, \mathbf{x}) p(\mathbf{y}_1 \,|\, \mathbf{x}) p(\mathbf{x})}{p(\mathbf{y}_2 \,|\, \mathbf{y}_1) p(\mathbf{y}_1)} \\ &= \frac{p(\mathbf{y}_2 \,|\, \mathbf{x}) p(\mathbf{x} \,|\, \mathbf{y}_1)}{p(\mathbf{y}_2 \,|\, \mathbf{y}_1)} \\ &= \frac{p(\mathbf{y}_2 \,|\, \mathbf{x}) p(\mathbf{x} \,|\, \mathbf{y}_1) p(\mathbf{x})}{\int p(\mathbf{y}_2 \,|\, \mathbf{x}) p(\mathbf{x} \,|\, \mathbf{y}_1) \, \mathrm{d}x}. \end{aligned} \tag{2.28}$$

2.2.2 Inference versus Learning

In probabilistic machine learning, it is important to make the distinction between *inference* and *learning*. As explained before, inference refers to the concept of fitting the model to observed data by computing the posterior. This concept is similar to how learning is viewed in classical machine learning, where the term refers to fitting the model to training data. In contrast, the concept of learning in probabilistic machine learning is more nuanced. It refers to estimating the values of the hyper-

[5] In a similar vein, for discrete problems the equivalent sum can be very expensive for high-dimensional state-spaces as it scales exponentially with dimension.

parameters (nuisance parameters) of the probabilistic model (see Section 2.1 for definition) from the available data.

Hyper-parameters index a set of models; hence, the problem of learning is equivalent to that of model selection[6]. Consider two different probabilistic models $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y})$ and $p_{\boldsymbol{\rho}}(\mathbf{x}, \mathbf{y})$ that are indexed by two different sets of hyper-parameters $\boldsymbol{\theta}$ and $\boldsymbol{\rho}$ respectively. This can also be equivalently written by explicitly conditioning the random variables on the model, *i.e.*, $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y}) \equiv p(\mathbf{x}, \mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\theta}})$ and $p_{\boldsymbol{\rho}}(\mathbf{x}, \mathbf{y}) \equiv p(\mathbf{x}, \mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\rho}})$, where $\mathcal{M}_{\boldsymbol{\theta}}$ and $\mathcal{M}_{\boldsymbol{\rho}}$ represent the model indexed by the hyper-parameters.

In the absence of knowledge about which model to choose, a reasonable criterion is to pick the model with the higher probability given the data, *i.e.*,

$$\mathcal{M}^{\star} = \arg \max_{\{\mathcal{M}_{\boldsymbol{\theta}}, \mathcal{M}_{\boldsymbol{\rho}}\}} (p(\mathcal{M}_{\boldsymbol{\theta}} \,|\, \mathbf{y}),\, p(\mathcal{M}_{\boldsymbol{\rho}} \,|\, \mathbf{y})). \tag{2.29}$$

One can make this selection with the odds ratio, *i.e.*, choose $\mathcal{M}_{\boldsymbol{\theta}}$ if $\frac{p(\mathcal{M}_{\boldsymbol{\theta}} \,|\, \mathbf{y})}{p(\mathcal{M}_{\boldsymbol{\rho}} \,|\, \mathbf{y})} \geq 1$, otherwise choose $\mathcal{M}_{\boldsymbol{\rho}}$. While the above gives an intuitive criterion to choose between models, in practice it can be intractable to compute the model posterior odds as it requires another application of Bayes's rule. However, one can reduce this task to a simpler one by noting that

$$\begin{aligned} \frac{p(\mathcal{M}_{\boldsymbol{\theta}} \,|\, \mathbf{y})}{p(\mathcal{M}_{\boldsymbol{\rho}} \,|\, \mathbf{y})} &= \frac{p(\mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\theta}}) p(\mathcal{M}_{\boldsymbol{\theta}})}{p(\mathbf{y})} \Big/ \frac{p(\mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\rho}}) p(\mathcal{M}_{\boldsymbol{\rho}})}{p(\mathbf{y})} \\ &= \frac{p(\mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\theta}}) p(\mathcal{M}_{\boldsymbol{\theta}})}{p(\mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\rho}}) p(\mathcal{M}_{\boldsymbol{\rho}})}, \end{aligned} \tag{2.30}$$

where $p(\mathbf{y} \,|\, \mathcal{M}_{\boldsymbol{\theta}}) = \int p(\mathbf{y} \,|\, \mathbf{x}, \mathcal{M}_{\boldsymbol{\theta}}) p(\mathbf{x} \,|\, \mathcal{M}_{\boldsymbol{\theta}}) \, \mathrm{d}\mathbf{x} \equiv \int p_{\boldsymbol{\theta}}(\mathbf{y} \,|\, \mathbf{x}) p_{\boldsymbol{\theta}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} = p_{\boldsymbol{\theta}}(\mathbf{y})$ and equivalently for $\mathcal{M}_{\boldsymbol{\rho}}$. Notice that (2.30) rewrites the model posterior odds ratio as the ratio of model likelihoods multiplied by the ratio of the model priors. Furthermore, note that the model likelihood is nothing but the marginal likelihood in (2.27) under this specific model that appears in Bayes's rule in (2.26). Assuming that the two models are apriori equally likely, one can see that learning can be performed by comparing the marginal likelihood - which is also referred to as the *model evidence*

[6]It is worth noting that this is not a universal view on model selection. Some literature differentiates between model selection, and hyper-parameter tuning (also known as parameter estimation or as system identification). In this view, model selection refers to a discrete choice between competing models that are structurally different from each other; whereas, hyper-parameter tuning is viewed as a second level inference problem that uses maximum likelihood on a continuous set of variables. This thesis considers both problems as model selection (learning) problems, since both problems are addressed using the mechanism of the model evidence [MacKay, 2002]. The view adopted in this thesis is advocated in Rasmussen and Williams [2006].

due to this property - of one model versus the other.

In case of multiple models are under consideration, one can extend this criterion by simply choosing the model with the highest marginal-likelihood, under the assumption of uniformly distributed models. This is known as *model evidence maximisation* or *type-II maximum likelihood*. It also holds if the space of models is continuous, where one can maximise the parameterised marginal likelihood analytically or numerically.

Finally, one can also adopt a Bayesian treatment for the learning problem by explicitly computing the model posterior via some inference methods and then averaging over the different models weighted by their posterior probability.

2.3 Variational Inference

Variational Inference (VI) is one of the most widely used approximate Bayesian inference schemes in machine learning. In general, VI approximates probability densities through optimisation [Blei et al., 2017]. In the specific case of Bayesian inference, VI is used to find an approximation to the probability density of the posterior distribution. More explicitly, the standard VI procedure for approximate Bayesian inference aims at finding a probability density that minimises a discrepancy between itself and the true posterior density. This, of course, is an arduous task, unless the approximating density is restricted to be a member of an "easy-to-work-with" family.

2.3.1 The Mathematical Formulation of VI

VI aims to approximate the true posterior density $p(\mathbf{x} \,|\, \mathbf{y})$ with a simpler density $q(\mathbf{x})$, such that a certain measure of discrepancy D between the two densities is minimised, *i.e.*,

$$q^\star(\mathbf{x}) = \arg\min_{q \in \mathcal{Q}} D(q(\mathbf{x}) \,\|\, p(\mathbf{x} \,|\, \mathbf{y})), \tag{2.31}$$

where $\mathcal{Q}$ denotes a family of densities. In most settings, D is set to be a statistical divergence [7], in particular, the Kullback-Leibler (KL) divergence defined as:

$$\mathrm{KL}\,(q \,\|\, p) = \int_{\mathcal{X}} q \log \frac{q}{p} \,\mathrm{d}\mu, \tag{2.32}$$

[7] A *statistical divergence* or *contrast functional* is a function $D(\cdot \,\|\, \cdot) : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ over the product space $\mathcal{S} \times \mathcal{S}$ of densities with common support such that 1) $D(q \,\|\, p) \geq 0,\ \forall\ q, p \in \mathcal{S}$ and 2) $D(q \,\|\, p) = 0 \iff q = p$ [Eguchi et al., 1985]. Note that D is not necessarily symmetric.

where $\mathcal{X}$ is the common probability space and μ is a reference measure. In this thesis, $\mathcal{X}$ will be set to the Euclidean space $\mathbb{R}^n$ and μ is the Lebesgue measure (or the counting measure for the case of discrete random variables). The KL divergences between the approximating density and the true posterior with respect to the Lebesgue measure is written as

$$\mathrm{KL}\left(q(\mathbf{x}) \,\|\, p(\mathbf{x} \,|\, \mathbf{y})\right) = \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \,\mathrm{d}\mathbf{x}, \tag{2.33}$$

where the integration is over the common support of the two densities. Hence, the standard variational optimisation problem that gives the VI solution is written as

$$\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q(\mathbf{x}) \,\|\, p(\mathbf{x} \,|\, \mathbf{y})\right). \tag{2.34}$$

The objective function in (2.34) cannot be computed exactly since $p(\mathbf{x} \,|\, \mathbf{y})$ is unknown (if it was known then the inference problem is already solved!). However, one can easily formulate an alternative optimisation problem, whose solution coincides with (2.34). Recall that, by Bayes rule $p(\mathbf{x} \,|\, \mathbf{y}) = \frac{p(\mathbf{y} \,|\, \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}$. Substituting that into the KL divergence in (2.34) yields

$$\begin{aligned}
\mathrm{KL}\left(q(\mathbf{x}) \,\|\, p(\mathbf{x} \,|\, \mathbf{y})\right) &= \mathrm{KL}\left(q(\mathbf{x}) \,\|\, \frac{p(\mathbf{y} \,|\, \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}\right) \\
&= \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{\frac{p(\mathbf{y} \,|\, \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}} \,\mathrm{d}\mathbf{x} \\
&= \underbrace{\log p(\mathbf{y})}_{\text{Log-marginal likelihood}} - \underbrace{\int q(\mathbf{x}) \log \frac{p(\mathbf{y} \,|\, \mathbf{x})p(\mathbf{x})}{q(\mathbf{x})} \,\mathrm{d}\mathbf{x}}_{\text{Evidence Lower Bound}}.
\end{aligned} \tag{2.35}$$

The first term on the right-hand side of (2.35) is the logarithm of the marginal likelihood (also known as the evidence) and the second term is called the Evidence Lower Bound (ELBO), since it lower bounds the evidence term due to the non-negativity of the KL divergence on the left-hand side. Notice the log-marginal likelihood does not involve the approximate posterior $q(\mathbf{x})$, hence it can be discarded from the optimisation problem. Consequently, the VI problem can be rewritten as the minimisation of the negative ELBO, *i.e.*

$$q^\star(\mathbf{x}) = \arg\min_{q \in \mathcal{Q}} -\mathrm{ELBO}(q). \tag{2.36}$$

Note that, unlike (2.34), the objective function in (2.36) is computable. Finally, an

alternative for the ELBO can be written as

$$\begin{aligned}\text{ELBO}(q) &= \int q(\mathbf{x}) \log \frac{p(\mathbf{y} \mid \mathbf{x}) p(\mathbf{x})}{q(\mathbf{x})} \, \mathrm{d}\mathbf{x} \\ &= \int q(\mathbf{x}) \log p(\mathbf{y} \mid \mathbf{x}) - \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \, \mathrm{d}\mathbf{x} \\ &= \underbrace{\mathbb{E}_{q(\mathbf{x})}[\log p(\mathbf{y} \mid \mathbf{x})]}_{\text{Expected log-likelihood}} - \underbrace{\text{KL}\left(q(\mathbf{x}) \,\|\, p(\mathbf{x})\right)}_{\text{KL regulariser}}. \end{aligned} \tag{2.37}$$

The form of the ELBO in (2.37) offers an alternative view on the VI problem, where it could be understood under the lens of empirical loss minimisation, albeit in the space of measures rather than parameters [Zellner, 1988]. The optimisation problem in VI trades-off the fit to the observed data encoded in the expected log-likelihood term with fidelity to the prior in the KL regulariser term.

2.3.2 Flavours of Approximate Posterior Families for Variational Inference

In general, it is possible to obtain the exact posterior as a solution to the VI problem if the family $\mathcal{Q}$ is set to all probability measures $\mathcal{P}$ [Zellner, 1988]. This choice, however, is usually computationally infeasible and one sets $\mathcal{Q} \subset \mathcal{P}$ to avoid this intractability. This introduces bias to the inference problem but at the trade-off of computational tractability and lower variance (*e.g.*, compared to Monte Carlo methods). This section will briefly discuss choices of $\mathcal{Q}$ that are commonly used in probabilistic machine learning applications.

2.3.2.1 The Mean-field Approximating Family

The simplest choice for the approximating family $\mathcal{Q}$ is to assume that the latent space has an independence structure *a posteriori* and set $\mathcal{Q}$ to be the family of all distributions that adheres to this independence structure [Bishop, 2006]. More concretely, if the latent variable $\mathbf{x}$ can be partitioned into P disjoint elements, *i.e.*, $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_P]$, one can assume that the approximate posterior factorises with respect to these groups such that

$$q(\mathbf{x}) = \prod_{i=1}^{P} q_i(\mathbf{x}_i). \tag{2.38}$$

This choice is known as the *mean-field* approximating family and leads to the following *local* solution for each of the factors under the mean-filed assumption

$$q_i^\star(\mathbf{x}_i) \propto \exp\left(\mathbb{E}_{q_{-i}(\mathbf{x})}[\log p(\mathbf{y}, \mathbf{x})]\right), \tag{2.39}$$

where $q_{-i}(\mathbf{x}) = \prod_{j \neq i \wedge j \in \{1,\cdots,P\}} q_j(\mathbf{x}_j)$. Note that the solution in (2.39) is not a global solution to the VI problem as it assumes all the factors other than i are fixed. However, one can still use this solution as an update step to a coordinate descent algorithm that minimises the negative ELBO, by cycling through the factors and iteratively updating each at a time [Bishop, 2006; Blei et al., 2017]. In this case, convergence to a local optimum is guaranteed as the ELBO is convex with respect to each of the factors $q_i(\mathbf{x}_i)$ [Boyd and Vandenberghe, 2004; Bishop, 2006].

2.3.2.2 Parameterised Approximations

Another option for restricting the variational family $\mathcal{Q}$ is to specify a functional form for $q(\mathbf{x})$ parameterised by a set of variational parameters $\boldsymbol{\phi} \in \Phi$, which will be denoted as $q_{\boldsymbol{\phi}}(\mathbf{x})$. In this case, the ELBO can be written as a function of the variational parameters, *i.e.*, $\text{ELBO}(q_{\boldsymbol{\phi}}) =: \text{ELBO}(\boldsymbol{\phi})$, and the optimisation can be performed in the parameter space. This reduces the variational optimisation problem into a standard parametric optimisation problem, where many off-the-shelf optimisers can be used to optimise the ELBO. This is particularly appealing when the gradients of the ELBO w.r.t. $\boldsymbol{\phi}$ are available, which enables using gradient-based optimisation algorithms. However, one needs to be careful when using this family as the choice of parameterisation can have a significant effect on the VI solution [Gorinova et al., 2020].

One can also employ the mean-field assumption in conjunction with the parameterised approximations to simplify the objective function and reduce the number of parameters.

2.3.2.3 Amortised Approximations

In the case of latent variable models, the interest is usually in the marginals of the latent variables per observation point. In this case, it is advantageous to parameterise the approximate posterior marginals with a function of the observables. For a latent variable $\mathbf{x}_n$ associated with an observation $\mathbf{y}_n$, the approximate posterior marginal can be written as

$$q_n(\mathbf{x}_n) = q_{f_{\boldsymbol{\phi}}(\mathbf{y}_n)}(\mathbf{x}_n), \tag{2.40}$$

where $f_{\boldsymbol{\phi}}(\cdot)$ is a function of the observables that is parameterised by $\boldsymbol{\phi}$ and outputs the parameters of q_n. A standard example of this family of approximations is a Gaussian marginal whose mean vector and covariance matrix are the outputs of a neural network [Kingma and Welling, 2014]. One typically assumes a mean-field factorisation for the full-approximate posterior such that

$$q(\{\mathbf{x}_n\}_{n=1}^N) = \prod_{n=1}^N q_n(\mathbf{x}_n) = \prod_{n=1}^N q_{f_{\boldsymbol{\phi}}(\mathbf{y}_n)}(\mathbf{x}_n). \tag{2.41}$$

The main benefit in this type of approximations over free-form mean-field approximations is scalability in the number of data points, where the number of variational parameters is fixed and independent of the size of the dataset. As such, this type of approximations are known as *amortised* approximations [Gershman and Goodman, 2014; Rezende et al., 2014; Kingma et al., 2015]. Another advantage for this family is that it enables obtaining reasonable guesses for posterior marginals of the latents corresponding to unseen data, where one can simply evaluate the parameterising function on new data points to obtain the posterior marginals of their latents.

While this type of approximation provides significant gains in computation, it is not as expressive as the free-form approximations. This is mainly due to the limited capacity of the amortising function $f_{\boldsymbol{\phi}}(\cdot)$ [Cremer et al., 2018]. The difference in expressiveness between the free-form approximations and the amortised approximations is known as the *amortisation gap*, which reduces as the capacity of $f_{\boldsymbol{\phi}}(\cdot)$ increases.

2.3.3 Optimising the Variational Objective

When a mean-field factorisation is assumed with free-form distributions, it is possible to optimise the ELBO by performing the coordinate updates in (2.39) and cycling through the factors as discussed in Section 2.3.2.1. For exponential-family conditional models, the optimal local coordinate descent update will be in the same family and can be written as an update in parameters of the corresponding exponential family form [Ghahramani and Beal, 2000]. Note that in the general case, there are no guarantees on the convexity of the ELBO, even when a mean-field factorisation is assumed. Consequently, the coordinate descent updates, as well as other methods that will be presented later, are only guaranteed to arrive at a local optimum.

For the general parameterised case, including amortised approximations, one can optimise the ELBO by following its gradient with respect to the variational

parameters. Like many parametric optimisation problems, VI can be solved with gradient-based methods such as Gradient Descent, Quasi-Newton, *etc.* The challenge in this case is to compute or estimate the gradient of the ELBO. The rest of this section will explore some of the techniques that are used to compute this gradient.

2.3.3.1 Closed-form gradients and automatic differentiation

Closed-form gradients can be obtained by analytically computing the expectations in (2.37) and differentiating with respect to the variational parameters. In the early days of VI, the ELBO gradient was computed analytically and the user had to implement it in code to be able to perform the optimisation. However, more recently, automatic differentiation has been used to automate this task, where the user only implements the objective function (after analytically computing the expectations) as a computational graph and the automatic differentiation engine outputs the gradient [Kucukelbir et al., 2017]. The applications of VI in this thesis generally use automatic differentiation to obtain the gradient.

2.3.3.2 Mini-batching and stochastic gradients

Large datasets create computational problems for inference algorithms in general due to the requirements to process the entire dataset to make inferential queries. Standard VI suffers from the same problem where the computation of the expected log-likelihood term becomes prohibitive for large datasets. Stochastic Variational Inference (SVI) [Hoffman et al., 2013], presents a solution to this problem, using a stochastic estimate of the gradient from data subsamples. Consequently, the gradient estimate is used in a (stochastic) gradient-based optimiser to optimise the ELBO.

Let $\mathbf{y} = [\mathbf{y}_1, \cdots, \mathbf{y}_N]$ denote a dataset of N points. For independently and identically distributed data, the joint distribution factorises as $p(\mathbf{y}, \mathbf{x}) = p(\mathbf{x}) \prod_{n=1}^{N} p(\mathbf{y}_n \mid \mathbf{x})$ (or $p(\mathbf{y}, \mathbf{x}) = \prod_{n=1}^{N} p(\mathbf{y}_n \mid \mathbf{x}_n) p(\mathbf{x}_n)$ in the case of latent variable models that assume prior independence). Substituting this form into the ELBO in (2.35) (or (2.37)) de-

composes its expression into a sum of terms each corresponding a single data-point,

$$\begin{aligned}\text{ELBO}(\boldsymbol{\phi}) &= \int q_{\boldsymbol{\phi}}(\mathbf{x}) \log \prod_{n=1}^{N} \frac{p(\mathbf{y}_n \,|\, \mathbf{x}) p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} \,\mathrm{d}\mathbf{x} \\ &= \sum_{n=1}^{N} \int q_{\boldsymbol{\phi}}(\mathbf{x}) \log \frac{p(\mathbf{y}_n \,|\, \mathbf{x}) p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} \,\mathrm{d}\mathbf{x} \\ &= \sum_{n=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})} \left[\log \frac{p(\mathbf{y}_n \,|\, \mathbf{x}) p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} \right]. \end{aligned} \tag{2.42}$$

Cheap estimates of the ELBO gradient can be obtained by sub-sampling the dataset. Let $\mathcal{B} \subset \{1, \cdots, N\}$ be a random subset of indices, then the gradient of the ELBO can be estimated as

$$\nabla_{\boldsymbol{\phi}} \text{ELBO}(\boldsymbol{\phi}) \approx \frac{N}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})} \left[\log \frac{p(\mathbf{y}_b \,|\, \mathbf{x}) p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} \right]. \tag{2.43}$$

The stochastic gradient in (2.43) can be used to optimise the ELBO by updating the values of the variational parameters $\boldsymbol{\phi}$ with Stochastic Gradient Descent (SGD) [Robbins and Monro, 1951] (or variants such as Adam [Kingma and Ba, 2015]).

Note that for SVI to be applicable, the variational approximation needs to have global parameters that are not associated with a specific data-point and can be updated each step. This criterion is satisfied in the amortised parameterisation case; however, in the general parameterisation case, the model needs to contain global latent parameters in order for this to hold.

2.3.3.3 Intractable expectations and Monte Carlo gradients

Obtaining the gradient of the ELBO is still possible even if the expectations cannot be computed in closed-form. One can obtain an unbiased estimate of the ELBO gradient with Monte Carlo (MC) sampling at the expense of increased variance. In general, various unbiased MC gradient estimators could be used for the ELBO gradient (see Mohamed et al. [2020] for a comprehensive overview); however, this section will only focus on two.

The first estimator is known as the *score function* estimator or *Reinforce*

[Williams, 1992] and can be derived as follows:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\text{ELBO}(\boldsymbol{\phi}) =& \nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}\left[\log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right] = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{x}) \log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\,\mathrm{d}\mathbf{x} \\ =& \int \nabla_{\boldsymbol{\phi}} \left[q_{\boldsymbol{\phi}}(\mathbf{x}) \log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right]\mathrm{d}\mathbf{x} \\ =& \int (\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})) \log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\,\mathrm{d}\mathbf{x} + \int q_{\boldsymbol{\phi}}(\mathbf{x})\nabla_{\boldsymbol{\phi}} \log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\,\mathrm{d}\mathbf{x} \\ =& \int q_{\boldsymbol{\phi}}(\mathbf{x}) \frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} \log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\,\mathrm{d}\mathbf{x} - \int q_{\boldsymbol{\phi}}(\mathbf{x})\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \qquad (2.44) \\ =& \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}\left[(\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})) \log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right], \qquad (2.45)\end{aligned}$$

where one can arrive at (2.45) from (2.44) by noting that $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}) = \frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}$ for the first term, and $\int q_{\boldsymbol{\phi}}(\mathbf{x})\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{x})\,\mathrm{d}\mathbf{x} = 0$. Exchanging the derivatives and integrals is allowed by the dominated convergence theorem [Capinski and Kopp, 2013].

One can estimate the expression in (2.45) using MC sampling to arrive at the score function estimator,

$$\hat{g}_{\text{Reinforce}}(\boldsymbol{\phi}) := \frac{1}{S}\sum_{s=1}^{S} \left(\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})\right) \log \frac{p(\mathbf{y}\,|\,\mathbf{x}^{(s)})p(\mathbf{x}^{(s)})}{q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})}, \qquad \mathbf{x}^{(s)} \overset{\text{i.i.d.}}{\sim} q_{\boldsymbol{\phi}}(\mathbf{x}). \tag{2.46}$$

The score function estimator usually has a high variance [Mohamed et al., 2020]; however, it is still an appealing choice as it is general purpose and does not require $p(\mathbf{y}\,|\,\mathbf{x})$ to be differentiable.

The second estimator is known as the *reparameterisation* estimator or the *pathwise* estimator [Rezende et al., 2014; Kingma et al., 2015; Titsias and Lázaro-Gredilla, 2014]. The reparameterisation estimator is applicable to continuous random variables that can be generated by pushing a base randomness through a differentiable function. For a parametric differentiable map $\mathcal{T}_{\boldsymbol{\phi}}(\cdot)$, the reparameterisation estimator with respect to $\boldsymbol{\phi}$ exists if

$$\mathbf{x} \sim q_{\boldsymbol{\phi}}(\mathbf{x}) \quad \equiv \quad \mathbf{x} = \mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}), \quad \boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon}), \tag{2.47}$$

where $q(\boldsymbol{\epsilon})$ is a base distribution that does not depend on $\boldsymbol{\phi}$. The procedure on the right-hand side of the equivalence relation in (2.47) is know as the *sampling path* of $\mathbf{x}$. If this parametric sampling path exists then, by the Law of the Unconscious

Statistician [Grimmett and Stirzaker, 2001], one can write the ELBO gradient as

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\text{ELBO}(\boldsymbol{\phi}) =& \nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}\left[\log \frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right] = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{q(\boldsymbol{\epsilon})}\left[\log \frac{p(\mathbf{y}\,|\,\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))p(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))}{q_{\boldsymbol{\phi}}(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))}\right] \\ =& \mathbb{E}_{q(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\phi}} \log \frac{p(\mathbf{y}\,|\,\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))p(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))}{q_{\boldsymbol{\phi}}(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))}\right],\end{aligned} \tag{2.48}$$

where exchanging the derivatives and integrals is again allowed by the dominated convergence theorem.

The expression in (2.48) gives rise to the reparameterisation estimator for the ELBO gradient given as

$$\hat{g}_{\text{reparam}}(\boldsymbol{\phi}) := \frac{1}{S}\sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log \frac{p(\mathbf{y}\,|\,\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(s)}))p(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(s)}))}{q_{\boldsymbol{\phi}}(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(s)}))}, \qquad \boldsymbol{\epsilon}^{(s)} \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\epsilon}). \tag{2.49}$$

The reparameterisation estimator is unbiased and typically exhibits less variance than the score function estimator. Unfortunately, this type of estimators is not general-purpose as it requires a differentiable sampling path, as well as a differentiable likelihood $p(\mathbf{y}\,|\,\mathbf{x})$. This means that it is only applicable to continuous latent variables since discrete variables do not admit a differentiable sampling path. However, some work has been done on deriving continuous relaxations for the discrete case at the expense of introducing bias to the estimator [Maddison et al., 2017; Jang et al., 2017].

2.4 Inference in Gaussian Process Models

Inference in models containing GPs can be more slightly more challenging than in the parametric case. Since GPs are by definition stochastic processes, one is usually interested in obtaining the posterior distribution over the entire process rather than its finite realisation. Fortunately, one can use the marginalisation and conditioning properties of GPs presented in Section 2.1.2.1 to aid in this endeavour. the following explores inference in models involving GPs and DGPs.

2.4.1 Exact Inference in Gaussian Process Regression

Recall the GP regression setting of Section 2.1.2.2, where from (2.22) and (2.23) one can write

$$p(\mathbf{Y}, \mathbf{F}) = \mathcal{N}(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X})). \tag{2.50}$$

Now consider another point in the process $\mathbf{f}^*$ corresponding to the index $\mathbf{x}^*$, one can write the joint as

$$p(\mathbf{Y}, \mathbf{F}, \mathbf{f}^*) = \mathcal{N}\left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{x}^*) \end{bmatrix}, \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathrm{I}_N & k(\mathbf{X}, \mathbf{x}^*) \\ k(\mathbf{x}^*, \mathbf{X}) & k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right). \tag{2.51}$$

One can then use the Gaussian identities to obtain a conditional distribution on $\mathbf{Y}$, *i.e.*,

$$p(\mathbf{F}, \mathbf{f}^* \mid \mathbf{Y}) = \mathcal{N}(\mu_{\text{post}}, \Sigma_{\text{post}}), \tag{2.52}$$

where,

$$\mu_{\text{post}} = m(\mathbf{X}_{\text{aug}}) + k(\mathbf{X}_{\text{aug}}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathrm{I}_N)^{-1}\mathbf{Y}, \tag{2.53}$$

$$\Sigma_{\text{post}} = k(\mathbf{X}_{\text{aug}}, \mathbf{X}_{\text{aug}}) - k(\mathbf{X}_{\text{aug}}, \mathbf{X})\left(k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathrm{I}_N\right)^{-1} k(\mathbf{X}, \mathbf{X}_{\text{aug}}), \tag{2.54}$$

and $\mathbf{X}_{\text{aug}} := \begin{bmatrix} \mathbf{X} \\ \mathbf{x}^* \end{bmatrix}$. The distribution in (2.52) is the posterior distribution of a finite realisation of the process $\mathbf{f}^*$ given the data Y. Notice that this is also a finite marginal of another GP which, by Kolmogorov's extension theorem, is given as $\mathcal{GP}(m_{\text{post}}(\cdot),\, k_{\text{post}}(\cdot, \cdot))$, where

$$m_{\text{post}}(\cdot) = m(\cdot) + k(\cdot, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathrm{I}_N)^{-1}\mathbf{Y}), \tag{2.55}$$

$$k_{\text{post}}(\cdot, \cdot) = k(\cdot, \cdot) - k(\cdot, \mathbf{X})\left(k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathrm{I}_N\right)^{-1} k(\mathbf{X}, \cdot). \tag{2.56}$$

Finally, going back to (2.50), one can obtain the marginal likelihood of this model by integrating out the latent variables $\mathbf{F}$,

$$p(\mathbf{Y}) = \int p(\mathbf{Y}, \mathbf{F})\, \mathrm{d}\mathbf{F} = \mathcal{N}(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathrm{I}_N). \tag{2.57}$$

As explained in Section 2.2.2, the marginal likelihood in (2.57) can be used to learn the hyper-parameters of the model such as the kernel parameters and the mean function.

2.4.2 Sparse Variational Inference in Gaussian Process Model

Exact inference and learning in GP models is computationally expensive, since evaluating the posterior and the marginal likelihood is $\mathcal{O}(N^3)$ where N is the number of observations, *i.e.*, number of rows of $\mathbf{Y}$. Furthermore, exact inference is intractable for non-conjugate models, leaving only a very small host of problems where GP

posteriors could be computed exactly. There is a large body of work that deals with the problem of intractability by approximating the posterior using different methods such as Laplace's approximation, Variational Inference, Expectation Propagation [Rasmussen and Williams, 2006], Markov Chain Monte Carlo approximations [Filippone et al., 2013], *etc.*

For the problem of scalability, by far the most popular solution is Sparse Approximations, which aim to reduce the size of the problem by performing inference on a subset of the latent variables and, consequently, conditioning on them [Snelson and Ghahramani, 2005; Quiñonero-Candela and Rasmussen, 2005; Titsias, 2009]. Most of this work is interpreted as an approximation to the model rather than to the posterior [Quiñonero-Candela and Rasmussen, 2005], with one notable exception: the *sparse variational inference* framework for GPs. This framework approximates the posterior process with a sparse approximation using variational inference to not only improve the scalability of the inference, but to also allow for inference in non-conjugate models.

Consider a general discriminative GP model specified in Section 2.1.2.2. The joint model is given by

$$p(\mathbf{F}, \mathbf{Y}) = p(\mathbf{Y} \mid \mathbf{F})p(\mathbf{F}), \tag{2.58}$$

where $p(\mathbf{Y} \mid \mathbf{F})$ is a likelihood parameterised by $\mathbf{F}$ and $p(\mathbf{F})$ is the marginal of the GP prior evaluated at the indices X, *i.e.*, $p(\mathbf{F}) = \mathcal{N}(m(\mathbf{X}), k(\mathbf{X}, \mathbf{X}))$. The goal of the inference is to find the posterior measure of the stochastic process given the data, *i.e.*, $p(f(\cdot) \mid \mathbf{Y})$. In the exact inference case with conjugate likelihoods (see Section 2.4.1) this was arrived at by using a Kolmogorov's extension theorem argument. Unfortunately, for variational inference (see Section 2.3), this argument does not hold as approximating the marginals with VI and then extending them to a stochastic process might not yield a principled approximation to the posterior process. Therefore, one needs to directly approximate the posterior process rather than its marginals. In the VI setting, this requires defining the KL-divergence between stochastic processes rather than densities. This is not straightforward since an infinite-dimensional analogue to the Lebesgue measure that dominates the true and approximating posterior measures does not exist. Therefore, more sophisticated mathematical machinery is required to define the KL-divergence between stochastic processes. Unfortunately, this is beyond the scope of this thesis; however, the interested reader is referred to Matthews et al. [2016] for an in-depth treatment for this problem. The rest of this section sketches the variational approximation for the finite marginals of the posterior process, noting that the results carry over *mutatis mutandis* to the infinite-dimensional case using the arguments in Matthews et al.

[2016].

The Sparse Variational Approximation Consider the target posterior density $p(\mathbf{F}^*, \mathbf{F} \mid \mathbf{Y})$, where $\mathbf{F}^*$ denotes a *finite* set of function values corresponding to indices in the set $\mathbf{X}^*$. Also consider a further partition into two sets, $\mathbf{F}^* = \begin{bmatrix} \mathbf{F}^{*\setminus -} \\ \bar{\mathbf{F}} \end{bmatrix}$, where $\bar{\mathbf{F}}$ are M function values corresponding to indices in the set $\mathbf{Z}$. The elements of $\mathbf{Z}$ are referred to as the *inducing locations* or *inducing inputs* and $\bar{\mathbf{F}}$ are called the *inducing variables*. One can think of $\mathbf{F}^*$ as a very large set of function values that are of interest in prediction; however, it needs to be stressed that this set is finite and does not represent the whole process. One can use VI to approximate $p(\mathbf{F}^*, \mathbf{F} \mid \mathbf{Y}) = p(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F} \mid \mathbf{Y})$ by another density $q(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F}) := p(\mathbf{F}^{*\setminus -}, \mathbf{F} \mid \bar{\mathbf{F}}) q(\bar{\mathbf{F}})$, where $p(\mathbf{F}^{*\setminus -}, \mathbf{F} \mid \bar{\mathbf{F}})$ is posited to be the marginal of a conditional GP (*cf.* (2.16) & (2.17)) and $q(\bar{\mathbf{F}}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$ is a free-form Gaussian density.

Consequently, one can perform VI by minimising the KL divergence from the true posterior $p(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F} \mid \mathbf{Y})$ to its approximation $q(\mathbf{F}^{*\setminus -}, \bar{F}, \mathbf{F})$, which is given as

$$
\begin{aligned}
&\mathrm{KL}\Big(q(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F}) \,\|\, p(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F} \mid \mathbf{Y})\Big) \\
&\quad = \int q(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F}) \log \frac{q(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F})}{p(\mathbf{F}^{*\setminus -}, \bar{\mathbf{F}}, \mathbf{F} \mid \mathbf{Y})} \,\mathrm{d}\mathbf{F}^{*\setminus -} \,\mathrm{d}\bar{\mathbf{F}} \,\mathrm{d}\mathbf{F} && (2.59) \\
&\quad = \int p(\mathbf{F}^{*\setminus -}, \mathbf{F} \mid \bar{\mathbf{F}}) q(\bar{\mathbf{F}}) \log \frac{p(\mathbf{F}^{*\setminus -}, \mathbf{F} \mid \bar{\mathbf{F}}) q(\bar{\mathbf{F}}) p(\mathbf{Y})}{p(\mathbf{F}^{*\setminus -}, \mathbf{F} \mid \bar{F}) p(\bar{\mathbf{F}}) p(\mathbf{Y} \mid \mathbf{F})} \,\mathrm{d}\mathbf{F}^{*\setminus -} \,\mathrm{d}\bar{\mathbf{F}} \,\mathrm{d}\mathbf{F} && (2.60) \\
&\quad = \int p(\mathbf{F}^{*\setminus -}, \mathbf{F} \mid \bar{\mathbf{F}}) q(\bar{\mathbf{F}}) \log \frac{q(\bar{\mathbf{F}}) p(\mathbf{Y})}{p(\bar{\mathbf{F}}) p(\mathbf{Y} \mid \mathbf{F})} \,\mathrm{d}\mathbf{F}^{*\setminus -} \,\mathrm{d}\bar{\mathbf{F}} \,\mathrm{d}\mathbf{F} && (2.61) \\
&\quad = -\int p(\mathbf{F} \mid \bar{\mathbf{F}}) q(\bar{\mathbf{F}}) \log p(\mathbf{Y} \mid \mathbf{F}) \,\mathrm{d}\bar{\mathbf{F}} \,\mathrm{d}\mathbf{F} + \int q(\bar{\mathbf{F}}) \log \frac{q(\bar{\mathbf{F}})}{p(\bar{\mathbf{F}})} \,\mathrm{d}\bar{\mathbf{F}} + \log p(\mathbf{Y}) && (2.62) \\
&\quad = \underbrace{-\mathbb{E}_{q(\mathbf{F})}[\log p(\mathbf{Y} \mid \mathbf{F})] + \mathrm{KL}\left(q(\bar{\mathbf{F}}) \,\|\, p(\bar{\mathbf{F}})\right)}_{\text{Negative ELBO}} + \underbrace{\log p(\mathbf{Y})}_{\text{Log-marginal likelihood}}, && (2.63)
\end{aligned}
$$

where the factorisation of the true posterior in (2.60) is possible by assuming that $\mathbf{F}$ is sufficient for $\mathbf{Y}$ and $q(\mathbf{F})$ over which the expectation term is taken is $q(\mathbf{F}) = \int p(\mathbf{F} \mid \bar{\mathbf{F}}) q(\bar{\mathbf{F}}) \,\mathrm{d}\bar{\mathbf{F}} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$
\begin{aligned}
\boldsymbol{\mu} &= m(\mathbf{X}) + k(\mathbf{X}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} (\mathbf{m} - m(\mathbf{Z})), && (2.64) \\
\boldsymbol{\Sigma} &= k(\mathbf{X}, \mathbf{X}) - k(\mathbf{X}, \mathbf{Z}) k(\mathbf{Z}, \mathbf{Z})^{-1} (k(\mathbf{Z}, \mathbf{Z}) - \mathbf{S}) k(\mathbf{Z}, \mathbf{Z})^{-1} k(\mathbf{Z}, \mathbf{X}). && (2.65)
\end{aligned}
$$

As noted before, while this derivation is applied to the finite case, the form of KL-divergence between the approximating process and the true posterior process still reduces to (2.63)[Matthews et al., 2016]. For the infinite dimensional case, the distribution of the approximating processes is given as $q(f(\cdot), \bar{\mathbf{F}}, \mathbf{F}) = p(f(\cdot) \,|\, \bar{\mathbf{F}}) p(\mathbf{F} \,|\, \bar{\mathbf{F}}) q(\bar{\mathbf{F}})$, where $p(f(\cdot) \,|\, \bar{\mathbf{F}})$ is the GP conditional defined in (2.16) and (2.17) and conditioned on $\mathbf{Z}$.

The ELBO term in (2.63) is parameterised by the variational parameters $\mathbf{m}$ and $\mathbf{S}$, as well as by the inducing locations $\mathbf{Z}$. These can be jointly optimised using numerical optimisation methods such as LBFG-S [Nocedal and Wright, 2000]. Alternatively, in the case of GP regression with Gaussian noise, the ELBO can be optimised in closed form with respect to $\mathbf{m}$ and $\mathbf{S}$ leaving only $\mathbf{Z}$ for numerical optimisation [Titsias, 2009]. In both situations, inference is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ for a single output dimension. Furthermore, it is also possible to apply SVI (*cf.* Section 2.3.3) to this ELBO by mini-batching the data, which reduced the cost of inference to $\mathcal{O}(|\mathcal{B}|M^2)$, where $|\mathcal{B}|$ is the size of the mini-batch. However, in this case, the only way to optimise the ELBO is through stochastic numerical algorithms, such as SGD [Hensman et al., 2013]. Finally, computing the expectation term in (2.63) exactly is only possible for certain likelihoods such as the Gaussian or the Poisson; therefore, one can use numerical solutions, such as sampling and quadrature [Hensman et al., 2015], to compute this term.

2.4.3 Inference in Deep Gaussian Process Models

Inference in DGPs (see Section 2.1.2.3) is harder than in GPs due to their compositional formulation. For instance, exact inference is not possible even in the Gaussian likelihood case due to the non-linearity in the kernels [Damianou and Lawrence, 2013]. While there are many formulations for approximate inference in DGP models [Bui et al., 2016; Cutajar et al., 2017; Havasi et al., 2018], this thesis focuses on one particular method based on the sparse variational approximation, known as the *doubly stochastic sparse variational approximation* [Salimbeni and Deisenroth, 2017].

Consider a discriminative DGP model with L layers with the following joint model,

$$p(\{\mathbf{F}^l\}_{l=1}^L, \mathbf{Y}) = p(\mathbf{Y} \,|\, \mathbf{F}^L) \left(\prod_{l=2}^{L} p(\mathbf{F}^l \,|\, \mathbf{F}^{l-1}) \right) p(\mathbf{F}^1), \tag{2.66}$$

where $p(\mathbf{Y} \,|\, \mathbf{F}^L)$ is a likelihood parameterised by the function values of the final layer $\mathbf{F}^L$, $p(\mathbf{F}^l \,|\, \mathbf{F}^{l-1})$ are the marginals of the GP prior for layer each layer l evaluated

at the previous function values $\mathbf{F}^{l-1}$, and $p(\mathbf{F}^1)$ is the marginal of the GP prior for the first layer evaluated at the indices $\mathbf{X}$.

Similar to the shallow GP case, the goal is to find the distribution of the posterior processes $p(f^1(\cdot), \ldots, f^L(\cdot) \,|\, \mathbf{Y})$. For this, consider again the finite marginals case as in Section 2.4.2, where the process in layer l is represented by a finite set of points $(\mathbf{F}^{*l}, \mathbf{F}^l)$ and $\mathbf{F}^{*l} = \begin{bmatrix} \mathbf{F}^{*\backslash -} \\ \bar{\mathbf{F}}^l \end{bmatrix}$. Hence, the posterior can be represented by

$$\begin{aligned} &p(\mathbf{F}^{*1}, \mathbf{F}^1, \ldots, \mathbf{F}^{*L}, \mathbf{F}^L \,|\, \mathbf{Y}) \\ &\qquad = p(\mathbf{F}^{1*\backslash -}, \bar{\mathbf{F}}^1, \mathbf{F}^1, \ldots, \mathbf{F}^{L*\backslash -}, \bar{\mathbf{F}}^L, \mathbf{F}^L \,|\, \mathbf{Y}) \\ &\qquad = \frac{p(\mathbf{Y} \,|\, \mathbf{F}^L) r(\mathbf{F}^{L*\backslash -}, \mathbf{F}^L) p(\bar{\mathbf{F}}^L) \ldots r(\mathbf{F}^{1*\backslash -}, \mathbf{F}^1) p(\bar{\mathbf{F}}^1)}{p(\mathbf{Y})}, \end{aligned} \tag{2.67}$$

where $r(\mathbf{F}^{l*\backslash -}, \mathbf{F}^l) = p(\mathbf{F}^{l*\backslash -}, \mathbf{F}^l \,|\, \bar{\mathbf{F}}^l, \mathbf{F}^{l-1*\backslash -}, \mathbf{F}^{l-1})$ is the GP conditional marginal ((2.16) & (2.17)), conditioned on the inducing variables $\bar{\mathbf{F}}^l$ evaluated at the function values from the previous layer $\mathbf{F}^{l-1*\backslash -}, \mathbf{F}^{l-1}$ (no conditioning for the first layer), $p(\bar{\mathbf{F}}^l)$ is the GP marginal prior for layer l at the inducing locations $\mathbf{Z}^l$. Note that this factorisation assumes independent inducing variables between layers, *i.e.*, the inducing locations are not propagated through the GP cascade (Ustyuzhaninov et al. [2020] relax this assumption).

This posterior can be approximated by the following structured mean-field density

$$\begin{aligned} &p(\mathbf{F}^{1*\backslash -}, \bar{\mathbf{F}}^1, \mathbf{F}^1, \ldots, \mathbf{F}^{L*\backslash -}, \bar{\mathbf{F}}^L, \mathbf{F}^L \,|\, \mathbf{Y}) \\ &\qquad \approx q(\mathbf{F}^{1*\backslash -}, \bar{\mathbf{F}}^1, \mathbf{F}^1) \ldots q(\mathbf{F}^{L*\backslash -}, \bar{\mathbf{F}}^L, \mathbf{F}^L \,|\, \mathbf{F}^{L*\backslash -}, \mathbf{F}^{L-1}) \qquad (2.68) \\ &\qquad := r(\mathbf{F}^{1*\backslash -}, \mathbf{F}^1) q(\bar{\mathbf{F}}^1) \ldots r(\mathbf{F}^{L*\backslash -}, \mathbf{F}^L) q(\bar{\mathbf{F}}^L). \qquad (2.69) \end{aligned}$$

As in the shallow case, $q(\bar{\mathbf{F}}^l) = \mathcal{N}(\mathbf{m}^l, \mathbf{S}^l)$ is a free-form Gaussian density.

One can formulate the VI objective by computing the KL-divergence from (2.69) to (2.67) to yield the ELBO,

$$\text{ELBO}(\{\mathbf{m}^l, \mathbf{S}^l, \mathbf{Z}^l\}_{l=1}^L) = \mathbb{E}_{q(\mathbf{F}^L)}[\log p(\mathbf{Y} \,|\, \mathbf{F}^L)] - \sum_{l=1}^{L} \text{KL}\left(q(\bar{\mathbf{F}}^l) \,\|\, q(\bar{\mathbf{F}}^l)\right). \tag{2.70}$$

The expectation term in (2.70) is intractable since computing $q(\mathbf{F}^L)$ requires marginalising all the function values in the previous layer $\{\mathbf{F}_l\}_{l=1}^L$. However, sampling from it is trivial through ancestral sampling, where each $\mathbf{F}^l$ is sampled from

$q(\mathbf{F}^l) = \int p(\mathbf{F}^l \,|\, \bar{\mathbf{F}}^l) q(\bar{\mathbf{F}}^l)\, \mathrm{d}\bar{\mathbf{F}}^l$ and fed into the next layer. Since, $q(\mathbf{F}^l)$ is Gaussian, it is possible to construct a reparameterisation estimator for the gradient of (2.70) with respect to the variational parameters. Consequently, the ELBO can be optimised with gradient-based optimisers such as SGD [Robbins and Monro, 1951] and Adam [Kingma and Ba, 2015].

It is worth stressing that there are alternative approximate inference formulations for DGPs. Some are based on VI such as Cutajar et al. [2017], where the weight-space view [Rasmussen and Williams, 2006] is taken to represent the DGP with basis functions derived by sampling random features from the covariance function of each layer. Others use alternative approximate inference strategies such as Markov Chain Monte Carlo methods (MCMC) in Havasi et al. [2018] or Expectation Propagation (EP) in Bui et al. [2016].

2.5 Sequential Monte Carlo

Monte Carlo (MC) sampling is an alternative method for approximate Bayesian inference. Instead of approximating the true posterior distribution $p(\mathbf{x} \,|\, \mathbf{y})$ with a simpler distribution $q(\mathbf{x})$, one aims at drawing samples from the true posterior, which are then used to compute the desired quantities; usually expectations of test functions with respect to the posterior, *i.e.*, $\mathbb{E}_{p(\mathbf{x} \,|\, \mathbf{y})}[\varphi(\mathbf{x})]$, where $\varphi(\cdot)$ is a test function.

There are many schemes to draw samples from a target distribution such as Inversion Sampling, Rejection Sampling, Markov Chain Monte Carlo (MCMC), Quasi-Monte Carlo (QMC), *etc.* (*cf.* Owen [2013]). However, this section deals with a specific algorithm known as Importance Sampling (IS) and two of its extensions known as Sequential Importance Sampling (SIS) and Sequential Monte Carlo (SMC).

To simplify notation in this section, the posterior density $p(\mathbf{x} \,|\, \mathbf{y})$ will be written as $\pi(\mathbf{x})$ and, for consistency, the prior density $p(\mathbf{x})$ will be written as $\pi_0(\mathbf{x})$. Most probabilistic queries involving $\pi(\mathbf{x})$ can be addressed by computing the expectation $\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})] = \int \varphi(\mathbf{x}) \pi(\mathrm{d}\mathbf{x})$, where $\varphi(\cdot)$ is the test function and $\pi(\mathrm{d}\mathbf{x})$ denotes the posterior measure. Therefore, the following exposition will focus on the computation of this expectation.

2.5.1 Importance Sampling

When $\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})]$ is unavailable in closed form, the Monte Carlo method allows for the numerical approximation of this expectation via sampling [Owen, 2013], *i.e.*

$$\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})] \approx \frac{1}{N}\sum_{s=1}^{S}\varphi(\mathbf{x}^{(s)}), \qquad \mathbf{x}^{(s)} \sim \pi(\mathbf{x}). \tag{2.71}$$

This turns the problem of integration into the problem of drawing samples from $\pi(\mathbf{x})$. Most sampling schemes involve drawing samples from an alternative distribution, known as the *proposal distribution*, and consequently transforming the samples such that they follow the original distribution which is called the *target distribution*. Importance Sampling (IS) is one of those schemes. IS involves sampling from a proposal distribution $q(\mathbf{x})$ and then correcting for the discrepancy between $\pi(\mathbf{x})$ and $q(\mathbf{x})$ by weighting those samples [Owen, 2013]. More formally, IS can be seen as a change of measure procedure on the original expectation, *i.e.*

$$\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})] = \int \varphi(\mathbf{x})\pi(\mathrm{d}\mathbf{x}) \tag{2.72}$$

$$= \int \varphi(\mathbf{x})\frac{\pi(\mathrm{d}\mathbf{x})}{q(\mathrm{d}\mathbf{x})}q(\mathrm{d}\mathbf{x}) \tag{2.73}$$

$$= \mathbb{E}_{q(\mathbf{x})}\left[\varphi(\mathbf{x})\frac{\pi(\mathrm{d}\mathbf{x})}{q(\mathrm{d}\mathbf{x})}\right], \tag{2.74}$$

where $q(\mathrm{d}\mathbf{x})$ denotes the proposal measure and $\frac{\pi(\mathrm{d}\mathbf{x})}{q(\mathrm{d}\mathbf{x})}$ is the Radon-Nikodym derivative of $\pi(\mathrm{d}\mathbf{x})$ w.r.t. $q(\mathrm{d}\mathbf{x})$. If both $\pi(\mathrm{d}\mathbf{x})$ and $q(\mathrm{d}\mathbf{x})$ are dominated by the Lebesgue measure (which is assumed throughout this thesis), this derivative can be written as the ratio of the two densities $\frac{\pi(\mathbf{x})}{q(\mathbf{x})}$. Thus, the IS estimate of the expectation is given by

$$\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})] = \mathbb{E}_{q(\mathbf{x})}\left[\varphi(\mathbf{x})\frac{\pi(\mathbf{x})}{q(\mathbf{x})}\right] \approx \frac{1}{S}\sum_{s=1}^{S}\varphi(\mathbf{x}^{(s)})\frac{\pi(\mathbf{x}^{(s)})}{q(\mathbf{x}^{(s)})}, \qquad \mathbf{x}^{(s)} \sim q(\mathbf{x}). \tag{2.75}$$

The IS estimate in (2.75) assumes that both densities, $\pi(\mathbf{x})$ and $q(\mathbf{x})$, are normalised and can be evaluated. However, this is usually not the case in practical Bayesian inference since the posterior normalising constant is usually unknown. Hence, the IS estimator needs to be adjusted to take this into account. The Self-normalised Importance Sampling estimator provides an extension to the IS estimator that takes into account the absence of the normalising constant at the cost of some bias [Naesseth et al., 2019]. Let $\pi(x) = \frac{\Pi(\mathbf{x})}{Z}$, where $\Pi(\mathbf{x}) = p(\mathbf{y} \,|\, \mathbf{x})$ is the joint

distribution and $Z = p(\mathbf{y})$ is the normalising constant which is also the marginal likelihood. One can write

$$\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})] = \mathbb{E}_{q(\mathbf{x})}\left[\varphi(\mathbf{x})\frac{\pi(\mathbf{x})}{q(\mathbf{x})}\right] = \frac{1}{Z}\mathbb{E}_{q(\mathbf{x})}\left[\varphi(\mathbf{x})\frac{\Pi(\mathbf{x})}{q(\mathbf{x})}\right] \tag{2.76}$$

$$= \frac{\mathbb{E}_{q(\mathbf{x})}\left[\varphi(\mathbf{x})\frac{\Pi(\mathbf{x})}{q(\mathbf{x})}\right]}{\mathbb{E}_{\pi_0(\mathbf{x})}[p(\mathbf{y}\,|\,\mathbf{x})]} = \frac{\mathbb{E}_{q(\mathbf{x})}\left[\varphi(\mathbf{x})\frac{\Pi(\mathbf{x})}{q(\mathbf{x})}\right]}{\mathbb{E}_{q(\mathbf{x})}[\Pi(\mathbf{x})]}, \tag{2.77}$$

where a change of measure was applied on the normalising constant. Denoting $\omega(\mathbf{x}) := \frac{\Pi(\mathbf{x})}{q(\mathbf{x})}$ and evaluating both expectations with MC, one can arrive at the Self-normalised IS estimator

$$\mathbb{E}_{\pi(\mathbf{x})}[\varphi(\mathbf{x})] \approx \frac{\sum_{s=1}^{S}\varphi(\mathbf{x}^{(s)})\omega(\mathbf{x}^{(s)})}{\sum_{s=1}^{S}\omega(\mathbf{x}^{(s)})}, \qquad \mathbf{x}^{(s)} \sim q(\mathbf{x}). \tag{2.78}$$

The estimator in (2.78) is slightly biased since it is given as the ratio of two other MC estimators. However, this is a reasonable trade-off in practice since it provides means to sample from the posterior distribution without knowledge of the normalising constant. Furthermore, the quantity $\frac{1}{S}\sum_{s=1}^{S}\omega(\mathbf{x}^{(s)})$ provides an estimate of the marginal likelihood, which is computed for free when evaluating this estimator.

Denoting the normalised weights as $\mathsf{w}^{(s)} = \frac{\omega(\mathbf{x}^{(s)})}{\frac{1}{S}\sum_{s=1}^{S}\omega(\mathbf{x}^{(s)})}$, one can write the approximate posterior density obtained with Self-normalised Importance Sampling as

$$\pi(\mathbf{x}) \approx \hat{\pi}(\mathbf{x}) = \sum_{s=1}^{S}\mathsf{w}^{(s)}\delta_{\mathbf{x}^{(s)}}(\mathbf{x}), \tag{2.79}$$

where $\delta_y(\cdot)$ denotes the Dirac delta function.

Another way of seeing the estimator in (2.78), is to plug in the approximation (2.79) into the original expectation and then evaluate the integral in closed form. This provides an alternative view on importance sampling, where the target distribution is approximated by a set of particles sampled from the proposal distribution and weighted according to their likelihood. The algorithmic procedure for this posterior approximation is given in Algorithm 1. Note that for the remainder of this exposition, the Self-normalised Importance Sampling procedure will be simply referred to as Importance Sampling (IS).

2.5.2 Sequential Importance Sampling

The effectiveness of IS hinges on the proposal distribution $q(\mathbf{x})$ and proposal design is an important problem in the IS literature. While there are many ways one can

Algorithm 1 Posterior Approximation with (Self-normalised) Importance Sampling

Input: Observations $\mathbf{y}$, number of samples S, proposal distribution $q(\mathbf{x})$.
Sample: $\mathbf{x}_t^{(s)} \sim q(\mathbf{x})$ ▷ **for** $s = 1$ **to** S.
Weight: $\mathsf{w}^{(s)} \propto \frac{p(\mathbf{y} \mid \mathbf{x}^{(s)})\pi_0(\mathbf{x}^{(s)})}{q(\mathbf{x}^{(s)})}$ ▷ **for** $s = 1$ **to** S.
Return $\hat{\pi}(\mathbf{x}) = \sum_{s=1}^{S} \mathsf{w}^{(s)} \delta_{\mathbf{x}^{(s)}}(\mathbf{x})$

specify a proposal [Naesseth et al., 2019], this section will focus on auto-regressive designs and the Sequential Importance Sampling (SIS) algorithm [Doucet and Johansen, 2011]. SIS is a variant of IS that specifies an auto-regressive representation of the proposal distribution and computes the weights sequentially. In SIS, the proposal distribution is specified as

$$q_t(\mathbf{x}_{1:t}) = q_t(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) q_{t-1}(\mathbf{x}_{1:t-1}), \tag{2.80}$$

i.e., the proposal distribution decomposes into a sequence of conditional distributions. Notice that, in SIS the latent variables are now indexed by t. If the problem is sequential in nature, then the latent variables will admit this indexing; however, if the interest is in one set of latent variables, *e.g.*, $\mathbf{x}_T$, then $\mathbf{x}_{1:T-1}$ could be considered as auxiliary variables and marginalised out at the end of the sampling procedure [Naesseth et al., 2019].

