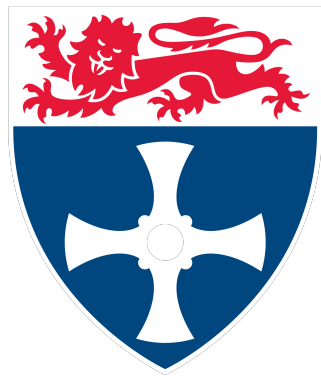


Variational Inference for Stochastic Processes



Tom Ryder

School of Maths, Statistics and Physics
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

April 2021

Acknowledgements

I would like to begin by thanking Dennis Prangle and Andy Golightly for supervising me during my PhD. I have been tremendously lucky to have had two knowledgeable, kind and – above all else – thoroughly likeable supervisors. They have made my PhD a joy to complete, and I thank them unreservedly.

I would additionally like to thank my undergraduate supervisor, James McLaughlin. Without the time he afforded me, the kindness he shows to his students, I would not likely have pursued a research career.

I am additionally grateful to the Centre for Doctoral Training in Cloud Computing for Big Data, the Engineering and Physical Sciences Research Council and Newcastle university, whose generous funding and support has presented me with enormous opportunity. I extend a special thanks to Jen Wood, who so excellently facilitated my research activities.

Abstract

Stochastic process models such as stochastic differential equations (SDEs), state-space models (SSMs), Gaussian processes (GPs) and latent force models (LFMs), provide a powerful collection of modelling techniques to better our understanding of many physical systems. In treating these models within the Bayesian paradigm, we further yield a rich expression of our uncertainty, and gain the ability to incorporate our prior beliefs. However, performing Bayesian posterior inference is not without significant challenge. Exact likelihood calculations can often be intractable, take an infeasibly long time to compute, or be challenging to approximate in the presence of missing data. Therefore, designing new approaches to perform Bayesian inference for this family of stochastic process models is of great scientific interest.

Variational inference (VI) has had great success in scaling Bayesian inference across a range of problem domains. Historically, however, its successful application to stochastic process models has been limited. The reason is two-fold. Firstly, mini-batch likelihood estimation techniques often employed by VI have only previously been applicable to models of independent data. Secondly, approximating distributions have often imposed unrealistic assumptions over the posterior. Fortunately, however, recent advances in generative modelling have provided the framework with which to solve these problems. Here, artificial neural networks can be used to flexibly construct powerful density approximations, which are then amenable to fast computation using modern GPUs. This is otherwise known as black-box-variational inference.

This thesis presents a collection of black-box variational methods for the purposes of approximate inference in SDEs, SSMs, GPs and LFMs. Here we leverage artificial neural networks to parametrise our approximate posterior distributions, permitting accurate inference in a short time. We begin by presenting two methods for SDE inference. The first, inspired by the Euler-Maruyama discretisation, approximates the discrete-time solution to a conditioned diffusion process using recurrent neural networks. The second, which extends the first, eschews a discretisation scheme and approximates the continuous-time process directly. Finally we consider the use of normalising flows for inference using SSMs (including discrete-time SDEs), GPs and LFMs. Here we design a generative architecture that permits mini-batch optimization, allowing approximate inference for big data.

Contents

1	Introduction	1
1.1	Publications	3
1.2	Thesis structure	4
2	Data Modelling	7
2.1	Objectives in probabilistic modelling	7
2.2	Bayesian inference	10
2.3	Monte Carlo methods	12
2.3.1	Importance sampling	13
2.3.2	MCMC	14
2.4	Variational inference	20
2.4.1	Stochastic gradient variational inference	22
3	Stochastic Processes	27
3.1	Introduction to stochastic processes	27
3.1.1	Markov processes	28
3.2	State-space models	29
3.3	Diffusion processes	31
3.3.1	Numerical solutions	32
3.3.2	The conditioned process	34
3.4	Gaussian process regression	35

3.4.1	Closed-form solutions	37
3.4.2	Gaussian process state-space models	40
4	Variational inference for discretised SDEs	43
4.1	Inference for SDEs	43
4.1.1	Bridge constructs	45
4.1.2	Feed-forward neural networks	46
4.2	Variational inference for SDEs	47
4.2.1	Related work	47
4.2.2	Approximate posterior density	48
4.2.3	Algorithm and implementation details	50
4.3	Experiments	53
4.3.1	Lotka-Volterra	53
4.3.2	Epidemic model	57
4.4	Conclusion	61
5	Continuous-time methods	63
5.1	Limits of SDE inference in discrete time	63
5.2	Continuous-time variational objective function	65
5.2.1	Continuous-time Kullback–Leibler divergence	66
5.2.2	Continuous-time ELBO	67
5.2.3	Choice of variational drift function	68
5.3	Gradient estimation of an SDE system	69
5.3.1	Strategies	69
5.3.2	Numerical solver	71
5.4	Algorithm and implementation details	74
5.4.1	Approximate parameter posterior	74
5.4.2	Variable time grid	74

5.4.3	Ensuring positivity	74
5.4.4	Algorithm	75
5.4.5	Optimisation and implementation details	75
5.5	Examples	76
5.5.1	Ornstein-Uhlenbeck process	76
5.5.2	Stochastic Lorenz	77
5.5.3	Lorenz 96	79
5.5.4	Epidemic model	81
5.6	Conclusion	85
6	Scalable State-Space Inference with Normalising Flows	87
6.1	Inference for SSMs	88
6.1.1	Related work	89
6.2	Normalising flows	90
6.2.1	Inverse autoregressive flows	91
6.3	Local inverse autoregressive flows	92
6.3.1	Multivariate case	94
6.3.2	Sampling	95
6.4	Variational inference for SSMs	96
6.4.1	Optimisation algorithm	98
6.4.2	Side information	98
6.5	Implementation details	99
6.6	Experiments	100
6.6.1	AR(1) Model	100
6.6.2	Lotka-Volterra	102
6.6.3	FitzHugh-Nagumo	103
6.6.4	Log-Gaussian stochastic volatility	104

6.7	Application to GP regression and latent force models	105
6.7.1	Latent force models	106
6.7.2	Scalable inference for non-linear latent force models	108
6.7.3	Experiments	109
6.8	Conclusion	114
7	Conclusions	117
7.1	Outlook	118
	Bibliography	121
	Appendix A Stochastic processes	131
A.1	The Itô integral	131
A.2	Itô's Formula	132
A.2.1	Example: Ornstein-Uhlenbeck process	133
	Appendix B Variational inference for discretised SDEs	135
B.1	Derivation of a Brownian bridge	135
	Appendix C Continuous-time methods	139
C.1	Ornstein-Uhlenbeck SDE with end-point conditioning	139
	Appendix D Scalable State-Space Inference with Normalizing Flows	141
D.1	MCMC algorithm for AR(1) via forward-filter recursion	141
D.2	Experimental Details	142
D.3	Lotka-Volterra details	144
D.3.1	Restricting x to positive values	144
D.3.2	Multiple modes and pre-training	145
D.4	Likelihood approximation of LFMs	146
D.4.1	Discrete-time predictions	148

Chapter 1

Introduction

Time-series data is a fundamental resource in the statistical analysis of information systems. It is found ubiquitously across science and industry: from engineering [43] to epidemiology [46, 47], econometrics [22, 45] and biology [149, 54]. Developing methods to understand these data and the systems they represent is therefore an important area of active research in probabilistic machine learning and statistics. The goal of this thesis is to propose a collection of techniques to better understand these systems, combining advances in Bayesian statistics and neural networks.

Here we view time-series analysis in three constituent parts: *model selection*, *inference* and *decision*. A probabilistic model defines a statistical relationship between the observed data, *model parameters* and a latent process of *paths* assumed to generate the data; inference infers values of the latent parameters and paths most consistent with the data; inferred values are used to make decisions about the system. This thesis does not concern itself with challenges of either model selection or decision – but concentrates on the development of approaches for inference. To that end, the goal of this thesis is to present new methods permitting inference for models of time-series data within an *approximate Bayesian* framework. That is, we combine our model with a-priori information in the form of a *prior distribution*, and seek to learn about the system by approximating the distribution of latent paths and model parameters, conditioned on the data. This is otherwise known as the *posterior* distribution, for which we perform *posterior inference*.

In this thesis, we shall focus on three models of time-series data: *stochastic differential equations* (SDEs), *state-space models* (SSMs) and *Gaussian process* (GP) regression. We additionally consider *latent-force models* (LFMs), but view this as a subsidiary of GP regression. More broadly, all considered approaches are examples of *stochastic process* models. Whilst each model presents a unique probabilistic perspective, it is possible to consider them in a unifying framework: many GPs can be represented as SDEs, and SDEs can be

(approximately) represented by SSMs. Likewise, each model encounters similar challenges when attempting posterior inference. Current approaches are limited when faced with one (or a combination) of: big data [139, 127], missing data [57, 116], non-linear dynamics [42, 65] and non-Gaussian observation likelihoods [96, 64]. In that sense, advancing state-of-the-art solutions for the above-mentioned challenges can yield benefits when using any of these models to analyse a system. These challenges, with respect to each of considered models, are discussed in more detail as follows.

In the context of Bayesian inference for SDEs and SSMs, approaches typically utilise Monte-Carlo-based sampling techniques, such as Kalman and particle filters, along with Markov chain Monte Carlo (MCMC) (e.g. Särkkä [126], Doucet et al. [36], Cappé et al. [27], Andrieu et al. [8]). These techniques, however, involve evaluating or estimating the data probability under various choices of model parameters. Unfortunately, even the fastest schemes require one pass of linear-time complexity through the data, and can fail with collapse to a degenerate distribution [139]. Applying these methods to big data can therefore be impractically expensive.

In addition to challenges with big data, inference for SDEs and SSMs is challenged in the instance of missing data. That is, when observations of the latent paths are not available at all times, inference schemes must overcome the subsequent dependence between parameters and missing data. Monte Carlo strategies then typically require an approximation to the latent path conditional on model parameters and observations. The goal is to produce a path ‘bridging’ between a pair of observation times. These approximations are otherwise known as *bridge constructs* (e.g. Roberts and Stramer [116], Golightly and Wilkinson [57], Fuchs [47], van der Meulen et al. [137]). Designing a bridge construct with desirable features for a particular problem is, however, a challenging and time-consuming tuning task, where misspecification can yield impractically inefficient algorithms [116].

Unlike most SDEs and SSMs, a GP regression often possesses a closed-form solution. Access to this solution, however, is conditional on two criteria: the observation model must be Gaussian, and the data cannot be ‘big’. In the instance of non-Gaussian observations, the posterior is analytically intractable, with solutions often employing approximations to the posterior [96, 67]. Supposing a Gaussian observation likelihood, the closed-form solution (whilst analytically tractable) often takes an infeasibly long time to compute. Here we are required to invert the GP’s covariance matrix – an operation that scales cubically with respect to the data size. Solutions to this problem manifest in two research threads: *sparse Gaussian processes* [131, 26, 133, 68] or *Gaussian process state-space models* [127] (GPSSMs). The latter permits solution with Bayesian SSM techniques, which additionally allows solutions unconditional on the distribution of observations

This thesis approaches challenges of posterior inference for the stochastic process models described above using the tools of *variational inference*. Here we utilise advances in *likelihood-based generative modelling* to approximate complex posterior densities. These techniques (usually parametrised by neural networks) are designed to closely approximate arbitrarily complex probability density functions. Variational inference then seeks to minimise a divergence measure between the approximating and posterior density using the tools of stochastic optimisation. Advances in generative modelling [102, 79] and stochastic optimization [80, 112, 134] mean variational inference is fast, flexible and able to automate the selection of good approximations. Hence, we are able to successfully apply it to perform inference in the presence of challenges previously discussed.

1.1 Publications

The main contributions of this thesis are to devise variational inference methods for the stochastic process models described above. This thesis contains three full-length conference papers: two published, and one presently under review. The papers and their contributions are detailed as follows:

- (i) *Black-box variational inference for stochastic differential equations* (International Conference on Machine Learning, 2018) is a variational inference method for SDEs approximated in discrete time. This paper address the challenge of posterior inference in the presence of missing data and non-linear dynamics, effectively automating the often challenging task of bridge construct design. At the time of its publication, this paper permitted state-of-the-art approximate inference for an otherwise intractable problem class. We illustrate our approach on complex models from biology [58] and epidemiology [7], producing accurate parameter estimation in only a few hours.
- (ii) *Scalable approximate inference for state-space models with normalising flows* (Submitted) is a method for the variational approximation of SSMs. It proposes a new type of generative model, a local *inverse autoregressive flow*, which allows a variational inference method using only short portions of the data at each optimisation step. This approach principally solves the problem of posterior inference with big data, providing a method with $\mathcal{O}(1)$ computational complexity with respect to the data size. It additionally permits inference in the instance of missing data and non-linear dynamics. We illustrate our method on models from neuroscience [73, 137] and quantitative finance [56] – including an example with 1,000,000 data points – and achieve accurate parameter estimation in a short time.

- (iii) *Black-box inference for non-linear latent force models* (International Conference on Artificial Intelligence and Statistics, 2020) extends the approach of (ii) to GP regression and LFMs. It achieves this by formulating the problem as a SDE which is then discretised. This approach allows inference for GPs with non-Gaussian observation likelihoods and LFMs with non-linear dynamics – problems for which there is no closed-form solution. We illustrate this approach successfully on GP regressions of both Gaussian and non-Gaussian observation likelihoods (such as count data), before analysing multi-dimensional gene transcription data [14].

We additionally note that our most recent research (Chapter 5) has not yet been formulated into a paper (published or otherwise). This work nonetheless represents a significant contribution and can be viewed as an improved extension of paper (i). An outline of this chapter’s contribution – a new approach for continuous-time SDE inference – can be found in Section 1.2.

1.2 Thesis structure

This thesis can be viewed in three parts. Part I (Chapters 2 and 3) reviews foundational material on statistical inference, along with defining problem formulations and our choice of notation. In Part II (Chapters 4 and 5) we present new methods for performing approximate Bayesian inference with stochastic differential equations, using generative models parametrised by recurrent neural networks. Here we begin by operating in a discrete-time setting, before extending our techniques to continuous-time. In Part III (Chapter 6) we present an approach to perform inference for state-space models using mini-batch gradient descent, before considering its extension to Gaussian process regression and latent-force models. The contribution – and intended purpose – of each chapter is further detailed below:

Chapter 2 In this chapter we introduce the formative concepts of statistical inference using models of independent data. We then briefly review tools of computational Bayesian inference: Monte-Carlo simulation, importance sampling and the Metropolis-Hastings algorithm. We finish this Chapter by discussing variational inference, and reviewing its recent advances.

Chapter 3 Here we extend our review of independent models to stochastic processes and Markov chains. We define the likelihoods of SSMs, SDEs and GPs, along with the resulting posterior inference formulations.

Chapter 4 This chapter contains the work originally presented in *black-box variational inference for stochastic differential equations*.

Chapter 5 We extend the work of chapter 5 and consider approximate inference in continuous-time. We achieve this by evaluating our variational objective using a *path integral* [10] (i.e. an expectation with respect to the solution path of a stochastic differential equation). This represents our most recent research work, and is currently unpublished. We successfully employ this method on models of high-dimensional chaotic systems [89, 90] and infectious epidemiology [39, 44, 13] with challenging observations regimes.

Chapter 6 In this chapter we present the works of *scalable approximate inference for state-space models with normalising flows* and *black-box inference for non-linear latent force models*.

Chapter 2

Data Modelling

2.1 Objectives in probabilistic modelling

Probabilistic modelling seeks to relate observable data quantities $\mathcal{D}$ to some latent variables θ in an attempt to learn more about the data and/or the system it represents. In that sense, probabilistic modelling can be thought of as *information augmentation*: expanding our understanding of data beyond what is immediately observed. More formally, we express this with a model $p(\mathcal{D}|\theta)$, which encodes this relationship as a conditional probability density. This density is otherwise known as our *likelihood function*. Modelling then proceeds as follows:

Model selection We begin by selecting a model (or likelihood) for our data. This is sometimes obvious (e.g. using a binomial distribution for the number of heads tossed in n coin flips) or hard (e.g. modelling the result of an election). It is not always clear that a given modelling choice is correct, with model validation and improvement a separate, iterative step after inference. This process in-turn creates a ‘modelling loop’ [50].

Inference Given a model, we now wish to learn a value (or a distribution) for our latent variables that are most consistent with the observed data (e.g. in modelling n coin flips, what is the probability of tossing a head). This chapter will principally focus on the different ways in which we can perform inference given a model likelihood.

Decision and prediction After inferring values for our latent variables, modelling can now be used to make decisions. This can either be done through prediction (e.g. who does our model think is going to win the election), or direct interpretation of the latent variables (e.g. is the coin biased?).

Example: linear regression

Begin by assuming access to data $\mathcal{D} = \{y_i | i = 1, 2, \dots, n\}$, where y_i are scalars. An example dataset is shown in Figure 2.1. Now suppose we seek to relate an input variable x_i to each y_i with the following model

$$y_i = \theta_1 x_i + \theta_2 + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2), \quad (2.1)$$

where each ϵ_i is independently and identically distributed (i.i.d.), $\theta = (\theta_1, \theta_2)^T$ and $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2 . Here we assume σ to be known ($\sigma = 3$), but it is plausibly an additional object of inference.

We can alternatively express (2.1) as

$$y = X^T \theta + \epsilon, \quad (2.2)$$

where $y = (y_1, y_2, \dots, y_n)^T$, $\epsilon = (\epsilon_1, \epsilon_2, \dots, \epsilon_n)^T$ and X is a $2 \times n$ *design matrix* such that

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ 1 & 1 & \dots & 1 \end{bmatrix}. \quad (2.3)$$

It then follows that the likelihood function is given by

$$\begin{aligned} p(y|X, \theta) &= \prod_{i=1}^n p(y_i | x_i, \theta), \\ &= \prod_{i=1}^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta_1 x_i - \theta_2)^2}{2\sigma^2}\right), \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2} \|y - X^T \theta\|^2\right), \\ &= \mathcal{N}(y; X^T \theta, \sigma^2 I_n), \end{aligned} \quad (2.4)$$

where I_n is the $n \times n$ identity matrix, $\|\cdot\|$ is the Euclidean norm and $\mathcal{N}(y; \mu, \Sigma)$ is the multivariate normal density evaluated at y with mean μ and variance *matrix* Σ .

Maximum likelihood estimation

There is a broad tool-kit of theoretical and practical means by which we can perform inference given the likelihood in (2.4), with many of them discussed below. In the first instance, we shall consider inference using the likelihood function *only*. For this purpose we will use the method of maximum likelihood estimation (MLE). That is, we aim to find the value of $\theta \in \mathbb{R}^2$

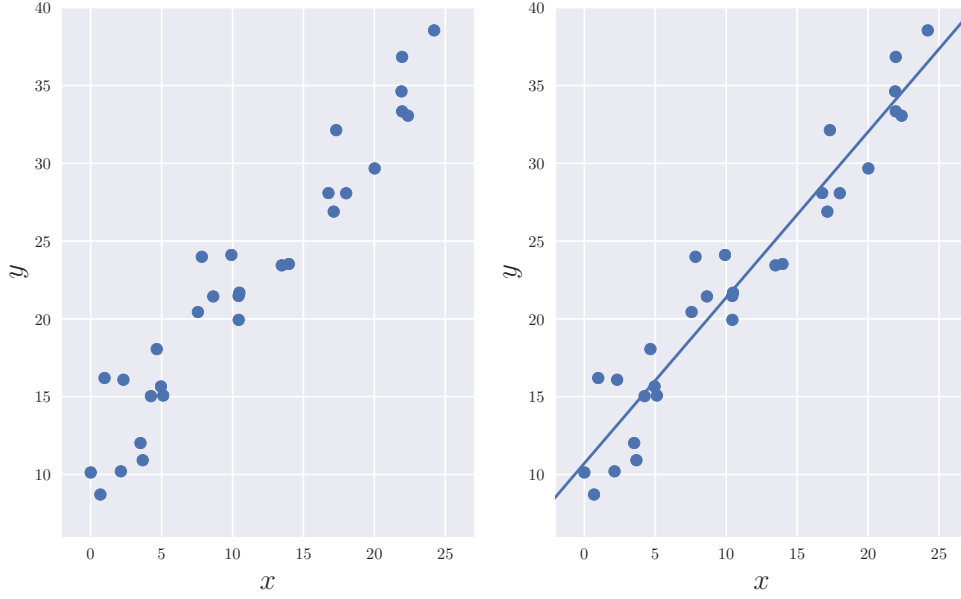


Figure 2.1 Left: Simulated data $\mathcal{D}$ used in the regression examples. Right: linear regression fitted to $\mathcal{D}$ using MLE.

satisfying

$$\hat{\theta} = \operatorname{argmax}_{\theta} \log p(y; X, \theta), \quad (2.5)$$

which, for (2.1), becomes

$$\hat{\theta} = \operatorname{argmax}_{\theta} \log \mathcal{N}(y; X^T \theta, \sigma^2 I). \quad (2.6)$$

In this case, the solution to (2.6) is available in a closed-form. We further note (2.7) and (2.8) are identical to estimators derived from ordinary least squares. See e.g. Casella and Berger [28] for a derivation. The resulting estimators are given by

$$\hat{\theta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad (2.7)$$

$$\hat{\theta}_2 = \bar{y} - \hat{\theta}_1 \bar{x}, \quad (2.8)$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ and $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$. For our data, we find $\hat{\theta} = (1.065, 10.719)^T$. A plot of the resulting best-fit using MLE estimates $\hat{\theta}$ is available in Figure 2.1.

2.2 Bayesian inference

Bayesian statistics extends likelihood-based inference by expressing $\mathcal{D}$ and θ through the joint probability density of the form

$$p(\mathcal{D}, \theta) = p(\mathcal{D}|\theta)p(\theta), \quad (2.9)$$

where $p(\theta)$ is the known as the *prior* over latent variables (see e.g. Gelman et al. [50], McElreath [92], or Albert [3] for a more detailed introduction). In this way, Bayesian inference permits the expression of uncertainty using a probabilistic measure of prior belief, treating θ as a random variable [50]. The target of inference, known as the *posterior distribution*, is then given by conditioning (2.9) on $\mathcal{D}$. This result known as *Bayes Theorem*

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}, \quad (2.10)$$

where $p(\mathcal{D})$ is the *normalizing constant* (or *model evidence*) and is given by

$$p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta)d\theta. \quad (2.11)$$

In practice (2.11) is often intractable (or takes an infeasibly long time to estimate well), with access only to the un-normalized posterior distribution

$$p(\theta, \mathcal{D}) = p(\mathcal{D}|\theta)p(\theta) \propto p(\theta|\mathcal{D}). \quad (2.12)$$

This challenge necessitates the use of approaches detailed in Sections 2.3 and 2.4.

Example: Bayesian linear regression

Again consider the simple linear regression example. We now seek to perform inference in the Bayesian sense (as per (2.10)). Whilst the posterior distribution is analytically intractable in most complex models, it is possible derive a closed-form solution in some simple examples (such as this one).

We begin by ascribing a prior to θ such that

$$p(\theta) = \mathcal{N}(\theta; 0, \Sigma), \quad (2.13)$$

where Σ is a 2×2 covariance matrix. Considering only the terms from the likelihood and the prior that depend on θ gives

$$p(\theta|X, y) \propto \exp\left(-\frac{1}{2\sigma^2} (y - X^T\theta)^T (y - X^T\theta)\right) \exp\left(-\frac{1}{2}\theta^T \Sigma^{-1}\theta\right),$$

where “completing the square” (see e.g. Gelman et al. [50]) gives

$$p(\theta|X, y) \propto \exp\left(-\frac{1}{2} \left(\theta - \sigma^{-2} (\sigma^{-2} X X^T + \Sigma^{-1})^{-1} X y\right)^T (\sigma^{-2} X X^T + \Sigma^{-1}) \left(\theta - \sigma^{-2} (\sigma^{-2} X X^T + \Sigma^{-1})^{-1} X y\right)\right). \quad (2.14)$$

Where we can recognise the form of (2.14) as the Gaussian probability density function (pdf), such that

$$p(\theta|\mathcal{D}) \sim \mathcal{N}\left(\sigma^{-2} (\sigma^{-2} X X^T + \Sigma^{-1})^{-1} X y, (\sigma^{-2} X X^T + \Sigma^{-1})^{-1}\right). \quad (2.15)$$

Letting $\Sigma = 10 \times I_2$ results in the posterior distributions presented in Figure 2.2, where the

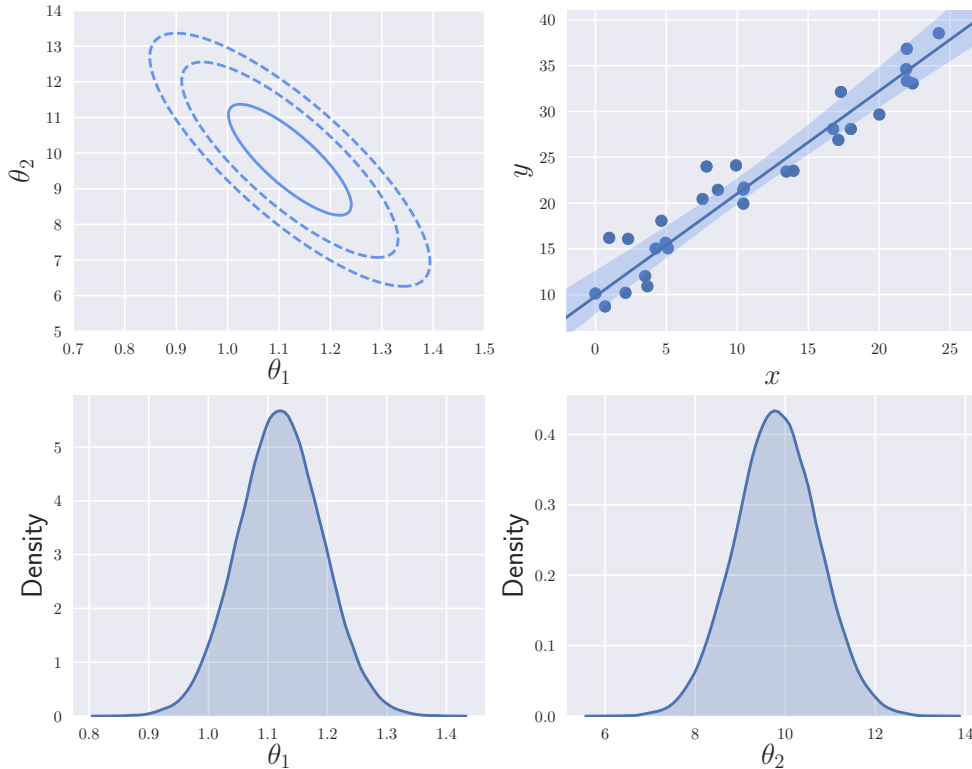


Figure 2.2 Bayesian linear regression with a closed-form solution. Top: contour plots of the joint θ density (left); 95% credible intervals for posterior regression model (right). Bottom: marginal posterior densities for θ_1 (left) and θ_2 (right).

95% credible intervals are available in a closed-form (e.g. Gelman et al. [50]).

2.3 Monte Carlo methods

In the context of posterior inference, given a target density $p(\theta|\mathcal{D})$, Monte Carlo methods provide a broad set of techniques that seek to:

1. *Sample* a collection of i.i.d. $\{\theta^{(i)}\}_{i=1}^N$ according to

$$\theta^{(i)} \sim p(\theta|\mathcal{D}). \quad (2.16)$$

2. *Integrate* some function $h(\cdot)$ with respect to the posterior distribution and calculate its expectation

$$\mathbb{E}_{\theta \sim p(\cdot|\mathcal{D})}[h(\theta)] = \int h(\theta) p(\theta|\mathcal{D}) d\theta. \quad (2.17)$$

In practice, however, producing i.i.d. samples may not be possible, and we instead approximate (2.16) using *dependent* samples (see Section 2.3.2). We can then use resulting samples $\{\theta^{(i)}\}_{i=1}^N$ to learn about $p(\theta|\mathcal{D})$. This approach is otherwise known as *Monte Carlo sampling*.

In the latter case, if we are able to sample from $p(\theta|\mathcal{D})$, the integral (2.17) can be estimated using

$$\mathbb{E}_{\theta \sim p(\cdot|\mathcal{D})}[h(\theta)] \approx \frac{1}{N} \sum_{i=1}^N h(\theta^{(i)}), \quad \theta^{(i)} \overset{i.i.d.}{\sim} p(\theta|\mathcal{D}), \quad (2.18)$$

which is otherwise known as *Monte Carlo integration*. The form of Monte Carlo integration presented in (2.18) is the simplest such form. In practice, more advanced methods relax the need to exactly and independently sample from posterior target (see Section 2.3.1).

Equation (2.18) further possesses a number of favourable features which are of much practical use later. That is, (2.18) is an *unbiased*

$$\mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N h(\theta^{(i)}) \right] = \mathbb{E}_{\theta \sim p(\cdot|\mathcal{D})}[h(\theta)], \quad (2.19)$$

and *consistent*

$$\frac{1}{N} \sum_{i=1}^N h(\theta^{(i)}) \rightarrow \mathbb{E}_{\theta \sim p(\cdot|\mathcal{D})}[h(\theta)], \quad N \rightarrow \infty, \quad (2.20)$$

estimator of (2.17) (a full derivation of these properties can be found in e.g. Robert and Casella [115]). The variance of the estimate from (2.18) is given by

$$\text{Var} \left[\frac{1}{N} \sum_{i=1}^N h(\theta^{(i)}) \right] = \frac{\text{Var}_{\theta \sim p(\cdot|\mathcal{D})} [h(\theta)]}{N}, \quad (2.21)$$

which decreases according to σ^2/N , where σ^2 is the variance of $h(\cdot)$ under the posterior distribution – assuming a finite $\text{Var}(h(\theta))$. Therefore, given access to sufficiently large N independent samples from our target distribution, the task of integration becomes feasible to a small approximation error. In that sense, the principle challenge in Monte Carlo methods is in sampling from analytically intractable distributions.

2.3.1 Importance sampling

Importance sampling (IS) (see e.g. Robert [114]) is a Monte Carlo integration strategy. It is not, in the first instance, a technique used for Monte Carlo sampling (although its application to this problem is considered later). It begins by selecting an *importance density* $q(\theta)$ which is easily sampled from, and satisfies $\text{supp } q(\theta) \supseteq \text{supp } p(\theta|\mathcal{D})$, where supp is the support of the distribution. Ideally, $\text{supp } q(\theta) = \text{supp } p(\theta|\mathcal{D})$. IS samples independent $\tilde{\theta}^{(1)}, \tilde{\theta}^{(2)}, \dots, \tilde{\theta}^{(N)}$ from q and calculates weights

$$w_i = \frac{p(\tilde{\theta}^{(i)}, \mathcal{D})}{q(\tilde{\theta}^{(i)})}. \quad (2.22)$$

Then, for any function h , we can estimate the expectation

$$\mathbb{E}_{\theta \sim p(\cdot|\mathcal{D})} [h(\theta)] \approx \frac{\sum_{i=1}^N h(\tilde{\theta}^{(i)}) w_i}{\sum_{i=1}^N w_i}. \quad (2.23)$$

This is consistent, but in practice q should approximate the posterior to achieve low-variance estimation at a feasible cost. Also note that $N^{-1} \sum_{i=1}^N w_i$ is an unbiased and consistent estimate of the model evidence.

An informal diagnostic for the quality of IS results is the *effective sample size* (ESS). This is defined as

$$N_{\text{eff}} = \frac{\left(\sum_{i=1}^N w_i \right)^2}{\sum_{i=1}^N w_i^2}. \quad (2.24)$$

For most functions $h(\theta)$, the variance of (2.23) approximately equals that of idealised Monte Carlo estimates based on N_{eff} independent samples from $p(\theta|\mathcal{D})$ [87].

Importance resampling

In addition to computing expectations, IS can be used to approximately sample from intractable target densities. This technique is known as *importance resampling*, and is a foundational tool for many Monte Carlo methods (e.g. *Sequential Monte Carlo* [36]).

Given a collection of importance weights $\{w_i\}_{i=1}^N$ from (2.22), importance resampling works by calculating the *normalized* weights

$$\tilde{w}_i = \frac{w_i}{\sum_{i=1}^N w_i}, \quad (2.25)$$

and then redrawing, with replacement, N $\tilde{\theta}$ values from $\{\tilde{\theta}^{(i)}\}_{i=1}^N$ according to a multinomial distribution, where the probability of sampling the i^{th} element is equal to $\tilde{w}_i$.

To illustrate this, suppose we have a target density $p(\theta) = \mathcal{N}(\theta|0, 1)$, truncated to $-6 \leq \theta \leq 6$, which we (obviously incorrectly) assume the inability to sample from directly. We then select a uniformly distributed importance density $q(\theta) = \mathcal{U}(\theta| -6, 6)$, and perform importance resampling. This process is presented in Figure 2.3, along with pdfs for both the target and importance densities, and a histogram estimating the resulting density based on the resampled points.

In practice, however, it is often challenging to choose appropriate importance densities that: a. remain computationally tractable; b. avoid large weights for all $\tilde{\theta}$. The latter cedes that it is insufficient to solely satisfy $\text{supp } q(\theta) \supseteq \text{supp } p(\theta|\mathcal{D})$. Indeed, $q(\cdot)$ should closely approximate $p(\cdot)$. Poor approximations – particularly if the tails of $q(\cdot)$ are too light – can lead to hugely inefficient algorithms. For example, using $q(\theta) = \mathcal{N}(\theta|0, 1)$ for $p(\theta) = \text{Cauchy}(\theta|0, 1)$.

2.3.2 MCMC

Markov chain Monte Carlo (MCMC) techniques are a large family of algorithms used to sample from distributions whose densities can only be evaluated up to proportionality (see e.g. Robert and Casella [115] or Gamerman and Lopes [49] for a rigorous introduction). MCMC works by constructing a *Markov chain* with *stationary distribution* equal to the target distribution. Provided convergence of the chain to its stationary distribution, any sample from that chain is guaranteed to produce asymptotically exact samples from the target [114]. MCMC algorithms can be additionally used as a technique for Monte Carlo integration. Whilst beyond the scope of what is presented here, proof of consistency under the central limit theorem for the resulting estimator (2.18) can be found in e.g. Gilks et al. [53]. Below we first review Markov chains and their stationary distributions. We then present one

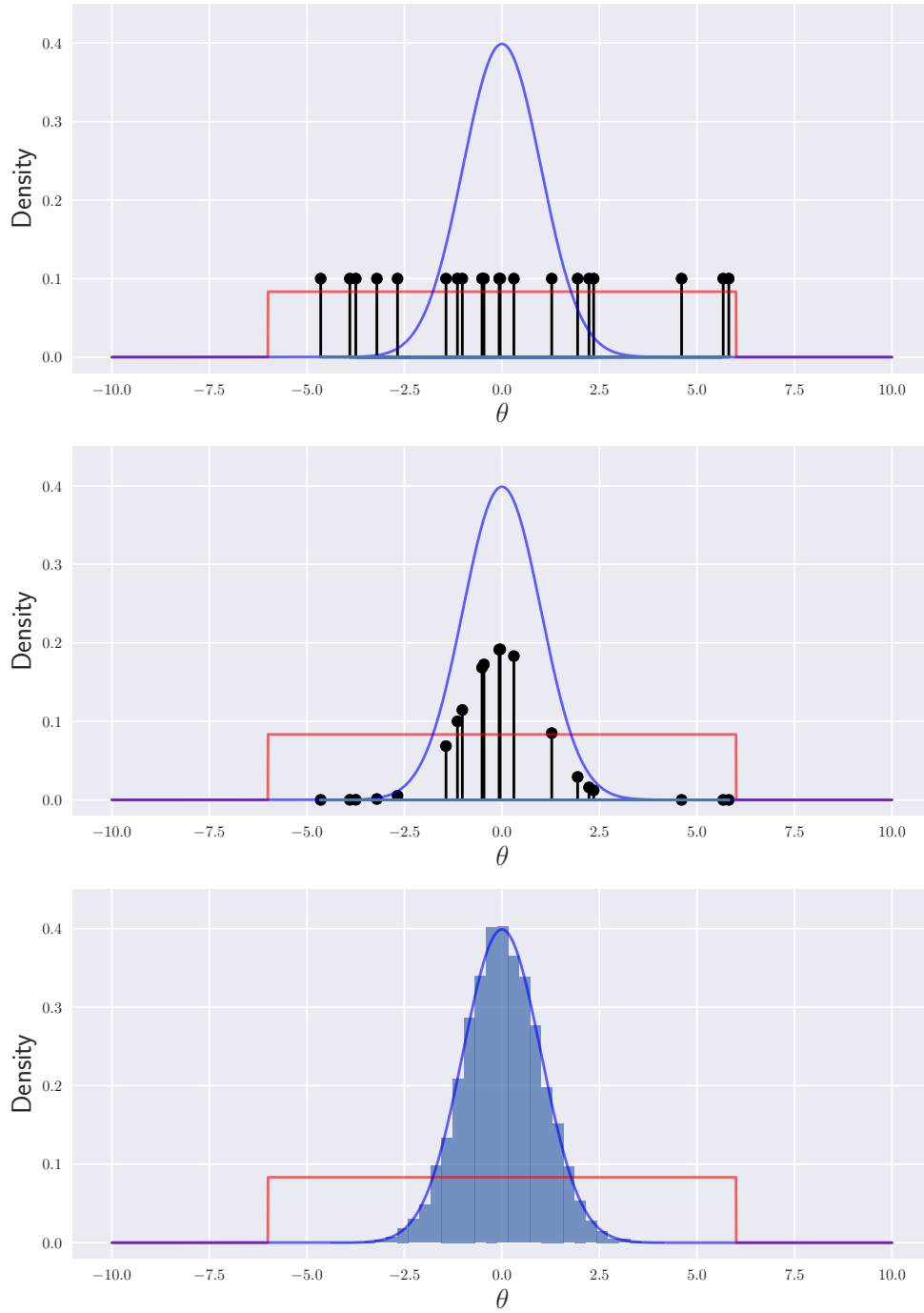


Figure 2.3 Bottom, middle and top: pdf of uniform $\mathcal{U}(-6, 6)$ (red line) and pdf of truncated $\mathcal{N}(0, 1)$ (blue line). Top only: 20 importance samples with lines of height proportional to their density under $q(\cdot)$. Middle only: the same 20 importance samples with lines of height proportional to their respective $\tilde{w}_i$. Bottom only: normalized histogram of 10,000 resampled points from the importance density.

common approach to constructing these chains before applying it in the context of posterior inference: the *Metropolis-Hastings* algorithm.

Markov chains

Consider an *ordered* set of random variables $\{\theta^{(i)}\}_{i=1}^N$, where each $\theta^{(i)} \in \mathbb{R}^d$ (we note the domain $\mathbb{R}^d$ can be broadly generalised [98]). Now suppose that the conditional probability density for the $i + 1^{th}$ variable can be written such that

$$f(\theta^{(i+1)}|\theta^{(i)}, \dots, \theta^{(1)}) = f(\theta^{(i+1)}|\theta^{(i)}), \quad (2.26)$$

where $f(\theta^{(i+1)}|\theta^{(i)})$ is *one-step transition kernel*. In this case, (2.26) defines a discrete-indexed, continuous-valued first-order *Markov chain* (e.g. [85, 98]). The essential property is that all future values are conditionally independent of the past values, given the current value. Note that the one-step transition kernel on the right-hand side of (2.26) does not directly depend on i . In other words, we assume *homogeneous* Markov chains. This Markov chain is further the first example of a *stochastic process* (see Chapter 3).

Stationary Distribution Let $\pi^{(i)}(\cdot)$ define a probability density function at index i . Further let ϑ index the value of the chain at step i , and θ index the value of the chain at step $i + 1$. We observe that

$$\pi^{(i+1)}(\theta) = \int_{\mathbb{R}} f(\theta|\vartheta) \pi^{(i)}(\vartheta) d\vartheta. \quad (2.27)$$

The stationary distribution is the probability density function $\pi(\cdot)$ satisfying

$$\pi(\theta) = \int_{\mathbb{R}} f(\theta|\vartheta) \pi(\vartheta) d\vartheta. \quad (2.28)$$

The stationary distribution is such that if, at some i , we have $\pi^{(i)}(\cdot) = \pi(\cdot)$ then $\pi^{(i+j)}(\cdot) = \pi(\cdot) \forall j > 0$. Informally, if a chain has a stationary distribution, then it can, under specific conditions, be seen as the chain's *limiting distribution*. A detailed discussion on the theoretical criteria for the convergence to this limit can be found in Gilks et al. [53], which is beyond the scope of what is considered here.

Metropolis-Hastings

Suppose we wish to sample from $p(\theta|\mathcal{D})$, assuming only access to (2.12). The Metropolis-Hastings algorithm constructs a Markov chain which asymptotically converges to $p(\theta|\mathcal{D})$ as its stationary distribution. It begins by introducing a one-step proposal density $k(\theta^*|\theta)$ (we further refer to this as a proposal *mechanism*). At each iteration, the transition proposal

density proposes a new value for θ^* , conditional on the current value in the chain $\theta^{(i)}$. This new value is then accepted with some *acceptance probability*, p_{accept} . The algorithm is presented in Algorithm 1.

In practice, it is only possible to run the Metropolis-Hastings algorithm for a finite number of iterations. It is therefore important to utilise diagnostic checks as to establish (approximate) convergence of the chain. Whilst there exist many convergence checks (see e.g. Gamerman and Lopes [49]), much of it is beyond the scope of what is presented here. Instead, we consider two informal (but important) diagnostic concepts illustrated below for the MCMC regression example: *burn-in* and *autocorrelation*.

Often initialisation of the chain at $\theta^{(0)}$ is such that we initially generate samples far from the target distribution. Therefore, it is helpful to discard the values for $\theta^{(i)}$ prior to approximate convergence. The period is known as burn-in. The burn-in period can be visually assessed using a *trace plot*: a plot of the chain's values against the iteration of the algorithm.

Ideally Monte Carlo sampling should return independent draws [53]; however, successive samples of the chain will be *dependent*. The extent of this dependence can be assessed using autocorrelation: the correlation present in the chain lagged ρ steps with itself. Or rather

$$r_\rho = \frac{\mathbb{E} \left[\left(\theta^{(i)} - \mu \right) \left(\theta^{(i+\rho)} - \mu \right) \right]}{\sigma^2}, \quad (2.29)$$

where μ and σ are the mean and variance of θ . The *integrated autocorrelation time* (ACT) of the chain is given by

$$\text{ACT} = 1 + 2 \sum_{k=1}^{\infty} r_k, \quad (2.30)$$

where, in practice, (2.30) and (2.29) require numerical estimation. The ACT represents the effective number of dependent samples that is equivalent to a single independent sample [129]. It is intuitively clear we should aim to minimise the sum on the right-hand side of (2.30), where autocorrelation plots (see e.g. Figure 2.4) should return values close to 0 for all $\rho > 0$. In the instance of highly correlated samples [53], the chain is commonly *thinned*. The resulting act of *thinning* is when we approximate the target with every n^{th} sample, where all intermediate points are discarded. We refer to the n we choose as the *thinning factor*. In practice, the initial choice of thinning factor is often the largest lag whose autocorrelation appears significantly different to zero. We further note that discarding samples in this way is non-desirable as it wastes otherwise useful computation. Preferably, thinning would only be used to avoid storing the entire chain when performing long runs.

Algorithm 1 The Metropolis-Hastings algorithm

Initialise $\theta^{(0)}$ and $i = 0$.

loop

Sample $\theta^* \sim k(\theta^*|\theta^{(i)})$.

Evaluate the acceptance probability

$$p_{\text{accept}} = \min \left(1, \frac{p(\mathcal{D}|\theta^*) p(\theta^*) k(\theta^{(i)}|\theta^*)}{p(\mathcal{D}|\theta^{(i)}) p(\theta^{(i)}) k(\theta^*|\theta^{(i)})} \right)$$

Set $i := i + 1$

Set $\theta^{(i)} = \theta^*$ with probability p_{accept} . Otherwise, set $\theta^{(i)} = \theta^{(i-1)}$.

end loop

Example: Bayesian linear regression with MCMC

Recall again the regression example of (2.1). Suppose we choose a prior that doesn't present us with a closed-form solution as in (2.15). For this purpose we choose vaguely informative proper Uniform priors, where

$$p(\theta_1) = \mathcal{U}(0, 2), \quad (2.31)$$

$$p(\theta_2) = \mathcal{U}(5, 15). \quad (2.32)$$

We perform inference using the Metropolis-Hastings algorithm. Here we select $f(\theta^*|\theta) = \mathcal{N}(\theta, 0.5I_2)$, and run Algorithm 1 for 1,000,000 iterations, with a burn-in of 10,000 and apply a thinning factor of 10. Autocorrelation and trace plots, along with posterior estimates, are presented in Figures 2.4, 2.5 and 2.6, respectively. For completeness, we note that when the proposal density is symmetric $k(\theta^*|\theta) = k(\theta|\theta^*)$ (as in this one), Algorithm 1 is known as the *Metropolis* algorithm.