Decomposing the proposal as in (2.80), allows for the recursive computation of the unnormalised importance weights $\omega_t(\mathbf{x})$ as follows

$$\omega_t(\mathbf{x}) = \frac{\Pi(\mathbf{x}_{1:t})}{q_t(\mathbf{x}_{1:t})} \tag{2.81}$$

$$= \frac{\Pi(\mathbf{x}_{1:t-1})}{q_{t-1}(\mathbf{x}_{1:t-1})} \frac{\Pi(\mathbf{x}_{1:t})}{\Pi(\mathbf{x}_{1:t-1}) q_t(\mathbf{x} \mid \mathbf{x}_{1:t-1})} \tag{2.82}$$

$$= \omega_{t-1}(\mathbf{x}) \frac{\Pi(\mathbf{x}_{1:t})}{\Pi(\mathbf{x}_{1:t-1}) q_t(\mathbf{x} \mid \mathbf{x}_{1:t-1})}. \tag{2.83}$$

This recursion inspires a sequential version of the IS algorithm, where the weights are updated according to (2.83). The SIS algorithm for approximating a posterior distribution is presented in Algorithm 2.

While conceptually elegant, SIS suffers from a major drawback: the variance of the weights scales unfavourably with the dimension T of the problem [Naesseth et al., 2019]. This causes *weight degeneracy* (also known as *particle degeneracy*), a phenomenon where a very small set of weights take on very large values while the

Algorithm 2 Posterior Approximation with Sequential Importance Sampling

Input: Observations $\mathbf{y}$, number of samples S, sequence of proposal distributions $\{q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1})\}_{t=1}^{T}$ with $q_1(\mathbf{x}_1 \,|\, \mathbf{x}_{1:0}) \equiv q_1(\mathbf{x}_1)$.
for $t = 1$ **to** T **do**
  Sample: $\mathbf{x}_t^{(s)} \sim q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1})$ ▷ **for** $s = 1$ **to** S.
  Weight: $\mathsf{w}_t^{(s)} \propto \mathsf{w}_{t-1}^{(s)} \frac{p(\mathbf{y} \,|\, \mathbf{x}_{1:t}^{(s)}) \pi_0(\mathbf{x}_{1:t}^{(s)})}{p(\mathbf{y} \,|\, \mathbf{x}_{1:t-1}^{(s)}) \pi_0(\mathbf{x}_{1:t-1}^{(s)}) q(\mathbf{x}_t^{(s)} \,|\, \mathbf{x}_{1:t-1}^{(s)})}$ ▷ **for** $s = 1$ **to** S.
end for
Return $\hat{\pi}(\mathbf{x}_{1:T}) = \sum_{s=1}^{S} \mathsf{w}^{(s)} \delta_{\mathbf{x}_{1:T}^{(s)}}(\mathbf{x}_{1:T})$ or $\hat{\pi}(\mathbf{x}) = \sum_{s=1}^{S} \mathsf{w}^{(s)} \delta_{\mathbf{x}_T^{(s)}}(\mathbf{x})$

rest approach zero. This means that the approximation to the target distribution is effectively represented by a very small set of particles. The next section discusses Sequential Monte Carlo algorithms which are designed to alleviate this problem.

2.5.3 Sequential Monte Carlo & Particle Filtering

Sequential Monte Carlo (SMC) methods are a family of sampling algorithms that extend SIS and are designed to mitigate the problem of particle degeneracy that SIS suffers from. The key innovation in SMC methods is the use of smart proposals that take into account information from the intermediate target distributions. More concretely, the SMC proposal distribution at time t is given as

$$q_t(\mathbf{x}_{1:t}) = q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1}) \pi_{t-1}(\mathbf{x}_{1:t-1}) \tag{2.84}$$

$$\approx q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1}) \hat{\pi}_{t-1}(\mathbf{x}_{1:t-1}). \tag{2.85}$$

This is contrasted with the SIS proposal in (2.80) which does not use information from the previous target.

In practice, the proposal alteration in (2.85) reduces to an extra sampling step where the particles at time $t-1$ are resampled according to their weights then propagated to the next iteration t. By convention, this thesis will execute the resampling step at the end of the iteration.

The resampling step can also be viewed from the perspective of evolutionary algorithms, where particles with high fitness (large weights) have a higher probability of surviving to the next round and propagating their off-spring than particles with low weights. This view is more intuitive in explaining how SMC algorithms reduce particle degeneracy.

Algorithm 3 Posterior Approximation with Sequential Monte Carlo

Input: Observations $\mathbf{y}$, number of samples S, sequence of proposal distributions $\{q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1})\}_{t=1}^{T}$ with $q_1(\mathbf{x}_1 \,|\, \mathbf{x}_{1:0}) \equiv q_1(\mathbf{x}_1)$.
for $t = 1$ **to** T **do**
  Sample: $\bar{\mathbf{x}}_t^{(s)} \sim q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1})$ ▷ **for** $s = 1$ **to** S.
  Weight: $\mathsf{w}_t^{(s)} \propto \frac{p(\mathbf{y} \,|\, \mathbf{x}_{1:t-1}^{(s)}, \bar{\mathbf{x}}_t^{(s)}) \pi_0(\mathbf{x}_{1:t-1}^{(s)}, \bar{\mathbf{x}}_t^{(s)})}{p(\mathbf{y} \,|\, \mathbf{x}_{1:t-1}^{(s)}) \pi_0(\mathbf{x}_{1:t-1}^{(s)}) q(\bar{\mathbf{x}}_t^{(s)} \,|\, \mathbf{x}_{1:t-1}^{(s)})}$ ▷ **for** $s = 1$ **to** S.
  Resample: $\mathbf{x}_t^{(s)} \sim \sum_{s=1}^{S} \delta_{\bar{\mathbf{x}}_t^{(s)}}(\mathbf{x})$ ▷ **for** $s = 1$ **to** S.
end for
Return $\hat{\pi}(\mathbf{x}_{1:T}) = \sum_{s=1}^{S} \mathsf{w}^{(s)} \delta_{\mathbf{x}_{1:T}^{(s)}}(\mathbf{x}_{1:T})$ or $\hat{\pi}(\mathbf{x}) = \sum_{s=1}^{S} \mathsf{w}^{(s)} \delta_{\mathbf{x}_T^{(s)}}(\mathbf{x})$

Deploying the SMC proposal in (2.85) yields the following weight update

$$\omega_t(\mathbf{x}) = \frac{\Pi(\mathbf{x}_{1:t})}{q_t(\mathbf{x}_{1:t})} \tag{2.86}$$

$$= \frac{\Pi(\mathbf{x}_{1:t})}{\pi(\mathbf{x}_{1:t-1}) q_t(\mathbf{x} \,|\, \mathbf{x}_{1:t-1})} \tag{2.87}$$

$$\propto \frac{\Pi(\mathbf{x}_{1:t})}{\Pi(\mathbf{x}_{1:t-1}) q_t(\mathbf{x} \,|\, \mathbf{x}_{1:t-1})}. \tag{2.88}$$

The SMC algorithm for approximating the posterior distribution is given in Algorithm 3. Note that the resampling step in this algorithm is given generically as multinomial sampling with replacement. In practice, many alternative choices can help in reducing the variance, *e.g.*, stratified resampling, systematic resampling and adaptive resampling. Nevertheless, discussion of more sophisticated resampling schemes is beyond the scope of this thesis.

As a final note, when deployed in the context of general state-space Hidden Markov Models, SMC algorithms are known as Particle Filters. While some authors use the two terms synonymously, this thesis makes this distinction clear.

2.6 Generalised Bayesian Inference

Bayesian inference is compelling when the probabilistic model is well-specified, *i.e.*, the modeller believes that the observations are generated from one of the likelihood models that are under consideration. This view is known as the M-closed view [Bernardo and Smith, 2009]. Under this view the posterior distribution captures all the uncertainty in the latent variables given the prior and the true model, allowing the modeller to make optimal decisions. However, in most real-world scenarios the true model is not possible to consider. If the modeller acknowledges that any model

specification is merely a proxy for the real-world data generating process and that the purpose of inference is to arrive at the "best" approximate belief distribution to inform a decision problem, then the modeller is set to be operating in an M-open perspective [Bernardo and Smith, 2009].

2.6.1 Inference as an Optimisation Problem

In the M-open perspective, one can view inference as an optimisation problem that minimises a statistical divergence between the assumed likelihood model and the true data generating mechanism. Indeed, standard Bayesian updating is an instance of this set-up which minimises the KL-divergence between the assumed likelihood model and the true data generating mechanism [Zellner, 1988; Bissiri et al., 2016; Jewson et al., 2018].

More generally, one does not even need to specify a model for the observations. It is sufficient to only consider a loss-function connecting the latent variables to the observations and update one's beliefs about the latent variables accordingly. This approach is proposed in Bissiri et al. [2016] under the name of Generalised Bayesian Inference (GBI).

Recall the simple Bayesian updating setup in (2.26). Bissiri et al. [2016] showed that (2.26) can be seen as a special case of a more general update rule, which can be described as a solution of an optimisation problem in the space of measures [Zellner, 1988]. In particular, let $L(\nu\,;\,p,\mathbf{y})$ be a loss-function where ν is a probability measure and p is the prior belief distribution, a belief distribution over $\mathbf{x}$ (the latent variables) that incorporates in the information in $\mathbf{y}$ (the observations) can be constructed by solving

$$\nu^\star = \arg\min_{\nu} L(\nu\,;\,p,\mathbf{y}). \tag{2.89}$$

To obtain a Bayes-type updating rule, one needs to specify this loss function as a sum of a "data term" and a "regularisation term" [Bissiri et al., 2016] given as

$$L(\nu\,;\,p,\mathbf{y}) = \lambda_1(\nu,\mathbf{y}) + \lambda_2(\nu,p), \tag{2.90}$$

where λ_1 defines a data dependent "loss" and λ_2 controls the discrepancy between the prior and the final belief distribution $\hat{\nu}$. Bissiri et al. [2016] show that the solution to the optimisation problem in (2.89) with a loss of the form (2.90) does indeed represent a belief distribution about the latent variables $\mathbf{x}$. Moreover, the form of (2.90) that satisfies the von Neumann-Morgenstern utility theorem [von

Neumann and Morgenstern, 1947] and Bayesian additivity[8] is given by

$$L(\nu\,;\,p,\mathbf{y}) = \int \ell(\mathbf{x},\mathbf{y})\nu(\mathrm{d}\mathbf{x}) + \mathrm{KL}(\nu\,||\,p), \tag{2.91}$$

which leads to a Bayes-type update [Bissiri et al., 2016; Jewson et al., 2018], given by

$$p_g(\mathbf{x}\,|\,\mathbf{y}) = p(\mathbf{x})\frac{\exp(-\ell(\mathbf{x},\mathbf{y}))}{Z}, \tag{2.92}$$

where $\ell(\mathbf{x},\mathbf{y})$ is is a loss function connecting the observations to the latent variables and $Z = \int \exp(-\ell(\mathbf{x},\mathbf{y}))p(\mathbf{x})\,\mathrm{d}\mathbf{x}$. In (2.92), the suffix g in $p_g(\cdot)$ is used to signify the generality of this distribution, *i.e.*, it is a posterior belief distribution that could potentially be different from the standard Bayes posterior.

2.6.2 The Special Case of Bayes's Rule

As a special case, if one defines $\ell(\mathbf{x},\mathbf{y})$ in (2.92) as the cross-entropy (derived from the KL-divergence) of an assumed likelihood, $p(\mathbf{y}\,|\,\mathbf{x})$, relative to the real data generating mechanism, $h_0(\mathbf{y})$ (approximated by its empirical measure, $\bar{h}_0(\mathbf{y})$, constructed from observations), the standard Bayes rule (2.26) arises as a solution. To see this, set

$$\begin{aligned}
\ell(\mathbf{x},\mathbf{y}) &= -\int \log p(\mathrm{y}\,|\,\mathbf{x})h_0(\mathrm{dy}) && (2.93)\\
&\approx -\int \log p(\mathrm{y}\,|\,\mathbf{x})\bar{h}_0(\mathrm{dy}) && (2.94)\\
&= -\int \log p(\mathrm{y}\,|\,\mathbf{x})\delta_{\mathbf{y}}(\mathrm{dy}) && (2.95)\\
&= -\log p(\mathbf{y}\,|\,\mathbf{x}). && (2.96)
\end{aligned}$$

Plugging (2.96) into (2.92) recovers Bayes's rule in (2.26).

The above motivates the use of Bayes's rule in M-open problems as a principled procedure to update belief distributions. However, in some real-world settings, Bayes's rule (*i.e.* the cross-entropy loss) might not be the best choice. This occurs when the assumed likelihood is greatly misspecified; for instance, in settings where it is known that some observations are contaminated with heavy-tailed noise that is not modelled by the assumed likelihood. In this example, the use of the cross-entropy loss can be detrimental since it gives large influence to fitting the tails of the

[8] Bayesian additivity, also referred to as coherence in Bissiri et al. [2016], says that applying a sequence of updates with subsets of the data should give rise to the same posterior distribution as single update employing all of the data.

data-generating process (*i.e.* zero-avoiding behaviour)[Jewson et al., 2018]. In this instance, other loss functions can provide more reliable posterior belief distributions. Chapter 6 is partly dedicated to addressing this problem in filtering settings.

Finally, the choice of the loss function is not restricted to information-theoretic losses such as the cross-entropy. Many losses from the machine learning literature (*e.g.* mean absolute error and the hinge loss) can be used in this setting.

2.7 Variance Reduction

For a generic test function $\phi(\mathbf{x})$, with $\mathbf{x} \sim p(\mathbf{x})$, let $\mathbb{E}\left[\phi(\mathbf{x})\right] \approx \hat{\phi}(\mathbf{x}) := \frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}^{(n)})$ where $\mathbf{x}^{(n)} \sim p(\mathbf{x})$. The variance of $\hat{\phi}(\mathbf{x})$ is given as $\frac{\sigma_\phi^2}{N}$, where σ_ϕ^2 is the variance of the random variable under the test function. This expression reveals that the variance of the estimators depends on two sources. The inherent variability in the estimated quantity σ_ϕ^2 and the number of MC samples N used in the estimator. Hence, to reduce this variance of an estimator, a simple strategy is to increase the number of MC samples. This is not an appealing choice in many applications as it can drastically increase the computation time for the estimator, especially in ML applications where $\phi(\mathbf{x})$ is usually expensive to compute and N is chosen to be very small, e.g., in many applications N is set to be 1.

An alternative strategy is to target σ_ϕ^2 directly. This can be done by constructing a different estimator $\tilde{\phi}(\mathbf{x})$ such that its expectation is equal to the original expectation required to be estimated, i.e., $\mathbb{E}[\tilde{\phi}(\mathbf{x})] = \mathbb{E}\left[\phi(\mathbf{x})\right]$. This strategy is commonly referred to as *variance reduction* and is very prevalent throughout the Monte Carlo methods literature [Owen, 2013].

This section reviews two standard variance reduction strategies for Monte Carlo methods: *control variates* and *importance sampling*. These are essential tools for developing methods that can improve the stability of inference procedures.

2.7.1 Control Variates

To reduce the variance of an unbiased stochastic estimator, one can construct an alternative estimator which has the same value in expectation but with lower variance. Control variates present an approach of doing so by injecting known information into the original estimator thus reducing its variance. More concretely, for an unbiased MC estimator $\hat{\phi}(\mathbf{x})$, define an new estimator $\tilde{\phi}_{\text{diff}}(\mathbf{x}) := \hat{\phi}(\mathbf{x}) - (\hat{\omega}(\mathbf{x}) - W)$, where $\omega(\cdot)$ is an alternative test function such that $\mathbb{E}[\omega(\mathbf{x})] = W$. It is easy to see that both $\hat{\phi}(\mathbf{x})$ and $\tilde{\phi}_{\text{diff}}(\mathbf{x})$ are equal in expectation (since $\hat{\omega}(\mathbf{x}) - W$ is zero in expectation). In this case, $\tilde{\phi}_{\text{diff}}(\mathbf{x})$ is known as the *difference* estimator [Owen, 2013], and $\omega(\mathbf{x})$ is

known as the control variate. The difference estimator has lower variance than the original MC estimator if $\phi(\mathbf{x})$ and $\omega(\mathbf{x})$ are functionally similar.

More generally, one can define the so called *regression* estimator as

$$\tilde{\phi}_{\text{reg}}(\mathbf{x}) := \hat{\phi}(\mathbf{x}) - c(\hat{\omega}(\mathbf{x}) - W), \tag{2.97}$$

where c is a constant term called the *control variate coefficient*. For the regression estimator, c can be chosen such the new estimator has a lower variance than the original estimator. Minimising the variance of the regression estimator variance $\text{Var}[\tilde{\phi}(\mathbf{x})]$ gives the optimal coefficient $c^\star = \text{Cov}[\phi(\mathbf{x}), \omega(\mathbf{x})]/\text{Var}[\omega(\mathbf{x})]$, which leads to

$$\text{Var}[\tilde{\phi}(\mathbf{x})] = (1 - \rho^2_{\phi,\omega})\,\text{Var}[\hat{\phi}(\mathbf{x})], \tag{2.98}$$

where $\rho_{\phi,\omega}$ is the Pearson correlation coefficient between $\phi(\mathbf{x})$ and $\omega(\mathbf{x})$. It is clear from the equation in (2.98) that the only requirement for variance reduction is that $\phi(\mathbf{x})$ and $\omega(\mathbf{x})$ are correlated. In practice, computing $c^\star$ is not possible, as $\text{Cov}[\hat{\phi}(\mathbf{x}), \hat{\omega}(\mathbf{x})]$ and $\text{Var}[\hat{\omega}(\mathbf{x})]$ cannot be evaluated exactly in most cases. Therefore, these need to be estimated, for example, from the sampling statistics. Chapter 4 is devoted to exploring two methods for estimating $c^\star$ in the context of optimisation of stochastic gradients.

As a final note, the control variate formulation was presented above for the scalar case. In general, the control variate can be vector values. In this case, the regression estimator is given by

$$\tilde{\phi}_{\text{reg}}(\mathbf{x}) := \hat{\phi}(\mathbf{x}) - \mathbf{c}^\intercal(\hat{\boldsymbol{\omega}}(\mathbf{x}) - \mathbf{W}), \tag{2.99}$$

where the heavy font indicates vector-valued terms, and the optimal control variate coefficient is

$$\mathbf{c}^\star = \text{Var}[\boldsymbol{\omega}(\mathbf{x})]^{-1}\,\text{Cov}[\phi(\mathbf{x}), \boldsymbol{\omega}(\mathbf{x})]. \tag{2.100}$$

2.7.2 Importance Sampling

Importance sampling can also be thought of as a variance reduction method. Recall that the variance of a standard MC estimator, $\hat{\phi}(\mathbf{x})$, is given by $\text{Var}[\hat{\phi}(\mathbf{x})] = \frac{\sigma^2_\phi}{N}$, where $\sigma^2_\phi = \int \phi^2(\mathbf{x})p(\mathbf{x})\,\mathrm{d}\mathbf{x} - \phi^2(\mathbf{x})\,\mathrm{d}\mathbf{x}$. Also recall the IS estimator for $\tilde{\phi}(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{p(x^{(n)})}{q(x^{(n)})}\phi(x^{(n)})$ in Section 2.5.1, where $x^{(n)} \sim q(\mathbf{x})$ and $q(\mathbf{x})$ is the proposal

distribution. The variance of this estimator is given as [Owen, 2013]

$$\text{Var}[\tilde{\phi}(\mathbf{x})] = \int \frac{\phi^2(\mathbf{x})p(\mathbf{x})}{q(\mathbf{x})} p(\mathbf{x})\, \mathrm{d}\mathbf{x}.$$

Taking the difference of the two estimators

$$\text{Var}[\hat{\phi}(\mathbf{x})] - \text{Var}[\tilde{\phi}(\mathbf{x})] = \int \left(1 - \frac{p(\mathbf{x})}{q(\mathbf{x})}\right) \phi^2(\mathbf{x})p(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{2.101}$$

one can see that for variance reduction to occur, the integral in (2.101) needs to be positive. This occurs for choices of $q(\mathbf{x})$ where $\frac{p(\mathbf{x})}{q(\mathbf{x})} > 1$ when $\phi^2(\mathbf{x})p(\mathbf{x})$ is small, and $\frac{p(\mathbf{x})}{q(\mathbf{x})} < 1$ when $\phi^2(\mathbf{x})p(\mathbf{x})$ is large, or $\frac{p(\mathbf{x})}{q(\mathbf{x})} < 1$ in any part of the domain.

CHAPTER 3

Multitask Learning with Gaussian Process Compositions

Compositional models enable solving complex learning tasks by combining simple building blocks and specifying the relationship between them. Many machine learning tasks go beyond single task prediction to involve multiple tasks that can be reasoned about jointly through the transfer of relevant information between the tasks. Compositionality is a desirable property in this scenario since it allows easy specification of the dependencies between the tasks and of the criteria for the transfer of information between them.

This chapter describes a compositional probabilistic model based on Gaussian process modules – the gold-standard for modern Bayesian non-parametric modelling – for learning multiple tasks simultaneously. This setting is known as *multi-task* learning, where the information transfer mechanism is specified in the model structure and is reasoned about jointly during the inference procedure.

3.1 Motivation

Multi-task learning is a broad framework that aims to leverage the shared information between the training signals of related tasks in order to improve generalisation across them. This framework has been applied to various models such as neural networks [Caruana, 1997], support vector machines [Evgeniou and Pontil, 2004] and probabilistic models [Bonilla et al., 2007].

In probabilistic machine learning, a widely successful formulation of multi-task learning builds on the Gaussian process (GP) literature. In general, GPs [Rasmussen and Williams, 2006] provide a powerful and flexible non-parametric family of machine learning models, suitable for many nonlinear tasks. Their multi-task counterparts have also been widely adopted as a viable probabilistic alternative to

classical multi-task learning models. However, the vast majority of GP multi-task models make a strong linearity assumption on the task dependencies, i.e. the tasks constitute a linear mixture of latent processes [Goovaerts, 1997; Teh et al., 2005; Bonilla et al., 2007; Alvarez et al., 2011; Wilson et al., 2012; Nguyen and Bonilla, 2014]. Indeed, some attempts have been made to introduce non-linear task relationships to GP models [Boyle and Frean, 2004; Alvarez and Lawrence, 2009; Alaa and van der Schaar, 2017; Requeima et al., 2019]; however, at the expense of additional limiting assumptions or deviation from the desirable plug-and-play nature of GP models, e.g. Alaa and van der Schaar [2017] and Requeima et al. [2019] are restricted to datasets with fully observed outputs for all input locations.

To further highlight the limitation of linear mixing models, consider the following toy example with two tasks composed by non-linearly combining two private processes and a shared process:

$$
\begin{aligned}
g(x) &= -\sin\left(8\pi(x+1)\right)/(2x+1) - x^4, \\
h_1(x) &= \sin(3x), \qquad h_2(x) = 3x, \\
f_1(x) &= \cos^2\left(g(x)\right) + h_1(x), \\
f_2(x) &= \sin\left(10x\right)g^2(x) + h_2(x),
\end{aligned}
$$

with $x \in [0, 1]$. Here, f_1 and f_2 are the two tasks, generated by g, h_1, h_2. Clearly, f_1 and f_2 have a complex non-linear relationship that linear mixing models struggle to capture as illustrated in Fig. 3.1.

To address this issue, this chapter proposes a Deep Gaussian Process (DGP) approach to non-linear multi-task learning (illustrated in blue in Fig. 3.1). DGPs a hierarchical composition of GPs that retain the advantages of GP-based models such as their non-parametric formulation, quantification of uncertainty and robustness to overfitting, while at the same time offering a richer class of models with the ability to learn complex representations from data. The proposed framework can capture highly non-linear relationships between tasks, extending the popular linear construction, while retaining most of its flexibility.

This chapter presents a compositional modelling strategy which assumes that similar tasks arise from a collection of underlying latent processes, some shared between the tasks and some that are task-specific. These are combined *non-linearly* using a further GP layer, enabling modelling of richer task relationships. Closed-form inference in the resulting model is intractable, thus the doubly stochastic variational approximation in Salimbeni and Deisenroth [2017] is extended to handle the

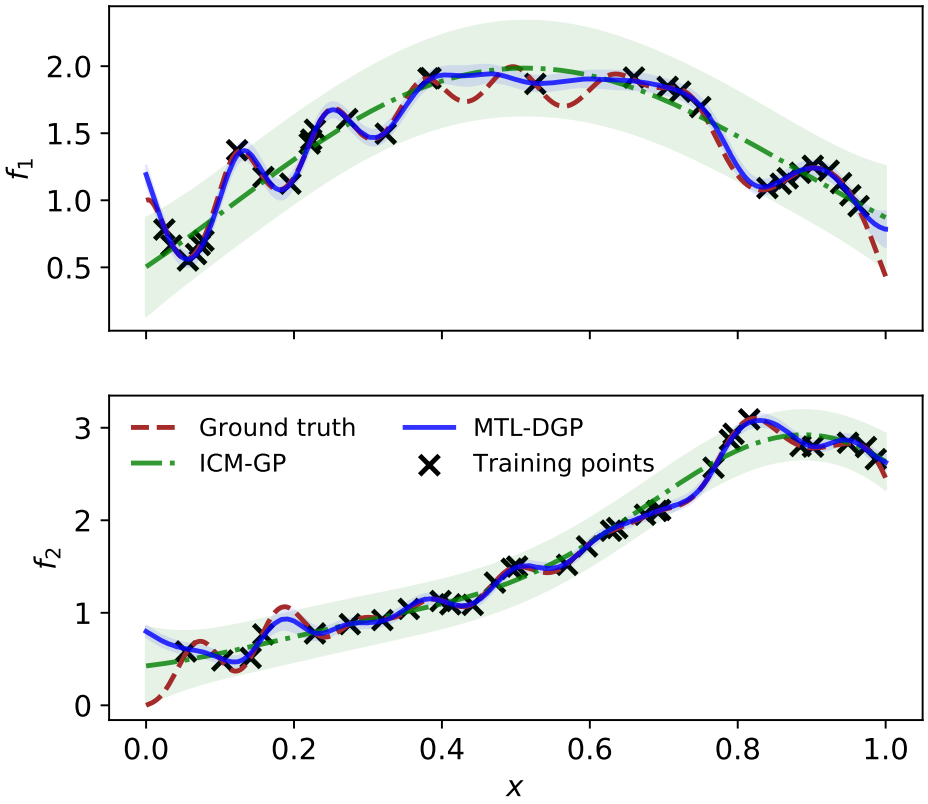


Figure 3.1: Illustration of the fit of linear (ICM-GP) vs non-linear (MTL-DGP) multi-task GP models on a toy dataset with non-linear dependencies between the tasks. The ICM-GP model suffers from negative transfer due to its linear coregionalization assumption. The non-linear multi-task DGP is robust against it.

multi-task DGP case, where the compositional nature of the model allows for easy derivation. The capabilities of this model are demonstrated through experimentation on three datasets, showing that the multi-task learning formulation for DGPs outperforms other single-task and multi-task GP-based models and neural networks on multiple benchmarks, illustrating the effectiveness of compositional models in this setting.

3.2 Background

3.2.1 Modelling with Gaussian Processes

As discussed in Section 2.1.2.2, GPs can be used in predictive modelling where the labels are modelled as a transformation of a non-parametric latent function of the

inputs [Rasmussen and Williams, 2006]

$$\mathbf{y}_n|f;\mathbf{x}_n \sim p(\mathbf{y}_n|f(\mathbf{x}_n)),$$

where f is a latent function drawn from a GP which is usually set to be zero-mean, i.e. $f(\cdot) \sim \mathcal{GP}(\mathbf{0}, k(\cdot,\cdot))$, and $p(\cdot)$ is an appropriate likelihood, e.g. Gaussian or Bernoulli. Exact inference in this model is possible only when the likelihood is Gaussian and with computational complexity of $O(N^3)$. To relax these constraints, one can use a sparse variational approximation that allows the use of other likelihoods and reduces the computational complexity to $O(NM^2)$ [Titsias, 2009; Hensman et al., 2013], where M is the number of inducing locations (pseudo inputs) which is typically much smaller than N.

The sparse variational approximation - presented in detail in Section 2.4.2 - seeks to approximate the true GP posterior p, with an approximate posterior q, by minimising the Kullback-Leibler (KL) divergence between q and p. This is equivalent to maximising a lower bound on the marginal likelihood of the model, known as the evidence lower bound (ELBO)

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{F},\mathbf{U})}\left[\log \frac{p(\mathbf{Y},\mathbf{F},\mathbf{U})}{q(\mathbf{F},\mathbf{U})}\right], \tag{3.1}$$

where $\mathbf{Y} = [\mathbf{y}_1,\ldots,\mathbf{y}_N]$ are the outputs, $\mathbf{F} = [f(\mathbf{x}_1),\ldots,f(\mathbf{x}_N)]$ are the latent function values and $\mathbf{U} = [f(\mathbf{z}_1),\ldots,f(\mathbf{z}_M)]$ are the function values at the inducing locations $\mathbf{z}_m$.

3.2.2 Extension to Deep Gaussian Processes

In addition to allowing for more scalable inference in standard GP models, the sparse variational approximation enables tractable inference in models involving the composition of GPs; for instance DGPs. Recall from Section 2.1.2.3, a DGP is a composition of functions with GP priors on each and i.i.d Gaussian noise between the layers [Damianou and Lawrence, 2013]

$$\mathbf{y}_n|f^L;\mathbf{x}_n \sim p(\mathbf{y}_n|f^L(f^{L-1}(\ldots f^1(\mathbf{x}_n)))),$$

where $f^l(\cdot) \sim \mathcal{GP}(m^l(\cdot), k^l(\cdot,\cdot))$ (to simplify notation, the inter-layer noise is absorbed into the kernel $k^l(\cdot,\cdot)$ for $l \in \{1,\ldots,L-1\}$). Inference in this model is possible through the use of the variational sparse approximation framework, where the ELBO is given by [Damianou and Lawrence, 2013; Salimbeni and Deisenroth,

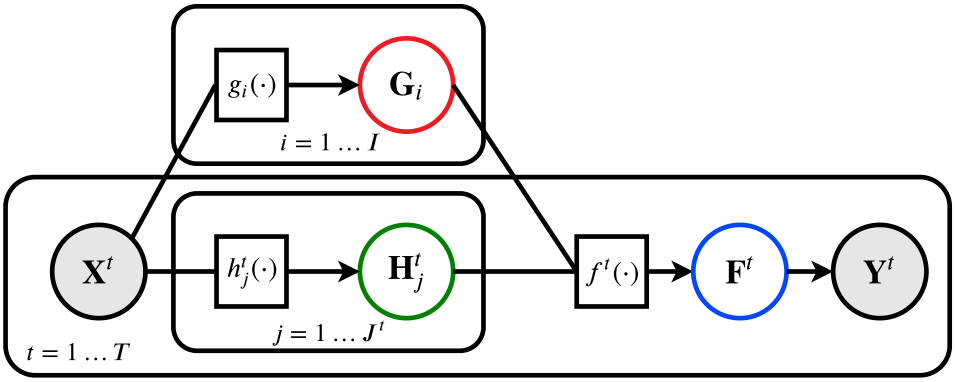


Figure 3.2: A graphical representation of the proposed non-linear multi-task DGP model. The unshaded circles represent latent variables, the shaded circles represent observed variables and the squares are computational graph nodes. Some circles are colour coded to match their corresponding terms in (3.3), (3.10) and (3.14).

2017]

$$\mathcal{L}_{\text{DGP}} = \mathbb{E}_{q(\{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L)} \left[\log \frac{p(\mathbf{Y}, \{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L)}{q(\{\mathbf{F}^l, \mathbf{U}^l\}_{l=1}^L)} \right]. \tag{3.2}$$

See Section 2.4.3 for more details.

3.3 Modelling Approach

3.3.1 DGP Multi-task Formulation

Consider the case with T tasks. Let $\mathbf{X}^t$ be the $N^t \times D^{\text{in}}$ data matrix for task $t \in \{1, \ldots, T\}$ and $\mathbf{Y}^t$ be an $N^t \times D^{\text{out}}$ matrix corresponding to the outputs for task t. This chapter proposes a model where all T tasks share a set of I latent representations $\{\mathbf{G}_i\}_{i=1}^I$ generated by a set of I functions $\{g_i(\cdot)\}_{i=1}^I$, where $g_i : \mathbb{R}^{D^{\text{in}}} \to \mathbb{R}^{D^{G_i}}$. In addition, they possess a set of J_t task specific representations $\{\mathbf{H}_j^t\}_{j=1}^{J^t}$ generated by a set of J^t task specific functions $\{h_j^t(\cdot)\}_{j=1}^{J^t}$ for each task t, where $h_j^t : \mathbb{R}^{D^{\text{in}}} \to \mathbb{R}^{D^{H_j^t}}$. The features in the combined latent space $\{\Lambda^t\}_{t=1}^T$ (with $\Lambda^t = \{\{\mathbf{G}_i\}_{i=1}^I, \{\mathbf{H}_j^t\}_{j=1}^{J^t}\}$) are warped with a task specific random function $f^t(\cdot)$ with $f^t : \mathbb{R}^{D^t} \to \mathbb{R}^{D^{\text{out}}}$ to generate noiseless outputs $\mathbf{F}^t$ for task t (Fig. 3.2).[1]

Independent GP priors are placed on both the shared and task specific latent

[1] $D^t = \sum_{i=1}^I D^{G_i} + \sum_{j=1}^J D^{H_j^t}$

functions:

$$
\begin{aligned}
g_i &\sim \mathcal{GP}(m_i(\mathbf{X}), k_i(\mathbf{X}, \mathbf{X}')), \\
h_j^t &\sim \mathcal{GP}(m_j^t(\mathbf{X}^t), k_j^t(\mathbf{X}^t, \mathbf{X}^{t\prime})), \\
f^t | g, h^t &\sim \mathcal{GP}(m^t(\Lambda^t), k^t(\Lambda^t, \Lambda^t)).
\end{aligned}
$$

This generative process describes a 2-level hierarchical model with GPs that can be formulated as a 2-layer DGP, where the first layer is responsible for extracting shared and task specific features, while the second layer transforms the features into task specific outputs.

Sparse Variational Approximation Exact inference in this model is intractable due to the non-linear dependencies introduced between the inner layers [Damianou and Lawrence, 2013] and between the tasks in the output layers. Hence, approximate inference is used where the doubly stochastic variational sparse approximation framework in Salimbeni and Deisenroth [2017] is extended to handle the multitask case.

Define the joint distribution of an expanded model as

$$
\begin{aligned}
&p(\{\mathbf{Y}^t, \mathbf{F}^t, \bar{\mathbf{F}}^t, \{\mathbf{H}_j^t, \bar{\mathbf{H}}_j^t\}_{j=1}^{J^t}\}_{t=1}^T, \{\mathbf{G}_i, \bar{\mathbf{G}}_i\}_{i=1}^I) = \\
&\prod_{t=1}^T \prod_{n=1}^{N^t} p(\mathbf{y}_n^t | \mathbf{F}^t) \prod_{t=1}^T p(\mathbf{F}^t | \bar{\mathbf{F}}^t, \Lambda^t) p(\bar{\mathbf{F}}^t) \prod_{t=1}^T \prod_{j=1}^{J^t} p(\mathbf{H}_j^t | \bar{\mathbf{H}}_j^t) p(\bar{\mathbf{H}}_j^t) \prod_{i=1}^I p(\mathbf{G}_i | \bar{\mathbf{G}}_i) p(\bar{\mathbf{G}}_i),
\end{aligned}
\tag{3.3}
$$

where the inducing variables $\{\bar{\mathbf{G}}_i\}_{i=1}^I, \{\{\bar{\mathbf{H}}_j^t\}_{j=1}^{J^t}\}_{t=1}^T, \{\bar{\mathbf{F}}^t\}_{t=1}^T$ are introduced and correspond to the values of the latent functions evaluated at a set of M inducing locations, i.e. $\bar{\mathbf{G}}_i = g_i(\mathbf{Z}_{G_i})$, $\bar{\mathbf{H}}_j^t = h_j^t(\mathbf{Z}_{\mathbf{H}_j^t})$, $\bar{\mathbf{F}}^t = f^t(\mathbf{Z}_{\mathbf{F}^t})$. Augmenting the model this way allows factorising the joint GP priors into the prior on the inducing variables and the GP conditional given the inducing variables [Titsias, 2009]. For instance, in the case of the tuple $(\mathbf{G}_i, \bar{\mathbf{G}}_i)$, one observes the following factorisation:

$$
p(\mathbf{G}_i, \bar{\mathbf{G}}_i; \mathbf{X}, \mathbf{Z}_{\mathbf{G}_i}) = p(\mathbf{G}_i | \bar{\mathbf{G}}_i; \mathbf{X}, \mathbf{Z}_{\mathbf{G}_i}) p(\bar{\mathbf{G}}_i; \mathbf{Z}_{\mathbf{G}_i}), \tag{3.4}
$$

where

$$
p(\bar{\mathbf{G}}_i; \mathbf{Z}_{\mathbf{G}_i}) = \mathcal{N}(\bar{\mathbf{G}}_i | m_i(\mathbf{Z}_{\mathbf{G}_i}), k_i(\mathbf{Z}_{\mathbf{G}_i}, \mathbf{Z}_{\mathbf{G}_i})), \tag{3.5}
$$

$$
p(\mathbf{G}_i | \bar{\mathbf{G}}_i; \mathbf{X}, \mathbf{Z}_{\mathbf{G}_i}) = \mathcal{N}(\mathbf{G}_i | \bar{\boldsymbol{\mu}}_i, \bar{\boldsymbol{\Sigma}}_i), \tag{3.6}
$$

with

$$\bar{\boldsymbol{\mu}}_i = m_i(\mathbf{X}) + \boldsymbol{\alpha}_i(\mathbf{X})^T(\bar{\mathbf{G}}_i - m_i(\mathbf{Z}_{\mathbf{G}_i})), \tag{3.7}$$

$$\bar{\boldsymbol{\Sigma}}_i = k_i(\mathbf{X},\mathbf{X}) - \boldsymbol{\alpha}_i(\mathbf{X})^T k_i(\mathbf{Z}_{\mathbf{G}_i},\mathbf{Z}_{\mathbf{G}_i})\boldsymbol{\alpha}_i(\mathbf{X}), \tag{3.8}$$

$$\boldsymbol{\alpha}_i(\mathbf{X}) = k_i(\mathbf{Z}_{\mathbf{G}_i},\mathbf{Z}_{\mathbf{G}_i})^{-1}k_i(\mathbf{Z}_{\mathbf{G}_i},\mathbf{X}). \tag{3.9}$$

Similarly for $(\mathbf{H}_j^t, \bar{\mathbf{H}}_j^t)$ and $(\mathbf{F}^t, \bar{\mathbf{F}}^t)$ taking the input pairs $(\mathbf{X}^t, \mathbf{Z}_{\mathbf{H}_j^t})$ and $(\Lambda^t, \mathbf{Z}_{\mathbf{F}^t})$ respectively.

Expanding the probability space this way allows defining an approximate variational posterior

$$q(\{\mathbf{F}^t, \bar{\mathbf{F}}^t, \{\mathbf{H}_j^t, \bar{\mathbf{H}}_j^t\}_{j=1}^{J^t}\}_{t=1}^T, \{\mathbf{G}_i, \bar{\mathbf{G}}_i\}_{i=1}^I) =$$
$$\prod_{t=1}^T p(\mathbf{F}^t|\bar{\mathbf{F}}^t,\Lambda^t)q(\bar{\mathbf{F}}^t)\prod_{t=1}^T\prod_{j=1}^{J^t} p(\mathbf{H}_j^t|\bar{\mathbf{H}}_j^t)q(\bar{\mathbf{H}}_j^t)\prod_{i=1}^I p(\mathbf{G}_i|\bar{\mathbf{G}}_i)q(\bar{\mathbf{G}}_i), \tag{3.10}$$

where the approximate posteriors q's on the right-hand side are parameterised Gaussians, e.g. $q(\bar{\mathbf{G}}_i) = \mathcal{N}(\bar{\mathbf{G}}_i|\mathbf{m}_i, \mathbf{S}_i)$ [2] and similarly for $\bar{\mathbf{H}}_j^t$ and $\bar{\mathbf{F}}^t$.

Evidence Lower Bound (ELBO) One can derive an ELBO by taking the expected log-ratio of the (3.3) and (3.10)

$$\mathcal{L} = \mathbb{E}_{q(\{\mathbf{F}^t,\bar{\mathbf{F}}^t,\{\mathbf{H}_j^t,\bar{\mathbf{H}}_j^t\}_{j=1}^{J^t}\}_{t=1}^T,\{\mathbf{G}_i,\bar{\mathbf{G}}_i\}_{i=1}^I)}\Bigg[\log \frac{\prod_{t=1}^T[\prod_{n=1}^{N^t} p(\mathbf{y}_n^t|\mathbf{F}^t)]\cancel{p(\mathbf{F}^t|\bar{\mathbf{F}}^t,\Lambda^t)}p(\bar{\mathbf{F}}^t)\prod_{j=1}^{J^t}\cancel{p(\mathbf{H}_j^t|\bar{\mathbf{H}}_j^t)}p(\bar{\mathbf{H}}_j^t)]\prod_{i=1}^I[\cancel{p(\mathbf{G}_i|\bar{\mathbf{G}}_i)}p(\bar{\mathbf{G}}_i)]}{\prod_{t=1}^T[\cancel{p(\mathbf{F}^t|\bar{\mathbf{F}}^t,\Lambda^t)}q(\bar{\mathbf{F}}^t)\prod_{j=1}^{J^t}\cancel{p(\mathbf{H}_j^t|\bar{\mathbf{H}}_j^t)}q(\bar{\mathbf{H}}_j^t)]\prod_{i=1}^I[\cancel{p(\mathbf{G}_i|\bar{\mathbf{G}}_i)}q(\bar{\mathbf{G}}_i)]} \Bigg]. \tag{3.11}$$

[2] $\mathbf{S}_i$ is usually parameterised as $\mathbf{L}_i\mathbf{L}_i^\intercal$, where is $\mathbf{L}_i$ is a triangular matrix. This parameterisation reduces the number of free parameters and insures that the Gaussian covariance matrix is symmetric and positive definite.

Since the approximate posterior q has a factorised form (3.10), one can rewrite equation (3.11) as follow:

$$\mathcal{L} = \sum_{t=1}^{T}\sum_{n=1}^{N^t} \mathbb{E}_{q(\{\mathbf{F}^t,\bar{\mathbf{F}}^t,\{\mathbf{H}_j^t,\bar{\mathbf{H}}_j^t\}_{j=1}^{J^t}\}_{t=1}^{T},\{\mathbf{G}_i,\bar{\mathbf{G}}_i\}_{i=1}^{I})}[\log p(\mathbf{y}_n^t|\mathbf{f}_n^t)]$$
$$-\sum_{t=1}^{T}\mathrm{KL}\left[q(\bar{\mathbf{F}}^t)\middle\|p(\bar{\mathbf{F}}^t)\right] - \sum_{t=1}^{T}\sum_{j=1}^{J^t}\mathrm{KL}\left[q(\bar{\mathbf{H}}_j^t)\middle\|p(\bar{\mathbf{H}}_j^t)\right] - \sum_{i=1}^{I}\mathrm{KL}\left[q(\bar{\mathbf{G}}_i)\middle\|p(\bar{\mathbf{G}}_i)\right] \tag{3.12}$$

The KL terms in equation (3.12) come from marginalising out the factors in the posterior that do not depend on the operand in the expectation terms. As discussed in Salimbeni and Deisenroth [2017], for a DGP the nth marginal of the top layer only depends on the nth marginals of all the previous layers. Hence, in the case of the presented model,

$$q(\mathbf{f}_n^t) = \int q(\mathbf{f}_n^t|\boldsymbol{\lambda}_n^t)q(\boldsymbol{\lambda}_n^t)d\mathbf{c}_\mathbf{n}^\mathbf{t}, \tag{3.13}$$

where $q(\mathbf{f}_n^t|\boldsymbol{\lambda}_n^t) = \int p(\mathbf{f}_n^t|\bar{\mathbf{F}}^t;\boldsymbol{\lambda}_n^t,\mathbf{Z}_{\mathbf{F}^t})q(\bar{\mathbf{F}}^t)d\bar{\mathbf{F}}^t$ and $q(\boldsymbol{\lambda}_n^t) = \int\prod_{j=1}^{J^t}p(\mathbf{h}_{nj}^t|\bar{\mathbf{H}}_j^t;\mathbf{x}_n^t,\mathbf{Z}_{\mathbf{H}_j^t})q(\bar{\mathbf{H}}_j^t)\prod_{i=1}^{I}p(\mathbf{g}_{ni}|\bar{\mathbf{G}}_i;\mathbf{x}_n^t,\mathbf{Z}_{\mathbf{G}_i})q(\bar{\mathbf{G}}_i)d\bar{\mathbf{H}}_j^t d\bar{\mathbf{G}}_i$. By simplifying the expected log-likelihood term, one can rewrite (3.12) as follows (the dependence on the inducing locations $\mathbf{Z}$ are reintroduced back to the notation for completion)

$$\mathcal{L} = \sum_{t=1}^{T}\sum_{n=1}^{N^t}\mathbb{E}_{q(\mathbf{f}_n^t)}[\log p(\mathbf{y}_n^t|\mathbf{f}_n^t)] - \sum_{t=1}^{T}\mathrm{KL}\left[q(\bar{\mathbf{F}}^t)\middle\|p(\bar{\mathbf{F}}^t;\mathbf{Z}_{\mathbf{F}^t})\right]$$
$$-\sum_{t=1}^{T}\sum_{j=1}^{J^t}\mathrm{KL}\left[q(\bar{\mathbf{H}}_j^t)\middle\|p(\bar{\mathbf{H}}_j^t;\mathbf{Z}_{\mathbf{H}_j^t})\right] - \sum_{i=1}^{I}\mathrm{KL}\left[q(\bar{\mathbf{G}}_i)\middle\|p(\bar{\mathbf{G}}_i;\mathbf{Z}_{\mathbf{G}_i})\right]. \tag{3.14}$$

An important note is that the integral in (3.13) is intractable, hence it is not possible to compute the expected log-likelihood term in (3.14) in closed form. However, it is straightforward to sample from the marginal posterior $q(\mathbf{f}_n^t)$ using the re-parameterisation trick [Rezende et al., 2014; Kingma et al., 2015], allowing the approximate computation of the lower bound and its gradients by Monte Carlo sampling. This bound has complexity $\mathcal{O}(NM^2(\sum_{i=1}^{I}D^{G_i}+\sum_{t=1}^{T}\sum_{j=1}^{J^t}D^{H_j^t}+D^{\text{out}}))$.

Are deeper architectures needed? The description of the model is presented for a 2-layer DGP; however, this framework is general and can be applied to a DGP with an arbitrary depth. Looking at the ELBO in (3.14), the contribution of the first layer appears in the KL terms involving the $\bar{\mathbf{G}}_i$'s and the $\bar{\mathbf{H}}_j^t$'s. Therefore, a

layer of this structure can, in principle, be added at any level of a DGP cascade by simply adapting the ELBO to include these terms. Since the sparse approximation and the inference framework in Salimbeni and Deisenroth [2017] do not require the computation of the full covariance within the layers, it is possible to propagate a Monte Carlo sample through this layer structure at any point in the cascade without the need to compute inter-task covariances. This enables approximate evaluation of the expected log-likelihood term in (3.14).

It is worth noting, however, that increasing the depth this way adds extra complexity to the model and potentially violates the generative assumption of the non-linear combination of shared and task-specific processes. Hence, it is recommended that such an extension should only be considered when the structure of the problem requires it.

3.3.2 Model Specification

So far, the description of the modelling approach has been in an unspecified setting. One can use the proposed framework as a base to construct a plethora of multi-task DGP models. This section discusses three straightforward instantiations of this framework.

The multiprocess Multi-task DGP (mMDGP) follows the formulation in Section 3.3.1 closely. The model consists of a two layer GP with the inner layer (latent space) split into a set of independent shared processes and independent task specific processes. The outputs of those processes are concatenated and fed into the top layer which constitute independent task specific GPs. The GPs in the top layer admit Automatic Relevance Determination (ARD) kernels [Rasmussen and Williams, 2006]

$$k(\lambda, \lambda') = \sigma^2 \kappa\Big(\sum_{i=1}^{D^\lambda} \omega_k s(\lambda_i - \lambda_i')\Big), \tag{3.15}$$

where $\kappa(\cdot)$ is a positive definite function, $s(\cdot)$ is a distance function and σ and ω_i are the kernel parameters, which are learned by maximising the ELBO as a surrogate to the model's marginal likelihood.

The use of the ARD kernel in the top layer enables each output process to weigh the information from the latent processes differently, thus balancing the use of shared information and task-specific information (see Fig. 3.5 in Section 3.5.2 for illustration). This can guard against some types of negative transfer. For instance, if a contaminating task is present in the dataset, the model can, in principle, learn

ARD weights such that all its information is absorbed in the task-specific components.

The shared Multi-task DGP (sMDGP) consists of a 2-layer DGP where all the latent processes are shared between the tasks with no task-specific processes. It also employs an ARD kernel in the top layer. The role of ARD in this version is akin to Manifold Relevance Determination [Damianou et al., 2012], where the ARD weights determine a soft segmentation of the latent space.

The Coregionalised Multi-task DGP (cMDGP) is similar to sMDGP, with a single shared latent space. The middle layer forms a collection of coregionalised GPs [Goovaerts, 1997; Alvarez et al., 2011], i.e. unlike sMDGP the outputs of the processes in the latent space are not concatenated but linearly mixed with different mixing coefficients for different tasks. In addition, the top layer is set to a single coregionalised GP that outputs all the tasks. This formulation is similar to Alaa and van der Schaar [2017], used for survival analysis with competing risks. However, the inference scheme presented in Alaa and van der Schaar [2017] is only applicable with certain covariance functions and is not suitable for the *asymmetric* multi-task learning case (where only a subset of the outputs is observed for a single input location). Conversely, using the doubly stochastic formulation for the ELBO allows using any valid covariance function and enables the inference on partially observed outputs.

3.4 Related Work

There is a rich literature on multi-task learning in GP models. Most GP multi-task models correlate the outputs by mixing a set of independent processes with a different set of coefficients for each output. Examples of such models include: the semi-parametric latent factor model (SLFM) [Teh et al., 2005], intrinsic coregionalisation model (ICM) [Bonilla et al., 2007; Skolidis and Sanguinetti, 2011] and the linear model for coregionalisation (LMC) [Goovaerts, 1997; Alvarez et al., 2011]. Nguyen and Bonilla [2014] extend SLFM to handle dependent tasks in very large datasets using the framework in Hensman et al. [2013]. In Titsias and Lazaro-Gredilla [2011], the authors give this type of model a Bayesian treatment by placing a spike-and-slab prior on the mixing coefficients, and in Aglietti et al. [2019], the authors treat the coefficients as stochastic processes indexed by task descriptors in a spatial Cox process setting.

More complex models that can handle complex relationships between the outputs are formulated in Wilson et al. [2012] and Nguyen and Bonilla [2013] introducing mixing weights with input dependencies, and in Boyle and Frean [2004] and Alvarez and Lawrence [2009] by convolving processes.

In the DGP literature, there are two notable formulations for the multi-task setting. Kandemir [2015] proposes to linearly combine the hidden layers of different DGP models trained on different tasks, inducing information transfer between the models. Alaa and van der Schaar [2017] uses the ICM kernel in the hidden and output layers of the DGPs for multi-task learning for survival analysis with competing risks. The framework in this chapter is a general formulation of the models above, replacing the linear mixing of latent processes with another GP, thus allowing to model more complex relationships between the tasks. The work of Requeima et al. [2019] on the Gaussian Process Autoregressive Regression (GPAR) model offers an alternative formulation to non-linear multi-output learning in GPs, This model assumes an inherent ordering of the outputs and places independent GP priors on them, where the current GP output is concatenated with the inputs of the proceeding GP at the observed location. This model can be interpreted as a particular DGP structure with skip connections. However, in the multi-task setting, this model assumes an inherent ordering of the tasks and that the dataset is closed downwards, i.e. for task t, all the previous $t-1$ outputs are observed for each input location in the dataset; otherwise this can be remedied with ad-hoc imputation of the missing data.

A similar idea to multi-task learning is multi-view learning, where multiple views of the data are combined into a single latent representation. In the Gaussian process literature, Manifold Relevance Determination (MRD) [Damianou et al., 2012] is a latent variable model for multi-view learning, where the latent space is softly separated into a component that is shared between all of the views and private components that are view-specific. The work in this chapter attempts the inverse mapping, i.e. disentangling multiple representations from a common feature space. In sMDGP, the idea of the soft separation in the latent space is used, and in mMDGP a hard separation is encoded between private components and task-specific components of the latent space.

The following explores briefly the connection of the proposed model to some of the above models and other models in the wider multi-task learning literature.

3.4.1 Linear Process Mixing

To make the relationship of the proposed model to other GP-based models more explicit, consider the following construction of a multi-output function $f(x)$ of T outputs, where the tth output is

$$f^t(\mathbf{x}) = \sum_{i=1}^{I} w_i^t g_j(\mathbf{x}) + \sum_{j=i}^{J} h_j^t(\mathbf{x}), \tag{3.16}$$

$\{w_i^t\}_{i=1}^I$ is a set of weights associated with output t, and the $g_j(\cdot)$'s and $h_j^t(\cdot)$'s are basis functions that are shared and output specific, respectively. In the probabilistic literature, the set of basis functions are modelled as independent GPs with their choice of covariance function determining the particular model, e.g. setting independent GP priors on the basis functions with different covariance functions recovers the Semi-parametric Latent Factor Model of Teh et al. [2005]. Since the output processes are a sum of GPs, it follows that they themselves are also GPs with a tractable cross-covariance that depends on the weights on the shared basis functions.

One can generalise the construction in (3.16) to

$$f^t(\mathbf{x}) = \varphi^t(g_1(\mathbf{x}), \ldots, g_I(\mathbf{x}), h_1^t(\mathbf{x}), \ldots, h_J^t(\mathbf{x})), \tag{3.17}$$

where $\varphi^t(\cdot)$ is an arbitrary function. Depending on the choice of $\varphi^t(\cdot)$, this construction combines the basis functions non-linearly; for instance, allowing interactions between the shared and output specific processes. Treating this construction probabilistically, one can place GP priors on the basis function as well as the warping function $\varphi^t(\cdot)$, which recovers the proposed model. It is important to note that due to the non-linearity of $\varphi^t(\cdot)$, the resulting output processes are no longer Gaussian and their cross-covariance cannot be characterised in terms of the shared basis functions only. As a side note, choosing a GP with linear covariance function as a prior on $\varphi^t(\cdot)$ recovers the linear model described in (3.16).

3.4.2 Process Convolution

One can formulate a GP as the convolution of a base process with a smoothing kernel, i.e.

$$f(\mathbf{z}) = \int_{\chi} G(\mathbf{x} - \mathbf{z}) u(\mathbf{z}) d\mathbf{z}, \tag{3.18}$$

where the smoothing kernel $G(\cdot)$ is square integrable (i.e. $\int G^2(x-u)du < \infty$) and the base process $u(\cdot)$ is a white noise process, or more generally any random process (although if non-Gaussian then the resulting convolution process is not a GP) [Calder and Cressie, 2007]. In the multi-output process convolution literature, different outputs are assumed to share the same base process u, with different smoothing kernels for each output. Hence, the cross-covariance between the output processes can be derived tractably as convolution is a linear operation.

Using (3.18) and conditioned on the latent processes, one can write the top layer of the presented model as

$$f^t(\mathbf{x}) = \int_{\mathcal{X}} G^t(\mathbf{g} - \mathbf{z}_g, \mathbf{h}^t - \mathbf{z}_h) u^t(\mathbf{z}) d\mathbf{z}, \tag{3.19}$$

assuming a separable smoothing kernel $G^t(\mathbf{g} - \mathbf{z}_g, \mathbf{h}^t - \mathbf{z}_h) = G^t(\mathbf{g} - \mathbf{z}_g)G^t(\mathbf{h}^t - \mathbf{z}_h)$ (which is the case for processes with ARD kernels), one can rewrite (3.19) as

$$\begin{aligned} f^t(\mathbf{x}) &= \int_{\mathcal{X}} G^t(\mathbf{g} - \mathbf{z}_g) G^t(\mathbf{h}^t - \mathbf{z}_h) u^t(\mathbf{z}) d\mathbf{z} \\ &= \int_{\mathcal{X}} G^t(\mathbf{g} - \mathbf{z}_g) v^t(\mathbf{z}_g) d\mathbf{z}_g, \end{aligned} \tag{3.20}$$

where v^t is also random process. Since by construction, g itself is a random (Gaussian) process, one can see that the output correlation is induced in the smoothing kernel rather than the base process. Conditioning the smoothing kernel on a random process is explored in Higdon [1998] and Higdon et al. [1999] to induce non-stationary behaviour in the output process. To the best of the author's knowledge this work is the first instance to consider this conditioning for information sharing behaviour.

3.4.3 Regularisation Methods

The ELBO defined in (3.14) has a similar formulation to the general multi-task objective function in regularisation-based multi-task learning literature [Evgeniou and Pontil, 2004]. A general formulation of which is given by

$$L = \frac{1}{T} \sum_{t=1}^{T} L^t(\theta, \phi; \mathbf{Y}^t) + \Omega(\theta) + \sum_{t=1}^{T} \Psi^t(\phi^t), \tag{3.21}$$

where θ are shared parameters, $\phi = \{\phi^t\}_{t=1}^T$ are task specific parameters and $\{L^t\}_{t=1}^T$, Ω and $\{\Psi^t\}_{t=1}^T$ are the task specific losses, shared regulariser and task specific regularisers respectively. The choice and strength of regularisation controls the information transfer between tasks.

A direct analogy can be drawn from (3.14) to (3.21), where the expected log-likelihood terms correspond to the loss functions and the KL terms correspond to the regularisers, respectively. This forms a new link between the probabilistic and the regularisation points-of-view of multi-task learning and motivates the prospect of choosing alternative prior processes to induce desirable behaviour in the model, e.g. sparsity in the latent space [Argyriou et al., 2007]. Furthermore, one can also frame (3.14) as a Generalised Variational Inference objective [Knoblauch et al., 2019], where the KL terms are viewed as prior regularisers. Changing the form of these regularisers can induce different information sharing behaviour in the approximate predictive posteriors of the tasks. Unfortunately, exploring this point further is beyond the scope of this work.

3.5 Experimental Evaluation

In this section, the three instantiations of the presented framework (described in Section 3.3.2) are tested on three datasets. The three models are are collectively referred to as **MDGP**. For all of the experiments, the chapter's proposal is compared to three other GP-based models: **iDGP**: independent single-task 2-layer DGPs [Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017] trained on each task individually; **iGP**: a independent single-task variational sparse Gaussian process [Hensman et al., 2013] trained on each task individually; **cGP**: a multi-task variational sparse Gaussian process [Nguyen and Bonilla, 2014; Bonilla et al., 2007]. This model uses the intrinsic coregionalisation kernel (ICM) [Goovaerts, 1997] for multi-task learning.

Furthermore, further comparisons are introduced to standard multi-task neural network of similar complexity [Bakker and Heskes, 2003]: **mANN2** and **mANN3**: two and three layer multi-head feed-forward networks respectively, with uncertainty computed by MC Dropout [Gal and Ghahramani, 2016]; **mBNN2** and **mBNN3**: two and three layer multi-head variational Bayesian neural networks respectively.

All the GP models use the Matérn-5/2 kernel and the same number of inducing points per task. The variational parameters and hyperparameters (of the kernels -including ARD weights- and likelihoods) are jointly learnt through the maximisation of the models' ELBO. The Adam Optimiser [Kingma and Ba, 2015] is used on the neural network and DGP-based models, and L-BFGS [Nocedal and Wright, 2000] is used for the shallow GP-based models (except when mini-batching occurs where Adam is used again). See Appendix A.1 for a more detailed description of the experimental setup.

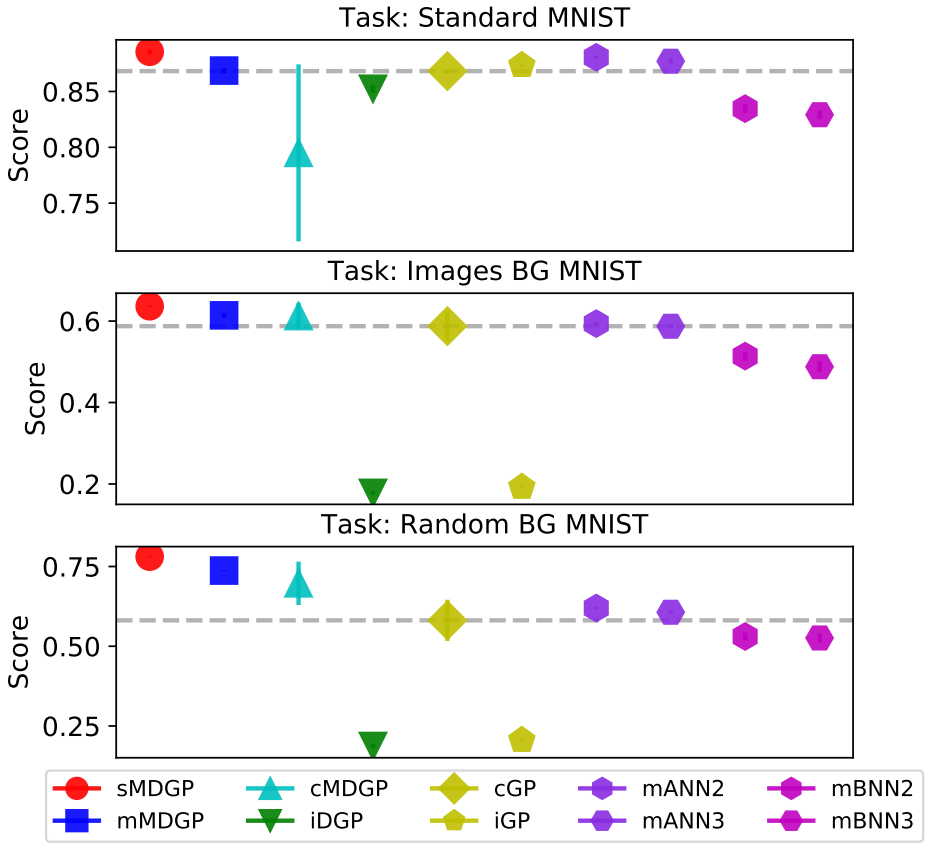


Figure 3.3: Average classification accuracy and its standard error on the MNIST variations experiments. The dashed line highlights the multi-task GP baseline. Higher is better.

3.5.1 MNIST Variations

This experiment uses the MNIST digits dataset [Lecun et al., 1998], as well as two other variations. The first variation, **Images Background (BG) MNIST**, replaces the background with randomly extracted patches from 20 black and white images. The second variation, termed **Random Background (BG) MNIST**, replaces the black background for the digits with random noise [3]. These two variations are more challenging for classification due to the presence of non-monochromatic

[3] https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits

backgrounds.

1000 images are sampled from the training split each of the three datasets. Hence, a total of 3000 training data points for 3 multi-class classification tasks are available. The average accuracy and its standard error are reported on the standard test split for the datasets over 10 runs. The purpose of this experiment is to test the ability of the model to transfer information from a relatively easy task, standard MNIST classification, to more challenging tasks, i.e. images BG MNIST and random BG MNIST.

Figure 3.3 suggests that MDGP models are successfully able to use common information shared between the three tasks to improve the learning performance on the difficult tasks (image BG MNIST and random BG MNIST). However, one can see that cMDGP suffers from negative transfer on the first task, due to its inability to weigh the contributions of the latent processes differently for each task. In contrast, mMDGP and sMDGP do not suffer from this problem, highlighting their robustness to this type of negative transfer from the use of ARD weights on the warping layer.

MDGP models achieve a considerable performance gain on the third task (random BG MNIST) compared to all the other models, by a factor of almost 50%. This task is the most challenging out of the three as the background here is not structured. This improvement highlights the importance of the non-linear mixing of the latent processes to deal with complex task structures.

To illustrate the learning behaviour of mMDGP, one can examine the inducing inputs from the inner layer. As specified in Section 3.3.2, mMDGP has an inner layer with hard separation between the task-specific and shared processes. This modelling choice was made expecting the task-specific part to learn private task information, while the shared part would learn global information shared across the tasks. Figure 3.4, obtained from one of the experiment runs, confirms this assumption. On the left one can see the initial value of an inducing location before optimisation. The optimisation transforms the initialisation in the task-specific component and the shared component of the inner layer in different ways. The task-specific component encodes the background information specific to the task, blurring the digit. The shared component, on the other hand, retains the structure of the digit removing background information.

Table 3.1: Average NLPP scores on the SARCOS dataset over 7 tasks. The figures presented are the mean score (and standard error) over 10 runs. The lowest statistically significant scores based on a Wilcoxon test are presented in boldface. Lower is better.

	Number of Training Inputs					
	100	200	500	1000	2000	5000
sMDGP	1.47(0.26)	**0.98**(**0.11**)	0.74(0.05)	**0.15**(**0.1**)	**−0.01**(**0.1**)	**−0.16**(**0.1**)
mMDGP	**1.34**(**0.21**)	1.05(0.10)	**0.67**(**0.12**)	**0.16**(**0.10**)	**0.01**(**0.11**)	−0.12(0.10)
cMDGP	**1.32**(**0.21**)	**0.95**(**0.1**)	0.72(0.06)	0.24(0.11)	0.05(0.10)	−0.09(0.10)
iDGP	**1.32**(**0.17**)	**0.99**(**0.11**)	0.75(0.06)	0.64(0.04)	0.36(0.09)	0.02(0.08)
cGP	1.51(0.06)	1.42(0.04)	1.35(0.04)	1.27(0.08)	0.12(0.11)	0.02(0.11)
iGP	1.43(0.03)	1.38(0.02)	1.34(0.03)	1.28(0.07)	1.17(0.09)	0.91(0.12)
mANN2	28.22(5.39)	10.56(1.69)	3.20(0.53)	1.66(0.33)	1.26(0.30)	1.16(0.31)
mANN3	21.72(4.20)	9.53(1.70)	3.14(0.51)	1.92(0.33)	1.48(0.34)	1.42(0.33)
mBNN2	1.58(0.22)	1.11(0.20)	**0.55**(**0.09**)	0.31(0.06)	0.16(0.04)	0.05(0.03)
mBNN3	1.71(0.28)	1.18(0.18)	**0.63**(**0.09**)	0.37(0.06)	0.21(0.04)	0.09(0.03)

3.5.2 SARCOS Robot Inverse Dynamics

This experiment considers the SARCOS regression dataset relating to the inverse dynamics problem for a seven degrees-of-freedom anthropomorphic robot arm [Vijayakumar et al., 2002; Rasmussen and Williams, 2006]. The dataset consists of 44,484 training observations with 21 input variables and 7 outputs (tasks). Additionally, there are 4,449 testing examples with all the 7 outputs available.

The experimental procedure starts by sampling N training data points from the training set, for N in $\{100, 200, 500, 1000, 2000, 5000\}$. This constitutes the training set for one experimental run. For each of the N sampled points select 1 of the 7 outputs uniformly at random. Split the experimental training set into 7 according to which joint the label corresponds to. These 7 splits constitute 7 tasks. The models are trained on the training set after standardising the features and the targets and test for all 7 labels on the test set. The negative log predictive probability (NLPP) and the root mean squared error (RMSE) are reported, aggregated across the 7 tasks over averaged 10 runs (with standard errors). The NLPP score is used to assess a model's ability to quantify uncertainty as it incorporates the mean prediction as well as the predictive variance, while the RMSE score is used to measure the quality of the mean predictions only. The results for this experiment are presented in Table 3.1 and Table A.2 (in the appendix).

Looking at the NLPP scores in Table 3.1, one observes that for all data regimes MDGP models perform well, outperforming the shallow models for all configurations. As the dataset size increases the MDGP models are able to outperform

the rest by a wider margin. The gain in performance of cMDGP is not as pronounced as for mMDGP and sMDGP, highlighting the importance of task-specific warping functions with ARD kernels for this problem.

To take a closer look at the effect of the ARD kernel, Fig. 3.5 shows a Hinton diagram of the ARD weights for one run of mMDGP on 5000 datapoints. The hard separation of latent processes is highlighted in the colour scheme. One can see that the model assigns different weights on the random processes for different tasks. It also on average assigns more weight to the shared processes than the task-specific processes, indicating that the model utilises the shared representation. A similar diagram for sMDGP is shown Fig. A.1 in the appendix.

3.5.3 FAIMS Diabetes Diagnosis

Field Asymmetric Ion Mobility Spectrometry (FAIMS) is a method for detecting Volatile Organic Compounds (VOCs) which for example contribute to a given odour. VOCs are known to carry information on a range of disease states in humans, and technologies such as FAIMS can be used to cheaply and non-invasively diagnose a patient's disease from a simple biological sample such as urine [Covington et al., 2015].

The author has access to data from a case-control study of 125 patients who have been tested for diabetes. 48 out of 125 have been found to have diabetes (the disease group), while the rest are disease-free (the control group). The data consists of three experimental runs per patient, corresponding to sequential FAIMS analyses on the same urine sample. These are expected to contain similar but not identical signals, as different VOCs evaporate at different rates, meaning the overall VOC signal varies over time. Each experimental run is treated as a task, i.e. three binary classification tasks in total. By doing so, one aims to share statistical strength between runs to improve the accuracy of prediction and overcome the scarcity of training data.

Due to the large size of the feature space and its sparsity, the Sparse Principal Component Analysis decomposition [Hastie et al., 2009; Mairal et al., 2009] is used on the features selecting the first 20 principal components. This pipeline is standard for this type of datasets [Martinez-Vernon et al., 2018]. A 10-fold cross-validation is performed on 70:30 train-test splits. The value of the area under the Receiver Operating Characteristic Curve (ROC-AUC) is reported for each task averaged across the folds, as well as the average ROC-AUC for all tasks and runs and their standard errors. A single task Random Forest classifier [Hastie et al., 2009] as a is included as

Table 3.2: ROC-AUC results on the FAIMS dataset averaged over 10 runs. The figures in parentheses are the standard errors. Higher is better.

	ROC-AUC: mean(standard error)			
Model	Task 1	Task 2	Task 3	All
sMDGP	0.73(0.03)	0.80(0.02)	**0.8(0.02)**	**0.78(0.02)**
mMDGP	**0.75(0.03)**	**0.81(0.02)**	**0.8(0.02)**	**0.78(0.02)**
cMDGP	0.72(0.02)	**0.81(0.02)**	0.79(0.02)	0.77(0.01)
iDGP	0.62(0.02)	0.75(0.02)	0.76(0.03)	0.71(0.02)
cGP	0.58(0.03)	0.64(0.02)	0.66(0.02)	0.63(0.02)
iGP	0.57(0.02)	0.56(0.03)	0.61(0.04)	0.58(0.02)
iRF	0.71(0.03)	0.71(0.02)	0.72(0.02)	0.71(0.01)
mANN2	0.54(0.03)	0.54(0.02)	0.50(0.03)	0.53(0.02)
mANN3	0.54(0.01)	0.52(0.03)	0.53(0.03)	0.53(0.02)
mBNN2	0.47(0.02)	0.48(0.02)	0.43(0.03)	0.46(0.01)
mBNN3	0.44(0.04)	0.48(0.04)	0.45(0.02)	0.45(0.02)

a baseline, which has been shown to perform well on this problem [Martinez-Vernon et al., 2018]. The results of this experiment are shown in Table 3.2.

One observes that MDGP models outperform the other GP-based models. They also significantly outperform the iRF baseline on the second task. Interestingly, both iDGP and cGP outperform iGP on average, indicating that representation learning as well as information sharing are important for this problem. MDGP models are able to leverage information transfer and nonlinear projection to perform well on this type of problems compared to the other tested models.

3.6 Concluding Remarks

This chapter introduced a new framework for multi-task learning for DGPs based on the composition of GP modules, presenting a general formulation of a multi-process layer structure that can learn shared information between the tasks, as well as task-specific information. Inference in this model is possible through the multi-task extension to the doubly stochastic variational approximation, with practically fast inference in CPU time (Table A.3 in Appendix A.2.2). Three instantiations of this framework with different properties were proposed. The experimental results showed that the multi-task formulation presented in this paper is indeed effective in capturing non-linear task relationships and improves the learning performance on multi-task problems.

The presented framework demonstrated the effectiveness of compositional structures for complex learning scenarios such as multi-task learning. The impor-

tance of compositionality is seen in the ease in which the complex modelling assumptions where specified, as well as in the straightforward extension of standard inference procedures to accommodate the increased complexity of the model. An interesting future research direction is to assess the viability of other inference algorithms such as the Random Feature Expansions of Cutajar et al. [2017] for this type of GP-based compositional models.

This model was implemented with GPflow [Matthews et al., 2017] and is publicly available on GitHub. [4]

[4] https://github.com/aboustati/dgplib

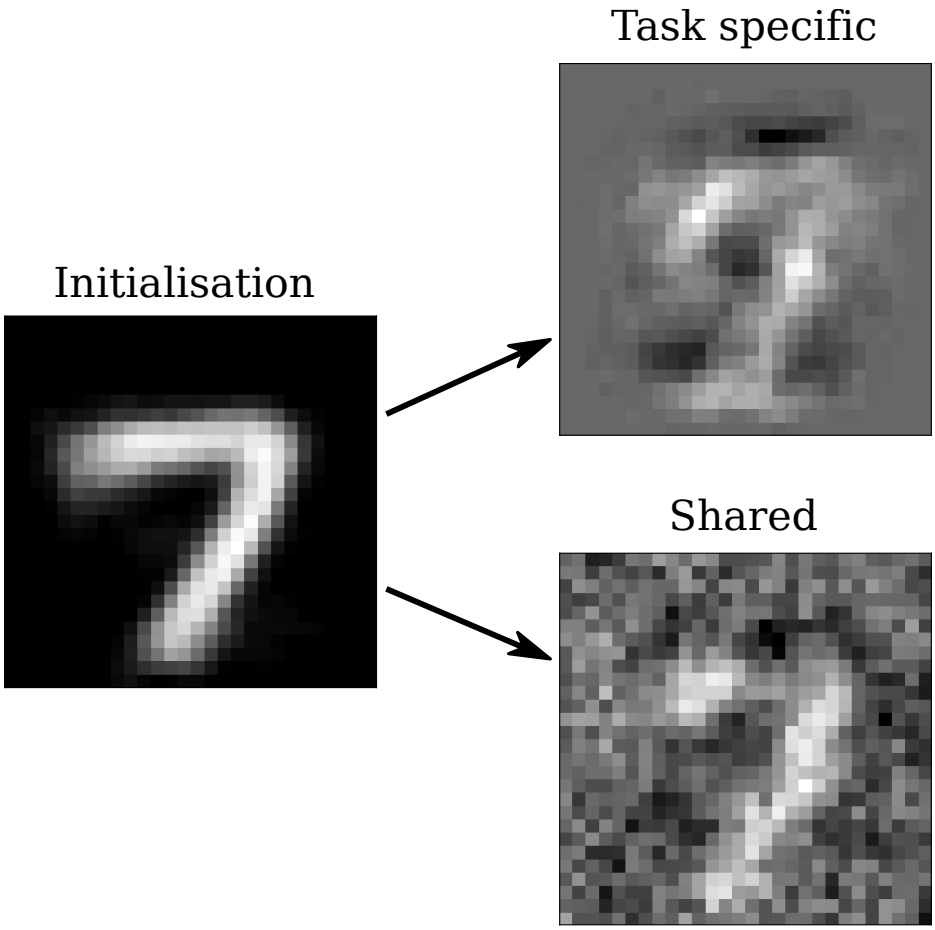


Figure 3.4: Example of the learning behaviour of mMDGP. The image on the left is the initial value for one of the inducing locations. The top right image is the learned inducing location for the task-specific layer for the Standard MNIST task. The bottom right image is the learned inducing location for the shared layer. The background information is encoded in the task-specific representation, where the digit is blurred The shared information contains the digit on an unstructured background.

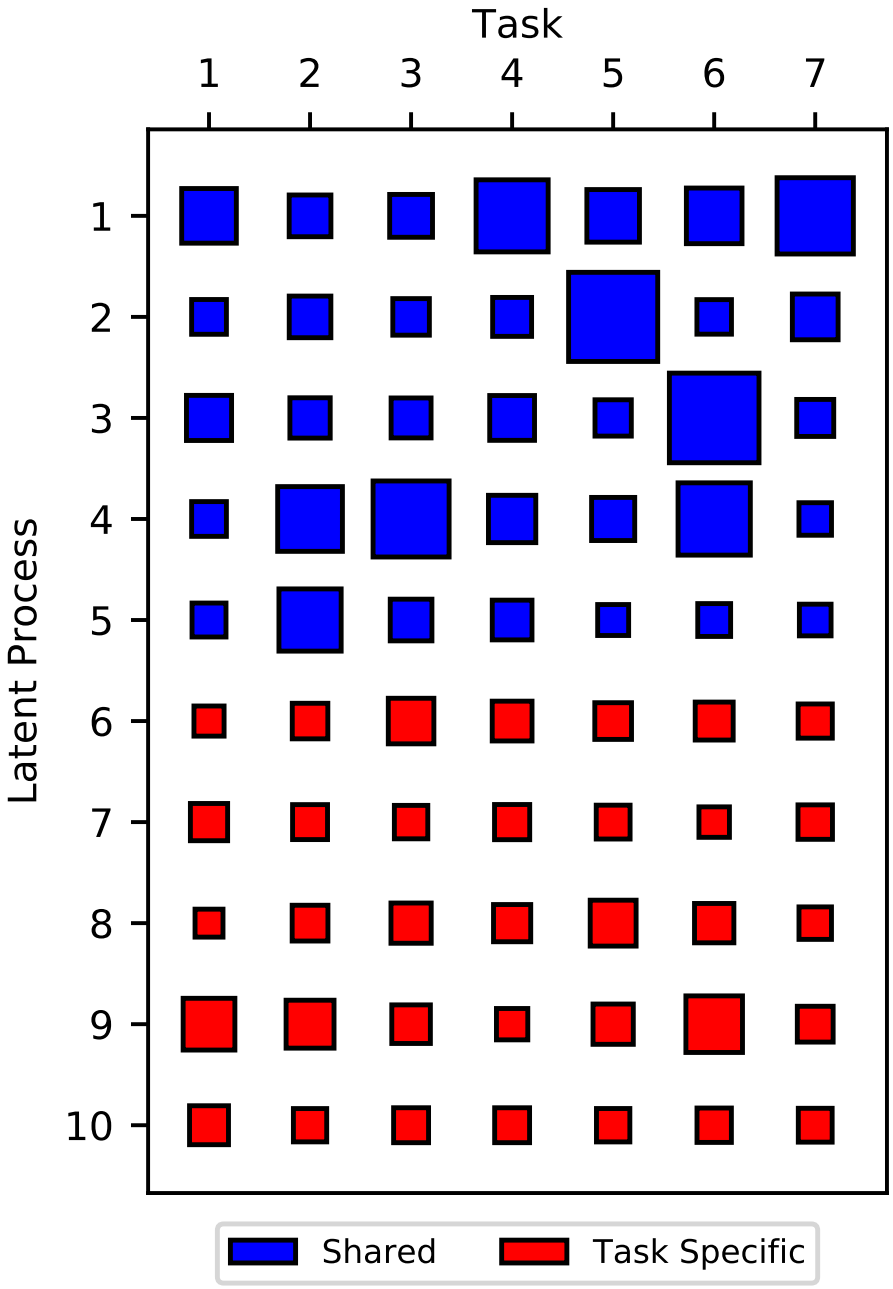


Figure 3.5: Hinton diagram showing the ARD weights for one of the MDGP runs on Sarcos with 5000 training datapoints. Blocks in blue are shared latent processes, while blocks in red are task-specific. This indicates that the model assigns different weights for the latent processes for each task.

CHAPTER 4

Amortised Variance Reduction

Inference in modern probabilistic models requires the estimation of quantities that are based on random variables. This usually introduces extra variance in the inference procedure leading to training instability. This is particularly significant for compositional models since they constitute multiple probabilistic modules that are evaluated jointly. Hence, to improve the stability of the inference in the presence of variance, it is important to examine variance reduction methods and their application to inference in probabilistic models.

This chapter studies the problem of training stability applied to Variational Inference (VI) procedures. VI is the go-to choice for inference in complex compositional models [Damianou and Lawrence, 2013; Tran et al., 2016] due to its scalability (in terms of both dataset size and model complexity) and computational efficiency. However, when applied to these types of models, it can suffer from instability due to the high variance induced by the complexity of the model. This chapter utilises the idea of control variates (see Section 2.7.1 for background) to reduce the variance in the stochastic optimisation problem in VI, thus improving the overall stability of this inference procedure.

4.1 Motivation

In general, many machine learning tasks can be formulated as an optimisation problem, where the model parameters $\boldsymbol{\theta}$ are inferred by optimising an objective function $\mathcal{L} = \sum_{n=1}^{N} \ell_n(\boldsymbol{\theta})$ which is a finite sum over contributions from individual data points n. In the specific case of VI, this type of objectives contain analytically intractable expectation terms, $\ell_n(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f_n(\boldsymbol{\epsilon}, \boldsymbol{\theta})]$, over a random variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. This is present in most formulations of VI such as Black Box Variational Inference [Ranganath et al., 2014], Variational Auto-Encoders [Kingma and Welling, 2014], or Deep

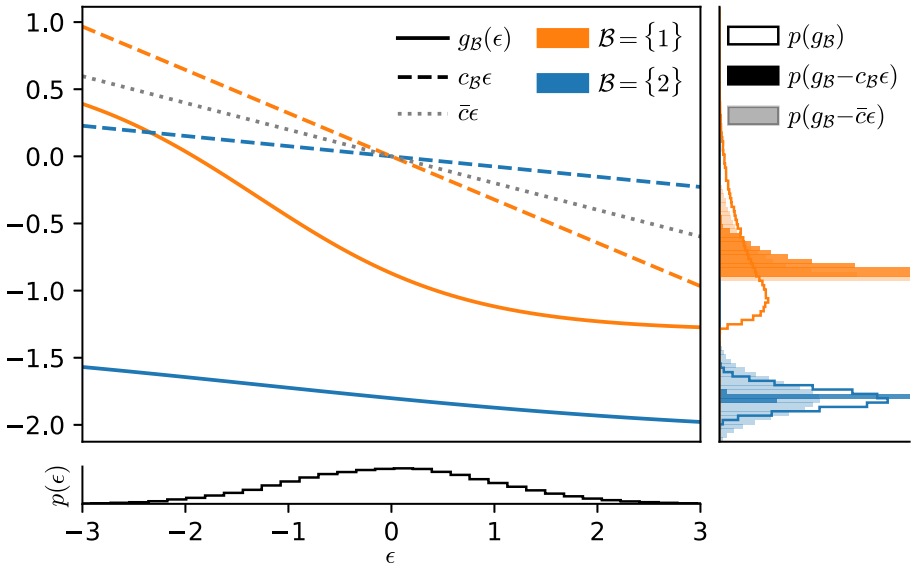


Figure 4.1: In reparameterised variational inference, the gradient is a function of the randomness sample $\epsilon \sim p(\epsilon)$. This relationship $g_{\mathcal{B}}(\epsilon)$ (solid lines) depends on the mini-batch $\mathcal{B}$ (orange vs blue). Here, the linear control variates with batch-dependent coefficients $c_{\mathcal{B}}\epsilon$ (dashed lines) and the best batch-independent control variate $\bar{c}\epsilon$ (dotted grey line) are shown. The right-hand plot shows the distribution of the expectation estimators for each mini-batch: no control variate (outline), batch-independent control variate (shaded), and batch-dependent control variate (filled). The batch-dependent control variate significantly reduces the variance, whereas here the batch-independent control variate increases the variance for the blue mini-batch.

Gaussian Processes [Salimbeni and Deisenroth, 2017].

In practice, such objectives are treated using Monte Carlo (MC) sampling to obtain an unbiased stochastic estimate of the expectation, $\hat{\ell}_n = \frac{1}{S}\sum_{s=1}^{S} f_n(\boldsymbol{\epsilon}_n^{(s)}, \boldsymbol{\theta})$, where $\boldsymbol{\epsilon}_n^{(s)} \sim p(\boldsymbol{\epsilon})$. Stochastic Gradient Descent (SGD) or other gradient-based optimisers are then used on the noisy gradients [Robbins and Monro, 1951] to learn the optimal parameters. For large N, the evaluation of the full sum in $\mathcal{L}$ is often computationally intractable. This is usually addressed by subsampling mini-batches $\mathcal{B} \subset \{1, \ldots, N\}$ of size $|\mathcal{B}|$ from the full data set, introducing additional noise and leading to a doubly stochastic objective function of the form:

$$\hat{\mathcal{L}} := \frac{N}{|\mathcal{B}|S} \sum_{b \in \mathcal{B}} \sum_{s=1}^{S} f_b(\boldsymbol{\epsilon}_b^{(s)}, \boldsymbol{\theta}), \tag{4.1}$$

with $\mathbb{E}[\hat{\mathcal{L}}] = \mathcal{L}$.

The variance of the gradients of $\mathcal{L}$ affects both the convergence rate of the optimisation and how close the optimiser can get to the optimum. This motivates various approaches for reducing either mini-batch variance (*e.g.*, Johnson and Zhang [2013]) or the variance due to MC estimation of the expectation [Ranganath et al., 2014; Roeder et al., 2017]. A common approach for variance reduction for Monte Carlo methods are control variates (see Section 2.7.1), which have recently been adopted in the optimisation literature (see Section 4.5 for details). Their focus is on deriving and applying control variate schemes to MC objectives, specifically in the context of VI.

So far the schemes in the literature do not consider the mini-batching case and do not explicitly take into account how the context of the data point b affects the dependence of f_b on $\boldsymbol{\epsilon}$. This dependence is illustrated in Fig. 4.1 with the example of Bayesian logistic regression. The gradient $g_{\mathcal{B}}(\boldsymbol{\epsilon})$ as a function of the randomness $\boldsymbol{\epsilon}$ of a doubly stochastic objective is shown for two different mini-batches $\mathcal{B}$. In this simplified case, each batch consists of a single context point. The different context points induce different relationships between randomness and the gradient. This means the two gradients correlate differently with the randomness, yielding different optimal control variates. For comparison, a batch-independent control variate is included, which is averaged over all contexts. As shown in the right-hand panel in Fig. 4.1, adapting the control variate to the batch significantly reduces the variance.

This chapter proposes a novel idea for computing control variates that adapt to the context (mini-batch) of the controlled estimators (the gradient). The new formulation takes into account the dependence of the MC estimate on the data by using a recognition network to learn an adaptive control variate coefficient. A low-variance objective is derived, to train the network to approximate the optimal control variate coefficient per batch. Additionally, two computationally cheaper, higher-variance alternatives are proposed to the aforementioned network objective. All control variate objectives re-use the already computed model gradients, and hence do not require extra back-propagation steps.

4.2 Method

This section introduces the mini-batch dependence of the gradients and the control variates and propose learning context-aware control variate coefficients and derives objectives for the control variate coefficients that allow amortisation through a recognition network.

4.3 Background & Notation

Recall that control variates aim to reduce the variance of an unbiased stochastic estimator[1] $\hat{g}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ for an intractable expectation $\mathbb{E}[g(\boldsymbol{\epsilon})]$, where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ is a random variable.

For a control variate $w(\boldsymbol{\epsilon})$ whose expectation W is known analytically, the regression estimator is given by

$$\tilde{g}(\boldsymbol{\epsilon}) = \hat{g}(\boldsymbol{\epsilon}) - c(\hat{w}(\boldsymbol{\epsilon}) - W). \tag{4.2}$$

In this chapter, the term control variate is overridden to refer to $(w(\boldsymbol{\epsilon}) - W)$ rather than $w(\boldsymbol{\epsilon})$. As discussed in Section 2.7.1, computing the optimal control variate coefficient $c^\star$ analytically is not possible and is usually estimated from the optimisation statistics, e.g., running averages [Paisley et al., 2012]. Another option is to pre-specify a fixed c [Miller et al., 2017; Grathwohl et al., 2018]. However, neither option is convincing for the doubly stochastic case. The first option has very a high variance due to the presence of mini-batch stochasticity in addition to sampling stochasticity. The second option is sub-optimal as fixing an arbitrary value for c does not guarantee the optimal variance reduction of Eq. (2.98).

In the next section, c is treated as a context-dependent adaptive parameter that is learned throughout the optimisation. This addresses the drawbacks in the two options above, providing a more stable mechanism to approximate this coefficient in the doubly stochastic setting, leading to optimal variance reduction.

4.3.1 Controlling Mini-batch Gradients

Gradient-based optimisation requires the derivatives of the objective (4.1) with respect to the model parameters $\{\theta_p\}_{p=1}^P$. The estimated gradient contains a sum over mini-batch elements b,

$$\frac{\partial \hat{\mathcal{L}}}{\partial \theta_p} \propto \sum_{b \in \mathcal{B}} \frac{\partial f_b}{\partial \theta_p}(\boldsymbol{\epsilon}_b, \boldsymbol{\theta}) = \sum_{b \in \mathcal{B}} \hat{g}_{bp}(\boldsymbol{\epsilon}_b) =: \hat{G}_p, \tag{4.3}$$

where $S = 1$ is chosen to simplify the notation; however, the extension to multiple MC samples is straightforward. Note that $\mathcal{B}$ is a random subset of $\{1, \ldots, N\}$, i.e., b are indices into the full data set, and each term gets its own realisation $\boldsymbol{\epsilon}_b$ of

[1] The $\hat{}$ symbol (as well as $\tilde{}$) on top of functions of random variables are used to denote the estimate of this function obtained by evaluating the relevant estimator. In the following the dependence on θ is dropped to lighten the notation.

the randomness. The aim is to improve the optimisation performance by reducing the variance of this gradient. As demonstrated in Fig. 4.1, each partial gradient estimator $\hat{g}_{bp}(\boldsymbol{\epsilon}_b)$ may have a different dependence on the randomness. To account for this, separate control variates for each term (data point) in the sum in (4.3) are introduced. For a single partial gradient, the controlled gradient estimator is defined as

$$\tilde{g}_{bp}(\boldsymbol{\epsilon}_b) := \hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^{\intercal}\hat{\mathbf{w}}(\boldsymbol{\epsilon}_b). \tag{4.4}$$

Here and in the following sub-section, the analytic expectation is subsumed into the definition of the control variate such that $\hat{\mathbf{w}}(\boldsymbol{\epsilon})$ already has zero mean. For simplicity, the same type of control variate is used for all parameters; however, in principle, different $\hat{\mathbf{w}}_p$ per parameter θ_p could be defined. In both cases, the coefficients $\mathbf{c}_{bp}$ are per-parameter. In general, the mapping $\hat{\mathbf{w}}(\boldsymbol{\epsilon})$ may have a different number of components than the randomness $\boldsymbol{\epsilon}$ itself; however, for simplicity, both $\boldsymbol{\epsilon}$ and $\hat{\mathbf{w}}(\boldsymbol{\epsilon})$ are assumed to be D-dimensional. Note that $\hat{\mathbf{w}}(\cdot)$ does not depend on the batch element b; the dependence is captured in the coefficients $\mathbf{c}_{bp}$, which is a vector of length D for each index pair b, p.

Specifying the problem this way allows for explicitly selecting a separate control variate coefficient per data point. Under this setting, the new estimator for the gradient is

$$\tilde{G}_p = \sum_{b\in\mathcal{B}} \tilde{g}_{bp}(\boldsymbol{\epsilon}_b) = \sum_{b\in\mathcal{B}} (\hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^{\intercal}\hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)). \tag{4.5}$$

The control variate coefficients $\mathbf{c}_{bp}$ can be set to optimally reduce the variance of $\tilde{G}_p$ by solving

$$\min_{\mathbf{C}} \operatorname{Tr}(\operatorname{Cov}[\tilde{\mathbf{G}}]), \tag{4.6}$$

where $\mathbf{C}$ is the collection of $\mathbf{c}_{bp}$ and has shape $N \times P \times D$, as separate coefficients are needed for all N data points.

Computing and storing these can be computationally prohibitive for large data sets. However, one can amortise the cost of this computation by using a recognition network $r_{\boldsymbol{\phi}} : \mathcal{Y} \to \mathbb{R}^{P\times D}$ that outputs the coefficients for each batch element throughout the optimisation, where

$$\mathbf{c}_{bp} = [r_{\boldsymbol{\phi}}(y_b)]_p \tag{4.7}$$

is a vector of dimension D and $y_b \in \mathcal{Y}$ are context points (*e.g.*feature vector and target for the bth data point in a supervised learning problem) and $\boldsymbol{\phi}$ are the recognition network parameters.

In practice, the recognition network outputs the control variate coefficients

per data point, which are then aggregated per batch. This ensures permutation invariance to the order of data points in the batch and allows for the randomisation of batch elements throughout the optimisation procedure.

As the control variate only adds zero-expectation terms to the gradients of the optimisation objective, the minima of the objective remain unchanged. This means that the extra parameters of the recognition network will *not* lead to overfitting. A theoretical analysis of the convergence is provided in Appendix B.1.

4.3.2 Training the Recognition Network

Intuitively, the recognition network $r_{\boldsymbol{\phi}}(\cdot)$ is required to output coefficients that minimise the variance of the controlled gradient estimator (4.5). This gives the training objective for the parameters $\boldsymbol{\phi}$:

$$\min_{\boldsymbol{\phi}} \operatorname{Tr}\left(\operatorname{Cov}[\tilde{\mathbf{G}}]\right) = \min_{\boldsymbol{\phi}} \sum_{p=1}^{P} \operatorname{Var}[\tilde{G}_p]. \tag{4.8}$$

The pth term in the sum in (4.8) is

$$\begin{aligned}
\operatorname{Var}[\tilde{G}_p] &= \operatorname{Var}\Big[\sum_{b\in\mathcal{B}} \left(\hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right)\Big] \\
&= \sum_{b\in\mathcal{B}} \operatorname{Var}\left[\hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] \\
&= \sum_{b\in\mathcal{B}} \Big(\operatorname{Var}\left[\hat{g}_{bp}(\boldsymbol{\epsilon}_b)\right] + \operatorname{Var}\left[\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] - 2\operatorname{Cov}\left[\hat{g}_{bp}(\boldsymbol{\epsilon}_b), \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right]\Big) \\
&= \text{const} + \sum_{b\in\mathcal{B}} \Big(\mathbb{E}[(\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2] - 2\mathbb{E}[(\hat{g}_{bp}(\boldsymbol{\epsilon}_b))(\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))]\Big).
\end{aligned} \tag{4.9}$$

The terms that do not contain $\mathbf{c}_{bp}$ and hence do not give gradients for $\boldsymbol{\phi}$ are discarded. For most problems, the expectations are intractable, and are estimated with MC sampling. Define

$$\tilde{V}_p := \sum_{b\in\mathcal{B}} \left((\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2 - 2(\hat{g}_{bp}(\boldsymbol{\epsilon}_b))(\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))\right). \tag{4.10}$$

One can now learn the optimal recognition network parameters $\boldsymbol{\phi}$ using SGD (or variants) on

$$\min_{\boldsymbol{\phi}} \sum_{p=1}^{P} \tilde{V}_p. \tag{4.11}$$

To learn the parameters $\boldsymbol{\phi}$, one needs to compute gradients of $\sum_p \tilde{V}_p$. Examining the chain rule around the outputs of the recognition network, $\frac{\partial \tilde{V}_p}{\partial c_{bpd}} \frac{\partial c_{bpd}}{\partial \boldsymbol{\phi}}$, the second term is computed by backpropagation through the network, and the cost of computing the first term depends on the form of the estimator $\tilde{V}_p$.

The recognition network objective in (4.10) requires the partial gradients *per data point* $\hat{g}_{bp}(\boldsymbol{\epsilon}_b)$ of the original objective function; hence, this is referred to as **partial gradients estimator**. In common reverse-mode automatic differentiation libraries such as TensorFlow and PyTorch, it requires $|\mathcal{B}|$ additional backward passes on the model objective, each at least $O(|\mathcal{B}|)$.[2] This becomes prohibitively expensive for large mini-batches. To overcome this limitation in current implementations, the next two sub-sections derive two additional estimators for the recognition network objective that are computationally cheaper, albeit with higher variance.

4.3.2.1 The Gradient Sum Estimator

To avoid the partial gradients in (4.10), one can return to the pth term of the sum in (4.8). Instead of taking the sum out of the variance, the sum over partial gradients is separated from the control variates:

$$\begin{aligned}\mathrm{Var}[\tilde{G}_p] &= \mathrm{Var}\left[\sum_{b\in\mathcal{B}} \hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] \\ &= \mathrm{Var}\left[\hat{G}_p - \sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right]. \end{aligned} \tag{4.12}$$

One can expand the variance of a sum of two terms as

$$\begin{aligned}\mathrm{Var}[\tilde{G}_p] &= \mathrm{Var}[\hat{G}_p] + \mathrm{Var}\left[\sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] - 2\,\mathrm{Cov}\left[\hat{G}_p, \sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] \\ &= \mathrm{const} + \sum_{b\in\mathcal{B}} \Big(\mathbb{E}[(\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2] - 2\mathbb{E}[(\hat{G}_p)(\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))]\Big), \end{aligned} \tag{4.13}$$

and by replacing the expectations with MC estimates, one arrives at a new estimator

$$\tilde{V}_p^{\mathrm{GS}} := \sum_{b\in\mathcal{B}} \left((\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2 - 2(\hat{G}_p)(\mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))\right). \tag{4.14}$$

This estimator is similar in form to the partial gradients estimator, replacing the gradient per data point with the sum over the whole mini-batch. It is called this the **gradient sum** estimator. As it does not require any additional backward passes,

[2] A new PyTorch library, BackPACK [Dangel et al., 2020], can now compute partial gradients with a low computational overhead. This was released after the writing and publication of the contributions of this chapter.

it is much cheaper to compute. One can intuitively see that this estimator has a higher variance than the partial gradients estimator as it additionally includes cross terms that would be zero in expectation.

4.3.2.2 The Squared Difference Estimator

Alternatively, one can continue from (4.12) by expanding the variance into moment expectations:

$$\mathrm{Var}[\tilde{G}_p] = \mathbb{E}\big[(\hat{G}_p - \sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2\big] - \big(\mathbb{E}\big[\hat{G}_p - \sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\big]\big)^2,$$

where the control variate term has no contribution inside the second expectation by definition, and $\mathbb{E}[\hat{G}_p]$ is a constant with respect to the recognition network parameters $\boldsymbol{\phi}$. Evaluating the remaining expectation using MC gives the **squared difference** estimator:

$$\tilde{V}_p^{\mathrm{SD}} := (\hat{G}_p - \sum_{b\in\mathcal{B}} \mathbf{c}_{bp}^\intercal \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2, \tag{4.15}$$

which is also cheap to compute. In contrast to $\tilde{V}_p^{\mathrm{GS}}$, it includes the second moment of $\hat{G}_p$. This is similar to a regression problem that uses $\hat{w}_d(\boldsymbol{\epsilon}_b)$ as zero-mean basis functions to approximate the gradient $\hat{G}_p$.

Both the gradient sum objective (4.13) and the squared difference objective (4.15) are *unbiased estimators* and not approximations of the recognition network objective in (4.6). Compared to the partial gradients objective (4.10), they have higher variance, which is empirically assessed in Appendix B.4.2.3.

The remainder of this section lays out pseudocode for joint optimisation of model and recognition network in Algorithm 4.

4.3.3 Pseudocode

Algorithm 4 describes the proposed amortised control variate scheme in pseudocode. For simplicity, the method is illustrated on Stochastic Gradient Descent (SGD) [Robbins and Monro, 1951]; however, any other gradient-based optimiser could be used instead. The procedure in Algorithm 4 is written in a functional style and represents a single update step for the parameters $\mathbf{theta} = (\theta_1, \ldots, \theta_P)$ of a generic doubly stochastic objective (4.1), $\texttt{objective}(\boldsymbol{\theta}, \mathcal{B}, \boldsymbol{\epsilon})$, where $\mathcal{B}$ is a set of mini-batch indices and $\boldsymbol{\epsilon}$ is a base randomness. The algorithm also updates the parameters $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_Q)$ of the recognition network $r_{\boldsymbol{\phi}}(\cdot)$ that amortises the coefficients of the control variate $\hat{\mathbf{w}}(\cdot)$.

This algorithm can be efficiently executed on two separate compute nodes by interleaving the evaluation of the target objective and its gradients on one node with the evaluation of the recognition network and its gradients on another, as suggested below with (O) and (R). The two nodes only need to sync in line 7. The update of the recognition network parameters can still run in parallel with the next evaluation of the target objective gradients.

Algorithm 4 Stochastic Gradient Descent step with amortised control variate gradients

1: **function** AMORTIZEDCVUPDATE($\{y_n\}$, $\texttt{objective}(\boldsymbol{\theta}, \mathcal{B}, \boldsymbol{\epsilon})$, θ, $\hat{\mathbf{w}}(\cdot)$, $r_{\boldsymbol{\phi}}(\cdot)$, $\boldsymbol{\phi}$, learning rates α & β)
2: $\quad \mathcal{B} \sim p(\mathcal{B})$ ▷ draw mini-batch
3: $\quad \boldsymbol{\epsilon}_b \sim p(\boldsymbol{\epsilon}) \quad \forall b \in \mathcal{B}$ ▷ draw base randomness
4: $\quad$ (O) $\hat{G}_p \leftarrow \frac{\partial}{\partial \theta_p}(\texttt{objective})(\boldsymbol{\theta}, \mathcal{B}, \{\boldsymbol{\epsilon}_b\}) \quad \forall p$ ▷ uncontrolled model gradients
5: $\quad$ (R) $\mathbf{c}_{bp} \leftarrow [r_{\boldsymbol{\phi}}(y_b)]_p \quad \forall b, p$ ▷ evaluate recognition network on context y_b for batch element b
6: $\quad$ (R) $\hat{H}_p \leftarrow \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^{\intercal} \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \quad \forall p$ ▷ control variate
7: $\quad$ (O+R) $\tilde{G}_p \leftarrow \hat{G}_p - \hat{H}_p \quad \forall p$ ▷ controlled model gradients
8: $\quad$ (O) $\theta_p \leftarrow \theta_p - \alpha \tilde{G}_p \quad \forall p$ ▷ SGD updates for model objective parameters [1]
9: $\quad$ (R) $\tilde{V}_p \leftarrow (\hat{G}_p - \hat{H}_p)^2 \quad \forall p$ ▷ recognition network objective [2]
10: $\quad$ (R) $\partial\phi_q \leftarrow \frac{\partial}{\partial \phi_q} \sum_{p=1}^{P} \tilde{V}_p \quad \forall q$ ▷ recognition network gradients
11: $\quad$ (R) $\phi_q \leftarrow \phi_q - \beta \partial\phi_q \quad \forall q$ ▷ SGD updates for recognition network parameters [1]
12: $\quad$ **return** model objective parameters θ, recognition network parameters $\boldsymbol{\phi}$
13: **end function**

4.4 Illustrative Example: A Control Variate for Gaussian Base Randomness

So far the discussion has been general, but to implement a control variate, one needs to specify both the distribution of the base randomness $\boldsymbol{\epsilon}$ and the functional form of

[1] For simplicity SGD updates are shown, but in principle, any other gradient-based optimiser can be used. The same holds for the recognition network updates.

[2] The squared difference objective (4.15) is used for illustrative purposes, but this can also be substituted with the partial gradients objective (4.10) or the gradient sum objective (4.13).

the control variate $\mathbf{w}(\boldsymbol{\epsilon}_n)$. In principle, any functional form for control variates from the literature can be used with this method, *e.g.*, Paisley et al. [2012]; Ranganath et al. [2014]; Miller et al. [2017]. For the sake of simplicity, this section illustrates the proposed methodology on a simpler control variate form for the special case of Gaussian base randomness, which is of direct interest to many applications in VI.

Without loss of generality, assume $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_D)$[3]. This section introduces explicit forms for $\mathbf{w}(\boldsymbol{\epsilon}_n)$ for this case, starting with linear control variates, and then extending the discussion to higher-order polynomials.

4.4.1 Linear Gaussian Control Variates

The simplest control variate is an element-wise linear function of $\boldsymbol{\epsilon}_n$,

$$\mathbf{w}(\boldsymbol{\epsilon}_n) = \boldsymbol{\alpha} + \boldsymbol{\beta} \odot \boldsymbol{\epsilon}_n, \qquad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_D), \tag{4.16}$$

with $\odot$ representing the element-wise product. Its expectation is $\mathbf{W} = \mathbb{E}[\mathbf{w}(\boldsymbol{\epsilon}_n)] = \boldsymbol{\alpha}$, and the control variate simplifies to $\boldsymbol{\beta} \odot \boldsymbol{\epsilon}_n$. $\boldsymbol{\beta}$ can be absorbed into the control variate coefficient $\mathbf{c}_{np}$, which results in the following controlled version of the gradient component p, for data point n:

$$\tilde{g}_{np}(\boldsymbol{\epsilon}_n) = \hat{g}_{np}(\boldsymbol{\epsilon}_n) - \mathbf{c}_{np}^{\intercal} \boldsymbol{\epsilon}_n. \tag{4.17}$$

Intuitively, one can think of control variates of this form as injecting the estimator with information on the linear dependence of the gradient on the noise. To understand this further, examine the first-order Taylor expansion of the gradient component $g_{np}(\boldsymbol{\epsilon}_n)$ around $\boldsymbol{\epsilon}_n = \mathbf{0}$,

$$g_{np}(\boldsymbol{\epsilon}_n) = g_{np}(\mathbf{0}) + \nabla g_{np}(\mathbf{0})^{\intercal} \boldsymbol{\epsilon}_n + O(\boldsymbol{\epsilon}_n^2). \tag{4.18}$$

If the gradient is sufficiently linear with respect to $\boldsymbol{\epsilon}_n$ (i.e., the $O(\boldsymbol{\epsilon}_n^2)$ terms are negligible), and when $\mathbf{c}_{np}$ is a good approximation to the Jacobian at $\mathbf{0}$, the estimator in (4.17) will have low variance.

4.4.2 Higher-order Polynomials

In general, the gradient is unlikely to be linear with respect to the noise, especially for complicated models and objectives. To overcome this, higher-order polynomials can be used to capture some of the non-linear dependence of the gradient on the

[3] In the general case where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, one can simply apply the location-scale reparameterisation $\boldsymbol{\epsilon} = \boldsymbol{\mu} + \mathrm{Cholesky}(\Sigma)\boldsymbol{\epsilon}_0$ with $\boldsymbol{\epsilon}_0 \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_D)$.

noise. Consider

$$\mathbf{w}(\boldsymbol{\epsilon}_n) = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \odot \boldsymbol{\epsilon}_n^k, \tag{4.19}$$

where the kth power is evaluated element-wise. $\mathbf{W}$ can be easily computed and would correspond to the sum of diagonal parts of the first K moment tensors of the multivariate Gaussian distribution, scaled by $\boldsymbol{\alpha}_k$. For instance, for $K = 2$ the control variate is given by

$$\mathbf{w}(\boldsymbol{\epsilon}_n) = \boldsymbol{\alpha}_1 \odot \boldsymbol{\epsilon}_n + \boldsymbol{\alpha}_2 \odot (\boldsymbol{\epsilon}_n^2 - \mathrm{diag}(\mathrm{I}_D)). \tag{4.20}$$

This can again be simplified by absorbing the $\boldsymbol{\alpha}_k$ into the control variate coefficient, with slight adjustments to the controlled gradient estimator, according to the following observation:

Remark 1: A linear combination of control variates is also a valid control variate, *i.e.*, $\tilde{g}_{np}(\boldsymbol{\epsilon}_n) = \hat{g}_{np}(\boldsymbol{\epsilon}_n) - \sum_{k=1}^{K} (\mathbf{c}_{np}^{(k)})^\intercal (\hat{\mathbf{w}}_k(\boldsymbol{\epsilon}) - \mathbf{W}_k)$ is unbiased. By considering each term in (4.20) as a separate control variate, one can write the pth component of the controlled gradient at n as

$$\tilde{g}_{np}(\boldsymbol{\epsilon}_n) = \hat{g}_{np}(\boldsymbol{\epsilon}_n) - (\mathbf{c}_{np}^{(1)})^\intercal \boldsymbol{\epsilon}_n - (\mathbf{c}_{np}^{(2)})^\intercal (\boldsymbol{\epsilon}_n^2 - \mathrm{diag}(\mathrm{I}_D)). \tag{4.21}$$

The same construction trivially extends to $K > 2$.

4.4.3 Brief Discussion

The simple examples of the linear and polynomial control variates presented above illustrate the importance of choosing a good control variate coefficient. For instance, in the linear case in (4.17) the control variate function $\mathbf{w}(\boldsymbol{\epsilon}_n) = \boldsymbol{\epsilon}_n$ does not provide any extra information on the estimator on its own, as it essentially just adds noise to the MC estimate. However, the selection of a good control variate coefficient $\mathbf{c}_n$ for data point n introduces structure to the noise, that contains information about the behaviour of the controlled quantity with respect to the Gaussian noise in the form of the Jacobian in (4.18). Indeed the optimal coefficient $\mathbf{c}_n^*$ for the linear control variate contains the Jacobian term.

4.5 Related Work

Control variates are widely used to reduce the gradient variance of stochastic objectives, mainly motivated by VI. A comprehensive review can be found in Geffner

and Domke [2018]. Some of the relevant work is highlighted here and compared to the contribution in this chapter.

Paisley et al. [2012] first introduce the idea of using control variates to reduce the gradient variance in VI. They propose using a bound on the objective or an approximation of the model as control variates. Ranganath et al. [2014] build on this work, using the score function of the approximate posterior to control the gradient of Black Box Variational Inference objectives.

Inspiration for this work comes from Grathwohl et al. [2018], where they use a recognition network to approximate the model and its gradient as a control variate. Miller et al. [2017] approximate the reparameterisation gradient for Gaussian variational distributions by its first-order Taylor expansion, using this approximation as a control variate. The work in this chapter is related to this construction where the recognition network can be viewed as a cheap approximation to the linear term in the Taylor expansion of the gradient (*i.e.*, the Hessian of the model objective) in the case of the linear construction in Section 4.4.

The unifying work of Geffner and Domke [2018] categorises different control variate schemes for VI objectives. Additionally, they propose combining them to achieve greater variance reduction. They derive an optimal rule for this combination based on Bayesian risk minimisation.

The related works listed above do not consider the effect of mini-batching on the proposed control variates; therefore, the work in this chapter *complements* the methods mentioned above. Indeed, Geffner and Domke [2018] show that a combination of control variates is usually more desirable than a single scheme. The method that will be presented can be considered an extra addition to the control variate toolkit for doubly stochastic objectives, to take the effect of mini-batch stochasticity on the control variates into account. It can also be combined with other variance reduction methods such as extra sampling.

4.6 Experimental Validation

The discussion thus far applied to the general class of doubly stochastic objectives. For experimental validation, the focus will be on objectives from VI problems. Amortising the computation of the control variate coefficients in this setting is advantageous since context arises naturally from the data in the underlying models.

In this section, the aim is to answer three questions: a) To what extent can amortisation with a recognition network reduce the variance compared to a fixed context-free control variate coefficient? b) How well can the recognition network be

trained in an online setting? c) What difference can an amortised control variate make in practice?

4.6.1 Setup

The questions above are investigated on a classification task on the *titanic* dataset using a Bayesian logistic regression model and on a regression task on the *airfoil* dataset using a Deep Gaussian Process (DGP) [Salimbeni and Deisenroth, 2017]. A detailed description of the models and datasets can be found in Appendix B.2.