Limitations of MCMC

Whilst MCMC presents theoretical guarantees of asymptotic accuracy, it often comes at the cost of computational speed. MCMC algorithms – with admission of broad generality – typically suffer from two limitations:

Serial computation Implicit to the nature of a Markov chain is the sequential form of random draws. Each draw further requires evaluation of the likelihood over the full data. As such, MCMC does not typically scale well to inference using *big data*, requiring many expensive likelihood evaluations to be performed in serial. Whilst there have been attempts to utilise parallel computation in MCMC [95, 128] or data sub-sampling

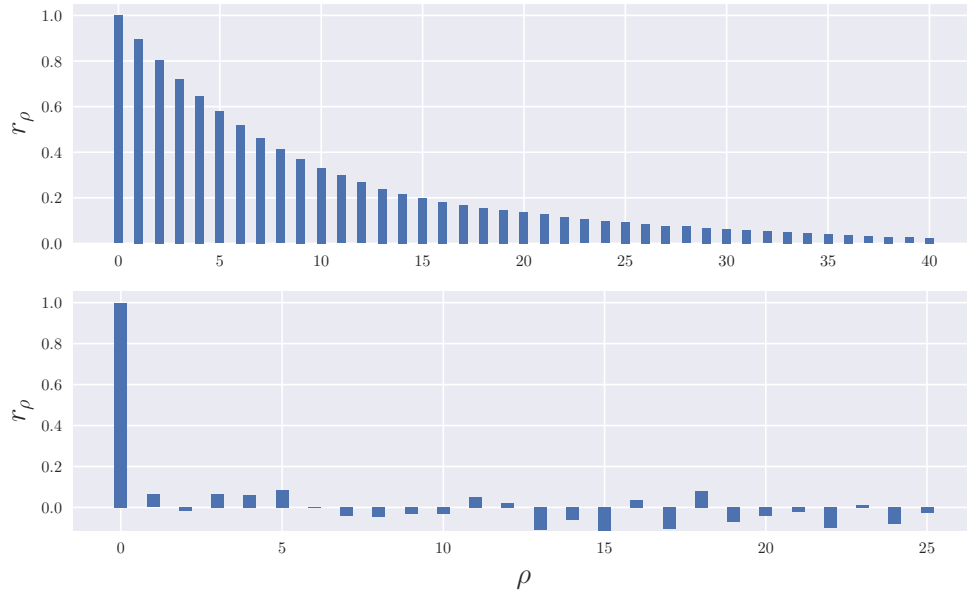


Figure 2.4 Autocorrelation plots for θ_1 in the MCMC linear regression example. Top: raw output with burn-in removed. Bottom: after applying a thinning factor of 10.

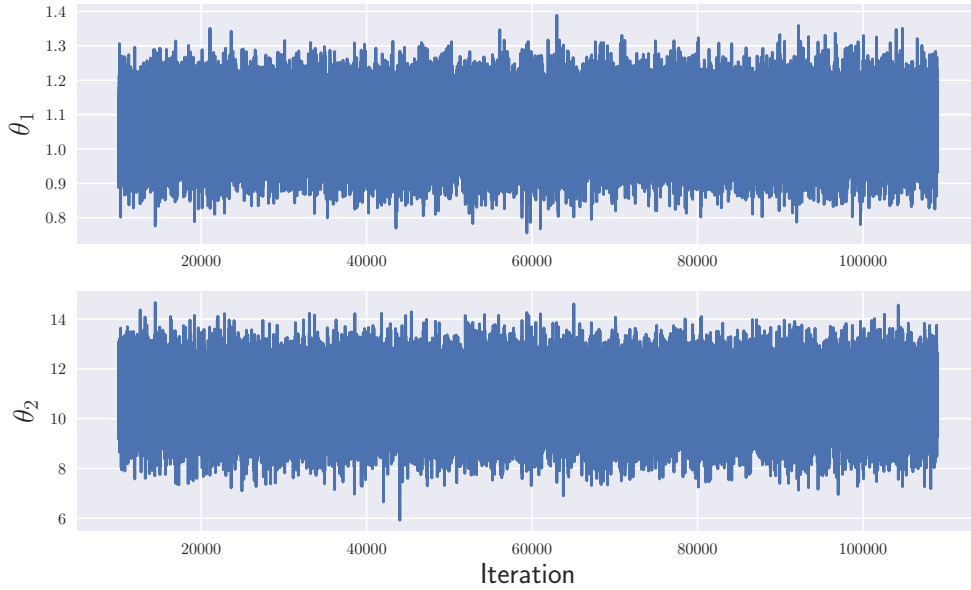


Figure 2.5 Thinned MCMC trace plots after burn-in for the linear regression example.

[108, 20], such approaches are required to eschew asymptotic guarantees or impose unrealistic assumptions on complex posteriors.

Proposal distributions Unlike in the toy regression example, posterior distributions are often high dimensional with a complex dependency structure between dimensions. As such, naive selection of proposal mechanisms for the latent variables will lead to impractically

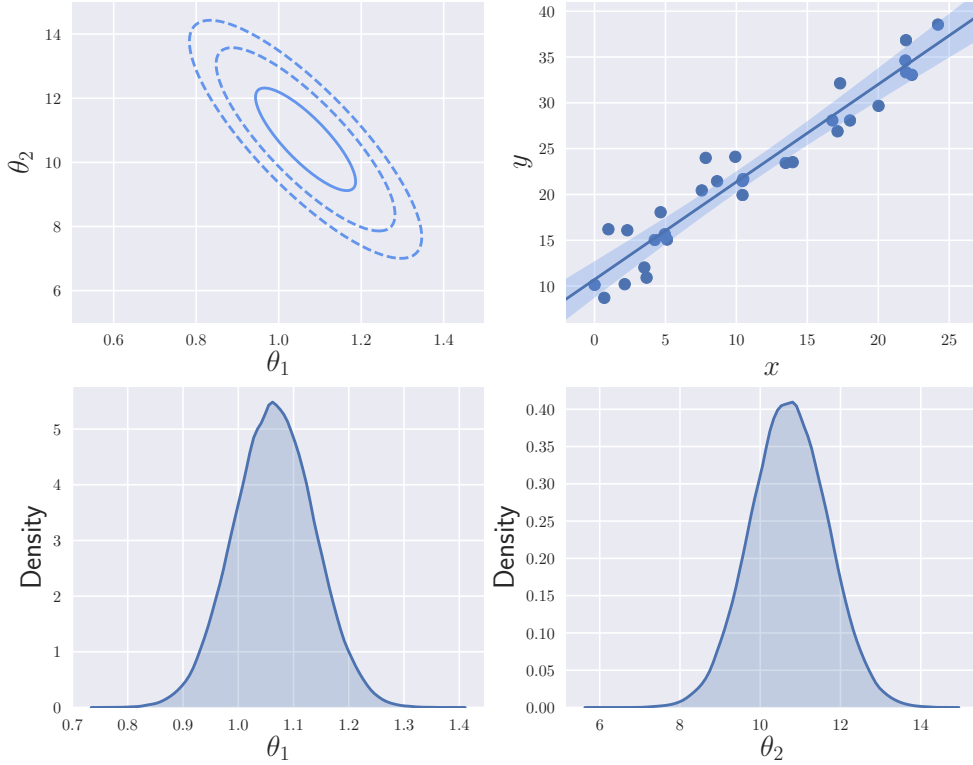


Figure 2.6 Linear regression with MCMC. Top: contour plots of the kernel density estimates (KDE) for the joint posterior θ density (left); 95% credible intervals for posterior regression model (right). Bottom: marginal posterior KDE for θ_1 (left) and θ_2 (right).

low acceptance rates, with the chain ‘stuck’ for many iterations. Indeed, much manual work is sometimes required to tune its hyper-parameters (e.g. the covariance matrix of the Normal transition kernel in the regression example). Some MCMC schemes, such as *Hamiltonian Monte Carlo* (HMC), possess few tuning requirements, but sometimes struggle with high posterior dependence [18].

2.4 Variational inference

One promising alternative to MCMC is *variational inference*. Variational inference (see e.g. Blei et al. [23]) recasts Bayesian inference from a sampling problem, as in Monte Carlo, to one of *optimisation*. In doing so, variational inference benefits from a large class of highly parallelisable optimisation procedures. Also, as we shall latterly see, variational inference is further amenable to fast data sub-sampling methods, allowing inference for big data in a reasonable time-scale. Variational inference proceeds by introducing a family of approximations to the posterior, $q(\theta; \phi)$, indexed by a set of *variational parameters*, ϕ .

Optimisation is then used to find the ϕ minimising the Kullback-Leibler (KL) divergence

$$KL(q(\theta; \phi) || p(\theta | \mathcal{D})) = \mathbb{E}_{\theta \sim q(\cdot; \phi)} [\log q(\theta; \phi) - \log p(\theta | \mathcal{D})]. \quad (2.33)$$

We note, however, that (2.33) depends on the evaluation of the posterior $p(\theta | \mathcal{D})$, which is typically assumed to be analytically intractable. Fortunately we can re-write the KL divergence as

$$KL(q(\theta; \phi) || p(\theta | \mathcal{D})) = \log p(\mathcal{D}) - \mathbb{E}_{\theta \sim q(\cdot; \phi)} [\log p(\theta, \mathcal{D}) - \log q(\theta; \phi)], \quad (2.34)$$

where only the (now tractable) terms in the expectation depend on the variational parameters. As such, *maximisation* of this term *minimizes* the KL divergence. This resulting objective is known as the evidence lower bound (ELBO) [74]

$$\mathcal{L}(\phi) = \mathbb{E}_{\theta \sim q(\cdot; \phi)} [\log p(\theta, \mathcal{D}) - \log q(\theta; \phi)], \quad (2.35)$$

where the optimal $q(\theta; \phi)$ is an approximation to the posterior distribution. In recognising that the KL divergence (2.34) is always non-negative, it is further shown that

$$\log p(\mathcal{D}) \geq \mathcal{L}(\phi) = \mathbb{E}_{\theta \sim q(\cdot; \phi)} [\log p(\theta, \mathcal{D}) - \log q(\theta; \phi)], \quad (2.36)$$

where optimising this objective maximises the *evidence lower bound* (hence, ELBO) of the marginal likelihood. The relationship between the ELBO and the model evidence has further formed the basis for (2.35) as a means of model comparison [16].

Despite its obvious promise, variational inference is not without its own limitations. Most prominently, variational inference – unlike MCMC – does not come with robust theoretical guarantees. In fact, as is discussed in Section 2.4.1, the techniques commonly used to optimise ϕ do not guarantee convergence to a global minima. We further note that the *zero-forcing* nature of (2.35) can render the posterior resulting from ELBO maximisation over-concentrated, unless the approximating family is rich enough to allow particularly close matches to the posterior [142]. We further demonstrate this effect in the linear regression example of section 2.4.1 (Figure 2.8). In that sense, variational inference should be viewed as a (hugely useful) substitute to MCMC in some situations where its analysis is computationally intractable (or, at least, infeasibly time-consuming). Even in the instance where MCMC *is* tractable, variational inference provides a useful companion tool, allowing for the rapid exploration of alternate models.

2.4.1 Stochastic gradient variational inference

There are a variety of strategies used to optimise ϕ and maximise the ELBO (see e.g. coordinate ascent variational inference (CAVI) [21]; variational message passing (VMP) [151]; variational expectation-maximisation (EMV) [16]). Here we shall focus on the most popular: optimising (2.35) with the use of *gradient descent* (e.g. [24]). Gradient descent updates the value of ϕ at the n^{th} iteration according to

$$\phi^n = \phi^{n-1} - \alpha_n \nabla_{\phi} \mathcal{C}(\phi^{n-1}), \quad (2.37)$$

where α_n is the *learning rate* (or step size) associated with iteration n , ∇_{ϕ} is the gradient operator taken with respect to ϕ and $\mathcal{C}(\cdot)$ is our *objective function*, which we take to be $-\mathcal{L}(\cdot)$. In the simplest sense, gradient descent seeks to minimise the (negative) objective function (hence, maximising the ELBO) by taking scaled ‘steps’ in the direction of the largest gradient. More generally, updates of the following form can be used

$$\phi^n = \phi^{n-1} - \gamma(\nabla_{\phi} \mathcal{C}(\phi^{n-1}), \psi), \quad (2.38)$$

where $\gamma(\cdot)$ is an *optimiser* and ψ are its hyper-parameters. Popular choices for $\gamma(\cdot)$ include updates with momentum terms [107] and the ADAM optimizer [78].

It is further the case that gradient descent methods only guarantee convergence to a local minima. Local minima are only additionally global minima if the objective function is *convex* [70]. In practice, however, most variational objective functions represent high-dimensional, *non-convex* optimisation surfaces, with multiple local minima and saddle points [23]. Such approaches therefore typically offer no guarantees of convergence to a global minima

Calculating gradients with respect to ϕ in the expectation (2.35) for use in gradient updates (2.38) is often challenging. Using an unbiased estimate of the gradient in (2.37) results in *stochastic gradient descent* (SGD) (e.g. [120], amongst others). Optimising the ELBO using SGD is otherwise known as *stochastic gradient variational Bayes* (SGVB) [80]. Unlike CAVI and VMP, which are limited to inference on exponential-family models with conjugate priors (with the exception of multinomial and binary regression [82]), SGVB is broadly applicable. SGVB is limited only by the optimiser responsible for α_n , the numerical tractability of the family posited for $q(\cdot)$ and the ability to derive unbiased ELBO gradient estimates.

SGD then reduces the update rule (2.37) to

$$\phi^n = \phi^{n-1} - \alpha_n \widetilde{\nabla_{\phi} \mathcal{C}}(\phi^{n-1}), \quad (2.39)$$

where $\widetilde{\nabla_{\phi} \mathcal{C}}(\cdot)$ is an unbiased estimate to the gradient of (2.35).

Numerical Monte Carlo solutions to the approximation of $\widetilde{\nabla}_\phi \mathcal{C}(\phi)(\cdot)$ include the REINFORCE method [150, 55] (also known as a likelihood ratio or score function estimator). The REINFORCE method uses the ‘log-derivative trick’ to express the gradient of $q(\cdot)$ as

$$\nabla_\phi q(\theta; \phi) = q(\theta; \phi) \nabla_\phi \log q(\theta; \phi), \quad (2.40)$$

which allows the ELBO gradient to be written as

$$\begin{aligned} \nabla_\phi \mathcal{L}(\phi) &= \mathbb{E}_{\theta \sim q(\cdot; \phi)} [(\log p(\theta, \mathcal{D}) - \log q(\theta; \phi)) \nabla_\phi \log q(\theta; \phi)], \\ \implies \nabla_\phi \mathcal{C}(\phi) &= \mathbb{E}_{\theta \sim q(\cdot; \phi)} [(\log q(\theta; \phi) - \log p(\theta, \mathcal{D})) \nabla_\phi \log q(\theta; \phi)]. \end{aligned} \quad (2.41)$$

It then follows that the unbiased gradient estimator is given by

$$\widetilde{\nabla}_\phi \mathcal{C}(\phi) = \frac{1}{N} \sum_{i=1}^N \left[(\log q(\theta^{(i)}; \phi) - \log p(\theta^{(i)}, \mathcal{D})) \nabla_\phi \log q(\theta^{(i)}; \phi) \right], \quad (2.42)$$

where $\theta^{(i)}$ is the i -th θ sample used for Monte Carlo estimation. The resulting gradient estimates are, however, typically found to exhibit high variance [100] (see Gal [48] for a detailed variance analysis across all numerical estimators).

One alternative approach to mitigating the high variance of Monte Carlo estimators is to use the *reparameterization trick* ([80], [112] and [134]). The reparameterization trick seeks to express the latent variables θ as the output of an invertible deterministic function, $g(\epsilon, \phi)$, for some random variable ϵ with a fixed, tractable distribution $p(\epsilon)$, such that

$$\theta^{(i)} = g(\epsilon^{(i)}, \phi), \quad \epsilon^{(i)} \stackrel{i.i.d.}{\sim} p(\epsilon). \quad (2.43)$$

Using the *Law of the Unconscious Statistician* [119], it is then possible to re-write (2.35) as

$$\mathcal{L}(\phi) = \mathbb{E}_\epsilon [\log p(g(\epsilon, \phi) | \mathcal{D}) - \log q(g(\epsilon, \phi))]. \quad (2.44)$$

This reformulation gives rise to the following gradient

$$\begin{aligned} \nabla_\phi \mathcal{L}(\phi) &= \nabla_\phi \mathbb{E}_\epsilon [\log p(g(\epsilon, \phi) | \mathcal{D}) - \log q(g(\epsilon, \phi))], \\ &= \mathbb{E}_\epsilon [\nabla_\phi \{\log p(g(\epsilon, \phi) | \mathcal{D}) - \log q(g(\epsilon, \phi))\}], \\ \implies \nabla_\phi \mathcal{C}(\phi) &= \mathbb{E}_\epsilon [\nabla_\phi \{\log q(g(\epsilon, \phi)) - \log p(g(\epsilon, \phi) | \mathcal{D})\}], \end{aligned} \quad (2.45)$$

which results in the unbiased Monte Carlo estimate

$$\widetilde{\nabla}_\phi \mathcal{C}(\phi) = \frac{1}{N} \sum_{i=1}^N \left[\nabla_\phi \{\log q(\theta^{(i)}; \phi) - \log p(\theta^{(i)}, \mathcal{D})\} \right]. \quad (2.46)$$

Estimating $\widetilde{\nabla}_{\phi} \mathcal{C}(\phi)$ in this way is otherwise known as the *reparameterization gradient*. A more detailed discussion on the derivation of the above can be found in e.g. Mohamed et al. [94]. In practice, we typically calculate the gradients in (2.46) using automatic differentiation [83].

Mini-batch variational inference

Suppose we have an i.i.d. model for our data $\mathcal{D} = \{y_i | i = 1, 2, \dots, n\}$ such that

$$p(\mathcal{D}|\theta) = \sum_{i=1}^n \log p(y_i|\theta), \quad (2.47)$$

where $\log p(y_i|\theta)$ is the log-density for a single y_i . Substituting (2.47) into (2.35), the ELBO becomes

$$\mathcal{L}(\phi) = \mathbb{E}_{\theta \sim q(\cdot; \phi)} \left[\sum_{i=1}^n \log p(y_i|\theta) + \log p(\theta) - \log q(\theta; \phi) \right]. \quad (2.48)$$

Further define the subset $B \subseteq 1:n$ with length m , such that B is chosen uniformly at random, without replacement. We refer to the subset of the data which B defines as a *mini-batch*. Using a mini-batch allows for an unbiased estimate of the ELBO

$$\mathcal{L}(\phi) = \mathbb{E}_{\theta \sim q(\cdot; \phi)} \left[\frac{n}{m} \sum_{i \in B} \log p(y_i|\theta) + \log p(\theta) - \log q(\theta; \phi) \right], \quad (2.49)$$

where the resulting unbiased Monte Carlo estimate of (2.35) is

$$\tilde{\mathcal{L}}(\phi) = \frac{1}{N} \sum_{j=1}^N \left[\frac{n}{m} \sum_{i \in B} \log p(y_i|\theta^{(j)}) + \log p(\theta^{(j)}) - \log q(\theta^{(j)}; \phi) \right]. \quad (2.50)$$

Similarly, it follows that (2.46) becomes

$$\widetilde{\nabla}_{\phi} \mathcal{C}(\phi) = \frac{1}{N} \sum_{j=1}^N \left[\nabla_{\phi} \left\{ \log q(\theta^{(j)}; \phi) - \frac{n}{m} \sum_{i \in B} \log p(y_i|\theta^{(j)}) - \log p(\theta^{(j)}) \right\} \right]. \quad (2.51)$$

In this way, we are able to implement efficient data sub-sampling and scale inference to a high-dimensional $\mathcal{D}$, side-stepping the challenge of evaluating the full-data likelihood at each iteration. This is a technique not easily achieved with MCMC.

It is further the case that $g(\cdot)$ is often a highly parameterised function approximator – most commonly, a neural network. In this instance, ϕ are the parameters of the approximator (in the case of a neural network, its weights and biases). Inference with $g(\cdot)$ parametrised in this creates a collection of methods known as *black-box variational inference* [109] (BBVI).

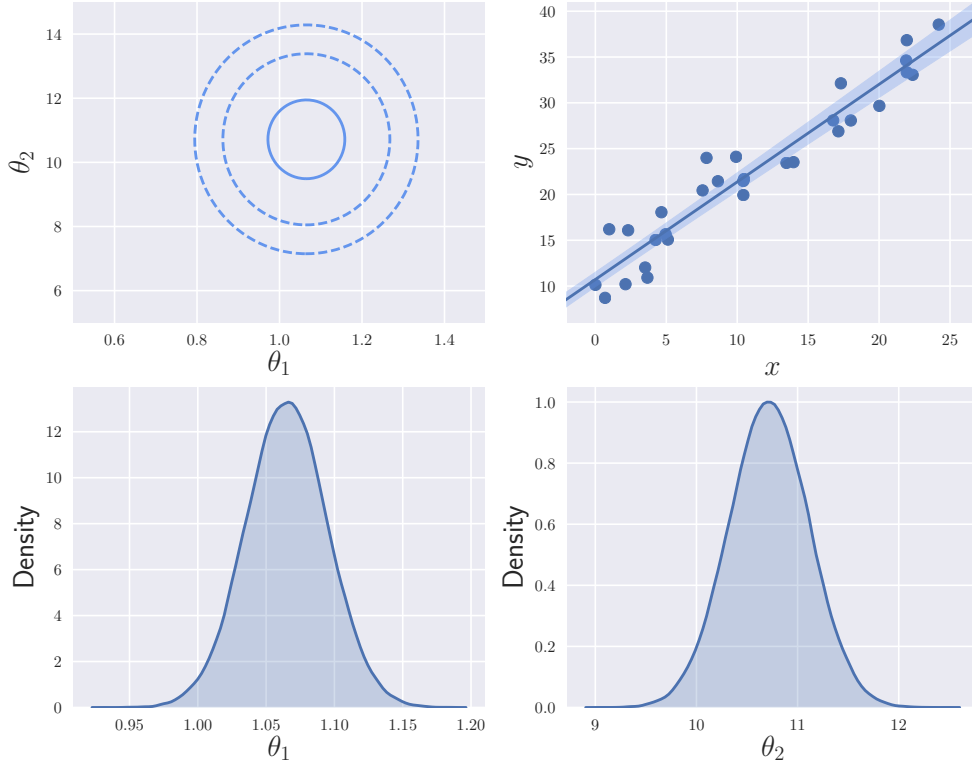


Figure 2.7 Linear regression with variational inference. Top: contour plots of the joint θ density (left); 95% credible intervals for posterior regression model (right). Bottom: approximate marginal posterior densities for θ_1 (left) and θ_2 (right).

Example: linear regression with variational inference

To illustrate performing inference with the reparameterization trick, we again revisit the regression example of (2.1). We begin by positing a family of distributions for our variational posterior, $q(\theta; \phi_\theta)$. For this we use the simple *mean-field Gaussian* approximation

$$q(\theta; \phi) = \prod_{i=1}^2 \mathcal{N}(\theta_i; \mu_i, \sigma_i^2), \quad (2.52)$$

with $\phi = (\mu_1, \mu_2, \sigma_1, \sigma_2)$. Recall that $\mathcal{N}(\cdot; \mu, \sigma^2)$ is the Gaussian density with mean μ and variance σ^2 . Hence the components of θ are independent Gaussians. Using this simple family of approximate distributions not only serves to elucidate the methodology, but further demonstrates the limitations of this approach.

To express θ using the reparameterization trick, we write

$$\theta^{(i)} = g(\epsilon^{(i)}, \phi_\theta) = S\epsilon^{(i)} + \mu, \quad (2.53)$$

where $\epsilon \sim \mathcal{N}(0, I_2)$, $S = \text{diag}(\sigma_1, \sigma_2)$ and $\mu = (\mu_1, \mu_2)$.

Again assuming $\mathcal{U}(0, 2)$ and $\mathcal{U}(5, 15)$ priors for θ_1 and θ_2 , we perform inference on the data presented in Figure 2.1, using $N = 30$ samples to estimate the ELBO. Resulting estimates for $p(\theta|\mathcal{D})$ are available in Figure 2.7 along with credible intervals for the posterior regression model. We additionally compare the joint posterior density estimates of mean-field variational inference and MCMC in Figure 2.7. Here we note the insufficiently expressive approximation results in a posterior that has found the mode well but has underestimated the support of $p(\theta|\mathcal{D})$, a problem typical of variational inference with a naive choice of $q(\cdot)$.

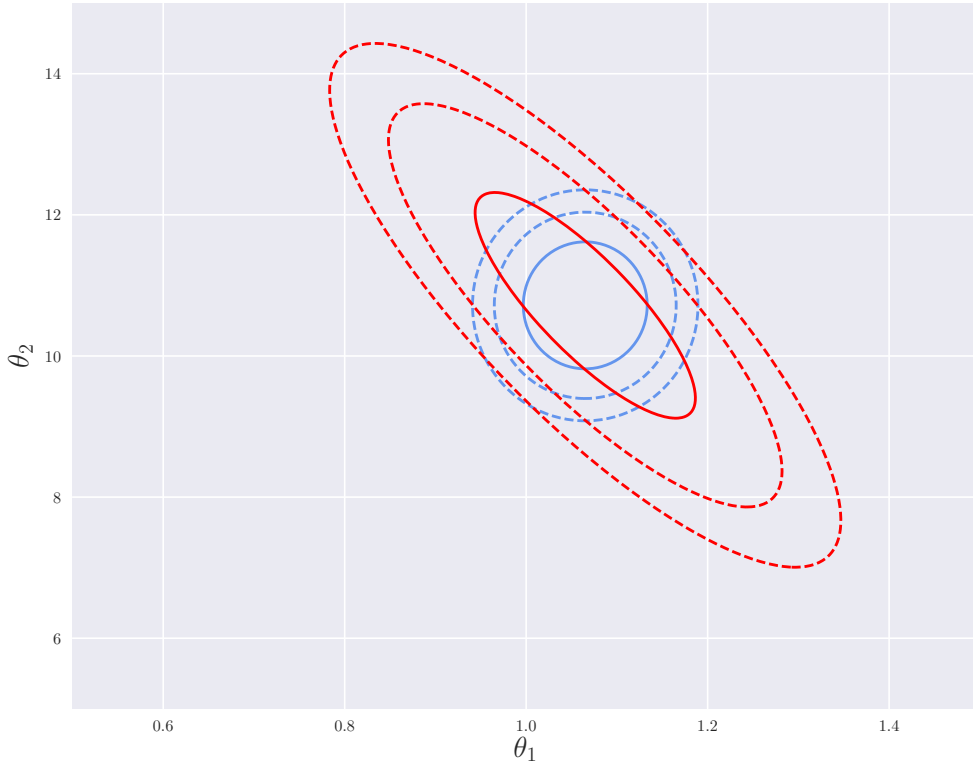


Figure 2.8 Contour plots of the joint θ density for MCMC (red) and the mean-field variational approximation (blue) for linear regression example.

Chapter 3

Stochastic Processes

3.1 Introduction to stochastic processes

Data can assume many forms. Images can be represented using RGB tensors; text into matrices with vector encoding [93]; social networks as large graphs of edges and nodes. It is often the case that these data – and the processes they represent – possess some inherent structure or form. When attempting to formulate a probabilistic interpretation for these data, we would like to take this into account, arming our data distribution with a broader sense of understanding.

For the purposes of this thesis, we focus on one such collection of models and the data they generate: *stochastic processes*. Intuitively, a stochastic process generates data indexed by some other variable (e.g. the position in a network). More formally, a stochastic process defines a probability distribution over a potentially infinite collection of random variables $\{x_t\}$, such that t is a member of some set. For our purposes, we will concentrate on the instance where t is a real number representing time.

In this chapter, we limit our discussion to three families of stochastic processes and the probability distributions they induce: state-space models (SSMs), stochastic differential equations (SDEs) and Gaussian processes (GPs). In presenting our methodology, we further concentrate on discrete-time representations – or approximations – to either the model likelihood or posterior distribution. As such, we omit considerations such as the Kolmogorov extension theorem and the broad generality it presents for all probability distributions defined on infinite-dimensional spaces (see e.g. Rosenthal [118] for an introduction, or Bhattacharya and Waymire [19] for a more thorough treatment).

Whilst we do not present a comprehensive overview of stochastic processes, to demonstrate how they interrelate – both to broader theory and to each other – we present a *roadmap*, of

conditional probability density governing the process at time t_i is

$$p(x_{t_i} | x_{t_{i-1}}, x_{t_{i-1}}, \dots, x_{t_0}) = p(x_{t_i} | x_{t_{i-1}}), \quad (3.1)$$

where $p(x_{t_i} | x_{t_{i-1}})$ is known as the *transition density*. Here, we omit an explicit dependency on time in the transition density and further assume times are *equally spaced*. As such, we examine the instance in which (3.1) is *time-homogeneous*, but nonetheless depends on $t_i - t_{i-1}$. We present the time-homogeneous case for ease of presentation; indeed, the considered models and proposed methods can easily be extended to the time-*inhomogeneous* case.

Continuous-time Markov processes Whilst the majority of our problem formulations reduce to a discrete-time setting, it is nonetheless instructive to informally examine the *continuous-time* Markov process. Whilst a more rigorous definition is beyond the scope of this thesis (see e.g. Øksendal [98]), we note that, after defining an infinitesimal time dt , the definition (3.1) can be intuitively extended, such that

$$p(x_{t+dt} | \{x_t | t \in [0, t]\}) = p(x_{t+dt} | x_t), \quad \forall t \in [0, \infty). \quad (3.2)$$

3.2 State-space models

SSMs provide a flexible and interpretable framework for modelling sequential data. SSMs assume a latent, first-order *Markov process* x at times $0 = t_0, t_1, \dots, t_P = T$. For ease of presentation, we assume these times are *evenly spaced*. More generally, as discussed in Section 3.4, many Gaussian process models can be represented in a state-space form [127, 145].

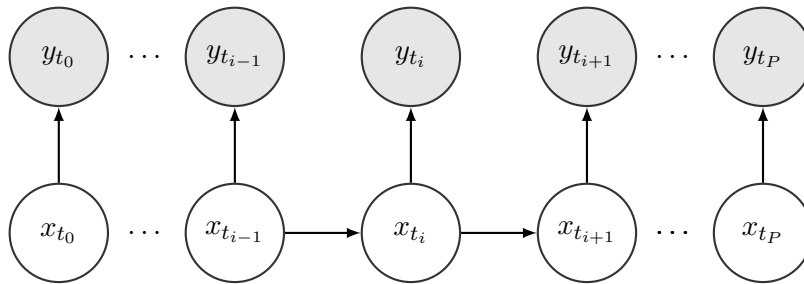


Figure 3.2 Directed graph demonstrating the dependency structure of observable states (grey) to latent states (white). Here the arrows demonstrate the conditional probability flow.

We focus on the case of continuous states $x_{t_i} \in \mathbb{R}^d$. The states evolve through a transition density $p(x_{t_i} | x_{t_{i-1}}, \theta)$, where θ denotes a vector of model parameters which govern the evolution of the process. Observations y_{t_i} are assumed available according to an *observation likelihood* $p(y_{t_i} | x_{t_i}, \theta)$, where the $\{y_{t_i}\}$ are conditionally independent given $\{x_{t_i}\}$. We ad-

ditionally assume that the initial state x_{t_0} is distributed according to $p(x_{t_0})$. Further, we choose x to denote the whole sequence of states and $x_{t_{a:b}}$ to denote a subsequence $(x_{t_i})_{a \leq i \leq b}$. More generally, we sometimes use $a:b$ to represent the sequence $(a, a+1, \dots, b)$. This system is visually described with the directed graph in Figure 3.2.

Example: random walk

Consider the simple univariate SSM defined by the generative rule

$$x_{t_{i+1}} = x_{t_i} + \epsilon_{t_i}, \quad \epsilon_{t_i} \sim \mathcal{N}(0, \sigma^2), \quad (3.3)$$

where the corresponding joint density is given by

$$p(x) = p(x_{t_0}) \prod_{i=1}^P \mathcal{N}(x_{t_i}; x_{t_{i-1}}, \sigma^2). \quad (3.4)$$

We can generate samples from this model by choosing an initial state $x_{t_0} \sim p(x_{t_0})$ and a value for σ . To illustrate this, we set $x_0 = 0$, $\sigma = 1$ and generated 30 samples at integer times with $P = 1000$. These samples are shown in Figure 3.3.



Figure 3.3 30 samples of a simple random walk as specified by (3.3).

Posterior inference

Assuming observation times indexed by the set $Q \subseteq 0:P$, with length $D + 1$, (broadly we will use the set Q to index observation times throughout this thesis) Bayesian inference for SSMs typically concerns itself with the posterior distribution of both model parameters and the latent process

$$p(x, \theta | y) \propto p(x | \theta) p(y | x, \theta) p(\theta), \quad (3.5)$$

where $p(\theta)$ is the prior ascribed to θ . Here the model likelihood is defined as a product over transition densities, so that

$$p(x | \theta) = p(x_{t_0}) \prod_{i=1}^P p(x_{t_i} | x_{t_{i-1}}, \theta), \quad (3.6)$$

where $p(x_{t_0})$ is the prior for x_{t_0} . The observation likelihood is given by

$$p(y | x, \theta) = \prod_{i \in Q} p(y_{t_i} | x_{t_i}, \theta). \quad (3.7)$$

3.3 Diffusion processes

Begin by considering the random walk SSM (3.3) modified to permit a scaling of the variance with respect to the time between equally spaced state realisations

$$x_{t_i} = x_{t_{i-1}} + \sqrt{\Delta t} \epsilon_{t_i}, \quad \epsilon_{t_i} \sim \mathcal{N}(0, \sigma^2), \quad (3.8)$$

where $\Delta t = t_i - t_{i-1} = (t_P - t_0) / P$ is the *time-step*. Now suppose we are interested in examining the behaviour of (3.8) in the limit of $\Delta t \rightarrow 0$. Informally, this limit defines the continuous-time process

$$x_t = x_0 + \sigma \int_0^t dw_s, \quad (3.9)$$

where w_t is a *Brownian motion* process [140], σ^2 is the *infinitesimal variance* (for completeness, (3.9) is said to satisfy a *generalised* Brownian motion). Formally, the univariate Brownian motion process $\{w_t, t \geq 0\}$ is characterised by the following properties:

1. $w_0 = 0$.
2. $w_{t_{i+1}} - w_{t_i}$ and $w_{t_i} - w_{t_{i-1}}$ are independent $\forall 0 \leq t_{i-1} < t_i < t_{i+1} < \infty$.
3. $w_{t_i} - w_{t_{i-1}} \sim \mathcal{N}(0, t_i - t_{i-1}) \forall 0 \leq t_{i-1} < t_i < \infty$.

In this way, Brownian motion can be viewed as the continuous-time analogue of a process with independent normal increments.

The integral in (3.9), however, cannot be interpreted in the traditional Riemann sense. This necessitates its evaluation with stochastic calculus. For this purpose, we consider the integral in the *Itô* sense, and as such (3.9) defines an *Itô process* [98]. More details on the *Itô* integral can be found in Appendix A.1.

Stochastic differential equations

It is now possible to extend the simplest such *Itô* process (3.9) to include deterministic changes, along with a more flexible parametrisation of the infinitesimal variance. In this way, Brownian motion permits the powerful stochastic extension to deterministic systems described using differential operators (i.e., *ordinary differential equations* – e.g. [30]).

Begin by considering the continuous process $\{x_t, t \geq 0\}$, with $x_t \in \mathbb{R}^d$, defined by

$$x_t = x_0 + \int_0^t \alpha(x_s, \theta) ds + \int_0^t \sqrt{\beta(x_s, \theta)} dw_s, \quad (3.10)$$

where α is a d -dimensional *drift vector*, β is a $d \times d$ positive definite *diffusion matrix* ($\sqrt{\beta}$ is such that $\sqrt{\beta}\sqrt{\beta}^T = \beta$), θ is a collection of parameters that govern the aforementioned functions, and w_t is a d -vector of standard and uncorrelated Brownian motion processes, in which each component is a univariate Brownian motion.

Considering (3.10) in differential form yields a *stochastic differential equation* (SDE)

$$dx_t = \alpha(x_t, \theta)dt + \sqrt{\beta(x_t, \theta)}dw_t. \quad (3.11)$$

We further assume that $\alpha(\cdot)$ and $\beta(\cdot)$ are sufficiently regular that (3.11) has a weak non-explosive solution [98]. In this case, (3.11) defines a *diffusion process*, which evolves over time as a composition of deterministic and stochastic functions. As such, SDEs are a broad and interpretable set of powerful modelling tools used extensively in fields such as econometrics [22, 45], biology [54, 58], physics [138] and epidemiology [47]. Such processes are always Markovian.

3.3.1 Numerical solutions

It is only possible to work with closed-form solutions to SDEs (and therefore evaluate the model likelihood) in special cases. More details on the means to achieve exact solutions (i.e. *Itô's formula*), along with examples, are contained Appendix A.2. It is therefore common to

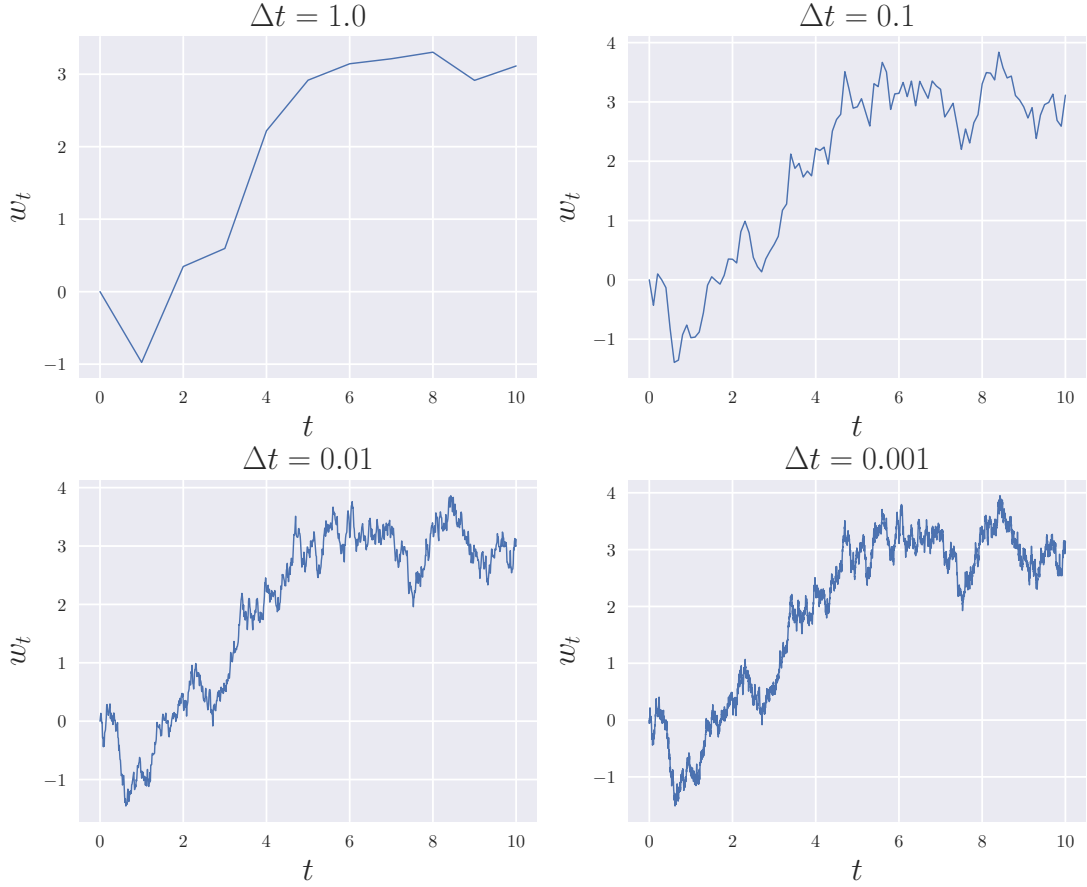


Figure 3.4 Skeleton Brownian motion paths sampled at different values of Δt .

use a numerical approximation, which utilises a discrete-time representation. The simplest and most common such discretization scheme is the Euler-Maruyama scheme. Here the diffusion process is defined only on a grid of time points, and the transition density between successive diffusion states is approximated as Gaussian. Whilst there are many alternate discretization schemes (see e.g. Kloeden and Platen [81]), higher-order techniques do not typically admit a tractable approximating transition density.

For concreteness, we present discretization for the case of equally spaced observations with $t_0 = 0$. This approach can be easily adapted to alternative specifications of time points, such as those necessitated by irregularly-spaced observation times. We begin by introducing a regular grid of times $t_0, t_1, \dots, t_{P-1}, t_P = T$, where $\Delta t = T/P$. For simplicity, we again assume observations are included in this grid, such that $Q \subseteq 0:P$, where Q indexes observation times. The Euler-Maruyama approximation of the SDE is given by

$$\Delta x_t = x_{t_{i+1}} - x_{t_i} = \alpha(x_{t_i}, \theta) \Delta t + \sqrt{\beta(x_{t_i}, \theta) \Delta t} z_{i+1}, \quad (3.12)$$

where z_{i+1} is an independent $\mathcal{N}(0, I_d)$ realisation and I_d is the $d \times d$ identity matrix. Taking the limit $\Delta t \rightarrow 0$ in an appropriate way recovers the exact SDE [127, 98]. The transition density is therefore

$$p(x_{t_{i+1}}|x_{t_i}, \theta) = \mathcal{N}(x_{t_{i+1}} - x_{t_i}; \alpha(x_{t_i}, \theta)\Delta t, \beta(x_{t_i}, \theta)\Delta t). \quad (3.13)$$

The model likelihood is then defined as

$$p(x|\theta) = \prod_{i=0}^{P-1} \mathcal{N}(x_{t_{i+1}} - x_{t_i}; \alpha(x_{t_i}, \theta)\Delta t, \beta(x_{t_i}, \theta)\Delta t). \quad (3.14)$$

In this sense (and as illustrated in Figure 3.1) we consider an SDE discretized with the Euler-Maruyama scheme as a special instance of a SSM. Indeed, (3.3) can be viewed as an example of the Euler-Maruyama scheme applied to Brownian motion ($dx_t = dw_t$), where $\Delta t = 1$ and $\sigma = 1$. Furthermore, unique to Brownian motion – as trivially follows from property (3) – simulating in this way is equivalent to its exact solution, with the resulting realisations on a discrete-time grid otherwise known as *skeleton paths*. We also visualise the effect of Δt on simulated Brownian motion paths in Figure 3.4.

3.3.2 The conditioned process

Whilst numerical solvers provide a framework to simulate a process given some initial condition, they do not extend to simulation conditioned on future observations. It is of much practical interest (as latterly discussed) to simulate from an SDE conditioned on a future state. Indeed, much work in this thesis is dedicated to overcoming the challenges resulting from this, and it is therefore useful to present the following result as a rigorous basis for future understanding.

Consider the process defined by conditioning (3.11) on an initial state, x_t and an exactly observed future state, x_{t_j} , where $t_j > t$. This conditioned process itself satisfies an SDE (see e.g. Rogers and Williams [117]) with drift and diffusion

$$\hat{\alpha}(x_t, \theta) = \alpha(x_t, \theta) + \beta(x_t, \theta) \nabla_{x_t} \log p(x_{t_j}|x_t, \theta), \quad (3.15)$$

$$\hat{\beta}(x_t, \theta) = \beta(x_t, \theta), \quad (3.16)$$

where $p(x_{t_j}|x_t, \theta)$ is the transition density of the unconditioned continuous process. Requiring this gradient renders the resulting SDE (generally) intractable, even under discretization (see Appendix B.1 for an example of a tractable conditioned process). We further note Oppen [99] presents a result for conditioning on multiple observations – both past and present – observed with error, where the conclusions are nonetheless similar. That is:

- The conditioned process satisfies an SDE.
- The diffusion matrix is unaltered.
- The drift is modified with the addition of an extra term.
- The resulting SDE is generally numerically intractable and impossible to directly simulate from.

The consequences for SDE inference are discussed in Chapter 4, and additionally motivate the work presented in Chapters 4 and 5.

3.4 Gaussian process regression

Stochastic process models like SDEs and SSMs typically offer clear model interpretability. For example, the univariate Cox-Ingersoll-Ross SDE is commonly used to model short-term interest rates [32]. The SDE is given by

$$dx_t = \theta_1 (\theta_2 - x_t) dt + \sigma \sqrt{x_t} dw_t, \quad (3.17)$$

where parameters $\theta = (\theta_1, \theta_2, \theta_3)'$ represent the mean-reversion rate, process mean and volatility, respectively. Access to posterior estimates for these parameters can therefore be a hugely useful resource, augmenting system knowledge beyond the estimation of x . However, it is sensible to ask: what if the Cox-Ingersoll-Ross doesn't 'fit' the data well? Even if it does 'fit well', what's to say it provides the 'best fit'? In this way, using an SDE or SSM for posterior inference can be interpreted as placing a strong prior over the latent process dynamics. It is often the case that these assumptions are hard to make.

Whilst sacrificing interpretability, one alternate stochastic process model that avoids this challenge is a *Gaussian process* (GP) regression [110, 97]. Given a dataset of D input times and observations $\mathcal{D} = \{(t_i, y_{t_i}) | i \in Q\}$, a GP regression, unlike e.g. discretised SDE inference, extends beyond learning a distribution over latent variables x and instead more flexibly learns a *distribution over functions* $f(\cdot)$.

A GP is defined by the property that any finite-dimensional collection of $f(\cdot)$ is multivariate Gaussian [110]

$$p(f(t_0), \dots, f(t_{n-1}), f(n)) = \mathcal{N}(f_\mu(t), K(t, t)), \quad (3.18)$$

where $t = (t_0, \dots, t_{n-1}, t_n)$, $K(\cdot, \cdot)$ is a positive semidefinite *covariance matrix* and $f_\mu(\cdot)$ is a *mean function*, commonly taken to be 0.

Each element of $K(\cdot, \cdot)$ is defined according to

$$K(t, t) = \begin{pmatrix} k(t_0, t_0) & k(t_0, t_1) & \dots & k(t_0, t_n) \\ k(t_1, t_0) & k(t_1, t_1) & \dots & k(t_1, t_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(t_n, t_0) & k(t_n, t_1) & \dots & k(t_n, t_n) \end{pmatrix}, \quad (3.19)$$

where $k(\cdot, \cdot)$ is the *kernel* function. The kernel is typically a function of additional hyperparameters, which we suppress for notational simplicity. The choice of kernel function will further determine the properties of $f(t)$ (stationarity, smoothness, periodicity, etc), and is said (along with the mean function) to induce a *prior over functions*.

A GP then relates observations to observation times according to

$$p(y|f(t), \theta) = \prod_{i \in Q} p(y_{t_i}|f(t_i), \theta), \quad (3.20)$$

where θ are the hyperparameters of the observation model. In practice, we typically assume an i.i.d. additive Gaussian noise model

$$y_{t_i} = f(t_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2), \quad (3.21)$$

eschewing the – often simpler – case of exact observations. Here we focus on the instance of $f: \mathbb{R} \rightarrow \mathbb{R}$, but note GP regression methods extend to $f: \mathbb{R}^d \rightarrow \mathbb{R}$ for arbitrary d .

A non-comprehensive definition of popular GP kernel functions and their properties is as follows, with an emphasis on the kernels we latterly use. Indeed, for a more complete overview into a broader collection of covariance kernels, see e.g. Rasmussen and Williams [110], amongst others.