The Bayesian logistic regression model uses the reparameterisation gradient formulation of the VI problem. A Gaussian approximate posterior is chosen, where the mean vector and the full covariance matrix are learned. A unit Gaussian prior is placed on the weights.

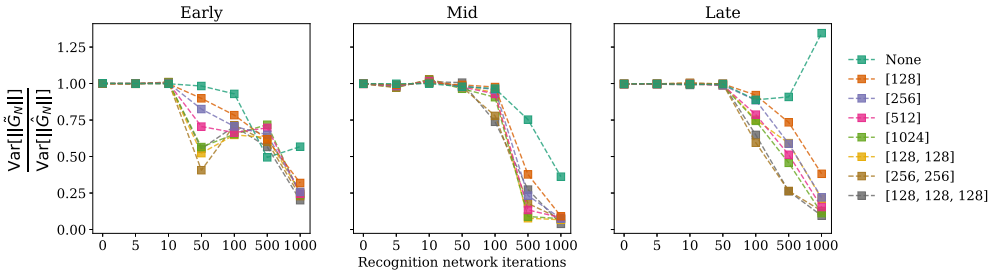
The DGP model uses a 2-layer construction with an inner layer dimension of 5, and a Squared Exponential kernel for the GP priors. The inference uses the doubly stochastic formulation [Salimbeni and Deisenroth, 2017]. The parameters of the approximate Gaussian posterior are learned, while the hyperparameters are kept fixed. Finally, the inducing locations are fixed and selected as the centroids of k-means clusters from the data.

Throughout, Adam [Kingma and Ba, 2015] is used for optimising both the model objective function and the recognition network objective. The gradient estimators use a single MC sample, and when stated, are controlled with the linear a or the quadratic control variates introduced in Section 4.4. The recognition network is initialised with Xavier initialisation [Glorot and Bengio, 2010] and use ReLU activations in the hidden layers.

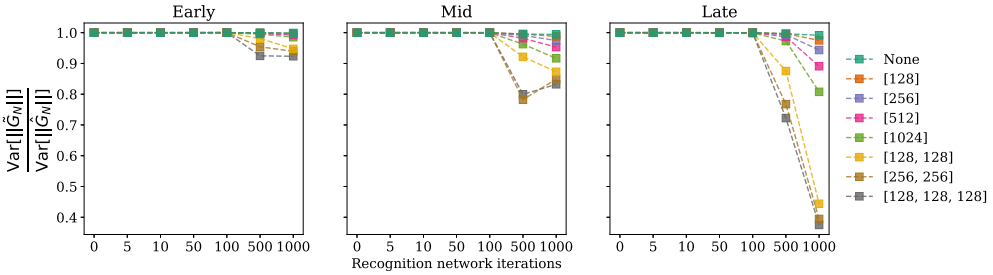
This chapter's proposal is compared to a context-free control variate. In this instance, this is implemented as an optimisable quantity that does not depend on data and uses the same optimisation objectives, (4.13) & (4.15), as the recognition network, i.e., $\mathbf{c}$ is independent of the mini-batch $\mathcal{B}$ in these objectives. This is equivalent to approximating the coefficient with an exponentially weighted moving average of the empirical covariance of the gradient and the control variate estimates.

4.6.2 Verification of Variance Reduction

The first question considered is whether the recognition network has the capacity to amortise the control variate coefficients and how well it can learn these versus a context-free coefficient? To test this, the model parameters are frozen at three points in the optimisation, early (10 steps), mid (200 steps), and late (1000 steps). In each



(a) Logistic regression on *titanic*.



(b) DGP on *airfoil*.

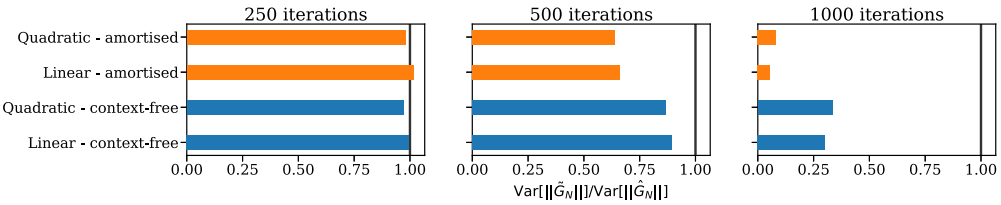
Figure 4.2: Variance reduction at different points in the objective optimisation (lower is better); early: 10 steps, mid: 200 steps, late: 1000 steps. The results are shown for the *linear* control variate from Section 4.4. The different colours represent different recognition network architectures. The recognition network training uses the *squared difference* objective optimised with Adam with learning rate of 10^{-2} for the logistic regression and 10^{-3} for the DGP. For a small number of iterations on the recognition network, it struggles to learn a good control variate coefficient. Continuing the network optimisation, it is able to learn good control variate coefficients that significantly reduce the variance in comparison to the context-free coefficient. Also notable is that the variance reduction is more pronounced at the later stages of the model optimisation.

period, the control variate coefficients are optimised for 1000 iterations. The variance reduction is recorded at different points of the recognition network optimisation by sampling 100 gradient values and computing their empirical variance. The variance reduction is measured by the ratio $\text{Var}[\|\tilde{G}_N\|]/\,\text{Var}[\|\hat{G}_N\|]$, where $\tilde{G}_N$ and $\hat{G}_N$ are the controlled and uncontrolled gradients, respectively, over the mini-batch (size 10), and $\|\cdot\|$ is the gradient norm. Different network sizes are compared to examine the effect of network architecture on variance reduction.

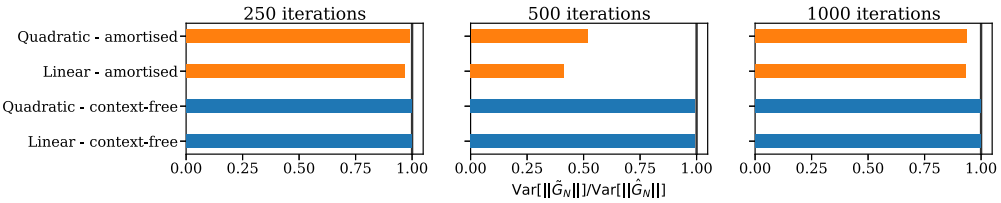
Fig. 4.2 shows that amortising the control variate coefficients induces greater variance reduction than context-free coefficients (labelled as *None* in the figure) The variance reduction does not occur immediately, as the control variate coeffi-

cients need to be optimised in all cases to reduce the variance. Also notable is that the amount of variance reduction depends on the optimisation stage of the model; at later stages of the model optimisation, the variance reduction is more pronounced. This is likely a property of both the model and the control variate where the gradients at the beginning of the optimisation have more pronounced non-linearity with respect to the noise. This can also be seen in the amount of variance reduction in logistic regression compared to the DGP. The gradients in the logistic regression models are approximately linear with respect to the noise, while the DGP gradients have a more complex dependency on the noise. Finally, one can see that the variance reduction potential depends on the capacity of the network, where wider and deeper networks learn better control variate coefficients. Deeper networks reduce the variance more than wider networks, corresponding to a highly non-linear mapping from the context points to the control variate coefficient.

4.6.3 Simultaneous Optimisation of Objective Function and Control Variate Coefficient



(a) Logistic regression on *titanic*.



(b) DGP on *airfoil*.

Figure 4.3: Gradient variance ratio at three different points in the joint optimisation of the model and control variate parameters (lower is better); the vertical line corresponds to a ratio of 1 (i.e. no reduction). Both the model and the control variate objectives are optimised with Adam with learning rate of 10^{-2} for the model and 10^{-2} and 10^{-3} for the logistic regression and DGP control variate coefficients respectively. The recognition network learns a good control variate coefficient and continues to improve throughout the optimisation, outperforming the context-free control variate.

In practice, the recognition network must be able to learn the control variate coefficients while the model is being optimised, resulting in a moving target. In this sub-section, the viability of chasing this target is investigated by simultaneously optimising the model objective and recognition network. A recognition network with three layers of size 128 each is used, as this architecture showed the largest variance reduction in Section 4.6.2. In each step in the optimisation procedure, one gradient estimate of the model objective is computed for a mini-batch of size 10. Consequently, a single Adam step is taken on the recognition network, then the control variate correction is applied to the sampled gradient. Finally, an Adam step is taken on the model parameters with the controlled gradient. The variance of the gradient is measured at different periods in the optimisation by sampling 100 gradient values at each period and taking the empirical variance of their norm.

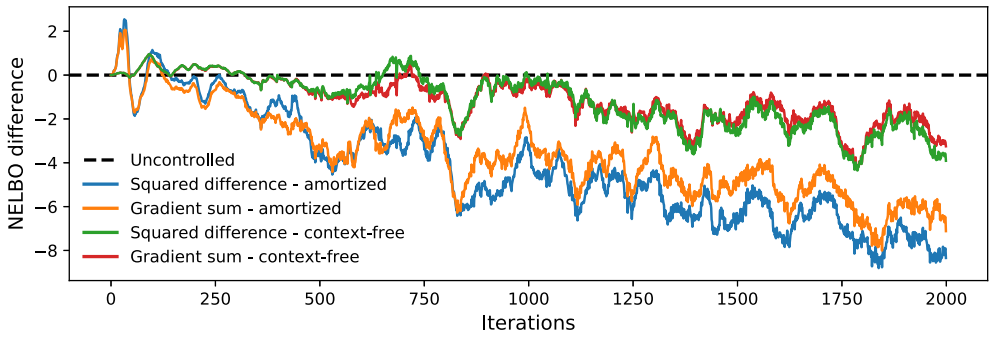
The recognition network is able to learn good control variate coefficients in this dynamic regime, see Fig. 4.3. The variance reduction improves later on in the optimisation as observed in Section 4.6.2. Again, one observes that the amortised control variate results in greater variance reduction than the context-free one.

4.6.4 Practical Effectiveness

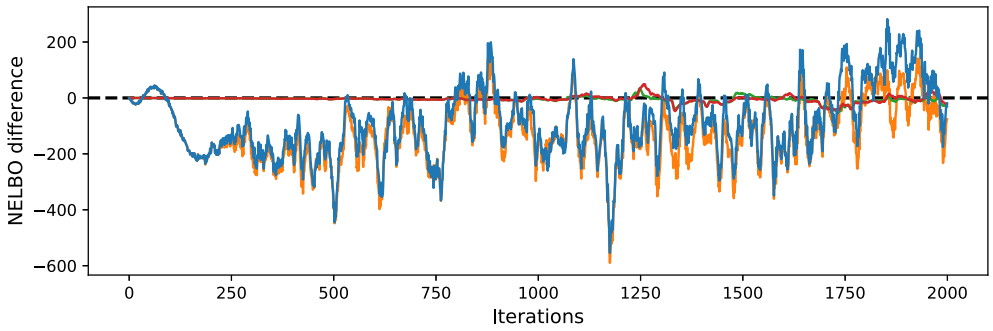
To show how this approach works in practice, it is used for training the logistic regression and DGP models. The alternating optimisation procedure described in Section 4.6.3 is applied to each for 2000 iterations with mini-batches of size 10. The mean value of the Negative Evidence Lower Bound (NELBO) is recorded, from 100 MC samples for the logistic regression and 10 MC samples for the DGP at every iteration computed on the full data sets.

The resulting traces are shown in Fig. 4.4; in both cases, one can see that the optimisation with controlled gradients starts in a worse regime than the uncontrolled gradients (curves on or above the dashed line); however, it improves as better control variate coefficients are learned. The gap between the one-sample MC estimator and the controlled estimators widens later for logistic regression and fluctuates for the DGP with some instability in the end. This is because the linear control variate sufficiently approximates the dependence of the gradient on the randomness in the case of logistic regression, whereas for the DGP this dependence is more complex. This can be verified in Fig. 4.3b, where the variance reduction diminishes later on in the optimisation.

For both models, amortising the control variate coefficients results in lower NELBO values on average in comparison to the uncontrolled and the context-free controlled cases. One also notices that the optimisation of the control variate coef-



(a) Logistic regression on *titanic*.



(b) DGP on *airfoil*.

Figure 4.4: The difference between optimisation traces for different control variate objectives, using the uncontrolled one-sample MC estimate of the gradient as a baseline (lower is better). *Linear* control variates are used in this experiment. The gap between the baseline and controlled models widens throughout the optimisation. Amortised control variate coefficients result in wider gaps indicating better optimisation performance.

ficients is insensitive to the choice of the objective function, with similar behaviour for the gradient sum and squared difference objectives for both the amortised and context-free cases.

Table 4.1 shows the average cost for the controlled optimisation steps for the two problems. Amortising the control variate coefficients with a recognition network of size [128, 128, 128] has an additional overhead of around 25% on the context-free coefficient on the CPU, which is a good investment for high-variance objectives. The overhead depends on many factors such as the recognition network size, control variate formulation, mini-batch size and number of gradient components. These should all be taken into account when implementing this scheme.

Table 4.1: Average optimisation step time in *milliseconds* (on the CPU) for logistic regression and DGP for different *linear* control variate objective functions. Mean figures are presented with the standard error in parentheses. The statistics are computed based on 100 repetitions of 10 runs. The implementation uses TensorFlow 2.0 [Abadi et al., 2016] and GPflow [Matthews et al., 2017; van der Wilk et al., 2020].

Method	Logistic	DGP
Squared diff. - amortised	1.20(0.06)	3.77(0.22)
Grad. sum - amortised	1.25(0.11)	3.78(0.19)
Squared diff. - context-free	0.87(0.09)	3.17(0.13)
Grad. sum - context-free	0.84(0.08)	3.07(0.08)

4.7 Concluding Remarks

This chapter introduced a control variate formulation that exploits the structure of doubly stochastic objectives to reduce Monte Carlo sampling variance from mini-batch gradient estimators. This has direct application in VI problems, where the inference procedure is stabilised due to the resulting low variance gradients.

This contribution proposed three objectives for an amortising recognition network that can learn context-aware control variate coefficients. Training the network re-uses the gradients of the model objective and does not require additional passes through the model.

Empirical assessment showed that an approximation to the optimal control variate per mini-batch can be performed during optimisation and reduces the gradient variance in practice compared to a context-free global approach. In the experiments, linear and quadratic control variates for Gaussian base randomness were used, but this approach is general and can be applied to other control variate formulae and randomness schemes. This is particularly important for complex objectives such as in DGP models, where the simple linear and quadratic control variates fail to stabilise the optimisation in its later stages. This failure motivates combining this method with more adaptive control variates, *e.g.*, Radial Basis Network control variates.

The computational overhead of the proposed control variate could be further reduced with the right compute architecture. Although evaluating the target objective may not benefit from additional compute nodes, one can use a dedicated node for efficiently evaluating the recognition network in parallel as detailed in Section 4.3.3.

While this chapter's proposal aims to only reduce the MC sampling variance in doubly stochastic objectives, another promising research direction is to also target the variance due to mini-batching by combining the current scheme with other variance reduction methods targeting the data sub-sampling variance, *e.g.*, Stochastic Variance Reduced Gradient (SVRG) [Johnson and Zhang, 2013] and Stochastic Gradient Recursive Algorithm (SARAH) [Nguyen et al., 2017].

CHAPTER 5

A Low Variance Gradient Estimator for Variational Inference

As seen in the previous chapter, the stability of Variational Inference (VI) procedures for probabilistic models, especially compositional models, is inherently tied to the variance of the gradients of their objectives. While Chapter 4 studied the stabilisation of VI with doubly stochastic objectives, this chapter examines the special case of objectives that admit score function (Reinforce) estimators (*cf.* Section 2.3.3). This estimator is essential in making VI applicable to various problems such as the problem of inference for models with discrete latent structures or with non-differentiable likelihoods. However, the score function estimator is notoriously high variance, causing significant instability when optimising the VI objective. Hence, this chapter attempts to improve the stability of VI by introducing an alternative gradient estimator to the score function estimator, that is shown to have lower variance.

5.1 Motivation

Estimating the gradient of the expectation of a function is a problem with applications in many areas of machine learning, ranging from variational inference to reinforcement learning [Mohamed et al., 2020]. Different gradient estimators lead to different algorithms; two examples of estimators are the score function gradient [Williams, 1992] and the reparameterisation gradient [Kingma et al., 2015; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014]. In the specific case of VI, the variational optimisation problem can be solved with gradient-based stochastic optimisation tools when the ELBO is not available in closed form. Thus, MC estimators

of the ELBO gradient, $\nabla_{\boldsymbol{\phi}}\text{ELBO}(\boldsymbol{\phi})$, can be used in the optimisation. This chapter studies a multi-sample estimator of the gradient of the ELBO, which was first introduced by Salimans and Knowles [2014] and Kool et al. [2019]. This estimator is based on the score function method [Williams, 1992] with leave-one-out control variates and is referred to as *VarGrad* in this thesis.

This chapter explores the connection between this estimator and an alternative divergence measure between the variational distribution $q_{\boldsymbol{\phi}}(\mathbf{x})$ and the exact posterior $p(\mathbf{x} \mid \mathbf{y})$. This divergence differs from the standard KL used in VI, and is defined as the variance – under some arbitrary distribution $r(\mathbf{x})$ – of the log-ratio $\log \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x} \mid \mathbf{y})}$ and is called the *log-variance loss* [Nüsken and Richter, 2020]. It is possible to recover the gradient estimator of Salimans and Knowles [2014] and Kool et al. [2019] by taking the gradient with respect to the variational parameters $\boldsymbol{\phi}$ of the log-variance loss and evaluating the result at $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$. This property gives this gradient estimator its name, VarGrad, and suggests a simple algorithm for computing the gradient estimator, based on differentiating through the log-variance loss.

The relationship between VarGrad and the score function estimator [Williams, 1992; Carbonetto et al., 2009; Paisley et al., 2012; Ranganath et al., 2014] with a control variate is studied, where the control variate coefficients of VarGrad shown to be close to the (generally intractable) optimal coefficients. This difference is shown both theoretically and empirically to be small in many cases; for example when the KL from $q_{\boldsymbol{\phi}}(\mathbf{x})$ to the posterior is either small or large, which is generally the case in the late and early stages of the optimisation, respectively. This explains the success of the VarGrad estimator in stabilising inference procedures in a variety of settings [Kool et al., 2019, 2020].

Since it is based on the score function (*cf.* Section 2.3.3), VarGrad is a black-box, general-purpose estimator because it makes no assumptions on the model $p(\mathbf{x}, \mathbf{y})$, such as differentiability with respect to the latent variables $\mathbf{x}$. It introduces no additional parameters to be tuned and it is not computationally expensive. Section 4.6 shows empirically that VarGrad exhibits a favourable variance versus computation trade-off compared to other unbiased gradient estimators, including the score function gradient with control variates [Williams, 1992; Ranganath et al., 2014], REBAR [Tucker et al., 2017], RELAX [Grathwohl et al., 2018], and ARM [Yin and Zhou, 2019]. It also shows that VarGrad is successful in stabilising the inference procedure where the standard score function estimator is not.

5.2 Background

This section briefly recaps the score function estimator (introduced in (2.45) in Section 2.3.3) and reviews its improved version based on leave-one-out control variates.

Consider a probabilistic model $p(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ denotes the latent variables and $\mathbf{y}$ denotes the observations.The interest is in computing the posterior $p(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{x}, \mathbf{y})/p(\mathbf{y})$, where $p(\mathbf{y}) = \int p(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x}$ is the marginal likelihood. VI approximates the posterior $p(\mathbf{x} \mid \mathbf{y})$ with a parameterised family of distributions $q_{\boldsymbol{\phi}}(\mathbf{x})$ (with $\boldsymbol{\phi} \in \Phi$), called the variational family and finds the parameters $\boldsymbol{\phi}^\star$ that minimise the KL divergence, $\boldsymbol{\phi}^\star = \arg\min_{\boldsymbol{\phi} \in \Phi} \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \mid \mathbf{y})\right)$. As discussed in Section 2.3, this optimisation problem is intractable because the KL itself depends on the intractable posterior. Variational inference sidesteps this problem by maximising instead the ELBO

$$\mathrm{ELBO}(\boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{x})}\left[\log \frac{p(\mathbf{x}, \mathbf{y})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right], \tag{5.1}$$

which is a lower bound on the marginal likelihood, since $\log p(\mathbf{y}) = \mathrm{ELBO}(\boldsymbol{\phi}) + \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \mid \mathbf{y})\right)$. As the expectation in (5.1) is typically intractable, variational inference uses stochastic optimisation to maximise the ELBO. In particular, it forms unbiased Monte Carlo estimators of the gradient $\nabla_{\boldsymbol{\phi}} \mathrm{ELBO}(\boldsymbol{\phi})$. Note that the negative of the ELBO gradient coincides with the gradient of the KL divergence with respect to the variational parameters $\boldsymbol{\phi}$, since the marginal likelihood $p(\mathbf{y})$ does not depend on $\boldsymbol{\phi}$, *i.e.*, $\nabla_{\boldsymbol{\phi}} \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \mid \mathbf{y})\right) = -\nabla_{\boldsymbol{\phi}} \mathrm{ELBO}(\boldsymbol{\phi})$.

The score function estimator [Williams, 1992; Carbonetto et al., 2009; Paisley et al., 2012; Ranganath et al., 2014], also known as Reinforce, expresses the gradient as an expectation that depends on the log-ratio $q_{\boldsymbol{\phi}}(\mathbf{x})/p(\mathbf{x}, \mathbf{y})$ weighted by the score function $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})$. The resulting estimator is

$$\nabla_{\boldsymbol{\phi}} \mathrm{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \mid \mathbf{y})\right] \approx \hat{g}_{\mathrm{Reinforce}}(\boldsymbol{\phi}) = \frac{1}{S} \sum_{s=1}^{S} \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})}{p(\mathbf{x}^{(s)}, \mathbf{y})}\right) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}), \tag{5.2}$$

where $\mathbf{x}^{(s)} \overset{\text{i.i.d.}}{\sim} q_{\boldsymbol{\phi}}(\mathbf{x})$. Due to its high variance, the score function estimator requires additional variance reduction tricks in practice; Ranganath et al. [2014] use Rao-Blackwellization and control variates. The control variates used in this case are multiples of the score function, $\mathbf{a} \odot \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x})$, where $\odot$ denotes the Hadamard (element-wise) product and the coefficient $\mathbf{a}$ is chosen to minimise the estimator variance forming a regression estimator (2.97).

Salimans and Knowles [2014] and Kool et al. [2019] leverage the multi-sample estimator by using $S - 1$ samples to compute the control variate coefficient a and

then average over the resulting estimators, obtaining a leave-one-out estimator

$$\hat{g}_{\text{LOO}}(\boldsymbol{\phi}) = \frac{1}{S-1}\left(\sum_{s=1}^{S} f_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}) - \bar{f}_{\boldsymbol{\phi}} \sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})\right), \quad \mathbf{x}^{(s)} \overset{\text{i.i.d.}}{\sim} q_{\boldsymbol{\phi}}(\mathbf{x}), \tag{5.3}$$

where for simplicity of notation, $f_{\boldsymbol{\phi}}(\mathbf{x})$ is defined as the log-ratio $\log \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x},\mathbf{y})}$ and $\bar{f}_{\boldsymbol{\phi}}$ as its empirical average, which is a multi-sample Monte Carlo estimate of the negative ELBO, *i.e.*,

$$f_{\boldsymbol{\phi}}(\mathbf{x}) = \log \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x},\mathbf{y})} \quad \text{and} \quad \bar{f}_{\boldsymbol{\phi}} = \frac{1}{S}\sum_{s=1}^{S} f_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}) \approx -\text{ELBO}(\boldsymbol{\phi}). \tag{5.4}$$

The score function method makes no assumptions on the model $p(\mathbf{x},\mathbf{y})$ or the distribution $q_{\boldsymbol{\phi}}(\mathbf{x})$; the only requirements are to be able to sample from $q_{\boldsymbol{\phi}}(\mathbf{x})$ and to evaluate $\log q_{\boldsymbol{\phi}}(\mathbf{x})$ and $\log p(\mathbf{x},\mathbf{y})$.

5.3 The Log-Variance Loss and its Connection to VarGrad

This section establishes the connection between the leave-one-out estimator in (5.3) and a novel divergence, called the log-variance loss [Nüsken and Richter, 2020], which is introduced in in Section 5.3.1. Its connection to (5.3) is established in Section 5.3.2. Due to this connection, from this point this thesis refers to the estimator in (5.3) as *VarGrad*.

5.3.1 The Log-Variance Loss

The log-variance loss is defined as the variance, under some arbitrary distribution $r(\mathbf{x})$, of the log-ratio $\log \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}$. It has the property of reproducing the gradients of the KL divergence under certain conditions (see Proposition 5.3.2 for details).

Definition 5.3.1. *For a given distribution $r(\mathbf{x})$, the log-variance loss $\mathcal{L}_r(\cdot)$ is given by*

$$\mathcal{L}_r(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) = \frac{1}{2}\text{Var}_r\left[\log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right)\right]. \tag{5.5}$$

The distribution $r(\mathbf{x})$ is referred to as the *reference distribution* under which the discrepancy between $q_{\boldsymbol{\phi}}(\mathbf{x})$ and the posterior $p(\mathbf{x}\,|\,\mathbf{y})$ is computed. When the support of the reference distribution contains the supports of $q_{\boldsymbol{\phi}}(\mathbf{x})$ and $p(\mathbf{x}\,|\,\mathbf{y})$,

(5.5) is a divergence; it is zero if and only if $q_{\phi}(\mathbf{x}) = p(\mathbf{x} \,|\, \mathbf{y})$. The factor $1/2$ in (5.5) is included because it simplifies some expressions later in this section.

It is worth noting that, for a fixed reference distribution $r(\mathbf{x})$, one can fit the variational parameters by minimising (5.5),

$$\boldsymbol{\phi}^{\star} = \arg\min_{\boldsymbol{\phi} \in \Phi} \mathcal{L}_r(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \,|\, \mathbf{y})). \tag{5.6}$$

(5.6) leads to an alternative variational inference problem. This optimisation problem has the same solution $\boldsymbol{\phi}^{\star}$ as standard variational inference based on the KL divergence when the variational family contains the true posterior, *i.e.*, when $p(\mathbf{x} \,|\, \mathbf{y}) \in \{q_{\boldsymbol{\phi}}(\mathbf{x}) : \boldsymbol{\phi} \in \Phi\}$. In such case, the minimum of both objectives coincide and is attained at $q_{\boldsymbol{\phi}^{\star}}(\mathbf{x}) = p(\mathbf{x} \,|\, \mathbf{y})$. In contrast, when the variational family does not contain the true posterior, the problem in (5.6) may lead to a different solution that depends on the reference distribution $r(\mathbf{x})$.

Under certain conditions, the gradient of the log-variance loss and the gradient of the standard KL divergence coincide. In particular, taking the gradient of (5.5) with respect to the variational parameters $\boldsymbol{\phi}$ *and then* evaluating the result for a reference distribution $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$ gives the gradient of the KL. This property is detailed in Proposition 5.3.2.

Proposition 5.3.2. *The gradient with respect to $\boldsymbol{\phi}$ of the log-variance loss, evaluated at $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$, equals the gradient of the KL divergence,*

$$\nabla_{\boldsymbol{\phi}} \mathcal{L}_r(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \,|\, \mathbf{y}))\Big|_{r=q_{\boldsymbol{\phi}}} = \nabla_{\boldsymbol{\phi}} KL(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \,|\, \mathbf{y})). \tag{5.7}$$

Proof. See Appendix C.2.1. □

Proposition 5.3.2 implies that the gradient of the KL divergence can be estimated by instead estimating the gradient of the log-variance loss.

Remark 5.3.3. *The result in Proposition 5.3.2 is obtained by setting $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$* after *taking the gradient with respect to $\boldsymbol{\phi}$. The same result does not hold if $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$ is set before differentiating.*

5.3.2 VarGrad: Derivation of the Gradient Estimator from the Log-Variance Loss

One can show a connection between the leave-one-out estimator in Equation (5.3) [Salimans and Knowles, 2014; Kool et al., 2019] and the log-variance loss from Section 5.3.1. The log-variance loss is usually intractable as it depends on the posterior

$p(\mathbf{x} \mid \mathbf{y})$, similar to the KL divergence in standard VI. However, the marginal likelihood $p(\mathbf{y})$ can be dropped from the definition in (5.5) since it has zero variance, *i.e.*

$$\mathcal{L}_r(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \mid \mathbf{y})) = \frac{1}{2}\mathrm{Var}_r\left[\log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}, \mathbf{y})}\right)\right] = \frac{1}{2}\mathrm{Var}_r\left[f_{\boldsymbol{\phi}}(\mathbf{x})\right], \tag{5.8}$$

where $f_{\boldsymbol{\phi}}(z)$ is defined in (5.4).

Empirically, the log-variance loss can be estimated from S MC samples,

$$\mathcal{L}_r(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x} \mid \mathbf{y})) \approx \frac{1}{2(S-1)}\sum_{s=1}^{S}\left(f_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}) - \bar{f}_{\boldsymbol{\phi}}\right)^2, \quad \mathbf{x}^{(s)} \overset{\text{i.i.d.}}{\sim} r(\mathbf{x}), \tag{5.9}$$

where Bessel's correction [Kenney, 1939] is used to ensure unbiased estimation.

Applying Proposition 5.3.2 by differentiating through (5.9), one arrives at the VarGrad estimator

$$\hat{g}_{\text{VarGrad}}(\boldsymbol{\phi}) = \frac{1}{S-1}\left(\sum_{s=1}^{S} f_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}) - \bar{f}_{\boldsymbol{\phi}}\sum_{s=1}^{S}\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})\right), \tag{5.10}$$

with $\mathbf{x}^{(s)} \overset{\text{i.i.d.}}{\sim} q_{\boldsymbol{\phi}}(\mathbf{x})$. The expression for VarGrad in (5.10) is identical to the expression of the leave-one-out estimator in (5.3) and is an unbiased estimator of the gradient of the standard KL (and equivalently the ELBO).

Deriving VarGrad as above gives a convenient way of computing this estimator; hence, from a probabilistic programming perspective, setting the reference $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$ *after* differentiating w.r.t. $\boldsymbol{\phi}$ amounts to sampling $\mathbf{x}^{(s)} \sim q_{\boldsymbol{\phi}}(\mathbf{x})$ and detaching the resulting samples from the computational graph. This novel algorithmic procedure for computing this estimator is given in Algorithm 5. Its implementation is simple: one only needs to sample $\mathbf{x}^{(s)} \sim q_{\boldsymbol{\phi}}(\mathbf{x})$, apply the `stop_gradient` operator[1], evaluate the log-ratio $f_{\boldsymbol{\phi}}(\mathbf{x}^{(s)})$ for each sample, and then differentiate through the empirical variance of this log-ratio.

5.4 Relationship to Reinforce with Score-based Control Variates

This section develops VarGrad's relationship to the score function estimator (Reinforce), specifically when the latter is used with a control variate. One can recover the optimal control variate coefficient for the Reinforce estimator by adding a correction

[1]This is available in most modern automatic differentiation frameworks, albeit under different names.

Algorithm 5 Pseudocode for VarGrad

Input: Variational parameters $\boldsymbol{\phi}$, data $\mathbf{y}$
  for $s = 1, \ldots, S$ **do**
    $\mathbf{x}^{(s)} \leftarrow \texttt{sample}(q_{\boldsymbol{\phi}}(\cdot))$ ▷ Sample from the approximate posterior
    $\mathbf{x}^{(s)} \leftarrow \texttt{stop\_gradient}(\mathbf{x}^{(s)})$ ▷ Detach samples from the computational graph
    $f_{\boldsymbol{\phi}}^{(s)} \leftarrow \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}) - \log p(\mathbf{x}^{(s)}, \mathbf{y})$ ▷ Estimate the negative ELBO
  end for
  $\widehat{\mathcal{L}} \leftarrow \frac{1}{2}\texttt{Variance}(\{f_{\boldsymbol{\phi}}^{(s)}\}_{s=1}^{S})$ ▷ Estimate the log-variance loss
  return $\texttt{grad}(\widehat{\mathcal{L}})$ ▷ Differentiate through the loss w.r.t. $\boldsymbol{\phi}$

term $\boldsymbol{\delta}^{\mathrm{CV}}$ to the control variate coefficient induced by VarGrad $\mathbf{a}_{\mathrm{VarGrad}}$.

5.4.1 Reinforce with Score Control Variates

Due to its high variance, Reinforce typically needs variance reduction in practice. Ranganath et al. [2014], introduces a regression estimator (2.97) where the score is used as a control variate

$$\hat{g}_{\mathrm{CV}}(\boldsymbol{\phi}) = \hat{g}_{\mathrm{Reinforce}}(\boldsymbol{\phi}) - \mathbf{a} \odot \left(\frac{1}{S} \sum_{s=1}^{S} \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{x}^{(s)}) \right). \tag{5.11}$$

From (2.100), the optimal control variate coefficient for the i^{th} gradient component is given as

$$a_i^{\star} = \frac{\mathrm{Cov}_{q_{\boldsymbol{\phi}}} \left(f_{\boldsymbol{\phi}}(\mathbf{x}) \partial_{\boldsymbol{\phi}_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}), \partial_{\boldsymbol{\phi}_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}) \right)}{\mathrm{Var}_{q_{\boldsymbol{\phi}}} \left(\partial_{\boldsymbol{\phi}_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}) \right)}. \tag{5.12}$$

The coefficient in (5.12) cannot usually be computed in closed-form due to the intractable expectations in the fraction. In practice, this coefficient is estimated with extra MC samples; however, this estimation can suffer from both bias and variance issues since it is takes the form of a ratio of two MC estimators [2].

5.4.2 VarGrad as an Approximation to Reinforce with Optimal Control Variate Coefficients

One can recover VarGrad (5.10), up to a factor of proportionality, by setting the control variate coefficient in (5.11) to $\mathbf{a}_{\mathrm{VarGrad}} := \mathbf{a} = \bar{f}_{\boldsymbol{\phi}}\mathbf{1}$, where $\mathbf{1}$ is a vector of ones. In this case, the proportionality relation is $\hat{g}_{\mathrm{VarGrad}} = \frac{S}{S-1}\hat{g}_{\mathrm{CV}}$. Furthermore,

[2] Note that this does not alter the bias and variance of the original regression estimator as any value of a yields an unbiased estimator; however, this can affect the extent of variance reduction that the regression estimator induces.

the VarGrad coefficient, $\mathbf{a}_{\text{VarGrad}}$, can be related to the optimal coefficient $\mathbf{a}^\star$ as follows:

Lemma 5.4.1. *The optimal control variate coefficient can be written as the expected value of* $\mathbf{a}_{VarGrad}$ *plus a* control variate correction *term* $\boldsymbol{\delta}^{CV}$*, i.e.,*

$$\mathbf{a}^* = \mathbb{E}_{q_{\boldsymbol{\phi}}}[\mathbf{a}_{VarGrad}] + \boldsymbol{\delta}^{CV} = -\text{ELBO}(\boldsymbol{\phi}) + \boldsymbol{\delta}^{CV}, \tag{5.13}$$

where $\mathbf{a}_{VarGrad} = \bar{f}_{\boldsymbol{\phi}}$ *and the components of the correction term* $\boldsymbol{\delta}^{CV}$ *are given by*

$$\delta_i^{CV} = \frac{\text{Cov}_{q_{\boldsymbol{\phi}}}\left(f_{\boldsymbol{\phi}}(\mathbf{x}), \left(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x})\right)^2\right)}{\text{Var}_{q_{\boldsymbol{\phi}}}\left(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x})\right)}. \tag{5.14}$$

Proof. See Appendix C.2.2. □

In certain settings, the correction term $\boldsymbol{\delta}^{\text{CV}}$ becomes negligible, implying that $\hat{g}_{\text{VarGrad}}$ and $\hat{g}_{\text{Reinforce}}$ equipped with the optimal control variate coefficients behave almost identically. These settings are explored analytically in Appendix C.1. In addition, Section 5.6 provides empirical evidence of this finding.

5.4.3 Variance of the VarGrad Estimator

The variance of $\hat{g}_{\text{VarGrad}}(\boldsymbol{\theta})$ can be compared to the variance of $\hat{g}_{\text{Reinforce}}(\boldsymbol{\theta})$ as follows

Proposition 5.4.2. *Consider the two gradient estimators* $\widehat{g}_{Reinforce}(\boldsymbol{\phi})$ *and* $\widehat{g}_{VarGrad}(\boldsymbol{\phi})$*, each with* S *Monte Carlo samples, as defined in* (5.2) *and* (5.10)*, respectively. If*

$$-\frac{\delta_i^{CV}}{\mathbb{E}_{q_{\boldsymbol{\phi}}}[\mathbf{a}_{VarGrad}]} = \frac{\delta_i^{CV}}{\text{ELBO}(\boldsymbol{\phi})} < \frac{1}{2} \tag{5.15}$$

then there exists $S_0 \in \mathbb{N}$ *such that*

$$\text{Var}\left(\hat{g}_{VarGrad,i}(\boldsymbol{\phi})\right) \leq \text{Var}\left(\hat{g}_{Reinforce,i}(\boldsymbol{\phi})\right), \qquad \text{for all} \quad S \geq S_0. \tag{5.16}$$

Proof. See Appendix C.2.4. □

If the correction $\boldsymbol{\delta}^{\text{CV}}$ is negligible (see Proposition C.1.1 for analytical guarantees on this case), then the assumption in Equation (5.15) is satisfied and Proposition 5.4.2 guarantees that VarGrad has lower variance than Reinforce when S is large enough. Appendix C.1.2 considers the relationship between the two estimators w.r.t. the dimensionality of the latent variables.

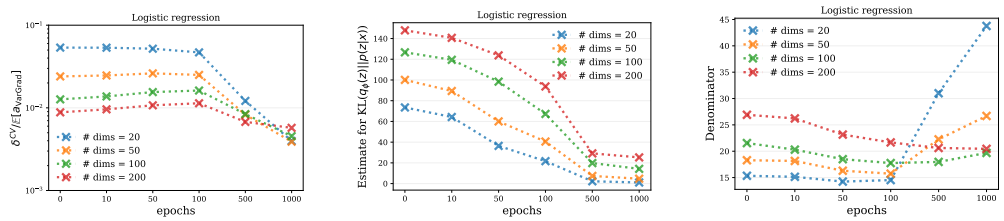
5.5 Related Work

In the last few years, many gradient estimators of the ELBO have been proposed; see Mohamed et al. [2020] for a comprehensive review. Among those, the score function estimators [Williams, 1992; Carbonetto et al., 2009; Paisley et al., 2012; Ranganath et al., 2014] and the reparameterisation estimators [Kingma and Welling, 2014; Rezende et al., 2014; Titsias and Lázaro-Gredilla, 2014], as well as combinations of both [Ruiz et al., 2016; Naesseth et al., 2017], are arguably the most widely used. NVIL [Mnih and Gregor, 2014] and MuProp [Gu et al., 2016] are unbiased gradient estimators for training stochastic neural networks.

Other gradient estimators are specific for discrete-valued latent variables. The concrete relaxation [Maddison et al., 2017; Jang et al., 2017] describes a way to form a biased estimator of the gradient, which REBAR [Tucker et al., 2017] and RELAX [Grathwohl et al., 2018] use as a control variate to obtain an unbiased estimator. Other recent estimators have been proposed by Lee et al. [2018]; Peters and Welling [2018]; Shayer et al. [2018]; Cong et al. [2019]; Yin and Zhou [2019]; Yin et al. [2019]. Section 5.6 compares VarGrad with some of these estimators, showing that it exhibits a favourable performance versus computational complexity trade-off.

The VarGrad estimator was first introduced by Salimans and Knowles [2014] and Kool et al. [2019]. It also relates to VIMCO [Mnih and Rezende, 2016] in that it is a leave-one-out estimator. This chapter has described an alternative derivation of VarGrad, based on the log-variance loss.

The log-variance loss from Section 5.3.1 defines an alternative divergence between the approximate and the exact posterior distributions. In the context of optimal control of diffusion processes and related forward-backward stochastic differential equations, it arises naturally to quantify the discrepancy between measures on path space [Nüsken and Richter, 2020]. Other forms of alternative divergences have also been explored in the VI literature; for example the χ^2-divergence [Dieng et al., 2017], the Rényi divergence [Li and Turner, 2016], the Langevin-Stein [Ranganath et al., 2016a], the α-divergence [Hernández-Lobato et al., 2016], other f-divergences [Wang et al., 2018], a contrastive divergence [Ruiz and Titsias, 2019], and also the inclusive KL [Naesseth et al., 2020].



(a) (b) (c)

Figure 5.1: Verification of Proposition C.1.1 and Remark C.1.3 on the logistic regression model. (a) shows that the ratio $\left|\delta_i^{\mathrm{CV}}/\mathbb{E}_{q_\phi}[\mathbf{a}_{\mathrm{VarGrad}}]\right|$ is small and uniformly bounded over epochs, illustrating that the VarGrad estimator stays close to the optimal control variate coefficients during the whole optimisation procedure. Additionally, this ratio decreases with increasing dimensionality of the latent variables. (b) displays an estimate of the KL divergence across epochs and demonstrate the beneficial effect of higher dimensions, since the bound of Equation (C.3) is expected to scale like $\mathcal{O}(\mathrm{KL}^{-1/2})$ in the early phase. (c) plots an estimate of the denominator of the bound (Equation (C.3)), which increases or stays constant over epochs, demonstrating that the ratio in Equation (C.3) stays stable (and small) over epochs.

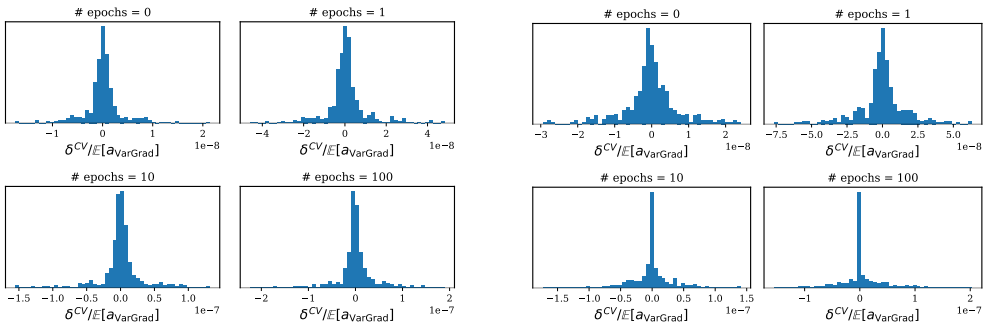
5.6 Experiments

In order to empirically verify its properties, VarGrad was tested on two popular models: Bayesian logistic regression on a synthetic dataset and a Discrete Variational Auto-Encoder (DVAE) [Salakhutdinov and Murray, 2008; Kingma and Welling, 2014] on a fixed binarisation of Omniglot [Lake et al., 2015]. The DVAE results are illustrated with a two-layer linear model [Maddison et al., 2017], which is an example of a compositional latent variable model. The details of the experimental setup are in Appendix C.3. The code for the experiments is publicly available at `https://github.com/aboustati/vargrad`.

5.6.1 Closeness to the optimal control variate

Section 5.4 showed that VarGrad is close to the optimal control variate and Appendix C.1 relates this to the ratio $\left|\delta_i^{\mathrm{CV}}/\mathbb{E}_{q_\phi}[\mathbf{a}_{\mathrm{VarGrad}}]\right|$, which is usually small over the whole optimisation procedure. This behaviour is expected to be even more pronounced with growing dimensionality of the latent space. Figure 5.1 confirms this result by showing the ratio $\left|\delta_i^{\mathrm{CV}}/\mathbb{E}_{q_\phi}[\mathbf{a}_{\mathrm{VarGrad}}]\right|$ for the logistic regression model. It also shows the change with the number of iterations of the KL divergence (panel b) and the denominator of the bound in Equation (C.3) (panel c).

The ratio $\left|\delta_i^{\mathrm{CV}}/\mathbb{E}_{q_\phi}[\mathbf{a}_{\mathrm{VarGrad}}]\right|$ is also observed to be very small for the DVAE



(a) One layer non-linear (3 layers) DVAE.

(b) Two-layer linear DVAE.

Figure 5.2: The distribution of $\frac{\delta_i^{\text{CV}}}{\mathbb{E}[\mathbf{a}_{\text{VarGrad}}]}$ associated with the biases of two DVAE encoders with 200 latent dimensions. The DVAEs are trained on Omniglot with the log-variance loss using Adam with a learning rate of 0.001. The estimates are computed with MC sampling with 2000 samples.

in both the linear and non-linear settings in Fig. 5.2. This indicates that VarGrad can be used as a computationally cheaper substitute to a controlled Reinforce estimator, circumventing the need for extra computation to compute the optimal control variate coefficient. Results on how this affects the variance of the gradient estimator in practice are shown in the next section.

5.6.2 Variance reduction and computational cost

Figure 5.3 shows the variance of different gradient estimators across optimisation iterations in the logistic regression setting. VarGrad is seen to have significantly lower variance across all iterations in comparison to the standard Reinforce estimator (5.2). In fact, there is only a small observed difference between the variance of VarGrad and the variance of an *oracle estimator* based on Reinforce with access to the optimal control variate coefficient $\mathbf{a}^\star$. Figure 5.3 also shows the variance of the *sampled estimator*, the Reinforce-based regression estimator with an estimate of the optimal control variate coefficient; this confirms the difficulty of estimating the coefficient in it in practice. All methods use $S = 4$ Monte Carlo samples, and the control variate coefficient is estimated with either 2 extra samples (*sampled estimator*) or 1,000 samples (*oracle estimator*).

A similar trend can be observed for the DVAE in Figure 5.4, where VarGrad is compared to a wider list of estimators from the DVAE literature. VarGrad achieves considerable variance reduction over the adaptive (RELAX) [Grathwohl et al., 2018]

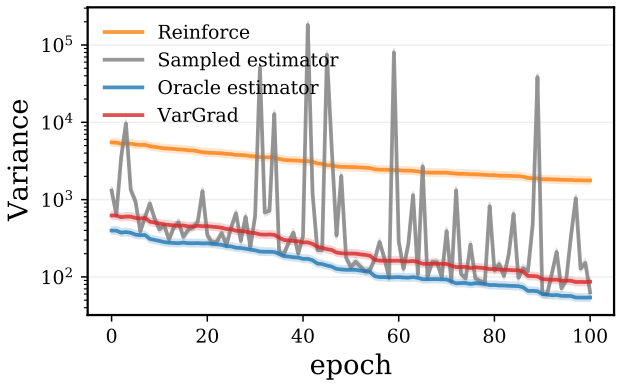


Figure 5.3: Estimates of the variance of the gradient component w.r.t. the posterior mean of one of the weights for the logistic regression model. The variance of VarGrad is close to the *oracle estimator* based on Reinforce with access to the optimal control variate coefficient $\mathbf{a}^\star$. Moreover, the *sampled estimator* (based on Reinforce with an estimate of $\mathbf{a}^\star$) shows the difficulty of estimating the optimal control variate coefficient in practice.

and non-adaptive (Controlled Reinforce) [Ranganath et al., 2014] model-agnostic estimators. Structured adaptive estimators such as Dynamic REBAR [Tucker et al., 2017] and RELAX + REBAR [Grathwohl et al., 2018] start with a higher variance at the beginning of optimisation, which reduces towards the end. ARM [Yin and Zhou, 2019], which uses antithetic sampling, achieves the most reduction; however, it is only applicable to models with Bernoulli latent variables. Notably, the extra variance reduction seen in some of the methods does not translate to better optimisation performance on this example as seen in Figure 5.5.

Finally, Figure 5.5 compares VarGad with other estimators by training a DVAE on Omniglot. The figure shows the negative ELBO as a function of the epoch number (left plot) and against the wall-clock time (right plot) for three different Adam [Kingma and Ba, 2015] learning rates: 0.001, 0.0005 and 0.0001. The negative ELBO is computed on the standard test. VarGrad achieves similar performance to state-of-the-art estimators, such as REBAR [Tucker et al., 2017], RELAX [Grathwohl et al., 2018], and ARM [Yin and Zhou, 2019], while being simpler to implement (see Algorithm 5) and without any tunable hyperparameters.

In comparison to the standard score function estimator, Figure 5.5 shows that VarGrad induces more stable inference, where for the large learning rate of 0.001, one can see that optimisation with the score function estimator diverges. Optimisation with VarGrad does not suffer from this issue and converges at a fast

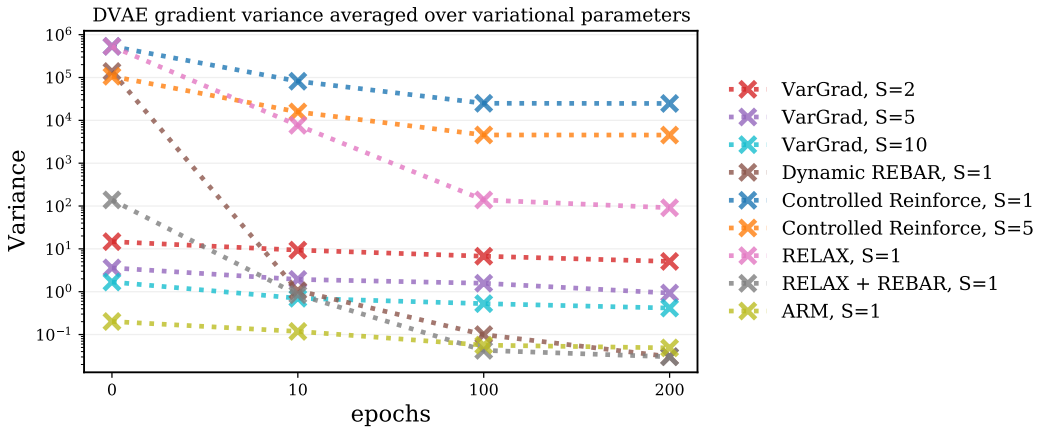


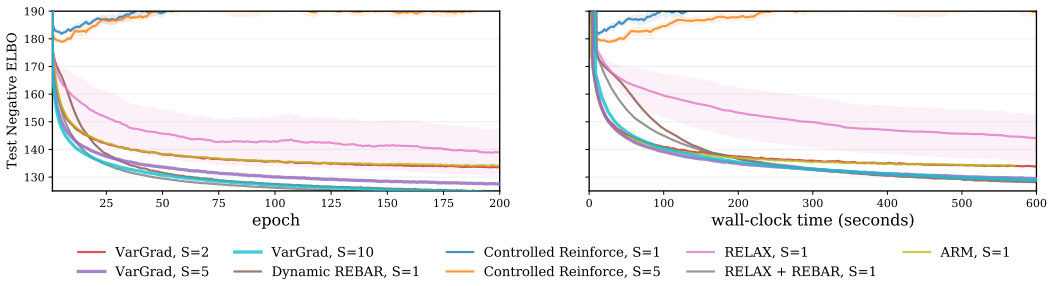
Figure 5.4: Estimates of the gradient variance of a two-layer linear DVAE at 4 points during the optimisation for different gradient estimators. The plot compares VarGrad to Reinforce with score function control variates [Ranganath et al., 2014], dynamic REBAR [Tucker et al., 2017], RELAX, RELAX + REBAR [Grathwohl et al., 2018] and ARM [Yin and Zhou, 2019]. The number of samples used to compute each gradient estimator is given in the figure legend.

rate compared to the other estimators.

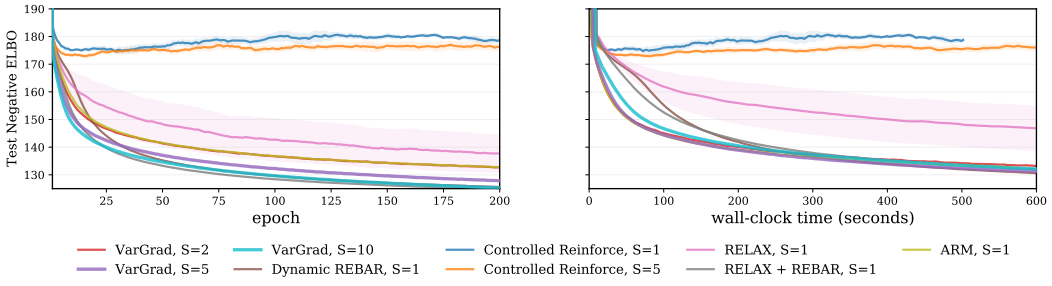
5.7 Concluding Remarks

This chapter studied a new perspective on the VarGrad estimator, an estimator of the gradient of the KL that is based on Reinforce with leave-one-out control variates. This estimator is an important tool in the VI toolbox, that makes it applicable to inference problems for complex models with discrete latent structures. This chapter established the connection between VarGrad and a novel divergence, known as the log-variance loss [Nüsken and Richter, 2020]. It also showed that, under certain conditions, the VarGrad control variate coefficients are close to the optimal ones. This reduction in variance induces more stability during the optimisation of the VI objective. This increased stability was illustrated on a simple logistic regression model (in Fig. 5.3) and a more complex compositional latent variable model, the two-layer linear DVAE (in Fig. 5.5).

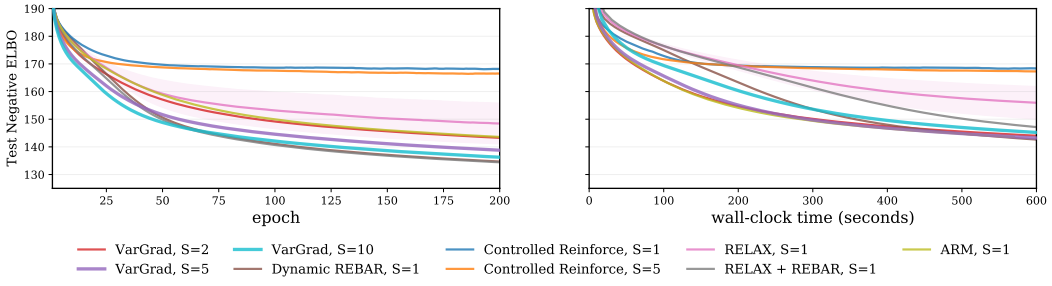
In this chapter, the log-variance loss was used as a proxy loss whose gradient matches that of the ELBO under certain conditions. An interesting idea for future work is to explore the direct optimisation of this loss for alternative choices of the reference distribution $r(\mathbf{x})$, thus generalising the VarGrad estimator.



(a) Learning rate of 0.001



(b) Learning rate of 0.0005



(c) Learning rate of 0.0001

Figure 5.5: Optimisation trace versus epoch (left) and wall-clock time (right) for a two-layer linear DVAE on a fixed binarisation of Omniglot. The plot compares VarGrad to Reinforce with score function control variates [Ranganath et al., 2014], dynamic REBAR [Tucker et al., 2017], REALX, RELAX + REBAR [Grathwohl et al., 2018] and ARM [Yin and Zhou, 2019]. The number of samples used to compute each gradient estimator is given in the figure legend. VarGrad demonstrates favourable scalability and performance when compared to the other estimators.

CHAPTER 6

Generalised Bayesian Filtering

The previous two chapters studied methods to alleviate the instability of inference methods due to the variance induced when estimating quantities that are based on random variables. Another principal source of instability is due to the training data itself. If the observed data differs from its the assumed distribution in a probabilistic model, then the inference procedure becomes unstable and can fail to produce a reliable posterior distribution. Therefore, it is essential to consider this potential mismatch between reality and assumptions when building probabilistic models and performing inference on them.

Models and inference procedures that are stable in the presence of this type of mismatch are known as robust methods and are commonplace in statistics and machine learning. This chapter studies the robustness problem in sequential probabilistic models. It builds on the idea of Generalized Bayesian Inference, deriving a new sampling scheme that can handle contamination in the observations; hence, providing robust inference in these types of models.

6.1 Motivation

Estimating the hidden states in dynamical systems is a long-standing problem in many fields of science and engineering. This can be formulated as an inference problem of a general state-space hidden Markov model (HMM) defined via two processes, *the hidden process* $(\mathbf{x}_t)_{t\geq 0}$, and *the observation process* $(\mathbf{y}_t)_{t\geq 1}$. More precisely, consider the general state-space hidden Markov models of the form

$$\mathbf{x}_t \sim \pi_0(\mathbf{x}_0), \tag{6.1}$$

$$\mathbf{x}_t|\mathbf{x}_{t-1} \sim f_t(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{6.2}$$

$$\mathbf{y}_t|\mathbf{x}_t \sim g_t(\mathbf{y}_t|\mathbf{x}_t), \tag{6.3}$$

where $\mathbf{x}_t \in \mathcal{X}$ for $t \geq 0$, $\mathbf{y}_t \in \mathcal{Y}$ for $t \geq 1$, f_t is a Markov kernel on $\mathcal{X}$ and $g_t : \mathcal{Y} \times \mathcal{X} \to \mathbb{R}_+$ is the likelihood function. For convenience, assume $\mathcal{X} \subseteq \mathbb{R}^{D_x}$ and $\mathcal{Y} \subseteq \mathbb{R}^{D_y}$; however, the extension to general Polish spaces follows directly. The key inference problem in this model class is estimating is the *filtering distributions*, i.e. the posterior distributions of the hidden states $(\mathbf{x}_t)_{t\geq 0}$ given the observations $\mathbf{y}_{1:t}$ denoted as $(\pi_t(\mathbf{x}_t|\mathbf{y}_{1:t}))_{t\geq 1}$. This is commonly known as *Bayesian filtering* [Anderson and Moore, 1979; Särkkä, 2013].

Under assumptions of linearity and Gaussianity, the inference problem for the hidden states of HMMs can be solved analytically via the Kalman filter [Kalman, 1960]. However, inference for general HMMs of the form (6.1)–(6.3) with nonlinear, non-Gaussian transitions and likelihoods lacked a general, principled solution until the arrival of the particle filtering schemes [Gordon et al., 1993]. Particle filters (PFs) have become ubiquitous for Bayesian filtering in the general setting. In short, the PFs retain a weighted collection of Monte Carlo samples representing the filtering distribution $\pi_t(\mathbf{x}_t|\mathbf{y}_{1:t})$ and recursively approximate the sequence of distributions $(\pi_t)_{t\geq 0}$ using a particle mutation-selection scheme [Doucet et al., 2000] (*cf.* Section 2.5.3).

While PFs (and other inference schemes for HMMs) implicitly assume that the assumed model is well-specified, it is important to consider whether the proposed model class includes the true data-generating mechanism (DGM). In particular, for general state-space HMMs, misspecification can occur if the true dynamics of the hidden process significantly differ from the assumed model f_t, or if the true observation model is markedly different from the assumed likelihood model g_t, e.g. corruption by heavy-tailed noise. The latter case is of widespread interest within the field of *robust statistics* [Huber, 1981] and has recently attracted significant interest in the machine learning community [Futami et al., 2018].

When the true DGM cannot be modelled, one principled approach to address misspecification is Generalized Bayesian Inference (GBI) [Bissiri et al., 2016] (*cf.* Section 2.6). Recall that this approach views classical Bayesian inference as a loss minimisation procedure in the space of probability measures, a view first developed by Zellner [1988]. In particular, the standard Bayesian update can be derived from this view, where a loss function is constructed using the Kullback-Leibler (KL) divergence from the empirical distribution of the observations to the assumed likelihood [Bissiri et al., 2016]. The KL divergence is sensitive to outliers [Knoblauch et al., 2019], hence the overall inference procedure is not robust to observations that are incompatible with the assumed model. A principled remedy is to replace the KL divergence with an alternative discrepancy, such as the β-divergence, which

makes the overall procedure more robust [Cichocki and Amari, 2010] while retaining interpretability.

Previous work on robust particle filters has considered the handling of outliers, sensor failures and misspecification of the transition model [Pitt and Shephard, 1999; Maiz et al., 2009, 2012; Xu et al., 2013; Calvet et al., 2015; Teixeira et al., 2017; Hu et al., 2007; Akyildiz and Míguez, 2020]. However, these approaches are either based on problem-specific heuristic outlier detection schemes or make strong assumptions about the DGM to justify the use of heavy-tailed distributions [Xu et al., 2013]. This requires knowledge of the contamination mechanism that is implicitly embedded in the likelihood.

This chapter proposes a principled approach to robust filtering that does not impose additional modelling assumptions. The GBI approach of Bissiri et al. [2016] is adapted to the Bayesian filtering setting, where sequential Monte Carlo (SMC) methods for inference are developed. The performance of this approach is illustrated using the β-divergence, to mitigate the effect of outliers. This approach significantly improves the PF performance in settings with contaminated data, while retaining a general and principled approach to inference.

6.2 Background and Notation

6.2.1 Notation

The following notation is used throughout this chapter. The space of bounded, Borel measurable functions on $\mathcal{X}$ is denoted as $B(\mathcal{X})$. The Dirac measure located at $\mathbf{z}$ is denoted as $\delta_{\mathbf{z}}(\mathrm{d}\mathbf{x})$ and note that $f(\mathbf{z}) = \int f(\mathbf{x})\delta_{\mathbf{z}}(\mathrm{d}\mathbf{x})$ for $f \in B(\mathcal{X})$. Denote the Borel subsets of $\mathcal{X}$ as $\mathcal{B}(\mathcal{X})$ and the set of probability measures on $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$ as $\mathcal{P}(\mathcal{X})$. For a probability measure $\mu \in \mathcal{P}(\mathcal{X})$ and $\varphi \in B(\mathcal{X})$, write $\mu(\varphi) := \int \varphi(\mathbf{x})\mu(\mathrm{d}\mathbf{x})$. Given a probability measure μ, the notation is abused to denote its density with respect to the Lebesgue measure as $\mu(\mathbf{x})$.

6.2.2 Generalized Bayesian Inference (GBI)

Recall from Section 2.6 that Generalized Bayesian Inference (GBI) [Bissiri et al., 2016] extends standard Bayesian inference to the M-open setting, where the generative model is not well-specified. In particular, it shows that the standard Bayes update rule in (2.26) can be seen as a special case of the more general loss-based update rule in (2.92). This is presented again in the sequel in the notation of this chapter.

For a prior belief distribution π_0 and a loss function $\ell(\mathbf{x}, \mathbf{y})$ connecting the observations to the model parameters, one can obtain a posterior belief distribution $\pi_G(\mathbf{x})$ on the model parameters with the following rule:

$$\pi_G(\mathbf{x}) := \pi_G(\mathbf{x} \,|\, \mathbf{y}) = \pi_0(\mathbf{x}) \frac{G(\mathbf{x} \,|\, \mathbf{y})}{Z}, \tag{6.4}$$

with $G(\mathbf{x} \,|\, \mathbf{y}) := \exp(-\ell(\mathbf{x}, \mathbf{y}))$ and $Z := \int G(\mathbf{x} \,|\, \mathbf{y}) \pi_0(\mathbf{x}) \, \mathrm{d}\mathbf{x}$. Specifying $\ell(\mathbf{x}, \mathbf{y})$ as the cross-entropy (from the KL-divergence) of the assumed likelihood relative to the empirical distribution of the data recovers the standard Bayes update.

As noted before, the standard Bayes update is not robust to outliers due to the properties of KL divergence [Knoblauch et al., 2019]. Hence, substituting the cross-entropy with a more robust loss such as the β-cross-entropy [Futami et al., 2018], based on the β-divergence, can make the inference more robust. Specifically, in this setting the generalised Bayes update for the likelihood $g(\mathbf{y} \,|\, \mathbf{x})$ is written as

$$\pi(\mathbf{x}) = \pi_0(\mathbf{x}) \frac{G^\beta(\mathbf{y} \,|\, \mathbf{x})}{Z_\beta}, \tag{6.5}$$

where

$$G^\beta(\mathbf{y} \,|\, \mathbf{x}) = \exp\left(\frac{1}{\beta} g(\mathbf{y} \,|\, \mathbf{x})^\beta - \frac{1}{\beta + 1} \int g(\mathbf{y}' \,|\, \mathbf{x})^{\beta+1} \, \mathrm{d}\mathbf{y}' \right). \tag{6.6}$$

One can consider $G^\beta(\mathbf{y} \,|\, \mathbf{x})$ as a generalised likelihood, resulting from the use of a different loss function compared to the standard Bayes procedure. Here β is a hyperparameter that is selected depending on the degree of misspecification. In general $\beta \in (0, 1)$ and

$$\lim_{\beta \to 0} G^\beta(\mathbf{y} \,|\, \mathbf{x}) = g(\mathbf{y} \,|\, \mathbf{x}). \tag{6.7}$$

Thus, intuitively, β values near zero are suitable for mild model misspecification and β values near one are suitable when the assumed model is expected to significantly deviate from the true model. The experimental section devotes some attention to the selection of β and sensitivity analysis.

Generalised Bayesian updating enables robustification against outliers if a suitable divergence is chosen [Ghosh and Basu, 2016; Knoblauch et al., 2018, 2019].

6.2.3 Sequential Monte Carlo for HMMs

One can use SMC algorithms in Section 2.5.3 to conduct inference in HMMs of the form (6.1)–(6.3) where $\pi_0(\cdot)$ is a prior probability distribution on the initial state $\mathbf{x}_0$, $f_t(\mathbf{x}|\mathbf{x}')$ is a Markov transition kernel on $\mathcal{X}$ and $g_t(\mathbf{y}_t \,|\, \mathbf{x}_t)$ is the likelihood for observation $\mathbf{y}_t$. This assumes that the hidden process and observation processes are realised at discrete time intervals, *i.e.*, $\mathbf{x}_{1:T}$ represents the hidden process with $\mathbf{x}_t \in \mathcal{X}$ and $\mathbf{y}_{1:T}$ an observation process with $\mathbf{y}_t \in \mathcal{Y}$, and the observation sequence $\mathbf{y}_{1:T}$ is assumed to be fixed but otherwise arbitrary.

The typical interest in probabilistic models is the estimation of expectations of general test functions with respect to the posterior distribution, in this case, of the hidden process $\pi_t(\mathbf{x}_t \,|\, \mathbf{y}_{1:t})$ and the associated joint distributions $\mathsf{p}_t(\mathbf{x}_{0:t} \,|\, \mathbf{y}_{1:t})$. More precisely, given a bounded test function $\varphi \in B(\mathcal{X})$, the interest is in estimating integrals of the form

$$\pi_t(\varphi) = \int \varphi(\mathbf{x}_t)\pi_t(\mathbf{x}_t \,|\, \mathbf{y}_{1:t}). \tag{6.8}$$

Kalman filtering [Kalman, 1960; Anderson and Moore, 1979] can be used to obtain closed form expressions for $(\pi_t, \mathsf{p}_t)_{t\geq 0}$ if f_t and g_t are linear-Gaussian. However, for non-linear or non-Gaussian cases, the target distributions are almost always intractable, requiring an alternative approach, such as SMC methods [Doucet et al., 2000; Doucet and Johansen, 2011], which are known as Particle Filters (PFs) when employed in the HMM setting.

In a typical iteration, a PF method proceeds as follows: given a collection of samples $\{\mathbf{x}_{t-1}^{(i)}\}_{i=1}^N$ representing the posterior $\pi_{t-1}(\mathbf{x}_{t-1} \,|\, \mathbf{y}_{1:t-1})$, it first samples from a (possibly observation dependent) proposal $\bar{\mathbf{x}}_t^{(i)} \sim q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})$. It then computes weights for each sample (particle) $\bar{\mathbf{x}}_{t-1}^{(i)}$ in the collection for a given observation $\mathbf{y}_t$, evaluating its fitness with respect to the likelihood g_t as $\mathsf{w}_t^{(i)} \propto g_t(\mathbf{y}_t \,|\, \bar{\mathbf{x}}_t^{(i)}) \frac{f_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \mathbf{x}_{t-1}^{(i)})}{q_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t)}$, where $\sum_{i=1}^N \mathsf{w}_t^{(i)} = 1$. Finally, an optional re-sampling step [1] is used to prevent degeneracy, leading to $\mathbf{x}_t^{(i)} \sim \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\bar{\mathbf{x}}_t^{(i)}}(\mathrm{d}\mathbf{x}_t)$. One can then construct the empirical measure $\pi_t^N(\mathrm{d}\mathbf{x}_t \,|\, \mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^N \delta_{\mathbf{x}_t^{(i)}}(\mathrm{d}\mathbf{x}_t)$, and the estimate of $\pi_t(\varphi)$ in (6.8) is given by

$$\pi_t^N(\varphi) = \frac{1}{N}\sum_{i=1}^N \varphi(\mathbf{x}_t^{(i)}). \tag{6.9}$$

[1] In the simplest form, drawing N times with replacement from the weighted empirical measure to obtain an unweighted sample whose empirical distribution approximates the same target; see [Gerber et al., 2019] for an overview of resampling schemes and their properties.

Algorithm 6 The generalised particle filter

Input: Observation sequence $\mathbf{y}_{1:T}$, number of samples N, proposal distributions $q_{1:T}(\cdot)$.
Initialize: Sample $\{\bar{\mathbf{x}}_0^{(i)}\}_{i=1}^N$ for the prior $\pi_0(\mathbf{x}_0)$.
for $t = 1$ **to** T **do**
  Sample: $\bar{\mathbf{x}}_t^{(i)} \sim q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t)$ ▷ **for** $i = 1$ **to** N.
  Weight: $\mathsf{w}_t^{(i)} \propto \exp(-\ell(\bar{\mathbf{x}}_t^{(i)}, \mathbf{y}_t)) \frac{f_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \mathbf{x}_{t-1}^{(i)})}{q_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t)}$ ▷ **for** $i = 1$ **to** N.
  Resample: $\mathbf{x}_t^{(i)} \sim \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\bar{\mathbf{x}}_t^{(i)}}(\mathrm{d}\mathbf{x}_t)$ ▷ **for** $i = 1$ **to** N.
end for

If the proposal is chosen as the transition density, *i.e.*, $q_t(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t) = f_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}^{(i)})$, the bootstrap particle filter (BPF) [Gordon et al., 1993] is obtained. This corresponds to the simple procedure of sampling $\bar{\mathbf{x}}_t^{(i)}$ from $f_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}^{(i)})$, and setting its weight $\mathsf{w}_t^{(i)} \propto g_t(\mathbf{y}_t \,|\, \bar{\mathbf{x}}_t^{(i)})$.

6.3 Generalised Bayesian filtering

6.3.1 A simple generalised particle filter

As explained in Section 6.2.2, given a standard probability model comprised of the prior $\pi_0(\mathbf{x})$ and a likelihood $g(\mathbf{y} \,|\, \mathbf{x})$, the general Bayes update defines an alternative, generalised likelihood $G(\mathbf{y} \,|\, \mathbf{x})$. The sequence of generalised likelihoods, denoted as $G_t(\mathbf{y}_t|\mathbf{x}_t)$ for $t \geq 1$, in an HMM yields a joint generalised posterior density which factorises as

$$\mathsf{p}_t(\mathbf{x}_{0:t} \,|\, \mathbf{y}_{1:t}) \propto \pi_0(\mathbf{x}_0) \prod_{k=1}^{t} f_k(\mathbf{x}_k \,|\, \mathbf{x}_{k-1}) G_k(\mathbf{y}_k \,|\, \mathbf{x}_k), \tag{6.10}$$

where $G_t(\mathbf{y}_t \,|\, \mathbf{x}_t) := \exp(-\ell_t(\mathbf{x}_t, \mathbf{y}_t))$. Inference can be done via SMC applied to this sequence of twisted probabilities defining a Feynman-Kac flow [Del Moral, 2004].

Comparing the update rule in (6.4) to the standard Bayes update suggests a generalisation of the particle filter. In particular, under the model in (6.1)–(6.3), one can perform generalised inference using $(f_t)_{t\geq 1}$ as usual, but replacing the likelihood with $(G_t)_{t\geq 1}$. Hence, a generalised sequential importance resampling PF (given fully in Algorithm 6) keeps the sampling step intact, but applies a different weight computation step $\mathsf{w}_t^{(i)} \propto \exp(-\ell(\bar{\mathbf{x}}_t^{(i)}, \mathbf{y}_t)) \frac{f_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \mathbf{x}_{t-1}^{(i)})}{q_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_t)}$. Indeed, most PFs (including the APF, see Algorithm 8 in the appendix) and related algorithms can

be adapted to the GBI context.

6.3.2 The β-BPF and the β-APF

The β-BPF is derived by selecting $\ell_t(\mathbf{x}_t, \mathbf{y}_t)$ as the β-divergence and applying the BPF procedure with the associated generalised likelihood. In this case, the loss is

$$\ell_t^\beta(\mathbf{x}_t, \mathbf{y}_t) = \frac{1}{\beta+1}\int g_t(\mathbf{y}'_t \,|\, \mathbf{x}_t)^{\beta+1}\,\mathrm{d}\mathbf{y}'_t - \frac{1}{\beta}g_t(\mathbf{y}_t \,|\, \mathbf{x}_t)^\beta. \tag{6.11}$$

One can then construct the general β-likelihood as

$$G_t^\beta(\mathbf{y}_t \,|\, \mathbf{x}_t) \propto \exp(-\ell_t^\beta(\mathbf{x}_t, \mathbf{y}_t)). \tag{6.12}$$

In this instance, the use of the β-divergence provides the sampler with robust properties [Cichocki and Amari, 2010]. This can informally be seen from the form of the loss function in (6.11), where positive values of β temper the likelihood extending its tails making the loss more forgiving to outliers. The β-BPF procedure is given in Algorithm 7 in the appendix. The β-APF (Algorithm 8 in the appendix) is an Auxiliary Particle Filter [Pitt and Shephard, 1999; Johansen and Doucet, 2008] adapted to the GBI setting, and is derived similarly to the β-BPF.

Note that the integral term in (6.11) is independent of $\mathbf{x}_t$ and can be absorbed, without evaluation, into the normalising constant when $\mathbf{x}_t$ is a location parameter for a symmetric $g_t(\cdot)$ and $\mathcal{Y}$ is a linear subspace of $\mathbb{R}^{D_y}$ (recall that $\mathbf{y}_t \in \mathcal{Y}$). More generally, if $g_t(\cdot)$ is a member of the exponential family, the integral can be computed by identifying $g_t^\beta(\cdot)$ with the kernel of another member of the same family with canonical parameters scaled by β. The overhead of computing $G_t^\beta(\cdot)$ is negligible in this instance, which is not too restrictive in the context of misspecified models. For other likelihoods, unbiased estimators for $G_t^\beta(\cdot)$, e.g. Poisson estimator [Beskos et al., 2006], can be used in a random weight particle filter framework [Fearnhead et al., 2008], where the overhead of computing $G_t^\beta(\cdot)$ will depend on the variance of the estimator and the convergence results from this setting apply but as [Fearnhead et al., 2008] demonstrate this cost need not be prohibitive.

6.3.3 Selecting β

It is often the case that the primary goal of inference, particularly in the presence of model misspecification, is prediction. Hence, this thesis proposes choosing divergence parameters that lead to maximally predictive posterior belief distributions. In particular, for the β-BPF and β-APF, define $\mathcal{L}_\beta(\mathbf{y}_t, \check{\mathbf{y}}_t)$ as a loss function of the

observations $\mathbf{y}_t$ and the predictions $\check{\mathbf{y}}_t$. One can then choose β as the solution to the following decision-theoretic optimisation problem:

$$\min_{\beta} \mathsf{agg}_{t=1}^{T}(\mathbb{E}_{p(\check{\mathbf{y}}_t|\mathbf{y}_{1:t-1})}\mathcal{L}_{\beta}(\mathbf{x}_t, \check{\mathbf{x}}_t)), \tag{6.13}$$

where agg denotes an aggregating function. This approach requires some training data to allow the selection of β. In filtering contexts, this can be historical data from the same setting or other available proxies. For offline inference one could also employ the actual data within this framework. Since, this proposal relies on the quality of the observations, which in the case of outlier contamination is violated by definition. To remedy this, one can choose robust versions for agg and $\mathcal{L}$, e.g. the median and the (standardised) absolute error respectively.

6.4 Theoretical guarantees

Theoretical guarantees for SMC methods can be extended to the generalised Bayesian filtering setting. Since the generalised Bayesian filters can be seen as standard SMC methods with modified likelihoods, the same analytical tools can be used in this setting. This section provides guarantees for the β-BPF but emphasises that the same results can be obtained much more broadly (including for the β-APF via the approach of Johansen and Doucet [2008]). The generalised filters and generalised posteriors for the HMM in the β-divergence setting are denoted as π_t^{β} and p_t^{β} respectively. Consequently, corresponding quantities constructed by the β-BPF are denoted as $\pi_t^{\beta,N}$ and $\mathsf{p}_t^{\beta,N}$.

Although the generalised likelihoods $G_t^{\beta}(\mathbf{y}_t \,|\, \mathbf{x}_t)$ are not normalised, they can be considered as potential functions [Del Moral, 2004]. Since $G_t^{\beta}(\mathbf{y}_t \,|\, \mathbf{x}_t) < \infty$ whenever $g_t(\mathbf{y}_t \,|\, \mathbf{x}_t) < \infty$ and β is fixed, one can adapt the standard convergence results into the generalised case.

Assumption 1. *For a fixed arbitrary observation sequence $\mathbf{y}_{1:T} \in \mathcal{Y}^T$, the potential functions $(G_t^{\beta})_{t\geq 1}$ are bounded and*

$$G_t^{\beta}(\mathbf{y}_t \,|\, \mathbf{x}_t) > 0, \quad \forall t \in \{1, \ldots, T\} \quad \textit{and} \quad \mathbf{x}_t \in \mathcal{X}.$$

This assumption holds for most used likelihood functions and their generalised extensions.

Theorem 6.4.1. *For any $\varphi \in B(\mathcal{X})$ and $p \geq 1$,*

$$\|\pi_t^{\beta,N}(\varphi) - \pi_t^{\beta}(\varphi)\|_p \leq \frac{c_{t,p,\beta}\|\varphi\|_\infty}{\sqrt{N}}, \tag{6.14}$$

where $c_{t,p,\beta} < \infty$ is a constant independent of N.

The proof sketch and the constant $c_{t,p,\beta}$ are given in Appendix D.2.1. This L_p bound provides a theoretical guarantee on the convergence of particle approximations to generalised posteriors. The special case when $p = 2$ also provides the error bound for the mean-squared error. It is well known that Theorem 6.4.1 with $p > 2$ leads to to a law of large numbers via Markov's inequality and a Borel-Cantelli argument:

Corollary 6.4.2. *Under the setting of Theorem 6.4.1,*

$$\lim_{N\to\infty} \pi_t^{\beta,N}(\varphi) = \pi_t^{\beta}(\varphi) \quad \text{a.s.,} \quad \text{for} \quad t \geq 1. \tag{6.15}$$

Finally, a central limit theorem for estimates of expectations with respect to the smoothing distributions can be obtained by considering the path space $\mathcal{X}^{\otimes t}$. Recall the joint posterior $\mathsf{p}_t^{\beta}(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ and consider a test function $\varphi_t : \mathcal{X}^{\otimes t} \to \mathbb{R}$. Denote $\overline{\varphi}_t^{\beta} := \int \varphi_t^{\beta}(\mathbf{x}_{1:t})\mathsf{p}_t^{\beta}(\mathbf{x}_{1:t}|\mathbf{y}_{1:t})$ and denote the β-BPF estimate of $\overline{\varphi}_t$ with $\overline{\varphi}_t^{\beta,N} := \int \varphi_t(\mathbf{x}_{1:t})\mathsf{p}_t^{\beta,N}(\mathrm{d}\mathbf{x}_{1:t})$.

Theorem 6.4.3. *Under the regularity conditions given in [Chopin, 2004, Theorem 1],*

$$\sqrt{N}\left(\overline{\varphi}_t^{\beta,N} - \overline{\varphi}_t^{\beta}\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma_{t,\beta}^2(\varphi_t)\right),$$

as $N \to \infty$ where $\sigma_{t,\beta}^2(\varphi_t) < \infty$.

The expression for $\sigma_{t,\beta}^2(\varphi_t)$ is given in Appendix D.2.1. These results illustrate that the standard guarantees for generic particle filtering methods extend to the generalised case.

6.5 Experiments

This section focuses on β-BPF illustrating its the properties and empirically verifying its robustness. It includes four experiments in both linear and non-linear settings and for different types of contamination. Furthermore, Section 6.5.2 specifically investigates the β-APF comparing its behaviour to the β-BPF. Throughout, the *normalised mean squared error (NMSE)* and the *90% empirical coverage* are

used as goodness-of-fit measures. The NMSE scores indicate the mean fit for the inferred posterior distribution and the empirical coverage measures the quality of its uncertainty quantification. Any claim in performance difference is based on the Wilcoxon signed-rank test. Further results and in-depth details of the experimental setup are given in Appendix D.4 and Appendix D.3 respectively.

6.5.1 A Linear-Gaussian state-space model

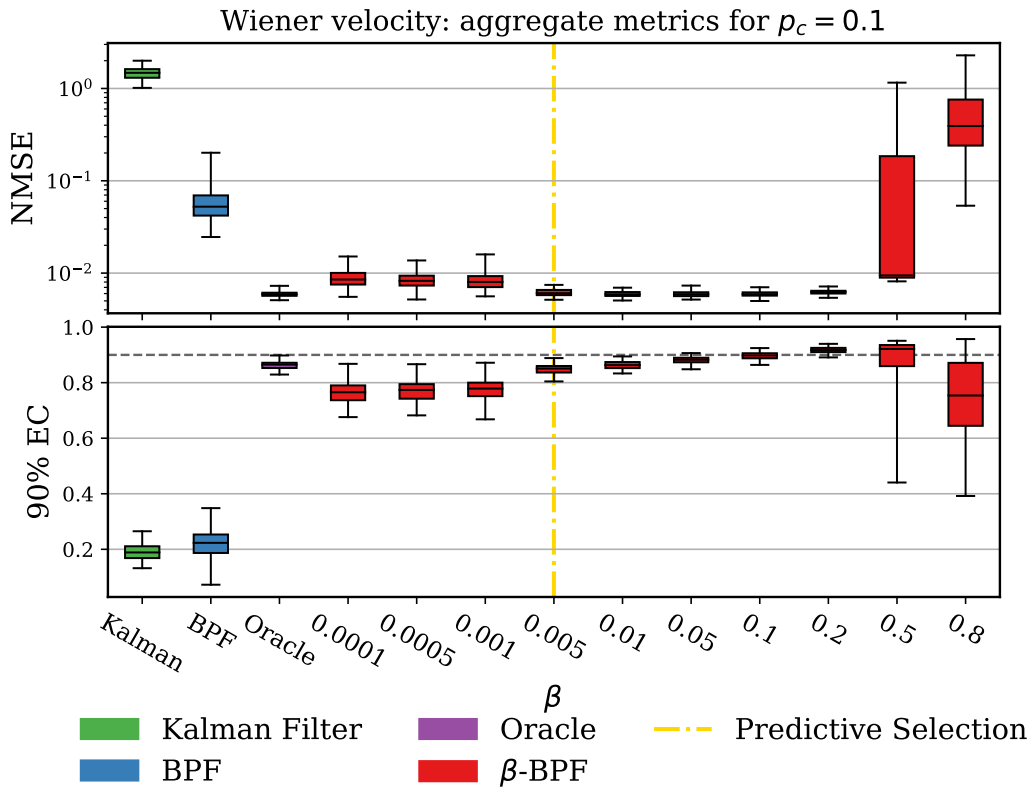


Figure 6.1: The mean metrics over state dimensions for the Wiener velocity example with $p_c = 0.1$. The top panel presents the NMSE results (lower is better) and the bottom panel presents the 90% empirical coverage results (higher is better), on 100 runs. The vertical dashed line in gold indicate the value of β chosen by the selection criterion in Section 6.3.3. The horizontal dashed line in black in the lower panel indicates the 90% mark for the coverage.

The Wiener velocity model [Särkkä and Solin, 2019] is a standard model in the target tracking literature, where the velocity of a particle is modelled as a Wiener process. The discretised version of this model can be represented as a

Linear-Gaussian State-Space model (LGSSM),

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \boldsymbol{\nu}_{t-1}, \qquad \boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{6.16}$$

$$\mathbf{y}_t = \mathbf{H}\mathbf{x}_t + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma), \tag{6.17}$$

where $\mathbf{A}, \mathbf{Q}$ are state-transition parameters dictated by the continuous-time model and $\mathbf{H}$ is the observation matrix (see Appendix). This model was simulated in two-dimensions with $\Sigma = \mathbf{I}$, contaminating the observations with a large scale, zero-mean Gaussian, $\mathcal{N}(0, 100^2)$ with probability p_c. The aim is to obtain the filtering density under the heavily-contaminated setting where optimal filters struggle to perform. This scheme compares the proposed β-BPF, for a range of values for β, to the standard BPF with a Gaussian likelihood (BPF), the (optimal) Kalman filter and an Oracle BPF with likelihood corresponding to the true generative model, *i.e.*, with a Gaussian mixture likelihood with mixture components matching the noise processes and mixture probabilities matching contamination probability.

The goal of this section is to answer four questions in this simple setup: (a) Does the β-BPF produce accurate and well-calibrated posterior distributions in the presence of contaminated data? (b) Is it sensitive to the choice β? (c) Does the method described in Section 6.3.3 for selecting β return a near-optimal result? (d) How does the robustification procedure compare to the inference with knowledge of the true model?

Fig. 6.1 shows the results for $p_c = 0.1$. Observe that (a) the β-BPF outperforms the Kalman filter and the standard BPF for $\beta \leq 0.2$ while producing well-calibrated posteriors accounting for the uncertainty (for $\beta \in [0.01, 0.2]$ the coverage approaches the 90% threshold), (b) drastic performance gains are seen (with median NMSE scores around 10× smaller than the BPF and 100× smaller that the Kalman filter) for a large range of β values, (c) the β-choice heuristic [2] chooses a well-performing β (gold vertical lines in Fig. 6.1), and (d) the performance of the β-BPF is very close the Oracle (with knowledge of the true model) for a range of β values. Note that, for most values of β, the β-BPF significantly outperforms both the Kalman filter and the standard BPF predictively. The full set of results for the predictive performance are presented in Table D.1 in Appendix D.4.1.

[2] This choice criterion is applied on an alternative dataset that is obtained from the same simulation but with 90% fewer observations.

6.5.2 Terrain Aided Navigation

Terrain Aided Navigation (TAN) is a challenging estimation problem, where the state evolution is defined as in (6.16) (in three dimensions), but with a highly non-linear observation model,

$$\mathbf{y}_t = h(\mathbf{x}_t) + \boldsymbol{\epsilon}_t, \tag{6.18}$$

where $h(\cdot)$ is a non-linear function, typically including a non-analytic Digital Elevation Map (DEM). This problem simulates the trajectory of an aeroplane or a drone over a terrain map, where its elevation is observed over the terrain and its distance from its take-off hub from on-board sensors (see supplement for more details). For this example, transmission failure of the measurement system was simulated as impulsive noise on the observations, i.e., i.i.d. draws from a Student's t distribution with $\nu = 1$ degrees of freedom, *i.e.*, $\boldsymbol{\epsilon}_t \sim (1 - p_c)\mathcal{N}(0, 20^2) + p_c t_{\nu=1}(0, 20^2)$.

Both the β-BPF and the β-APF are applied to this problem and are compared to the standard BPF with the Gaussian (BPF) and two other robust PF methods from the literature: Student's t (t-BPF) [Xu et al., 2013] and the APF [Pitt and Shephard, 1999]. The degrees of freedom for the t-BPF is set to the same value as the contamination $\nu = 1$.

From Fig. 6.2, for low contamination, both the β-BPF and the β-APF outperform the standard Gaussian BPF, the t-BPF and the APF. This shows that the use of t-distribution for the low contamination setting is inappropriate. This gap in the performance tightens, naturally, as p_c grows since t-distribution becomes a good model for the observations. Notably, the performance gaps between the standard PFs and their β-robustified counterparts are similar, indicating that the use of the β-divergence in a particle filtering procedure does indeed robustify the inference.

Figure 6.3 shows the filtering distributions for the sixth state dimension (vertical velocity) obtained from an illustrative run with $p_c = 0.1$. The left panel shows the filtering distributions from the (Gaussian) BPF (up) and the β-BPF (down). The locations of the most prominent outliers are marked with dashed vertical lines in black. Fig. 6.3 displays the significant difference between the two approaches: while the uncertainty for the standard BPF collapses when it meets the outliers, e.g. around $t = 1700$, the β-BPF does not suffer from this problem. This performance difference is partly related to the stability of the weights. The right panel in Fig. 6.3 demonstrates the effective sample size (ESS) with time for the two filters showing that the β-BPF consistently exhibits larger ESS values, avoiding particle degeneracy. The ESS values for the BPF, on the other hand, sharply decline when

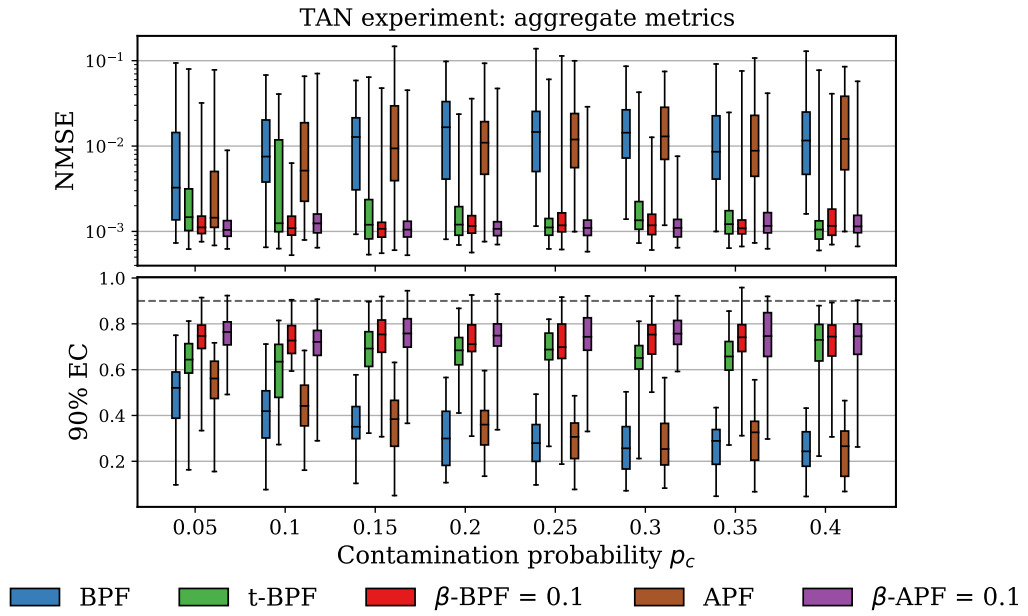


Figure 6.2: The mean metrics over state dimensions for the TAN example for different p_c. The top panel presents the NMSE results (lower is better) and the bottom panel presents the 90% empirical coverage results (higher is better), both evaluated on 50 runs. The horizontal dashed line in black in the lower panel indicate the 90% mark for the coverage.

it meets outliers. A similar result is observed for the APF versus the β-APF in the figures in the Appendix D.4.2. Further results on predictive performance can be found in Appendix D.4.2.

6.5.3 Asymmetric Wiener Velocity

In the case of simple, symmetric noise settings with additive contamination the use of heavy-tailed likelihoods such as Student's t may be still seen as a viable alternative to robustify the inference. However, there are some realistic settings in which such off-the-shelf heavy-tailed replacements are not feasible or require considerable model-specific work. Consider, as a simple illustration, the Wiener velocity example in Section 6.5.1, where the observation noise in (6.17) is replaced with

$$\boldsymbol{\epsilon}_t \sim \mathbb{1}_{[-\infty,0]}\mathcal{N}(0,1) + \mathbb{1}_{[0,+\infty]}\mathcal{N}(0,10^2). \tag{6.19}$$

This simulates an asymmetric noise scenario. The observations are further contaminated with *multiplicative* exponential noise, i.e. $\boldsymbol{\epsilon}_t \leftarrow \xi\boldsymbol{\epsilon}_t$, for $\xi \sim \text{Exp}(1000)$ with

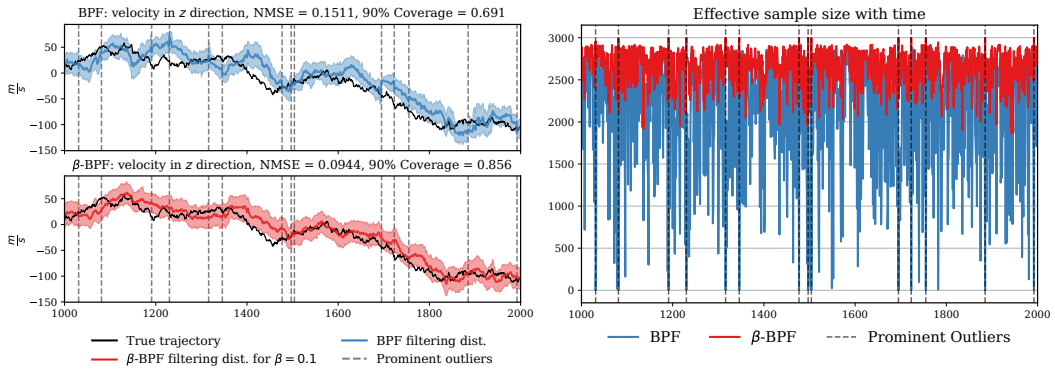


Figure 6.3: The left panel shows the inferred marginal filtering distributions for the velocity in the z direction for the BPF and β-BPF with $\beta = 0.1$. The right panel shows the effective sample size with time. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black in both panels.

probability p_c. This sums up to a multiplicatively corrupted asymmetric noise distribution which could, for example, represent a sensor with asymmetric noise profile in a failing regime which occasionally exhibits excessive gain.

For this example, it is easy to derive a BPF with the asymmetric likelihood. It is also easy to extend this likelihood to the β-BPF case. Thus the BPF and the β-BPF ($\beta = 0.1$) are tested versus two versions of the t-BPF, where the t-likelihood is set to a short scale, $\sigma = 1$, in one version and to a long scale, $\sigma = 10$, in the other version.

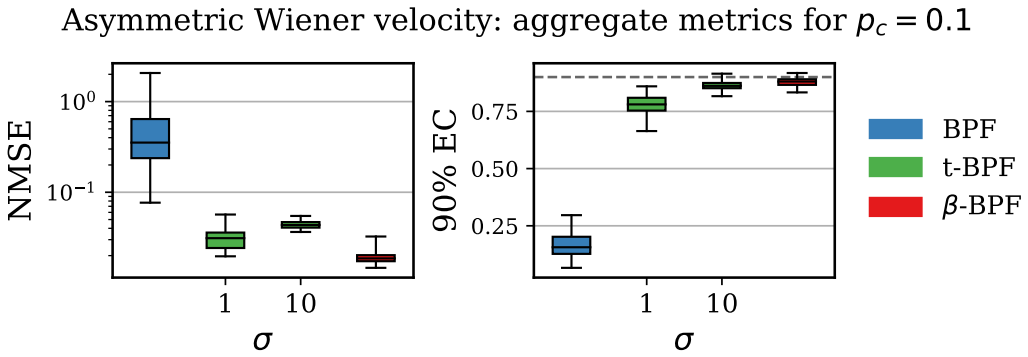


Figure 6.4: The mean metrics over state dimensions for the asymmetric Weiner velocity example with $p_c = 0.1$. The left panel presents the NMSE results (lower is better) and the right panel presents the 90% empirical coverage results (higher is better), evaluated on 100 runs. The x-axis ticks indicate the scale used for Student's t likelihood. The horizontal dashed line in black in the right panel indicates the 90% mark for the coverage.

Fig. 6.4 shows the results for this experiment. The BPF is unable to handle the multiplicative exponential contamination, as can be seen by the NMSE values. It also provides poor posterior coverage. The t-BPF fares better with this type of contamination where one can see a trade-off between accuracy and coverage depending on the chosen scale of the likelihood. This is due to the symmetry of the t-distribution which overestimates one of the tails depending on the scale. The β-BPF does not have this trade-off and outperforms the t-BPF on both metrics.

While one might attempt to model the noise with an asymmetric construction of the t-distribution which approximates the noise structure, in more general settings using heavy-tailed distributions requires approximations of the noise structure and making modelling choices which could be arbitrarily complex. This is in contrast to specifying a single tuning parameter as in the β-divergence case. The β-BPF requires no further modelling than the original problem and can be used as a drop-in replacement for nearly all types of likelihood structures.

6.5.4 London air quality Gaussian process regression

To measure air quality, London authorities use a network of sensors around the city recording pollutant measurements. Sensor measurements are susceptible to significant outliers due to environmental effects, manual calibration and sensor deterioration. This experiment uses Gaussian process (GP) regression to infer the underlying signal from a PM2.5 sensor.

For 1-D time series data, GP inference [Rasmussen and Williams, 2006] can be accelerated to linear time in the number of observations by formulating an equivalent stochastic differential equation whose solution precisely matches the GP under consideration [3] [Särkkä et al., 2013]. The resulting model is a LGSSM of the form (6.16)–(6.17) where the smoothing distribution matches the GP marginals at discrete-times. One can then apply smoothing algorithms, such as Rauch Tung Striebel (RTS) [Rauch et al., 1965] or Forward Filters Backward Smoothing (FFBS) [Briers et al., 2010], to obtain the GP posterior. These require a forward filtering step with the Kalman filter for RTS or a PF for FFBS. A Matérn 5/2 GP with fixed hyperparameters was fit to a time series from one of the sensors. The median of the signals from the wider sensor network was computed to obtain a simple approximation of the ground truth.

Section 6.5.4 compares results with a Gaussian likelihood for GP regression with Kalman (RTS) smoothing, the standard BPF (FFBS) and two runs for the

[3] The SDE representation of a GP depends on the form of the covariance function. This experiment uses a GP with the Mateŕn 5/2 kernel, which admits a dual SDE representation.

β-BPF (FFBS) ($\beta = 0.1$ by predictive selection as Section 6.3.3 and $\beta = 0.2$ by overall best performance). For both choices of β, the β-BPF outperforms all other methods on both metrics .

	median (IQR)	
Filter (Smoother)	NMSE	EC
Kalman (RTS)	0.144(0)	0.685(0)
BPF (FFBS)	0.116(0.015)	0.650(0.020)
($\beta = 0.1$)-BPF (FFBS)	0.061(0.003)	0.760(0.015)
($\beta = 0.2$)-BPF (FFBS)	**0.059**(**0.002**)	**0.803**(**0.020**)

Table 6.1: GP regression NMSE (lower is better) and 90% empirical coverage for the credible intervals of the posterior predictive distribution, on 100 runs. **Bold** indicates statistically significant best result from Wilcoxon signed-rank test. All presented results are statistically different from each other according to the test.

To further investigate the GP solution of the β-BPF (FFBS), Fig. 6.5 shows the fit for $\beta = 0.1$ in comparison to Kalman (RTS) smoothing. The latter is sensitive to outliers forcing the GP mean towards them while the β-BPF is robust and ignores them.

6.6 Concluding Remarks

This chapter addressed the problem of robustness in sequential probabilistic models. It presented a generalised filtering framework based on GBI, which tackles likelihood misspecification in general state-space HMMs. This approach leveraged SMC methods, where analytical results were extended to the generalised case. The β-BPF and β-APF were presented as simple instantiations of this approach based on the β-divergence which provide the inference with robustness against observation contamination. While standard algorithms failed in both linear and non-linear settings when met with outliers, the proposed filters exhibited robustness against them showing significant gains in performance.

Throughout this chapter, the model parameters were assumed to be fixed. Relaxing this assumption opens up the question of the online learning of the model parameters (system identification) in the presence of misspecification. The presented framework can directly incorporate most estimators found in the SMC literature, and the computation of derivatives can be tackled with automatic differentiation tools. However, there is still a need to validate this proposition and characterise the

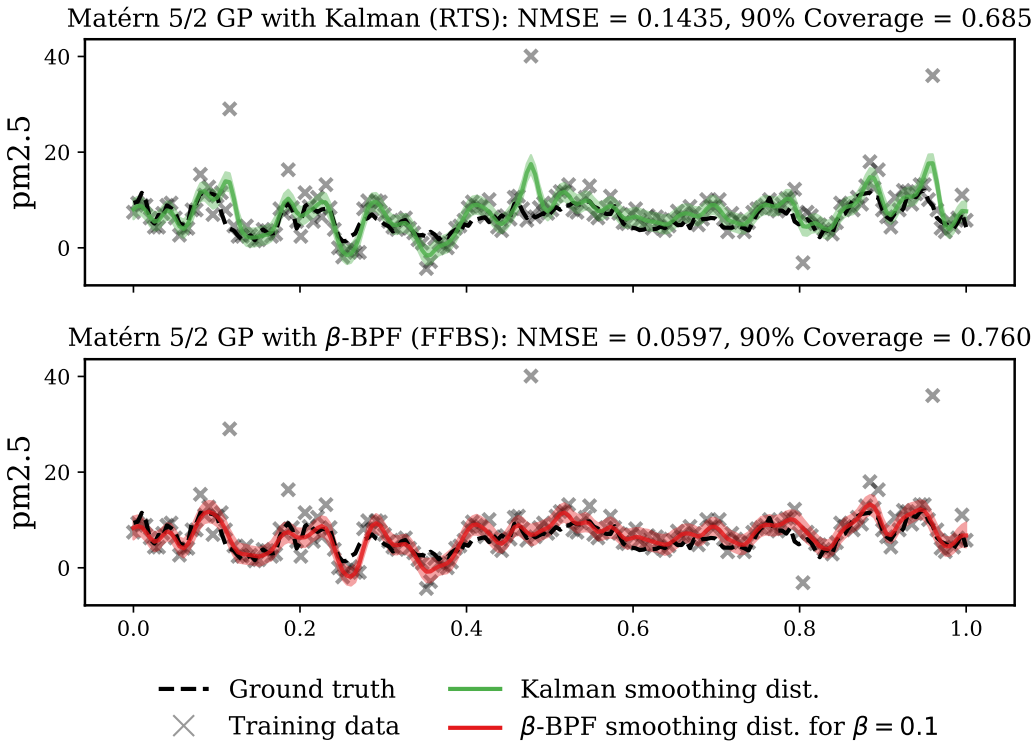


Figure 6.5: The GP fit on the measurement time series for one of the London air quality sensors. The top panel shows the posterior from the Kalman (RTS) smoothing. The bottom panel shows the posterior from the β-BPF (FFBS) for $\beta = 0.1$. The Kalman (RTS) solution is sensitive to the outliers forcing the GP mean towards them, while the β-BPF (FFBS) solution is robust.

effect of data contamination on the quality of parameter estimation.

The code for this chapter is publicly available at https://github.com/aboustati/robust-smc.

CHAPTER 7

Conclusions

The goal of this thesis was to contribute to the field of probabilistic machine learning by formulating methods that endow probabilistic models with compositionality, stability and robustness. The focus on these properties was motivated by the desire to make probabilistic models more flexible (compositionality) and more reliable (stability and robustness). This thesis presented four novel ideas addressing these issues and demonstrating their importance on a range of experimental settings.

This section summarises the contributions of this thesis and signposts directions for future research.

7.1 Summary of Contributions

Addressing compositionality, this thesis studied this property in GP models in Chapter 3. It formulated a non-linear multitask learning framework based on the composition of GP modules to yield a multitask version of the DGP. It showed that, due to the compositional nature of this model, approximate inference can be conducted by simply extending standard inference procedures for GPs. Furthermore, it showed how this compositional framework relates to other ideas in the literature such as linear compositions and regularisation. The experimental results showed the importance of compositionality in multi-task learning problems, where the proposed approach outperformed others on a variety of benchmarks.

Chapter 4 and Chapter 5 examined the problem of stability and contributed two methods for producing low variance gradient estimators for VI. Chapter 4 studied the stability in the generic case of doubly stochastic objectives, an important type of objectives in VI. It proposed amortising the computation of the control variate coefficients with a recognition network that cheaply approximates the optimal control variate coefficients per mini-batch. One of the main contributions of this chapter was to derive various computationally feasible objective functions for training the

recognition network. The control variate method of this chapter was illustrated on linear and quadratic control variates, with experiments showing successful variance reduction that leads to the stabilisation of the optimisation procedure.

Chapter 5 addressed the problem of reducing the variance of score function gradients by optimising a proxy objective function whose gradients match that of the ELBO in VI. The new gradient estimator was shown to have low variance, thus inducing stability in VI, especially for models with a discrete latent structure. The new estimator was proven to approximate the optimal score-based control variate for Reinforce gradients. The experimental results showed its effectiveness in stabilising the training of Discrete Variational Auto-Encoders.

Chapter 6 dealt with robustness in sequential models. Through the lens of GBI, it described a novel framework to perform robust inference with SMC algorithms. The use of the β-divergence as a loss function for GBI provided PFs with robustness against contaminated data. The empirical results showed the importance of robust inference in real-world applications, where non-robust methods were unable to produce a good fit on a range of datasets.

7.2 Future Research Directions

The work in this thesis creates many avenues for future research in the topics of compositionality, stability and robustness in probabilistic models.

The Big Picture The end goal of studying the three topics is to transition beyond machine learning models to machine learning *systems*. Machine learning systems integrate all aspects of the modelling pipeline, from data-preparation to decision-making, into an end-to-end system. Two important questions come to light from this perspective: How can the propagation of uncertainty, from one stage of the pipeline to the next, be handled? How can the modularity of the system be insured? The adoption of compositional probabilistic models as building blocks addresses both questions, but consideration should be given to some of the challenges that arise from their use. For instance, the propagation of uncertainty relies on the quality of uncertainty estimation, in each component, at each stage. Hence, an important area of investigation is the study of reliable inference procedures that can produce high-quality posterior approximations. The modern formulation of VI is appealing in this setting, as it provides a simple optimisation-driven interface for inference between the different parts of the system. However, the use of simple parametric approximations to the posterior distributions often hinders the effectiveness of VI

and the quality of the posterior uncertainty that it outputs. Although, there has recently been significant progress on devising more expressive approximations, such as Normalising Flows [Rezende and Mohamed, 2015], Hierarchical VI posteriors [Ranganath et al., 2016b] and Implicit Mixture distributions [Domke and Sheldon, 2018]. These approximations are more flexible than standard parametric VI approximations, but suffer from additional computational overhead. In Chapter 3, the use of the Variational Sparse Approximation illustrates this problem. The parametric mean-field assumption on the DGP posterior enables efficient inference, but is unable to capture the covariance between the modules, both within and across the DGP layers. A potential solution to this problem is to correlate the mean-field components of the approximate posterior with parameterised copulas that can learn some of the correlation between the modules. This is an example of the Hierarchical VI posteriors mentioned above.

One can also look to a different solution to the inference problem than VI. Markov Chain Monte Carlo (MCMC) has been a mainstay in approximate inference in probabilistic machine learning since the inception of the field [Neal, 1996]. While MCMC is considered the "gold standard" in approximate inference, its use in large compositional models has been limited due to its substantial computational requirements, especially with large datasets. Recently, however, there have been many developments towards reducing this cost by devising new sampling algorithms [Welling and Teh, 2011; Chen et al., 2014] or extending current algorithms to work with data sub-sampling [Quiroz et al., 2019]. This culminated in the wide adoption of MCMC sampling algorithms for training Bayesian deep neural networks [Wenzel et al., 2020], another paradigm of compositional models. While it is difficult to predict what the future might hold for these different algorithmic paradigms, it is clear that the developments in computer hardware will have a big influence on their adoption for inference in machine learning systems.

Stability also plays an important role in the systems view of machine learning. This thesis's contributions on stability were mainly focused on reducing the variance of the ELBO gradient in VI. The method of control variates was used to motivate the two contributions on stability. However, the use of control variates as a stabilising mechanism is not restricted to optimisation-based inference methods like VI. In fact, the method was first conceived to stabilise sampling algorithms such as MCMC, and there has recently been a large body of work exploring its use for large-scale and high-dimensional inference problems [Müller et al., 2020; Si et al., 2020]. For instance, the amortised control variates method in Chapter 4 was an example of an adaptive optimiser [Andrychowicz et al., 2016], but similar adaptive sampling

algorithms can be constructed for other inference paradigms. Regardless of the inference procedure, the study of inference stability is essential in developing reliable probabilistic machine learning systems.

Furthermore, the ideas on stability that were presented in this thesis are only the first steps towards developing more stable inference procedures for large-scale, modular and end-to-end systems. In this thesis, these ideas were applied to a single model. However, the application of similar types of stabilising mechanisms to multiple models and datasets can be very beneficial, as it could enable efficient and reliable retraining of large machine learning systems, making their deployment more viable.

Finally regarding robustness, this thesis only addressed the problem of robust inference under data contamination, but this is only one example of misspecification that can occur in real-world scenarios. The performance of probabilistic machine learning systems depends on the type of unseen situations, especially those involving adversarial attacks, dataset shifts or model misspecification. The use of the β-divergence in the GBI framework was shown to be effective against the data contamination setting; however, the effectiveness of this framework for other misspecification scenarios was not considered. This question paves the way for another line of future research that investigates the efficacy of alternative probabilistic inference frameworks such as GBI to other misspecification scenarios.

APPENDIX A

Appendix for Multitask Learning with Gaussian Process Compositions

A.1 Experiment Details

A.1.1 MNIST Variations

This section the experimental setup used to obtain the results in Section 3.5.1.

Data The three background datasets from https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits was used. For each dataset, 1000 images were sampled from the training split and combined into a single 3 task multi-class classification dataset. For testing, the whole test split for each task was used.

Experimental procedure The training dataset sub-sampling procedure was repeated 10 times to obtain 10 different training datasets. The models were fit on each of the of the training datasets and the average accuracy and its standard error on the test split were reported.

GP models initialisation All GP models use the Matérn-5/2 covariance function. For the covariance functions that act on the data, the kernel parameters are initialised to the same values for all the models. The lengthscale is set to $l = 40$ and the signal variance $v = 30$. One exception is the mMDGP model where the kernel parameters in the shared component are initialised as before, but the task specific component is initialised to $l = 20$ and $v = 60$. This is done to break the

symmetry with the shared component. Additionally, for the deep models, the kernel parameters on the top layer are initialised to $l = 20$ and $v = 30$. All models use the same initial set of inducing inputs per task, initialised as the k-means centroids from the training data. Each task admits 50 inducing inputs. Deep models have an inner layer of size 30.

Neural network architecture All neural networks have inner layers with 128 neurons with ReLU activations. The standard networks use dropout with probability 0.2 and $L2$ regularisation. The Bayesian neural networks use a standard normal prior on the weights. All weights are initialised with Glorot initialisation.

Optimisation Shallow GP models use L-BFGS. Deep GP models use Adam with a learning rate 0.01 run for 10000 iterations. Mini-batching is used on the DGP models with a mini-batch size of 100. The standard neural networks use Adam with a learning rate of 0.001, 10000 iterations and mini-batch size of 64. The Bayesian neural networks use Adam with learning rate 0.0001, 10000 iterations and mini-batch size of 64.

A.1.2 SARCOS Robot Inverse Dynamics

This section details the experimental setup used to obtain the results in Section 3.5.2.

Data The dataset from `http://www.gaussianprocess.org/gpml/data/` was used. This dataset relates to the inverse dynamics problem for a seven degrees-of-freedom anthropomorphic robot arm [Vijayakumar et al., 2002; Rasmussen and Williams, 2006]. The dataset consists of 44,484 training observations with 21 variables (7 joint positions, 7 joint velocities, 7 joint accelerations) and the corresponding real-valued torques for 7 joints, i.e. 7 tasks corresponding to each joint torque. Additionally, there are 4,449 testing examples with all the 7 joint torques available for each.

Experimental procedure For N in $\{100, 200, 500, 1000, 2000, 5000\}$, N training data points are sampled from the training set. This constitutes the training set for one experimental run. For each of the N sampled points select 1 of the 7 labels uniformly at random. Split the experimental training set into 7 according to which joint the label corresponds to. These 7 splits constitute 7 tasks. The models are trained on the training set after standardising the features and the targets and test for all 7 labels on the test set. For each N, the experimental procedure is repeated for 10 times.

GP models initialisation All GP models use the Matérn-5/2 covariance function. For the covariance functions that act on the data, the kernel parameters are initialised to the same values for all the models. The lengthscale is set to $l = 10$ and the signal variance $v = 1$. One exception is the mMDGP model where the kernel parameters in the shared component are initialised as before, but the task specific component is initialised to $l = 10$ and $v = 0.5$. This is done to break the symmetry with the shared component. Additionally, for deep models, the kernel parameters on the top layer are initialised to $l = 10$ and $v = 1$. All models use the same initial set of 100 inducing inputs distributed per task, initialised by random sampling from the training data. The likeihood noise variance is initialised to $\sigma^2 = 10^{-6}$. Deep models have an inner layer of size 10.

Neural network architecture All neural networks have inner layers with 128 neurons with ReLU activations. The standard networks use dropout with probability 0.2 and $L2$ regularisation. The Bayesian neural networks use a standard normal prior on the weights. All weights are initialised with Glorot initialisation.