Radial basis function For some t_i and t_j , the *radial basis function* (RBF) covariance kernel is defined by

$$k(t_i, t_j) = \sigma_c^2 \exp\left(-\frac{(t_i - t_j)^2}{2\lambda}\right), \quad (3.22)$$

where λ is known as the *characteristic length-scale*, which governs the intrinsic variability, and σ_c^2 is the variance. Samples from the RBF are infinitely differentiable [110], resulting in a very smooth GP. Example realisations from GPs defined by this kernel are presented in Figure 3.5.

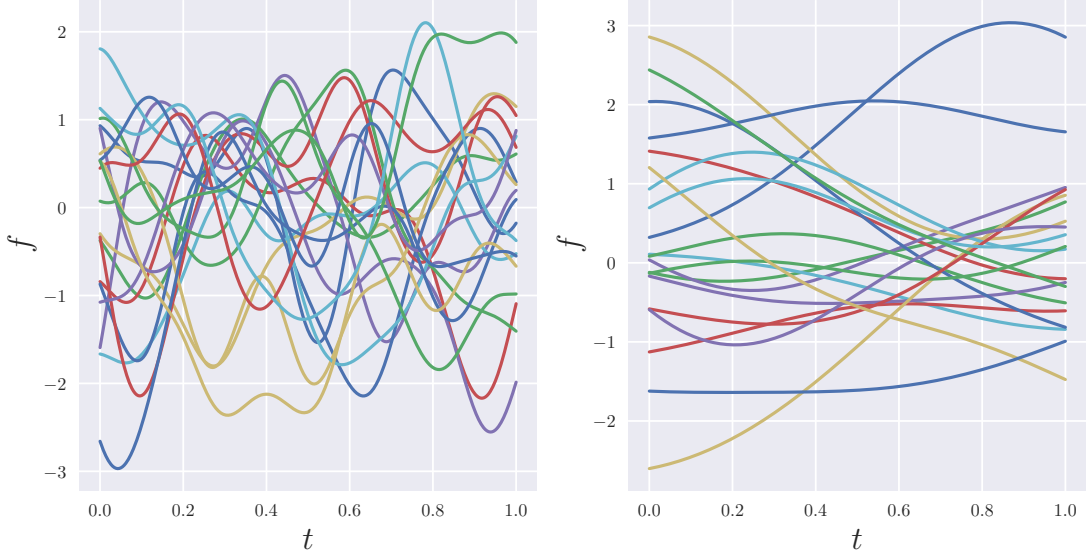


Figure 3.5 20 realisations from a GP prior with $f_\mu(t) = 0$ and an RBF kernel where $\lambda = 0.5$ (left) and $\lambda = 0.1$ (right). In both kernels examples $\sigma_c = 1.0$.

Matérn The Matérn covariance kernel is specified by

$$k(t_i, t_j) = \sigma_c^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} |t_i - t_j|}{\lambda} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu} |t_i - t_j|}{\lambda} \right), \quad (3.23)$$

where σ_c^2 is the variance, $\Gamma(\cdot)$ is the gamma function, λ is the characteristic length-scale, ν is the smoothness parameter and $K_\nu(\cdot)$ is the modified Bessel function [1]. The Matérn family of covariance kernels, unlike the RBF, returns samples of a finitely differentiable process, where the degree of differentiability is determined by ν . That is, (3.23) is k -times differentiable if $\nu > k$. Such is the significance of ν , Matérn kernels are usually specified as per Matérn- ν (e.g. Matérn- $5/2$, which is twice differentiable). Example realisations from GPs defined by this kernel are displayed in Figure 3.6.

3.4.1 Closed-form solutions

In GP inference, given a *training* set $\mathcal{D}$, our objective is to infer the distribution over functions for a set of *test* inputs $t^* = (t_1^*, t_2^*, \dots, t_n^*)'$, with corresponding outputs $f^* = f(t^*) = (f(t_1^*), f(t_2^*), \dots, f(t_n^*))'$. We then target the posterior over functions (often referred to as the *posterior process*), characterised by

$$p(f^* | \mathcal{D}, \theta) \sim \mathcal{N}(\mu^*, \Sigma^*). \quad (3.24)$$

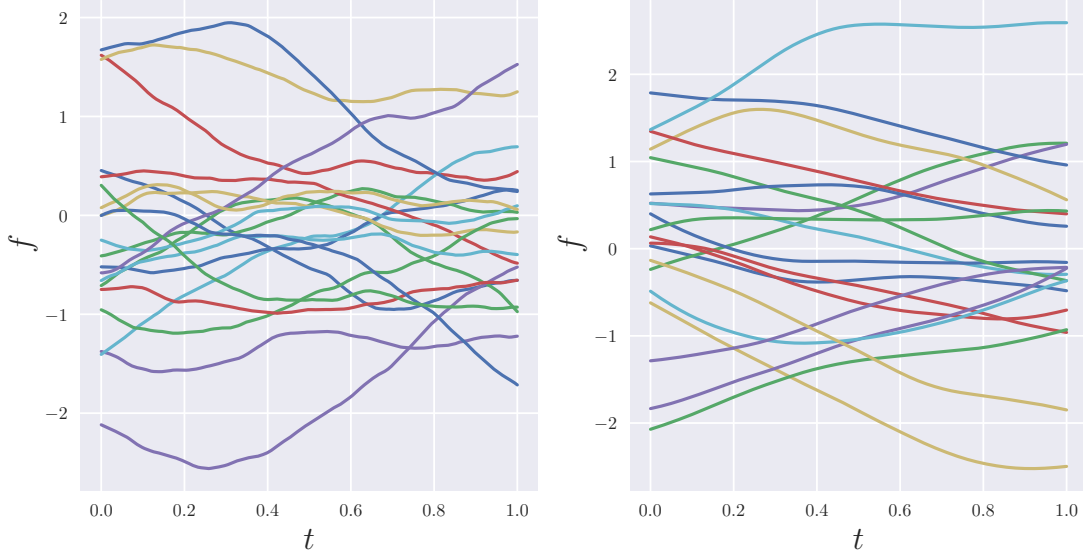


Figure 3.6 20 realisations from a GP prior with $f_\mu(t) = 0$ and an Matérn-3/2 (left) and Matérn-5/2 (right). In both examples $\sigma_c = \lambda = 1.0$.

For completeness, we proceed assuming both known kernel and observation-model hyperparameters. Where hyperparameters are additional targets of inference, they can be estimated using e.g. a maximum likelihood framework [110] or with Bayesian methods [50].

Assuming a zero-mean GP prior with arbitrary covariance kernel, we begin by obtaining

$$f(t) \sim \mathcal{N}(0, K(t, t)). \quad (3.25)$$

It follows from (3.21) that

$$\text{Cov}(y_{t_i}, y_{t_j}) = k(t_i, t_j) + \sigma^2 \delta_{ij}, \quad (3.26)$$

where δ_{ij} is the Kronecker delta. Assuming observations according to (3.21) we have

$$y \sim \mathcal{N}(0, K(t, t) + \sigma^2 I_D), \quad (3.27)$$

where I_D is the D -dimensional identity matrix. Similarly

$$\begin{aligned} \text{Cov}(f^*, y) &= \text{Cov}(f(t^*), f(t) + \epsilon), \\ &= \text{Cov}(f(t^*), f(t)) + \text{Cov}(f(t^*), \epsilon), \\ &= K(t, t^*) + 0. \end{aligned} \quad (3.28)$$

Hence, the joint distribution of observations y and test outputs f^* is given by

$$\begin{pmatrix} y \\ f^* \end{pmatrix} \sim \mathcal{N} \left\{ 0, \begin{pmatrix} K(t, t) + \sigma^2 I_D & K(t, t^*) \\ K(t^*, t) & K(t^*, t^*) \end{pmatrix} \right\}. \quad (3.29)$$

From standard multivariate Gaussian conditioning results, it then follows that conditioning f^* on y obtains a closed-form solution of the form (3.24) with

$$\mu^* = K(t^*, t) [K(t, t) + \sigma^2 I_D]^{-1} y, \quad (3.30)$$

$$\Sigma^* = K(t^*, t^*) - K(t^*, t) [K(t, t) + \sigma^2 I_D]^{-1} K(t, t^*). \quad (3.31)$$

To illustrate this in practice, we use the results of equations (3.30) and (3.31) to fit a GP regression to a toy data example. For this purpose, we simulate 10 data points in the interval $[0, 1]$ from the arbitrary function

$$\mathcal{F}(t) = -\cos(2\pi t) + \frac{1}{2} \sin(3\pi t) + 0.5t,$$

and choose to observe them as per equation (3.21) with $\sigma^2 = 0.1$. We select a Matérn-5/2 covariance kernel, with $\lambda = 0.5$, $\sigma_c = 3.0$, and choose $f_\mu = 0$. The simulated data, along with 95% credible regions for both the prior and posterior processes is displayed in Figure 3.7.

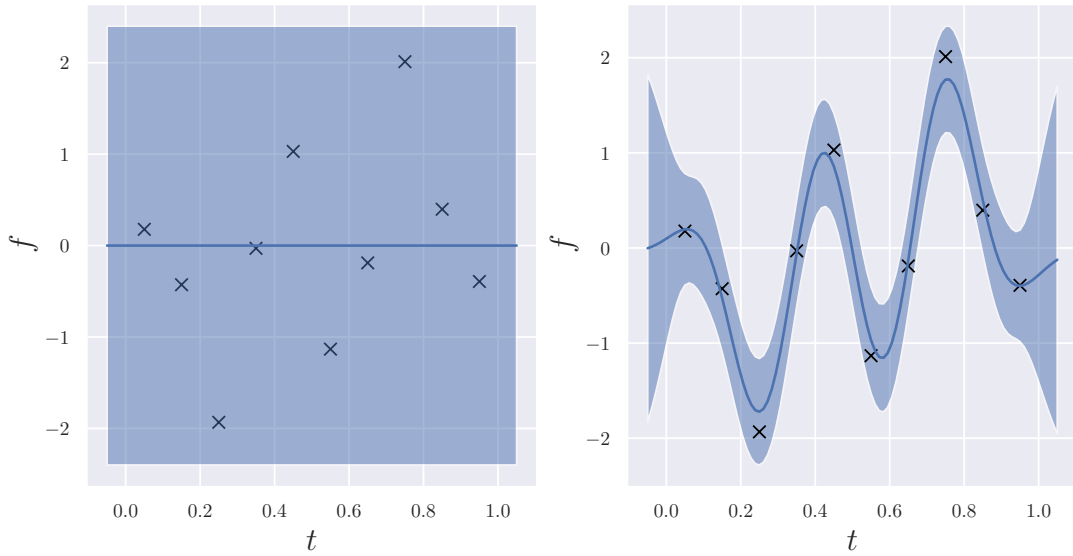


Figure 3.7 Gaussian process regression performed using a Matérn-5/2 covariance kernel. GP mean (dark blue line), 95% credible regions (light blue shaded region) and simulated data (black crosses) for prior (left) and posterior (right) processes.

A closed-form solution, however, is typically only available should the observation model be Gaussian – as in equation (3.21). In the instance of non-Gaussian observation likelihoods, the posterior is commonly intractable. This challenge is the first of two major GP limitations (the other being that of fitting GPs at scale – see Section 3.4.2). As such, this problem represents an active area of research, with solutions often utilising approximations to the posterior [96, 67]. We further present a solution to this problem, with examples, in Section 6.7.3.

3.4.2 Gaussian process state-space models

Supposing a Gaussian observation likelihood, the closed-form posterior – whilst analytically tractable – presents a significant computational hurdle. That is, equations (3.30) and (3.31) require both the inversion and storage of the covariance matrix – operations that carry $\mathcal{O}(D^3)$ and $\mathcal{O}(D^2)$ complexities, respectively. As such, naive implementation of the solution will scale poorly to even moderately large datasets. Indeed, even 2,000 data points will prove prohibitive on most modern desktop CPUs. Solutions to this problem manifest in two distinct research threads: sparse Gaussian processes [131, 26] (often possessing a variational form [133, 68]) or Gaussian process state-space models [127] (GPSSMs). The latter, where we will focus, permits solution with ensemble filtering algorithms [77]. These techniques, in contrast, typically scale with $\mathcal{O}(D)$.

More specifically, given a finitely differentiable kernel, or a continuously differentiable kernel up to some m^{th} -order approximation, we can treat the problem of GP regression as a continuous-time SDE (note that, for simplicity, we assume a zero-mean function for all of our GPSSMs). A detailed description of appropriate finitely differentiable kernels is presented in Särkkä and Solin [127] (further, an m^{th} -order approximation to the continuously differentiable RBF kernel is available in Solin and Särkkä [132]). The resulting SDE, unlike many of our considered SDE models, is both linear and time-invariant (LTI), due to the time-inhomogeneous coefficients. That is, a linear SDE is one that satisfies the drift and diffusion functions

$$\begin{aligned}\alpha(x_t, \theta) &= a_0(t) x_t + a_1(t), \\ \beta(x_t, \theta) &= b_0(t),\end{aligned}$$

where, in the time-homogeneous case, a_0 , a_1 and b_0 are constant. The SDE representation of a GP regression is then most commonly presented as [127]

$$a_0 f(t) + \sum_{i=1}^{m-1} a_i \frac{d^i}{dt^i} f(t) = \varsigma \omega_t, \quad (3.32)$$

where ω_t is a unit *white-noise process*, and variance $\varsigma \in \mathbb{R}$ along with scalar constants a_i , $i = 0, \dots, m$ are derived from the choice of kernel $k(\cdot, \cdot)$ and its hyperparameters (we note Särkkä and Solin [127] present these values for many popular GP kernels). Here, for ease of presentation, we consider only the instance of a *univariate* white-noise process (see e.g. Särkkä and Solin [127] for a more general treatment). That is, a univariate unit white-noise process is one that satisfies:

1. $\omega_{t_{i-1}} \perp\!\!\!\perp \omega_{t_i} \forall 0 \leq t_{i-1} < t_i < \infty$.
2. Is a GP with $E[\omega_t] = 0$ and *Dirac-delta-correlation* $E[\omega_{t_i}\omega_{t_j}] = \delta(t_j - t_i) \forall i, j \in \mathbb{Z}_0^+$.

In that sense, white noise is the time-differential of a Brownian motion process, such that

$$\omega_t = \frac{dw_t}{dt} \implies w_t = \int \frac{\omega_t}{dt} dt. \quad (3.33)$$

The result in (3.32) can alternatively be expressed in its *companion form* [62]

$$\begin{aligned} \frac{d\chi_t}{dt} &= F\chi_t + L\varsigma\omega_t, \\ \implies d\chi_t &= F\chi_t dt + L\varsigma\omega_t dt, \\ &= F\chi_t dt + L\varsigma dw_t, \end{aligned} \quad (3.34)$$

where

$$\chi_t = \begin{pmatrix} f(t) \\ \frac{df(t)}{dt} \\ \vdots \\ \frac{d^{m-1}f(t)}{dt^{m-1}} \end{pmatrix}, F = \begin{pmatrix} 0 & 1 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & & 0 & 1 \\ -\alpha_0 & \dots & -\alpha_{m-2} & -\alpha_{m-1} \end{pmatrix}, L = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \quad (3.35)$$

The GP can be recovered according to

$$f(t) = H\chi_t, \quad (3.36)$$

where $H = [1, 0, \dots, 0]$.

As discussed in Section 3.3.1, solutions to (3.34) may be intractable. This typically necessitates some discrete-time approximation of the likelihood. Approximating the likelihood in this way recasts the GP regression posterior inference problem as that of a SSM (3.5)

$$p(f^*|\mathcal{D}, \theta) \propto p(\chi_{t_0}) \prod_{i=1}^P p(\chi_{t_i}|\chi_{t_{i-1}}, \theta) \prod_{i \in Q} p(y_{t_i}|H\chi_{t_i}, \theta), \quad (3.37)$$

where $0 = t_0, t_1, \dots, t_{P-1}, t_P = T$ is a uniform grid of discrete times as presented in Section 3.3.1. Approximations to compute the transition densities $p(\chi_{t_i}|\chi_{t_{i-1}}, \theta)$ could then assume

the form of an iterative solver (e.g. the Euler-Maruyama scheme). However, naive implementation of the Euler-Maruyama scheme proves prohibitive when using it to approximate the solution of (3.34). That is, all but one of the state dimensions are deterministic. As such, the diffusion matrix for use in (3.13) is given as

$$\beta(\chi_{t_i}, \theta) = I_m L \varsigma = \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & 0 & 0 \\ 0 & \dots & 0 & \varsigma \end{pmatrix}, \quad (3.38)$$

which is – crucially – not positive-definite, due to the zero elements on the leading diagonal. Therefore, whilst simulation under the Euler-Maruyama scheme is possible, more complex solvers are required to overcome the numerical challenges of degenerate components in (3.37) for use in likelihood estimation. We discuss alternative techniques, specific to our intended use-case, employed to solve (3.34) in Section 6.7, chapter 6.

Chapter 4

Variational inference for discretised SDEs

In this chapter we introduce work originally presented in Ryder et al. [124], which was published at ICML in 2018. We begin by discussing the difficulties associated with SDE inference, focusing on overcoming the challenge of *bridge construct* design (see Section 4.1.1). Working with the Euler-Maruyama discretisation scheme, we then present a variational method to jointly perform SDE inference for both the parameters and latent diffusion paths. We use a standard mean-field variational approximation of the parameter posterior, and introduce a recurrent neural network to approximate the posterior for the diffusion paths conditional on the parameters. The resulting black-box variational inference method can be applied to any SDE system with light tuning requirements. We illustrate the method on a Lotka-Volterra system and an epidemic model, producing accurate parameter estimates in a few hours.

4.1 Inference for SDEs

Suppose we have $D + 1$ observations of some process $\{x_t, t \geq 0\}$, $x \in \mathbb{R}^d$, for which we would like to fit an SDE of the form

$$dx_t = \alpha(x_t, \theta)dt + \sqrt{\beta(x_t, \theta)}dw_t. \quad (4.1)$$

We recall that α is a d -dimensional *drift vector*, β is a $d \times d$ positive definite *diffusion matrix* ($\sqrt{\beta}$ is such that $\sqrt{\beta}\sqrt{\beta}^T = \beta$) and w_t is a d -vector of standard and uncorrelated Brownian motion processes. For the ease of presentation, observation times $t_{j \in Q}$ are *equally spaced*, for some index set Q of length $D + 1$. We let y_{t_j} be a vector of d_0 observations at time t_j , for

some $d_0 \leq d$. Following Golightly and Wilkinson [57], among others, we assume that

$$y_{t_j} = F'x_{t_j} + \epsilon_{t_j}, \quad \epsilon_{t_j} \sim \mathcal{N}(0, \Sigma), \quad (4.2)$$

where F is a constant $d \times d_0$ matrix, and Σ is a $d_0 \times d_0$ matrix, which may be assumed known or the object of inference. For the latter case Σ should be a specified function of θ . In this way, we flexibly permit observation for any subset of the latent process with additive Gaussian noise. Whilst we do not consider the possibility here, the extension to tractable non-Gaussian observation likelihoods and irregularly spaced times is trivial. That is, in a variational framework, given a sufficiently expressive approximating density, we are limited only by which distributions we can evaluate under – and not those from which we can sample from.

In the paradigm of Bayesian inference, we would now like to analyse the system through the posterior density by estimating

$$p(\theta, x|y) \propto p(\theta)p(x|\theta)p(y|x, \theta). \quad (4.3)$$

As detailed in section 3.3.1 – and assuming the observation model of equation (4.2) – (4.3) is tractable under the Euler-Maruyama scheme, with

$$p(x|\theta) = \prod_{i=0}^{P-1} \mathcal{N}(x_{t_{i+1}} - x_{t_i}; \alpha(x_{t_i}, \theta)\Delta t, \beta(x_{t_i}, \theta)\Delta t), \quad (4.4)$$

$$p(y|x, \theta) = \prod_{i \in Q} \mathcal{N}(y_{t_i}; F'x_{t_i}, \Sigma). \quad (4.5)$$

Monte Carlo inference algorithms could now in principle sample from (4.3). In practice, however, the high dimension and complex dependency structure can make this a difficult task. For example, suppose we only have observations of a process which occur at $t = 0, 10, 20, 30$, and utilise a discretization of $\Delta t = 0.1$ in (4.4) (indeed, this is an observation regime which we shall later consider). That is, we are presented with observations at only 4 of our 301 discrete states – or, rather, we have 297 ‘*missing*’ observations. In this way, along with the unknown parameters θ in the description of the SDE, we consider an unknown latent diffusion path x . Inference schemes must therefore overcome the subsequent dependence between parameters and missing data, requiring samples from a posterior process *in-between* available observations.

Ideally, in such a setting, one could derive a closed-form solution for the conditioned process, i.e. $p(x|y, \theta)$, requiring only a mechanism for the approximation of parameters θ . The conditioned process is, however, generally intractable (see Section 3.3.2). Therefore any Monte Carlo inference scheme additionally requires a proposal for the latent path consistent

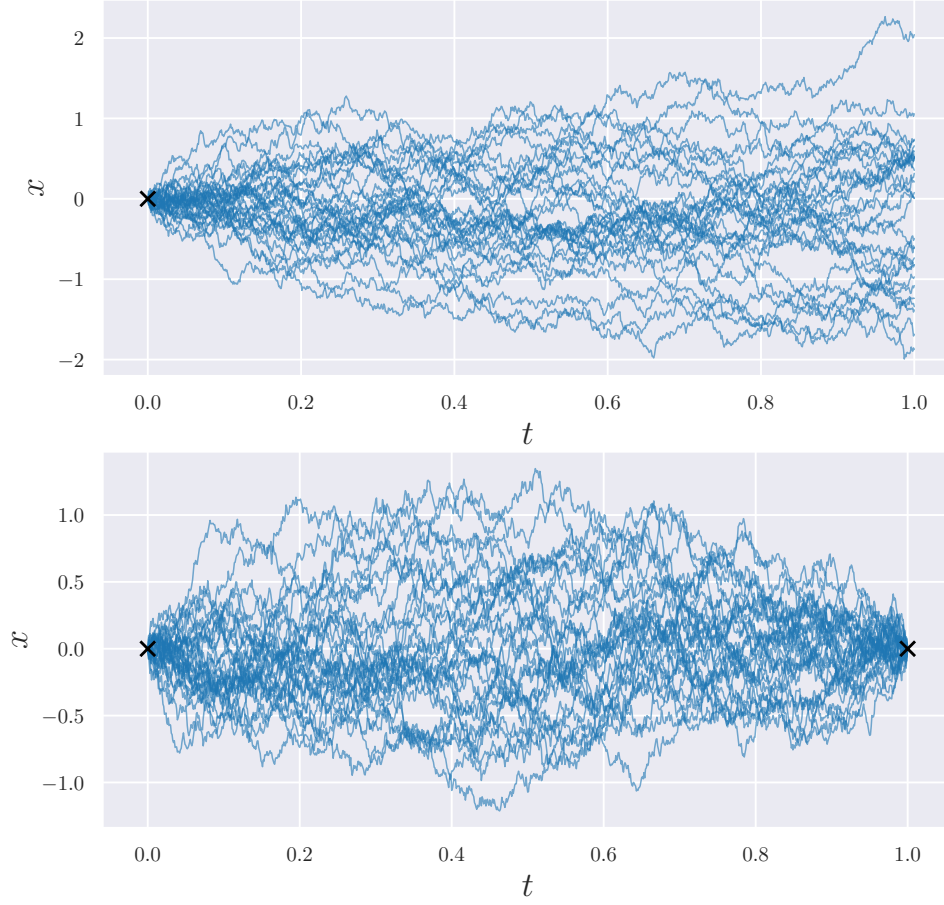


Figure 4.1 Left: 50 samples of standard Brownian motion with $\Delta t = 0.01$ from initial condition $x_{t_0} = 0$ (black cross). Right: 50 samples of a Brownian bridge between the x_{t_0} and $x_{t_N} = 1$, with $\Delta t = 0.01$.

with the data. Poor proposal mechanisms lead to inefficient Monte Carlo algorithms e.g. impractically low acceptance rates in MCMC [116].

4.1.1 Bridge constructs

A popular approach in the Monte Carlo literature on SDEs is to introduce a *bridge construct*: an approximation to the discretized diffusion process conditional on the parameters and observations (i.e. approximations of equations (3.15) and (3.16), Section 3.3.2), derived using probability theory and various simplifying approximations. The goal is to produce a path ‘bridging’ between a pair of observation times.

Combining successive bridges for multiple observation times will form a complete diffusion path. These bridge constructs can then be used to produce proposals within Monte Carlo

algorithms (such as MCMC – see e.g. Roberts and Stramer [116], Golightly and Wilkinson [57], Fuchs [47], van der Meulen et al. [137]) for the latent process. However, designing a bridge construct with desirable features for a particular problem is a challenging and time-consuming tuning task (some particularly difficult regimes for bridge constructs are discussed in Section 4.3).

To illustrate this challenge more clearly, we refer to an example for which the conditioned process *is* analytically tractable, i.e. Brownian motion. Conditioning Brownian motion on a future observation, given some initial condition, is known as a *Brownian bridge* (see Appendix B.1 for a derivation). We present an example Brownian bridge, using an exactly observed future state, against unconditioned Brownian motion in Figure 4.1. In spite of its tractability, it serves to highlight the objective of an approximating bridge construct for use in e.g. MCMC.

4.1.2 Feed-forward neural networks

Much like hand designing a bridge construct, this and later chapters requires the high-fidelity approximations of unknown functions. The most natural way to approach this problem is with the use of *neural networks*. Whilst a rigorous treatment of neural networks is beyond the scope of this thesis, it is nonetheless instructive to briefly review their mathematical formulation (for an extended introduction see e.g. Goodfellow et al. [59]).

In its simplest form, a *feed-forward* neural network defines some $f : \mathbb{R}^d \rightarrow \mathbb{R}^n$. The network f is built of l non-linear (or *hidden*) layers. Each layer is composed of an affine transformation and a non-linearity (or a so-called *activation* function).

We begin by defining the output of our input (or first) layer as

$$z^1 = g(W^1 x + b^1), \quad (4.6)$$

where $x \in \mathbb{R}^{d \times 1}$ is the input, z^1 is the first *feature map*, $W^1 \in \mathbb{R}^{w_1 \times d}$ are the *weights*, $b^1 \in \mathbb{R}^{w_1 \times 1}$ are the biases, g is the activation function and w_1 is the *width* of our first layer. The most common activation function – and the one we use in our experiments – is the *rectified linear unit* (ReLU). That is, the ReLU function defines $g(\cdot)$ such that

$$g(z) = \max(0, z). \quad (4.7)$$

We can then define the resulting i^{th} hidden layers according to

$$z^{i+1} = g(W^i z^i + b^i), \quad (4.8)$$

such that $W^i \in \mathbb{R}^{w_i \times w_{i-1}}$, $b^i \in \mathbb{R}^{w_i \times 1}$ and w_i is the *width* of layer i .

The final layer can then be defined according to

$$y = g(W^l z^l + b^l), \quad (4.9)$$

where $W^l \in \mathbb{R}^{n \times w_{l-1}}$, $b^l \in \mathbb{R}^{n \times 1}$ and – assuming a real-valued output – g is taken to be the identity function.

In order to approximate a function with a neural network, we would typically use gradient descent (or variations) to update the so-called collection of *learnable* parameters $\phi = (W^i, b^i)$ at each layer. Assuming differentiability of some objective function (e.g. a Gaussian likelihood), this is achieved by taking gradients of the parameters with respect to the objective by recursively applying the chain rule *backwards* through the layers (i.e. in the opposite direction of the feed-forward computation). This process is otherwise known as *backpropagation* (e.g. Hecht-Nielsen [66]). In chaining successive non-linear transformations as of the kind in (4.6), (4.8) and (4.9), it is theoretically possible to approximate any function of arbitrary complexity. This result is known as the *universal approximation theorem* [33].

4.2 Variational inference for SDEs

We now propose a method to circumvent some of the challenges posed by sampling based SDE inference. Our approach uses recent advances in variational inference to jointly infer θ and – most critically – x , the conditioned diffusion process. In developing a family of highly expressive variational densities, we are able to automate the time-consuming (and often inefficient) task of bridge construct design and provide significant time advantages over existing MCMC methods (see Section 4.2.1). In contrast to our approach, MCMC methods require more tuning choices and can take days to run [147]. Our resulting black-box variational inference method is a simple and fast way to produce approximate inference for some SDE systems.

4.2.1 Related work

Several authors have looked at variational inference for SDEs [10] or related problems such as Markov jump processes [122] and state space models [11]. Archambeau et al. [10] formulates an approach for univariate SDEs with a constant diffusion matrix, and bases optimisation on variational calculus, eschewing a discretization scheme. Other approaches have considered a variational interpretation of the Fokker-Planck equation [75, 15]. Here Batz et al. [15] uses

variational (and kernel) methods to non-parametrically learn a drift function, as is similarly achieved in Duncker et al. [38] and Archambeau et al. [9] using Gaussian processes.

The novelty of our approach is to use: (1) stochastic optimisation rather than variational calculus; (2) a RNN-based variational approximation for the latent states instead of a mean-field one.

Our method can also be thought of as related to recent work on *normalising flows* (see Section 6.2 for an introduction) as variational approximations [111]. As in that work, our variational approximation can be viewed as transforming a $\mathcal{N}(0, I)$ sample vector by successive linear transformations to an approximate posterior sample (of the diffusion states in our case). Our work uses SDE theory to select simple and cheap transformations which produce a particularly good approximation.

Another Monte Carlo approach, separate from bridge constructs, is to perform approximate inference based on low dimensional summary statistics of the observations [105]. This results in a loss of information, which our approach avoids.

In some simple situations a discretized approximation of the conditioned process can be derived (see e.g. Papaspiliopoulos et al. [103]). Here, unlike the exact solution (Section 3.3.2), the diffusion matrix is scaled down as the observation time is approached. Intuitively this is appealing: conditioned paths converge towards the observation, so nearby random deviations are smaller in scale. Indeed, if the diffusion matrix remained unaltered, the approximation error induced by discretization would often require the use of a numerically infeasible step-size near the observations. This motivates us to use a variational approximation in which the diffusion matrix is not constrained to follow (3.16), and instead is allowed to shrink.

4.2.2 Approximate posterior density

We begin by introducing our variational approximation to the posterior (4.3), which we factorise as

$$q(\theta, x; \phi) = q(\theta; \phi_\theta)q(x|\theta; \phi_x). \quad (4.10)$$

These factors represent approximations to $p(\theta|y)$ and $p(x|\theta, y)$ respectively, which are described below. Here ϕ_θ and ϕ_x are the variational parameters for the two factors, and ϕ is the collection of all variational parameters.

Approximate parameter posterior

As earlier introduced in the linear regression example of Section 2.4.1, we use the mean-field Gaussian approximation for θ , such that

$$q(\theta; \phi_\theta) = \prod_{i=1}^c \mathcal{N}(\theta_i; \mu_i, \sigma_i^2), \quad (4.11)$$

where $\phi_\theta = (\mu_1, \dots, \mu_c, \sigma_1, \dots, \sigma_c)$ and $\theta = (\theta_1, \dots, \theta_c)$. Recall that to express θ using this reparameterization trick, we define

$$\theta = g_\theta(\epsilon_1, \phi_\theta) = S\epsilon_1 + \mu, \quad (4.12)$$

where $\epsilon_1 \sim N(0, I_c)$, $S = \text{diag}(s_1, \dots, s_c)$ and $\mu = (\mu_1, \dots, \mu_c)$. It may be additionally necessary to transform θ to an alternative parametrisation ϑ so that a Gaussian approximation is appropriate e.g. log-transforming parameters constrained to be positive.

Mean-field approximations, as previously detailed, are imperfect, often producing under-dispersed estimates of the posterior (see e.g. Blei et al. [23]). More sophisticated approximations could be used here instead, e.g. normalising flows [111]. Indeed, we often use normalizing flows for approximate parameter posteriors in later chapters. However, mean-field approximations suffice to give good parameter estimation in our examples, with this not considered the unique aspect of our work.

Approximating the conditioned diffusion process

Motivated by the result that a diffusion process conditioned on an exact observation is itself an SDE (see Section 3.3.2), we base $q(\theta; \phi_x)$ upon a discretised diffusion. In addition, as discussed in Section 4.2.1, we note that modification of the diffusion term is helpful when working with discretised approximations. We therefore, unlike the exact solution, permit modifications to the diffusion. A generative definition is

$$x_{t_{i+1}} = h\left(x_{t_i} + \tilde{\alpha}(x_{t_i}, y, \theta, t_i; \phi_x)\Delta t + \sqrt{\tilde{\beta}(x_{t_i}, y, \theta, t_i; \phi_x)\Delta t}z_{i+1}\right), \quad (4.13)$$

where $\tilde{\alpha}$ and $\tilde{\beta}$ are approximations to the drift and diffusion functions. Taking h as the identity function yields a discretised diffusion process. It is often the case, however, that some components of x are non-negative (e.g. in population dynamics); therefore, we often use h to impose positivity constraints on x – see Section 4.2.3.

The resulting variational density $q(x|\theta; \phi_x)$ is given by

$$\prod_{i=0}^{P-1} \mathcal{N}\left(x_{t_{i+1}} - x_{t_i}; \tilde{\alpha}(x_{t_i}, y, \theta, t_i; \phi_x) \Delta t, \tilde{\beta}(x_{t_i}, y, \theta, t_i; \phi_x) \Delta t\right) |\det J_i|, \quad (4.14)$$

where J_i is the Jacobian matrix associated with the transformation h in (4.13). To express x for purposes of the reparameterization gradient, let $\epsilon_2 \sim \mathcal{N}(0, I_{dP})$ be the flattened vector of $(z_1, z_2, \dots, z_P)$ realisations. Then apply (4.13) repeatedly. Let the outcome be represented by the function

$$x = g_x(\epsilon_2, \theta, \phi_x). \quad (4.15)$$

We use a vanilla, feed-forward neural network, with parameters ϕ_x , to serve as our functions $\tilde{\alpha}$ and $\tilde{\beta}$. At time t_i it acts as follows. The network's input, in addition to x_{t_i} , is composed of several features computed from: the current diffusion state x_{t_i} , the observations y , the parameters θ and the current time t_i . The full list of features, which we represent using ϱ , is described in full detail within Section 4.2.3. At each step, the network outputs a drift vector and diffusion matrix (see Section 4.2.3 for details of the latter), which are used to sample $x_{t_{i+1}}$ from (4.13). This state forms part of the neural network input at time t_{i+1} . The network then forms a *cell* of an overall recurrent neural network (RNN) structure for $q(x|\theta; \phi_x)$ (see e.g. Goodfellow et al. [59] for a thorough introductory treatment of RNNs). A diagram illustrating the generative process within a RNN cell for x_{t_i} is presented in Figure 4.2. Note that long-term memory features are not required as we only wish to approximate a diffusion process, which is memoryless.

4.2.3 Algorithm and implementation details

Ensuring positivity

In practice, we take h in equation (4.13) to be a function which applies the identity function to unconstrained components of the diffusion state, and the function $\text{softplus}(z) = \log(1 + e^z)$ to components with positivity constraints. This function produces strictly positive outputs while having little effect on positive inputs above 2. The latter property ensures our variational approximation remains similar to a discretised diffusion process for most positive values. Alternative transformations may be used should the state values commonly fall below 2. Transformations could then be administered with additional tuning parameters, allowing for suitable function shapes to be learned. This procedure, however, was not necessary for our examples.

Preliminary work found this transformation approach to enforcing positivity was much easier to implement in the variational framework than the reflection methods commonly used in

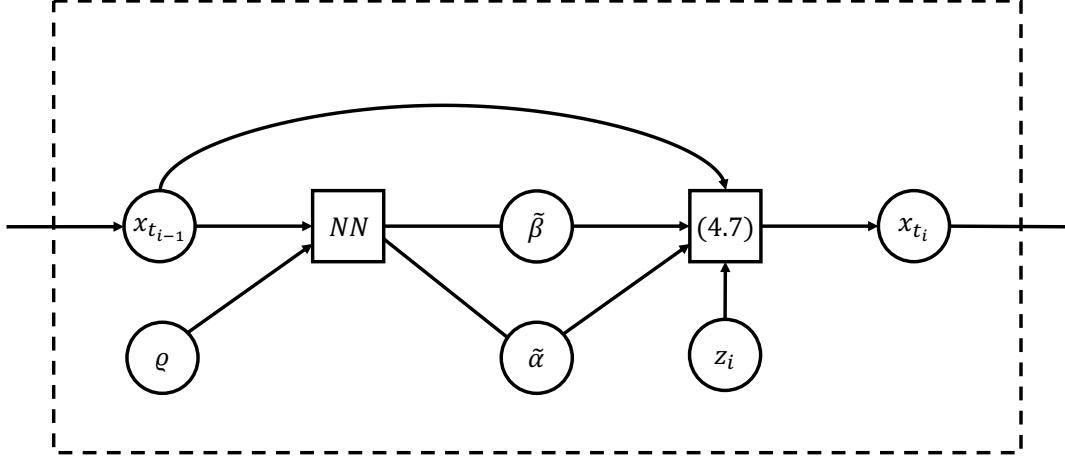


Figure 4.2 Illustration of the generative process and data flow within the proposed RNN cell. The dashed line encapsulates the cell, circles represent data nodes and squares represent functions. Note: NN represents the neural network.

Monte Carlo schemes. In a sense, this transformation can be interpreted as constraining the variational approximation based on prior beliefs about the non-negativity of diffusion paths.

Algorithm

Our overall inference algorithm is given in Algorithm 2. This aims to maximise the ELBO (equation (2.35), Section 2.4)

$$\mathcal{L}(\phi) = \mathbb{E}_{\theta, x \sim q(\cdot; \phi)} \left[\log \frac{p(\theta)p(x|\theta)p(y|x, \theta)}{q(\theta; \phi_\theta)q(x|\theta; \phi_x)} \right], \quad (4.16)$$

using the Monte Carlo gradient estimate (equation (2.46), Section 2.4.1)

$$\begin{aligned} \widetilde{\nabla_\phi \mathcal{C}}(\phi) &= -\widetilde{\nabla_\phi \mathcal{L}}(\phi), \\ &= -\frac{1}{N} \sum_{i=1}^N \left[\nabla_\phi \log \left\{ \frac{p(\theta^{(i)})p(x^{(i)}|\theta^{(i)})p(y|x^{(i)}, \theta^{(i)})}{q(\theta^{(i)}; \phi_\theta)q(x^{(i)}|\theta^{(i)}; \phi_x)} \right\} \right], \end{aligned} \quad (4.17)$$

where $\theta^{(i)} = g_\theta(\epsilon_1^{(i)}, \phi_\theta)$ and $x^{(i)} = g_x(\epsilon_2^{(i)}, \theta^{(i)}, \phi_x)$.

Algorithm 2 Black-box variational inference for SDEs

Initialise ϕ_0 and $k = 0$.

loop

Sample $\epsilon_1^{(i)}, \epsilon_2^{(i)}$ for $1 \leq i \leq N$.

Calculate $\widehat{\nabla_{\phi} \mathcal{C}}(\phi_k)$ using (4.17).

Calculate ϕ_{k+1} using stochastic gradient descent, or a similar algorithm, and increment k .

end loop

Implementation details

Given the current time t_i , the nearest future t_j and past observations times t_{j-1} , the RNN cell input is as follows:

- The parameters θ .
- The most recent latent state, $x_{t_{i-1}}$.
- The time until the next observation, $t_j - t_i$.
- The next observation time, t_j .
- The difference between the next observation and what the mean observation would be at the most recent latent state, $y_{t_j} - F'x_{t_{i-1}}$.

Exploratory work demonstrated that the RNN produced a much better approximation of the conditioned process using an augmented feature space as input – rather than simply x_{t_i}, y, θ and t_i .

Our RNN cell outputs a vector $\tilde{\alpha}$ and the coefficients of a lower-triangular matrix, M . We use M to represent the lower-triangular Cholesky decomposition of $\tilde{\beta}$. In this way, we ensure $MM^T = \tilde{\beta}$ is a positive semidefinite covariance matrix. In order to ensure M represents a valid Cholesky factor of $\tilde{\beta}$, the diagonal elements are transformed using the softplus function as to ensure positivity. We also regularise to avoid $\tilde{\beta}$ matrices with very small determinants, which could induce numerical instability when performing log-density calculations.

Algorithm 2 requires automatic differentiation of (4.17). This can be achieved using the standard tool-kit of backpropagation after *rolling-out* the RNN i.e. stacking m copies of the RNN cell to form a deep feed-forward network. (In practice, this approach is inefficient for long time series, and we discuss alternative procedures in chapter 6). The canonical challenge in training such networks, known as the exploding-gradient problem (17), often necessitates the use of gradient clipping to control for numerical instability. We follow Pascanu et al. [104] and perform gradient clipping using the L_1 norm.

To initialise ϕ_0 in Algorithm 2, we select ϕ_θ so that the margins of the variational approximation match those of the parameter priors. Standard choices from the neural network literature – random Gaussian weights and constant biases – are used for ϕ_x .

4.3 Experiments

We implement our method for two examples: (1) analysing synthetic data from a Lotka-Volterra SDE; (2) analysing real data from an SDE model of a susceptible-infectious-removed (SIR) epidemic. Our experiments include challenging regimes such as: (A) low-variance observations; (B) conditioned diffusions with non-linear dynamics; (C) unobserved time series; (D) widely spaced observation times; (E) data which is highly unlikely under the unconditioned model. Many of these violate the assumptions used by existing diffusion bridge constructs [148].

In all of the following experiments similar tuning choices worked well. We use a batch size of $N = 50$ in (4.17). Our RNN cell has four hidden layers each with 20 hidden units and rectified-linear activation. We implement our algorithms in Tensorflow using the Adam optimiser [78] and report results using an 8-core CPU.

We further use post-hoc importance sampling as a means to validate our approximate posteriors. In practice we will use our converged variational approximation as the importance density, and the ESS to assess whether this is sufficiently good to produce accurate estimates. That is, we calculate our importance weights such that

$$w_i = \frac{p(y|x^{(i)}) p(x^{(i)}|\theta^{(i)}) p(\theta^{(i)})}{q(\theta^{(i)}) q(x^{(i)}|\theta^{(i)})},$$

and ESS as-per equation (2.24). However, ESS values can be unstable for poor importance densities [113] so later we also consider other problem-specific evidence for the quality of our results

4.3.1 Lotka-Volterra

Lotka-Volterra models describe simple predator-prey population dynamics combining three types of event: prey reproduction, predation (in which prey are consumed and predators have the resources to reproduce) and predator death. Given careful parameter choice, Lotka-Volterra models can produce their canonical sinusoidal population dynamics (e.g. Figure 4.4). A SDE Lotka-Volterra model (for a derivation see e.g. Golightly and Wilkinson [58]) is

defined by

$$\alpha(x_t, \theta) = \begin{pmatrix} \theta_1 u_t - \theta_2 u_t v_t \\ \theta_2 u_t v_t - \theta_3 v_t \end{pmatrix}, \quad (4.18)$$

$$\beta(x_t, \theta) = \begin{pmatrix} \theta_1 u_t + \theta_2 u_t v_t & -\theta_2 u_t v_t \\ -\theta_2 u_t v_t & \theta_3 v_t + \theta_2 u_t v_t \end{pmatrix}, \quad (4.19)$$

where $x_t = (u_t, v_t)'$ represents the populations of prey and predators at time t . The parameters $\theta = (\theta_1, \theta_2, \theta_3)'$ control the rates of the three events described above.

Throughout the experiments which follow, we use a discretisation time step of $\Delta t = 0.1$ and an observation variance of $\Sigma = I$, which is small relative to the typical population sizes (see e.g. Figure 4.3). This observation scheme represents an additional challenge as it is typically more difficult to find diffusion bridges in low variance regimes, relative to the magnitude of x .

A single observation time with known parameters We begin with the case of a single observation time and known parameter values, where we follow Boys et al. [25] by taking $\theta = (0.5, 0.0025, 0.3)'$ and $x_0 = (71, 79)'$. This setting solely investigates our ability to learn x , without uncertainty in θ . In essence, this is the same problem as devising a bridge construct (described in Section 4.2.1).

We consider four different observations at time $t = 10$, listed in Table 4.1. For each example we train our variational approximation until convergence (assessed manually throughout this chapter), which takes roughly 20 minutes for the first 3 examples, and 90 minutes for the last. We then perform importance sampling using 500,000 samples from the fitted approximation. Table 4.1 shows the resulting ESS values. The first 3 rows in the table are typical observations under the model given our θ , while the final row represents highly unlikely observations (double those in the previous row). Figure 4.3 shows fitted diffusion paths for this case.

Table 4.1 Summary of importance sampling performance for the Lotka-Volterra example with a single observation time and known θ . Each row shows the prey (U_{10}) and predator (V_{10}) observations at time $t = 10$. For each, 500,000 iterations of importance sampling are performed, using variational inference output as the importance density, and the effective sample size (2.24) is shown.

U_{10}	V_{10}	ESS
15.3	298.2	184,329
46.7	389.1	212,313
108.7	503.4	196,956
217.4	1006.9	95,711

This example contains several challenging features: all those listed at the start of Section 4.3 except (C). While these features make it hard to use existing bridge constructs, our

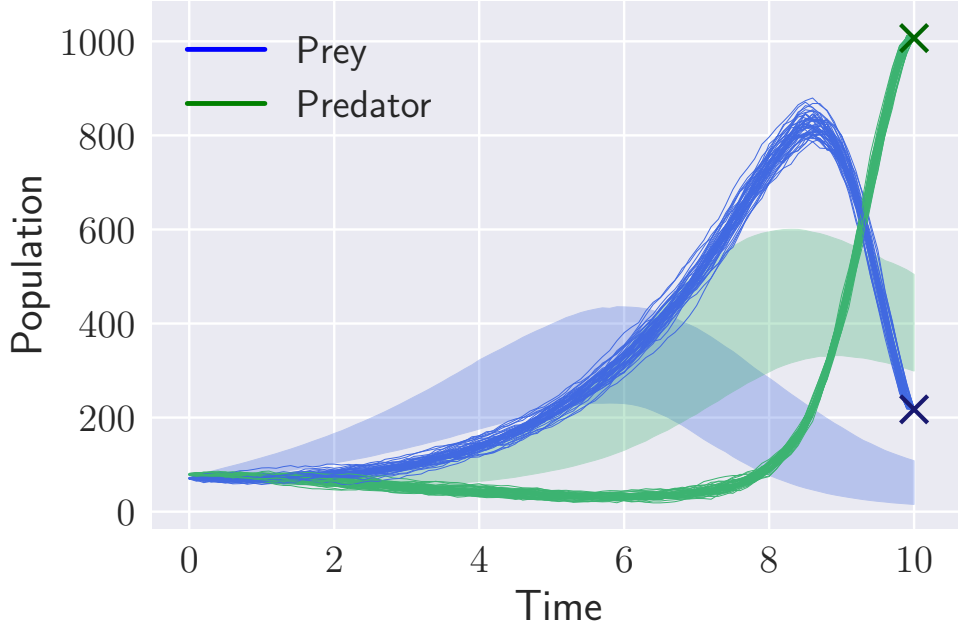


Figure 4.3 50 conditioned diffusion paths for the Lotka-Volterra example with a single observation time and known parameters. These are sampled from the trained variational approximation. The observations used are those in the last row of Table 4.1, representing highly unlikely observations. The shaded regions show typical paths of the unconditioned process (95% intervals estimated from repeated simulations.)

variational method produces a close approximation to the true posterior, as illustrated by the high ESS values.

The case of highly unlikely observations takes longer to train and receives a lower ESS, reflecting the presence of a more complicated diffusion path. To informally validate this, we found that a simpler RNN cell architecture suffices (less layers and fewer hidden units) for good performance in the other examples, but not this one.

Multiple observation times with known parameters We now extend the previous example to multiple observation times, $t = 0, 10, 20, 30, 40$. We analyse synthetic data, produced using the parameters specified previously (including observation noise with $\Sigma = I$). Here convergence takes 6 hours, and importance sampling with 500,000 samples produces an ESS of 96,812. The resulting diffusion paths, for sake of repetition, are not shown as they are very similar visually to the next example (see Figure 4.4).

This example illustrates that our method learns the conditioned process well even when there are several observation times.

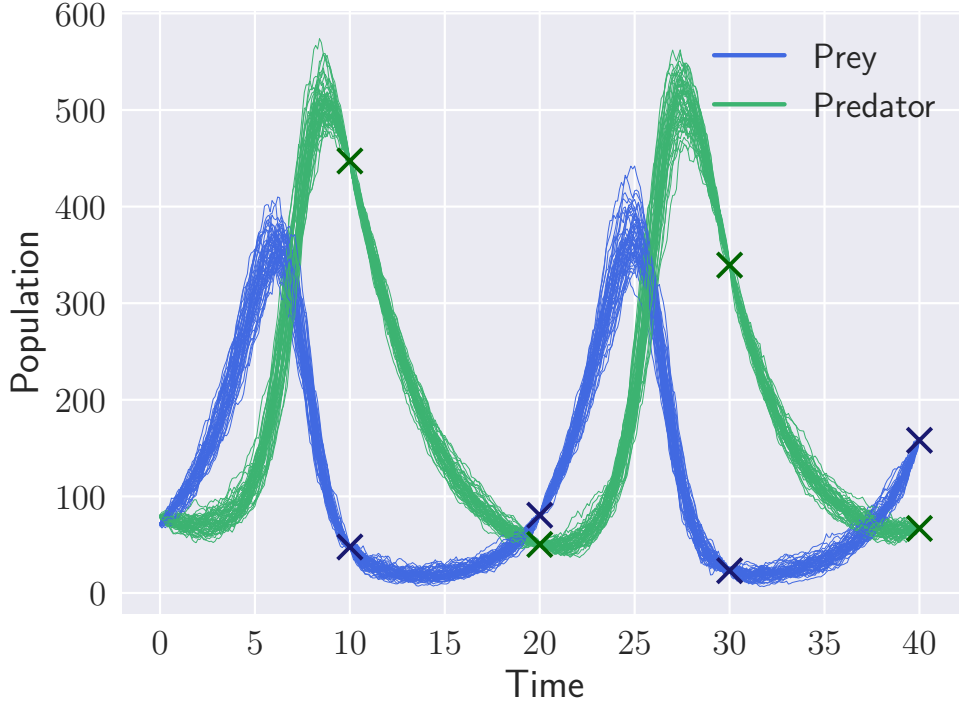


Figure 4.4 50 conditioned diffusion paths for the Lotka-Volterra example with multiple observation times and unknown parameters. They are sampled from the fitted variational approximation.

Multiple observation times with unknown parameters We now analyse the same synthetic data with unknown θ parameters. As these parameters must take positive values, we work with the log-transformed parameters ϑ and assume that these have independent $\mathcal{N}(0, 3^2)$ priors. Our results are shown after transforming back to the original parameterisation.

Convergence takes 2 hours, and importance sampling with 500,000 iterations produces an ESS of 635.4. Figure 4.4 shows 50 diffusion paths sampled from the fitted variational approximation. Figure 4.5 shows two estimates of the marginal parameter posteriors: the variational inference output, and a kernel density estimate based on importance sampling results.

The estimated posteriors give accurate estimates of the true parameter values. However, the low ESS here shows that both estimates of the parameter posteriors are imperfect approximations (also illustrated by the variational posterior estimates appearing overconcentrated compared to the importance sampling results.) Achieving good point estimates but imperfect posteriors is typical for variational inference [23].

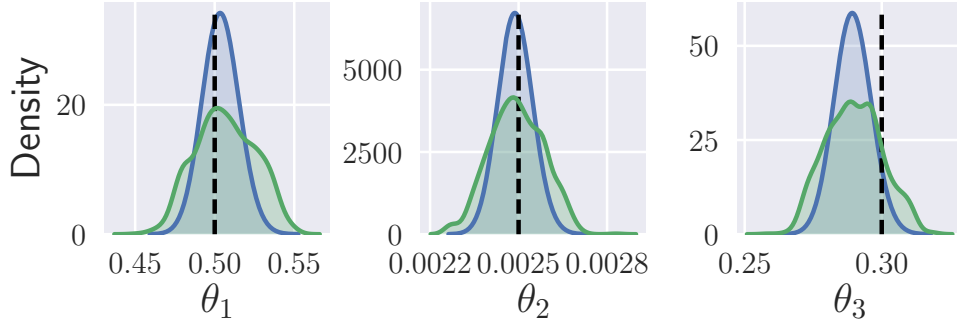


Figure 4.5 Marginal posterior density estimates for the Lotka-Volterra example from variational inference (blue) and importance sampling (green). The black dashed line represents the true parameter values.

4.3.2 Epidemic model

An SIR epidemic model [7] describes the spread of an infectious disease. The population is subdivided into those susceptible (s), infectious (i) and removed (r). Two types of event occur: susceptibles can be infected by the infectious, and the infectious eventually become removed (either through death or recovery). We assume a constant population size, implying that only the s and i population sizes need to be modelled.

An SIR epidemic model using SDEs is defined by

$$\alpha(x_t, \theta) = \begin{pmatrix} -\theta_1 s_t i_t \\ \theta_1 s_t i_t - \theta_2 i_t \end{pmatrix}, \quad (4.20)$$

$$\beta(x_t, \theta) = \begin{pmatrix} \theta_1 s_t i_t & -\theta_1 s_t i_t \\ -\theta_1 s_t i_t & \theta_1 s_t i_t + \theta_2 i_t \end{pmatrix}, \quad (4.21)$$

where $x_t = (s_t, i_t)'$ is the state of the system at time t , θ_1 is an infection parameter and θ_2 is a removal parameter. For a detailed derivation see Fuchs [47].

Our data is taken from an outbreak of influenza at a boys boarding school in 1978 [71]. Influenza was introduced to the population by a student returning from holiday in Hong Kong. Of the 763 boys at the school, 512 were infected within 14 days. We hence assume $x_0 = (762, 1)'$ is known and fixed. Observations for the number infectious are provided daily by those students confined to bed. We assume Gaussian observation error with unknown variance σ^2 . Our analyses again use a discretisation time step of $\Delta t = 0.1$. To demonstrate our method on a more complex extension, we also consider an alternative model with a time-varying infection parameter. Here we let $\vartheta_1 = \log \theta_1$ follow an Ornstein-Uhlenbeck process

$$d\vartheta_{t,1} = \theta_3 (\theta_4 - \vartheta_{t,1}) dt + \theta_5 dw_t, \quad (4.22)$$

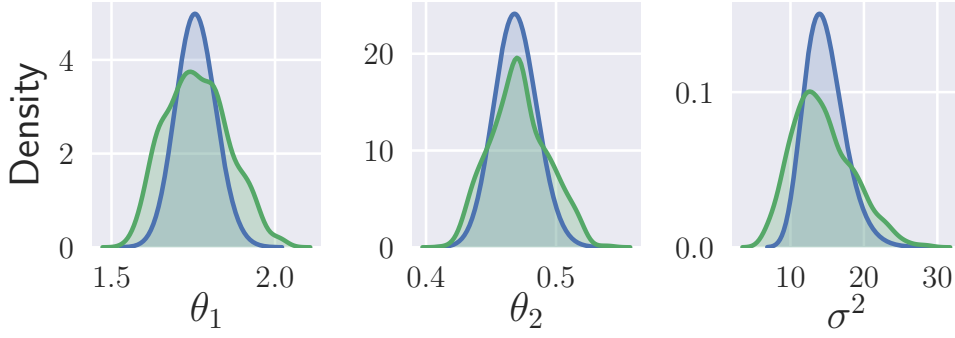


Figure 4.6 Marginal posterior density estimates from variational inference (blue) and importance sampling (green) for the SIR epidemic model with constant θ_1 .

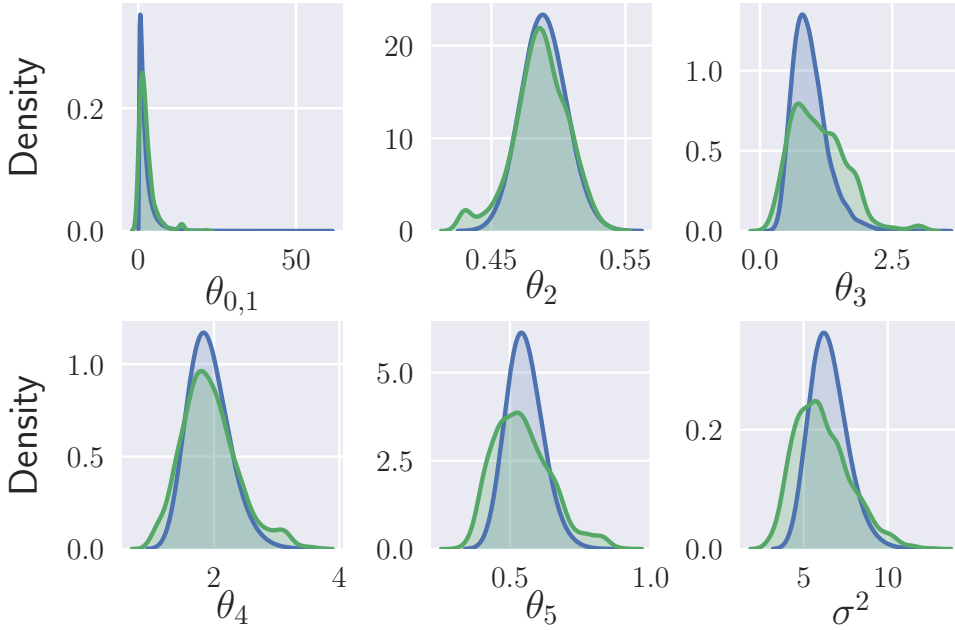


Figure 4.7 Marginal posterior density estimates from variational inference (blue) and importance sampling (green) for the SIR epidemic model with time varying θ_1 .

where θ_3 , θ_4 and θ_5 are the mean-reversion rate, process mean and volatility, respectively, and $\theta_{t,1}$ is the infection parameter at time t . Previous related work has focused on ODE epidemic models with time-varying parameters following SDEs [39, 34]. In contrast, our approach can easily be applied to a fully stochastic system.

Time-invariant infection parameter We infer the log-transformed parameters $\vartheta = (\log \theta_1, \log \theta_2, \log \sigma^2)'$ under independent $\mathcal{N}(0, 3^2)$ priors. Our results are shown after transforming back to the original parameterisation.

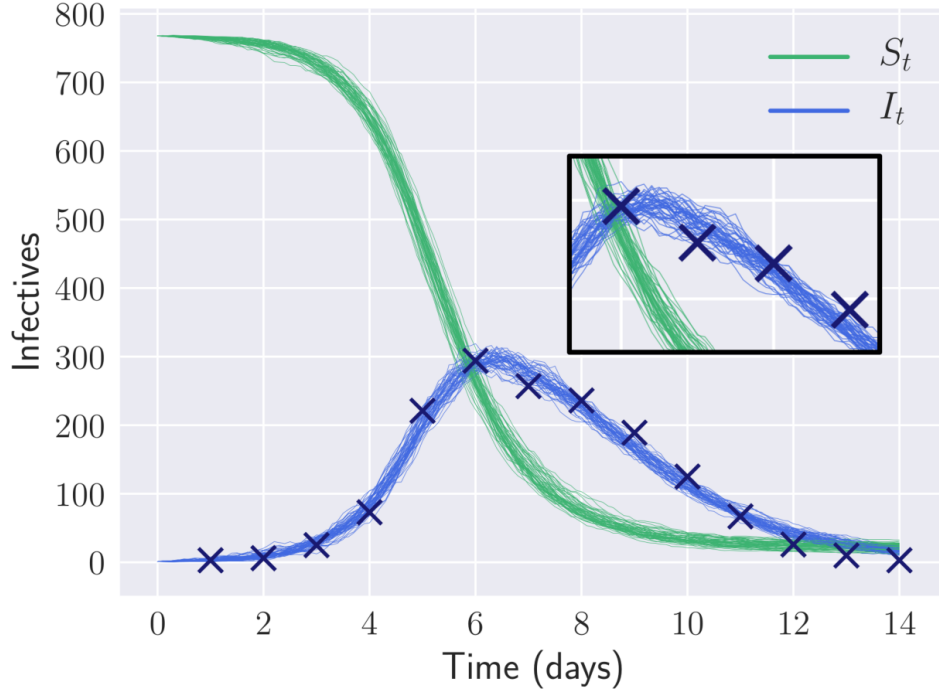


Figure 4.8 50 conditioned diffusion paths for the SIR epidemic model with constant θ_1 . These are sampled from the trained variational approximation. The observations are represented by dark blue crosses. The inset is intended to highlight the visual differences between the latent diffusion paths for the time-variant and time-invariant models (see Figure 4.9 for the former).

Convergence takes 2.5 hours, and importance sampling with 500,000 iterations produces an ESS of 718.2. Figure 4.6 shows two estimates of the marginal parameter posteriors: variational inference output, and a kernel density estimate based on importance sampling results. Figure 4.8 shows 50 diffusion paths sampled from the variational approximation.

The small ESS indicates the presence of some approximation error. However, the marginal parameter posteriors for θ_1 and θ_2 are very similar to those from the MCMC analysis of Fuchs [47, pg 293], despite some modelling differences (that analysis fixed $\sigma^2 = 0$ and used exponential priors for θ_1 and θ_2).

Time-variant infection parameter Here we infer the log-transformed parameters $\vartheta = (\log \theta_{0,1}, \log \theta_2, \log \theta_3, \log \theta_4, \log \theta_5, \log \sigma^2)'$ under independent $\mathcal{N}(0, 3^2)$ priors. Our results are shown after transforming back to the original parameterisation.

Convergence now takes 3 hours, and 500,000 iterations of importance sampling produces an ESS of 256.1. Figure 4.7 shows estimates of the marginal parameter posteriors, using variational inference and importance sampling outputs as before. Figures 4.9 (SIR) and 4.10

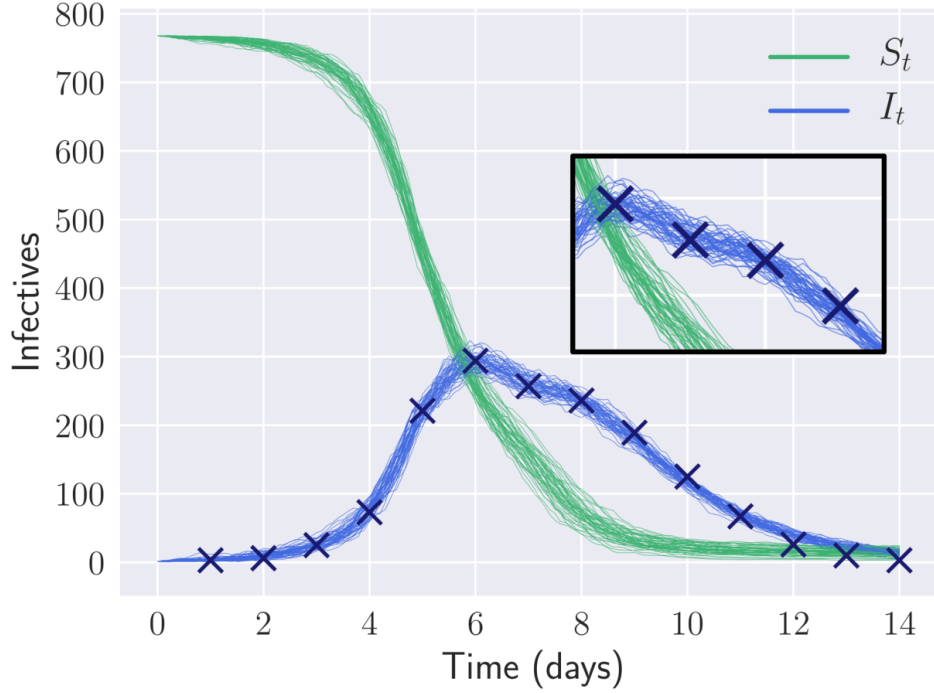


Figure 4.9 50 conditioned diffusion paths for the SIR epidemic model with time varying θ_1 . These are sampled from the trained variational approximation. The observations are represented by dark blue crosses. The inset is intended to highlight the visual differences between the latent diffusion paths for the time-variant and time-invariant models (see Figure 4.8 for the latter).

(Ornstein-Uhlenbeck) show 50 diffusion paths sampled from the variational approximation. Again the low ESS indicates some approximation error.

Model comparison The two models are seen to produce visually similar diffusion paths, but close inspection shows some differences. The time-invariant model paths for i_t appear smooth but slightly miss some of the observation points. The time-variant model paths for i_t are less smooth and more accurately capture the shape of the data. Correspondingly, the time-varying model infers a smaller σ^2 value. The most obvious difference in i_t paths occurs for the $t = 7, 8, 9$ observations, shown in the zoomed-in inset of Figures 4.8 and 4.9. Figure 4.10 shows that shortly before this the time-variant θ_1 values become constrained to smaller values.

Although the time-varying model appears to fit the data better, this is at the cost of increased model complexity, and could simply reflect *overfitting*. A better estimate of the parameter posteriors would allow formal model comparison based on importance sampling evidence estimates.

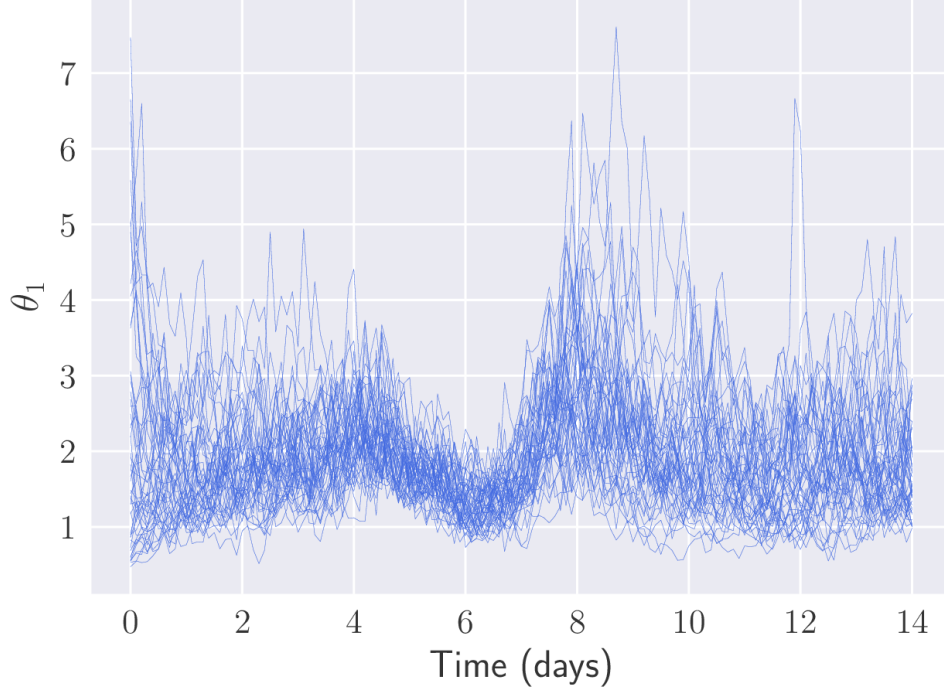


Figure 4.10 50 conditioned diffusion paths for θ_1 for the SIR epidemic model. These are sampled from the trained variational approximation.

To test the overall fit of the models we compared evidence estimates produced from our importance sampling output. (Recall that, as described in Section 2.3.1, $N^{-1} \sum_{i=1}^N w_i$ is an unbiased and consistent estimate of the model evidence.) Our evidence estimates were 2.3×10^{-33} (time-varying), and 2.4×10^{-33} (time-invariant), giving a *Bayes factor* (i.e. a ratio of comparing model evidences) of 1.04. This formal comparison shows there is very little to choose between these two models, and suggests it is plausible that the infection rate is constant over time.

4.4 Conclusion

In this chapter we have presented a black-box variational approach to inference for SDEs. Relative to existing methods, this approach is simple, practical and fast. We have successfully performed inference for a broad class of SDEs with minimal tuning requirements, automating the often challenging task of bridge construct design. Our empirical investigation shows we obtain close matches to the posterior of the conditioned diffusion paths. Furthermore, Ward and Álvarez [144] demonstrate our approximation is capable of producing conditioned paths statistically indistinguishable from the true posterior for a tractable SDE model. We

additionally demonstrate the possibility of approximate parameter inference (even with a simple choice of approximate posterior) with our results recovering known parameters for synthetic data (Section 5.1), and previous results for real data (Section 4.3.1), using only a few hours of computation on a desktop CPU.

Despite the method’s utility, there are two clear areas of future work that we will now explore: speed and accuracy. Broadly, these improvements can be surmised as follows:

Speed Whilst fast compared to existing approaches, RNNs are neither particularly amenable to parallelism or GPU computation. As such, application to a large m would, under the current approach, prove infeasibly time-consuming.

Accuracy In using a discretised SDE, we introduced approximation error to our posterior. The resulting approach then yields an approximation of an approximation – one that, of course, we can reduce with a smaller Δt , but not without incurring a large computational cost. Furthermore, unlike the true conditioned process (Section 3.3.2), we learn a diffusion matrix in our variational posterior. In an appeal to the underlying theory, we would ideally like to use the original diffusion matrix in equation (4.13). Further, this would result in avoiding learning a diffusion matrix, whose number of terms scales quadratically with $\dim(x)$.

To this end, our work now bifurcates into developing improvements in both of these key areas. In Chapter 6, we will consider the application of alternative generation techniques for high-dimensional distributions, developing a faster, more parallelizable algorithm better suited to big data. In Chapter 5, we present a more direct extension to this work, developing an approach that avoids discretizing the posterior directly. Here we further explore the possibility of avoiding a modified diffusion matrix (as per the theory of Section 3.3.2), along with a *transfer learning* technique to obtain an approximation to the limit of $\Delta t \rightarrow 0$.

Chapter 5

Continuous-time methods

5.1 Limits of SDE inference in discrete time

As similarly examined in chapter 4, we wish to perform Bayesian inference for SDEs by inferring the joint density of the parameters and latent diffusion paths, θ and x . Using the tools of variational inference, chapter 4 outlined an approach to achieve this. In that work we selected a uniform discretisation scheme and then performed inference for the discretised SDE. In presenting this approach, however, we recognised the limitations of working in discrete time; in turn, we saw the possibility for improvements to both speed and approximation accuracy. Chapter 6 demonstrates advancements in algorithmic speed, and here we focus on improvements to *accuracy*. More specifically, we direct our efforts toward mitigating various challenges introduced when approximating the SDE on a discrete-grid of uniform times. These challenges are discussed in more detail as follows:

Unevenly spaced observation times Use of a uniform discretisation scheme could prove prohibitive when working with unevenly spaced observation times. That is, the approach of chapter 4 must discretise with a uniform grid that includes all observations. Achieving this could require selecting a very small step-size. Use of a small step-size – and therefore a large number of recurrent neural network (RNN) cells – could make backpropagation extremely expensive for the calculation of (4.17).

Modifying the diffusion term The approach outlined in Chapter 4 learned a modified diffusion term (4.13) in the construction of a variational distribution. However, SDE literature [117] states the true conditioned process should retain the same diffusion term as the unconditioned SDE (see Section 3.3.2). Whilst retaining the original diffusion term is theoretically appealing, its use can make approximating the conditioned process challenging when working with a discretisation scheme. The approximation error

induced by discretisation would often require the use of an extremely small step-size to produce paths consistent with the observations at times where the conditioned SDE varies rapidly (e.g. near an observation). This problem is illustrated, using the Euler-Maruyama scheme and drift estimation technique of chapter 4, in Figure 5.1. As discussed, use of a small-step size could make this approach infeasibly expensive.

Discretising the SDE likelihood Most SDEs do not permit closed-form solutions (see Section 3.3.1), and will therefore require some numerical approximation (e.g. with the Euler-Maruyama scheme). Under discretisation, any variational approach then yields approximations of an approximation – and one that cannot be made arbitrarily accurate without incurring infeasible computational cost (i.e. by considering a smaller step-size). Further, even non-approximate inference for a discretised SDE still leads to an approximation error [149]. Whilst it is theoretically possible to consider alternate discretisation schemes of higher-order than the Euler-Maruyama (see e.g. Kloeden and Platen [81]), these approaches do not typically admit a tractable approximating density, and will only provide the means by which to forward-simulate.

Motivated by the analytic solution to the conditioned process (see Section 3.3.2), we now propose an extension to the work of chapter 4 that retains the original diffusion term. To achieve this – through careful parametrisation of the variational distribution – we approximate ideal gradients for inference under the continuous-time model, approximating the exact solution’s intractable drift function (as seen in equation (3.15), Section 3.3.2). Beyond use of the original diffusion function, this approach has a further significant advantage: approximating the conditioned process resolves to solving an augmented system of SDEs. That is, we are able to use the *continuous-time* Kullback–Leibler (KL) divergence as presented in Archambeau et al. [10], such that the variational objective function can be evaluated using a *path integral* (i.e. an expectation with respect to the solution path of a stochastic differential equation). Our augmented system of SDEs then contains: (1) our approximation to the conditioned SDE; (2) an ODE component which tracks the KL divergence. In this way we are able to approximate the exact solution to the augmented system and the gradients needed for variational inference using numerical solvers.

In more detail, we choose to approximate the SDE on a non-constant, tapered grid of intermediate time-points using the Euler-Maruyama scheme, and calculate gradients by backpropagating through the solver. This enables us to produce paths consistent with the data and simultaneously retain the original diffusion term. Whilst it would have been possible to consider an adaptive numerical scheme, we found these approaches difficult to implement with currently available software implementations (this is discussed more in Section 5.3).

We further note that after the development of our approach, Li et al. [86] proposed a similar method for SDE inference. Unlike our approach, they calculate gradients using an extension to

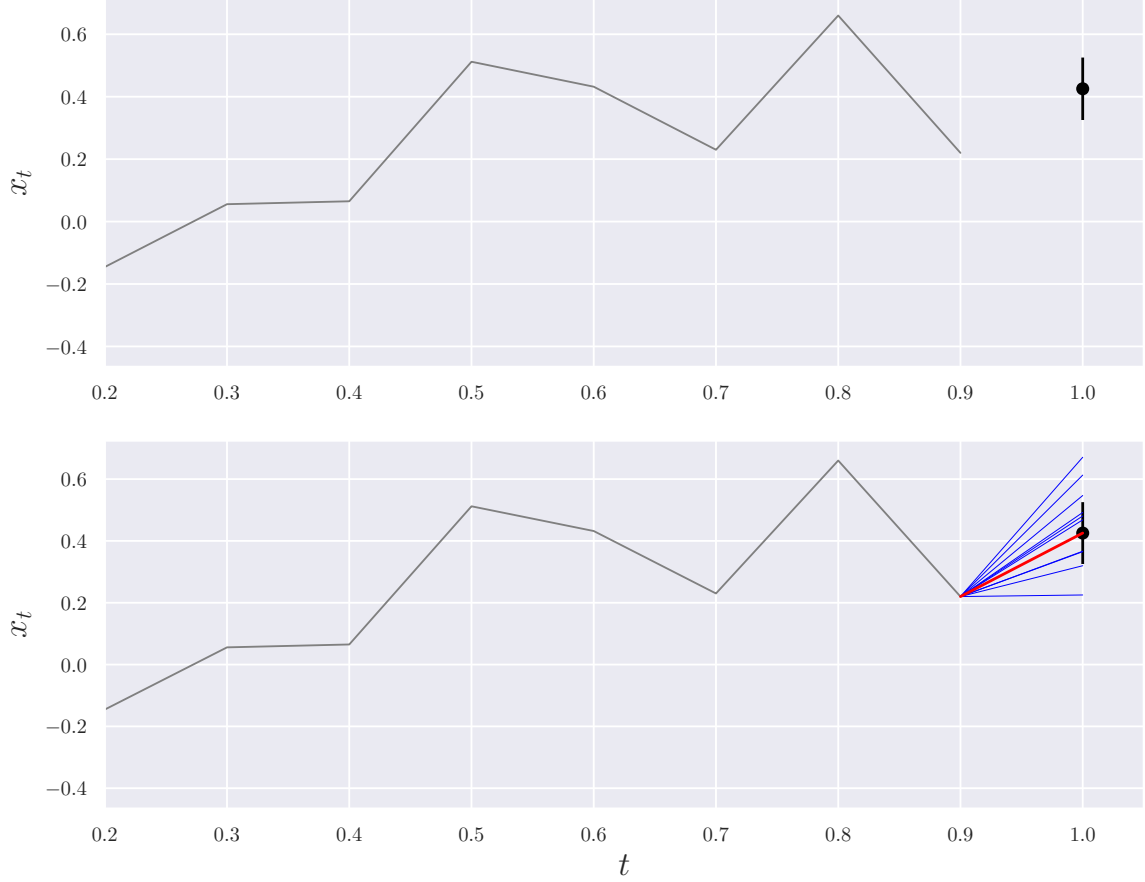


Figure 5.1 Plots demonstrating how a trained variational approximation, using a large step-size ($\Delta t = 0.1$) and unmodified diffusion matrix often fails to produce a path consistent with the observation for some univariate SDE. Both: observation as black circle, with error bars three times the observation model standard deviation. Top: a sample of the approximating process one time-step short of the observation. Bottom: calculated drift function (red) along with 10 realisations from the resulting drift *and* unmodified diffusion (blue).

the *adjoint sensitivity method* [106]. We discuss use of the adjoint method and its application to this problem in Section 5.3.

5.2 Continuous-time variational objective function

Consider some latent process $\{x_t, t \geq 0\}$, $x \in \mathbb{R}^d$, observed at times t_i , for $i \in \mathcal{Q}$ (where $\mathcal{Q}$ is a set of length $D + 1$ indexing observation times), according to

$$y_{t_j} \sim p(y_{t_j} | Fx_{t_j}, \theta), \quad (5.1)$$

where F is a $d_0 \times d$ matrix, for some $0 < d_0 \leq d$, which may be known or the object of inference. We once more concern ourselves with the challenge of fitting an SDE of the form

$$dx_t = \alpha(x_t, \theta)dt + \sqrt{\beta(x_t, \theta)}dw_t, \quad x_{t_0} = x_0, \quad (5.2)$$

where we recall that α is a d -dimensional *drift vector*, β is a $d \times d$ positive definite *diffusion matrix* ($\sqrt{\beta}$ is such that $\sqrt{\beta}\sqrt{\beta}^T = \beta$), w_t is a d -vector of standard and uncorrelated Brownian motion processes and x_0 is an initial condition.

Using the tools of variational inference, we again intend to analyse the system by approximating the posterior density $p(\theta, x|y)$. However, unlike in our previous approach, we design our approximating diffusion process to permit an alternate variational objective: a continuous-time ELBO. As opposed to maximization of the discrete-time ELBO where we calculated the gradient of an *approximation*, we are now able to directly approximate the gradient of the *exact solution*.

5.2.1 Continuous-time Kullback–Leibler divergence

In order to consider the Kullback–Leibler divergence in continuous time, we select our approximating diffusion process to satisfy

$$d\tilde{x}_t = \tilde{\alpha}(\tilde{x}_t, \theta; \phi_x)dt + \sqrt{\beta(\tilde{x}_t, \theta)}dw_t, \quad (5.3)$$

where $\tilde{\alpha}(\cdot; \phi_x)$ is a variational approximation to the drift function, ϕ_x is a collection of variational parameters and $\beta(\cdot; \phi_x)$ is the diffusion function as in (5.2).

We now consider the *Radon–Nikodym* derivative [98] of the process (5.3) with respect to the original diffusion x_t (5.2). Using Girsanov’s theorem [86, 99] it can be show that this is given by

$$\exp\left(-\frac{1}{2}k_T\right), \quad (5.4)$$

subject to $x_0 = \tilde{x}_0$. Here k_T is evaluated as an integral of the form

$$k_s = \int_{t=0}^{t=s} [\alpha(\tilde{x}_t, \theta) - \tilde{\alpha}(\tilde{x}_t, \theta)]^T \beta(\tilde{x}_t, \theta)^{-1} [\alpha(\tilde{x}_t, \theta) - \tilde{\alpha}(\tilde{x}_t, \theta)] dt, \quad (5.5)$$

with initial condition $k_0 = 0$. Full details of this derivation are presented in Li et al. [86], with an informal proof found in Archambeau et al. [10]. In the latter we observe that (5.5) can be recovered in the limit of $\Delta t \rightarrow 0$ between two discretised diffusion processes (much like the objective in Chapter 4).

Here we interpret $-1/2 k_T$ as a continuous-time KL divergence from the approximating diffusion to the target process. Further, should (5.3) and (5.2) not share the same diffusion terms, then the KL divergence between them is not bound to be finite (hence, we could not consider its application for the approach of Chapter 4 where we modify this term).

For our purposes, we are then able augment the SDE system (5.3) with (5.5) considered in differential form (that is, as an ODE)

$$dk_t = [\alpha(\tilde{x}_t, \theta) - \tilde{\alpha}(\tilde{x}_t, \theta)]^T \beta(\tilde{x}_t, \theta)^{-1} [\alpha(\tilde{x}_t, \theta) - \tilde{\alpha}(\tilde{x}_t, \theta)] dt, \quad (5.6)$$

which allows simultaneous numerical solution of (5.6) and (5.3) according to

$$a_t = \begin{pmatrix} k_t \\ x_t \end{pmatrix} = \begin{pmatrix} [\alpha(\tilde{x}_t, \theta) - \tilde{\alpha}(\tilde{x}_t, \theta)]^T \beta(\tilde{x}_t, \theta)^{-1} [\alpha(\tilde{x}_t, \theta) - \tilde{\alpha}(\tilde{x}_t, \theta)] \\ \alpha(\tilde{x}_t, \theta) \end{pmatrix} dt + \begin{pmatrix} 0 \\ \sqrt{\beta(\tilde{x}_t, \theta)} \end{pmatrix} dw_t, \quad (5.7)$$

where a_t is the state vector of our augmented system.

In this way, we are able to approximate the exact solution (and its gradients) (5.5) using numerical solvers of arbitrary precision. We note that, unlike the Euler-Maruyama scheme, our choice of numerical solvers are not restricted to those that permit a tractable approximating density, which further introduces the possibility of an infinitely expressive posterior approximation (we discuss our approach to this in Section 5.3.2).

5.2.2 Continuous-time ELBO

The ODE (5.6) only represents a divergence measure between the approximating process and the *unconditioned* SDE (5.2), given values for the model parameters. Our full variational objective must produce paths consistent with the data. To achieve this, we proceed in the usual way and maximise the ELBO. Use of (5.5) in the ELBO then yields the *continuous-time ELBO*.

We begin by positing some variational distribution for parameters and the initial condition $q_0(\theta, \tilde{x}_0; \phi_\theta)$ (our choice for $q_0(\cdot; \phi_\theta)$ is detailed in Section 5.4). Given priors for the model parameters and initial condition, $p(\theta)$ and $p(x_0|\theta)$, the full objective function (equation (2.35), Section 2.4) is then

$$\begin{aligned} \mathcal{L}(\phi) = \mathbb{E}_{\theta, \tilde{x} \sim q(\cdot; \phi)} & \left[\log q_0(\theta, \tilde{x}_0; \phi_\theta) - \log p(\tilde{x}_0|\theta) - \right. \\ & \left. - \log p(\theta) - \sum_{i \in Q} \log p(y_{t_i}|\tilde{x}_{t_i}, \theta) + k_T/2 \right]. \end{aligned} \quad (5.8)$$

The derivation of (5.8) is intuitively similar to that which is presented in Section 2.4; however, we refer the reader to Li et al. [86] for full details.

Monte Carlo gradient estimates (equation (2.46), Section 2.4.1) for use in stochastic optimization can then be calculated using the *reparameterization trick*

$$\begin{aligned}\widehat{\nabla_{\phi}\mathcal{C}}(\phi) &= -\widehat{\nabla_{\phi}\mathcal{L}}(\phi), \\ &= -\frac{1}{N}\sum_{i=1}^N\left\{\nabla_{\phi}\left[\log q_0\left(\theta^{(i)},\tilde{x}_0^{(i)};\phi_{\theta}\right)-\log p\left(\tilde{x}_0^{(i)}|\theta^{(i)}\right)-\right.\right. \\ &\quad \left.\left.-\log p\left(\theta^{(i)}\right)-\sum_{i\in Q}\log p\left(y_{t_i}|\tilde{x}_{t_i}^{(i)},\theta^{(i)}\right)+k_T^{(i)}/2\right]\right\},\end{aligned}\quad (5.9)$$

for some $\theta^{(i)} = g_{\theta}(\epsilon_1^{(i)}, \phi_{\theta})$ and $\tilde{x}^{(i)} = g_x(\epsilon_2^{(i)}, \theta^{(i)}, \phi_x)$. (We note $k_T^{(i)}$ does not require reparameterization, as it follows deterministically from $\theta^{(i)}$ and $\tilde{x}^{(i)}$.) However, obtaining gradients of the augmented SDE system (i.e. of $\tilde{x}_t$ and k_T) for use in (5.9) is difficult, as the system does not permit closed-form solutions. Our approach to the problem is discussed in Section 5.3.

5.2.3 Choice of variational drift function

We now seek to define a form of the variational drift function (5.3) for use in the continuous-time KL divergence. To proceed, we begin by recalling the process defined by conditioning (5.2) on an initial state, x_t and an exactly observed future state, x_{t_j} , for $t_j > t$ (see Section 3.3.2), given by

$$\hat{\alpha}(x_t, \theta) = \alpha(x_t, \theta) + \beta(x_t, \theta)\nabla_{x_t}\log p\left(x_{t_j}|x_t, \theta\right), \quad (5.10)$$

$$\hat{\beta}(x_t, \theta) = \beta(x_t, \theta), \quad (5.11)$$

where $p\left(x_{t_j}|x_t, \theta\right)$ is the transition density of the unconditioned process. In defining an approximating SDE (5.3), we would ideally use (5.10) for drift estimation. However, requiring the gradient of a typically intractable transition density renders this approach infeasible.

Alternatively, we choose our approximation to define the following drift and diffusion functions

$$\tilde{\alpha}(\tilde{x}_t, \theta) = \alpha(\tilde{x}_t, \theta) + \sqrt{\beta(\tilde{x}_t, \theta)}\gamma(x_t, \theta; \phi_x), \quad (5.12)$$

$$\tilde{\beta}(\tilde{x}_t, \theta) = \beta(\tilde{x}_t, \theta), \quad (5.13)$$

where $\sqrt{\beta(\tilde{x}_t, \theta)}$ represents the matrix square-root as a lower-triangular Cholesky decomposition, and $\gamma(\tilde{x}_t, \theta; \phi_x) \in \mathbb{R}^{d \times 1}$. Indeed, the measure (5.5) is only guaranteed to be finite

should the diffusion term in the approximating SDE be identical to that of the target process [10]. In that sense, our approximation to the conditioned process must be considered in the form presented here [99].

Substituting (5.12) into (5.6) yields additional computational benefit

$$\begin{aligned}
dk_t &= \left(\alpha(\tilde{x}_t, \theta) - \alpha(\tilde{x}_t, \theta) - \sqrt{\beta(\tilde{x}_t, \theta)} \gamma(\tilde{x}_t, \theta; \phi_x) \right)^T \beta(\tilde{x}_t, \theta)^{-1} \times \\
&\quad \times \left(\alpha(\tilde{x}_t, \theta) - \alpha(\tilde{x}_t, \theta) - \sqrt{\beta(\tilde{x}_t, \theta)} \gamma(\tilde{x}_t, \theta; \phi_x) \right) dt, \\
&= \left(-\sqrt{\beta(\tilde{x}_t, \theta)} \gamma(\tilde{x}_t, \theta; \phi_x) \right)^T \beta(\tilde{x}_t, \theta)^{-1} \left(-\sqrt{\beta(\tilde{x}_t, \theta)} \gamma(\tilde{x}_t, \theta; \phi_x) \right) dt, \\
&= \beta(\tilde{x}_t, \theta) \beta(\tilde{x}_t, \theta)^{-1} \gamma(\tilde{x}_t, \theta; \phi_x)^T \gamma(\tilde{x}_t, \theta; \phi_x) dt, \\
&= \gamma(\tilde{x}_t, \theta; \phi_x)^T \gamma(\tilde{x}_t, \theta; \phi_x) dt,
\end{aligned} \tag{5.14}$$

where we have replaced the computational challenge of matrix inversion with calculation of the Cholesky decomposition for (5.12).

5.3 Gradient estimation of an SDE system

Our approach to accessing gradients of the SDE system for the stochastic optimization of (5.8) proceeds as follows. We (1) select an approach for the discrete-time forward-simulation of the SDE system, and then (2) access gradients of the resulting approximate solutions. In this Section we discuss our approach to the problem, along with some promising alternatives. We begin by examining available choices of numerical solution, before selecting a corresponding method for gradient calculation.

5.3.1 Strategies

Numerical solvers

As the augmented system of SDEs is analytically intractable, we are required to approximate its solution. For this purpose, we consider two class of numerical solver: adaptive and non-adaptive methods. Non-adaptive schemes, like the Euler-Maruyama, discretise with a fixed grid of uniform times a-priori; whereas adaptive approaches (e.g. [84]) calculate time-steps ‘adaptively’. The latter is designed to minimise discretisation error (and choose a correspondingly small step-size) at times where the SDE varies more rapidly.

As discussed, limitations of a working with a uniform time grid can make adaptive approaches seem appealing. In theory, a good adaptive solver could circumvent the challenge of producing

diffusion paths consistent with difficult observation regimes (where you would expect the SDE to vary rapidly close to the observation). However, there is no guarantee that an adaptive approach would produce reliable step-size estimates when the variational approximation does not closely resemble the target SDE (e.g. early in training). Furthermore, adaptive approaches can be additionally challenging to implement in automatic differentiation frameworks, requiring backpropagation through the solver. When undertaking this research, we did not find current implementations suitable for this purpose. (We note the method of Li et al. [86] does not backpropagate through the solver, but nonetheless poses its own potential implementation difficulties – see Section 5.6). For simplicity, we therefore use a non-adaptive scheme (details of our approach can be found in Section 5.3.2).

Estimating gradients

In order to estimate gradients for our objective function (5.8), we must calculate derivatives of approximate solutions to our augmented system. For this purpose, we can proceed in one of two ways: backpropagation through the numerical solver, or with use of the *adjoint sensitivity method* [106]. These approaches – their benefits and limitations – are discussed as follows.

Assuming differentiability of our numerical scheme, we can perform backpropagation directly through the solver. Whilst this method is theoretically simple and easy to implement, it comes with the computational overhead of performing both a forward (i.e. approximating the system’s solution) *and* backwards pass (calculating gradients of the approximate solution), along with storing all of the states approximated by the solver. Additionally, calculating gradients in this way is susceptible to numerical instabilities, such as the previously discussed *exploding gradient problem* [17].

The *adjoint sensitivity method*, first proposed by Pontryagin [106] – and most recently used in combination with deep learning for ODEs by Li et al. [86] and adapted for SDEs in Chen et al. [29] – calculates derivatives as the solution of an auxiliary system of differential equations. Unlike backpropagation, the adjoint approach possesses an $\mathcal{O}(1)$ memory cost (although it incurs a slightly larger time complexity versus backpropagating through the solver – see [29] for an in-depth comparison of asymptotic complexities). In this way, we could consider the gradients as an extension of our SDE system, and approximate the solution and its gradients in one forward pass.

Advantages of the adjoint method led us to experiment with its application for our use-case. Unfortunately, the SDE method of Li et al. [86] was published *after* the development of our approach. We therefore sought only to prototype the applicability of the adjoint method proposed for *ODEs* [29]. When applied to estimating the gradients of an SDE system, we

found this implementation insufficient. Indeed, Li et al. [86] latterly proved that the auxiliary ODE system was incorrect for a stochastic extension. As such, we therefore calculate gradients via backpropagation.

5.3.2 Numerical solver

In lieu of adopting an adaptive approach, most forward-solvers require a pre-defined grid of discrete-times. As discussed, use of an unmodified diffusion term can make application of a uniform grid challenging. Alternatively, as a trade-off between numerical tractability and model fidelity, we propose a non-uniform tapered grid of intra-observation time points for use in a forward solver. Whilst we are free to employ a broad collection of numerical schemes, we again utilise the Euler-Maruyama (Section 3.3.1). Our method is detailed as follows.