Optimisation Shallow GP models use L-BFGS for dataset sizes less than or equal to 1000, and Adam with the same settings as deep models otherwise. DGP models use Adam with a learning rate 0.01 run for 10000 iterations. Mini-batching is used on the DGP models when the dataset size is greater than 1000 with a mini-batch size of 500. The standard neural networks use Adam with a learning rate of 0.0001, 10000 iterations and mini-batch size of 64. The Bayesian neural networks use Adam with learning rate 0.0001, 10000 iterations and mini-batch size of 64.

A.1.3 FAIMS Diabetes Diagnosis

This section details the experimental setup used to obtain the results in Section 3.5.3.

Data The data from a case-control study of 125 patients who have been tested for diabetes was used. 48 out of 125 have been found to have diabetes (the disease group), while the rest are disease-free (the control group). The data consists of three experimental runs per patient, corresponding to sequential FAIMS analyses on the same urine sample. Each experimental run is treated as a task, i.e. three binary classification tasks in total.

Experimental procedure Sparse Principal Component Analysis decomposition was performed on the features selecting the first 20 principal components. 10-fold

cross-validation was performed on 70:30 train-test splits.

GP models initialisation All GP models use the Matérn-5/2 covariance function. For the covariance functions that act on the data, the kernel parameters are initialised to the same values for all the models. The lengthscale is set to $l = 0.5$ and the signal variance $v = 1.5$. One exception is the mMDGP model where the kernel parameters in the shared component are initialised as before, but the task specific component is initialised to $l = 1.5$ and $v = 1.5$. This is done to break the symmetry with the shared component. Additionally, for deep models, the kernel parameters on the top layer are initialised to $l = 1$ and $v = 2$. All models use the same initial set of inducing inputs per task, initialised as the k-means centroids from the training data. Each task admits 30 inducing inputs. Deep models have an inner layer of size 3.

Neural network architecture All neural networks have inner layers with 128 neurons with ReLU activations. The standard networks use dropout with probability 0.2 and $L2$ regularisation. The Bayesian neural networks use a standard normal prior on the weights. All weights are initialised with Glorot initialisation.

Optimisation Shallow GP models use L-BFGS. DGP models use Adam with a learning rate 0.001 run for 20000 iterations. The standard neural networks use Adam with a learning rate of 0.0001, 20000 iterations and mini-batch size of 64. The Bayesian neural networks use Adam with learning rate 0.0001, 20000 iterations and mini-batch size of 64.

A.2 Further Results

A.2.1 MNIST Variations

Table A.1: Classification accuracy on the MNIST variations experiment using all three tasks. Higher is better

	Accuracy									
Task	sMDGP	mMDGP	cMDGP	iDGP	cGP	iGP	mANN2	mANN3	mBNN2	mBNN3
Standard MNIST	**0.89**(**0.0**)	0.87(0.00)	0.80(0.08)	0.85(0.00)	0.87(0.00)	0.87(0.00)	0.88(0.00)	0.88(0.00)	0.83(0.00)	0.83(0.00)
Random BG MNIST	**0.64**(**0.0**)	0.61(0.01)	0.61(0.03)	0.18(0.00)	0.59(0.04)	0.19(0.00)	0.59(0.00)	0.59(0.00)	0.51(0.01)	0.49(0.01)
Images BG MNIST	**0.78**(**0.0**)	0.74(0.00)	0.70(0.07)	0.19(0.01)	0.58(0.06)	0.21(0.01)	0.62(0.00)	0.61(0.00)	0.53(0.01)	0.53(0.01)
All	**0.77**(**0.0**)	0.74(0.00)	0.70(0.06)	0.41(0.01)	0.68(0.04)	0.42(0.00)	0.70(0.00)	0.69(0.00)	0.63(0.01)	0.61(0.01)

A.2.2 SARCOS Robot Inverse Dynamics

Table A.2: Average RMSE scores on the SARCOS score over 7 tasks. The figures presented are the mean score (and standard error) over 10 runs. The lowest statistically significant scores based on a Wilcoxon test are presented in boldface. Lower is better.

	Number of Training Inputs					
	100	200	500	1000	2000	5000
sMDGP	0.78(0.05)	0.64(0.05)	0.53(0.03)	**0.31**(**0.03**)	**0.26**(**0.02**)	**0.22**(**0.02**)
mMDGP	0.80(0.05)	0.69(0.05)	0.51(0.05)	**0.31**(**0.03**)	**0.26**(**0.03**)	0.23(0.02)
cMDGP	0.78(0.07)	0.64(0.05)	0.52(0.03)	0.33(0.03)	0.27(0.03)	0.23(0.02)
iDGP	0.78(0.06)	0.64(0.05)	0.53(0.03)	0.47(0.02)	0.36(0.03)	0.26(0.02)
cGP	0.98(0.01)	0.96(0.01)	0.93(0.03)	0.86(0.06)	0.29(0.03)	0.26(0.03)
iGP	0.98(0.02)	0.95(0.02)	0.92(0.02)	0.89(0.05)	0.80(0.07)	0.63(0.07)
mANN2	**0.73**(**0.06**)	**0.58**(**0.04**)	**0.41**(**0.03**)	0.34(0.02)	0.31(0.02)	0.30(0.02)
mANN3	0.77(0.07)	0.62(0.05)	0.45(0.03)	0.39(0.02)	0.36(0.02)	0.35(0.02)
mBNN2	0.80(0.07)	0.62(0.05)	0.44(0.03)	0.34(0.02)	0.27(0.02)	**0.22**(**0.02**)
mBNN3	0.79(0.07)	0.64(0.05)	0.46(0.03)	0.36(0.02)	0.29(0.02)	0.24(0.02)

Table A.3: ELBO computation cost in CPU time for different multitask GP models. These are timed on a system with the following specifications: OS: Linux 4.9.202-1-Manjaro with X96-64 instruction set. CPU: Intel Core i7-4712HQ @ 2.30GHz. RAM: 16GB.

SARCOS: ELBO computation time (batch size = 500)	
Model	Wall-clock time in milliseconds
sMDGP	7.89(0.15)
mMDGP	8.89(0.25)
cMDGP	7.67(0.29)
iDGP	6.43(0.16)
cGP	2.41(0.04)
iGP	3.22(0.06)

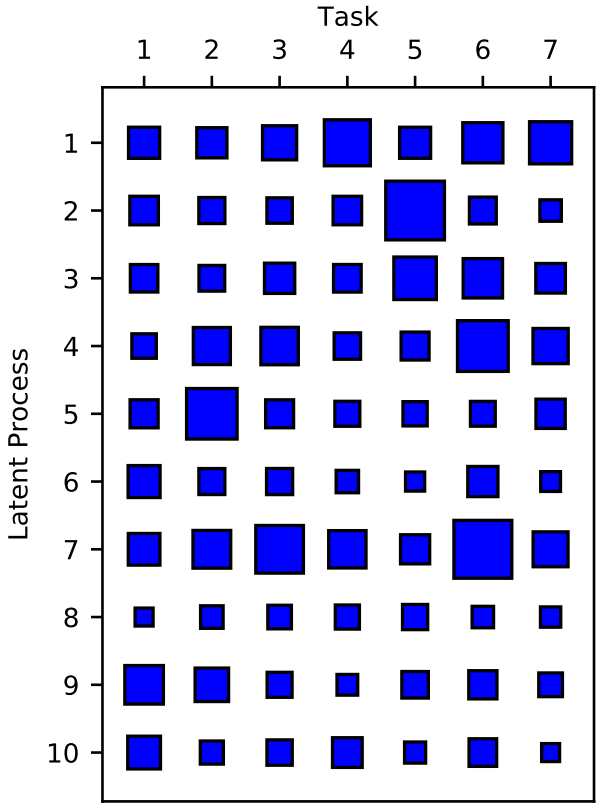


Figure A.1: Hinton diagram illustrating the ARD weights for on of the sMDGP runs on Sarcos with 5000 training datapoints. This indicates that the model assigns different weights for the latent processes for each task.

APPENDIX B

Appendix for Amortised Variance Reduction

B.1 Theoretical Analysis

B.1.1 Convergence Results

Control variates are introduced to reduce the gradient variance and thereby improve the optimisation behaviour. This section shows how controlling the gradient can improve the convergence behaviour of Stochastic Gradient Descent (SGD) [Robbins and Monro, 1951], in an idealised scenario.

Consider the function $f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) : \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R}$, where $\boldsymbol{\epsilon}$ is a random variable distributed according to $p(\boldsymbol{\epsilon})$. The goal is to solve the following optimisation problem using SGD,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta})]. \tag{B.1}$$

$\mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta})]$ is assumed to be strongly convex and smooth as defined below.

SGD starts with an initial guess for the optimal parameter $\boldsymbol{\theta}_0$. The parameter is then sequentially updated according to

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \hat{\mathbf{g}}(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t), \tag{B.2}$$

where $\boldsymbol{\epsilon}_t$ is an independent realisation (over t) of $\boldsymbol{\epsilon}$ at time t, $\hat{\mathbf{g}}(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t)$, and $\eta_t \in \mathbb{R}$ is a constant.

To reduce the variance in the update direction, one can set

$$\tilde{\mathbf{g}}(\boldsymbol{\epsilon}, \boldsymbol{\theta}) = \hat{\mathbf{g}}(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - \mathbf{c}(\boldsymbol{\epsilon}, \boldsymbol{\theta}), \tag{B.3}$$

where $\mathbf{c}(\boldsymbol{\epsilon}, \boldsymbol{\theta}) : \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R}^P$ is a control term designed to reduce the randomness

in $\hat{\mathbf{g}}(\boldsymbol{\epsilon}, \boldsymbol{\theta})$. This gives a new update rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \tilde{\mathbf{g}}(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t). \tag{B.4}$$

One can show show that adding a control term as in (B.3) yields a better convergence rate for SGD under the following assumptions:

Assumption 2 (Smoothness)**.** *$f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ is L smooth in $\boldsymbol{\theta}$ (for some $L > 0$), i.e.*

$$f(\boldsymbol{\epsilon}, \boldsymbol{\theta}') - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \leq (\boldsymbol{\theta}' - \boldsymbol{\theta})^\intercal \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) + \frac{1}{2} L \|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2^2,$$

Assumption 3 (Strong Convexity)**.** *$f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ is H strongly convex in $\boldsymbol{\theta}$ (for some $H > 0$), i.e.*

$$f(\boldsymbol{\epsilon}, \boldsymbol{\theta}') - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \geq (\boldsymbol{\theta}' - \boldsymbol{\theta})^\intercal \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) + \frac{1}{2} H \|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2^2,$$

Assumption 4 (Efficient Control Variate)**.** *The norm of the controlled gradient is bounded by a constant M as*

$$\mathbb{E}[\|\nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}, \boldsymbol{\theta})\|_2^2] \leq M \mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)].$$

Theorem B.1.1 (Convergence Rate)**.** *Under assumptions 2, 3 and 4, with $\eta_t = \eta \leq \frac{1}{2L+M}$ for all t, the update rule given by (B.4) offers a linear convergence rate. Specifically, there exists $0 < c < 1$ (depending on H, L and M as specified in Appendix B.1.2) such that*

$$\mathbb{E}\left[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2\right] \leq c^t \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2. \tag{B.5}$$

Proof. See Appendix B.1.2. □

Although the above assumptions do not necessarily hold in practice, Theorem B.1.1 is useful in giving intuition into the convergence speed of the method proposed in Chapter 4. This theorem gives a sufficient condition for the control variate which guarantees a fast convergence rate similar to that of (non-stochastic) gradient descent and Stochastic Variance Reduced Gradient descent (SVRG) [Johnson and Zhang, 2013]. Specifically, if Assumption 3 holds, this method offers the so-called linear convergence rate.

Assumption 3 can be relaxed to the following assumption.

Assumption 5. *The norm of the controlled gradient is bounded as*

$$\mathbb{E}[\|\nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}, \boldsymbol{\theta})\|_2^2] \leq \bar{M}.$$

Under assumptions 2, 3 and 5, with $\eta_t = \eta \leq \frac{1}{2L}$ for all t, for the update rule given by (B.4), there exists $0 < \bar{c} < 1$ (depending on H and L as specified in Appendix B.1.2) such that

$$\mathbb{E}\left[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2\right] \leq \bar{c}^t \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2 + \frac{2\eta^2 \bar{M}(1-\bar{c}^t)}{1-\bar{c}}. \tag{B.6}$$

This result shows that the more efficient the control variate (the smaller $\bar{M}$), the smaller the error in optimisation ($\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2$). For the detail on the proof of (B.6) see Appendix B.1.2.

B.1.2 Proofs for Convergence Results

Proof of Theorem B.1.1. The following lemma is established based on the smoothness and strong convexity of $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$.

Lemma B.1.2. *By smoothness (Assumption 1) and strong convexity (Assumption 2) of* $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$,

$$\mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)] \geq \frac{1}{2L}\mathbb{E}\left[\|\nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)\|_2^2\right]. \tag{B.7}$$

Let $h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) = f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*)$. By convexity of $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$, $h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \geq 0$. Let $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta})$. Notice that $\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) = \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)$.

$$\begin{aligned} h(\boldsymbol{\epsilon}, \boldsymbol{\theta}') - h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) &\leq -\eta \nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta})^\intercal \nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) + \frac{1}{2} L \eta^2 \|\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\|_2^2 \\ &= (\frac{1}{2} L \eta^2 - \eta) \|\nabla h(w, \epsilon))\|_2^2. \end{aligned}$$

With the choice of $\eta = \frac{1}{L}$ and noticing $h(\boldsymbol{\epsilon}, \boldsymbol{\theta}') \geq 0$:

$$-h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \leq -\frac{1}{2L} \|\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\|_2^2.$$

Taking expectations,

$$\begin{aligned}\frac{1}{2L}\mathbb{E}[\|\nabla h(\boldsymbol{\epsilon},\boldsymbol{\theta}))\|_2^2] &\leq \mathbb{E}[h(\boldsymbol{\epsilon},\boldsymbol{\theta})] \\ &= \mathbb{E}[f(\boldsymbol{\epsilon},\boldsymbol{\theta}) - f(\boldsymbol{\epsilon},\boldsymbol{\theta}^*) - \nabla f(\boldsymbol{\epsilon},\boldsymbol{\theta}^*)^\top(\boldsymbol{\theta}-\boldsymbol{\theta}^*)] \\ &\leq \mathbb{E}[f(\boldsymbol{\epsilon},\boldsymbol{\theta}) - f(\boldsymbol{\epsilon},\boldsymbol{\theta}^*)],\end{aligned}$$

which completes the proof of this lemma.

Now, let $v_t = \nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1}) - c(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1})$. Then

$$\begin{aligned}&\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \\ &\quad= \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 - 2\eta(\boldsymbol{\theta}_{t-1}-\boldsymbol{\theta}^*)^\top\mathbb{E}[v_t] + \eta^2\mathbb{E}[\|v_t\|_2^2] \\ &\quad= \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 - 2\eta(\boldsymbol{\theta}_{t-1}-\boldsymbol{\theta}^*)^\top\mathbb{E}[\nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1})] + \eta^2\mathbb{E}[\|v_t\|_2^2] \\ &\quad\leq \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 - 2\eta\mathbb{E}[[f(\boldsymbol{\epsilon},\boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon},\boldsymbol{\theta}^*)] + \eta^2\mathbb{E}[\|v_t\|_2^2]. \qquad \text{(B.8)}\end{aligned}$$

For the last term $\mathbb{E}[\|v_t\|_2^2]$,

$$\begin{aligned}&\mathbb{E}[\|v_t\|_2^2] \\ &\quad= \mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}^*) + \nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1})\|^2\Big] \\ &\quad\leq 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}^*)\|_2^2\Big] \\ &\qquad\quad + 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1})\|_2^2\Big] \\ &\quad\leq (4L+2M)\mathbb{E}[f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}_{t-1},\boldsymbol{\theta}^*)], \qquad \text{(B.9)}\end{aligned}$$

where the last inequality holds by Assumption 3.

By strong convexity of $f(\boldsymbol{\epsilon},\boldsymbol{\theta})$,

$$\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 \leq \frac{2}{H} f(\boldsymbol{\epsilon},\boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon},\boldsymbol{\theta}^*). \qquad \text{(B.10)}$$

Combining the last 3 inequalities yields

$$\begin{aligned}&\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \\ &\quad\leq \mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] - 2\eta(1-\eta(2L+M))\mathbb{E}[f(\boldsymbol{\epsilon},\boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon},\boldsymbol{\theta}^*)] \\ &\quad\leq \Big(1 - \eta H(1-\eta(2L+M))\Big)\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2].\end{aligned}$$

For $\eta \leq \frac{1}{2L+M}$ and $c = (1 - \eta H(1 - \eta(2L + M))$,

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \leq c^t \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2,$$

which completes the proof of Theorem B.1.1.

When Assumption 3 does not hold and Assumption 4 holds, following the same line of reasoning and replacing the upper bound on $\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|_2^2\Big]$ with $\bar{M}$,

$$\begin{aligned}
&\mathbb{E}[\|v_t\|_2^2] \\
&= \mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) + \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|^2\Big] \\
&\leq 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*)\|_2^2\Big] \\
&\quad + 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|_2^2\Big] \\
&\leq 4L\mathbb{E}[f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*)] + 2\bar{M}. \qquad \text{(B.11)}
\end{aligned}$$

Combining inequalities (B.8), (B.10) and (B.11),

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \\
&\leq \Big(1 - \eta H(1 - 2L\eta)\Big)\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] + 2\eta^2\bar{M}.
\end{aligned}$$

For $\eta \leq \frac{1}{2L}$ and $\bar{c} = (1 - \eta H(1 - 2L\eta))$,

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \leq \bar{c}\,\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] + 2\eta^2\bar{M}.$$

Equivalently,

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] + \frac{2\eta^2\bar{M}}{\bar{c} - 1} \leq \bar{c}\Big(\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] + \frac{2\eta^2\bar{M}}{\bar{c} - 1}\Big),$$

which shows

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] + \frac{2\eta^2\bar{M}}{\bar{c} - 1} \leq \bar{c}^t\Big(\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2 + \frac{2\eta^2\bar{M}}{\bar{c} - 1}\Big).$$

Thus, for the $\mathbb{E}[\|\theta_t - \theta^*\|_2^2]$,

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \leq \bar{c}^t\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2 + \frac{2\eta^2\bar{M}(\bar{c}^t - 1)}{\bar{c} - 1}.$$

□

B.2 Description of Experiment Models

This section give detailed descriptions of the models used in the experiments in Section 4.6 in Chapter 4.

B.2.1 Logistic Regression

Summary A classification task on the *titanic* dataset with a Bayesian logistic regression model. Inference is performed using the reparameterisation gradient formulation of VI, where a Gaussian approximate posterior was selected and its mean vector and full covariance matrix are learnt. A unit Gaussian prior was placed on the weights.

Dataset The *titanic* dataset consists of 2201 training examples. Each data point comprises a binary class label and a feature vector of length 4. Standard normalisation was performed on the features, *i.e.*, subtracting the mean and dividing by the standard deviation. The dataset can be obtained from the following URL: http://persoal.citius.usc.es/manuel.fernandez.delgado/papers/jmlr/data.tar.gz.

Model description In the Bayesian logistic regression model, the output log-odds are modelled as a linear transformation of the inputs, i.e., $\text{logit}(t) = \boldsymbol{\omega}^\intercal \mathbf{x} \iff t = s(\boldsymbol{\omega}^\intercal \mathbf{x})$, where t is the target output, $\mathbf{x}$ is a feature vector (with a leading element with value 1 to represent the bias), $\boldsymbol{\omega}$ is a weight vector and $s(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid function. This model can be treated probabilistically by setting a prior on the weights $\boldsymbol{\omega}$ and a likelihood model on the targets t. The following model is specified:

$$\begin{aligned} p(\boldsymbol{\omega}) &= \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \\ p(t|\boldsymbol{\omega}) &= \text{Bernoulli}(s(\boldsymbol{\omega}^\intercal \mathbf{x})). \end{aligned}$$

Inference problem The aim is to find the posterior distribution of the weight given the targets $\mathbf{T} = (t_1; \ldots; t_N)$,

$$p(\boldsymbol{\omega}|\mathbf{T}) = \frac{\prod_{n=1}^{N} p(t_n|\boldsymbol{\omega}) p(\boldsymbol{\omega})}{\int \prod_{n=1}^{N} p(t_n|\boldsymbol{\omega}) p(\boldsymbol{\omega}) d\boldsymbol{\omega}}.$$

In VI, $p(\boldsymbol{\omega}|\mathbf{T})$ is approximated with another distribution $q(\boldsymbol{\omega}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$, which can be obtained by minimising the Negative Evidence Lower Bound (NELBO):

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= -\sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\omega})}[\log p(t_n|\boldsymbol{\omega})] + \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})) \\ &\approx -\sum_{n=1}^{N} \frac{1}{S}\sum_{s=1}^{S} \log p(t_n|\boldsymbol{\omega}(\boldsymbol{\epsilon}_n^{(s)})) + \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})), \qquad \boldsymbol{\epsilon}_n^{(s)} \sim \mathcal{N}(0, \mathrm{I}), \quad \text{(B.6)}\end{aligned}$$

where $\boldsymbol{\theta} = (\mathbf{m}, \mathbf{L})$ with $\mathbf{S} = \mathbf{L}\mathbf{L}^\intercal$ and $\boldsymbol{\omega}(\boldsymbol{\epsilon}) = \mathbf{m} + \mathbf{L}\boldsymbol{\epsilon}$ is overloaded to represent the location-scale transformation (reparameterisation trick).

Instantiation For the experiments in Section 4.6, the model is instantiated by setting the prior covariance to $\boldsymbol{\Sigma} = \mathrm{I}$. For the doubly stochastic objective function in (B.6),the number of Monte Carlo samples is set to $S = 1$ and the N data points are subsampled into mini-batches of size $|\mathcal{B}| = 10$ (some results for $|\mathcal{B}| = 100$ are also shown in Appendix B.4.1).

B.2.2 Deep Gaussian Processes

Summary A regression task on the *airfoil* dataset using a Deep Gaussian Process (DGP).A 2-layer model with an inner-layer dimension of 5 was used, and a Squared Exponential kernel for the GP priors. The doubly stochastic formulation of the VI problem [Salimbeni and Deisenroth, 2017] was used. The parameters of the approximate Gaussian posterior are learnt, keeping the hyperparameters fixed. The inducing locations are fixed and selected as the centroids of k-means clusters from the data.

Dataset The *airfoil* dataset consists of 1500 training examples. Each data point comprises a target label in $\mathbb{R}$ and a feature vector of length 5. Standard normalisation was performed on the features, *i.e.*, subtracting the mean and dividing by the standard deviation. The dataset can be obtained from the following URL: `https://drive.google.com/file/d/0BxWe_IuTnMFcYXhxdUNwRHBKTlU/view`.

Model description The description of the 2-layer DGP model used in this experiment is given in the following. For a target $t \in \mathbb{R}$ with corresponding inputs $\mathbf{x} \in \mathcal{F}$, the input-to-output relationship is modelled as

$$t = f_2(f_1(\mathbf{x})) + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2).$$

GP priors were set on f_1 and f_2, yielding the following full probabilistic model:

$$\begin{aligned}
p(\mathbf{u}_1) &= \mathcal{N}(0, \mathbf{K}_{\mathbf{u}_1\mathbf{u}_1}), \\
p(\mathbf{f}_1|\mathbf{u}_1) &= \mathcal{N}(\mathrm{id}(\mathbf{X}) + \mathbf{K}_{\mathbf{f}_1\mathbf{u}_1}\mathbf{K}_{\mathbf{u}_1\mathbf{u}_1}^{-1}\mathbf{u}_1, \mathbf{K}_{\mathbf{f}_1\mathbf{f}_1} - \mathbf{K}_{\mathbf{f}_1\mathbf{u}_1}\mathbf{K}_{\mathbf{u}_1\mathbf{u}_1}^{-1}\mathbf{K}_{\mathbf{f}_1\mathbf{u}_1}^\intercal), \\
p(\mathbf{u}_2) &= \mathcal{N}(0, \mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}), \\
p(\mathbf{f}_2|\mathbf{u}_2) &= \mathcal{N}(\mathbf{K}_{\mathbf{f}_2\mathbf{u}_2}\mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}^{-1}\mathbf{u}_2, \mathbf{K}_{\mathbf{f}_2\mathbf{f}_2} - \mathbf{K}_{\mathbf{f}_2\mathbf{u}_2}\mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}^{-1}\mathbf{K}_{\mathbf{f}_2\mathbf{u}_2}^\intercal), \\
p(t \,|\, \mathbf{f}_1, \mathbf{f}_2) &= \mathcal{N}(f_2(\mathbf{f}_1), \sigma^2\mathrm{I}).
\end{aligned}$$

Here, $\mathbf{Z}_1$ is a design matrix of M inducing locations. $\mathbf{K}_{\mathbf{u}_1,\mathbf{u}_1} = k_1(\mathbf{Z}_1, \mathbf{Z}_1)$ is an $M \times M$ Gram matrix generated by a kernel function $k_1(\cdot,\cdot)$. Similarly $\mathbf{K}_{\mathbf{f}_1\mathbf{f}_1} = k_1(\mathbf{X}, \mathbf{X})$ is an $N \times N$ Gram matrix, where $\mathbf{X}$ is the design matrix of N inputs $\mathbf{x}_n$, and $\mathrm{id}(\cdot)$ is the identity function. $\mathbf{K}_{\mathbf{u}_2\mathbf{u}_2} = k_2(\mathbf{Z}_2, \mathbf{Z}_2)$ is also an $M \times M$ Gram matrix generated by $k_2(\cdot,\cdot)$, where $\mathbf{Z}_2$ are the inducing locations for the second layer, and $\mathbf{K}_{\mathbf{f}_2,\mathbf{f}_2} = k_2(f_1(\mathbf{X}), f_1(\mathbf{X}))$ is an $N \times N$ Gram matrix of the transformation of the inputs by the first layer. Finally, $\mathbf{u}_1$ and $\mathbf{u}_2$ are the auxiliary inducing points. In-depth treatment on the DGP model can be found in Section 2.1.2.3 & Section 2.4.3 in Chapter 2 and in Salimbeni and Deisenroth [2017].

Inference problem The aim is to find the posterior distribution $\mathbf{f}_1$ and $\mathbf{f}_2$ (and by extension $\mathbf{u}_1$ and $\mathbf{u}_2$) given the targets $\mathbf{T} = (t_1; \ldots; t_N)$,

$$p(\mathbf{u}_1, \mathbf{f}_1, \mathbf{u}_2, \mathbf{f}_2|\mathbf{T}) = \frac{\prod_{n=1}^N p(t_n|\mathbf{f}_1, \mathbf{f}_2)p(\mathbf{f}_2|\mathbf{u}_2)p(\mathbf{u}_2)p(\mathbf{f}_1|\mathbf{u}_1)p(\mathbf{u}_1)}{\int \prod_{n=1}^N p(t_n|\mathbf{f}_1, \mathbf{f}_2)p(\mathbf{f}_2|\mathbf{u}_2)p(\mathbf{u}_2)p(\mathbf{f}_1|\mathbf{u}_1)p(\mathbf{u}_1)\,\mathrm{d}\mathbf{f}_2\,\mathrm{d}\mathbf{u}_2\,\mathrm{d}\mathbf{f}_1\,\mathrm{d}\mathbf{u}_1}.$$

The doubly stochastic formulation of VI for DGPs [Salimbeni and Deisenroth, 2017] for this intractable problem is used, where $p(\mathbf{u}_1, \mathbf{f}_1, \mathbf{u}_2, \mathbf{f}_2|\mathbf{T}) \approx p(\mathbf{f}_2|\mathbf{u}_2)q(\mathbf{u}_2)p(\mathbf{f}_1|\mathbf{u}_1)q(\mathbf{u}_1)$, with $q(\mathbf{u}_1) = \mathcal{N}(\mathbf{m}_1, \mathbf{L}_1\mathbf{L}_1^\intercal)$ and $q(\mathbf{u}_2) = \mathcal{N}(\mathbf{m}_2, \mathbf{L}_2\mathbf{L}_2^\intercal)$. The parameters $\boldsymbol{\theta} = (\mathbf{m}_1, \mathbf{L}_1, \mathbf{m}_2, \mathbf{L}_2)$ of the approximation can be learned by optimising the following NELBO:

$$\begin{aligned}
\mathcal{L}(\theta) &= -\sum_{n=1}^N \mathbb{E}_{q(\mathbf{f}_2|\mathbf{f}_1)q(\mathbf{f}_1)}[\log p(t_n|\mathbf{f}_1, \mathbf{f}_2)] + \mathrm{KL}(q(\mathbf{u}_1)||p(\mathbf{u}_1)) + \mathrm{KL}(q(\mathbf{u}_2)||p(\mathbf{u}_2)) \\
&\approx -\sum_{n=1}^N \frac{1}{S}\sum_{s=1}^S \log p(t_n|\mathbf{f}_1(\boldsymbol{\epsilon}_{1n}^{(s)}), \mathbf{f}_2(\boldsymbol{\epsilon}_{2n}^{(s)}))] \\
&\quad + \mathrm{KL}(q(\mathbf{u}_1)||p(\mathbf{u}_1)) + \mathrm{KL}(q(\mathbf{u}_2)||p(\mathbf{u}_2)), \qquad \boldsymbol{\epsilon}_{1n}^{(s)}, \boldsymbol{\epsilon}_{2n}^{(s)} \sim \mathcal{N}(0, \mathrm{I}),
\end{aligned} \tag{B.7}$$

where $\mathbf{f}_1(\boldsymbol{\epsilon}) = \bar{\mathbf{m}}_1 + \bar{\mathbf{L}}_1\boldsymbol{\epsilon}$ is overloaded to represent the location-scale transformation, and $\bar{\mathbf{m}}_1$ and $\bar{\mathbf{L}}_1$ are the parameters of the marginal $q(\mathbf{f}_1) = \int p(\mathbf{f}_1|\mathbf{u}_1)q(\mathbf{u}_1)\,\mathrm{d}\mathbf{u}_1$. Similarly for $\mathbf{f}_2(\boldsymbol{\epsilon})$.

Instantiation For the experiments in Section 4.6, the model is instantiated by setting $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ as Squared Exponential kernels, with lengthscales fixed to 2 and signal variances also fixed to 2. The number of inducing locations was set to $M = 10$ and their values to the centroids of k-means clusters of the real inputs from the dataset. The likelihood variance was set to $\sigma^2 = 0.01$ and the width of the GP inner layer to 5. For the doubly stochastic NELBO in (B.7), the number of Monte Carlo samples was set to $S = 1$ and the N data points are subsampled into mini-batches of size $|\mathcal{B}| = 10$ (some results for $|\mathcal{B}| = 100$ are also shown in Appendix B.4.1).

B.3 Verification of Variance Reduction

This section provides extra results for the experiment in Section 4.6.2 for different recognition network objectives and different Adam learning rates on these objectives. The corresponding configurations are in the figure captions.

B.3.1 Logistic Regression Results on the *Titanic* Dataset

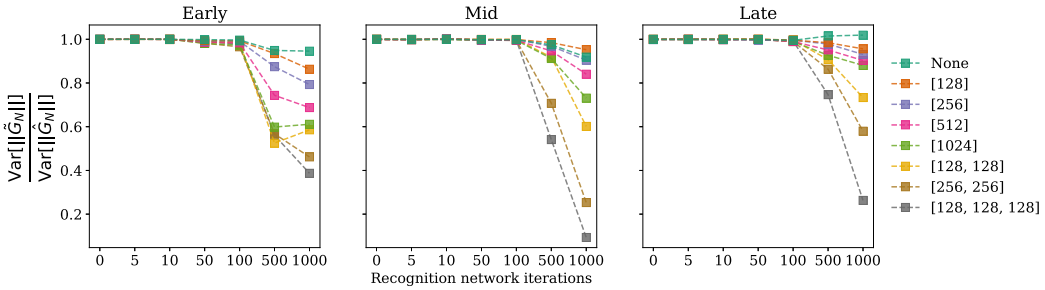


Figure B.1: Logistic regression. Squared difference objective. Recognition network learning rate = 10^{-3}.

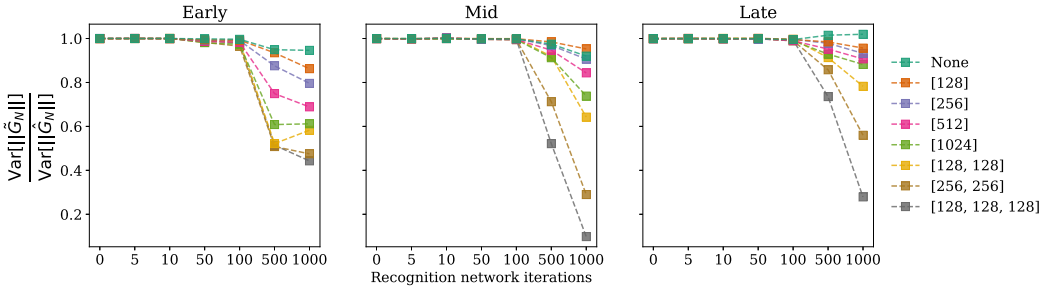


Figure B.2: Logistic regression. Gradient sum objective. Recognition network learning rate = 10^{-3}.

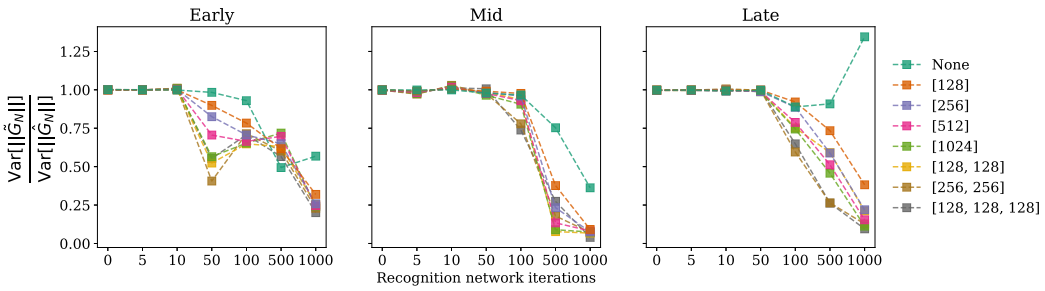


Figure B.3: Logistic regression. Squared difference objective. Recognition network learning rate = 10^{-2}.

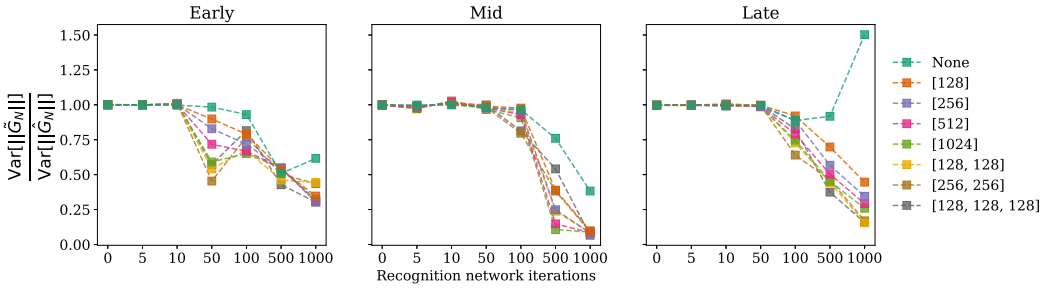


Figure B.4: Logistic regression. Squared difference objective. Recognition network learning rate = 10^{-2}.

B.3.2 Deep Gaussian Process Results on the *Airfoil* Dataset

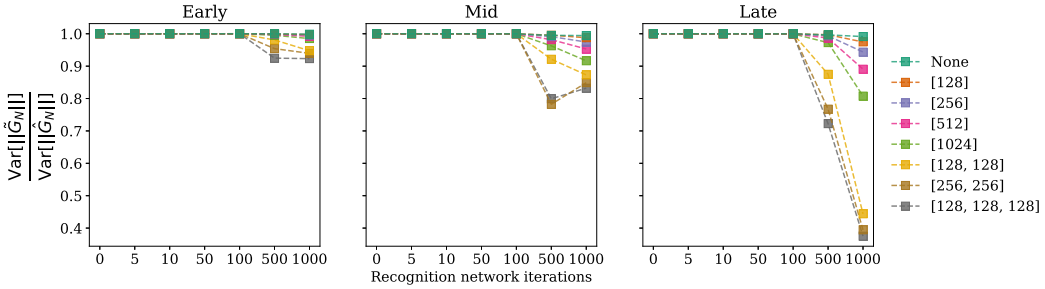


Figure B.5: DGP. Squared difference objective. Recognition network learning rate = 10^{-3}.

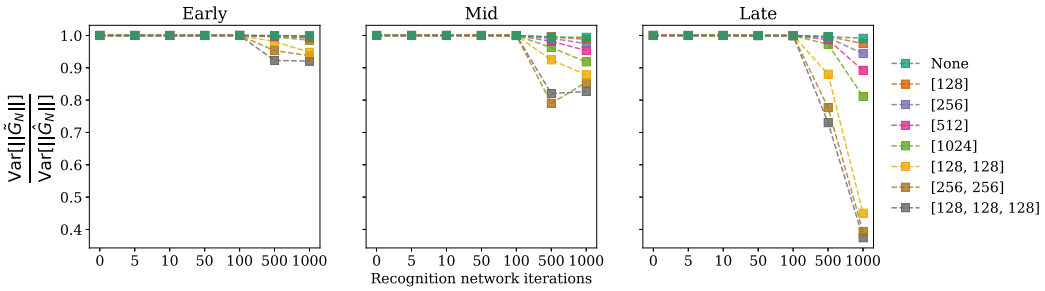


Figure B.6: DGP. Gradient sum objective. Recognition network learning rate = 10^{-3}.

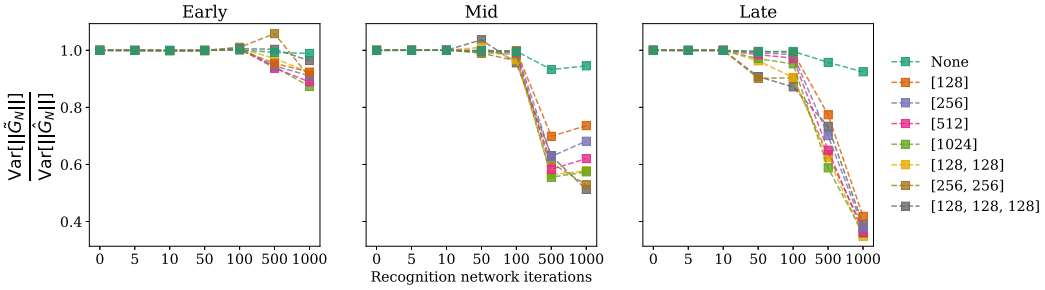


Figure B.7: DGP. Squared difference objective. Recognition network learning rate = 10^{-2}.

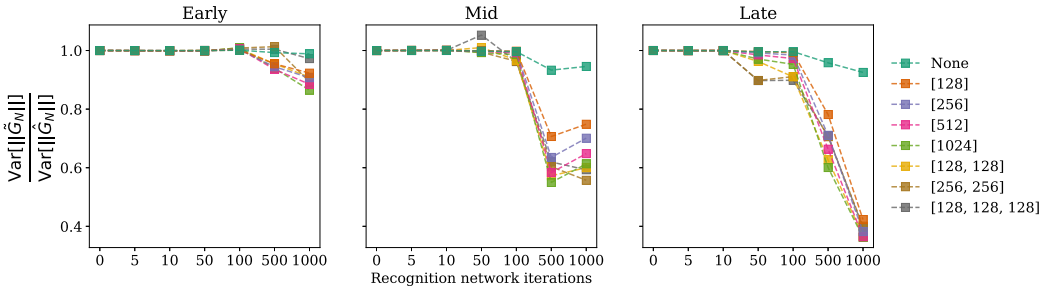


Figure B.8: DGP. Gradient sum objective. Recognition network learning rate = 10^{-2}.

B.4 Simultaneous Optimisation of Model and Recognition Network

This section provides extra results for the experiment in Section 4.6.3 for different recognition network objectives, different Adam learning rates on these objectives and different the mini-batch sizes. The corresponding configurations are in the figure captions.

B.4.1 Logistic Regression Results on the *Titanic* Dataset

B.4.1.1 Small mini-batch size, $|\mathcal{B}| = 10$

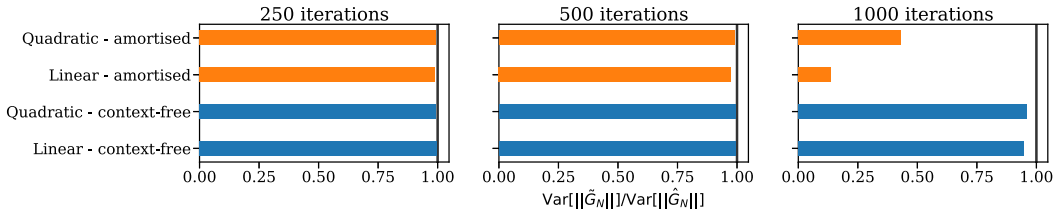


Figure B.9: Logistic regression. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

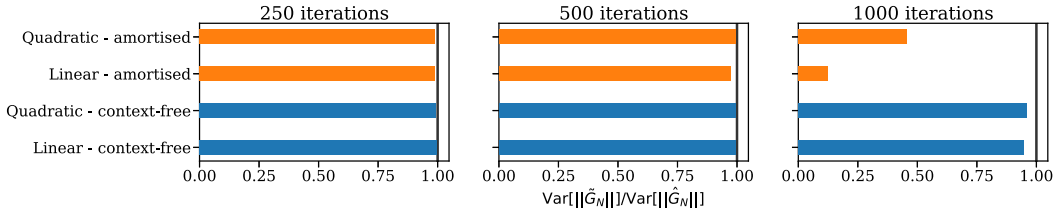


Figure B.10: Logistic regression. Gradient sum objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

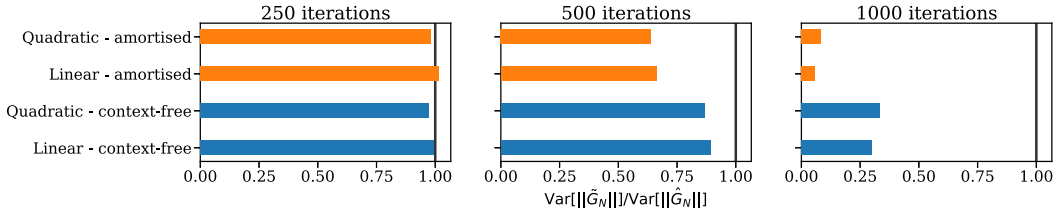


Figure B.11: Logistic regression. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-2}$.

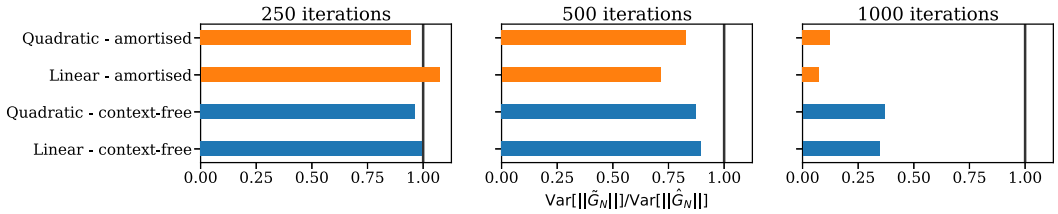


Figure B.12: Logistic regression. $|\mathcal{B}| = 10$. Gradient sum objective. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-2}$.

B.4.1.2 Large mini-batch size, $|\mathcal{B}| = 100$

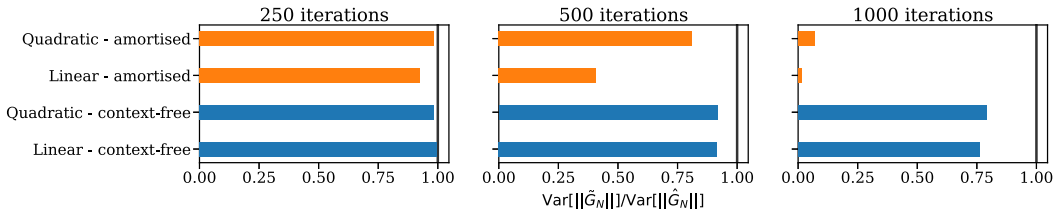


Figure B.13: Logistic regression. Squared difference objective. Network of size [128, 128, 128]. $|\mathcal{B}| = 100$. Recognition network learning rate $= 10^{-3}$.

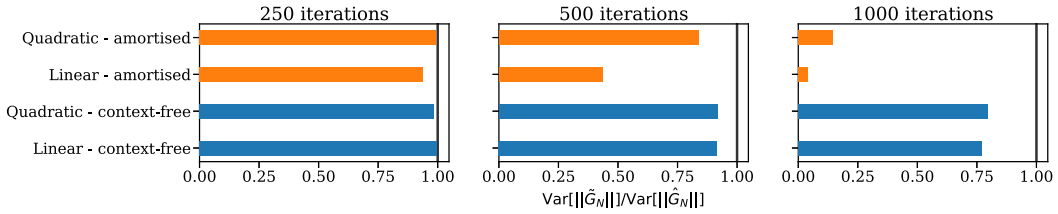


Figure B.14: Logistic regression. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-3}$.

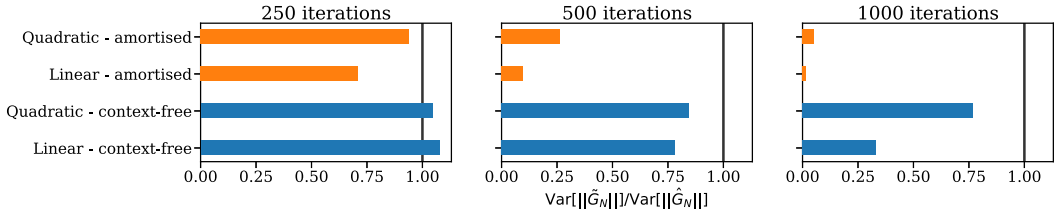


Figure B.15: Logistic regression. Squared difference objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-2}$.

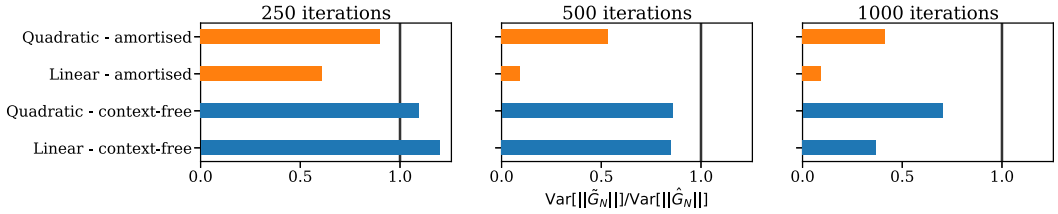


Figure B.16: Logistic regression. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-2}$.

B.4.2 Deep Gaussian Process Results on the *Airfoil* Dataset

B.4.2.1 Small mini-batch size, $|\mathcal{B}| = 10$

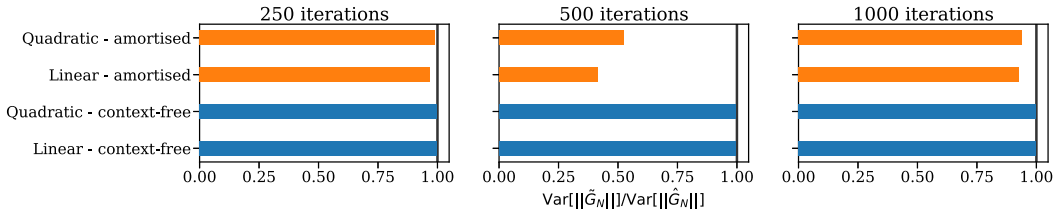


Figure B.17: DGP. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-3}$.

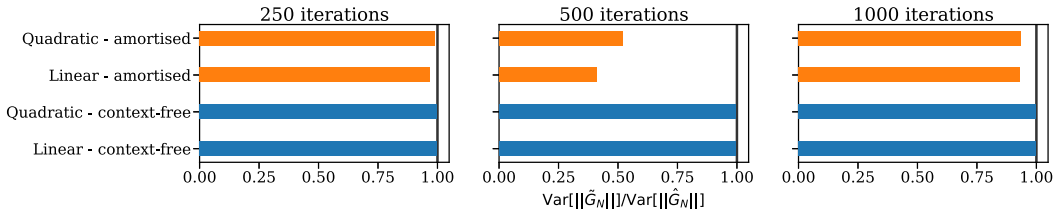


Figure B.18: DGP. Gradient sum objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-3}$.

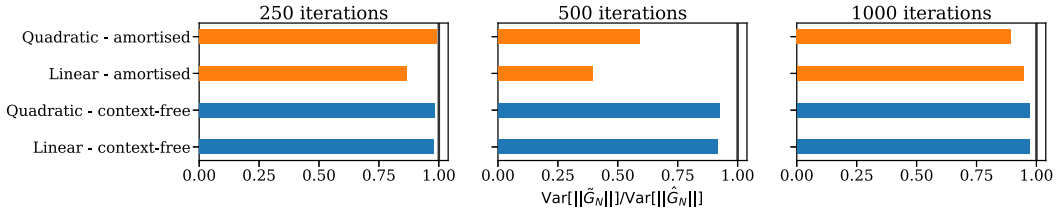


Figure B.19: DGP. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

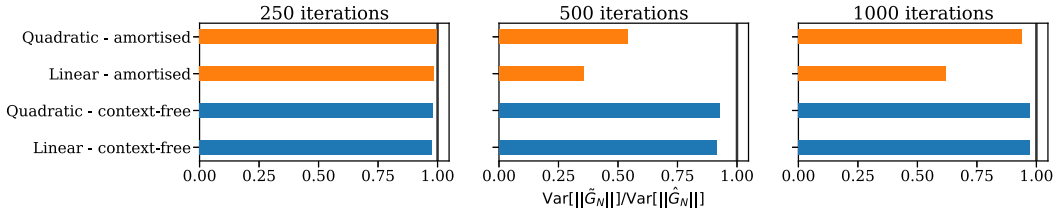


Figure B.20: DGP. Gradient sum objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

B.4.2.2 Large mini-batch size, $|\mathcal{B}| = 100$

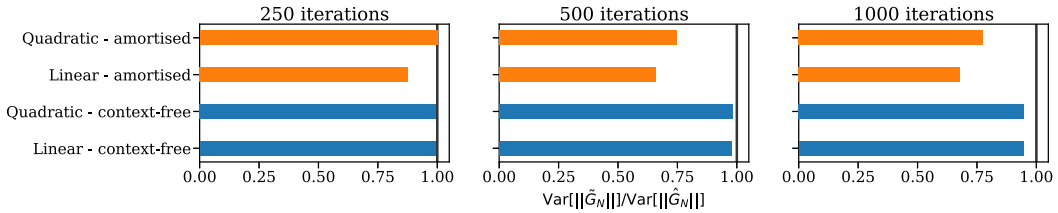


Figure B.21: DGP. Squared difference objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

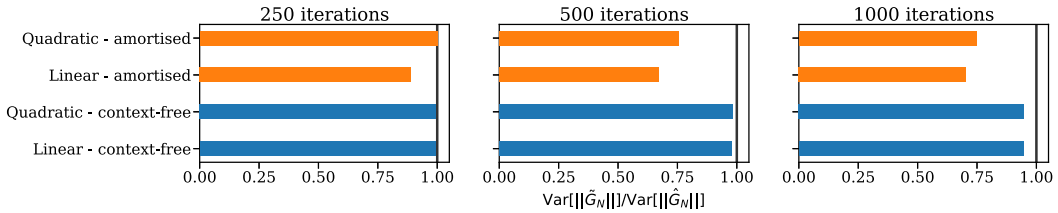


Figure B.22: DGP. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

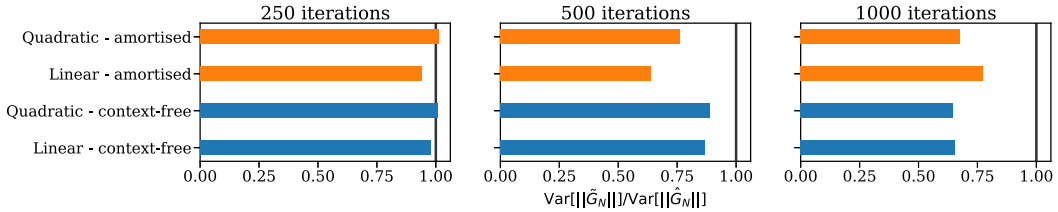


Figure B.23: DGP. Squared difference objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-2}$.

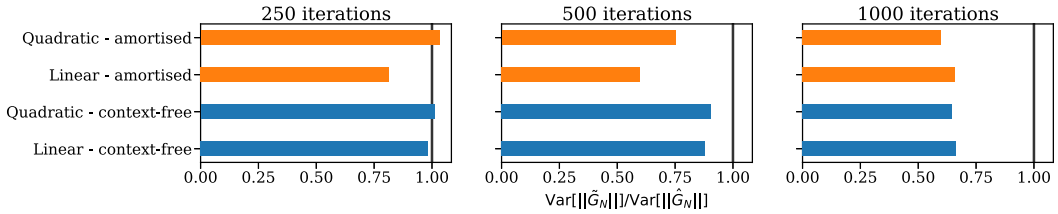


Figure B.24: DGP. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-2}$.

B.4.2.3 Variance Reduction Comparison with the Partial Gradients Objective

This section provides results comparing the *partial gradients* objective in (4.10), with the *gradient sum* (4.13) and *squared difference* (4.15) objectives. Fig. B.25 shows that the partial gradients objective induces further variance reduction than the alternatives due to its lower variance. However, note that in practice partial gradients are expensive to compute with current automatic differentiation libraries, therefore it is not feasible to use this objective from a computational point-of-view.

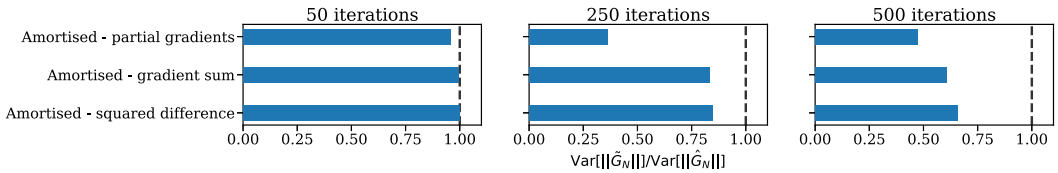


Figure B.25: DGP. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate $= 10^{-3}$.

APPENDIX C

Appendix for A Low Variance Gradient Estimator for Variational Inference

C.1 Further Analytical Results

This section contains further analytical results supplementing the analysis in (5.4).

C.1.1 Scale of δ^{CV}

In certain settings, the correction term $\boldsymbol{\delta}^{\mathrm{CV}}$ becomes negligible. This implies that $\hat{g}_{\mathrm{VarGrad}}(\boldsymbol{\phi})$ and $\hat{g}_{\mathrm{CV}}(\boldsymbol{\phi})$ (Reinforce with optimal score function control variate coefficients) behave almost identically.