Begin by considering an interval $[t_{j-1}, t_j]$, where t_{j-1} and t_j are observation times. We transform the uniform grid $(\Delta\tau, 2\Delta\tau, \dots, 1 - \Delta\tau)$, with step-size $\Delta\tau$, based on a log-ratio

$$f(t) = t_{j-1} + (t_j - t_{j-1}) t^{\log(\delta)/\log(\eta)}, \quad (5.15)$$

where $\delta, \eta \in (0, 1)$ and $\delta > \eta$. In practice, we let δ and η vary throughout training (discussed further in Section 5.4). Here we define the transformed time-step between any two concurrent grid times t_i and t_{i+1} as Δt_i . A generative definition for forward-solution of (5.3) is then

$$\Delta \tilde{x}_{t_i} = \tilde{x}_{t_{i+1}} - \tilde{x}_{t_i} = \tilde{\alpha}(\tilde{x}_{t_i}, \theta) \Delta t_i + \sqrt{\beta(\tilde{x}_{t_i}, \theta) \Delta t_i} z_{i+1}, \quad (5.16)$$

where z_{i+1} is an independent $N(0, I_d)$ realisation and I_d is the $d \times d$ identity matrix. It then follows that (5.6) can be estimated by

$$k_T = \sum_{i=1}^P [\alpha(\tilde{x}_{t_i}, \theta) - \tilde{\alpha}(\tilde{x}_{t_i}, \theta)]^T \beta(\tilde{x}_{t_i}, \theta)^{-1} [\alpha(\tilde{x}_{t_i}, \theta) - \tilde{\alpha}(\tilde{x}_{t_i}, \theta)] \Delta t_i, \quad (5.17)$$

where $\tilde{\alpha}(\cdot; \phi_x)$ is some variational approximation to the drift function (detailed in Section 5.2.3) and P is the total number of discrete times employed by the Euler-Maruyama scheme. In practice, each pair of concurrent observation times may necessitate different choices of $\Delta\tau$ to accommodate irregularly spaced observations. Further, to more clearly illustrate this approach, Figure 5.2 displays a comparison simulation of some univariate SDE for both a constant and tapered grid of discrete-times.

Applying (5.12) to (5.16) then generatively defines samples from our variational distribution

$$\tilde{x}_{t_{i+1}} = h \left(\tilde{x}_{t_i} + \left[\alpha(\tilde{x}_{t_i}, \theta) + \sqrt{\beta(\tilde{x}_{t_i}, \theta)} \gamma(\tilde{x}_{t_i}, \theta; \phi_x) \right] \Delta t_i + \sqrt{\beta(\tilde{x}_{t_i}, \theta) \Delta t_i} z_{i+1} \right), \quad (5.18)$$

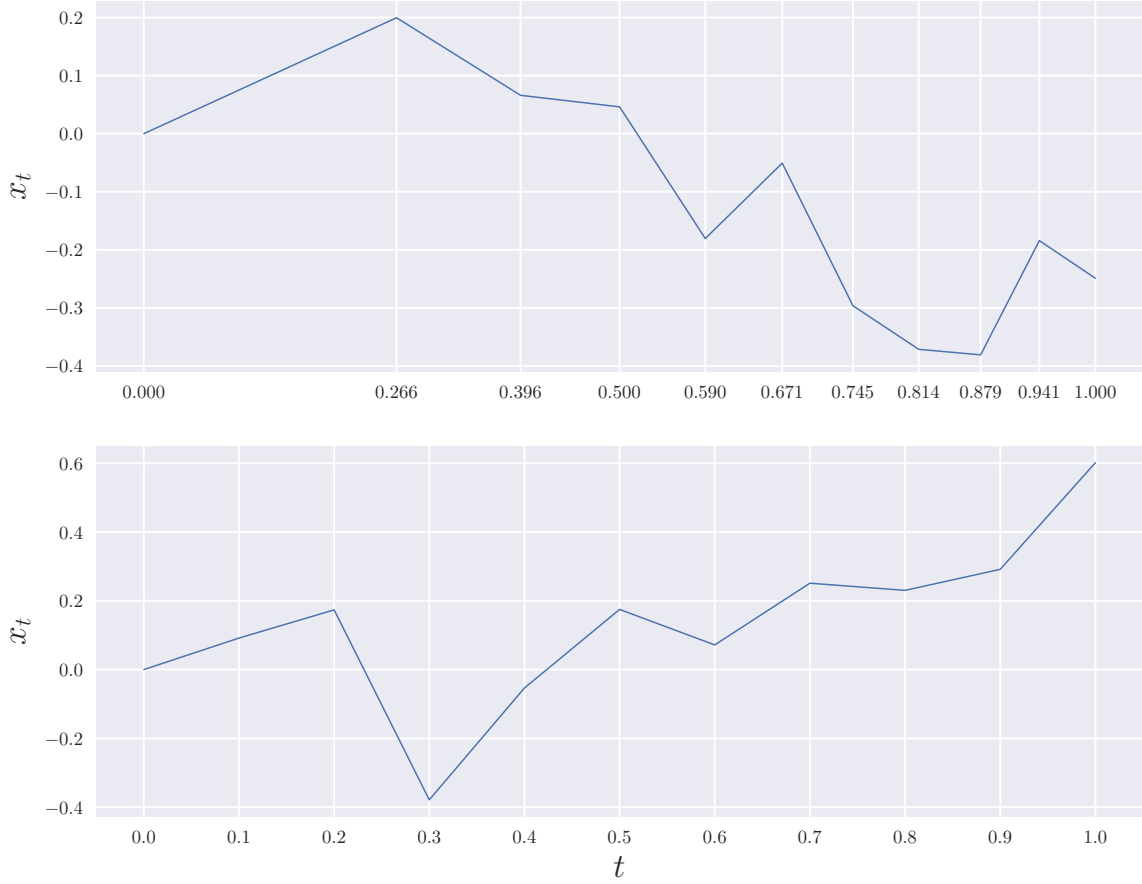


Figure 5.2 A comparison simulation of some univariate SDE between times $[t_{j-1}, t_j] = [0, 1]$, with $\Delta t = 0.1$, using the Euler-Maruyama scheme with a constant [bottom] and tapered grid [top] of times. The tapered grid is presented with $(\delta, \eta) = (0.5, 0.3)$. Here x-axis times represent points of discrete realisations.

where $h(\cdot)$ is used to impose positivity constraints on some components of $\tilde{x}$ (discussed in Section 5.4.3). To express (5.18) for purposes of the reparameterization gradient, let $\epsilon_2 \sim \mathcal{N}(0, I_{dP})$ be the flattened vector of $(z_1, z_2, \dots, z_P)$ realisations. Applying (5.18) repeatedly yields

$$\tilde{x} = g_x(\epsilon_2, \theta, \phi_x). \quad (5.19)$$

As in chapter 4, we use a neural network, with parameters ϕ_x , to serve as our function $\gamma(\tilde{x}_t, \theta; \phi_x)$. Similarly, iterative calculation of $\gamma(\tilde{x}_t, \theta; \phi_x)$ defines an overall recurrent neural network structure. At time t_i , it acts as follows. The network's input, in addition to $\tilde{x}_{t_i}$, is composed of several features computed from: the current diffusion state $\tilde{x}_{t_i}$, the observations y , the parameters θ and the current time t_i (the neural network input is described in full detail within Section 5.4). At each step, the network outputs a $d \times 1$ vector to act as $\gamma(\tilde{x}_{t_i}, \theta; \phi)$, which is used to sample $\tilde{x}_{t_{i+1}}$ from (5.16).

Variable tapered grid

As discussed, there are a number of challenges resulting from use of a uniform time-grid in a non-adaptive solver. These challenges, detailed in Section 5.1, include: (1) producing paths consistent with challenging observations, and (2) approximation error induced by the discretisation scheme. Whilst these challenges could be circumvented with utilisation of a small step-size, we observe that doing so is computationally intensive. In considering our tapered time-grid, we supposed a solution to (1) – but it does not solve (2). In fact, the tapered grid may often necessitate occasionally large step-sizes, inducing a larger discretisation error between selected adjacent times (as seen in Figure 5.2).

To address this, we propose sampling a new η and δ in (5.15) at the start of *each iteration* (the means of sampling is discussed in Section 5.4.2). This has two effects: we still remain able to ‘taper’ toward observations during training; and we produce a $\gamma(\cdot)$ that ‘generalises’ to a much larger collection of intermediate time-points than would be possible if we use a fixed grid of times. In this way we are then able to use our trained approximation in a post-training step to sample a final solution from a much finer time grid than the one used during training. In sampling a trained solution from a finer grid of times, we are therefore able to minimise discretisation error – and solve problem (2).

In more detail, we would ideally consider the problem of estimating the drift function at all times. Whilst this is infeasible, we *can* approximate it across a random discrete subset of times at each iteration (such that η and δ in (5.15) are allowed to vary during training). Treating a finite subset of tapered intra-observation times as a mini-batch of *training points* for drift approximation produces a $\gamma(\cdot)$ that is empirically well-behaved if we take smaller discrete time-steps in our solver (we demonstrate this in Section 5.5). In other words, we are able to achieve an approximation that closely describes the conditioned drift between an arbitrarily large number of intermediate observation time points, without the impracticality of training with an extremely small Δt . In that sense, we loosely view our approximation as permitting a style of *transfer learning* (see e.g. Yosinski et al. [153] and Goodfellow et al. [59]) for $\gamma(\cdot)$ to a much finer discretisation, with no extra training.

5.4 Algorithm and implementation details

5.4.1 Approximate parameter posterior

For simplicity, we again use the mean-field Gaussian approximation for θ , such that

$$q(\theta; \phi_\theta) = \prod_{i=1}^c \mathcal{N}(\theta_i; \mu_i, \sigma_i^2), \quad (5.20)$$

where $\phi_\theta = (\mu_1, \dots, \mu_c, \sigma_1, \dots, \sigma_c)$ and $\theta = (\theta_1, \dots, \theta_c)$. Recall that to express θ using this reparameterization trick, we define

$$\theta = g_\theta(\epsilon_1, \phi_\theta) = S\epsilon_1 + \mu. \quad (5.21)$$

where $\epsilon_1 \sim N(0, I_c)$, $S = \text{diag}(s_1, \dots, s_c)$ and $\mu = (\mu_1, \dots, \mu_c)$. Again, it may be additionally necessary to transform θ to an alternative parametrisation so that a Gaussian approximation is appropriate e.g. log-transforming parameters constrained to be positive.

5.4.2 Variable time grid

As discussed, at each training iteration we typically draw new values of η and δ for use in (5.15). We begin by sampling

$$\eta \sim \mathcal{U}(0.1, 0.5), \quad (5.22)$$

where $\mathcal{U}$ is the uniform distribution. As to preserve $\delta > \eta$ (such that the times ‘taper’ toward observations), we then draw

$$\delta \sim \mathcal{U}(\eta + 0.1, \eta + 0.4). \quad (5.23)$$

Empirically we found that varying the placement of intra-observation time-points (and the severity of the tapering scheme) permitted better generalisation when sampling a final solution from a more regular grid of times.

In order to obtain a fine time-grid for sampling the converged solution and minimising discretisation error, we suppose a uniform grid by selecting $\delta = \eta$. We then choose some small $\Delta\tau$ – therefore recovering the Euler-Maruyama scheme of Section 3.3.1.

5.4.3 Ensuring positivity

As discussed, we are often required to enforce positivity constraints on some components of x (e.g. when modelling population dynamics). However, unlike in chapter 4, use of the continuous-time ELBO makes taking $h(\cdot)$ as the softplus function more challenging. That

is, the approach as detailed in Section 4.2.3 would alter the diffusion matrix at all times, invalidating our ELBO calculations. As an alternative, we propose taking $h(\cdot)$ to be an ELU-like transformation. For this purpose we introduce some small, positive value k , where k represents a prior on the lower bound of the conditioned SDE x . Any positive-valued dimension in x is then subject to the transformation

$$h(x) = \begin{cases} x & x > k \\ k \exp(x - k) & x \leq k \end{cases}. \quad (5.24)$$

In this way, $h(\cdot)$ is the identity function for values larger than k , and slowly decays to zero for smaller inputs. Whilst this method will additionally alter the diffusion matrix, it only does so (unlike the softplus transformation) for $x < k$. Nonetheless, we accept this approach is a pragmatic one, where its occasional modification of the diffusion matrix is not without possible complication. We therefore regard finding a more theoretically appealing approach as an obvious area of future work.

5.4.4 Algorithm

Our overall inference algorithm is given in Algorithm 3. This aims to maximise the ELBO (5.8) using Monte Carlo gradient estimates (5.9). For simplicity, we present the instance of equally spaced observation times, such that we sample one value of η and δ across all intra-observation times at the start of each iteration.

Algorithm 3 variational inference for SDEs by approximating the continuous-time gradients

Initialise ϕ_0 and $k = 0$.

loop

Sample δ and η .

Sample $\epsilon_1^{(i)}, \epsilon_2^{(i)}$ for $1 \leq i \leq N$.

Calculate $\widehat{\nabla_{\phi} \mathcal{C}}(\phi_k)$ using (5.9).

Calculate ϕ_{k+1} using stochastic gradient descent, or a similar algorithm, and increment k .

end loop

Sample $\epsilon_1^{(i)}, \epsilon_2^{(i)}$ for some small $\Delta t \forall i$

5.4.5 Optimisation and implementation details

Given the current time t_i , the nearest future t_j and past observations times t_{j-1} , the RNN cell input is as follows:

- The parameters θ .

- The most recent latent state, $x_{t_{i-1}}$.
- The time until the next observation, $t_j - t_i$.
- The next observation time, t_j .
- The difference between the next observation and what the mean observation would be at the most recent latent state, $y_{t_j} - F'x_{t_{i-1}}$.

Algorithm 2 requires automatic differentiation of (4.17). This can be achieved using the standard tool-kit of backpropagation after *rolling-out* the RNN i.e. stacking m copies of the RNN cell to form a deep feed-forward network. We again follow Pascanu et al. [104] and perform gradient clipping using the L_1 norm.

5.5 Examples

We implement our experiments for examples designed to challenge our ability to: (A) recover a close approximation to the analytical solution for a tractable SDE; (B) perform inference for high-dimensional chaotic systems; (C) extend the epidemic model of Section 4.3.2 to higher dimensions, with a more challenging observation regime.

In all of the following experiments similar tuning choices worked well. We use a batch size of $N = 50$ in (5.9). Our RNN cell has five hidden layers each with 30 hidden units and rectified-linear activation. We implement our algorithms in PyTorch using the Adam optimiser [78] and report results using an 8-core CPU.

As discussed, our approach utilises two discretisation schemes: a tapered grid for use in training (5.15) (which we is changed at the start of each iteration), and a fine grid from which we sample the final solution. Here the final experimental results in all of our examples are drawn from a uniform grid with $\Delta t = 1 \times 10^{-6}$. Choices of the training hyperparameter $\Delta \tau$ are detailed specific to each example.

5.5.1 Ornstein-Uhlenbeck process

Begin by considering the simple univariate Ornstein-Uhlenbeck process satisfying the following SDE

$$dx_t = \theta_1 (\theta_2 - x_t) dt + \sqrt{\theta_3} dw_t, \quad (5.25)$$

where θ_1 , θ_2 and θ_3 represent the mean-reversion rate, process mean and the infinitesimal variance, respectively. As detailed in Appendix A.2.1, (5.25) permits a closed-form solution for the transition density, which resolves as Gaussian between adjacent states.

Now suppose we further choose to observe the process (5.25) with additive Gaussian noise

$$y_{t_j} \sim \mathcal{N}(x_{t_j}, \sigma^2). \quad (5.26)$$

In such a case, the SDE (5.25) conditioned on (5.26) is additionally tractable (see Appendix C.1 for a derivation). Using this example, we are therefore able to investigate our method's ability to approximate the analytical solution on fine grid of times where the discretisation error is negligible.

For our data we simulate one synthetic realisation at $t = 1.0$, with known initial condition $x_0 = 5.0$, under parameters, $\theta = (0.5, 5.0, 1.0)$, which we assume known. The single observation is then assumed available according to (5.26), where $\sigma = 0.1$ (further assumed known). We perform inference for x using $\Delta\tau = 0.1$ in (5.15). Our analysis took 3 minutes on a single CPU.

Figure 5.3 displays samples from the variational approximation on the typical time-grids used for both training and final sampling. Here we see that the tapered grid used for an example training iteration struggles to produce paths consistent with the observation (indeed: it insufficiently ‘tapers’ toward the data). However, upon introducing a finer grid of intermediate time-points, the approximation generalises well – with no extra training – producing paths consistent with the data.

For visual comparison between the exact solution and our variational approximation, Figure 5.4 displays 95% credible regions for samples from both processes. Here we note that the credible regions are virtually identical. To more robustly investigate the accuracy of our approximation, we employ the two-sample kernel test as presented by Gretton et al. [61] (here we use a Gaussian kernel). Due to the challenge of performing the test in extremely high dimensions (sampling with $\Delta t = 1 \times 10^{-6}$ up to $T = 1.0$ results in 1,000,000 latent states), we compare using a subset of 20 times $t = 0.05, 0.10, \dots, 0.95, 1.00$. Testing against the null hypothesis (with 10,000 samples) that the samples came from the same distribution, we calculate a p-value of 0.87.

5.5.2 Stochastic Lorenz

The Lorenz 63 dynamical system [89], originally presented to model atmospheric convection, is a canonical model of chaotic behaviour. For our purposes, we examine an SDE version as

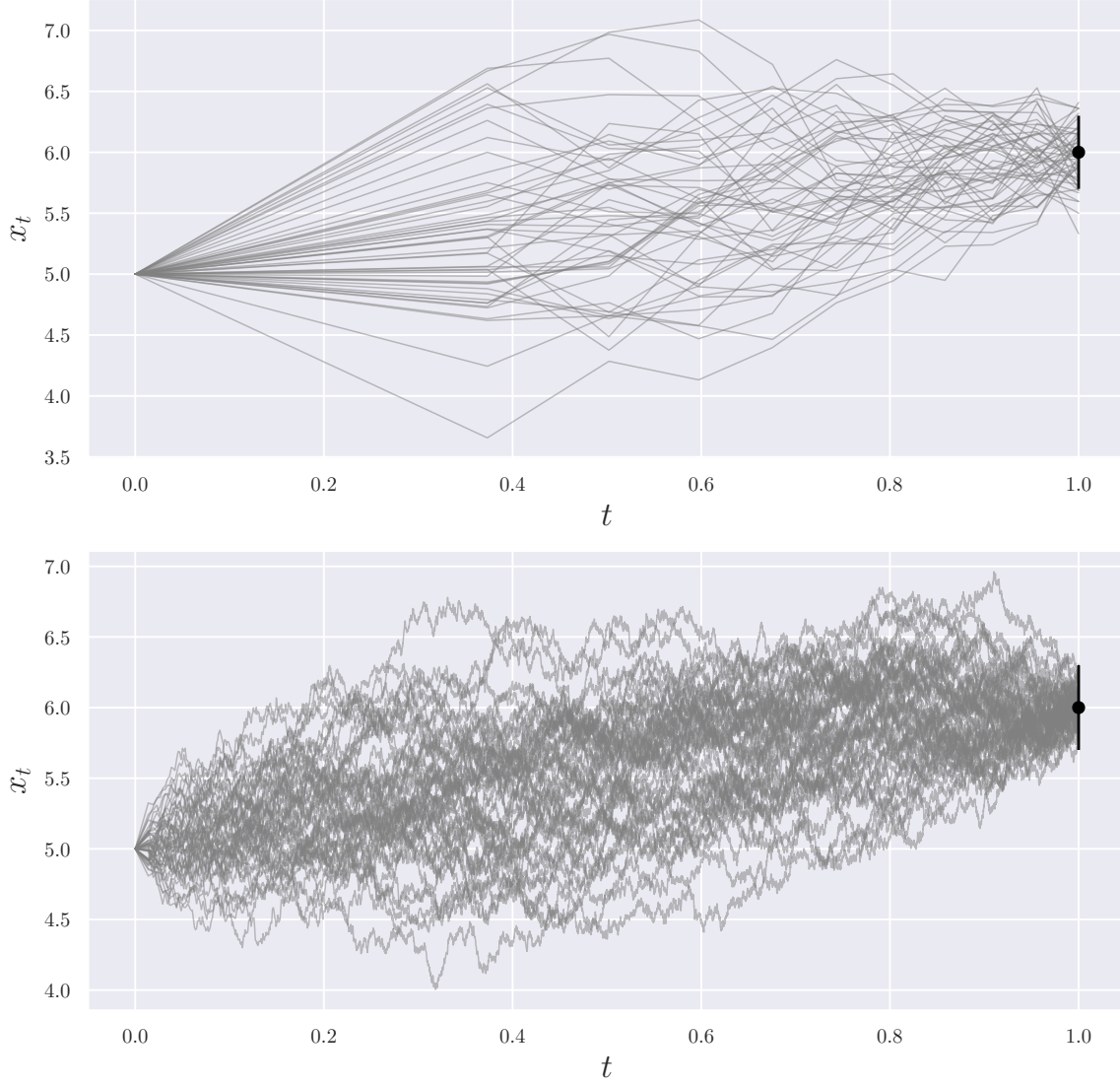


Figure 5.3 Top: sample from the trained variational approximation with $\Delta\tau = 0.1$ in (5.15). Bottom: sample from the trained variational approximation from a uniform grid with $\Delta t = 1 \times 10^{-6}$. Top and bottom: observation (black dot) with three times standard deviation (black lines).

presented in Vrettas et al. [141], which defines drift and diffusion functions

$$\alpha(x_t, \theta) = \begin{pmatrix} \theta_1 (x_{t,2} - x_{1,t}) \\ \theta_2 x_{t,1} - x_{t,2} - x_{t,1} x_{t,3} \\ x_{t,1} x_{t,2} - \theta_3 x_{t,3} \end{pmatrix}, \quad (5.27)$$

$$\beta(x_t, \theta) = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}. \quad (5.28)$$

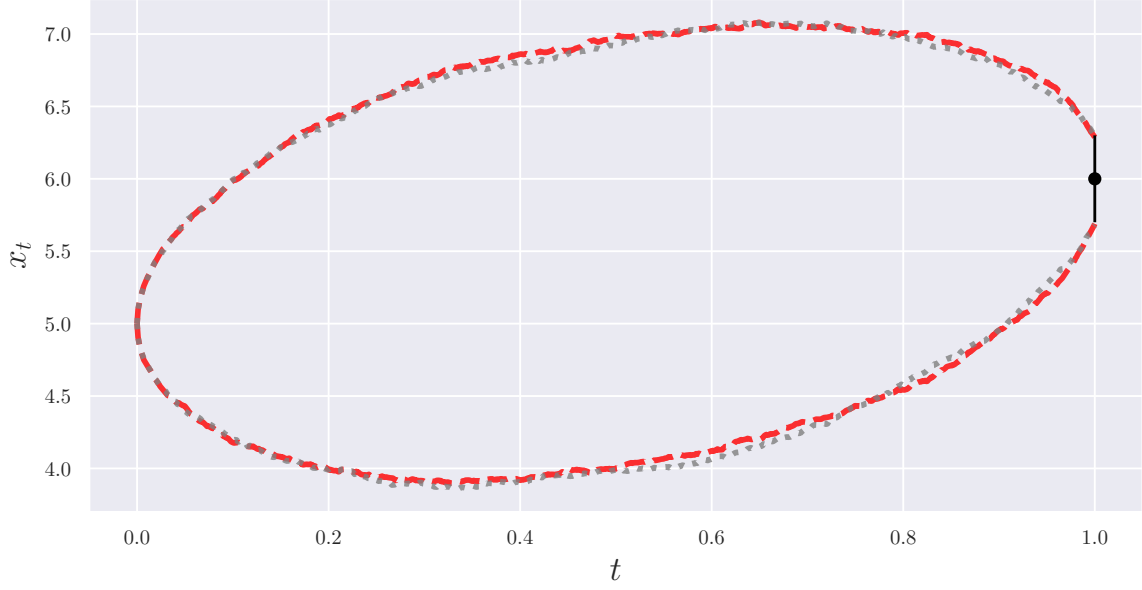


Figure 5.4 95% credible regions for exact solution (red dash) and variational approximation (grey dot) estimated using 10,000 samples. Available observation (black dot) with three times standard deviation (black lines).

Here $x_{t,i}$ denotes the i^{th} dimension of x_t , such that $x_t = (x_{t,1}, x_{t,2}, x_{t,3})$. This setting investigates our ability to perform inference with multi-dimensional, non-linear systems.

For our data we simulate synthetic realisations under parameters known to produce chaotic behaviour, $\theta = (10, 8/3, 28)$, up to $T = 10$, and let $\sigma_1 = \sigma_2 = \sigma_3 = 1.0$, which we further assume known. Observations of this process are then assumed available subject to independent Gaussian noise, such that $y_{t_i} \sim \mathcal{N}(x_{t_i}, I_3)$, at $t = 0.2, 0.4, \dots, 9.8, 10.0$. We perform inference under $\mathcal{N}(0, 10^2)$ priors for $\log \theta_1, \log \theta_2$ and $\log \theta_3$, utilising $\Delta t = 0.02$ in (5.15). Our analysis took 50 minutes on a single CPU.

Figure 5.5 displays estimates of the latent diffusion paths as 95% credible regions, and Figure 5.6 shows marginal density estimates for the parameters θ . We note that both the posterior estimates of x and θ have successfully recovered the latent path and parameter values used for simulation, respectively.

5.5.3 Lorenz 96

Extending inference of chaotic systems to higher dimensions, we now examine the Lorenz 96 [90]. Here we consider a stochastic extension (as similarly examined in Drovandi et al. [37]),

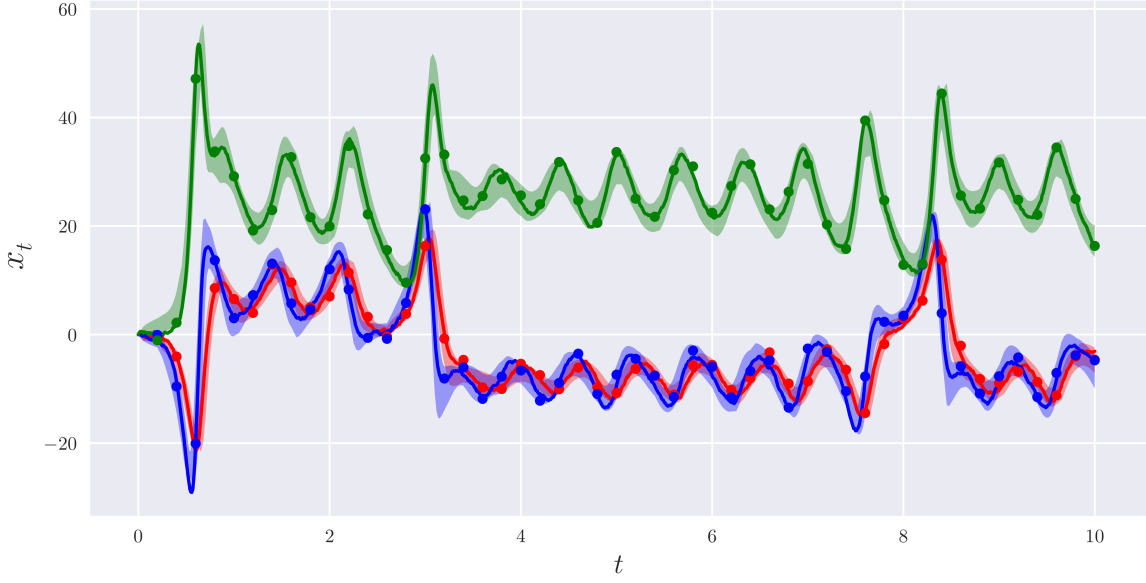


Figure 5.5 95% shaded credible regions for $x_{t,1}$ (green), $x_{t,2}$ (blue), $x_{t,3}$ (red). Available data for the respective components are presented as dots. Latent x used for simulations are presented as bold lines.

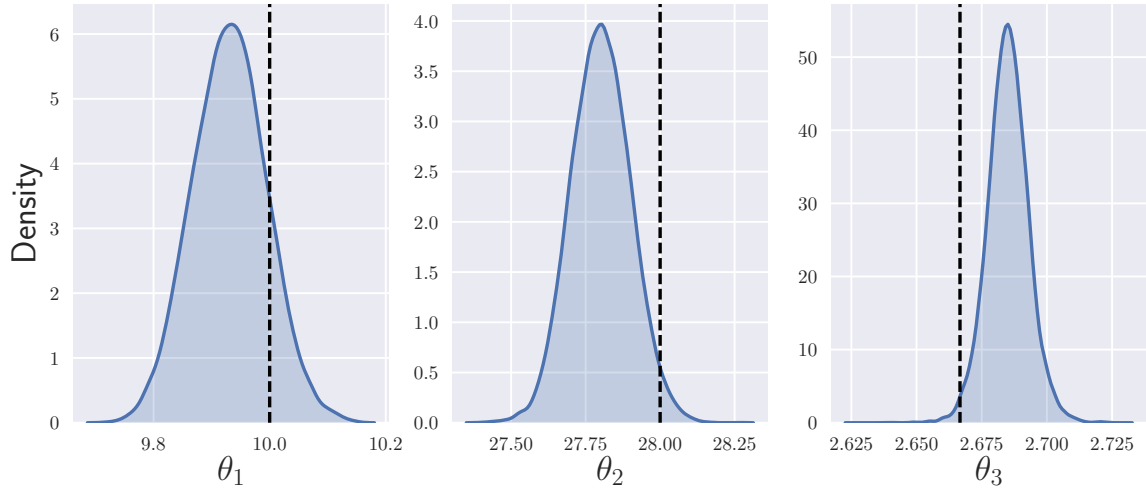


Figure 5.6 Marginal posterior density estimates from variational inference for the Lorenz 68 SDE, with true values displayed with dashed black lines.

in which the i^{th} dimension of the n -dimensional system state satisfies an SDE of the form

$$dx_{t,i} = [x_{t,i-1} (x_{t,i+1} - x_{t,i-2}) - x_{t,i} + \theta_1] dt + \theta_2 dw_t, \quad i = 1, 2, \dots, n, \quad (5.29)$$

where θ_1 is the forcing term, and θ_2 is the infinitesimal standard deviation. In practice, all indices are subject to modular arithmetic (such that e.g. the state index $i + 1$ is calculated

by $i + 1 \pmod{n}$), which we suppress for notational simplicity. In analysing this model, we test our capacity to approximate the dynamics of non-linear models with arbitrarily high dimensions.

For our data we simulate synthetic realisations from an $n = 16$ dimensional system under parameters known to produce chaotic behaviour, $\theta = (8, 0.5)$, up to $T = 5$. Observations of this process are then assumed available subject to independent Gaussian noise, such that $y_{t_i} \sim \mathcal{N}(x_{t_i}, I_n)$, at $t = 0.2, 0.4, \dots, 4.8, 5.0$. We perform inference under $\mathcal{N}(0, 10^2)$ priors for $\log \theta_1$ and $\log \theta_2$, utilising $\Delta t = 0.02$ in (5.15). Our analysis took 150 minutes on a single CPU.

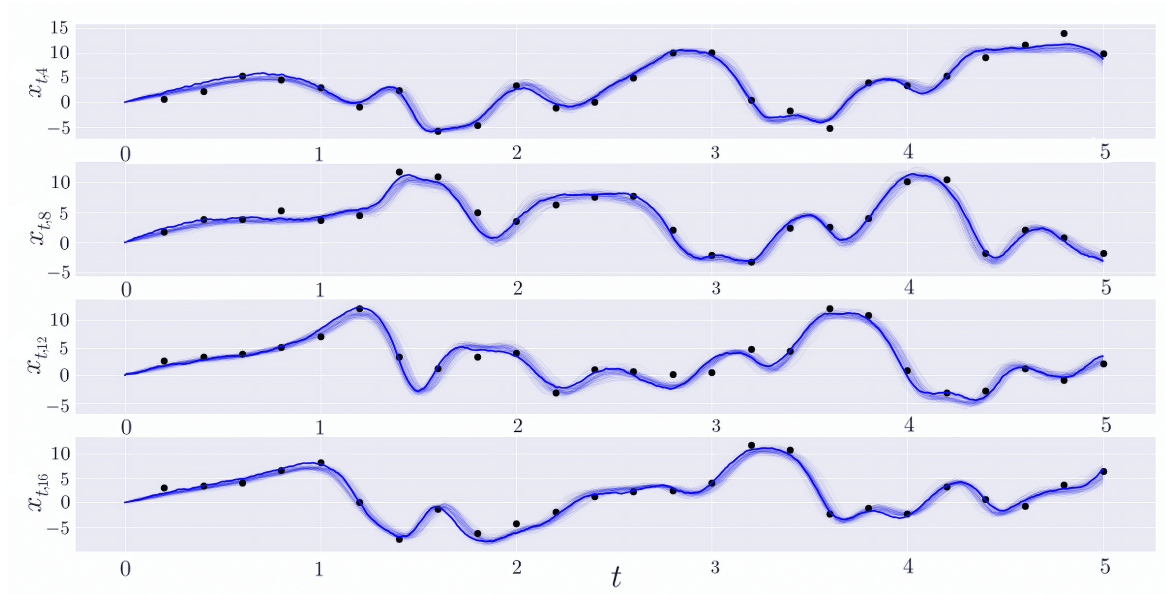


Figure 5.7 50 samples from the trained variational approximation for $x_{t,4}$, $x_{t,8}$, $x_{t,12}$ and $x_{t,16}$. Available data for the respective components are presented as dots. Latent x used for simulations are presented as bold lines.

Figure 5.7 displays samples of the latent diffusion paths for state dimensions 4, 8, 12 and 16. Figure 5.8 shows marginal density estimates for the parameters θ . We again note that both the posterior estimates of x and θ have successfully recovered the latent path and parameter values used for simulation, respectively.

5.5.4 Epidemic model

Similarly to the SIR model (Section 4.3.2), an SEIR epidemic model (e.g. Dureau et al. [39], Endo et al. [44], Baguelin et al. [13]) describes the spread of an infectious disease. The population is subdivided into susceptible (s), exposed (e), infective (i), and removed (r). Here three types of event occur: susceptibles can become infected by the infectious, infected

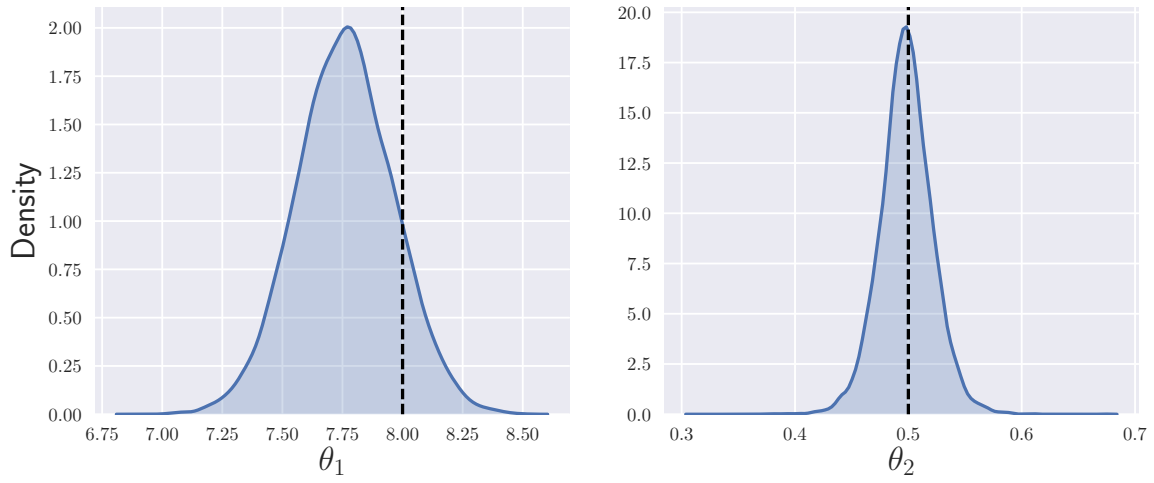


Figure 5.8 Marginal posterior density estimates from variational inference for the Lorenz 96 SDE model, with true values displayed with dashed black lines.

but not infective can become infective, and the infectious eventually become removed (either through death or recovery). The dynamics of this system, illustrating transitions between states, is displayed in Figure 5.9. In this way, the SEIR augments the SIR by introducing the notion of an incubation period through the addition of an e state. We further proceed assuming a constant population size, implying that only the s , e and i populations require modelling.

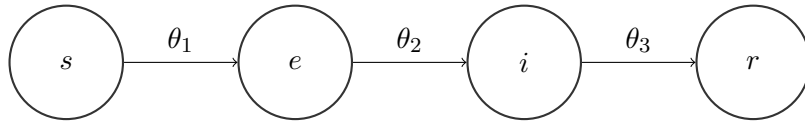


Figure 5.9 Directed graph demonstrating state transition and respective reaction rates (described below) for the SEIR epidemic model.

The SEIR is typically examined as a deterministic system of ODEs, but here we consider a stochastic extension. The resulting SDE has drift and diffusion functions

$$\alpha(x_t, \theta) = \begin{pmatrix} -\theta_1 s_t i_t \\ \theta_1 s_t i_t - \theta_2 e_t \\ \theta_2 e_t - \theta_3 i_t \end{pmatrix}, \quad (5.30)$$

$$\beta(x_t, \theta) = \begin{pmatrix} \theta_1 s_t i_t & -\theta_1 s_t i_t & 0 \\ -\theta_1 s_t i_t & \theta_1 s_t i_t + \theta_2 e_t & -\theta_2 e_t \\ 0 & -\theta_2 e_t & \theta_2 e_t + \theta_3 i_t \end{pmatrix}, \quad (5.31)$$

where $x_t = (s_t, e_t, i_t)'$ is system state at time t , θ_1 is the effective contact rate, and θ_2^{-1} , θ_3^{-1} govern the average period spent in states e and i , respectively. Following Dureau et al. [39], we additionally choose to let the contact rate θ_1 follow a geometric Brownian motion to suppose time-varying infection dynamics

$$d \log(\theta_1) = \varsigma dw_t. \quad (5.32)$$

For our data we consider synthetic realisations designed to mimic the observations available to Dureau et al. [39] of the A/H1N1v influenza pandemic in England, 2009 [44]. Here the authors use a deterministic SEIR model, with diffusion-driven contact rate; however we now consider a fully stochastic extension. In this example, data is available in the form of weekly GP diagnoses, which is assumed subject to additive Gaussian noise

$$\log(y_{t_i}) \sim \mathcal{N}\left(\log\left(\int_{t_i-7}^{t_i} \theta_1 e_t dt\right), \sigma^2\right), \quad (5.33)$$

where each t_i is a unit day. Letting $x_0 = (998, 1, 1)$ and $\theta_{1,0} = 2.5$, we then simulate observations under parameters $\theta_2 = 0.15$, $\theta_3 = 0.45$, up to $T = 70$, and let $\varsigma = 1.0$ (5.32), which we assume known. Weekly observations are then available according to (5.33) (i.e. every 7 unit times), with $\sigma^2 = 10.0$. We perform inference under $\mathcal{N}(0, 10^2)$ priors for $\log \theta_{1,0}$, $\log \theta_2$, $\log \theta_3$ and $\log \sigma$, utilising $\Delta t = 0.02$ in (5.15). Our analysis took 180 minutes on a single CPU.

Figure 5.10 display samples of the latent diffusion paths and marginal density estimates for the unknown parameters. We again note that both the posterior estimates have successfully recovered the latent path and parameter values used for simulation, respectively. This is similarly achieved in the synthetic-data example contained in Dureau et al. [39], but without the extension to a full system of SDEs. As could be anticipated, we additionally observe that after the s population becomes 0 (at around $t = 33$), the effective contact rate can no longer be inferred. As such, $\log \theta_1$ begins to approximate unconditioned Brownian motion. In

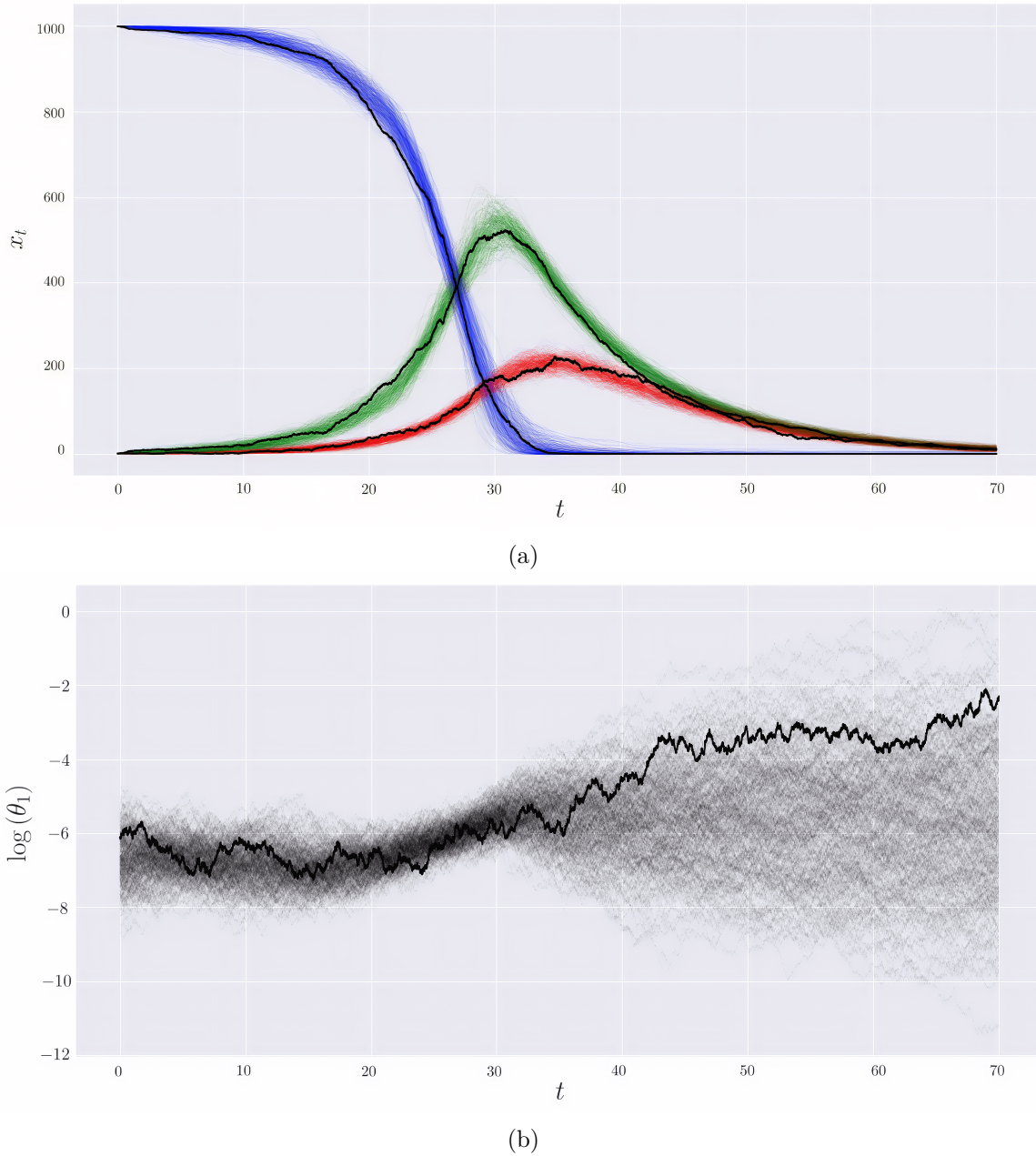


Figure 5.10 (a) 50 conditioned diffusion paths for the SIER epidemic model. s_t (blue), i_t (green), e_t (red) and ground-truth values (black).
 (b) 50 conditioned diffusion paths for θ_1 for the SIR epidemic model, with ground-truth diffusion used for simulation (bold black line).

comparison to the epidemic model of Section 4.3.2, we are likewise able to achieve accurate parameter estimation in a short time, but examine a higher-dimensional system, with a more challenging observation scenario, and sample the result on a finer grid of time.

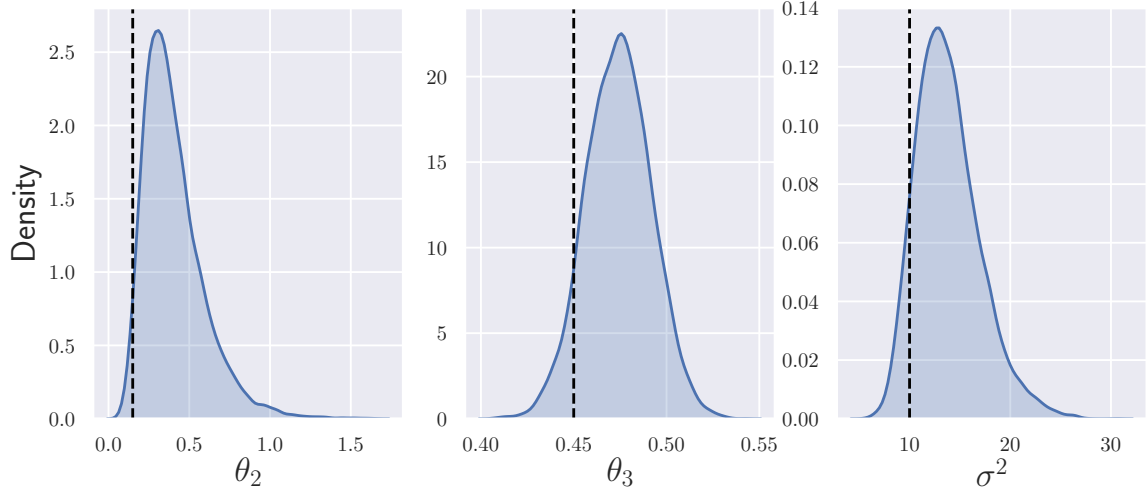


Figure 5.11 Marginal posterior density estimates from variational inference for the SIER epidemic model, with true values displayed with dashed black lines.

5.6 Conclusion

We have successfully extended the approach of Chapter 4 to a continuous-time setting. Our principal contributions are as follows: we avoid the undesirable approximation error introduced by directly discretising the SDE likelihood; and we avoid the need to learn a diffusion function, yielding a cheaper and more theoretically appealing generative process. In this way, we were able to approximate ideal gradients for inference under the continuous-time model – as opposed to calculating gradients of the approximate solution. We subsequently employed this approach to successfully perform inference for a broad class of SDEs, recovering latent diffusion paths and models parameters across a wide-range of problems. We additionally demonstrated a style of *transfer learning*, where we were able to learn an approximating drift function that closely describes the conditioned drift between an arbitrarily large number of intermediate observation times. This enables a post-training sampling step on a finer time grid which minimises discretisation error in our final approximation. This is achieved without the impracticality of training with an extremely small step-size.

Our approach, however, possesses some outstanding questions and potential limitations. Recall the challenge with use of an unmodified diffusion matrix was in approximating a solution that was consistent with the data (which is especially challenging when the infinitesimal variance of the process is large relative to the variance of the observation model). Our approach to this problem was two-fold: we began by learning an approximating solution on a variable tapered grid, which then permitted a final approximation on a finer uniform discretisation. We note, however, that the tapered grid can represent a non-trivial hyper-parameter choice – one where misspecification could lead to bias in parameter estimation. That is, should we

regularly choose a discretisation that insufficiently tapered towards difficult observations, it is conceivable that parameters would be chosen to artificially minimise the infinitesimal variance of the approximating diffusion. Further, in varying the placement of our intra-observation time-points, we supposed this effectively allowed us to learn a drift function at all times (in a similar way to the continuous-time solution). In presenting this approach, however, we did not consider how this technique (vs e.g. a non-variable tapered grid) affected parameter posteriors. That is: does this approach actually enhance our approximations? Future work could consider this question.

Future work should further consider two areas of possible improvement: numerical approximation with adaptive solvers, or calculating gradients with the stochastic adjoint method [86]. Adaptive schemes present a elegant solution to many of the challenges with numerical approximation; the adjoint method offers computational advantage when dealing with high-dimensional or long time-series data. Both extensions, however, are not without potential challenges. These challenges, with respect to each extension, are discussed in more detail as follows.

Whilst approximating gradients with the adjoint method is theoretically attractive, we found it to posses practical limitations. That is, the approach requires calculating gradients with respect to *each sample* in every iteration’s batch of solutions for Monte Carlo objective estimation. Modern automatic differentiation libraries (e.g. TensorFlow, PyTorch) are not especially amenable to this style of gradient computation. Indeed, libraries will natively aggregate gradients across an *entire* batch for use in stochastic gradient descent. Whilst it is possible to implement the adjoint approach, it is non-trivial to do so in a parallel way. That is, this approach may, in practice, prove to be slower (or at least more complicated to implement) than backpropagation.

Whilst adaptive approaches theoretically present a means by which to produce diffusion paths consistent with difficult observations, this requires the solver to implicitly learn where those observations are – and make smaller step-sizes in the preceding times. There is no guarantee that the solver would achieve this, nor *when* this would happen during training. In practice, we need the numerical solver to quickly propose a time-grid that enables the learning of diffusion paths consistent with the observations, preventing convergence to biased parameter estimates. Conversely, this is not a problem with our tapered-grid approach.

Chapter 6

Scalable State-Space Inference with Normalising Flows

In this chapter we begin by presenting the work of Ryder et al. [123]. This paper is currently under review, but builds on Ryder et al. [124], which appeared at the Bayesian Deep Learning workshop at NeurIPS in 2018. We commence by examining the difficulties of state-space modelling as applied to large datasets, before proposing the extension of mini-batch variational methods to overcome subsequent challenges. In order to achieve this we introduce a novel generative model as our variational approximation, a local inverse autoregressive flow, which allows a subsequence of the latent state to be sampled without drawing from its entire distribution. We subsequently demonstrate the ability to perform training iterations using short portions of the time series at low computational cost. We illustrate our method on AR(1), Lotka-Volterra, FitzHugh-Nagumo and stochastic volatility models, achieving accurate parameter estimation in a short time.

We then present work originally appearing in Ward et al. [145] (of which we co-author) that was published at AISTATS in 2020. This work extends the chapter’s earlier techniques, and proposes a method for inference of Gaussian process state-models (GPSSMs) and non-linear latent force models (which are – for practical purposes – a special class of GPSSM whose underlying differential equations are non-linear). We illustrate this approach on GP regressions of both Gaussian and non-Gaussian observation likelihoods (such as count data), before analysing real and synthetic data using latent force models.

6.1 Inference for SSMs

Given a collection of $D + 1$ observations y of a latent process x , we now concern ourselves with the challenge of fitting a state-space model (SSM). Begin by recalling the posterior distribution up to proportionality

$$p(x, \theta | y) \propto p(x | \theta) p(y | x, \theta) p(\theta), \quad (6.1)$$

with

$$p(x | \theta) = p(x_{t_0}) \prod_{i=1}^P p(x_{t_i} | x_{t_{i-1}}, \theta), \quad (6.2)$$

$$p(y | x, \theta) = \prod_{i \in Q} p(y_{t_i} | x_{t_i}, \theta), \quad (6.3)$$

where $Q \subseteq 0:P$.

Now suppose we are interested in estimating (6.1) where P in (6.2) (i.e. the total number of discrete times) is large. Standard solutions to this problem would typically utilise sampling-based approaches such as Kalman and particle filters, along with Markov chain Monte Carlo (MCMC) (e.g. Särkkä [126], Doucet et al. [36], Cappé et al. [27]). All of these methods, however, involve evaluating or estimating the likelihood (6.2) under various choices of θ . Unfortunately, even the fastest such schemes require one $O(P)$ pass through the data, often failing with collapse to a degenerate distribution [139]. Sampling-based approaches can be additionally inefficient in challenging observation regimes, such as when $D \ll P$ (see e.g. bridge construct design in discretised SDE inference - Section 4.1.1), or when $\dim(y_{t_i}) < \dim(x_{t_i})$. Applying these methods to a long time series and/or to data partially observed can therefore be impractically expensive.

In contrast, for models of independent data, it is possible to derive an unbiased estimate the log-likelihood using a short *mini-batch*, at an $O(1)$ cost only (see Section 2.4.1). This allows scalable inference methods based on stochastic gradient optimisation e.g. maximum likelihood or variational inference. As such, we now propose a mini-batch variational inference method for SSMs, in the case of continuous states i.e. $x_{t_i} \in \mathbb{R}^d$. In order to achieve this, we introduce a family of variational approximations $q(\theta, x; \phi)$ based on *normalising flows* (see Section 6.2) [111] such that:

1. It is possible to sample a subsequence $(x_{t_i})_{a \leq i \leq b}$ without sampling the entire x chain.
2. The log-density of mini-batch samples can be evaluated, and used to estimate the log-density of the entire chain.

3. The variational approximation has a fixed number of parameters, which does not grow with P .

To make these properties possible, we impose a restriction that x_{t_i} and x_{t_j} are independent for $|i - j|$ sufficiently large. Hence our approach assumes no long-range dependence in the posterior for x .

Despite this restriction, we show that our approach works well in several examples. These include various challenging observation regimes: sparse observation times, partial observation of the components of x_{t_i} and low observation variance. Our flexible variational family produces good posterior estimates in these examples: at best our variational output is indistinguishable from the true posterior. Additionally, we demonstrate that we can scale up to a challenging example with $P = 1,000,000$ states in 180 minutes.

6.1.1 Related work

Several variational inference methods for SSMs have been previously proposed using stochastic optimisation. Each method has further used a different variational family for x . These approaches include: a multivariate normal distribution with tridiagonal covariance structure [11], an autoregressive distribution (chapter 4), a particle filter [69]. However, all of these methods have an $O(P)$ cost for each iteration of training and/or require storing $O(P)$ parameters. We note the method of Archer et al. [11] could marginalise its normal distribution to avoid the cost of training; however this would still possess a high memory cost of $O(P)$ parameters, which will require $\geq O(P)$ iterations to learn.

Parallel Wavenet [136] uses a normalising flow based generative model for sequence data similar to ours. This incorporates long range dependence through dilated convolutions, while we use only short range dependence to allow mini-batch inference.

The local normalising flow we use is very similar to a *masked convolutional generative flow* (MACOW) [91]. The novelty of our approach is that we apply this flow to fast variational inference for time series, while Ma and Hovy [91] focus on modelling and sampling image data.

A recent sampling-based scheme employing a related strategy to ours for scalable SSM inference is presented in Aicher et al. [2]. This approach uses stochastic gradient MCMC with *buffered* gradient estimates, with estimates based on running a particle filter on a short subsequence of data. Like our contribution, this approach neglects long-range dependence.

6.2 Normalising flows

A normalising flow (see e.g. Papamakarios et al. [101] for a rigorous introduction) represents a random object x as $g_\nu \circ \dots \circ g_2 \circ g_1(z^0)$: a composition of *learnable* bijections of a base random object z^0 , sampled from some simple probability distribution (we typically take this to be draws from a multivariate normal). In this way, normalising flows present a suite of tools allowing one to construct probability distributions of arbitrary complexity. This process is broadly illustrated in Figure 6.1. In practice, we often choose neural networks to parametrise each $g_j(\cdot)$ transformation. Normalising flows proceed, with a focus on our application, as follows.

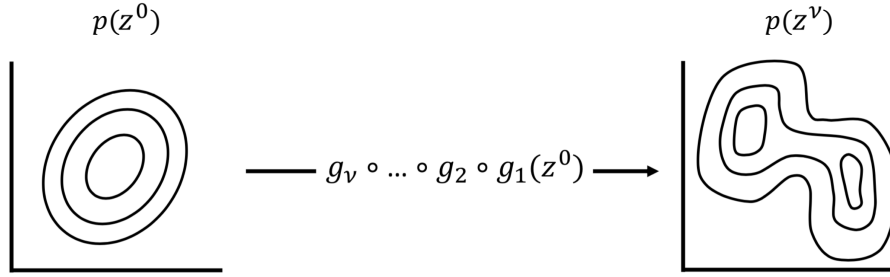


Figure 6.1 Illustration of of a normalising flow. Here we transform a base random variable $z^0 \in \mathbb{R}^2$ through ν bijections to an output z^ν , with probability density function $p(z^\nu)$.

We begin by supposing $x = x_{t_{1:P}}$ and $x_{t_i} \in \mathbb{R}$ (later we consider vector x_{t_i} s). For ease of presentation, we further assume a fixed x_{t_0} , but note it is trivial to extend our method for x through $0:P$. We then take $z^0 = z_{1:P}$ as $\mathcal{N}(0, I_P)$ variables, where I_P is the $P \times P$ identity matrix. We further use z^j to denote the result of z^0 through $g_j \circ \dots \circ g_2 \circ g_1(z^0)$. By the standard change of variable result (e.g. [121]), the log-density of x is

$$\log q(x) = \log p(z^\nu) = \log \mathcal{N}(z^0; 0, I_P) - \sum_{j=1}^{\nu} \log |\det J_j|, \quad (6.4)$$

where J_j is the Jacobian matrix of the transformation g_j given input $g_{j-1} \circ \dots \circ g_2 \circ g_1(z)$. This result defines a critical condition for the construction of flow-based probability models: the transformations $g_j(\cdot)$ must be both *invertible* and *differentiable*. However, naive implementation of transformations for which this condition is satisfied is usually insufficient. In practice, each bijection represents a trade-off between flexibility and computational tractability, with the calculation of $\log |\det J_j|$ often proving prohibitive. To this end, a key research theme has been designing flexible transformations for which the calculation of $\log |\det J_j|$ is computationally efficient (see e.g. Papamakarios et al. [102], Dinh et al. [35], Rezende

and Mohamed [111] for prominent examples). We now discuss one promising solution to the construction of a normalising flow: *inverse autoregressive flows* (IAFs).

6.2.1 Inverse autoregressive flows

IAFs define a composition of *affine* transformations, which, for any given layer, transform an input z^{in} to output z^{out} according to

$$z_i^{\text{out}} = \mu_i(z_{1:i-1}^{\text{in}}) + \sigma_i(z_{1:i-1}^{\text{in}})z_i^{\text{in}}, \quad (6.5)$$

with $\sigma_i > 0$. This transformation *scales* and *shifts* each z_i^{in} . It is common for the scale and shift values, σ_i and μ_i , to be computed by neural networks.

The shift and scale functions for z_i^{out} in (6.5) possess an *autoregressive property* (hence the name). That is, they depend only on z^{in} through z_j^{in} with $j < i$. Hence the Jacobian matrix of the transformation is diagonal with non-zero entries $\sigma_{1:P}$, i.e. reducing the otherwise complex determinant computation to the product of diagonal elements.

The log-density of an IAF made of ν affine layers is

$$\log q(x) = \log \mathcal{N}(z^0; 0, I_P) - \sum_{j=1}^{\nu} \sum_{i=1}^P \log \sigma_i^j, \quad (6.6)$$

where σ_i^j is the shift value for the i th input to the j th affine layer. We further illustrate the generative process using equation (6.5) for a single random variable z_i^1 in Figure 6.2.

Typically an IAF alternates affine layers with *permutation layers*, using order-reversing (or random) permutations. Such layers have Jacobians with absolute determinant 1. Therefore the log-density calculation is unchanged (interpreting j in (6.6) to index the j th *affine layer* not the j th layer of any type).

For our purposes, we may additionally wish to restrict the output of a given IAF with the addition of a final bijection layer. For example – as has been commonly desired – to ensure all x_{t_i} s are positive, an element-wise softplus transformation may be applied. Such transformations will require the addition of an extra term to $\log q(x)$. Full details can be found in Appendix D.3.

IAFs are flexible and, for small P , allow fast sampling – e.g. using GPUs – along with fast calculation of a sample’s log-density. However IAFs become expensive for big P , with large neural networks needed to map between sequences of length P .

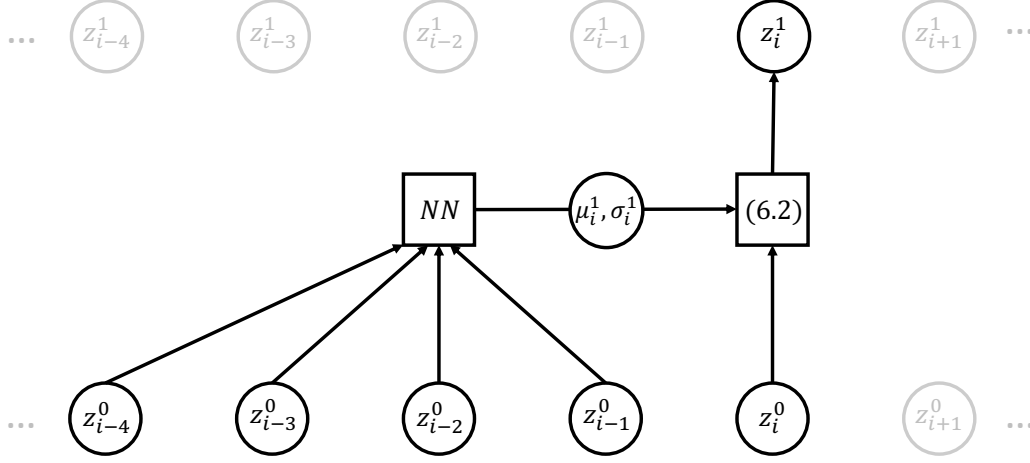


Figure 6.2 Illustration of the generative process and data flow within a standard IAF affine layer. Circles represent data nodes and squares represent functions. For ease of presentation, we truncate the input to $z_{i-4:i-1}^0$, but in practice it extends to all such $z_{1:i-1}^0$. Recall that NN represents a neural network.

One efficient approach for the construction of IAFs is to use a single neural network to output all μ_i, σ_i values for a particular affine layer. Such networks use *masked dense* transformations so that (μ_i, σ_i) depends only on $z_{1:i-1}^{\text{in}}$ as required (e.g. Kingma et al. [79], Germain et al. [51], Papamakarios et al. [102]). In the resulting IAF each affine layer is based on a different neural network of this form. We’ll refer to this as a *masked IAF*.

6.3 Local inverse autoregressive flows

We propose reducing the number of weights that IAFs require with use of a convolutional neural network (CNN) to calculate the μ_i and σ_i values for use in an affine layer. We call this a *local IAF*. Here we explain the principal idea by presenting a version for scalar z_i . In Section 6.3.1 we extend this to the case where z_i is a vector.

To define a local IAF we describe how a single affine layer produces its shift and scale values. The affine layer uses a CNN with input z^{in} , a vector of length P . Let h^k represent the k th hidden layer of the CNN, a matrix of dimension (P, n_k) where n_k is the number of convolutional filters in this layer. The first layer applies a convolution with receptive field length ℓ . This is an *off-centre convolution* such that row i of h^1 is a transformation of $z_{i-\ell:i-1}^{\text{in}}$. We further use zero-padding by taking $z_i^{\text{in}} = 0$ for $i < 0$. The following hidden layers are

length-1 convolutions, so row i of h^{k+1} is a transformation of row i of h^k . The output, h^n , is a matrix of dimension $(P, 2)$ whose i th row contains μ_i and σ_i . The final layer applies a softplus activation to produce the σ_i values, ensuring they are positive. An identity activation is used to produce the μ_i values. The μ_i and σ_i values are used in (6.5) to produce the output of the affine layer. A diagram summarising this process and the resulting dependency structure is displayed in Figure 6.3.

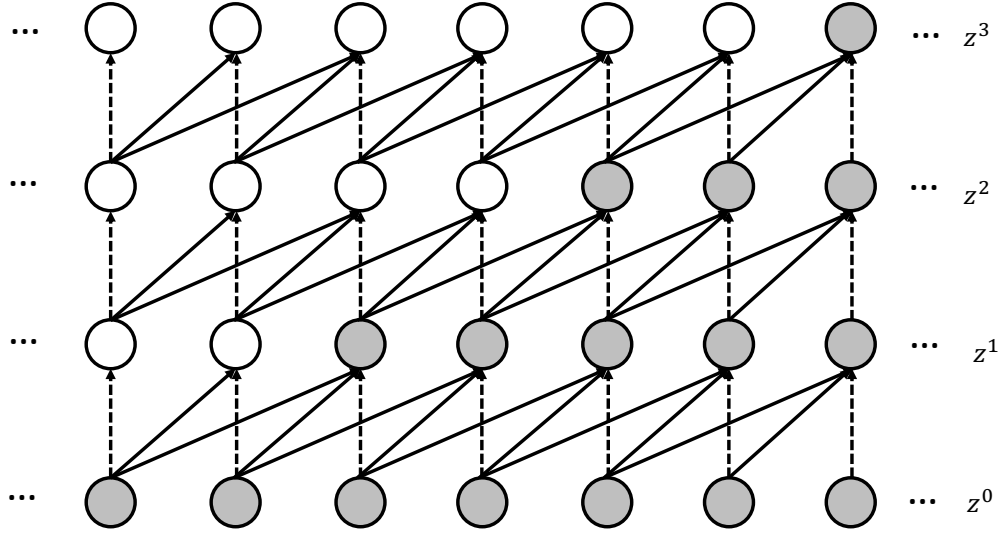


Figure 6.3 Illustration of the generative process and data flow within stacked *local* IAF, with $\nu = 3$ and $\ell = 2$. Black solid lines represent an implicit dependency, learned via a CNN. Black dashed lines represent an explicit dependency, as per equation (6.5). Grey nodes illustrate the dependency structure for last visible element in the output layer.

A local IAF composes several affine layers of the form just described. Some properties of the distribution for the output sequence x are:

1. No long-range dependence: x_{t_i} and x_{t_j} are independent if $|i - j| > \nu\ell$, where ν is the number of affine layers.
2. Stationary local dependence: the distributions of $x_{t_{i:j}}$ and $x_{t_{i+a:j+a}}$ are the same for most choices of a . (There are some differences for subsequences near to the start of x due to zero-padding.)

We relax stationary local dependence by injecting *local side information* to the CNN i.e. giving an extra feature vector s_i as input for each position i in the first CNN layer. We also use *global* side information to allow x to depend on the parameter values θ i.e. giving θ as extra input for every position i . See Section 6.4.2 for details of all the side information we use in practice.

To improve the flexibility of the IAF model, affine layers can be alternated with order-reversing permutations. (Random permutations would not be suitable, as they would disrupt our ability to sample subsequences quickly, as described in Section 6.3.2.) In our proposed method, we consider local IAFs *without* order-reversing permutation layers, as we found these models already sufficiently flexible for our examples¹.

6.3.1 Multivariate case

Here we generalise a local IAF to the case where $x_{t_i} \in \mathbb{R}^d$. We now take z to be a sequence $z_1, z_2, \dots, z_P$ of independent random $\mathcal{N}(0, I_d)$ vectors.

A local IAF affine layer posits the transformation

$$z_i^{\text{out}} = \mu_i + \sigma_i \odot z_i^{\text{in}}. \quad (6.7)$$

Here the vector z_i^{in} is scaled (elementwise multiplication by vector σ_i) and shifted (addition of vector μ_i).

In the scalar case above it was important to allow complex dependencies *between* z_i^{out} values. Now we must also allow dependencies *within* each z_i^{out} vector. To do so we adapt the coupling layers approach of [35].

We use an extra k subscript to denote the k th component of a vector e.g. z_{ik}^{in} . We select some $a \approx d/2$. For $k \leq a$, we take $\mu_{ik} = 0$ and $\sigma_{ik} = 1$, so that $z_{ik}^{\text{out}} = z_{ik}^{\text{in}}$. For $k > a$, we compute μ_{ik} and σ_{ik} using a CNN, modifying the scalar case as follows. Now row i of h^1 is a transformation of $z_{i-\ell:i-1}^{\text{in}}$ (the ℓ vectors preceding z_i^{in}), and also z_{ik}^{in} for $k \leq a$ (the part of z_i^{in} not being modified). The output h^n is now a tensor of dimension $(P, d - a, 2)$ containing μ_{ik} and σ_{ik} values for $k > a$.

This affine layer does not transform the first a components of z_i^{in} . To allow different components to be transformed in each layer, we permute components between affine layers. For example for $d = 2$ this permutation layer transforms z^{in} to z^{out} by $z_{i1}^{\text{out}} = z_{i2}^{\text{in}}$, $z_{i2}^{\text{out}} = z_{i1}^{\text{in}}$.

The log-density is now

$$\log q(x) = \sum_{i=1}^P \lambda_i, \quad \lambda_i = \log \mathcal{N}(z^0; 0, I_P) - \sum_{k=1}^d \sum_{j=1}^{\nu} \log \sigma_{ik}^j, \quad (6.8)$$

¹ Avoiding order-reversing permutations also simplifies our mathematical presentation. If they are included, the main change needed is to interpret σ_i^j in (6.8) as the shift value in the j th affine layer which is applied to its $\rho_j(i)$ th input where ρ_j is the overall permutation up to this layer. Also the details of Section 6.3.2 on sampling a subsequence must be modified.

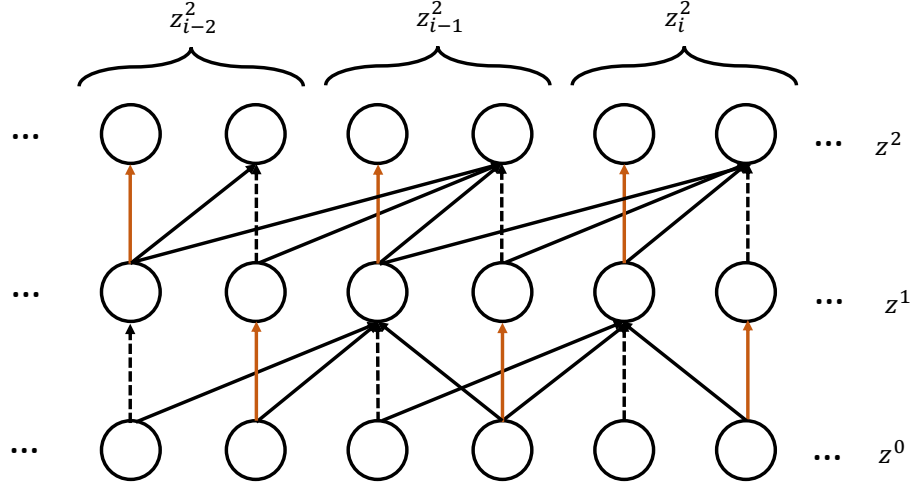


Figure 6.4 Illustration of the generative process and data flow within a multivariate IAF with local permutation, displayed with *flattened* z^i s. Here $\nu = 2$ and $\ell = 2$. Black solid lines represent an implicit dependency, learned via a CNN. Black dashed lines represent an explicit dependency, as per equation (6.5). Red lines denote the instance of $z_{ik}^{\text{out}} = z_{ik}^{\text{in}}$.

where σ_{ik}^j is the k th entry of the shift vector for position i output by the j th affine layer. Decomposing $\log q(x)$ into λ_i contributions will be useful in Section 6.4.

6.3.2 Sampling

Sampling from a local IAF is straightforward. We begin by drawing the base random object z . This is a sequence of length P (of scalars or vectors – the sampling process is similar in either case). We now apply the IAF’s layers to this in turn. To apply an affine layer, we pass the input (and any side information) through the layer’s CNN to calculate the shift and scale values, before applying the affine transformation. The final output is the sampled sequence x , where the associated cost of sampling is $O(P)$.

In the next section, we will often wish to sample a short subsequence $x_{t_{u:v}}$. It is possible to do this at $O(1)$ cost with respect to P . As such, we now describe how to sample a subsequence $x_{t_{u:v}}$ from $q(x|\theta; \phi_x)$ without the need to sample the entire x sequence.

We begin by letting z^0 be the base random sequence and z^j be the sequence after j affine layers. We further assume no layers permuting the sequence order – but do allow layers permuting the components within each vector in the sequence. Suppose there are ν affine layers, so the output is $x = z^\nu$. Algorithm 4 presents the multivariate case where $x_{t_i}, z_i^j, \mu_i^j, \sigma_i^j$

are all vectors in $\mathbb{R}^d$. This includes $d = 1$ as a special case. We denote the k th entry of σ_i^j as σ_{ik}^j . Each iteration of the algorithm – save the last – must therefore sample z_i^j vectors

Algorithm 4 Sampling a subsequence from a local IAF

- 1: Sample z_i^0 for $u - c_0 \leq i \leq v$. These are sampled from independent $\mathcal{N}(0, I_d)$ distributions except when $i \notin 1:P$. In the latter case z_i^0 is a vector of zeros.
- 2: **for** $1 \leq j \leq \nu$ **do**
- 3: Apply the CNN with input $z_{u-c_{j-1}:v}^{j-1}$.
- 4: Extract μ_i^j and σ_i^j output for $u - c_j \leq i \leq v$.
- 5: Calculate $z_{u-c_j:v}^j$ using affine transformation (6.7).
- 6: Permute components in z^j if necessary.
- 7: **end for**
- 8: Return sampled subsequence $x_{t_{u:v}} = z_{u:v}^\nu$, and log density contributions $\lambda_{u:v}$, where

$$\lambda_i = \log \mathcal{N}(z_i^0; 0, I_d) - \sum_{k=1}^d \sum_{j=1}^{\nu} \log \sigma_{ik}^j.$$

over an interval of i which is wider than simply $a:b$. The number of extra z_i^j s required at the lower end of this interval is

$$c_j = (\nu - j)\ell. \quad (6.9)$$

In other words, at each iteration the required interval shrinks by ℓ , the length of the receptive field for z .

6.4 Variational inference for SSMs

We begin by defining our variational family to approximate the SSM posterior (6.1)

$$q(\theta, x; \phi) = q(\theta; \phi_\theta)q(x|\theta; \phi_x), \quad (6.10)$$

where $\phi = (\phi_\theta, \phi_x)$. We use a masked IAF for $q(\theta; \phi_\theta)$ and a local IAF for $q(x|\theta; \phi_x)$. The latter is a distribution for x conditional on θ , so we inject θ as side information. See Section 6.4.2 for more details.

The masked IAF maps a base random vector z_θ to θ using parameters ϕ_θ , as described in Section 6.2.1. The local IAF maps θ and a sequence of vectors z_x to x using parameters ϕ_x as described in Section 6.3.1. Hence we have a mapping from $\epsilon = (z_\theta, z_x)$ to $g(\epsilon, \phi) = (\theta, x)$, allowing us to use the reparameterisation trick below.

This section derives a mini-batch optimisation algorithm to train ϕ based on sampling short x subsequences, so that the cost of one training iteration is $O(1)$ rather than $O(P)$. The

algorithm is applicable for scalar or multivariate x_{t_i} . In this presentation we assume that $D = P$ i.e. there are observations at all states. In practice, however, we often relax this assumption by removing any terms involving y_{t_i} for $i \notin \mathcal{Q}$. In such a setting, the inference challenges is similar to that of bridge construct design (see Section 4.1.1).

For our variational family (6.10), the ELBO (equation (2.35), Section 2.4) is

$$\mathcal{L}(\phi) = \mathbb{E}_{\theta, x \sim q(\cdot; \phi)} [\log p(\theta, x, y) - \log q(\theta; \phi_\theta) - \log q(x|\theta; \phi_x)]. \quad (6.11)$$

Substituting (6.1) and (6.8) into (6.11) gives

$$\begin{aligned} \mathcal{L}(\phi) = \mathbb{E}_{\theta, x \sim q(\cdot; \phi)} & \left[\log p(\theta) - \log q(\theta; \phi_\theta) + \log p(y_{t_0}|x_{t_0}, \theta) + \right. \\ & \left. + \sum_{i=1}^P \{ \log p(x_{t_i}|x_{t_{i-1}}, \theta) + \log p(y_{t_i}|x_{t_i}, \theta) - \lambda_i \} \right]. \end{aligned} \quad (6.12)$$

In order to achieve unbiased mini-batch estimates of (6.12) (as detailed in Section 2.4.1), introduce batches $B_1, B_2, \dots, B_b$: length M sequences² of consecutive integers partitioning $1:P$. Draw κ uniformly from $1:b$. Then an unbiased estimate of (6.12) is

$$\begin{aligned} \mathcal{L}(\phi) = \mathbb{E}_{\theta, x \sim q(\cdot; \phi)} & \left[\log p(\theta) - \log q(\theta; \phi_\theta) + \log p(y_{t_0}|x_{t_0}, \theta) + \right. \\ & \left. + \frac{P}{M} \sum_{i \in B_\kappa} \{ \log p(x_{t_i}|x_{t_{i-1}}, \theta) + \log p(y_{t_i}|x_{t_i}, \theta) - \lambda_i \} \right]. \end{aligned} \quad (6.13)$$

Hence an unbiased estimate of the ELBO gradient (equation (2.46), Section 2.4.1) is

$$\begin{aligned} \widetilde{\nabla}_\phi \mathcal{L}(\phi) = \frac{1}{N} \sum_{j=1}^N \nabla_\phi & \left\{ \left[\log p(\theta^{(j)}) - \log q(\theta^{(j)}; \phi_\theta) + \log p(y_{t_0}|x_{t_0}^{(j)}, \theta^{(j)}) + \right. \right. \\ & \left. \left. + \frac{P}{M} \sum_{i \in B_\kappa} \{ \log p(x_{t_i}^{(j)}|x_{t_{i-1}}^{(j)}, \theta^{(j)}) + \log p(y_{t_i}^{(j)}|x_{t_i}^{(j)}, \theta^{(j)}) - \lambda_i \} \right] \right\}, \end{aligned} \quad (6.14)$$

where $(\theta^{(j)}, x^{(j)}) = g(\epsilon^{(j)}, \phi)$ and $\epsilon^{(1)}, \dots, \epsilon^{(N)}$ are independent ϵ samples. We again take $\widetilde{\nabla}_\phi \mathcal{C}(\phi)$ to be

$$\widetilde{\nabla}_\phi \mathcal{C}(\phi) = -\widetilde{\nabla}_\phi \mathcal{L}(\phi). \quad (6.15)$$

² To generalise to unequal batch lengths, take $\Pr(\kappa) = |B_\kappa|/P$, and replace P/M in (6.13) with $P/|B_\kappa|$.

6.4.1 Optimisation algorithm

Algorithm 5 presents our mini-batch training procedure. Each iteration of Algorithm 5 involves sampling a subsequence of x values of length $M + 1$. The cost is $O(1)$ with respect to the total length of the sequence P , assuming constant M . This compares favourably to the $O(P)$ cost of sampling the entire x sequence.

Note that line 7 involves sampling $x_{t_{u-1}}$, the state immediately *before* the current batch of interest. This is needed for the $p(x_{t_i}|x_{t_{i-1}}, \theta)$ term in (6.13) when $i = u$.

Algorithm 5 Mini-batch variational inference for state space models

- 1: Initialise ϕ_θ, ϕ_x .
 - 2: **loop**
 - 3: Sample a batch κ uniformly from $1:b$. Let u and v denote the endpoints of B_κ .
 - 4: Calculate $\widetilde{\nabla_\phi \mathcal{C}}(\phi)$ from (6.14), generating the terms in the sum as follows.
 - 5: **for** $1 \leq j \leq N$ **do**
 - 6: Sample $\theta^{(j)} \sim q(\theta; \phi_\theta)$.
 - 7: Sample $x_{t_{u-1:v}}^{(j)}$ from $q(x|\theta; \phi_x)$ (unless $u = 1$ in which case sample $x_{t_{u:v}}$), calculating corresponding $\lambda_{u:v}$ values. See Algorithm 4 for details.
 - 8: Calculate $\widetilde{\nabla_\phi \mathcal{C}}(\phi)$ using automatic differentiation of (6.14).
 - 9: **end for**
 - 10: Update ϕ_θ, ϕ_x using stochastic gradient optimisation.
 - 11: **end loop**
-

6.4.2 Side information

We inject side information into the first layer of the CNN for each affine transformation. We include the parameters θ as global side information. Also we provide local side information, encoding information in y local to i which is useful for inferring the state x_{t_i} .

In more detail, first we define s_i to be a vector of data features relevant to x_{t_i} . We pick these so that s_i exists for all i even if (1) no y_{t_i} observations exist for x_{t_i} or (2) i is outside the range $0:P$. The data features we use in our examples are listed in the next section.

The side information corresponding to the i th position in the sequence processed by the CNN is θ and the vector $s_{i-\ell':i+\ell'}$. The tuning parameter ℓ' is a receptive field length (like ℓ earlier). This receptive field extends in both directions from the sequence position i , so it can take account of both recent and upcoming observations. The side information is encoded using a feed-forward network, and this vector is then used as part of the input to the first layer of the CNN.

6.5 Implementation details

Optimisation We use the AdaMax optimiser [78], due to its robustness to occasional large gradient estimates. These sometimes occurred in our training procedure when different batches of the time series had very different properties. See Appendix D.2 for its tuning choices. To stabilise optimisation, we also follow [104] and clip gradients using the global L_1 norm.

Variational approximation for θ For $q(\theta; \phi_\theta)$ we use a masked IAF as described in Section 6.2.1. In all of our examples, this alternates between 5 affine layers and random permutations. Each affine transformation is based on a masked feed-forward network of 3 layers with 10 hidden units.

Pre-training We found that pre-training our variational approximation to sensible initial values reduced the training time. A general framework for this is to train $q(\theta; \phi_\theta)$ to be close to the prior, and $q(x|\theta; \phi_x)$ to be close to the observations, or some other reasonable initial value. See Appendix D.2 for details of how we implemented this in our examples.

One of our examples required more complex pre-training. See Appendix D.3 for details. Although our method does sometimes require such non-trivial tuning choices, so do most other competing methods for Bayesian inference of SSMs (see e.g. [130]).

Local side information Our local side information vector s_i is made up of:

- Time t_i .
- Binary variable indicating whether or not $i \in Q$ (i.e. whether there is an observation of x_{t_i}).
- Vector of observations y_{t_i} if $i \in Q$. Replaced by the next recorded observation vector if $i \notin Q$, or by a vector of zeros if there is no next observation.
- Time until next observation (omitted in settings where every i has an observation).
- Binary variable indicating whether $i \in 1:P$ (as our receptive field for s_i can stretch beyond this).

Choice of ℓ' Throughout we use $\ell' = 10$. We found that this relatively short receptive field length for local side information was sufficient to give good results for our examples.

6.6 Experiments

Here we present results of our method applied to several examples. All results were obtained using an NVIDIA Titan XP and an 8 core CPU. Full details of tuning choices used in these experiments are listed in Appendix D.2.

6.6.1 AR(1) Model

We begin by considering the AR(1) model

$$x_{t_{i+1}} = \theta_1 + \theta_2 x_{t_i} + \theta_3 \varepsilon, \quad (6.16)$$

with $\varepsilon \sim \mathcal{N}(0, 1)$ and $x_{t_0} = 10$. We assume observations $y_{t_i} \sim \mathcal{N}(x_{t_i}, 1)$ for $i \in 0:P$, and independent $\mathcal{N}(0, 10^2)$ priors on $\theta_1, \theta_2, \log \theta_3$.

Using this application, we investigate how our method scales with longer data sequences, and how it is affected by the receptive field length ℓ . To judge the accuracy of our variational results we compare it to near-exact posterior inference using an MCMC method for this model described in Appendix D.1.

Effect of observation sequence length We simulated a synthetic dataset for each of four T values (recall $t_P = T$): 5000, 10000, 50000, 100000 under true parameter values $\theta = (5.0, 0.5, 3.0)$ and integer time steps, i.e. $t_i \in \mathbb{Z}_0^+ \forall i$. In such a case, $T = P$. We then inferred θ under the priors specified above, fixing the hyperparameters so that the cost per iteration for each setting is constant.

Figure 6.5a plots the accuracy of the variational approximation for θ against the number of iterations performed. Accuracy is measured as the Maximum Mean Discrepancy (MMD) [61] between the variational approximation and MCMC output (here we use MMD with a Gaussian kernel).

In all cases, variational inference approximates the posterior well. Furthermore, the number of training iterations required remains similar as T increases.

As a further check on the quality of the posterior approximation, Figure 6.5b shows a good match between marginal posteriors for MCMC and variational output for the case $T = 5000$. Here, as for other T values, the 10,000th iteration is achieved after ~ 3 minutes of computation. In comparison, the cost per iteration of MCMC is roughly proportional to T .

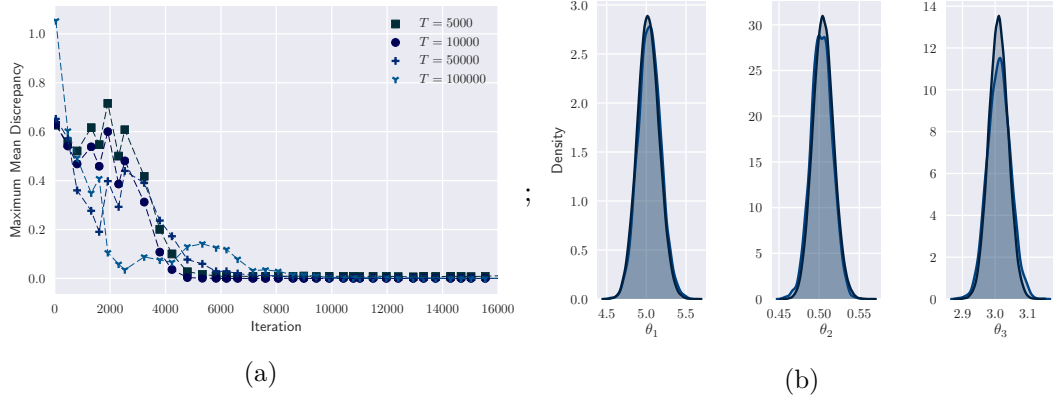


Figure 6.5 (a) Maximum mean discrepancy between variational and MCMC output for θ for the AR(1) model.

(b) Marginal posterior density plots, comparing MCMC (blue) to variational (black) output. The latter is the output after 10,000 iterations.

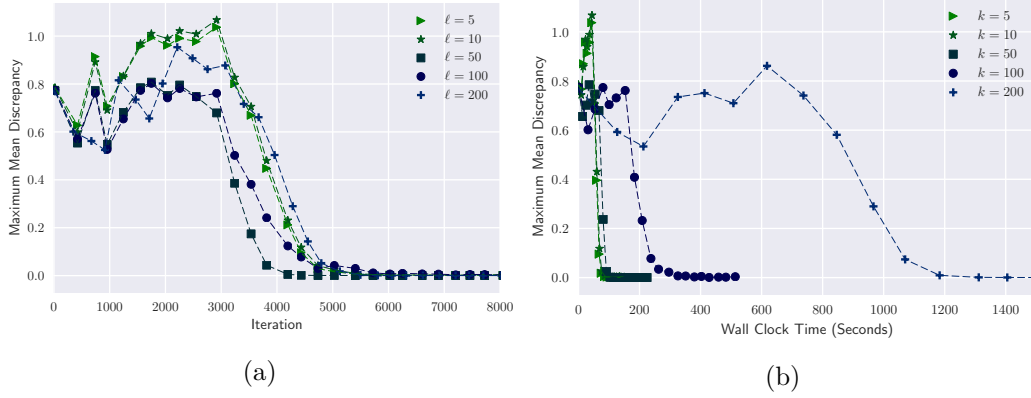


Figure 6.6 Maximum mean discrepancy between variational and MCMC output for θ for the AR(1) model under a range of receptive field lengths ℓ . The horizontal axis shows (a) number of training iterations (b) wall-clock training time.

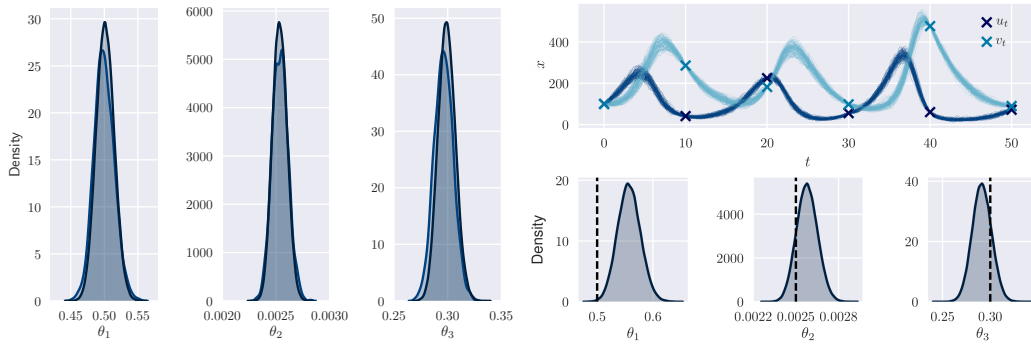


Figure 6.7 Lotka-Volterra results. Left: marginal density plots for setting (a) (dense observations), comparing MCMC (blue) to variational (black) output, after 30,000 training iterations. Right: variational output for setting (b) (sparse observations) - again after 30,000 training iterations. Right (top): 100 samples from variational output for x , with observations displayed as crosses. Right (bottom): marginal density plots of variational output for θ with true values displayed with dashed black lines.

Effect of receptive field length We now consider the $T = 5000$ dataset from the previous example, and investigate the effect of receptive field length ℓ . This test, at least in the simple AR(1) model, helps validate the appropriateness of our assumption of no long-range dependency. Figure 6.6 displays the MMD against both iteration and wall-clock time for $\ell \in \{5, 10, 50, 100, 200\}$. In all cases the variational output converges to a good approximation of the posterior. Convergence takes a similar number of iterations for all choices of ℓ , but wall-clock time per iteration increases with ℓ .

6.6.2 Lotka-Volterra

Next we test our method on short time series with complex dynamics. We use a version of the Lotka-Volterra model, which describes simple predator-prey population dynamics under three events: prey reproduction, predation (in which prey are consumed and predators have the resources to reproduce) and predator death.

Recall from Section 4.3.1, an SDE Lotka-Volterra model is defined by drift and diffusion functions

$$\alpha(x_t, \theta) = \begin{pmatrix} \theta_1 u_t - \theta_2 u_t v_t \\ \theta_2 u_t v_t - \theta_3 v_t \end{pmatrix}, \quad (6.17)$$

$$\beta(x_t, \theta) = \begin{pmatrix} \theta_1 u_t + \theta_2 u_t v_t & -\theta_2 u_t v_t \\ -\theta_2 u_t v_t & \theta_3 v_t + \theta_2 u_t v_t \end{pmatrix}, \quad (6.18)$$

where $x_t = (u_t, v_t)$ represents population sizes of prey and predators. The parameters $\theta = (\theta_1, \theta_2, \theta_3)$ control the rates of the three events described above. We consider a discretised version of this SDE, as described in Section 3.3.1, with $\Delta t = 0.1$ and $(u_0, v_0) = (100, 100)$. We simulated synthetic realisations under parameters $\theta = (0.5, 0.0025, 0.3)$ of x_{t_i} for $i \in 1:500$. We used these to construct two datasets, with observations at (a) $i = 0, 10, 20, \dots, 500$ (‘dense’, i.e. every unit time) (b) $i = 0, 100, 200, \dots, 500$ (‘sparse’, i.e. every ten unit times). We further note the similarity of observation setting (b) to those of the experiments in Section 4.3.1. We assume noisy observations $y_{t_i} \sim \mathcal{N}(x_{t_i}, I_2)$ and independent $\mathcal{N}(0, 10^2)$ priors for $\log \theta_1, \log \theta_2, \log \theta_3$.

Unlike our previous examples, there is a restriction here that u_i and v_i are positive. We enforce this by using a final softplus transformation in our flow for x . Another unusual feature of this example is that we found multiple posterior modes, and needed to pre-train carefully as to control which mode we converged to. See Appendix D.3 for more details of both these issues.

For observation setting (a), we compared our results to near-exact posterior samples from the MCMC algorithm of [57, 47]. This uses a Metropolis-within-Gibbs scheme with carefully chosen proposal constructs. As discussed previously in detail (i.e. Section 4.1.1), designing suitable proposals can be challenging, particularly in sparse observation regimes [148]. Consequently we were unable to use MCMC in setting (b).

Figure 6.7 (left plot) displays the visual similarity between marginal densities estimates from variational and MCMC output in setting (a). The VI output is taken from the 30,000th iteration, after ≈ 10 minutes of computation. Figure 6.7 (right plots) shows variational output for θ and x in setting (b) after ≈ 20 minutes of computation. These results are consistent with the ground truth parameter values and x path. VI using an autoregressive distribution for x has also performed well in a similar scenario (chapter 4, Section 4.3.1), but required more training time (roughly 2 hours).