Proposition C.1.1. *$\boldsymbol{\delta}^{CV}$ is small in comparison to $\mathbb{E}_{q_{\boldsymbol{\phi}}}[\mathbf{a}_{VarGrad}]$ if the KL divergence between $q_{\boldsymbol{\phi}}(\mathbf{x})$ and $p(\mathbf{x}\,|\,\mathbf{y})$ is either large or small. Assume that $q_{\boldsymbol{\phi}}(\mathbf{x})$ has lighter tails than the posterior $p(\mathbf{x}\,|\,\mathbf{y})$, i.e., there exists a constant $C > 0$ such that*

$$\sup_{\mathbf{x}} \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})} < C. \tag{C.1}$$

Furthermore, define the kurtosis of the score function,

$$\mathrm{Kurt}[\partial_{\phi_i} \log q_{\phi}] = \frac{\mathbb{E}_{q_{\phi}}[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^4]}{(\mathbb{E}_{q_{\boldsymbol{\phi}}}[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2])^2}, \tag{C.2}$$

and assume that it is bounded, i.e., $\mathrm{Kurt}[\partial_{\phi_i} \log q_{\phi}] < \infty$. Then, the ratio between the control variate correction $\boldsymbol{\delta}^{CV}$ and the expected control variate coefficient of

VarGrad can be upper bounded by

$$\left|\frac{\boldsymbol{\delta}_i^{CV}}{\mathbb{E}_{q_\phi}[\mathbf{a}_{VarGrad}]}\right| \leq \frac{2\sqrt{C\,\mathrm{Kurt}[\partial_{\phi_i}\log q_\phi]}}{\left|\sqrt{KL(q_\phi(\mathbf{x})\,||\,p(\mathbf{x}\,|\,\mathbf{y}))} - \frac{\log p(\mathbf{y})}{\sqrt{KL(q_\phi(\mathbf{x})\,||\,p(\mathbf{x}\,|\,\mathbf{y}))}}\right|}. \tag{C.3}$$

Proof. See Appendix C.2.3. □

Remark C.1.2. *The variational approximation $q_\phi(\mathbf{x})$ typically underestimates the spread of the posterior $p(\mathbf{x}\,|\,\mathbf{y})$ [Blei et al., 2017]; hence, the assumption in Equation (C.1) is usually satisfied in practice after a few iterations of the optimisation algorithm. The kurtosis $\mathrm{Kurt}[\partial_{\phi_i}\log q_\phi]$ quantifies the weight of the tails of the variational approximation in terms of the score function.*

Remark C.1.3. *The upper bound in Equation (C.3) identifies two regimes. When $KL(q_\phi(\mathbf{x})\,||\,p(\mathbf{x}\,|\,\mathbf{y}))$ is large, the bound asserts that the relative error satisfies*

$$\left|\frac{\delta_i^{\mathrm{CV}}}{\mathbb{E}_{q_\phi}[\mathbf{a}_{VarGrad}]}\right| \lessapprox \mathcal{O}\left(KL(q_\phi(\mathbf{x})\,||\,p(\mathbf{x}\,|\,\mathbf{y}))^{-1/2}\right), \tag{C.4}$$

as the second term in the denominator of Equation (C.3) becomes negligible. This can happen in the early stages of the optimisation process, in which case $\boldsymbol{\delta}^{CV}$ is expected to be small. Since the KL divergence increases with the dimensionality of the latent variable $\mathbf{x}$ (see Appendix C.2.6), Equation (C.3) also implies that the ratio becomes smaller as the number of latent variables grows. Moreover, if the minimum KL divergence between the variational family and the true posterior is large (i.e., if the best candidate in the variational family is still far away from the target), the correction term δ_i^{CV} can be negligible during the whole optimisation procedure, which is often the case in practice.

In the regime where $KL(q_\phi(\mathbf{x})\,||\,p(\mathbf{x}\,|\,\mathbf{y}))$ approaches zero (i.e., towards the end of the optimisation process if the variational family is well specified or includes the posterior), then Equation (C.3) implies that

$$\left|\frac{\delta_i^{CV}}{\mathbb{E}_{q_\phi}[\mathbf{a}_{VarGrad}]}\right| \lessapprox \mathcal{O}\left(KL(q_\phi(\mathbf{x})\,||\,p(\mathbf{x}\,|\,\mathbf{y}))^{1/2}\right). \tag{C.5}$$

In this regime, the error w.r.t. the optimal control variate coefficient decreases with the KL divergence. The estimates in Equation (C.4) and Equation (C.5) combined suggest that the relative error remains bounded throughout the optimisation. This is verified experimentally in Section 4.6.

C.1.2 Effect of Latent Variable Dimension on the Variance of VarGrad

Corollary C.1.4. *Let S be the number of samples and D the dimension of the latent variable $\mathbf{x}$. Furthermore, let the assumptions of Proposition C.1.1 be satisfied and assume that $KL(q_{\phi}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y}))$ is strictly increasing in D. Then there exist $S_0, D_0 \in \mathbb{N}$ such that*

$$\mathrm{Var}\left(\hat{g}_{VarGrad,i}(\boldsymbol{\phi})\right) \leq \mathrm{Var}\left(\hat{g}_{Reinforce,i}(\boldsymbol{\phi})\right), \qquad \text{for all } S \geq S_0 \text{ and } D \geq D_0. \quad \text{(C.6)}$$

Proof. See Appendix C.2.5. □

C.2 Proofs of Analytical Results

C.2.1 Proof of Proposition 5.3.2

Consider the gradient of the KL divergence,

$$\nabla_{\boldsymbol{\phi}}\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) = \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})\, \mathrm{d}\mathbf{x} + \int \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right) \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}\, \mathrm{d}\mathbf{x}, \quad \text{(C.7)}$$

where the first term can be dropped since $\int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})\, \mathrm{d}\mathbf{x} = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\mathbf{x})\, \mathrm{d}\mathbf{x} = \nabla_{\boldsymbol{\phi}}(1) = 0$.

Now consider the gradient of the log-variance loss. Using the definition from Equation (5.5), one can see that

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathcal{L}_r(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) &= \frac{1}{2}\nabla_{\boldsymbol{\phi}} \int \log^2\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right) r(\mathbf{x})\, \mathrm{d}\mathbf{x} - \frac{1}{2}\nabla_{\boldsymbol{\phi}} \left(\int \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right) r(\mathbf{x})\, \mathrm{d}\mathbf{x}\right)^2 \\ &= \int \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right) \frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} r(\mathbf{x})\, \mathrm{d}\mathbf{x} - \left(\int \log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right) r(\mathbf{x})\, \mathrm{d}\mathbf{x}\right) \left(\int \frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{x})} r(\mathbf{x})\, \mathrm{d}\mathbf{x}\right).\end{aligned}$$

When the gradient is evaluated at $r(\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{x})$, the right-most term vanishes, since $\int \frac{\nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})}{r(\mathbf{x})} r(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\mathbf{x})\, \mathrm{d}\mathbf{x} = 0$. Thus, the gradient of the log-variance loss is equal to the gradient of the KL divergence.

C.2.2 Proof of Lemma 5.4.1

Proof. First, notice that $\mathrm{Var}_{q_{\boldsymbol{\phi}}}\left(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x})\right) = \mathbb{E}_{q_{\boldsymbol{\phi}}}[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2]$ since $\mathbb{E}_{q_{\boldsymbol{\phi}}}[\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x})] = 0$. Then

$$a_i^\star = \frac{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[f_{\boldsymbol{\phi}}(\mathbf{x})(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right]}{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right]} \tag{C.8}$$

$$= \frac{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[f_{\boldsymbol{\phi}}(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[f_{\boldsymbol{\phi}}(\mathbf{x})\right]\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[f_{\boldsymbol{\phi}}(\mathbf{x})\right]\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right]}{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right]} \tag{C.9}$$

$$= \mathbb{E}_{q_{\boldsymbol{\phi}}}[\bar{f}_{\boldsymbol{\phi}}] + \delta_i^{\mathrm{CV}}. \tag{C.10}$$

The last line uses the fact that $\mathbb{E}_{q_{\boldsymbol{\phi}}}[f_{\boldsymbol{\phi}}(\mathbf{x})] = \mathbb{E}_{q_{\boldsymbol{\phi}}}[\bar{f}_{\boldsymbol{\phi}}]$. □

C.2.3 Proof of Proposition C.1.1

Proof. Note that

$$\left|\frac{\delta_i^{\mathrm{CV}}}{\mathbb{E}_{q_{\boldsymbol{\phi}}}[\mathbf{a}_{\mathrm{VarGrad}}]}\right| = \left|\frac{\mathrm{Cov}_{q_{\boldsymbol{\phi}}}\left(f_{\boldsymbol{\phi}}(\mathbf{x}), (\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right)}{\mathbb{E}_{q_{\boldsymbol{\phi}}}[f_{\boldsymbol{\phi}}(\mathbf{x})]\,\mathrm{Var}_{q_{\boldsymbol{\phi}}}\left(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x})\right)}\right|$$

$$= \left|\frac{\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\left(f_{\boldsymbol{\phi}}(\mathbf{x}) - \mathbb{E}_{q_{\boldsymbol{\phi}}}[f_{\boldsymbol{\phi}}(\mathbf{x})]\right)(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right]}{\mathbb{E}_{q_{\boldsymbol{\phi}}}[f_{\boldsymbol{\phi}}(\mathbf{x})]\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2\right]}\right|, \tag{C.11}$$

where the fact that $\mathbb{E}_{q_{\boldsymbol{\phi}}}[\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x})] = 0$ was used. From

$$\mathbb{E}[f_{\boldsymbol{\phi}}(\mathbf{x})] = -\mathrm{ELBO}(\boldsymbol{\phi}) = \mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) - \log p(\mathbf{y}),$$

and using the Cauchy-Schwarz inequality, Equation (C.11) can be bounded from above by

$$\frac{\left(\mathrm{Var}_{q_{\boldsymbol{\phi}}}\left(\log \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right)\right)^{1/2}}{\left|\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) - \log p(\mathbf{y})\right|}\left(\frac{\mathbb{E}_{q_{\boldsymbol{\phi}}}[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^4]}{(\mathbb{E}_{q_{\boldsymbol{\phi}}}[(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}(\mathbf{x}))^2])^2}\right)^{1/2}. \tag{C.12}$$

The second factor equals $\sqrt{\mathrm{Kurt}[\partial_{\phi_i} \log q_{\boldsymbol{\phi}}]}$. To bound the first factor, notice that

$$\left(\mathrm{Var}_{q_{\boldsymbol{\phi}}}\left(\log \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right)\right)^{1/2} \leq \left(\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log^2 \frac{q_{\boldsymbol{\phi}}(\mathbf{x})}{p(\mathbf{x}\,|\,\mathbf{y})}\right]\right)^{1/2} = \left(\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log^2 \frac{p(\mathbf{x}\,|\,\mathbf{y})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right]\right)^{1/2} \tag{C.13}$$

$$\leq \left(2\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\exp\left(\left|\log \frac{p(\mathbf{x}\,|\,\mathbf{y})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right|\right) - 1 - \left|\log \frac{p(\mathbf{x}\,|\,\mathbf{y})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right|\right]\right)^{1/2}, \tag{C.14}$$

where the estimate

$$e^x - 1 - x = \sum_{n=0}^{\infty} \frac{x^n}{n!} - 1 - x = \sum_{n=2}^{\infty} \frac{x^n}{n!} \geq \frac{1}{2}x^2, \quad x \geq 0, \tag{C.15}$$

with $x = \left|\log \frac{p(\mathbf{x}|\mathbf{y})}{q_{\boldsymbol{\phi}}(\mathbf{x})}\right|$ was used. Now use [Ghosal et al., 2000, Lemma 8.3] to bound Equation (C.14) from above by

$$2\sqrt{C}h(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})), \tag{C.16}$$

where

$$h(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) = \sqrt{\int \left(\sqrt{q_{\boldsymbol{\phi}}(\mathbf{x})} - \sqrt{p(\mathbf{x}\,|\,\mathbf{y})}\right)^2 \mathrm{d}\mathbf{x}} \tag{C.17}$$

is the Hellinger distance. [Reiss, 2012, Lemma A.3.5] gives the bound $h(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y})) \leq \sqrt{\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y}))}$. Combining these estimates yields the claimed result. □

C.2.4 Proof of Proposition 5.4.2

Proof. Start by defining the short-cuts

$$A = f_{\boldsymbol{\phi}}(\mathbf{x}), \qquad B = \left(\partial_{\phi_i} \log q_{\boldsymbol{\phi}}\right)(\mathbf{x}). \tag{C.18}$$

Now compute the difference of the variances of the estimators to the leading order in S,

$$\begin{aligned}
\mathrm{Var}(\widehat{g}_{\mathrm{Reinforce},i}) - \mathrm{Var}(\widehat{g}_{\mathrm{VarGrad},i}) = \\
&\frac{1}{S}\mathrm{Var}(AB) + \frac{S-2}{S(S-1)}\mathbb{E}\left[(A-\mathbb{E}[A])(B-\mathbb{E}[B])\right]^2 && \text{(C.19a)}\\
&- \frac{\mathrm{Var}(A)\,\mathrm{Var}(B)}{S(S-1)} - \frac{1}{S}\mathbb{E}\left[(A-\mathbb{E}[A])^2(B-\mathbb{E}[B])^2\right] && \text{(C.19b)}\\
=&\frac{1}{S}\left(\mathbb{E}\left[A^2B^2\right] - \mathbb{E}[AB]^2\right) + \frac{S-2}{S(S-1)}\mathbb{E}[AB]^2 && \text{(C.19c)}\\
&- \frac{1}{S}\left(\mathbb{E}\left[A^2B^2\right] - 2\mathbb{E}[A]\mathbb{E}\left[AB^2\right] + \mathbb{E}[A]^2\mathbb{E}\left[B^2\right]\right) + \mathcal{O}\left(\frac{1}{S^2}\right) && \text{(C.19d)}\\
=&-\frac{1}{S(S-1)}\mathbb{E}[AB]^2 - \frac{1}{S}\mathbb{E}[A]\left(\mathbb{E}[A]\mathbb{E}\left[B^2\right] - 2\mathbb{E}\left[AB^2\right]\right) + \mathcal{O}\left(\frac{1}{S^2}\right) && \text{(C.19e)}\\
=&\frac{1}{S}\mathbb{E}[A]\left(2\mathbb{E}\left[AB^2\right] - \mathbb{E}[A]\mathbb{E}\left[B^2\right]\right) + \mathcal{O}\left(\frac{1}{S^2}\right) && \text{(C.19f)}\\
=&\frac{1}{S}\mathbb{E}[A]\mathbb{E}\left[B^2\right]\left(2\delta_i^{\mathrm{CV}} + \mathbb{E}[A]\right) + \mathcal{O}\left(\frac{1}{S^2}\right) && \text{(C.19g)}
\end{aligned}$$

and note that with $\mathbb{E}\left[B^2\right] > 0$ the leading term is positive if

$$\mathbb{E}[A]\delta_i^{\mathrm{CV}} + \frac{1}{2}\mathbb{E}[A]^2 > 0, \tag{C.19h}$$

which is equivalent to the statement in the proposition. □

C.2.5 Proof of Corollary C.1.4

Proof. Note that with Proposition C.1.1,

$$\left|\frac{\delta_i^{\mathrm{CV}}}{\mathbb{E}_{q_\phi}[\mathbf{a}_{\mathrm{VarGrad}}]}\right| \to 0 \tag{C.20}$$

for $D \to \infty$, assuming that $\mathrm{KL}(q_\phi(\mathbf{x}) \,||\, p(\mathbf{x}\,|\,\mathbf{y}))$ is strictly increasing in D. Therefore, for large enough D, the condition from Proposition 5.4.2 (see Equation (5.15)), is fulfilled and the statement follows immediately. □

C.2.6 Dimension-dependence of the KL-divergence

The following lemma shows that the KL-divergence increases with the number of dimensions. This result follows from the chain-rule of KL divergence, see, *e.g.*, Cover

and Thomas [2012].

Lemma C.2.1. *Let $u^{(D)}(z_1, \ldots, z_D)$ and $v^{(D)}(z_1, \ldots, z_D)$ be two arbitrary probability distributions on $\mathbb{R}^D$. For $J \in \{1 \ldots, D\}$ denote their marginals on the first J coordinates by $u^{(J)}$ and $v^{(J)}$, i.e.,*

$$u^{(J)}(z_1, \ldots, z_J) = \int \cdots \int u^{(D)}(z_1, \ldots, z_D)\, \mathrm{d}z_{J+1} \ldots \mathrm{d}z_D, \tag{C.21}$$

and

$$v^{(J)}(z_1, \ldots, z_J) = \int \cdots \int v^{(D)}(z_1, \ldots, z_D) dz_{J+1} \ldots \mathrm{d}z_D. \tag{C.22}$$

Then

$$\mathrm{KL}(u^{(1)} \,||\, v^{(1)}) \leq \mathrm{KL}(u^{(2)} \,||\, v^{(2)}) \leq \ldots \leq \mathrm{KL}(u^{(D)} \,||\, v^{(D)}), \tag{C.23}$$

i.e. the function $J \mapsto \mathrm{KL}(u^{(J)} \,||\, v^{(J)})$ is increasing.

C.3 Details of the Experiments

C.3.1 Logistic Regression

This section describes the experimental setup of the Bayesian logistic regression example which was discussed in Section 4.6.

Data. A synthetic dataset with $N = 100$ was used, where input-output pairs are generated as follows: sample a design matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ for the inputs uniformly on $[-1, 1]$, random weights $\mathbf{w} \in \mathbb{R}^D$ from $\mathcal{N}(0, 25\, \mathrm{I}_D)$ and a random bias $b \in \mathbb{R}$ from $\mathcal{N}(0, 1)$. Set $\mathbf{p} = \sigma(\mathbf{X}\mathbf{w} + b\,\mathbf{1})$, where $\mathbf{1}$ is an N-dimensional vector of ones and $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic sigmoid applied elementwise. Finally, sample the outputs $Y \sim \mathrm{Bernoulli}(\mathbf{p})$.

Training. For all the experiments listed in the main text, the VarGrad estimator was used for the gradients of the logistic regression models. The models were trained using stochastic gradient descent [Robbins and Monro, 1951] with a learning rate of 0.001.

Estimation of intractable quantities. To estimate the intractable quantities in Figure 5.1, Monte Carlo sampling was used with 2000 samples for $\boldsymbol{\delta}^{\mathrm{CV}}$ and $\mathbb{E}_{q_\phi}[\mathbf{a}_{\mathrm{VarGrad}}]$. The KL divergence was estimated with the identity $\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{x}) | p(\mathbf{x} \,|\, \mathbf{y})) = \log p(\mathbf{y}) - \mathrm{ELBO}(\boldsymbol{\phi})$, where $\log p(\mathbf{y})$ is estimated using importance sampling with

10000 samples and ELBO($\boldsymbol{\phi}$) using standard Monte Carlo sampling with 2000 samples.

For the variance estimates in Figure 5.3, 1000 Monte Carlo samples were used. As explained in the main text, to estimate the control variate coefficients, either 2 samples for the *sampled estimator* or 1000 samples for the *oracle estimator* were used.

C.3.2 Discrete VAEs

This section describes the experimental setting for the Discrete VAE, where it closely follows the setup in Maddison et al. [2017], which was also replicated in Tucker et al. [2017] and Grathwohl et al. [2018]. As the aim is to compare the usefulness of different estimators in the optimisation and time their run-times, the various benchmarks were re-implemented using JAX [Bradbury et al., 2018; Hennigan et al., 2020]. Extra care was taken to be as faithful as possible to the implementation description in their respective papers as well as in optimising the run-time of the implementations.

Data. A fixed binarisation of Omniglot [Lake et al., 2015] was used, where it was binarised at the standard cut-off of 0.5. The standard train/test splits for this dataset was used.

Model Architectures. The DVAE experiments used the *two-layer linear* architecture, which has 2 stochastic binary layers with 200 units each, which was used in Maddison et al. [2017]. Furthermore, for some results, a *one layer non-linear architecture* was also used. For both models, the decoders mirror the corresponding encoders. A Bernoulli(0.5) prior was used on the latent space with fixed parameters.

Training. For training the models, the Adam optimiser [Kingma and Ba, 2015] was used with learning rates 0.001, 0.0005 and 0.0001.

Estimation of intractable quantities . Monte Carlo sampling with 2000 samples was used for $\boldsymbol{\delta}^{\mathrm{CV}}$ and $\mathbb{E}_{q_{\boldsymbol{\phi}}}[\mathbf{a}_{\mathrm{VarGrad}}]$. Due to the high memory requirements of these computations and sparsity of the weight gradients, these quantities were only only computed for the biases. To estimate the gradient variances 1000 Monte Carlo samples were used.

APPENDIX D

Appendix for Generalised Bayesian Filtering

D.1 β-PF

D.1.1 Outline derivation of the loss in (6.11)

To arrive at the expression of the loss in (6.11), recall the formula for the beta divergence [Cichocki and Amari, 2010]

$$\begin{aligned} D_B^\beta(\mathbf{P} \,||\, \mathbf{Q}) =& \frac{1}{\beta(\beta+1)} \int (p^{\beta+1}(x) + \beta q^{\beta+1}(x) - (\beta+1)p(x)q^\beta(x)) \,\mathrm{d}\mu(x) \\ =& C_{\mathbf{P}} + \frac{1}{\beta+1} \int q^{\beta+1}(x) \,\mathrm{d}x - \frac{1}{\beta} \int q(x) \mathbf{P}(\mathrm{d}x) \end{aligned}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are probability measures on the measurable space $(X, \mathcal{A})$ and μ is a finite or σ-finite measure on this space, such that $\mathbf{P} \ll \mu$ and $\mathbf{Q} \ll \mu$ are absolutely continuous w.r.t. μ and $C_{\mathbf{P}}$ is a constant independent of $\mathbf{Q}$. Finally, $p = \frac{\mathrm{d}\mathbf{P}}{\mathrm{d}\mu}$ and $q = \frac{\mathrm{d}\mathbf{Q}}{\mathrm{d}\mu}$ are densities and the Radon-Nikodym derivatives for $\mathbf{P}$ and $\mathbf{Q}$ w.r.t. μ. Comparison with (2.91) yields (6.6) directly.

D.1.2 β-BPF

This section provides the algorithmic procedure in Algorithm 7 for the β-BPF that is investigated in this main chapter.

Algorithm 7 β-Bootstrap Particle Filter

Input: Observation sequence $\mathbf{y}_{1:T}$, number of samples N.
Initialise: Sample $\{\bar{\mathbf{x}}_0^{(i)}\}_{i=1}^N$ for the prior $\pi_0(\mathbf{x}_0)$.
for $t = 1$ **to** T **do**
  Sample: $\check{\mathbf{x}}_t^{(i)} \sim f_t(\mathbf{x}_t \,|\, \bar{\mathbf{x}}_{t-1}^{(i)})$ ▷ **for** $i = 1$ **to** N.
  Weight: $\mathsf{w}_t^{(i)} \propto G_t^\beta(\check{\mathbf{x}}_t^{(i)})$ ▷ **for** $i = 1$ **to** N.
  Resample: $\bar{\mathbf{x}}_t^{(i)} \sim \sum_{i=1}^N \mathsf{w}_t^{(i)} \delta_{\check{\mathbf{x}}_t^{(i)}}(\mathrm{d}\mathbf{x}_t)$ ▷ **for** $i = 1$ **to** N.
end for

D.1.3 β-APF

This section provides the algorithmic procedure in Algorithm 8 for the β-APF. Here q_t denotes the proposal distribution at time t which in the case of the fully-adapted APF would be chosen to be the conditional density of $\mathbf{x}_t$ given $\mathbf{x}_{t-1}$ and $\mathbf{y}_t$ but in general would be chosen as some approximation of this distribution and $\check{G}_t^\beta(\mathbf{x}_{t-1})$ is chosen as an approximation of the predictive generalised likelihood, i.e. $\check{G}_t^\beta(\mathbf{x}_{t-1}) \approx \int G_t^\beta(\mathbf{x}_t) f_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \,\mathrm{d}\mathbf{x}_t$.

Algorithm 8 β-Auxiliary Particle Filter

Input: Observation sequence $\mathbf{y}_{1:T}$, number of samples N.
Initialise: Sample $\{\bar{\mathbf{x}}_0^{(i)}\}_{i=1}^N$ independently from the prior $\pi_0(\mathbf{x}_0)$.
for $t = 1$ **to** T **do**
  Sample:

$$k^{(i)} \sim \mathbb{P}\left(i = k \,|\, \mathbf{y}_t\right) \propto \mathsf{w}_{t-1}^{(i)} \check{G}_t^\beta(\bar{\mathbf{x}}_t^{(i)})$$
$$\bar{\mathbf{x}}_t^{(i)} \sim q_t(\bar{\mathbf{x}}_t \,|\, \bar{\mathbf{x}}_{t-1}^{k^{(i)}})$$

  ▷ **for** $i = 1$ **to** N.
  Weight:

$$\mathsf{w}_t^{(i)} \propto \frac{f_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \bar{\mathbf{x}}_{t-1}^{k^{(i)}}) G_t^\beta(\bar{\mathbf{x}}_t^{(i)})}{q_t(\bar{\mathbf{x}}_t^{(i)} \,|\, \bar{\mathbf{x}}_{t-1}^{k^{(i)}}) \check{G}_t^\beta(\bar{\mathbf{x}}_{t-1}^{k^{(i)}})}$$

  ▷ **for** $i = 1$ **to** N.
end for

As in the case of the standard APF, the use of reference points obtained from the current states in which one sets $\check{G}_t^\beta(\mathbf{x}_{t-1}) = G_t^\beta(\mu_t(\mathbf{x}_{t-1}))$ with $\mu_t(\mathbf{x}_{t-1}) = \int \mathbf{x}_t f(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \,\mathrm{d}x_t$ is one simple approach to this, but one which does not work well in full generality because it is underdispersed with respect to the true predictive generalised likelihood (*cf.* [Johansen and Doucet, 2008]). In general, better performance can be obtained by developing a good bespoke approximation to the pre-

dictive generalised likelihood and the locally-optimal proposal density for any given application. However, in order to provide a simple generically-applicable strategy that is reasonably robust, one suggestion is to sett the proposal equal to the transition density, *i.e.*, $q_t = f_t$, and to use a stabilised version of the simple approximation to the predictive likelihood, provided by

$$\check{G}_t^\beta(\mathbf{x}_{t-1}) = G_t^\beta(\mu_t(\mathbf{x}_{t-1})) + c_t$$

where c_t is a constant chosen, *e.g.*, as $0.05 \sup_{\mathbf{x}} G_t^\beta(\mathbf{x})$ to avoid any instability in the weighting step. Such a strategy was advocated in the iterated version of this algorithm described by Guarniero et al. [2017] which could in principle also be adapted to the GBI setting.

D.2 Theoretical analysis

D.2.1 Proof of Theorem 6.4.1

This is an adaptation of a well-known proof, hence the results will only be sketched, providing the constant $c_{t,p,\beta}$.

The result is proved via induction. For $t = 0$, the result in the theorem holds trivially, as it corresponds to the i.i.d. case , *e.g.*, Del Moral [2004, Lemma 7.3.3] provides an explicit constant. Hence, as an induction hypothesis, assume

$$\|\pi_{t-1}^{\beta,N}(\varphi) - \pi_{t-1}^{\beta}(\varphi)\|_p \leq \frac{c_{t-1,p,\beta}\|\varphi\|_\infty}{\sqrt{N}}, \tag{D.1}$$

where $c_{t-1,p,\beta} < \infty$ is independent of N. After the sampling step, obtain the predictive particles $\bar{\mathbf{x}}_t^{(i)}$ are obtained and the predictive measure

$$\hat{\pi}_t^{\beta,N}(\mathrm{d}\mathbf{x}_t \,|\, \mathbf{y}_{1:t-1}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{\bar{\mathbf{x}}_t^{(i)}}(\mathrm{d}\mathbf{x}_t),$$

is formed. One can show that [Míguez et al., 2013, Lemma 1]

$$\|\hat{\pi}_t^{\beta,N}(\varphi) - \hat{\pi}_t^{\beta}(\varphi)\|_p \leq \frac{c_{1,t,p,\beta}\|\varphi\|_\infty}{\sqrt{N}}, \tag{D.2}$$

where $c_{1,t,p,\beta} < \infty$ is a constant independent of N. After the computation of weights, one can construct

$$\check{\pi}_t^{\beta,N}(\mathrm{d}\mathbf{x}_t) = \sum_{i=1}^{N} \mathsf{w}_t^{(i)} \delta_{\bar{\mathbf{x}}_t^{(i)}}(\mathrm{d}\mathbf{x}_t). \tag{D.3}$$

Following again [Míguez et al., 2013, Lemma 1], one readily obtains

$$\|\pi_t^{\beta}(\varphi) - \check{\pi}_t^{\beta,N}(\varphi)\|_p \leq \frac{c_{2,t,p,\beta}\|\varphi\|_\infty}{\sqrt{N}}, \tag{D.4}$$

where

$$c_{2,t,p,\beta} = \frac{2\|G_t^\beta\|_\infty c_{1,t,p,\beta}}{\hat{\pi}_t(G_t^\beta)} < \infty,$$

where $\hat{\pi}_t(G_t^\beta) > 0$. Finally, performing multinomial resampling leads to a condition-

ally i.i.d. sampling case, which yields

$$\|\breve{\pi}_t^{\beta,N}(\varphi) - \pi_t^{\beta,N}(\varphi)\|_p \leq \frac{c_{3,t,p,\beta}\|\varphi\|_\infty}{\sqrt{N}}. \tag{D.5}$$

Combining (D.4) and (D.5) yields the result with $c_{t,p,\beta} = c_{2,t,p,\beta} + c_{3,t,p,\beta}$.

D.2.2 Proof of Theorem 6.4.3

The Proposition in Johansen and Doucet [2008] which provides explicit expressions for sequential importance resampling based particle filters within the general frameworks of Del Moral [2004]; Chopin [2004]; the same argument holds *mutatis mutandis* in the context of the β-BPF. The asymptotic variance expression $\sigma^2_{t,\beta}(\varphi)$ is given as follows. For $t = 1$ [Johansen and Doucet, 2008],

$$\sigma^2_{1,\beta}(\varphi) = \int \frac{\mathsf{p}_1^\beta(\mathbf{x}_1 \,|\, \mathbf{y}_1)}{f_1(\mathbf{x}_1)}(\varphi_1(\mathbf{x}_1) - \overline{\varphi}_1)^2 \,\mathrm{d}\mathbf{x}_1,$$

where $f_1(\mathbf{x}_1) := \int \mu_0(\mathbf{x}_0) f_1(\mathbf{x}_1 \,|\, \mathbf{x}_0) \,\mathrm{d}\mathbf{x}_0$. Then, for $t > 1$ [Johansen and Doucet, 2008]

$$\begin{aligned}
\sigma^2_{t,\beta} = &\int \frac{\mathsf{p}_t^\beta(\mathbf{x}_1 \,|\, \mathbf{y}_{1:t})^2}{f_1(\mathbf{x}_1)} \left(\int \varphi_t(\mathbf{x}_{1:t}) \mathsf{p}_t^\beta(\mathbf{x}_{2:t} \,|\, \mathbf{y}_{2:t}, \mathbf{x}_1) \,\mathrm{d}\mathbf{x}_{2:t} - \overline{\varphi}_t \right)^2 \mathrm{d}\mathbf{x}_1 \\
&+ \sum_{k=2}^{t-1} \int \frac{\mathsf{p}_k^\beta(\mathbf{x}_{1:k} \,|\, \mathbf{y}_{1:t})^2}{\mathsf{p}_{k-1}^\beta(\mathbf{x}_{1:k-1} \,|\, \mathbf{y}_{1:k-1}) f_k(\mathbf{x}_k \,|\, \mathbf{x}_{k-1})} \\
&\left(\int \varphi_t(\mathbf{x}_{1:t}) \mathsf{p}_t^\beta(\mathbf{x}_{k+1:t} \,|\, \mathbf{y}_{k+1:t}, \mathbf{x}_k) \,\mathrm{d}\mathbf{x}_{k+1:t} - \overline{\varphi}_t \right)^2 \mathrm{d}\mathbf{x}_{1:k} \\
&+ \int \frac{\mathsf{p}_t^\beta(\mathbf{x}_{1:t} \,|\, \mathbf{y}_{1:t})^2}{\mathsf{p}_{t-1}^\beta(\mathbf{x}_{1:t-1} \,|\, \mathbf{y}_{1:t-1}) f_t(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})} (\varphi_t(\mathbf{x}_{1:t}) - \overline{\varphi}_t)^2 \,\mathrm{d}\mathbf{x}_{1:t}.
\end{aligned}$$

D.3 Experiment Details

D.3.1 Evaluation Metrics

The following metrics metrics are used to evaluate the experiments:

The Normalised Mean Squared Error (NMSE) is computed per state dimension j as

$$\text{NMSE}_j = \frac{\left\| \sum_{t=1}^{T} x_{tj} - \hat{x}_{tj} \right\|_2^2}{\sum_{t=1}^{T} \|x_{tj}\|_2^2}, \tag{D.6}$$

with $\hat{x}_{tj} = \frac{1}{N} \sum_{i=1}^{N} \bar{x}_{tj}^{(i)}$, *i.e.*, the mean over resampled particles (trajectories).

The 90% Emprical Coverage (EC) is computed per state dimension j as

$$\text{EC}_j = \frac{\sum_{t=1}^{T} \mathbb{1}_{C_t}(x_{tj})}{T}, \tag{D.7}$$

with

$$C_t = \{z | z \in [\text{q}_{0.05}(\{\bar{x}_{tj}^{(i)}\}_{i=1}^{N}), \text{q}_{0.95}(\{\bar{x}_{tj}^{(i)}\}_{i=1}^{N})]\},$$

where q is the quantile function.

Predictive Median Absolute Error (MedAE) is computed per observation dimension j as

$$\text{MedAE} = \text{MEDIAN}_{t \in \{1,\ldots,T\}} \left(|\hat{y}_{tj} - y_{tj}| \right), \tag{D.8}$$

where $\hat{\mathbf{y}}_t \sim \sum_{i=1}^{N} \mathsf{w}_t^i g_t(\mathbf{y} \,|\, \mathbf{x}_t^{(i)})$.

Aggregation: Metrics are often presented as aggregates over the state dimensions, which are simply the mean of the metric across the state dimensions.

D.3.2 Details on the implementation of the selection criterion in Section 6.3.3

From (6.13), agg was chosen as the median and $\mathcal{L}$ as the absolute error. When the observations are multidimensional, the average loss was taken, weighted by the inverse of the median of each dimension.

The scores for different values of β from a grid were computed and the β that minimises the score was chosen. For multiple runs, the modal value of the β's over all the runs was reported.

In the interest of simplicity, the entire observation sequence from an alternative realisation of the same simulation was used to compute the score. However, in practice one might one to tune β on a sub-sequence to avoid extra computation.

D.3.3 Wiener velocity model experiment details (Section 6.5.1)

This section details the experimental setup used to obtain the results for Section 6.5.1.

Simulator settings The data was synthesised with a Python simulator utilising NumPy. The system was discretised with $\Delta\tau = 0.1$ and simulated for 100 time steps, *i.e.*. 1000 time points in total were obtained. For the state evolution process in Eq. (6.16), the transition matrix was set to $\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta\tau & 0 \\ 0 & 1 & 0 & \Delta\tau \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and the transition covariance matrix to $\mathbf{Q} = \begin{bmatrix} \frac{\Delta\tau^3}{3} & 0 & \frac{\Delta\tau^2}{2} & 0 \\ 0 & \frac{\Delta\tau^3}{3} & 0 & \frac{\Delta\tau^2}{2} \\ \frac{\Delta\tau^2}{2} & 0 & \Delta\tau & 0 \\ 0 & \frac{\Delta\tau^2}{2} & 0 & \Delta\tau \end{bmatrix}$. For the observation process in Eq. (6.17), the observation matrix was set to $\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ and the noise covariance to $\Sigma = \mathbf{I}$. The initial state of the simulator is set to $\mathbf{x}_0 = [140, 140, 50, 0]$.

Contamination To simulate contaminated observations extra i.i.d. Gaussian noise with a standard deviation of 100.0 was applied additively to the observation sequence with probability p_c per observation.

Sampler settings The samplers were intialised by sampling from the prior given by $\mathcal{N}(\mathbf{x}_0, \mathbf{Q})$ with $\mathbf{x}_0$ being the initial state of the simulator and $\mathbf{Q}$ as above. The likelihood covariance was set to the simulator noise covariance and the number of samples to 1000.

Experiment settings Each experiment consists of 100 runs, where all samplers are seeded with the same seed per run; however, the seeds vary across the runs. The same state sequence was used for all runs obtained from the simulator as above. However, each run simulates a new observation sequence (*i.e.*, the observations noise changes per run).

D.3.4 Terrain Aided Navigation (TAN) experiment details (Section 6.5.2)

This section details the experimental setup used to obtain the results for Section 6.5.2.

Simulator settings The data was synthesised with a Python simulator utilising NumPy. The system was discretised with $\Delta\tau = 0.1$ and simulated for 200 time steps, *i.e.*, 2000 time points were obtained in total. For the state evolution process in Eq. (6.16)the transition matrix was set to

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & \Delta\tau & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta\tau & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta\tau \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

and the transition covariance matrix to

$$\mathbf{Q} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 36 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0841 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.207936 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5.29 \end{bmatrix}.$$

For the observation process, the non-linear observation function was set to

$$h(\mathbf{x}_t) = \begin{bmatrix} x_{t3} - \mathrm{DEM}(x_{t1}, x_{t2}) \\ \sqrt{(x_{t1} - x_{01})^2 + (x_{t2} - x_{02})^2} \end{bmatrix},$$

where DEM is a non-analytic Digital Elevation Map. For this simulation the DEM was set to

$$\mathrm{DEM(a,\ b)} = \mathrm{peaks}(q \cdot a, q \cdot b) + \sum_{i=1}^{6} \alpha_i \sin(\omega_i \cdot q \cdot a) \cos(\psi \cdot q \cdot b),$$

with $\mathrm{peaks}(c, d) = 200(3(1-c)^2 \exp(-c^2 - (d+1)^2) - 10(\frac{c}{5} - c^3 - d^5) \exp(-c^2 - d^2) - \frac{1}{3}\exp(-(x+1)^2 + y^2))$, $\boldsymbol{\alpha} = [300, 80, 60, 40, 20, 10]$, $\boldsymbol{\omega} = [5, 10, 20, 30, 80, 150]$, $\boldsymbol{\psi} = [4, 10, 20, 40, 90, 150]$ and $q = \frac{3}{2.96 \times 10^4}$. The noise covariance $\Sigma = \sigma^2\mathbf{I}$ with $\sigma^2 = 400$. The initial state of the simulator was set $\mathbf{x}_0 = [-7.5 \times 10^3, 5 \times 10^3, 1.1 \times 10^3, 88.15, -60.53, 0]$.

Contamination To simulate contaminated observations extra i.i.d. Student's t noise was applied additively, with 1 degree of freedom and scale σ, where σ is given

as above. The contamination was applied to observation instances with probability p_c per observation.

Sampler settings The samplers were initialised by sampling from the prior given by $\mathcal{N}(\mathbf{x}_0, \mathbf{Q})$ with $\mathbf{x}_0$ being the initial state of the simulator and $\mathbf{Q}$ as above. The likelihood covariance was set to the simulator noise covariance and the number of samples to 3000. For the APFs, the same design choices outlined in Appendix D.1.3 were made, *i.e.*, setting the proposal density to the transition density and stabilising the predictive likelihood approximation with the given additive constant.

Experiment settings Each experiment consists of 50 runs, where all samplers are seeded with the same seed per run; however, the seeds vary across the runs. The same state sequence was used for all runs obtained from the simulator as above. However, each run simulates a new observation sequence (i.e. the observation noise changes per run).

D.3.5 Asymmetric Wiener velocity model experiment details (Section 6.5.3)

This section details the experimental setup used to obtain the results for Section 6.5.3.

Simulator settings The same simulator settings as in Appendix D.3.3 were used, but changing the observation noise to $\mathbb{1}_{[-\infty,0]}\mathcal{N}(0,1) + \mathbb{1}_{[0,+\infty]}\mathcal{N}(0,10^2)$.

Contamination To simulate contaminated observations i.i.d. Exponential noise with a scale of 1000 was applied multiplicatively with probability $p_c = 0.1$ per observation.

Sampler settings The samplers were initialised by sampling from the prior given by $\mathcal{N}(\mathbf{x}_0, \mathbf{Q})$ with $\mathbf{x}_0$ being the initial state of the simulator and $\mathbf{Q}$ as above. We set the number of samples to 1000.

Experiment settings The same settings as in Appendix D.3.3 were used.

D.3.6 Air quality experiment details (Section 6.5.4)

This section details the setup used to obtain the results for Section 6.5.4.

Data The data was obtained from https://www.londonair.org.uk/london/asp/datadownload.asp, and a time window of 200 hours was selected. No preprocessing was performed on the data.

Kernel The Mateŕn 5/2 kernel was used and the lengthscale was set to $l = 0.03$ and the signal variance was set to $\sigma_s^2 = 32$. The SDE representation of the Mateŕn 5/2 GP was discretised with a stepsize $\Delta\tau = 0.005$ to obtain an LGSSM of the form (6.16)-(6.17), with transition matrix

$$\mathbf{A} = \exp(\Delta\tau\mathbf{F}) = \exp(\Delta\tau \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\lambda^3 & -3\lambda^2 & -3\lambda \end{bmatrix}),$$

where $\lambda = \frac{\sqrt{5}}{l}$ and transition covariance matrix $\mathbf{Q} = \mathbf{P}_\infty - \mathbf{A}\mathbf{P}_\infty\mathbf{A}^\intercal$, with $\mathbf{P}_\infty = \begin{bmatrix} \sigma_s^2 & 0 & \kappa \\ 0 & \kappa & 0 \\ -\kappa & 0 & \sigma_s^2\lambda^4 \end{bmatrix}$, where $\kappa = \frac{\sigma_s^2\lambda^2}{3}$. For the observation process in (6.17), the observation matrix was set to $\mathbf{H} = [1, 0, 0]$ and the noise variance $\sigma^2 = 1$. The prior on the initial state $\mathbf{x}_0$ is given as $\mathcal{N}(\mathbf{m}, \mathbf{S})$, where $\mathbf{m}^\intercal = [0, 0, 0]$ and $S = \mathbf{P}_\infty$.

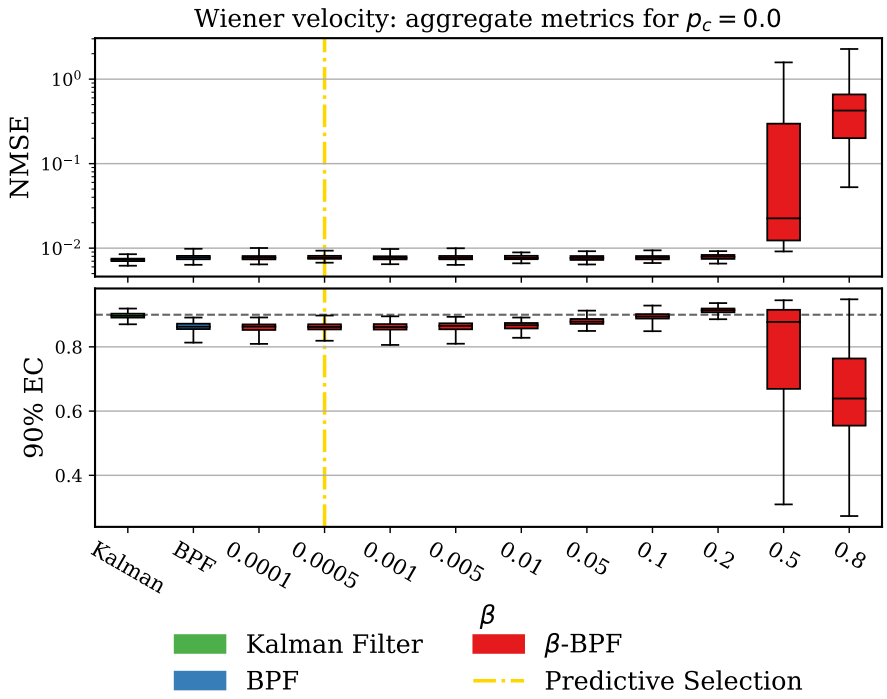
Sampler settings The samples were intialised by sampling from the prior $\mathcal{N}(\mathbf{m}, \mathbf{S})$. The number of samples was set to 1000.

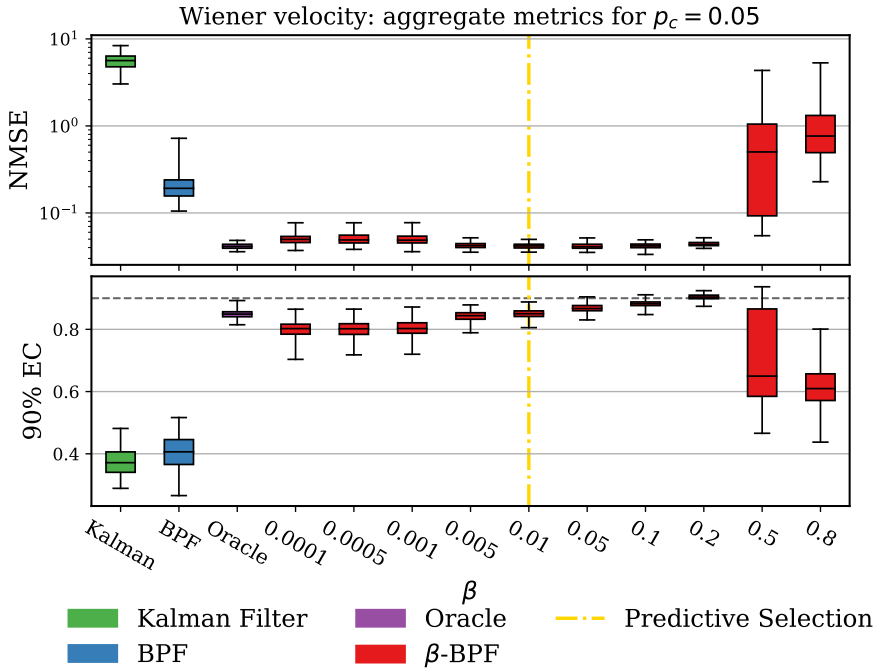
Smoother settings The number of samples was set to 1000 for the FFBS smoother.

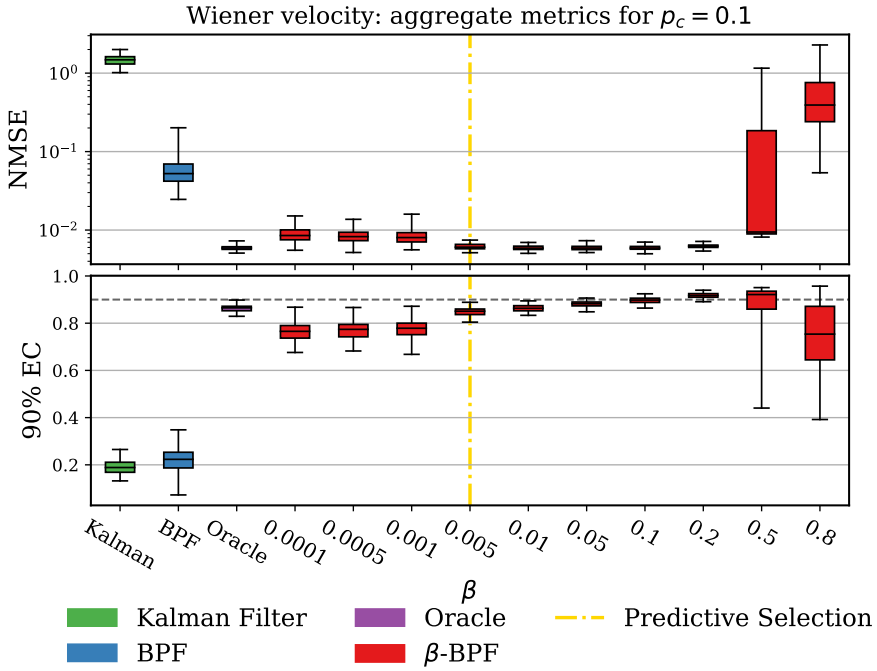
Experiment settings The sampling procedure was repeated for 100 runs, where the samplers are seeded differently for each runs. The seeds are shared among samplers for each run. The Kalman filter does not require multiple runs as the solution is deterministic.

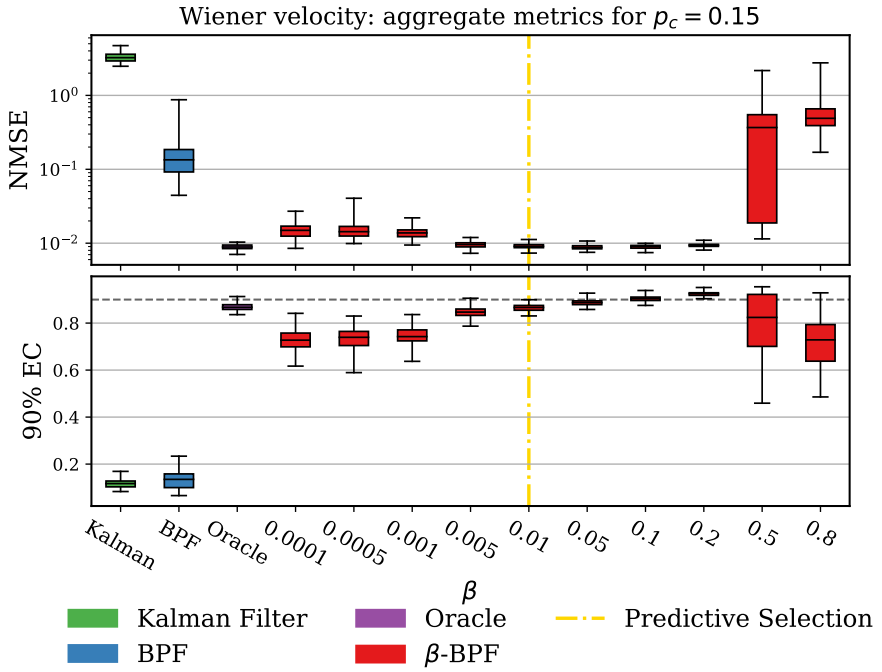
D.4 Further results

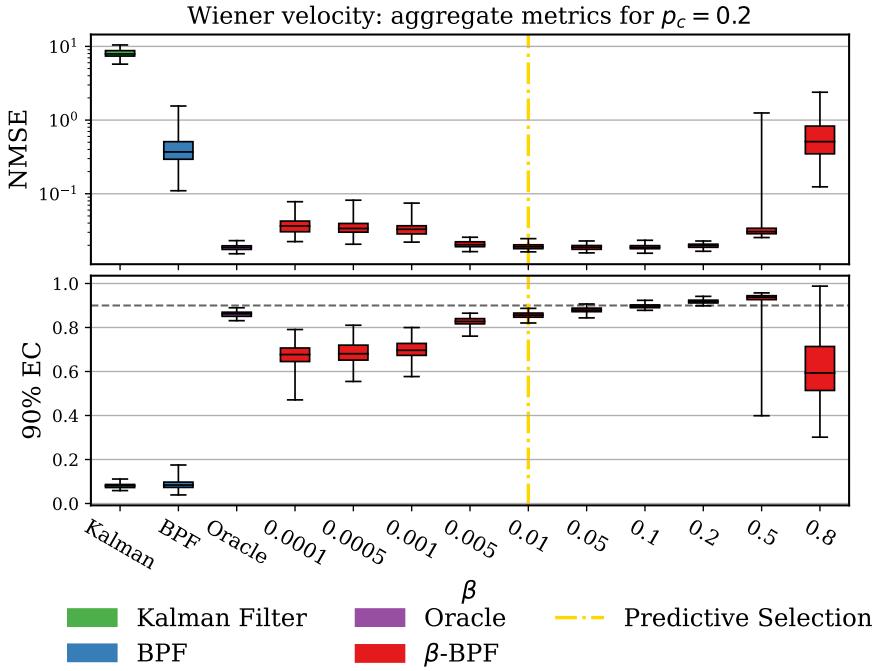
D.4.1 Wiener velocity experiment

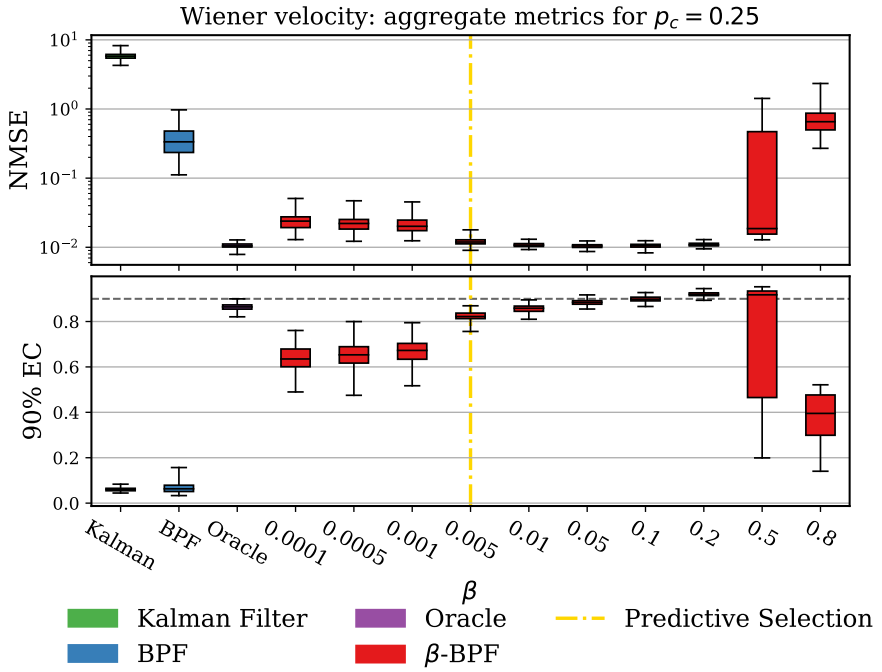


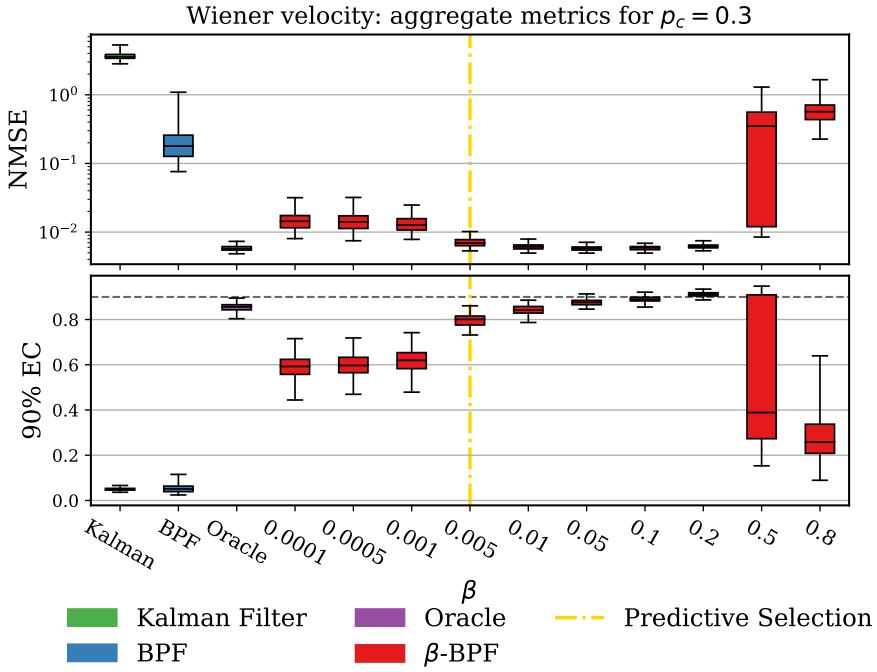

Wiener velocity: aggregate metrics for $p_c = 0.05$
NMSE
90% EC
Kalman
BPF
Oracle
0.0001
0.0005
0.001
0.005
0.01
0.05
0.1
0.2
0.5
0.8
β
Kalman Filter
BPF
Oracle
β-BPF
Predictive Selection


Wiener velocity: aggregate metrics for $p_c = 0.1$
NMSE
90% EC
Kalman
BPF
Oracle
0.0001
0.0005
0.001
0.005
0.01
0.05
0.1
0.2
0.5
0.8
β
Kalman Filter
BPF
Oracle
β-BPF
Predictive Selection


Wiener velocity: aggregate metrics for $p_c = 0.15$
NMSE
90% EC
Kalman
BPF
Oracle
0.0001
0.0005
0.001
0.005
0.01
0.05
0.1
0.2
0.5
0.8
β
Kalman Filter
BPF
Oracle
β-BPF
Predictive Selection


Wiener velocity: aggregate metrics for $p_c = 0.2$
NMSE
90% EC
Kalman
BPF
Oracle
0.0001
0.0005
0.001
0.005
0.01
0.05
0.1
0.2
0.5
0.8
β
Kalman Filter
BPF
Oracle
β-BPF
Predictive Selection


Wiener velocity: aggregate metrics for $p_c = 0.25$
NMSE
90% EC
Kalman
BPF
Oracle
0.0001
0.0005
0.001
0.005
0.01
0.05
0.1
0.2
0.5
0.8
β
Kalman Filter
BPF
Oracle
β-BPF
Predictive Selection


Wiener velocity: aggregate metrics for $p_c = 0.3$
NMSE
90% EC
Kalman
BPF
Oracle
0.0001
0.0005
0.001
0.005
0.01
0.05
0.1
0.2
0.5
0.8
β
Kalman Filter
BPF
Oracle
β-BPF
Predictive Selection

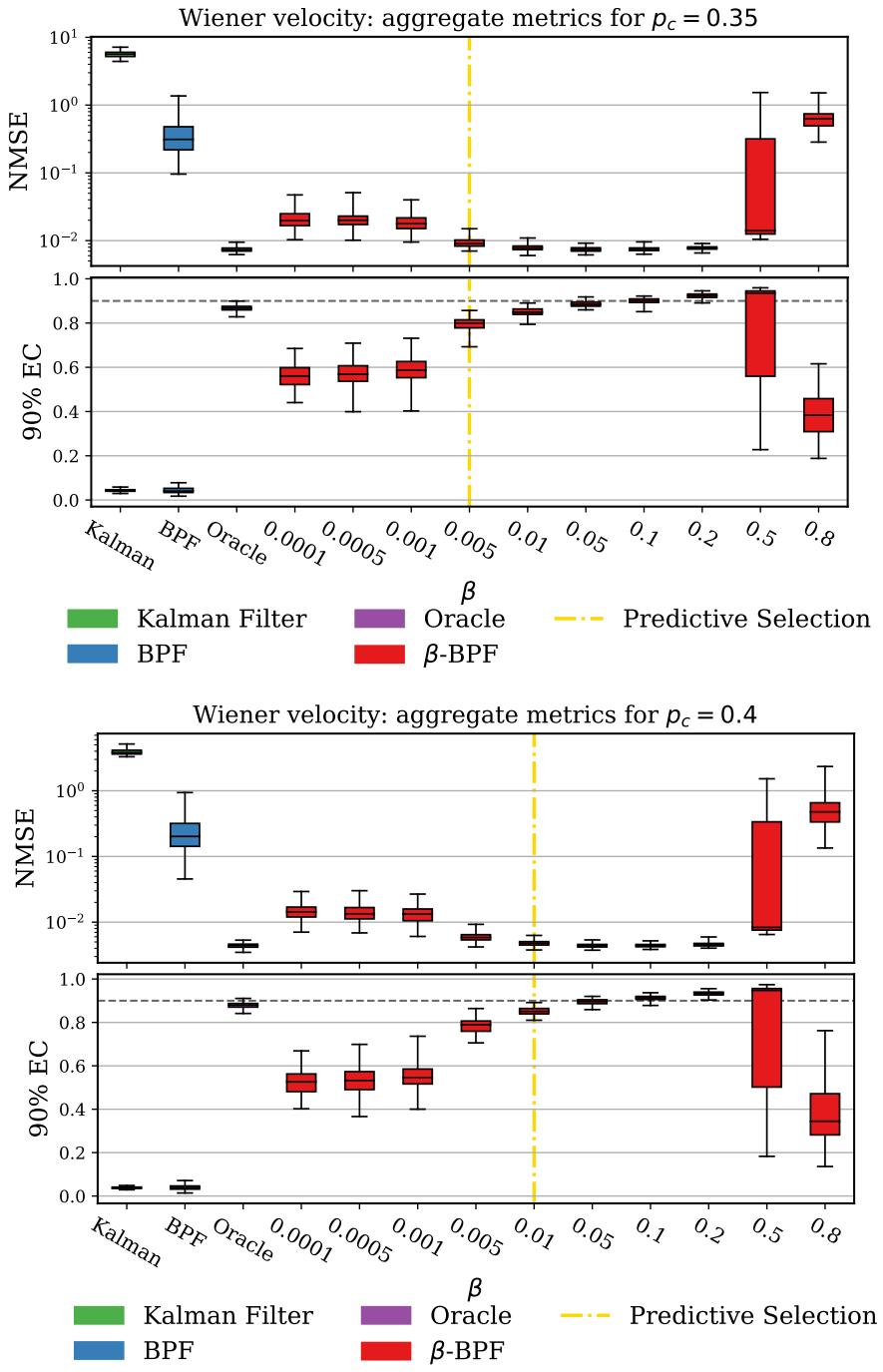


Figure D.1: The mean metrics over state dimensions for the Wiener velocity example. The top panel presents the NMSE results (lower is better) and the bottom panel presents the 90% emprirical coverage results (higher is better), on 100 runs. The vertical dashed line in gold indicate the value of β chosen by the selection criterion in Section 6.3.3. The horizontal dashed line in black in the lower panel indicates the 90% mark for the coverage.

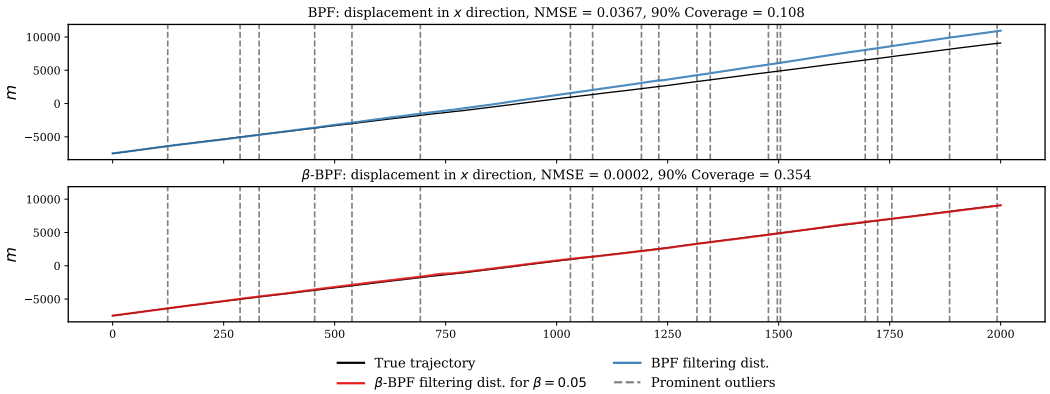


Figure D.2: Marginal filtering distributions for the Kalman filter, the BPF and the β-BPF.

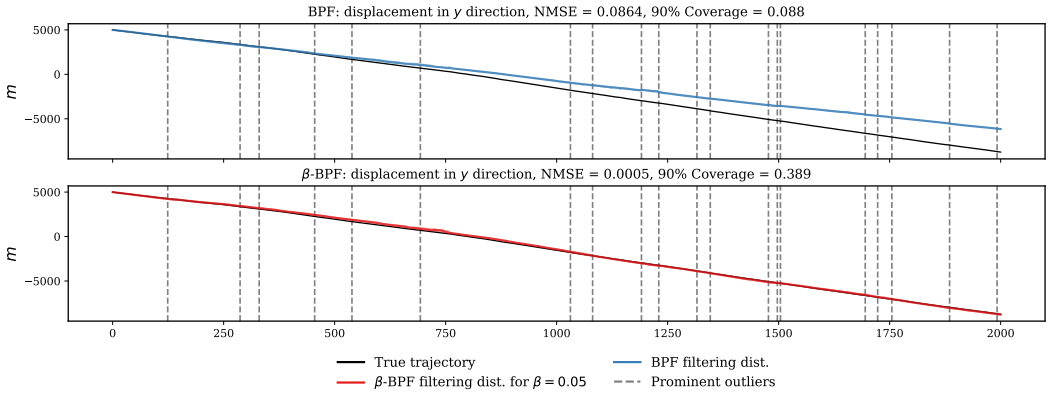


Figure D.3: Marginal filtering distributions for the Kalman filter, the BPF and the β-BPF.

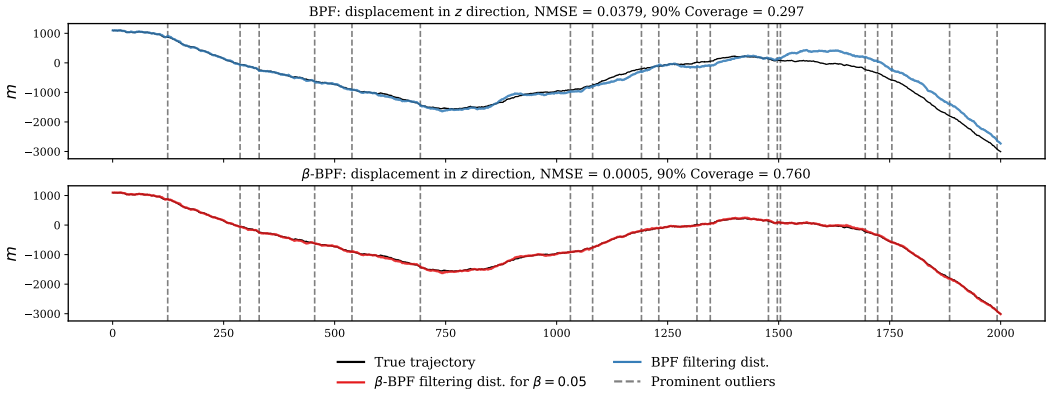


Figure D.4: Marginal filtering distributions for the Kalman filter, the BPF and the β-BPF.

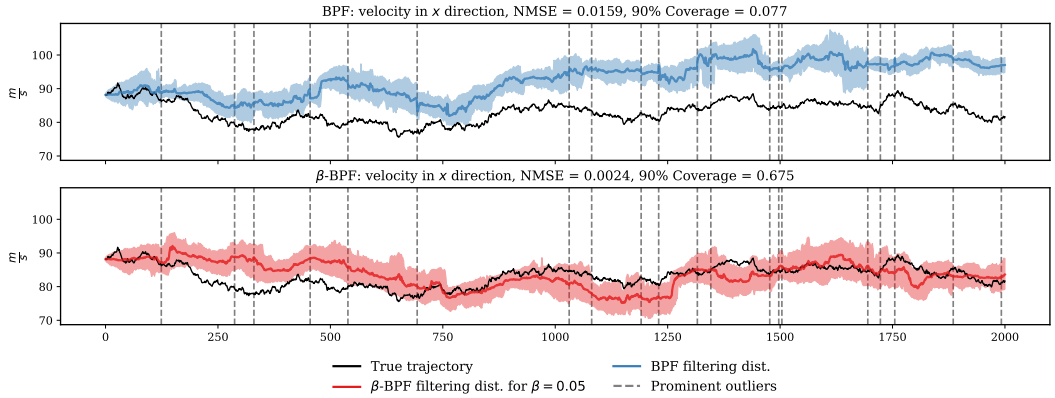


Figure D.5: Marginal filtering distributions for the Kalman filter, the BPF and the β-BPF.

	Predictive Median Absolute Error	
Filter	mean	standard error
Kalman Filter	5.23	0.06
BPF	2.78	0.09
$\beta = 0.0001$	0.97	0.00
$\beta = 0.0005$	0.97	0.00
$\beta = 0.001$	0.97	0.00
$\beta = 0.005$	0.90	0.00
$\beta = 0.01$	0.90	0.00
$\beta = 0.05$	0.90	0.00
$\beta = 0.1$	0.90	0.00
$\beta = 0.2$	0.92	0.00
$\beta = 0.5$	72.22	12.34
$\beta = 0.8$	226.61	11.62

Table D.1: Predictive results on the Weiner velocity example for $p_c = 0.1$. The one step ahead predictive performance is measure by the median absolute error. The figures are averaged across 100 runs and the standard error on the average score is provided.

D.4.2 TAN experiment

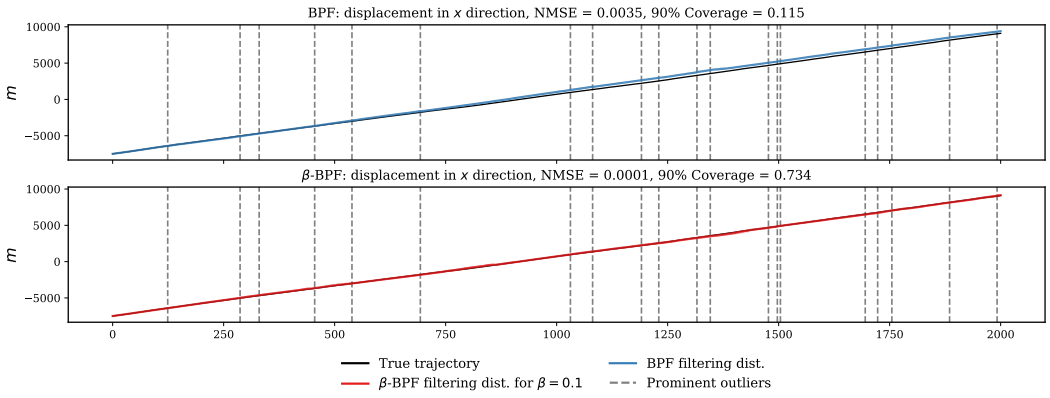


Figure D.6: Marginal filtering distributions for the BPF (top) and β-BPF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

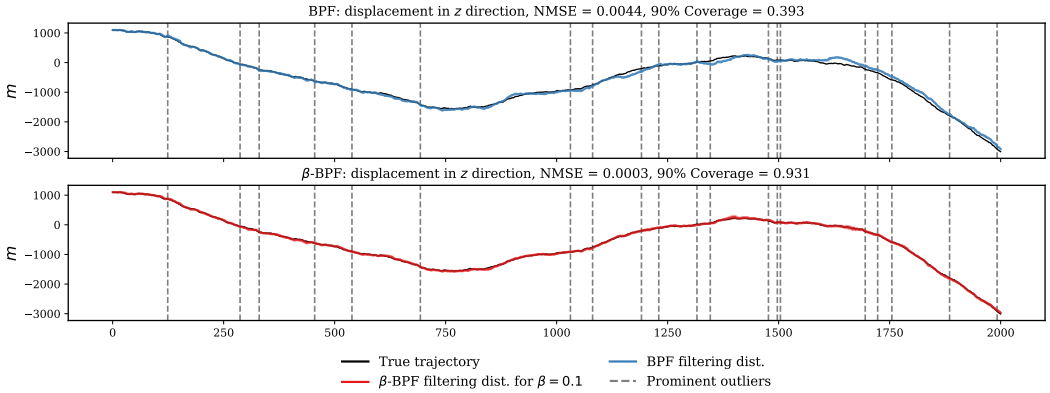


Figure D.8: Marginal filtering distributions for the BPF (top) and β-BPF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

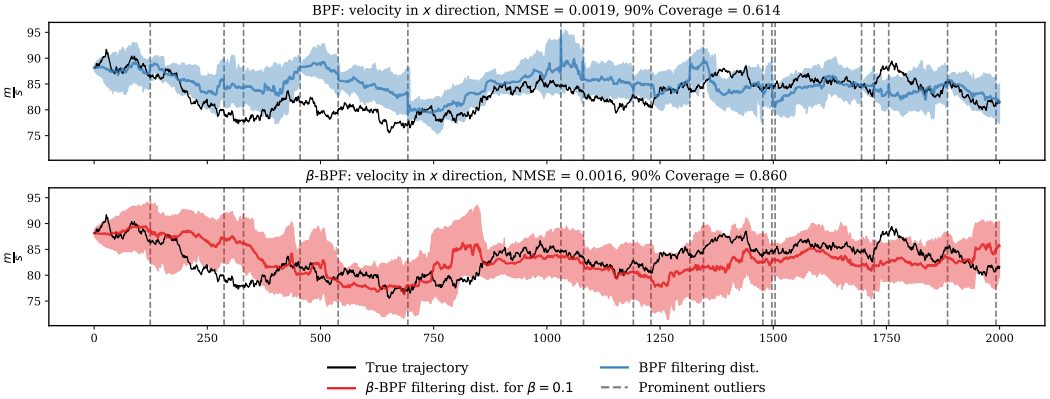


Figure D.9: Marginal filtering distributions for the BPF (top) and β-BPF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

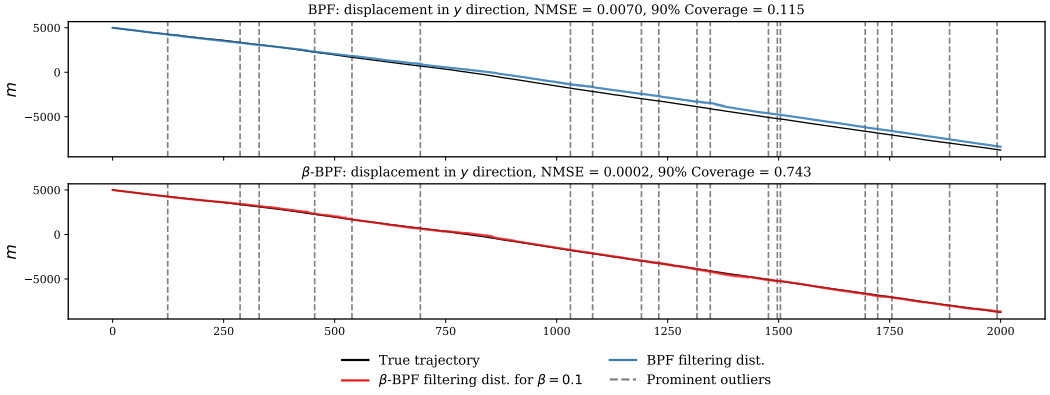


Figure D.7: Marginal filtering distributions for the BPF (top) and β-BPF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

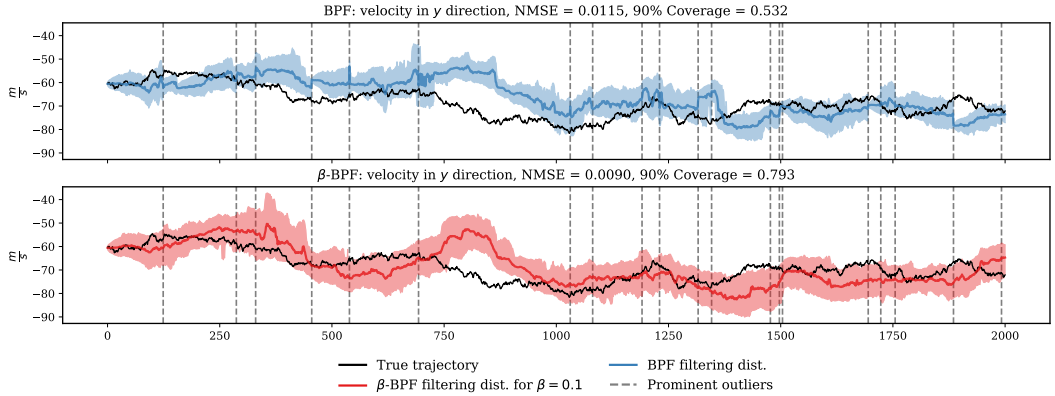


Figure D.10: Marginal filtering distributions for the BPF (top) and β-BPF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

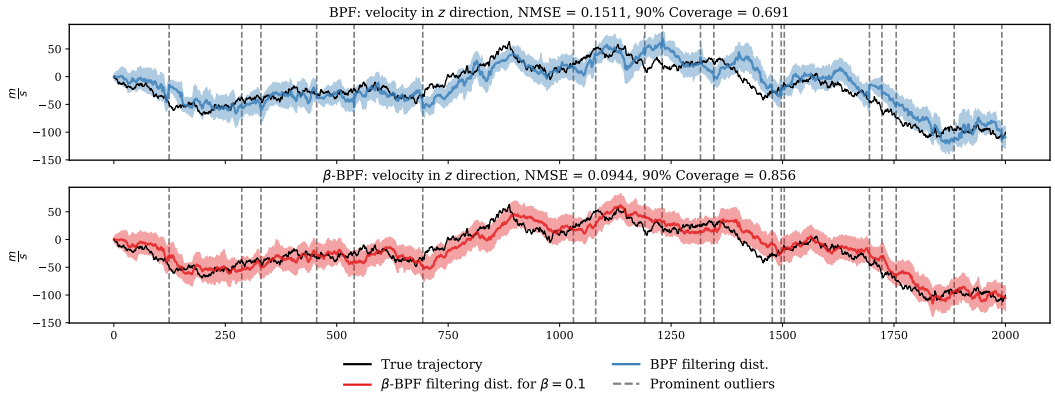


Figure D.11: Marginal filtering distributions for the BPF (top) and β-BPF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

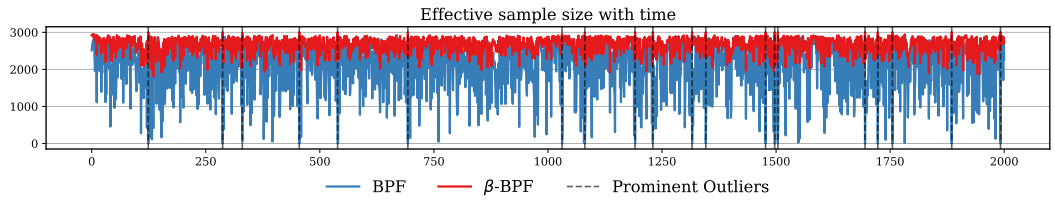


Figure D.12: Effective sample size with time for the BPF (top) and β-BPF with $\beta = 0.1$.

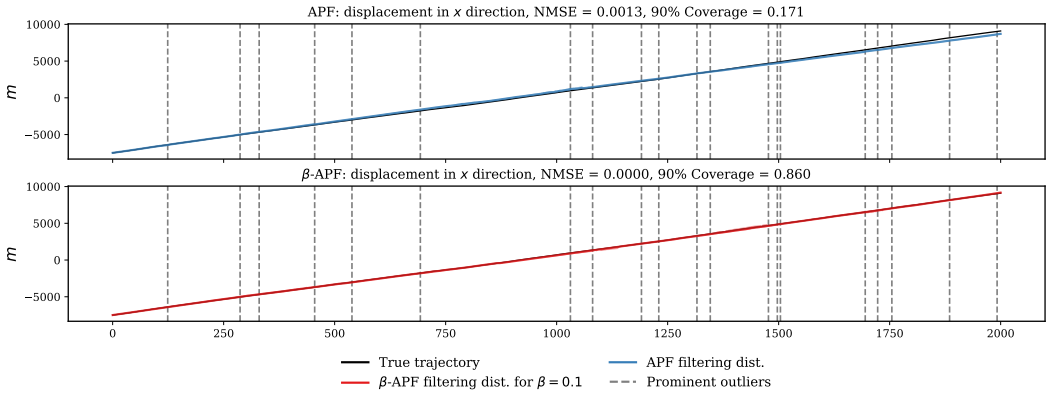


Figure D.13: Marginal filtering distributions for the APF (top) and β-APF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

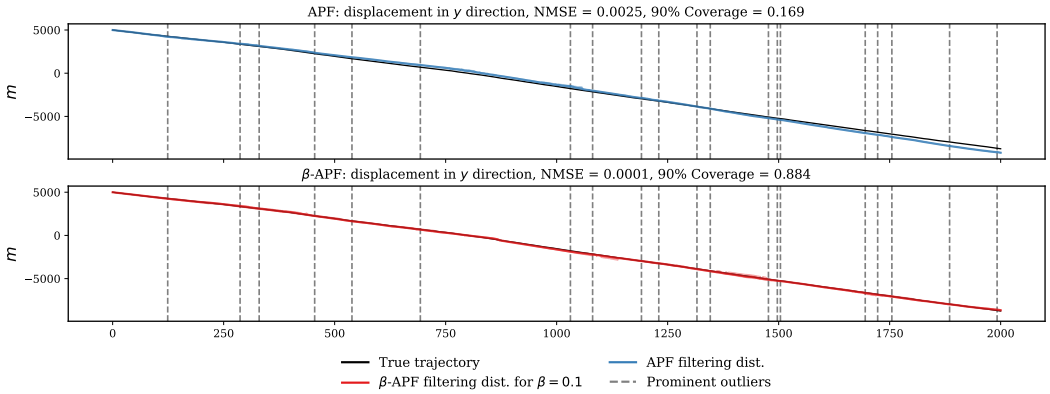


Figure D.14: Marginal filtering distributions for the APF (top) and β-APF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

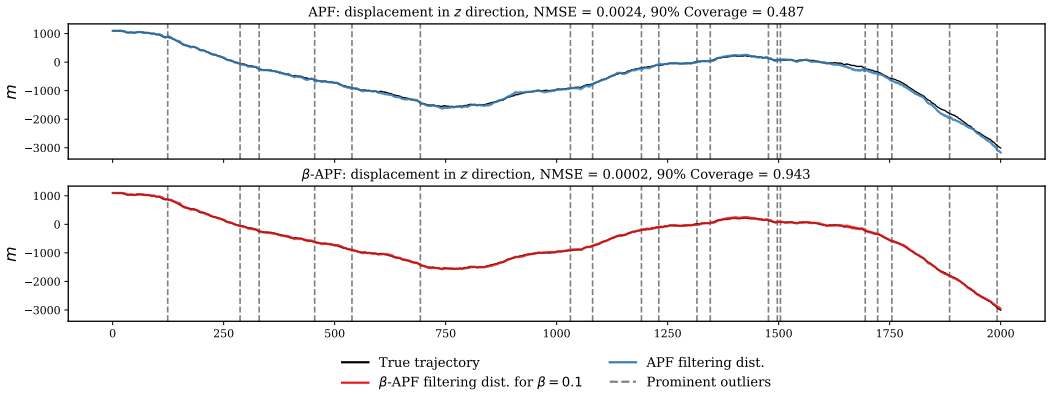


Figure D.15: Marginal filtering distributions for the APF (top) and β-APF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

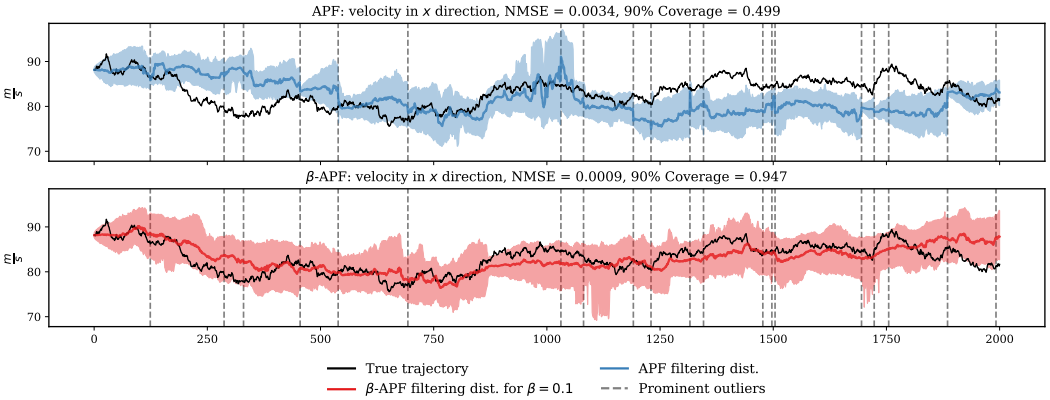


Figure D.16: Marginal filtering distributions for the APF (top) and β-APF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

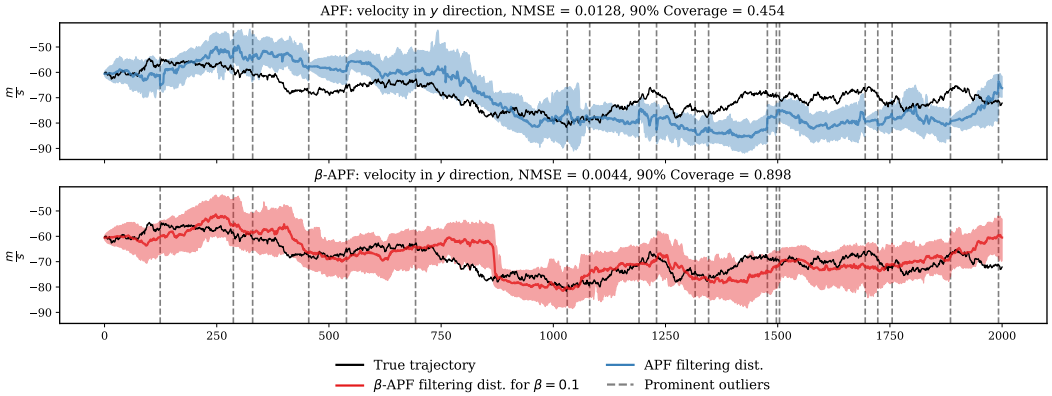


Figure D.17: Marginal filtering distributions for the APF (top) and β-APF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

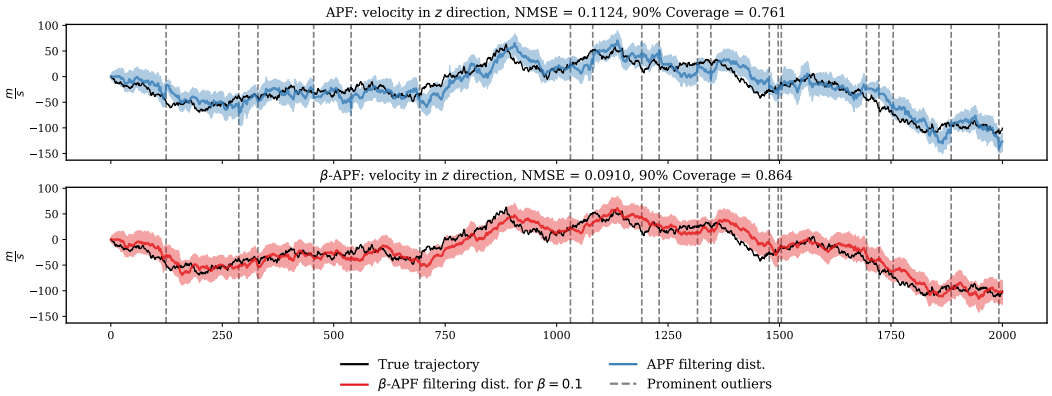


Figure D.18: Marginal filtering distributions for the APF (top) and β-APF (bottom) with $\beta = 0.1$. The locations of the most prominent (largest deviation) outliers are shown as dashed vertical lines in black.

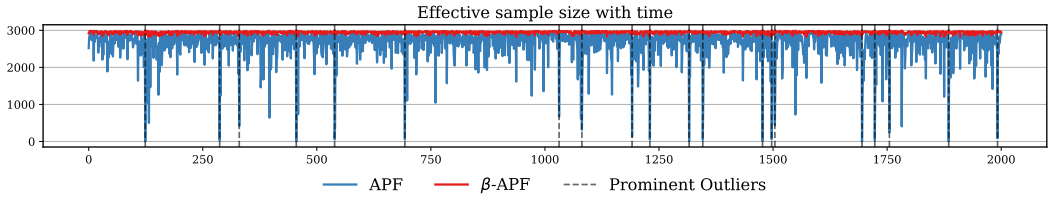


Figure D.19: Effective sample size with time for the APF (top) and β-APF with $\beta = 0.1$.

	p_c							
Filter	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4
BPF	16.63(0.06)	17.67(0.05)	17.88(0.06)	18.66(0.07)	19.68(0.08)	20.12(0.09)	20.96(0.08)	21.55(0.09)
t-BPF	16.33(0.05)	17.15(0.05)	17.15(0.05)	17.92(0.05)	18.72(0.06)	18.95(0.07)	19.71(0.09)	20.11(0.08)
β-BPF = 0.005	16.26(0.05)	17.01(0.05)	16.96(0.06)	17.60(0.05)	18.34(0.07)	18.48(0.06)	19.24(0.06)	19.60(0.07)
β-BPF = 0.01	16.23(0.04)	16.91(0.05)	16.65(0.05)	17.06(0.05)	17.74(0.05)	17.86(0.06)	18.43(0.05)	18.61(0.06)
β-BPF = 0.05	16.39(0.04)	16.97(0.05)	16.70(0.06)	17.23(0.06)	18.03(0.06)	17.84(0.06)	18.45(0.07)	18.78(0.08)
β-BPF = 0.1	17.46(0.05)	17.92(0.06)	17.90(0.11)	18.61(0.12)	19.49(0.11)	19.15(0.10)	19.76(0.10)	20.24(0.11)
β-BPF = 0.2	16.56(0.04)	17.07(0.05)	16.58(0.04)	17.43(0.04)	17.87(0.06)	17.85(0.05)	18.56(0.06)	18.84(0.06)
APF	15.96(0.05)	17.09(0.04)	17.34(0.05)	18.13(0.05)	19.04(0.08)	19.51(0.06)	20.67(0.07)	21.15(0.09)
β-APF = 0.005	15.71(0.04)	16.49(0.05)	16.57(0.05)	17.19(0.04)	17.80(0.05)	18.15(0.04)	18.96(0.07)	19.19(0.06)
β-APF = 0.01	15.69(0.04)	16.31(0.04)	16.31(0.04)	16.85(0.04)	17.47(0.05)	17.66(0.05)	18.46(0.05)	18.66(0.05)
β-APF = 0.05	15.69(0.04)	16.26(0.04)	16.01(0.04)	16.53(0.03)	17.17(0.05)	17.14(0.06)	17.83(0.05)	17.92(0.05)
β-APF = 0.1	15.84(0.04)	16.46(0.05)	16.16(0.04)	16.56(0.04)	17.30(0.05)	17.16(0.04)	17.89(0.05)	18.09(0.05)
β-APF = 0.2	16.90(0.06)	17.35(0.06)	17.32(0.09)	17.68(0.06)	18.78(0.08)	18.40(0.06)	18.87(0.06)	19.28(0.08)

Table D.2: Predictive results on the TAN example. The one step ahead predictive performance is measure by the median absolute error. The figures are averaged across 50 runs and the standard error on the average score is provided.

D.4.3 London air quality experiment

Table D.3: GP regression NMSE (higher is better) and 90% empirical coverage for the credible intervals of the posterior predictive distribution, on 100 runs. The **bold font** indicate the statistically significant best result according to the Wilcoxon signed-rank test. All presented results are statistically different from each other according to the test.

	median (IQR)	
Filter (Smoother)	NMSE	EC
Kalman (RTS)	0.144(0)	0.685(0)
BPF (FFBS)	0.116(0.015)	0.650(0.020)
($\beta = 0.005$)-BPF (FFBS)	0.102(0.014)	0.67(0.025)
($\beta = 0.01$)-BPF (FFBS)	0.077(0.007)	0.705(0.015)
($\beta = 0.05$)-BPF (FFBS)	0.063(0.003)	0.735(0.015)
($\beta = 0.1$)-BPF (FFBS)	0.061(0.003)	0.760(0.015)
($\beta = 0.2$)-BPF (FFBS)	**0.059**(**0.002**)	**0.803**(**0.020**)

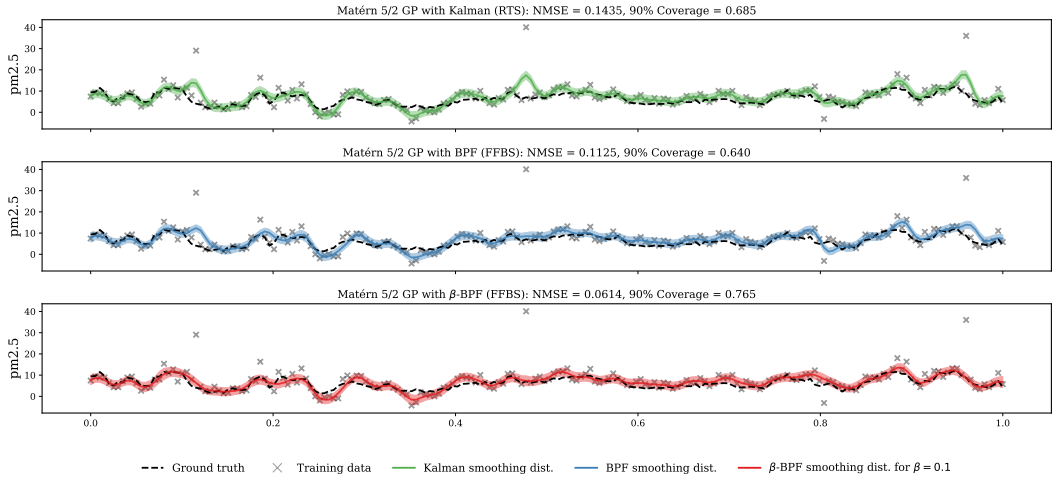


Figure D.20: The GP fit on the measurement time series for one of the London air quality sensors. The top panel shows the posterior from the Kalman (RTS) smoothing. The middle panel shows the posterior from the BPF (FFBS). The bottom panel shows the posterior from the β-BPF (FFBS) for $\beta = 0.1$.

Bibliography

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016.

V. Aglietti, T. Damoulas, and E. V. Bonilla. Efficient inference in multi-task Cox process models. In *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, Okinawa, Japan, 2019.

Ö. D. Akyildiz and J. Míguez. Nudging the particle filter. *Statistics and Computing*, 30(2), 2020.

A. M. Alaa and M. van der Schaar. Deep Multi-task Gaussian Processes for Survival Analysis with Competing Risks. In *Advances in Neural Information Processing Systems (NIPS) 30*, Long Beach, USA, 2017.

M. Alvarez and N. D. Lawrence. Sparse convolved Gaussian Process for multi-output regression. In *Advances in Neural Information Processing Systems (NIPS) 21*, Vancouver, Canada, 2009.

M. Alvarez, L. Rosasco, and N. D. Lawrence. Kernels for Vector-Valued Functions: a Review. Technical Report, MIT, 2011.

B. D. O. Anderson and J. B. Moore. *Optimal filtering*. Englewood Cliffs, N.J. Prentice Hall, 1979.

M. Andrychowicz, M. Denil, S. G. Colmenarejo, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS) 29*, Barcelona, Spain, 2016.

O. Anschel, N. Baram, and N. Shimkin. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2007.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian Multitask Learning. *Journal of Machine Learning Research*, 4, 2003.

M. Bånkestad, J. Sjölund, J. Taghia, and T. B. Schön. The Elliptical Processes: a new family of flexible stochastic processes. *arXiv e-prints*, arxiv abs/2003.07201 [stat.ME], 2020.

T. Bayes. An essay towards solving a problem in the doctrine of chances. by the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical Transactions of the Royal Society of London*, (53), 1763.

J. M. Bernardo and A. F. Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

A. Beskos, O. Papaspiliopoulos, G. O. Roberts, and P. Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes. *Journal of the Royal Statistical Society, Series B*, 68(3), 2006.

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

P. G. Bissiri, C. C. Holmes, and S. G. Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5), 2016.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational Inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518), 2017.

E. V. Bonilla, K. M. Chai, and C. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems (NIPS) 20*, Vancouver, Canada, 2007.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2, 2002.

A. Boustati, T. Damoulas, and R. S. Savage. Non-linear multitask learning with Deep Gaussian processes. *arXiv e-prints*, arXiv:1905.12407 [stat.ML], 2019.

A. Boustati, Ö. D. Akyildiz, T. Damoulas, and A. M. Johansen. Generalised Bayesian filterting via sequential Monte Carlo. In *Advances in Neural Information Processing Systems (NeurIPS) 33*, Virtual, 2020a.

A. Boustati, S. Vakili, J. Hensman, and S. John. Amortized variance reduction for doubly stochastic objective. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, Virtual, 2020b.

A. Boustati, S. John, S. Vakili, and J. Hensman. Computational inference system, 2021. US Patent App. 16/984,824.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

P. Boyle and M. Frean. Dependent Gaussian Processes. In *Advances in Neural Information Processing Systems (NIPS) 17*, Vancouver, Canada, 2004.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, and S. Wanderman-Milne. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

M. Briers, A. Doucet, and S. Maskell. Smoothing algorithms for state space models. *Annals of the Institute of Statistical Mathematics*, 62(1), 2010.

T. D. Bui, D. Hernández-Lobato, J. M. Hernández-Lobato, Y. Li, and R. E. Turner. Deep Gaussian processes for regression using approximate expectation propagation. In *Proceedings of the 33nd International Conference on Machine Learning ICML*, New York, USA, 2016.

C. A. Calder and N. Cressie. Some topics in convolution-based spatial modeling. *Proceedings of the 56th Session of the International Statistics Institute*, 2007.

L. E. Calvet, V. Czellar, and E. Ronchetti. Robust filtering. *Journal of the American Statistical Association*, 110(512), 2015.

M. Capinski and P. E. Kopp. *Measure, integral and probability*. Springer Science & Business Media, 2013.

P. Carbonetto, M. King, and F. Hamze. A stochastic approximation method for inference in probabilistic graphical models. In *Advances in Neural Information Processing Systems (NIPS) 22*, Vancouver, Canada, 2009.

R. Caruana. Multitask Learning. *Machine Learning*, 28(1), 1997.

T. Chen, E. B. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China, 2014.

Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems (NIPS) 22*. Vancouver, Canada, 2009.

N. Chopin. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6), 2004.

A. Cichocki and S. Amari. Families of alpha beta and gamma divergences: Flexible and robust measures of similarities. *Entropy*, 12(6), 2010.

Y. Cong, M. Zhao, K. Bai, and L. Carin. GO gradient for expectation-based objectives. In *7th International Conference on Learning Representations (ICLR)*, New Orleans, USA, 2019.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

J. A. Covington, M. P. v. d. Schee, A. S. L. Edge, B. Boyle, R. S. Savage, and R. P. Arasaradnam. The application of FAIMS gas analysis in medical diagnostics. *Analyst*, 2015.

C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018.

K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random feature expansions for deep gaussian processes. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017.

A. C. Damianou and N. D. Lawrence. Deep Gaussian Processes. In *Proceedings of 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Scottsdale, USA, 2013.

A. C. Damianou, C. H. Ek, M. K. Titsias, and N. D. Lawrence. Manifold Relevance Determination. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, UK, 2012.

F. Dangel, F. Kunstner, and P. Hennig. BackPACK: Packing more into backprop. In *8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

P. Del Moral. *Feynman-Kac formulae: Genealogical and interacting particle systems with applications*. Springer, 2004.

A. B. Dieng, D. Tran, R. Ranganath, J. Paisley, and D. Blei. Variational inference via χ-upper bound minimization. In *Advances in Neural Information Processing Systems (NIPS) 30*, Long Beach, USA, 2017.

J. Domke and D. R. Sheldon. Importance weighting and variational inference. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, Montréal, Canada, 2018.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3), 2000.

D. Duvenaud, O. Rippel, R. P. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the 7th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Reykjavik, Iceland, 2014.

B. Efron and T. Hastie. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press, USA, 1st edition, 2016.

S. Eguchi et al. A differential geometric approach to statistical inference on the basis of contrast functionals. *Hiroshima Mathematical Journal*, 15(2), 1985.

T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, Seattle, USA, 2004.

P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts. Particle filters for partially-observed diffusion. *Journal of the Royal Statistical Society, Series B*, 70(4), 2008.

M. Filippone, M. Zhong, and M. Girolami. A comparative evaluation of stochastic-based inference methods for Gaussian process models. *Machine Learning*, 93(1), 2013.

F. Futami, I. Sato, and M. Sugiyama. Variational inference based on robust divergences. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, Lanzarote, Spain, 2018.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, USA, 2016.

T. Geffner and J. Domke. Using large ensembles of control variates for variational inference. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, Montréal, Canada, 2018.

M. Gerber, N. Chopin, and N. Whiteley. Negative association, ordering and convergence of resampling methods. *Annals of Statistics*, 47(4), 2019.

S. Gershman and N. Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual meeting of the Cognitive Science Society*, volume 36, 2014.

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

Z. Ghahramani and M. J. Beal. Propagation algorithms for variational Bayesian learning. In *Advances in Neural Information Processing Systems (NIPS) 13*, Denver, USA, 2000.

S. Ghosal, J. K. Ghosh, and A. W. van der Vaart. Convergence rates of posterior distributions. *Annals of Statistics*, 28(2), 2000.

A. Ghosh and A. Basu. Robust Bayes estimation using the density power divergence. *Annals of the Institute of Statistical Mathematics*, 68(2), 2016.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010.

P. Goovaerts. *Geostatistics for Natural Resources Evaluation*. Oxford University Press, 1997.

N. J. Gordon, D. J. Salmond, and A. F. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2), 1993.

M. I. Gorinova, D. Moore, and M. D. Hoffman. Automatic Reparameterisation of Probabilistic Programs. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Vienna, Austria, 2020.

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation.

In *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

G. Grimmett and D. Stirzaker. *Probability and random processes*. Oxford university press, 2001.

S. Gu, S. Levine, I. Sutskever, and A. Mnih. MuProp: Unbiased backpropagation for stochastic neural networks. In *4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

P. Guarniero, A. M. Johansen, and A. Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520), 2017.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

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 (NeurIPS) 31*, Montréal, Canada, 2018.

T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*, Bellevue, USA, 2013.

J. Hensman, A. G. d. G. Matthews, and Z. Ghahramani. Scalable variational gaussian process classification. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, San Diego, USA, 2015.

Y. Hernández-Lobato, J. M. amd Li, M. Rowland, D. Hernández-Lobato, T. Bui, and R. E. Turner. Black-box α-divergence minimization. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, New York, USA, 2016.

D. Higdon. A process-convolution approach to modelling temperatures in the north atlantic ocean. *Environmental and Ecological Statistics*, 1998.

D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian statistics*, 1999.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(4), 2013.

X.-L. Hu, T. B. Schon, and L. Ljung. A robust particle filter for state estimation with convergence results. In *46th IEEE Conference on Decision and Control*, New Orleans, USA, 2007. IEEE.

P. J. Huber. *Robust statistics*. John Wiley & Sons, 1981.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. In *5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003.

J. Jewson, J. Q. Smith, and C. Holmes. Principles of Bayesian inference using general divergence criteria. *Entropy*, 20(6), 2018.

A. M. Johansen and A. Doucet. A note on the auxiliary particle filter. *Statistics and Probability Letters*, 78(12), 2008.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS) 26*, Lake Tahoe, USA, 2013.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1), 1960.

M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv e-prints*, arXiv:1805.08845v1 [stat.ML], 2018.

M. Kandemir. Asymmetric Transfer Learning with Deep Gaussian Processes. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015.

J. F. Kenney. *Mathematics of statistics*. D. Van Nostrand, 1939.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, San Diego, California, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, Banff, Canada, 2014.

D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. In *Advances in Neural Information Processing Systems (NIPS) 28*, Montréal, Canada, 2015.

J. Knoblauch, J. E. Jewson, and T. Damoulas. Doubly robust Bayesian inference for non-stationary streaming data with β-divergences. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, Montréal, Canada, 2018.

J. Knoblauch, J. Jewson, and T. Damoulas. Generalized variational inference. *arXiv e-prints*, arXiv:1904.02063 [stat.ML], 2019.

A. N. Kolmogorov. *Foundations of the theory of probability.* Chelsea Publishing Company, 1950.

W. Kool, H. van Hoof, and M. Welling. Buy 4 REINFORCE samples, get a baseline for free! In *ICLR Workshop on Deep Reinforcement Learning Meets Structured Prediction*, 2019.

W. Kool, H. van Hoof, and M. Welling. Estimating gradients for discrete random variables by sampling without replacement. In *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18 (14), 2017.

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

P. S. Laplace. *Essai philosophique sur les probabilités.* 1814.

P. S. Laplace. *Théorie analytique des probabilités.* 1820.

N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6, 2005.

Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553), 2015.

W. Lee, H. Yu, and H. Yang. Reparameterization gradient for non-differentiable models. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, Montréal, Canada, 2018.

Y. Li and R. E. Turner. Rényi divergence variational inference. In *Advances in Neural Information Processing Systems (NIPS) 29*, Barcelona, Spain, 2016.

D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, USA, 2002.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, Montréal, Canada, 2009.

C. S. Maiz, J. Miguez, and P. M. Djuric. Particle filtering in the presence of outliers. In *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, 2009.

C. S. Maiz, E. M. Molanes-Lopez, J. Miguez, and P. M. Djuric. A particle filtering scheme for processing time series corrupted by outliers. *IEEE Transactions on Signal Processing*, 60(9), 2012.

A. S. Martinez-Vernon, J. A. Covington, R. P. Arasaradnam, S. Esfahani, N. O' Connell, I. Kyrou, and R. S. Savage. An improved machine learning pipeline for urinary volatiles disease detection: Diagnosing diabetes. *PLOS ONE*, 2018.

A. G. d. G. Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

A. G. d. G. Matthews, J. Hensman, R. E. Turner, and Z. Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statisticsc (AISTATS)*, Cadiz, Spain, 2016.

A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18, 2017.

J. Míguez, D. Crisan, and P. M. Djurić. On the convergence of two sequential Monte Carlo methods for maximum a posteriori sequence estimation and stochastic global optimization. *Statistics and Computing*, 23(1), 2013.

A. Miller, N. Foti, A. D' Amour, and R. P. Adams. Reducing reparameterization gradient variance. In *Advances in Neural Information Processing Systems (NIPS) 30*, Long Beach, USA, 2017.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Bejing, China, 2014.

A. Mnih and D. J. Rezende. Variational inference for Monte Carlo objectives. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, New York, USA, 2016.

A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NIPS) 20*, Vancouver, Canada, 2008.

S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21, 2020.

T. Müller, F. Rousselle, A. Keller, and J. Novák. Neural control variates. *ACM Transactions on Graphics*, 39(6), 2020.

C. Naesseth, F. J. R. Ruiz, S. Linderman, and D. M. Blei. Reparameterization gradients through acceptance-rejection methods. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, USA, 2017.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Elements of Sequential Monte Carlo. *Foundations and Trends in Machine Learning*, 12(3), 2019.

C. A. Naesseth, F. Lindsten, and D. M. Blei. Markovian score climbing: Variational inference with KL($p||q$). *ArXiv e-prints*, arXiv:2003.10374 [stat.ML], 2020.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017.

T. Nguyen and E. Bonilla. Efficient Variational Inference for Gaussian Process Regression Networks. In *Proceedings of 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Scottsdale, USA, 2013.

T. V. Nguyen and E. V. Bonilla. Collaborative Multi-output Gaussian Processes. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI)*, Quebec City, Canada, 2014.

E. Nikishin, P. Izmailov, B. Athiwaratkun, D. Podoprikhin, T. Garipov, P. Shvechikov, D. Vetrov, and A. G. Wilson. Improving stability in deep reinforcement learning with weight averaging. In *Uncertainty in Artificial Intelligence Workshop on Uncertainty in Deep Learning*, Monterey, USA, 2018.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer series in operations research. Springer, New York, NY, 2000.

N. Nüsken and L. Richter. Solving high-dimensional Hamilton-Jacobi-Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *arXiv e-prints*, arXiv:2005.05409 [math.OC], 2020.

A. B. Owen. *Monte Carlo theory, methods and examples.* 2013.

J. Paisley, D. Blei, and M. Jordan. Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, UK, 2012.

J. W. T. Peters and M. Welling. Probabilistic binary neural networks. *arXiv e-print*, 2018.

M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446), 1999.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65), 2005.

M. Quiroz, R. Kohn, M. Villani, and M.-N. Tran. Speeding up mcmc by efficient data subsampling. *Journal of the American Statistical Association*, 114(526), 2019.

R. Ranganath, S. Gerrish, and D. Blei. Black Box Variational Inference. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Reykjavic, Iceland, 2014.

R. Ranganath, J. Altosaar, D. Tran, and D. M. Blei. Operator variational inference. In *Advances in Neural Information Processing Systems (NIPS) 30*, Barcelona, Spain, 2016a.

R. Ranganath, D. Tran, and D. M. Blei. Hierarchical variational models. In *Proceedings of the 33rd International Conference on Machine Learning, (ICML)*, New York, USA, 2016b.

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

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *American Institute of Aeronautics and Astronautics Journal*, 3 (8), 1965.

R. Reiss. *Approximate distributions of order statistics: with applications to nonparametric statistics*. Springer Science & Business Media, 2012.

J. Requeima, W. Tebbutt, W. Bruinsma, and R. E. Turner. The Gaussian process autoregressive regression model (GPAR). In *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, Okinawa, Japan, 2019.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China, 2014.

L. Richter, A. Boustati, N. Nüsken, F. J. R. Ruiz, and Ö. D. Akyildiz. Vargrad: A low variance gradient estimator for variational inference. In *Advances in Neural Information Processing Systems (NeurIPS) 33*, Virtual, 2020.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

G. Roeder, Y. Wu, and D. Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems (NIPS) 30*, Long Beach, USA, 2017.

F. J. R. Ruiz and M. K. Titsias. A contrastive divergence for combining variational inference and MCMC. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, USA, 2019.

F. J. R. Ruiz, M. K. Titsias, and D. M. Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems (NIPS) 29*, Barcelona, Spain, 2016.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine learning (ICML)*, Helsinki, Finland, 2008.

T. Salimans and D. A. Knowles. On using control variates with stochastic approximation for variational bayes and its connection to stochastic linear regression. *arXiv e-print*, arXiv:1401.1022 [stat.CO], 2014.

H. Salimbeni and M. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS) 30*, Long Beach, USA, 2017.

S. Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

S. Särkkä and A. Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

S. Särkkä, A. Solin, and J. Hartikainen. Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering. *IEEE Signal Processing Magazine*, 30(4), 2013.

A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t processes as alternatives to gaussian processes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Reykjavik, Iceland, 2014.

O. Shayer, D. Levi, and E. Fetaya. Learning discrete weights using the local reparameterization trick. In *6th International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

S. Si, C. J. Oates, A. B. Duncan, L. Carin, and F.-X. Briol. Scalable control variates for Monte Carlo methods via stochastic optimization. *arXiv e-prints*, arXiv:2006.07487 [stat.ML], 2020.

G. Skolidis and G. Sanguinetti. Bayesian Multitask Classification With Gaussian Process Priors. *IEEE Transactions on Neural Networks*, 22(12), 2011.

E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems (NIPS) 18*, 2005.

Y. W. Teh, M. Seeger, and M. Jordan, I. Semiparametric Latent Factor Models. In *Proceedings of 10th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Barbados, 2005.

F. C. Teixeira, J. Quintas, P. Maurya, and A. Pascoal. Robust particle filter formulations with application to terrain-aided navigation. *International Journal of Adaptive Control and Signal Processing*, 31(4), 2017.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3), 1999.

M. K. Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Clearwater Beach, USA, 2009.

M. K. Titsias and M. Lazaro-Gredilla. Spike and slab variational inference for multitask and multiple kernel learning. In *Advances in Neural Information Processing Systems (NIPS) 25*, Granada, Spain, 2011.

M. K. Titsias and M. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, Biejing, China, 2014.

D. Tran, A. Kucukelbir, A. B. Dieng, M. R. Rudolph, D. Liang, and D. M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv e-prints*, arXiv:1610.09787 [stat.CO], 2016.

D. Tran, M. Dusenberry, M. van der Wilk, and D. Hafner. Bayesian layers: A module for neural network uncertainty. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, Vancouver, Canada, 2019.

G. Tucker, A. Mnih, C. J. Maddison, and J. Sohl-Dickstein. REBAR: low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems (NIPS) 30*, Long Beach, USA, 2017.

J. W. Tukey. The future of data analysis. *The annals of mathematical statistics*, 33 (1), 1962.

R. Turner, M. Deisenroth, and C. Rasmussen. State-space inference and learning with gaussian processes. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010.

I. Ustyuzhaninov, I. Kazlauskaite, M. Kaiser, E. Bodin, N. D. F. Campbell, and C. H. Ek. Compositional uncertainty in deep gaussian processes. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, Virtual, 2020.

M. van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv e-prints*, arXiv:2003.01115 [stat.ML], 2020.

S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. *Autonomous Robots*, 2002.

J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

D. Wang, H. Liu, and Q. Liu. Variational inference with tail-adaptive f-divergence. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, Montréal, Canada, 2018.

K. Wang, O. Hamelijnck, T. Damoulas, and M. Steel. Non-stationary non-separable random fields. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Vienna, Austria, 2020.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Belleview, USA, 2011.

F. Wenzel, K. Roth, B. S. Veeling, J. Swiatkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, Virtual, 2020.

C. Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems (NIPS) 9*, Denver, USA, 1997.

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

A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Atlanta, USA, 2013.

A. G. Wilson, D. A. Knowles, and Z. Ghahramani. Gaussian Process Regresssion Networks. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, Edinburgh, UK, 2012.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Cadiz, Spain, 2016.

D. Xu, C. Shen, and F. Shen. A robust particle filtering algorithm with non-Gaussian measurement noise using student-t distribution. *IEEE Signal Processing Letters*, 21(1), 2013.

M. Yin and M. Zhou. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *7th International Conference on Learning Representations (ICLR)*, New Orleans, USA, 2019.

M. Yin, Y. Yue, and M. Zhou. ARSM: Augment-REINFORCE-swap-merge estimator for gradient backpropagation through categorical variables. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, USA, 2019.

A. Zellner. Optimal information processing and Bayes's theorem. *The American Statistician*, 42(4), 1988.