6.6.3 FitzHugh-Nagumo

Here we test our method on a long time series with an unobserved component. For this purpose we use the FitzHugh-Nagumo model, which models spike-generation dynamics in squid giant axons. A SDE version based on [73, 137] is defined by drift and diffusion functions

$$\alpha(x_t, \theta) = \begin{pmatrix} \theta_1 (-v_t^3 + v_t - w_t + \theta_2) \\ \theta_3 v_t - w_t + 1.4 \end{pmatrix}, \quad (6.19)$$

$$\beta(x_t, \theta) = \begin{pmatrix} \theta_4 & 0 \\ 0 & \theta_5 \end{pmatrix}, \quad (6.20)$$

where $x_t = (v_t, w_t)$ represents the current membrane potential and latent recovery variables.

We consider a discretized version of this SDE, with $\Delta t = 0.1$ and $v_0 = 2, w_0 = 3$. We simulated synthetic data under parameter values $\theta = (2.0, 1.0, 1.5, 0.5, 0.3)$ up to $T = 1,000,000$, recording observations of v_t only at every t_i as to mimic a high-frequency observation scenario. We assume independent observations $y_{t_i} \sim \mathcal{N}(v_{t_i}, 0.1^2)$ and independent $\mathcal{N}(0, 10^2)$ priors for $\log \theta_1, \theta_2, \theta_3, \log \theta_4, \log \theta_5$.

For this longer time series, we found it necessary to modify our training procedure to avoid large gradient estimates. To do so we slowly exposed the model to more of the data over time. We began training using only the first 100,000 data points, and added an extra 100,000 data points every 1000 iterations.

Figure 6.8 displays estimates of the unobserved component w , and marginal density estimates for θ . The results are consistent with the ground truth parameter values and w path. The approximate posterior is sampled after ≈ 180 minutes of computation.

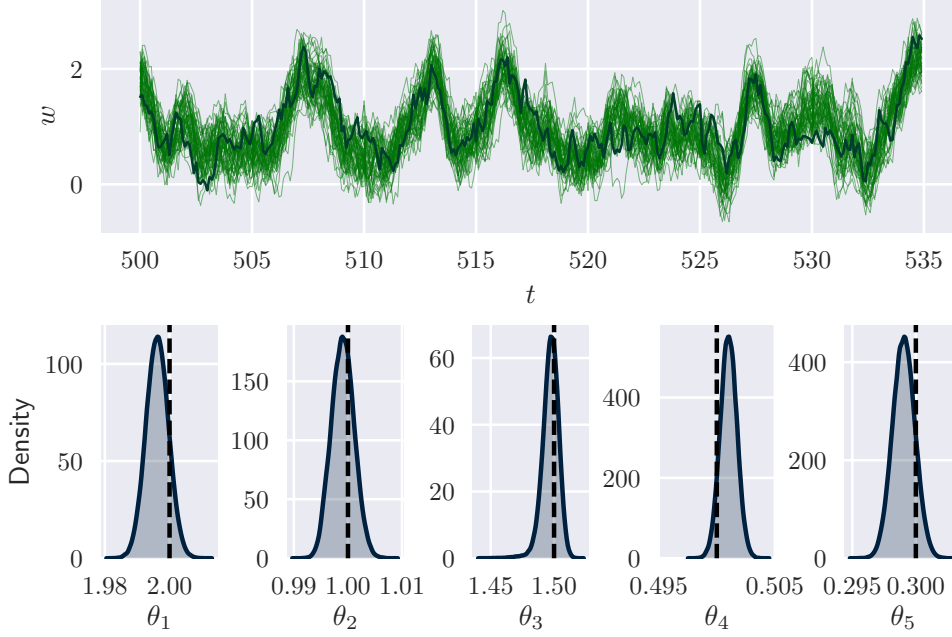


Figure 6.8 Fitzhugh-Nagumo results. Top: 100 samples from the variational posterior for the unobserved coordinate w (light green) and true values (dark green) for a short section of the time series. The horizontal axis shows $t = 0.1i$. Bottom: marginal density plots of variational output for θ with true values displayed with dashed black lines.

6.6.4 Log-Gaussian stochastic volatility

We extend our experiments to the analysis of real data, further illustrating our method on a log-Gaussian stochastic volatility model presented as a discretized SDE with drift and diffusion

$$\alpha(x, \theta) = \begin{pmatrix} \theta_1 r \\ \theta_2 - \theta_3 z \end{pmatrix}, \quad (6.21)$$

$$\beta(x, \theta) = \begin{pmatrix} r e^z & 0 \\ 0 & \theta_4^2 \end{pmatrix}, \quad (6.22)$$

where $x = (r, z)$ is the returns process and latent volatility factor, respectively. Similar discrete-time models have been analysed by Andersen and Lund [6], Eraker [45] and Durham and Gallant [40], but we use the form presented in Golightly and Wilkinson [56].

Similarly to Golightly and Wilkinson [56], we use 1508 weekly observations on the three-month U.S. Treasury bill rate for August 13, 1967 – August 30, 1996, and perform inference under independent $\mathcal{N}(0, 10^2)$ priors for $\theta_1, \theta_2, \log \theta_3, \log \theta_4$ and z_0 . We assume the returns process is fully observed without error and set $\Delta t = 1.0$. Our analysis took 20 minutes on a single

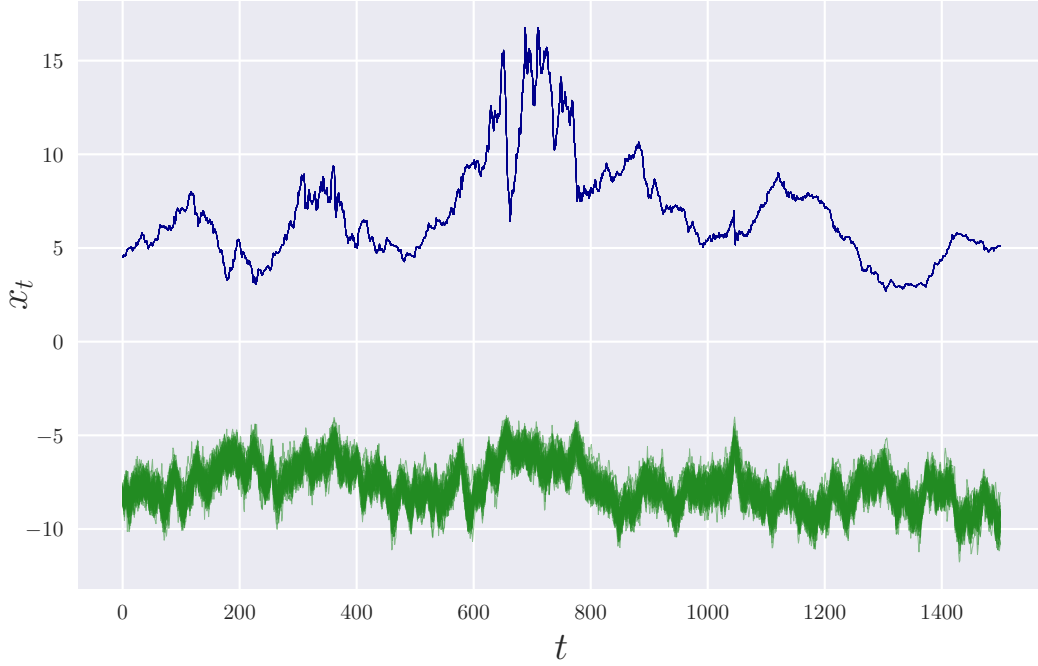


Figure 6.9 Stochastic volatility results. The observed returns process (blue line) and 50 samples from the variational posterior for the latent volatility path (green lines) in the log-Gaussian stochastic volatility model.

GPU. Figure 6.9 shows resulting estimates for the latent volatility path, and Figure 6.10 displays marginal density estimates for parameters θ . We note these posterior estimates are consistent with those obtained from the MCMC analysis in Golightly and Wilkinson [56].

6.7 Application to GP regression and latent force models

As discussed in Chapter 3, one way to think of Gaussian process (GP) regression is through its formulation as a state-space model: a *Gaussian process state-space model* (GPSSM). That is, recall we can represent the solution of GP regression, with assumed zero-mean function, as that of a linear time-invariant (LTI) SDE

$$a_0 f(t) + \sum_{i=1}^m a_i \frac{d^i}{dt^i} f(t) = \varsigma \omega_t, \quad (6.23)$$

where ω_t is a unit *white-noise process*, $\varsigma \in \mathbb{R}$, and a_i , $i = 0, \dots, m$ are some constant, scalar coefficients related to the hyperparameters of $k(\cdot, \cdot)$, the GP covariance kernel. Any method for SSM inference, assuming utilisation of a discretisation scheme, is therefore suitable for a GP regression, given a finitely differentiable covariance kernel (i.e. such that the GP can

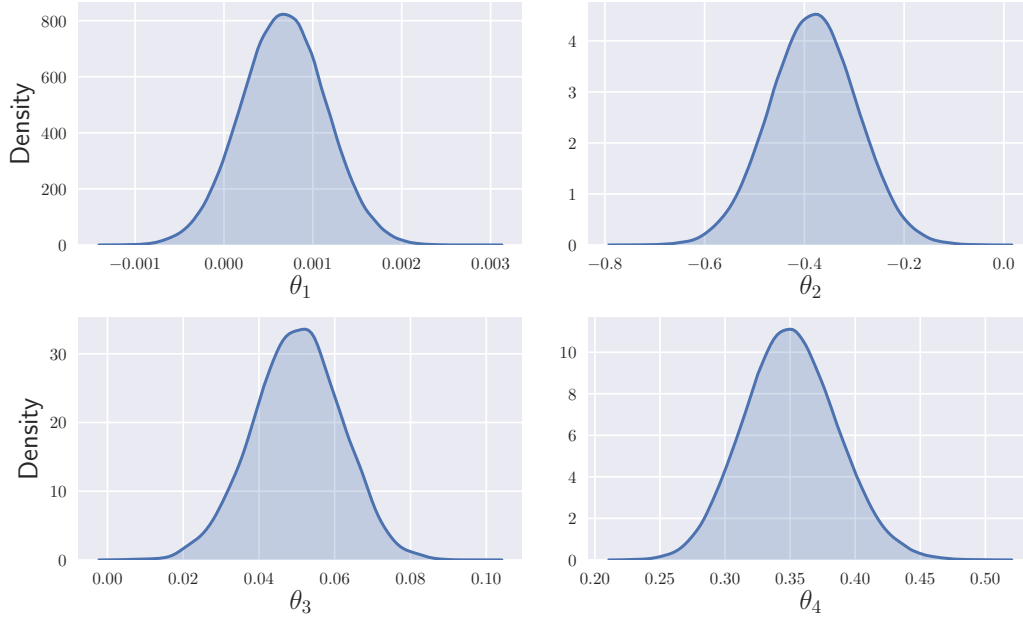


Figure 6.10 Marginal density plots of the variational posterior for θ in the log-Gaussian stochastic volatility model.

be represented as per (6.23) with finite m). Further, it is especially interesting to consider the application of any inference method that: (a) improves on an $\mathcal{O}(D)$ cost; or (b) permits the extension to non-Gaussian observation likelihoods (as the approach outlined here can achieve). As such, we now seek to apply our method for scalable SSM model inference to the problem of a GP regression.

6.7.1 Latent force models

Extending approximate inference for a GP regression, we additionally consider our approach as applied to *latent force models*. Whilst permitting the exact recovery of the GPSSM framework, latent force models are often defined by a *non-linear* differential equation (e.g. one in which $a_{0:m}$ in (6.23) are the output of some non-linear functions). In that sense, applying our approach to latent force models can be seen as simply extending the GPSSM framework to a non-linear class of differential equations, where we can apply our methods to an additional problem set with no extra modifications to our underlying method.

More specifically, latent force models (LFMs) are a class of models that can be used to combine known mechanistic systems with non-parametric representations of unknown forces

(e.g. using a GP). Begin by considering the system defined by the differential equation

$$\alpha_0(x_t, u_t, t; \theta) x_t + \sum_{i=1}^{n-1} \alpha_i(x_t, u_t, t; \theta) \frac{d^i}{dt^i} x_t = u_t, \quad (6.24)$$

where u_t is a *forcing term* and $\alpha_{0:n}(\cdot; \theta)$ are deterministic functions, with parameters θ . In a typical inference formulation, the forcing term u_t typically requires estimation given only observations of the system state, x_t (often according to e.g. (6.3)). It may further be the case that $\alpha_{0:n}(\cdot; \theta)$ are unknown, requiring simultaneous estimation. Parametric forms for u_t may be used to estimate the state; however, placing a GP prior over u_t gives rise to the LFM [5]. The resulting model can be transformed into a regression problem, and a probabilistic solution for x_t and u_t can be inferred.

As discussed, a special case of a LFM (6.24) is where $\alpha_{0:n}(\cdot; \theta)$ possesses the deterministic dynamics of a GP, and u_t is a GP with white-noise covariance. Defining (6.24) in this way returns a GPSSM as defined in (6.23). For example, define

$$f(t) \sim \mathcal{N}(0, K(t, t)), \quad (6.25)$$

where $K(t, t)$ is a Matérn-3/2. It then follows that [64]

$$\varphi^m f(t) + \sum_{i=1}^m \binom{m}{i} \varphi^{m-i} \frac{d^i}{dt^i} f(t) = \varsigma \omega_t, \quad (6.26)$$

where ω_t is a unit white-noise process with $\varsigma = \sigma_c^2 (2\varphi)^{2m-1} (m!) / (2m-2)!$ and $\varphi = \lambda^{-1} \sqrt{2m-2}$. Further, as described in Section 3.4, σ_c^2 is the variance and λ is the characteristic length-scale.

LFMs have been shown as effective ways to approximate latent forcing terms, constraining solutions with the defined mechanistic dynamics of the system. Problems where the dynamics are partially known, but where one or more terms $\alpha_{0:n}(\cdot; \theta)$ are non-linear occur in many areas, including biomechanics [14], population modelling [88], structural health monitoring [152] and control systems [31].

Inference for LFMs: Related work As discussed, closed-form solutions for LFMs can only be inferred exactly in the case that the underlying differential equation (6.24) is linear time-invariant (LTI). Alvarez et al. [5] use linearisation of terms in non-linear models, which greatly simplifies the mechanistic aspect of the model, while Hartikainen et al. [65] use non-linear filtering approaches for approximate inference. In the latter case, the results are effective for known parameters, but the sequential nature of the inference scheme leads to challenges in joint parameter estimation. Such challenges are the inspiration for the work by

Durrande et al. [42]. In the case where parameters are also unknown, sequential methods can struggle to perform joint inference effectively. An alternative inference scheme using Kalman filtering and Rauch-Tung-Streifel smoothing was introduced by Hartikainen and Särkkä [64], which infers the state sequentially.

6.7.2 Scalable inference for non-linear latent force models

We now present the approach of Section 6.4 applied to a inference on system with known dynamics, some unknown forcing term and model parameters, given partial and/or noisy observations of a system. That is, we present our methodology in the context of non-linear LFMs, but note this choice – as discussed – permits the flexibility to additionally express a GP regression.

In order to apply our approach for SSM inference, we must discretise the otherwise continuous-time LFM and approximate the likelihood. Indeed, LFMs (and GPSSMs) suppose an identical inference challenge to (6.1) upon definition of the discrete-time likelihood approximation. In that sense, defining a tractable approximating density is the principal challenge in considering this extension. As such, begin by supposing a LFM (6.24) defined on a uniform grid of discrete times up to $t_P = T$, with regular time-step Δt . We assume this process is observed according to equation (6.3). As detailed in Section 3.4.2, we can construct the companion form SDE of (6.23), according to

$$\frac{d\chi_t}{dt} = F\chi_t + L\zeta\omega_t, \quad (6.27)$$

$$\implies d\chi_t = F\chi_t dt + L\zeta dw_t, \quad (6.28)$$

where, in order to represent (6.24), we consider an *augmented* system state

$$\chi_t = \begin{pmatrix} x_t \\ \vdots \\ \frac{d^{n-2}x_t}{dt^{n-2}} \\ \frac{d^{n-1}x_t}{dt^{n-1}} \\ u_t \\ \vdots \\ \frac{d^{m-2}u_t}{dt^{m-2}} \\ \frac{d^{m-1}u_t}{dt^{m-1}} \end{pmatrix}, F = \begin{pmatrix} 0 & 1 & \dots & 0 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & & 0 & 1 & \vdots & & \ddots & \vdots \\ -\check{\alpha}_0 & \dots & -\check{\alpha}_{n-2} & -\check{\alpha}_{n-1} & -1 & \dots & \dots & 0 \\ 0 & \dots & \dots & 0 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & 0 & 1 \\ 0 & \dots & \dots & 0 & -a_0 & \dots & -a_{m-2} & -a_{m-1} \end{pmatrix}, \quad (6.29)$$

where $L = (0, \dots, 0, 0, \dots, 1)^T$. Here χ_t is the augmented system state containing *both* the $n - 1$ derivatives of x_t and the $m - 1$ derivatives of u_t . Further, $\check{\alpha}_{0:n-1} = \alpha_{0:n-1}(x_t, u_t, t; \theta)$

and $a_{0:m-1}$ are the scalar constants derived from the GP prior for u_t . In practice, F may be a function of both time and the system state, which we suppress for notational simplicity. Should we wish to represent a GP regression only, the augmented state reduces to (3.35), Section 3.4.2.

Upon discretization, the posterior then follows from (6.1)

$$p(x, u, \theta | y) \propto p(\chi_{t_0}) \prod_{i=0}^{P-1} p(\chi_{t_{i+1}} | \chi_{t_i}, \theta) \prod_{i \in Q} p(y_{t_i} | \chi_{t_i}, \theta). \quad (6.30)$$

As discussed in Section 3.4.2, using the Euler-Maruyama scheme for likelihood estimation of the companion-form SDE is challenging, due to the zero elements on the leading diagonal of the diffusion matrix. One solution is the use of an alternate discretisation scheme. We note that, should we have solely considered GPSSMs (i.e. where the differential equation was linear), we could have accessed the likelihood directly; however extension to *non-linear* LFMs necessitates numerical methods.

There are many approaches for likelihood estimation in non-linear systems, including the *extended Kalman filter* and sequential Monte Carlo [126]. For our purpose, we again approximate the transition density as Gaussian, but build moments using the prediction dynamics of the continuous-time *unscented Kalman filter* (UKF) (which is described in Appendix D.4). Here the transition density is given as

$$p(\chi_{t_{i+1}} | \chi_{t_i}, \theta) = \mathcal{N}(\chi_{t_{i+1}}; M_{t_{i+1}}, V_{t_{i+1}}), \quad (6.31)$$

where $M_{t_{i+1}}$ and $V_{t_{i+1}}$ are the one-step prediction moments computed using the UKF (the calculation of $M_{t_{i+1}}$ and $V_{t_{i+1}}$ is further detailed in Appendix D.4.1). Likelihood approximation in this way then permits the direct application of a local inverse autoregressive flow (6.8), along with ELBO estimation (6.14) and optimisation algorithm (as detailed in Section 6.4.1) to approximate the joint system state χ_t and parameters θ .

6.7.3 Experiments

This section demonstrates use-cases and flexibility of the proposed approach as applied to both a GP regression and non-linear LFMs. We begin by applying the approach to a tractable GP regression in order to demonstrate the ability to approximate samples and apply model criticism, comparing the approximations with the closed-form posterior in order to quantify accuracy. In the scope of GP regression, we further demonstrate that the method can be easily extended to problems with non-Gaussian observation likelihoods – an otherwise

non-trivial challenge. We then apply the approach to simulated and real non-linear LFMs, demonstrating joint inference for states, forcing terms and model parameters.

In all of the experiments that follow, we use a learning rate of 5×10^{-3} , and choose $\nu = 2d$, where d is the dimension of augmented latent state. Unscented filtering updates for likelihood approximation were implemented using the dynamics as described Särkkä [125].

Tractable GP regression

We start by testing our approach as applied to a tractable GP regression. For our data, we simulate from a zero-mean, Matérn-3/2 GP, with $\lambda = 2.0$ and $\sigma_c^2 = 1.0$ (both assumed known), on a regular grid of times separated by time-step of $\Delta t = 0.1$, up to $T = 40$. The regression problem is defined such that χ_t is the joint state that has dynamics defined in (6.26) with $m = 2$. We choose to observe this process at $i = 0, 20, \dots, 400$, subject to additive Gaussian noise $y_{t_i} \sim \mathcal{N}(H\chi_{t_i}, \sigma^2)$. Recall H is the binary mapping vector permitting recovery of the GP from the augmented state (equation (3.36), Section 3.4.2). We further let $\sigma^2 = 0.1$, which we assume known.

For the basis of our comparison, we use a kernel-based two-sample test [61] (based on the MMD as presented in Section 6.6.1) to compare samples from the trained variational approximation with samples from the true posterior. We apply the two-sample test using MMD with a Gaussian Kernel as a similarity metric between the approximate and true posterior. The MMD² value for the approximation of Matérn-3/2 GP fit is shown against the number of training steps in Figure 6.11. The plot demonstrates that as training increases, so does the quality of approximation. The black solid line along the lower part of the plot indicates the threshold above which the null hypothesis can be rejected with a 95% confidence. The trend observed thus indicates that with training the approximation is tending towards samples indistinguishable from the truth. For visual reference, two samples from the true posterior and approximation further demonstrate their similarity.

Non-Gaussian Likelihoods

We apply the regression problem representing a latent GP with Matérn-3/2 covariance as in the previous section, but condition on a simulated set of count data to demonstrate that the approach can be easily extended to problems with non-Gaussian likelihoods.

For our data, we again simulate from a zero-mean, Matérn-3/2 GP, with $\lambda = 2.0$ and $\sigma_c^2 = 1.0$ (both assumed known), on a regular grid of times separated by time-step of $\Delta t = 0.1$, up to $T = 40$, choosing to observe this process at $i = 0, 10, \dots, 400$. The regression problem is again defined such that χ_t is the joint state that has dynamics defined in (6.26) with $m = 2$.

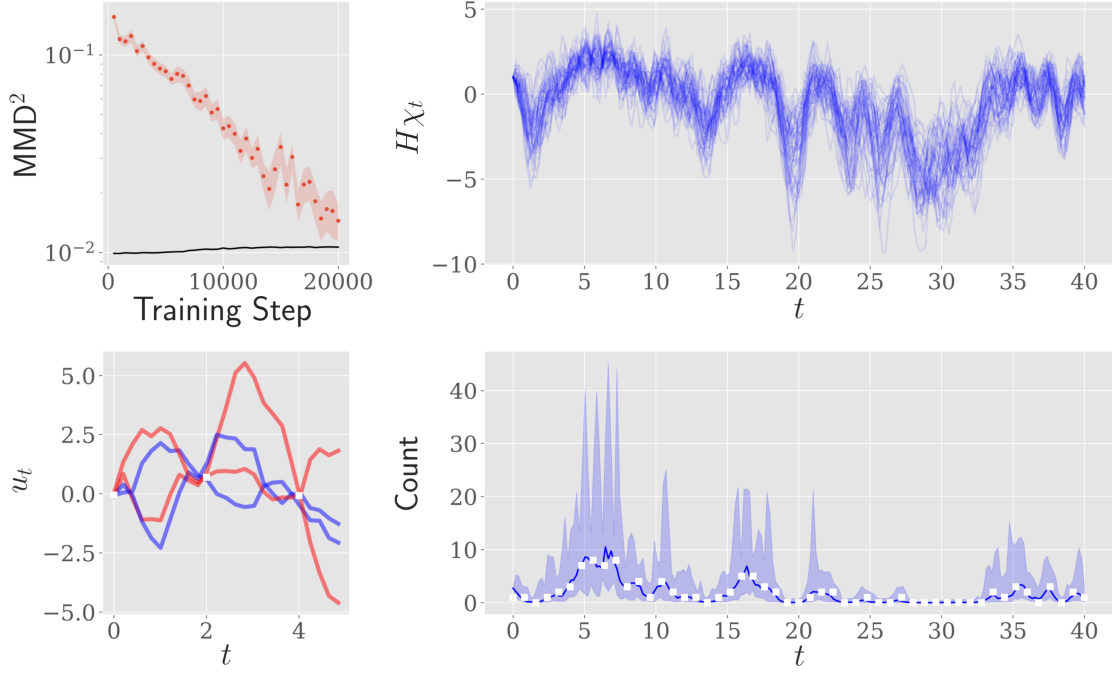


Figure 6.11 Plots of the GP regression examples for a Gaussian [left] and non-Gaussian [right] observation likelihood. Top left: maximum mean discrepancy (MMD^2) scores comparing samples from the variational approximation of GP with Matérn- $3/2$ covariance with samples the true posterior, plotted against training step. The black line indicates the threshold under which the null hypothesis can be accepted with 95% confidence. Bottom left: two samples from the true posterior (red) and variational approximation (blue) conditioned on some observations (white). Top right: samples from a variational approximation of a latent GP conditioned on count data. Bottom right: predictive density and count data plotted with 95% confidence intervals (shaded).

Observations are made available according to

$$p(y_{t_i} | \chi_{t_j}) = \mathcal{Pois}(y_{t_j} | \exp(H\chi_{t_j})),$$

where $\mathcal{Pois}(\cdot)$ is the density of the Poisson distribution. The simulated data and the variational distribution are shown in Figure 6.11, along with the approximated predictive density, which demonstrates that the approximation can capture the main features and uncertainty of the system. The negative log predictive density (NLPD) for the fit using the proposed approach is -0.12218 , versus -0.16576 for a GP fit with a Laplace approximation [110].

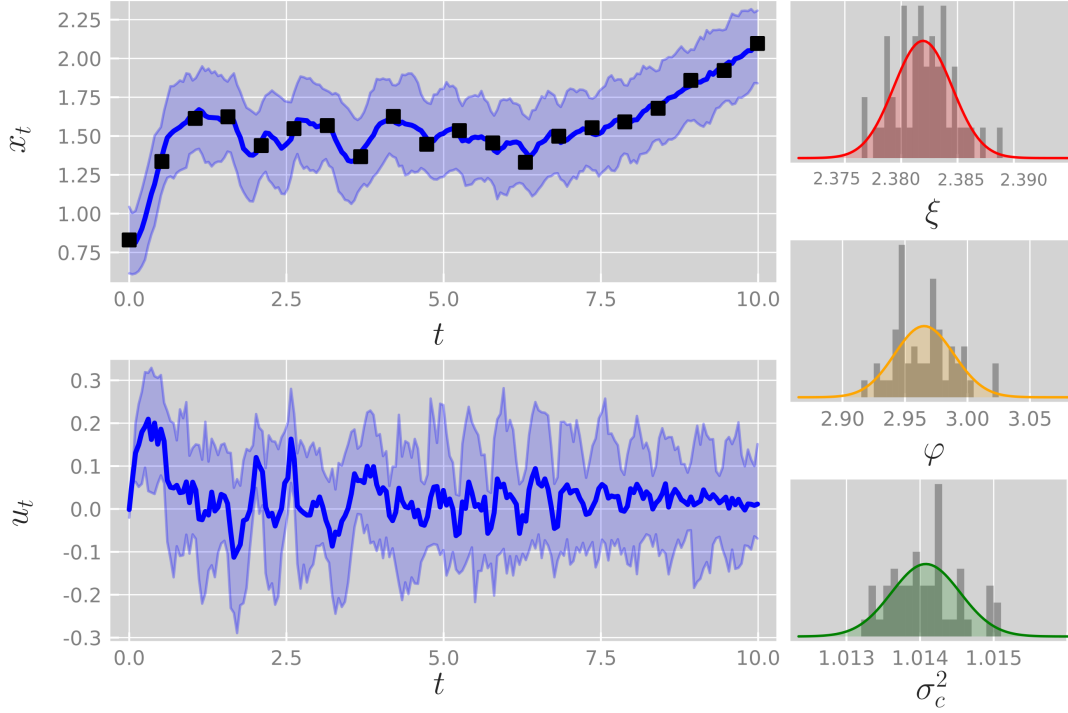


Figure 6.12 Latent state, x_t , and forcing term, u_t , estimates using the proposed approach, conditioned on noisy observations of x_t indicated by black squares. The shaded area indicates the central 95% sample quantile about the mean (solid blue). Plots on the right indicate the estimated marginals of the parameters and binned samples.

Toy problem

In this example, we jointly demonstrate inference of a latent input function on a non-linear ODE, along with parameters of the model and the GP kernel. For this purpose we consider a toy example with sinusoidal dynamics of an observable state, and place a Matérn-3/2 GP prior over the unknown input

$$\frac{d}{dt}x_t = -\frac{2}{3}\sin(\xi x_t) + u_t. \quad (6.32)$$

Parameters from both the latent state dynamics and input covariance $\theta = (\xi, \sigma_c^2, \varphi)'$ are assumed unknown. The approximate posterior was conditioned on observations generated from a sample solution to (6.32), with independent additive Gaussian noise, with variance $\sigma^2 = 0.1$ (assumed known). Here we perform inference using independent $\mathcal{N}(0, 10^2)$ priors for $\xi, \log \sigma_c^2, \log \varphi$.

Figure 6.12 shows the inferred posterior state and latent forcing term inferred using the variational approach. The marginals for the inferred parameters are also shown. We observe that the underlying forcing term has largest effect in the early parts of the model, $t < 5.0$, which corresponds to deviation from the fixed dynamics observed in y .

Gene Expression

In the final experiment, we consider a multi-output system using real-world data. We consider the transcriptional regulation model and corresponding data in Barenco et al. [14]: an ODE describing the dynamics of target gene expression that are regulated by an unobserved transcription factor, u_t . For each gene in the dataset, $x_t^{(d)}$, the dynamics can be described as

$$\frac{d}{dt}x_t^{(d)} = a_d - b_d x_t^{(d)} + s_d \frac{u_t}{\gamma_d + u_t}, \quad (6.33)$$

where each $x_t^{(d)}$ is the d^{th} gene (or dimension) of x_t . We assume the gene expressions are observed with independent additive Gaussian noise, with variance $\sigma^2 = 0.25$, and place a GP prior over $\log u_t$, as in Titsias and Lawrence [135]. Model parameters for each output include the basal transcription rate, a_d , decay rate, b_d , and sensitivity, s_d , which are all unknown. γ_d is the Michaelis constant specific for each gene. Additional shared parameters are those of the GP covariance function, characterising u_t .

We perform inference of the gene expression problem for 5 observed genes: DDB2, BIK, TNFRSF10b, CIP1/p21, and p26^{sen1}. A Matérn-3/2 GP prior is placed over the log transcription factor term, $\log u_t$, and we further estimate $\{a_d, b_d, s_d \mid d = 1, \dots, 5\}$, and GP parameters, φ and σ_c^2 . A softplus bijector was placed on the output paths representing x_d to enforce positivity.

Figure 6.13 show the inferred (unscaled) transcription factor concentration, u_t , and two of the gene expression states: DDB2 and p26^{sen1}. We observe that the estimates of observable states, $x_t^{(d)}$, capture the observations and general form.

We observe that the proposed approach can easily extend to multidimensional states, and represent multi-output problems. In practice, this is similar to process convolution representation of such problems [5], however we are able to perform inference over non-linear models, while jointly estimating system parameters.

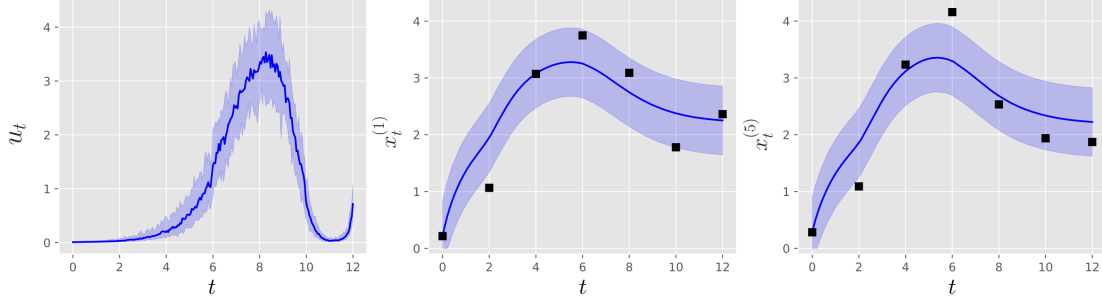


Figure 6.13 Inferred transcription factor concentration (left) and predicted gene expression for DDB2 (centre) and p26 sesn1 (right). Black squares indicate measured gene expression, and shaded region represented the 95th sample quantile about the mean.

6.8 Conclusion

In this chapter we have presented a variational inference method for SSMS based on local inverse autoregressive flows. This class of generative models have been successfully designed to approximate complex dependence in the conditional posterior $p(x|\theta, y)$ and scale favourably to a long time series. In particular, our method permits optimization with mini-batch gradient descent. Conversely, this is not something easily achieved with traditional iterative approaches, such as filtering methods or sequential Monte Carlo. We demonstrate that our method works well on many applications, with challenges including: an unobserved state component, sparse observation times, and a large number of observations. Further, we have outlined how this approach can be extended to approximations of a GP regression (both with Gaussian and non-Gaussian observed data likelihoods) and non-linear LFMs; in turn, we have presented a method that is simultaneously able to perform inference in many challenging problem settings.

There are, however, some limitations to the approach. Aside from the inability to capture long-range dependencies in x , the method has a tendency to over-fit (as, of course, is common with variational inference). This can be seen, for example, in Figure 6.13, where we observe narrow error bars in the latent GP for lower values of t . We further witness this in Figure 6.7, where the approximate parameter posteriors (left) are marginally more concentrated than their MCMC counterparts.

Future work could potentially investigate changing several aspects of the flow: alternating affine transformations with order-reversing permutations; using recently proposed variations such as Durkan et al. [41]; allowing some long-range dependence using a multi-scale architecture. It would be further interesting to experiment with more robust (or automated) initialisation techniques – especially as applied to a long time series. Whilst these problems

are endemic to most methods, we found it necessary to implement careful pre-training as to ensure numerical stability in more complex problems.

Chapter 7

Conclusions

In this thesis we have studied the problem of variational inference for a class of stochastic process models. Our approaches are designed to solve many of the practical challenges associated with SDE, SSM and GP posterior inference. The contribution of each chapter is briefly summarised below.

Chapter 4 proposed a solution to the problem of approximating a conditioned SDE in the presence of sparse observations and non-linear dynamics. Here we approximated the conditioned process by learning moments of a Gaussian transition density with a recurrent neural network on a discrete grid of times. Using this approach we successfully performed approximate inference for a broad class of SDEs, automating the task of bridge construct design. Furthermore, Ward and Álvarez [144] demonstrated this approach is capable of producing conditioned paths statistically indistinguishable from the true posterior for a tractable SDE model. We additionally demonstrated the possibility of approximate parameter inference, recovering accurate estimates in only a hours of computation on a desktop CPU. In contrast to this approach, MCMC methods require more tuning choices and can take days to run [147].

Chapter 5 recognised that in using a discretisation scheme, the approach of Chapter 4 introduced undesirable approximation error to the posterior. Furthermore, unlike the true conditioned process, this method learned a diffusion matrix in the variational posterior whose number of terms scales quadratically with the state dimension. To address these challenges, we again used a recurrent neural network – but only to parametrise the drift function of a continuous-time approximating SDE, which uses the unaltered diffusion term of the target process. This approach then permitted evaluation of our variational objective using a *path integral*. In this way, we were then able to approximate ideal gradients for inference under

the continuous-time model, approximating the exact solution’s intractable drift function – as opposed to calculating gradients of the approximate solution. We successfully used this approach to perform Bayesian inference for a broad class of SDEs: from a 16-dimensional chaotic system, to a novel stochastic model of infectious disease.

Chapter 6 began by considering a method for SSM inference based on local inverse autoregressive flows. This generative model was simultaneously able to approximate complex dependence in the conditional posterior $p(x|\theta, y)$ and scale favourably to a long time series, permitting optimization with mini-batch gradient descent. Conversely, this is not something easily achieved with traditional iterative approaches, such as filtering methods or sequential Monte Carlo, which typically require one full pass through the data at each iteration. We then outlined how this approach could be extended to approximations of a GP regression (both with Gaussian and non-Gaussian observed data likelihoods) and non-linear LFMs; in turn, we presented a method that is simultaneously able to perform inference in many challenging problem settings, with difficulties including: an unobserved state component, sparse observation times, a large number of observations and highly non-linear latent dynamics.

7.1 Outlook

Each of our developed methods proposes solutions to the problems posed by sampling-based Bayesian techniques. Whilst it more obvious when one should consider a variational framework, it is not immediately clear which of our methods should be used for a given problem type. As such, we briefly consider each of approaches, and comment on their applicability.

The approach of Chapter 4 is a remarkably simple way to model complex non-linear SDE systems (including GP regression [144]). It has, unlike approaches of Chapters 5 and 6, few significant tuning and/or pre-training choices. It is therefore a useful, pragmatic tool to rapidly prototype competing models and derive approximate parameter distributions than can effectively recover posterior modes.

As discussed, however, Chapter 4 is of limited theoretical appeal: it modifies the diffusion matrix, calculates gradient of the approximate solution (via the Euler-Maruyama discretisation of the SDE likelihood) and is unlikely to have a well-behaved continuous-time limit [86]. Chapter 5 therefore represents a more elegant solution to modelling SDEs and related systems. However, it is in pursuit of theoretical appeal that the approach, in comparison to Chapter 4, likely loses its robustness to optimization. Indeed, the implementation of a numerical solution for the augmented system is a non-trivial tuning choice – and one which could have significant consequences for the approximating solution. It is, however, plausible that future work could

successfully extend this approach to make use of either: (1) an adaptive solution or (2) the stochastic adjoint method. Both extensions could yield an approach which supersedes that of Chapter 4 – both in speed and approximation accuracy – with a comparable optimisation difficulty.

Chapters 4 and 5, in spite of their benefits, have inherently sequential generative procedures. Whilst often faster than competing sampling-based schemes, each iteration still requires simulation of the full latent path using recurrent neural architectures. These approaches, therefore, cannot be parallelised easily, and are not well-suited to extremely long data sequences. As a solution, Chapter 6 uses a novel flow-based generative model that is able to model complex non-linearities and permit mini-batch stochastic optimization. Unlike Chapters 4 and 5, the generative procedure can be performed in parallel, resulting in a very fast variational approach. Unfortunately, however, training flow-based generative models in extremely high-dimensions is an inherently difficult optimisation task. Further, the data seldom displays homogeneous behaviour across the entire sequence, greatly increasing early-training variance of mini-batch loss estimates. In practice, whilst this method effectively combines representational power with speed, allowing inference on longer data sequences than ever before, the resulting algorithm often needs careful fine-tuning and/or initialisation.

Bibliography

- [1] Abramowitz, M. and Stegun, I. A. (1965). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*.
- [2] Aicher, C., Putcha, S., Nemeth, C., Fearnhead, P., and Fox, E. B. (2019). Stochastic gradient MCMC for nonlinear state space models. *arXiv preprint arXiv:1901.10568*.
- [3] Albert, J. (2009). *Bayesian Computation with R*. Springer Publishing Company, Incorporated.
- [4] Allen, L. J. S. (2010). *An introduction to stochastic processes with applications to biology*. CRC Press.
- [5] Alvarez, M. A., Luengo, D., and Lawrence, N. D. (2013). Linear latent force models using Gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*.
- [6] Andersen, T. G. and Lund, J. (1997). Estimating continuous-time stochastic volatility models of the short-term interest rate. *Journal of Econometrics*.
- [7] Andersson, H. and Britton, T. (2000). *Stochastic Epidemic Models and Their Statistical Analysis*. Springer-Verlag.
- [8] Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society Series B*.
- [9] Archambeau, C., Cornford, D., Oppel, M., and Shawe-Taylor, J. (2007). Gaussian process approximations of stochastic differential equations. *Proceedings of Machine Learning Research*.
- [10] Archambeau, C., Oppel, M., Shen, Y., Cornford, D., and Shawe-Taylor, J. S. (2008). Variational inference for diffusion processes. In *Advances in Neural Information Processing Systems*.
- [11] Archer, E., Park, I. M., Buesing, L., Cunningham, J., and Paninski, L. (2016). Black box variational inference for state space models. In *International Conference on Learning Representations (ICLR) 2016, Workshops*.
- [12] Arnold, L. (1992). *Stochastic Differential Equations: Theory and Applications*. Krieger.
- [13] Baguelin, M., Hoek, A. J. V., Jit, M., Flasche, S., White, P. J., and Edmunds, W. J. (2010). Vaccination against pandemic influenza a/h1n1v in england: A real-time economic evaluation. *Vaccine*.
- [14] Barenco, M., Tomescu, D., Brewer, D., Callard, R., Stark, J., and Hubank, M. (2006). Ranked prediction of p53 targets using hidden variable dynamic modeling. *Genome biology*.

- [15] Batz, P., Ruttor, A., and Oppel, M. (2016). Variational estimation of the drift for stochastic differential equations from the empirical density. *Journal of Statistical Mechanics: Theory and Experiment*.
- [16] Beal, M. and Ghahramani, Z. (2002). The variational bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. *Statistics*.
- [17] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*
- [18] Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo.
- [19] Bhattacharya, R. and Waymire, E. (2009). *Stochastic Processes with Applications*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics.
- [20] Bierkens, J., Fearnhead, P., and Roberts, G. (2019). The zig-zag process and super-efficient sampling for bayesian analysis of big data. *The Annals of Statistics*.
- [21] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc.
- [22] Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*.
- [23] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*.
- [24] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.
- [25] Boys, R. J., Wilkinson, D. J., and Kirkwood, T. B. L. (2008). Bayesian inference for a discretely observed stochastic kinetic model. *Statistics and Computing*.
- [26] Candela, J. Q. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *JMLR*.
- [27] Cappé, O., Moulines, E., and Ryden, T. (2010). *Inference in Hidden Markov Models*. Springer.
- [28] Casella, G. and Berger, R. (2001). *Statistical Inference*. Duxbury Resource Center.
- [29] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*.
- [30] Coddington, E. A. and Levinson, N. (1955). *Theory of ordinary differential equations*. Tata McGraw-Hill Education.
- [31] Conte, G., Moog, C. H., and Perdon, A. M. (1999). Nonlinear control systems: An algebraic setting.
- [32] Cox, J. C., E, I. J., and Ross, S. (1985). A theory of the term structure of interest rates. *Econometrica*.
- [33] Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Eötvös Loránd University, Hungary*.

- [34] Del Moral, P. and Murray, L. M. (2015). Sequential Monte Carlo with highly informative observations. *SIAM/ASA Journal on Uncertainty Quantification*.
- [35] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*.
- [36] Doucet, A., de Freitas, N., and Gordon, N. (2001). *An Introduction to Sequential Monte Carlo Methods*. Springer New York.
- [37] Drovandi, C., Everitt, R. G., Golightly, A., and Prangle, D. (2019). Ensemble MCMC: Accelerating pseudo-marginal MCMC for state space models using the ensemble Kalman filter. *arXiv preprint arXiv:1906.02014*.
- [38] Duncker, L., Böhner, G., Boussard, J., and Sahani, M. (2019). Learning interpretable continuous-time models of latent stochastic dynamical systems. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR.
- [39] Dureau, J., Kalogeropoulos, K., and Baguelin, M. (2013). Capturing the time-varying drivers of an epidemic using stochastic dynamical systems. *Biostatistics*.
- [40] Durham, G. B. and Gallant, A. R. (2002). Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*.
- [41] Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. *arXiv preprint arXiv:1906.04032*.
- [42] Durrande, N., Adam, V., Bordeaux, L., Eleftheriadis, S., and Hensman, J. (2019). Banded matrix operators for Gaussian Markov models in the automatic differentiation era. In *Proceedings of Machine Learning Research*, Proceedings of Machine Learning Research. PMLR.
- [43] Elliott, R. J., Aggoun, L., and Moore, J. B. (2008). *Hidden Markov models: estimation and control*. Springer.
- [44] Endo, A., van Leeuwen, E., and Baguelin, M. (2019). Introduction to particle Markov chain Monte Carlo for disease dynamics modellers. *Epidemics*.
- [45] Eraker, B. (2001). MCMC analysis of diffusion models with application to finance. *Journal of Business & Economic Statistics*.
- [46] Fasiolo, M., Pya, N., and Wood, S. N. (2016). A comparison of inferential methods for highly nonlinear state space models in ecology and epidemiology. *Statistical Science*.
- [47] Fuchs, C. (2013). *Inference for Diffusion Processes: With Applications in Life Sciences*. Springer Science & Business Media.
- [48] Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.
- [49] Gamerman, D. and Lopes, H. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference, Second Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.
- [50] Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. Chapman and Hall/CRC.

- [51] Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*.
- [52] Gikhman, I. and Skorokhod, A. (1972). *Stochastic Differential Equations*. Springer-Verlag.
- [53] Gilks, W., Richardson, S., and Spiegelhalter, D. (1995). *Markov Chain Monte Carlo in Practice*. Taylor & Francis.
- [54] Gillespie, D. T. (2000). The chemical Langevin equation. *The Journal of Chemical Physics*.
- [55] Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*.
- [56] Golightly, A. and Wilkinson, D. J. (2006). Bayesian sequential inference for nonlinear multivariate diffusions. *Statistics and Computing*.
- [57] Golightly, A. and Wilkinson, D. J. (2008). Bayesian inference for nonlinear multivariate diffusion models observed with error. *Computational Statistics & Data Analysis*.
- [58] Golightly, A. and Wilkinson, D. J. (2011). Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface focus*.
- [59] Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
- [60] Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*.
- [61] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*.
- [62] Grewal, M. and Andrews, A. (2001). Kalman filtering: theory and practice using matlab. *New York: John Wiley and Sons*.
- [63] Griffiths, D. and Higham, D. (2010). *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. Springer.
- [64] Hartikainen, J. and Särkkä, S. (2010). Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *2010 IEEE international workshop on machine learning for signal processing*.
- [65] Hartikainen, J., Seppanen, M., and Särkkä, S. (2012). State-space inference for non-linear latent force models with application to satellite orbit prediction. *arXiv preprint arXiv:1206.4670*.
- [66] Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception*. Elsevier.
- [67] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. AUAI Press.
- [68] Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable Variational Gaussian Process Classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR.

- [69] Hirt, M. and Dellaportas, P. (2019). Scalable Bayesian learning for state space models using variational inference with SMC samplers. In *Artificial Intelligence and Statistics*.
- [70] Horst, R. (2002). *Introduction to Global Optimization (Nonconvex Optimization and Its Applications)*. Springer-Verlag.
- [71] Jackson, C., Vynnycky, E., Hawker, J., Olowokure, B., and Mangtani, P. (2013). School closures and influenza: systematic review of epidemiological studies. *BMJ Open*.
- [72] Jazwinski, A. H. (1970). In *Stochastic Processes and Filtering Theory*, Mathematics in Science and Engineering. Elsevier.
- [73] Jensen, A. C., Ditlevsen, S., Kessler, M., and Papaspiliopoulos, O. (2012). Markov chain Monte Carlo approach to parameter estimation in the FitzHugh-Nagumo model. *Physical Review E*.
- [74] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An introduction to variational methods for graphical models. *Journal of Machine Learning*.
- [75] Jordan, R., Kinderlehrer, D., and Otto, F. (1998). The variational formulation of the Fokker–Planck equation. *SIAM journal on mathematical analysis*.
- [76] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*. SPIE.
- [77] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*.
- [78] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR) 2015*.
- [79] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*.
- [80] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR) 2014*.
- [81] Kloeden, P. E. and Platen, E. (1992). *Numerical solution of stochastic differential equations / Peter E. Kloeden, Eckhard Platen*. Springer-Verlag Berlin, New York.
- [82] Knowles, D. A. and Minka, T. (2011). Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems 24*.
- [83] Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017). Automatic differentiation variational inference. *Journal of Machine Learning Research*.
- [84] Lamba, H. (2003). An adaptive timestepping algorithm for stochastic differential equations. *Journal of Computational and Applied Mathematics*.
- [85] Levin, D. A., Peres, Y., and Wilmer, E. L. (2006). *Markov chains and mixing times*. American Mathematical Society.
- [86] Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. (2020). Scalable gradients for stochastic differential equations.

- [87] Liu, J. S. (1996). Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*.
- [88] Liu, P.-P. (2010). An analysis of a predator–prey model with both diffusion and migration. *Mathematical and Computer Modelling*.
- [89] Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*.
- [90] Lorenz, E. N. (1996). Predictability: A problem partly solved. In *Proc. Seminar on predictability*.
- [91] Ma, X. and Hovy, E. (2019). Macow: Masked convolutional generative flow. *arXiv preprint arXiv:1902.04208*.
- [92] McElreath, R. (2016). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. CRC Press.
- [93] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*.
- [94] Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2019). Monte carlo gradient estimation in machine learning.
- [95] Neiswanger, W., Wang, C., and Xing, E. P. (2014). Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. AUAI Press.
- [96] Nickisch, H. and Rasmussen, C. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*.
- [97] O’Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B (Methodological)*.
- [98] Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Hochschultext Universitext. Springer.
- [99] Opper, M. (2019). Variational inference for stochastic differential equations. *Annalen der Physik*.
- [100] Paisley, J., Blei, D. M. and Jordan, M. I. (2012). Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. ACM.
- [101] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference.
- [102] Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*.
- [103] Papaspiliopoulos, O., Roberts, G. O., and Stramer, O. (2013). Data augmentation for diffusions. *Journal of Computational and Graphical Statistics*.
- [104] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning, ICML’13*. JMLR.org.

- [105] Picchini, U. (2014). Inference for SDE models via approximate Bayesian computation. *Journal of Computational and Graphical Statistics*.
- [106] Pontryagin, L. S. (2018). *Mathematical theory of optimal processes*. Routledge.
- [107] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*.
- [108] Quiroz, M., Kohn, R., Villani, M., and Tran, M.-N. (2019). Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*.
- [109] Ranganath, R., Gerrish, S., and Blei, D. (2014). Black box variational inference. In *Artificial Intelligence and Statistics*. PMLR.
- [110] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*. MIT Press Cambridge, MA.
- [111] Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*.
- [112] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR.
- [113] Riihimäki, J. and Vehtari, A. (2010). Gaussian processes with monotonicity information. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.
- [114] Robert, C. P. (2004). *Monte Carlo methods*. Wiley Online Library.
- [115] Robert, C. P. and Casella, G. (2005). *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- [116] Roberts, G. O. and Stramer, O. (2001). On inference for partially observed nonlinear diffusion models using the Metropolis-Hastings algorithm. *Biometrika*.
- [117] Rogers, L. C. G. and Williams, D. (2013). *Diffusions, Markov processes and martingales*. Cambridge University Press.
- [118] Rosenthal, J. (2006). *A First Look at Rigorous Probability Theory*. World Scientific.
- [119] Ross, S. M. (1997). *Introduction to Probability Models*. Academic Press.
- [120] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- [121] Rudin, W. (2006). *Real and Complex Analysis*. McGraw-Hill, Inc., USA.
- [122] Rutter, A., Sanguinetti, G., and Opper, M. (2010). Approximate inference for stochastic reaction processes. In Lawrence, N. D., Girolami, M., Rattray, M., and Sanguinetti, G., editors, *Learning and Inference in Computational Systems Biology*. MIT Press.
- [123] Ryder, T., Golightly, A., Matthews, I., and Prangle, D. (2019). Scalable approximate inference for state space models with normalising flows.

- [124] Ryder, T., Golightly, A., McGough, A. S., and Prangle, D. (2018). Black-box variational inference for stochastic differential equations. In *International Conference on Machine Learning*.
- [125] Särkkä, S. (2007). On unscented Kalman filtering for state estimation of continuous-time nonlinear systems. *IEEE Transactions on Automatic Control*.
- [126] Särkkä, S. (2013). *Bayesian filtering and smoothing*. Cambridge University Press.
- [127] Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Cambridge University Press.
- [128] Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2016). Bayes and big data: the consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*.
- [129] Sherlock, C. (2006). *Methodology for inference on the Markov modulated Poisson process and theory for optimal scaling of the random walk Metropolis*. PhD thesis.
- [130] Sherlock, C., Thiery, A. H., Roberts, G. O., Rosenthal, J. S., et al. (2015). On the efficiency of pseudo-marginal random walk metropolis algorithms. *The Annals of Statistics*.
- [131] Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18*. MIT Press.
- [132] Solin, A. and Särkkä, S. (2014). Explicit Link Between Periodic Covariance Functions and State Space Models. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR.
- [133] Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research.
- [134] Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. JMLR Workshop and Conference Proceedings.
- [135] Titsias, M. K. and Lawrence, Neil D Rattray, M. (2009). Efficient sampling for Gaussian process inference using control variables. In *Advances in Neural Information Processing Systems*.
- [136] van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel WaveNet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*.
- [137] van der Meulen, F., Schauer, M., and van Zanten, H. (2017). Bayesian estimation of discretely observed multi-dimensional diffusion processes using guided proposals. *Electronic Journal of Statistics*.
- [138] van Kampen, N. G. (2007). *Stochastic processes in physics and chemistry*. Elsevier.

- [139] Vieira, R. and Wilkinson, D. J. (2016). Online state and parameter estimation in dynamic generalised linear models. *arXiv preprint arXiv:1608.08666*.
- [140] V.P.L.S., R. B. F. H. M. . R. A. (1828). Xxvii. a brief account of microscopical observations made in the months of june, july and august 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *The Philosophical Magazine*.
- [141] Vrettas, M., Oppen, M., and Cornford, D. (2015). Variational mean-field algorithm for efficient inference in large systems of stochastic differential equations. *Physical Review E*.
- [142] Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*.
- [143] Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*.
- [144] Ward, W. O. and Álvarez, M. A. (2019). Variational bridge constructs for approximate Gaussian process regression. *arXiv preprint arXiv:1901.01727*.
- [145] Ward, W. O. C., Ryder, T., Prangle, D., and Álvarez, M. A. (2019). Black-box inference for non-linear latent force models.
- [146] West, M. and Harrison, J. (2006). *Bayesian forecasting and dynamic models*. Springer.
- [147] Whitaker, G. A., Golightly, A., Boys, R. J., and Sherlock, C. (2017a). Bayesian inference for diffusion-driven mixed-effects models. *Bayesian Analysis*.
- [148] Whitaker, G. A., Golightly, A., Boys, R. J., and Sherlock, C. (2017b). Improved bridge constructs for stochastic differential equations. *Statistics and Computing*.
- [149] Wilkinson, D. J. (2018). *Stochastic modelling for systems biology*. CRC press.
- [150] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.
- [151] Winn, J. and Bishop, C. (2005). Variational Message Passing. *Journal of Machine Learning Research*.
- [152] Worden, K., Barthorpe, R., Cross, E., Dervilis, N., Holmes, G., Manson, G., and Rogers, T. (2018). On evolutionary system identification with applications to nonlinear benchmarks. *Mechanical Systems and Signal Processing*.
- [153] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*.

Appendix A

Stochastic processes

A.1 The Itô integral

Begin by examining the random function (i.e. a deterministic function applied to a random object) $f(x_s, s)$, $s \in [0, t]$, satisfying

$$\int_0^t \mathbb{E} [f(x_s, s)^2] ds < \infty, \quad (\text{A.1})$$

where $\{x_t | t \geq 0\}$ is a stochastic process. We would now like to evaluate the integral

$$\int_0^t f(x_s, s) dw_s. \quad (\text{A.2})$$

The Itô integral proceeds by partitioning $[0, t]$ with m equidistant time points, such that

$$0 = \tau_0 < \tau_1 < \cdots < \tau_{m-1} < \tau_m = t, \quad (\text{A.3})$$

where $\Delta\tau = \tau_{i+1} - \tau_i$. Defining $\Delta w_{\tau_i} = w_{\tau_{i+1}} - w_{\tau_i}$, the Itô stochastic integral is given by

$$\int_0^t f(x_s, s) dw_s = \text{l.i.m.}_{m \rightarrow \infty} \sum_{i=0}^{m-1} f(x_{\tau_i}, \tau_i) \Delta w_{\tau_i}, \quad (\text{A.4})$$

where l.i.m. denotes the *mean-square limit*. In some simple cases, such as the integral in equation (3.9), it is possible to verify this result directly. For example, if we take $f(x_t, t) = 1$,

(A.2) becomes

$$\begin{aligned}
\int_0^t dw_s &= \text{l.i.m.}_{m \rightarrow \infty} \sum_{i=0}^{m-1} \Delta w_{\tau_i}, \\
&= \text{l.i.m.}_{m \rightarrow \infty} \{ (w_{\tau_1} - w_{\tau_0}) + (w_{\tau_2} - w_{\tau_1}) + \cdots + (w_{\tau_m} - w_{\tau_{m-1}}) \}, \\
&= \text{l.i.m.}_{m \rightarrow \infty} (w_{\tau_m} - w_{\tau_0}), \\
&= \text{l.i.m.}_{m \rightarrow \infty} (w_t - w_0), \\
&= w_t - w_0, \\
&= w_t.
\end{aligned}$$

If we further define the function $h(x_t, t)$ subject to

$$\int_0^t \mathbb{E} [h(x_s, s)^2] ds < \infty, \quad (\text{A.5})$$

it is possible to specify some of properties for the Itô integral. Namely:

1. The property of *linearity* (see e.g. Arnold [12] for a proof):

$$\int_0^t \{A f(x_s, s) + B h(x_s, s)\} dw_s = A \int_0^t f(x_s, s) dw_s + B \int_0^t h(x_s, s) dw_s, \quad (\text{A.6})$$

for constant A and B .

2. The property of *Itô isometry* (see e.g. Allen [4] for a proof):

$$\mathbb{E} \left[\left\{ \int_0^t f(x_s, s) dw_s \right\}^2 \right] = \mathbb{E} \left[\int_0^t f(x_s, s)^2 ds \right]. \quad (\text{A.7})$$

A.2 Itô's Formula

Recall the diffusion process $\{x_t, t \geq 0\}$ satisfying

$$x_t = x_0 + \int_0^t \alpha(x_s, s, \theta) ds + \int_0^t \sqrt{\beta(x_s, s, \theta)} dw_s, \quad (\text{A.8})$$

for which we now permit an explicit time-dependency. The process defined in (A.8) is therefore *time-inhomogeneous*. Now consider the real-valued function $g(\cdot)$, twice differentiable in x and once differentiable in t , with parameters ϑ . This function defines the process $\{z_t, t \geq 0\}$ according to

$$z_t = g(x_t, t, \vartheta), \quad (\text{A.9})$$

In practice, it is often the case that $\vartheta \subset \theta$. Itô's formula (see e.g. Gikhman and Skorokhod [52] for a derivation) then defines the SDE satisfying

$$\begin{aligned} dg(x_t, t) = & \left(\frac{\partial}{\partial t} g(x_t, t, \vartheta) + \alpha(x_t, t, \theta) \frac{\partial}{\partial x} g(x_t, t, \vartheta) + \frac{\beta(x_t, t, \theta)^2}{2} \frac{\partial^2}{\partial x^2} g(x_t, t, \vartheta) \right) dt + \\ & + \beta(x_t, t, \theta) \frac{\partial}{\partial x} g(x_t, t, \vartheta) dw_t. \end{aligned} \quad (\text{A.10})$$

In this way, Itô's formula is able to permit non-linear transformations of the underlying diffusion process x_t through the function $g(\cdot)$.

A.2.1 Example: Ornstein-Uhlenbeck process

We now present a simple instance for which the Itô formula can be used to derive a closed-form solution for an SDE. Here, for notational simplicity, we suppress time as an input, but note the time-homogeneous nature of the eventual solution. For this purpose, we examine the *mean-reverting* Ornstein-Uhlenbeck diffusion process satisfying

$$dx_t = \theta_1 (\theta_2 - x_t) dt + \sqrt{\theta_3} dw_t, \quad (\text{A.11})$$

where $x_t \in \mathbb{R}$. Here the parameters θ_1 , θ_2 and θ_3 represent the mean-reversion rate, process mean and the infinitesimal standard deviation, respectively.

We begin by selecting the our function $g(\cdot)$

$$g(x_t, t, \vartheta) = x_t e^{\theta_1 t}. \quad (\text{A.12})$$

Substituting (A.12) and (A.11) into (A.10) yields

$$\begin{aligned} dx_t e^{\theta_1 t} &= \left[\theta_1 x_t e^{\theta_1 t} + \theta_1 (\theta_2 - x_t) e^{\theta_1 t} \right] dt + \theta_3 e^{\theta_1 t} dw_t, \\ &= \theta_1 \theta_2 e^{\theta_1 t} dt + \theta_3 e^{\theta_1 t} dw_t. \end{aligned} \quad (\text{A.13})$$

Integrating out both sides of (A.13) then gives

$$\begin{aligned}
\int_0^t dx_s e^{\theta_1 s} &= \int_0^t \theta_1 \theta_2 e^{\theta_1 s} ds + \int_0^t \theta_3 e^{\theta_1 s} dw_s, \\
\Rightarrow [x_s e^{\theta_1 s}]_0^t &= [\theta_2 e^{\theta_1 s}]_0^t + \theta_3 \int_0^t e^{\theta_1 s} dw_s, \\
\Rightarrow x_t e^{\theta_1 t} - x_0 &= \theta_2 e^{\theta_1 t} - \theta_2 + \theta_3 \int_0^t e^{\theta_1 s} dw_s, \\
\Rightarrow x_t &= x_0 e^{-\theta_1 t} + \theta_2 (1 - e^{-\theta_1 t}) + \theta_3 e^{-\theta_1 t} \int_0^t e^{\theta_1 s} dw_s. \tag{A.14}
\end{aligned}$$

From linearity and Itô isometry, it follows that the integral in (A.14) defines

$$\int_0^t f(s) dw_s \sim \mathcal{N}\left(0, \int_0^t f(s)^2 ds\right). \tag{A.15}$$

That is, one can evaluate the integral

$$\begin{aligned}
\theta_3^2 e^{-2\theta_1 t} \int_0^t e^{2\theta_1 s} ds &= \theta_3^2 e^{-2\theta_1 t} \left[\frac{e^{2\theta_1 s}}{2\theta_1} \right]_0^t, \\
&= \theta_3^2 e^{-2\theta_1 t} \left(\frac{e^{2\theta_1 t}}{2\theta_1} - \frac{1}{2\theta_1} \right), \\
&= \frac{\theta_3^2}{2\theta_1} (1 - e^{-2\theta_2 t}), \tag{A.16}
\end{aligned}$$

where the transition density then follows from (A.15) and (A.14) according to

$$p(x_t | x_0, \theta) = \mathcal{N}\left(x_0 e^{-\theta_1 t} + \theta_2 (1 - e^{-\theta_1 t}), \frac{\theta_3^2}{2\theta_1} (1 - e^{-2\theta_2 t})\right). \tag{A.17}$$

Appendix B

Variational inference for discretised SDEs

B.1 Derivation of a Brownian bridge

Here we present a derivation of the transition density associated with a *generalised* Brownian bridge process. That is, an end-point conditioned on generalised Brownian motion. Generalised Brownian motion is the diffusion process $\{x_t, t \geq 0\}$ satisfying the SDE

$$dx_t = \theta_1 dt + \theta_2 dw_t, \tag{B.1}$$

for constant θ_1 and θ_2 , where $\theta = (\theta_1, \theta_2)'$. We note *standard* Brownian motion, as is presented in Section 4.1.1, can be recovered by setting $\theta_1 = 0$ and $\theta_2 = 1$.

Given a current state x_t , and some arbitrary initial state x_{t_s} (where we do not guarantee $t_s = 0$), we begin by obtaining the transition density $p(x_t|x_{t_s}, \theta)$ via direct integration. As previously, we suppress an explicit dependency on time for notational convenience. That is

$$\begin{aligned} \int_{t_s}^t dx_s &= \int_{t_s}^t \theta_1 ds + \int_{t_s}^t \theta_2 dw_s, \\ \implies [x_s]_{t_s}^t &= \theta_1 [s]_{t_s}^t + \theta_2 \int_{t_s}^t dw_s, \\ \implies x_t - x_{t_s} &= \theta_1 (t_s - t) + \theta_2 \int_{t_s}^t dw_s, \\ \implies x_t &= x_{t_s} + \theta_1 (t_s - t) + \theta_2 \int_{t_s}^t dw_s, \end{aligned} \tag{B.2}$$

where evaluating the remaining integral in the Itô sense (as demonstrated in Section A.1) yields

$$\implies x_t = x_{t_s} + \theta_1 (t_j - t) + \theta_2 (w_t - w_{t_s}). \quad (\text{B.3})$$

It then trivially follows from the definition of a Brownian motion (Section 3.3) that the transition density is defined by

$$p(x_t | x_{t_s}, \theta) = \mathcal{N}\left(x_{t_s} + \theta_1 (t - t_s), \theta_2^2 (t - t_s)\right). \quad (\text{B.4})$$

Now recall from Section 3.3.2 that a diffusion process with current state x_t , conditioned on an exactly observed future state x_{t_j} , satisfies an SDE with drift and diffusion

$$\hat{\alpha}(x_t, \theta) = \alpha(x_t, \theta) + \beta(x_t, \theta) \nabla_{x_t} \log p(x_{t_j} | x_t, \theta), \quad (\text{B.5})$$

$$\hat{\beta}(x_t, \theta) = \beta(x_t, \theta), \quad (\text{B.6})$$

where $t_j > t$. From (B.4), it follows that the transition density in (B.5) is given by

$$p(x_{t_j} | x_t, \theta) \sim \mathcal{N}\left(x_t + \theta_1 (t_j - t), \theta_2^2 (t_j - t)\right). \quad (\text{B.7})$$

As the density in (B.5) is Gaussian, it is possible to recover a closed form solution for its gradient term. That is, we note

$$\begin{aligned} \nabla_{x_t} \log p(x_{t_j} | x_t, \theta) &= \nabla_{x_t} \left[\log \left\{ \frac{1}{\theta_2 \sqrt{2\pi (t_j - t)}} e^{-\frac{(x_{t_j} - x_t - \theta_1 (t_j - t))^2}{2\theta_2^2 (t_j - t)}} \right\} \right], \\ &= \frac{x_{t_j} - x_t - \theta_1 (t_j - t)}{\theta_2^2 (t_j - t)}. \end{aligned} \quad (\text{B.8})$$

Substituting (B.8) into (B.5) then gives

$$\begin{aligned} \hat{\alpha}(x_t, \theta) &= \theta_1 + \theta_2^2 \left[\frac{x_{t_j} - x_t - \theta_1 (t_j - t)}{\theta_2^2 (t_j - t)} \right], \\ &= \frac{x_{t_j} - x_t}{t_j - t} \end{aligned} \quad (\text{B.9})$$

where the conditioned process therefore satisfies the SDE given by

$$dx_t = \left(\frac{x_{t_j} - x_t}{t_j - t} \right) dt + \theta_2 dw_t. \quad (\text{B.10})$$

It is further the case that (B.10) permits an exact solution via application of Itô's formula. We begin by selecting an integrating factor

$$g(x_t, t, \vartheta) = \frac{t_j x_t}{t_j - t}. \quad (\text{B.11})$$

Substituting (B.11) into Itô's formula (Section A.2, equation (A.10)) then gives

$$\begin{aligned} d\left(\frac{t_j x_t}{t_j - t}\right) &= \left[\frac{t_j x_t}{(t_j - t)^2} + \frac{t_j}{t_j - t} \left(\frac{x_{t_j} - x_t}{t_j - t} \right) \right] dt + \frac{t_j}{t_j - t} \theta_2 dw_t, \\ &= \frac{t_j x_{t_j}}{(t_j - t)^2} dt + \frac{t_j}{t_j - t} \theta_2 dw_t. \end{aligned} \quad (\text{B.12})$$

Integrating (B.12) over $[0, t]$ yields

$$\begin{aligned} \int_0^t d\left(\frac{t_j x_s}{t_j - s}\right) &= \int_0^t \frac{t_j x_t}{(t_j - s)^2} ds + \int_0^t \frac{t_j}{t_j - s} \theta_2 dw_s, \\ \Rightarrow \left[\frac{t_j x_s}{t_j - s} \right]_0^t &= \left[\frac{t_j x_{t_j}}{t_j - s} \right]_0^t + \int_0^t \frac{t_j}{t_j - s} \theta_2 dw_s, \\ \Rightarrow \frac{t_j x_t}{t_j - t} - x_0 &= \frac{t_j x_{t_j}}{t_j - t} - x_{t_j} + \int_0^t \frac{t_j}{t_j - s} \theta_2 dw_s, \\ \Rightarrow x_t &= x_0 + \frac{(x_{t_j} - x_0)t}{t_j} + \frac{t_j - t}{t_j} \int_0^t \frac{t_j}{t_j - s} \theta_2 dw_s. \end{aligned} \quad (\text{B.13})$$

Similarly as in Section A.2.1, equations (A.15) and (A.16), applying the Itô isometry to the stochastic integral in (B.13) results in the density

$$\begin{aligned} \int_0^t \frac{t_j}{t_j - s} \theta_2 dw_s &\sim \mathcal{N}\left(0, \int_0^t \left(\frac{t_j}{t_j - s}\right)^2 \theta_2^2 ds\right), \\ &\sim \mathcal{N}\left(0, \frac{t_j t}{t_j - t} \theta_2^2\right). \end{aligned} \quad (\text{B.14})$$

It follows from (B.13) and (B.14) that the transition density is given by

$$\begin{aligned} p(x_t | x_{t_j}, x_0, \theta) &\sim \mathcal{N}\left(x_0 + \frac{(x_{t_j} - x_0)t}{t_j}, \left(\frac{t_j - t}{t_j}\right)^2 \left(\frac{t_j t}{t_j - t} \theta_2^2\right) ds\right), \\ &\sim \mathcal{N}\left(x_0 + \frac{(x_{t_j} - x_0)t}{t_j}, \frac{(t_j - t)}{t_j} \theta_2^2\right). \end{aligned} \quad (\text{B.15})$$

That is, the transition density for Brownian motion conditioned on *both* the initial state x_0 and a future state x_{t_j} , i.e. a Brownian bridge.

Appendix C

Continuous-time methods

C.1 Ornstein-Uhlenbeck SDE with end-point conditioning

Begin by recalling the Ornstein-Uhlenbeck process satisfying the following SDE

$$dx_t = \theta_1 (\theta_2 - x_t) dt + \sqrt{\theta_3} dw_t, \quad (\text{C.1})$$

which (from Appendix A.2.1) permits the closed-form solution of the form

$$p(x_t|x_0, \theta) = \mathcal{N}\left(x_0 e^{-\theta_1 t} + \theta_2 (1 - e^{-\theta_1 t}), \frac{\theta_3^2}{2\theta_1} (1 - e^{-2\theta_1 t})\right). \quad (\text{C.2})$$

Now suppose we have access to a single observation of (C.1) at time T according to

$$y_T \sim \mathcal{N}(x_T, \sigma^2), \quad (\text{C.3})$$

and would like to derive a solution for the conditional probability density function

$$p(x_t|x_s, y_T, \theta), \quad (\text{C.4})$$

such that $0 \leq s < t < T$. To achieve this, we begin by observing that from (C.2) it trivially follows that

$$p(x_t|x_s, \theta) = \mathcal{N}\left(x_s e^{-\theta_1(t-s)} + \theta_2 (1 - e^{-\theta_1(t-s)}), \frac{\theta_3^2}{2\theta_1} (1 - e^{-2\theta_1(t-s)})\right). \quad (\text{C.5})$$

Hence, using (C.3) and (C.5), we obtain

$$p(y_T|x_s, \theta) = \mathcal{N}\left(x_s e^{-\theta_1(T-s)} + \theta_2(1 - e^{-\theta_1(T-s)}), \frac{\theta_3^2}{2\theta_1}(1 - e^{-2\theta_2(T-s)}) + \sigma^2\right). \quad (\text{C.6})$$

Note that the covariance between y_T and x_t (given x_s) can be found as

$$\text{cov}(y_T|x_t, \theta) = \frac{\theta_3^2}{2\theta_1}(e^{-\theta_1(T-t)} - e^{-\theta_1(T+t-2s)}). \quad (\text{C.7})$$

Now, the joint probability distribution of x_t and y_T (conditional on x_s) is given by

$$\begin{pmatrix} x_t \\ y_T \end{pmatrix} \Big| x_s, \theta \sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma}), \quad (\text{C.8})$$

where

$$\tilde{\mu} = \begin{pmatrix} x_s e^{-\theta_1(t-s)} + \theta_2(1 - e^{-\theta_1(t-s)}) \\ x_s e^{-\theta_1(T-s)} + \theta_2(1 - e^{-\theta_1(T-s)}) \end{pmatrix}, \quad (\text{C.9})$$

$$\tilde{\Sigma} = \begin{pmatrix} \frac{\theta_3^2}{2\theta_1}(1 - e^{-2\theta_2(t-s)}) & \frac{\theta_3^2}{2\theta_1}(e^{-\theta_1(T-t)} - e^{-\theta_1(T+t-2s)}) \\ \frac{\theta_3^2}{2\theta_1}(e^{-\theta_1(T-t)} - e^{-\theta_1(T+t-2s)}) & \frac{\theta_3^2}{2\theta_1}(1 - e^{-2\theta_2(T-s)}) + \sigma^2 \end{pmatrix}. \quad (\text{C.10})$$

From standard multivariate Gaussian conditioning results it then follows that the desired conditional density (C.4) is given by

$$p(x_t|x_s, y_T, \theta) = \mathcal{N}(\tilde{\mu}_1 + \tilde{\Sigma}_{12}\tilde{\Sigma}_{22}^{-1}(y_T - \mu_2), \tilde{\Sigma}_{11} - \tilde{\Sigma}_{12}\tilde{\Sigma}_{22}^{-1}\tilde{\Sigma}_{21}), \quad (\text{C.11})$$

where $\tilde{\Sigma}_{ij}$ is the $(i, j)^{\text{th}}$ element of the matrix, and μ_i is the i^{th} element of the vector. Hence, given any finite partition of $[0, T]$, (C.11) can be sampled recursively at the resulting intermediate time-points to produce skeleton paths of the condition process.

Appendix D

Scalable State-Space Inference with Normalizing Flows

D.1 MCMC algorithm for AR(1) via forward-filter recursion

Here we present an MCMC method for the AR(1) model. Assuming P observations, the marginal parameter posterior is given by

$$p(\theta|y_{t_{0:P}}) \propto p(\theta)p(y_{t_{0:P}}|\theta), \quad (\text{D.1})$$

where $p(y_{t_{0:P}}|\theta)$ is the marginal likelihood obtained from integrating out the latent variables from $p(\theta, x_{t_{0:P}}|y_{t_{0:P}})$. We sample the marginal parameter posterior $p(\theta|y_{t_{0:P}})$ using a random walk Metropolis-Hastings scheme.

This appendix describes the key step of evaluating the marginal likelihood given θ , which is achieved using a *forward filter*. See West and Harrison [146] for a general introduction to forward-filtering algorithms for linear state space models. We adapt this as follows.

As can be seen from (6.16), the AR(1) process is linear and Gaussian. Hence, for a Gaussian observation model of variance σ^2 , the marginal likelihood is tractable and can be efficiently computed via a forward-filter recursion. This utilises the factorisation

$$p(y_{t_{0:P}}|\theta) = p(y_{t_0}|\theta) \prod_{i=1}^P p(y_{t_i}|y_{t_{0:i-1}}, \theta), \quad (\text{D.2})$$

by recursively evaluating each term.

Suppose that $x_{t_i}|y_{t_{0:i}} \sim \mathcal{N}(a_i, c_i)$. Since $x_{t_0} = 10$ we can take $a_0 = 10, c_0 = 0$. It follows that

$$x_{t_{i+1}}|y_{t_{0:i}} \sim \mathcal{N}\left(\theta_1 + \theta_2 a_i, \theta_2^2 c_i + \theta_3^2\right), \quad (\text{D.3})$$

which, from the observation model, gives us the one-step-ahead forecast

$$y_{t_{i+1}}|y_{t_{0:i}} \sim \mathcal{N}\left(\theta_1 + \theta_2 a_i, \theta_2^2 c_i + \theta_3^2 + \sigma^2\right). \quad (\text{D.4})$$

Hence the marginal likelihood can be recursively updated using

$$p(y_{t_{0:i+1}}|\theta) = p(y_{t_{0:i}}|\theta)p(y_{t_{i+1}}|y_{t_{0:i}}, \theta), \quad (\text{D.5})$$

where $p(y_{t_{i+1}}|y_{t_{0:i}}, \theta)$ is the density of (D.4).

The next filtering distribution is obtained as $x_{t_{i+1}}|y_{t_{0:i+1}} \sim \mathcal{N}(a_{i+1}, c_{i+1})$ where

$$a_{i+1} = \theta_1 + \theta_2 a_i + \frac{(\theta_2^2 c_i + \theta_3^2)(y_{i+1} - \theta_1 - \theta_2 a_i)}{(\theta_2^2 c_i + \theta_3^2 + \sigma^2)}, \quad (\text{D.6})$$

$$c_{i+1} = \theta_2^2 c_i + \theta_3^2 - \frac{(\theta_2^2 c_i + \theta_3^2)^2}{(\theta_2^2 c_i + \theta_3^2 + \sigma^2)}. \quad (\text{D.7})$$

Evaluation of (D.3)-(D.7) for $i = 0, 1, \dots, P-1$ gives the marginal likelihood $p(y_{t_{0:P}}|\theta)$.

D.2 Experimental Details

This appendix lists tuning choices for our examples.

In all of our examples we set both N (number of samples used in ELBO gradient estimate) and M (batch length) equal to 50, and use $\nu = 3$ affine layers in our flow for x .

Each affine layer has a CNN with 4 layers of one-dimensional convolutional networks. Each intermediate layer has 50 filters, uses ELU activation and batch normalisation (except the output layer). Before being injected to the first CNN layer, side information vectors (see Section 6.4.2) are processed through a feed-forward network to produce an encoded vector of length 50. We use a vanilla feed-forward network of 50 hidden units by 3 layers, with ELU activation.

We use the AdaMax optimiser with tuning parameters $\beta_1 = 0.95$ (non-default choice) and $\beta_2 = 0.999$ (default choice). See the tables below for learning rates used.

Each experiment uses a small number of pre-training SGD iterations for ϕ_θ optimising $E_{\theta \sim q}[p(\theta)]$, the expected prior density. We separately pre-train ϕ_x to optimise an objective

detailed in the tables below. As we discuss in Section 6.5, where possible we aim to initialise x to be close to the observations, or some other reasonable initial value.

Choices specific to each experiment are listed below.

AR(1)

Learning rate	10^{-3}
Pre-training for x	500 iterations minimising $E_{\theta, x \sim q}[x - \hat{y} _2]$, where $\hat{y}$ is the observed data.
ℓ	10

Lotka-Volterra: data setting (a)

Learning rate	10^{-3}
Pre-training for x	500 iterations minimising $E_{\theta, x \sim q}[x - \hat{y} _2]$, where $\hat{y}$ is linear interpolation of the data.
ℓ	20

Lotka-Volterra: data setting (b)

Learning rate	5×10^{-4}
Pre-training for x	500 iterations maximising $E_{x \sim q}[p(x \theta^*)]$ where $\theta^* = (0.5, 0.0025, 0.3)$. See Appendix D.3 for more details.
ℓ	20

FitzHugh-Nagumo

Learning rate	5×10^{-4}
Pre-training for x	500 iterations minimising $E_{\theta, x \sim q}[x _2]$. Here the model has some unobserved components, so we cannot initialise x close to the observations. Instead we simply encourage x to take small initial values.
ℓ	20

D.3 Lotka-Volterra details

Here we discuss some methodology specific to the Lotka-Volterra example in more detail.

D.3.1 Restricting x to positive values

For our Lotka-Volterra model, $x_i = (u_i, v_i)$ represents two population sizes. Negative values don't have a natural interpretation, and also cause numerical errors in the model i.e. the matrix β in (6.18), Section 6.6.2, may no longer be positive definite so that a Cholesky factor, required in (3.12), Section 3.3.1, is not available.

Therefore we wish to restrict the support of $q(x|\theta; \phi_x)$ to positive values. To do so we add a final element-wise softplus bijection to our local IAF. Let $\tilde{x}$ be the output before this final bijection. The log density (6.8) gains an extra term to become

$$\log q(x) = \sum_{i=1}^P \mathcal{N}(z_i^0; 0, I_d) - \sum_{k=1}^d \sum_{j=1}^{\nu} \sum_{i=1}^P \log \sigma_{ik}^j - \sum_{k=1}^d \sum_{i=1}^P \gamma(\tilde{x}_{ik}), \quad (\text{D.8})$$

where γ is the derivative of the softplus function (i.e. the logistic function). The ELBO calculations remain unchanged except for taking

$$\lambda_i = \mathcal{N}(z_i^0; 0, I_d) - \sum_{k=1}^d \sum_{j=1}^m \log \sigma_{ik}^j - \sum_{k=1}^d \gamma(\tilde{x}_{ik}). \quad (\text{D.9})$$

We implement our method as before with this modification to λ_i .

D.3.2 Multiple modes and pre-training

Observation setting (b) of our Lotka-Volterra example has multiple posterior modes. Without careful initialisation of $q(x|\theta)$, the variational approach typically finds a mode with high frequency oscillations in x . An example is displayed in Figure D.1. The corresponding estimated maximum a-posteriori parameter values are $\hat{\theta} = (4.428, 0.029, 2.957)$.

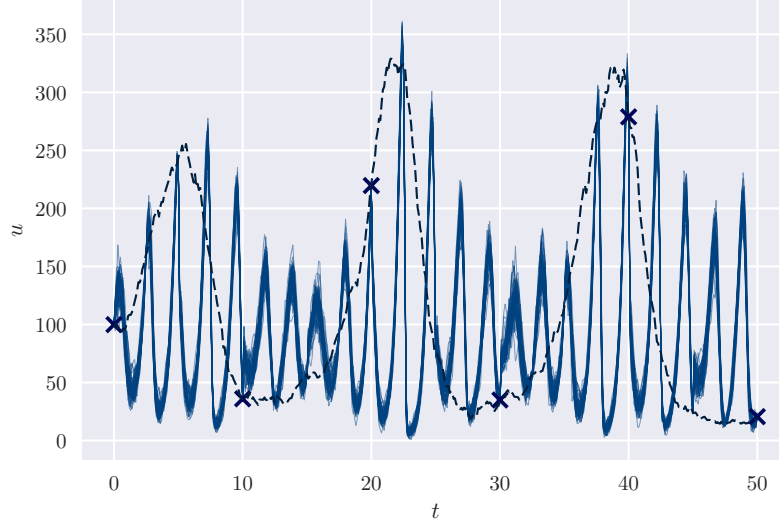


Figure D.1 Latent path (dashed line), available observations (crosses) and 50 samples of the variational x . Here, for ease of presentation, we present results for u only. The horizontal axis shows $t = 0.1i$.

Ideally we would aim to find the most likely modes and evaluate their posterior probabilities, but this is infeasible for our method. (It could be feasible to design a reversible jump MCMC [60] algorithm to do this, but we are unaware of such a method for this application.) Instead we attempt to constrain our analysis to find the mode we expect to be most plausible – that giving a single oscillation between each pair of data points. It is difficult to encode this belief in our prior distribution, so instead we use pretraining so that VI concentrates on this mode. This is comparable to the common MCMC tuning strategy of choosing a plausible initial value.

We use 500 pretraining iterations maximising the likelihood of $p(x|\theta^*)$, where $\theta^* = 0.1\hat{\theta}$. The basis for this choice is that periodic Lotka-Volterra dynamics roughly correspond to cycles in (u, v) space. Multiplying θ , the rate constants of the dynamics, by ξ should give similar dynamics but increase the frequency by a factor of ξ . Based on Figure D.1 we wish to reduce the frequency by a factor of 10, so we choose $\xi = 0.1$. Using this pre-training approach, we obtain the results shown in the main paper (Figure 6.7), corresponding to a more plausible mode.

D.4 Likelihood approximation of LFM

Begin by recalling the latent force model (LFM) defined by the differential equation

$$\alpha_0(x_t, u_t, t; \theta) x_t + \sum_{i=1}^{n-1} \alpha_i(x_t, u_t, t; \theta) \frac{d^i}{dt^i} x_t = u_t, \quad (\text{D.10})$$

where u_t is an m -times differentiable Gaussian process prior and $\alpha_{0:n}(\cdot; \theta)$ are deterministic functions, with parameters θ . Further recall the companion-form SDE (6.29) of (D.10)

$$\frac{d\chi_t}{dt} = F\chi_t + L\zeta\omega_t, \quad (\text{D.11})$$

$$\implies d\chi_t = F\chi_t dt + L\zeta dw_t, \quad (\text{D.12})$$

with

$$\chi_t = \begin{pmatrix} x_t \\ \vdots \\ \frac{d^{n-2}x_t}{dt^{n-2}} \\ \frac{d^{n-1}x_t}{dt^{n-1}} \\ u_t \\ \vdots \\ \frac{d^{m-2}u_t}{dt^{m-2}} \\ \frac{d^{m-1}u_t}{dt^{m-1}} \end{pmatrix}, F = \begin{pmatrix} 0 & 1 & \dots & 0 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & & 0 & 1 & \vdots & & \ddots & \vdots \\ -\check{\alpha}_0 & \dots & -\check{\alpha}_{n-2} & -\check{\alpha}_{n-1} & -1 & \dots & \dots & 0 \\ 0 & \dots & \dots & 0 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \vdots & \vdots & & 0 & 1 \\ 0 & \dots & \dots & 0 & -a_0 & \dots & -a_{m-2} & -a_{m-1} \end{pmatrix}, \quad (\text{D.13})$$

where $L = (0, \dots, 0, 0, \dots, 1)^T$. Here χ_t is the d -dimensional augmented system state, $\check{\alpha}_{0:n-1} = \alpha_{0:n-1}(x_t, u_t, t; \theta)$, and $a_{0:m-1}$ are the scalar constants derived from the GP prior for u_t . Again note F may in practice be a function of both time and the system state, which we choose to suppress for notational simplicity.

When the state χ_t is subject to *non-linear* dynamics, such that $\check{\alpha}_i$ in (D.13) are dependent on x , the likelihood is typically intractable. As discussed in chapter 3 (Section 3.4.2) it can be additionally challenging to approximate this likelihood when attempting to use the Euler-Maruyama scheme. As an alternative, we consider the application of the unscented Kalman filter [143] (UKF), which is broadly applicable to any non-linear state estimation problem (see e.g. Jazwinski [72] for a broader introduction to these problems types). The UKF uses a set of so-called *sigma points* to characterise moments of the marginal state density $\chi_t \sim p(\chi_t)$. Sigma points are individually propagated through the dynamics and can be combined by means of weighted sums to obtain approximations of the mean and covariance [76].

More specifically, and for our purposes, we approximate the likelihood of (D.13) using the continuous-time extension of the UKF as presented by Särkkä [125]. This defines dynamics for predicting the mean and variance, M_t and V_t , of marginal state χ_t according to

$$\chi_t \sim \mathcal{N}(M_t, V_t), \quad (\text{D.14})$$

using a set of *sigma points* $\Upsilon_t \in \mathbb{R}^{d \times \tilde{d}}$, where $\tilde{d} = 2d + 1$ (the construction of sigma points is described below). That is, for a given non-linear dynamical system (e.g. (D.12)), characterised with drift and diffusion function, $\alpha(\chi_t, \theta) = F\chi_t$ and $\beta(\chi_t, \theta) = \Sigma$ (the calculation of σ for our application is detailed below), along with sigma points Υ_t , the mean and variance of (D.14) can be defined according to the ordinary differential equations

$$\begin{aligned} dM_t &= F\Upsilon_t \kappa^{(M)} dt, \\ dV_t &= \left[\Upsilon_t K (F\Upsilon_t)^T + F\Upsilon_t K \Upsilon_t^T + \Sigma \right] dt, \end{aligned} \quad (\text{D.15})$$

where we take $\Sigma = \tilde{L}Q_c\tilde{L}^T$. Here Q_c represents the $m \times m$ steady-state covariance of the state-space GP prior of u_t , and $\tilde{L}$ is $d \times m$ masking term to map Q_c to $d \times d$. Further, the dynamics of Υ_t are as described in Särkkä [125].

Weights $\kappa^{(M)} \in \mathbb{R}^{\tilde{d}}$, $K \in \mathbb{R}^{\tilde{d}}$ are derived in matrix-form from the base density $\mathcal{N}(\chi_t, \Sigma)$ [76] and used in the construction of Υ_t [76]. The matrix of sigma points is defined according to

$$[\Upsilon_t]_i = \begin{cases} M_t & i = 0 \\ M_t + [\sqrt{(d+\eta)\Sigma}]_i & i = 1, \dots, d \\ M_t - [\sqrt{(d+\eta)\Sigma}]_{i-d} & i = d+1, \dots, 2d \end{cases}.$$

where $[\cdot]_i$ indicates the i^{th} column of a matrix, and $\eta = \alpha_{\Upsilon}^2(d + \gamma_{\Upsilon}) - \gamma_{\Upsilon}$ is the unscented scaling term, and α_{Υ} , β_{Υ} , γ_{Υ} are hyperparameters of the transform [76]. Relevant weight terms are then defined by

$$\begin{aligned} \kappa_0^{(M)} &= \eta(d + \eta)^{-1} \\ \kappa_0^{(c)} &= \eta(d + \eta + 1 - \alpha_{\Upsilon}^2 + \beta_{\Upsilon})^{-1} \end{aligned} \quad (\text{D.16})$$

$$\begin{aligned} \kappa_i^{(M)} &= \kappa_i^{(c)} = (2d + 2\eta)^{-1} \\ \kappa_{-1} &= I_{\tilde{d}} - \begin{bmatrix} \kappa^{(M)} & \dots & \kappa^{(M)} \end{bmatrix} \\ K &= \kappa_{-1} \text{diag}[\kappa^{(c)}] \kappa_{-1}^T, \end{aligned} \quad (\text{D.17})$$

where $\kappa_i^{(M)}$ and $\kappa_i^{(c)}$ are the i -th elements of the vectors $\kappa^{(M)}$ and $\kappa^{(c)}$, from index 0 to $2d$.

The moments (D.15) can then be solved using an ODE solver, such as a Runge-Kutta scheme. At observation times, the update-step of the discrete-time UKF can be used to update the estimate of χ_t , which can be used as an initial value for further prediction [125]. In practice, we will use these update steps to approximate the likelihood on a discrete grid of times.

D.4.1 Discrete-time predictions

In calculating $\mathcal{L}(\phi)$ (6.12), we must calculate the likelihood of our discretised joint system state

$$p(\chi|\theta) = \prod_{i=0}^{P-1} p(\chi_{t_{i+1}}|\chi_{t_i}, \theta). \quad (\text{D.18})$$

To achieve this, we assume the transition density is Gaussian, and build moments using a discretised solution of the ODE (D.15). For this purpose we construct discrete-time predictions of (D.15) using Euler's method, on a uniform grid with step size Δt [63]. For completeness, Euler's method as applied to the solution of ODEs is identical to the Euler-Maruyama scheme, but with the omission of stochastic terms. That is, given an ODE of the form

$$dx_t = f(x_t, \theta)dt, \quad (\text{D.19})$$

for arbitrary function $f(x_t, \theta)$, Euler's scheme defines a generative solution according to

$$x_{t_{i+1}} = x_{t_i} + f(x_{t_i}, \theta) \Delta t. \quad (\text{D.20})$$

As applied to (D.15), (D.20) estimates the one-step prediction mean and variance according to

$$M_{t_{i+1}} = \chi_{t_i} + F\Upsilon_{t_i}\kappa^{(M)}\Delta t, \quad (\text{D.21})$$

$$V_{t_{i+1}} = \Sigma + \left[\Upsilon_{t_i}K(F\Upsilon_{t_i})^T + F\Upsilon_{t_i}K\Upsilon_{t_i}^T + \Sigma \right] \Delta t, \quad (\text{D.22})$$

where we follow Julier and Uhlmann [76] and use $\alpha_{\Upsilon} = 1$, $\beta_{\Upsilon} = 0$, and $\gamma_{\Upsilon} = d$ in the calculation of Υ_t , $\kappa^{(M)}$ and K .

The one-step transition density is therefore approximated by

$$p(\chi_{t_{i+1}}|\chi_{t_i}, \theta) = \mathcal{N}(\chi_{t_{i+1}}; M_{t_{i+1}}, V_{t_{i+1}}), \quad (\text{D.23})$$

and (D.18) trivially becomes

$$p(\chi|\theta) = \prod_{i=0}^{P-1} \mathcal{N}(\chi_{t_{i+1}}; M_{t_{i+1}}, V_{t_{i+1}}). \quad (\text{D.24})$$