AUXILIARY VARIABLE MARKOV CHAIN MONTE CARLO METHODS

MATTHEW MCKENZIE GRAHAM

Doctor of Philosophy
University of Edinburgh
2017

Matthew Mckenzie Graham: *Auxiliary variable Markov chain Monte Carlo methods,* 2017.
Submitted for the degree of Doctor of Philosophy, University of Edinburgh.

SUPERVISOR:
Dr. Amos J. Storkey

ABSTRACT

Markov chain Monte Carlo (MCMC) methods are a widely applicable class of algorithms for estimating integrals in statistical inference problems. A common approach in MCMC methods is to introduce additional auxiliary variables into the Markov chain state and perform transitions in the joint space of target and auxiliary variables. In this thesis we consider novel methods for using auxiliary variables within MCMC methods to allow approximate inference in otherwise intractable models and to improve sampling performance in models exhibiting challenging properties such as multimodality.

We first consider the pseudo-marginal framework. This extends the Metropolis–Hastings algorithm to cases where we only have access to an unbiased estimator of the density of target distribution. The resulting chains can sometimes show 'sticking' behaviour where long series of proposed updates are rejected. Further the algorithms can be difficult to tune and it is not immediately clear how to generalise the approach to alternative transition operators. We show that if the auxiliary variables used in the density estimator are included in the chain state it is possible to use new transition operators such as those based on slice-sampling algorithms within a pseudo-marginal setting. This auxiliary pseudo-marginal approach leads to easier to tune methods and is often able to improve sampling efficiency over existing approaches.

As a second contribution we consider inference in probabilistic models defined via a generative process with the probability density of the outputs of this process only implicitly defined. The approximate Bayesian computation (ABC) framework allows inference in such models when conditioning on the values of observed model variables by making the approximation that generated observed variables are 'close' rather than exactly equal to observed data. Although making the inference problem more tractable, the approximation error introduced in ABC methods can be difficult to quantify and standard algorithms tend to perform poorly when conditioning on high dimensional observations. This oftens requires further approximation by reducing the observations to lower dimensional summary statistics.

We show how including all of the random variables used in generating model outputs as auxiliary variables in a Markov chain state can allow the use of more efficient and robust MCMC methods such as slice sampling and Hamiltonian Monte Carlo (HMC) within an ABC framework. In some cases this can allow inference when conditioning on the full set of observed values when standard ABC methods require reduction to lower dimensional summaries for tractability. Further we introduce a novel constrained HMC method for performing inference in a restricted class of differentiable generative models which allows conditioning the generated observed variables to be arbitrarily close to observed data while maintaining computational tractability.

As a final topic we consider the use of an auxiliary temperature variable in MCMC methods to improve exploration of multimodal target densities and allow estimation of normalising constants. Existing approaches such as simulated tempering and annealed importance sampling use temperature variables which take on only a discrete set of values. The performance of these methods can be sensitive to the number and spacing of the temperature values used, and the discrete nature of the temperature variable prevents the use of gradient-based methods such as HMC to update the temperature alongside the target variables. We introduce new MCMC methods which instead use a continuous temperature variable. This both removes the need to tune the choice of discrete temperature values and allows the temperature variable to be updated jointly with the target variables within a HMC method.

ACKNOWLEDGEMENTS

Put acknowledgments here.

Lorem ipsum at nusquam appellantur his, ut eos erant homero concluda turque. Albucius appellantur deterruisset id eam, vivendum partiendo dissentiet ei ius. Vis melius facilisis ea, sea id convenire referrentur, takimata adolescens ex duo. Ei harum argumentum per. Eam vidit exerci appetere ad, ut vel zzril intellegam interpretaris.

Errem omnium ea per, pro con populo ornatus cu, ex qui dicant nemore melius. No pri diam iriure euismod. Graecis eleifend appellantur quo id. Id corpora inimicus nam, facer nonummy ne pro, kasd repudiandae ei mei. Mea menandri mediocrem dissentiet cu, ex nominati imperdiet nec, sea odio duis vocent ei. Tempor everti appareat cu ius, ridens audiam an qui, aliquid admodum conceptam ne qui. Vis ea melius nostrum, mel alienum euripidis eu.

DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Edinburgh, August 2017

Matthew Mckenzie Graham

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

MCMC Markov chain Monte Carlo

MH Metropolis–Hastings

MI Metropolis independence

SS slice sampling

KL Kullback–Leibler

SDE stochastic differential equation

ABC approximate Bayesian computation

PM pseudo-marginal

APM auxiliary pseudo-marginal

HMC Hamiltonian Monte Carlo

SMC sequential Monte Carlo

CAVI coordinate ascent variational inference

SVI stochastic variational inference

BBVI black box variational inference

ADVI automatic differentiation variational inference

EP expectation propagation

AD automatic differentiation

VAE variational autoencoder

GAN generative-adversarial network

CPU central processing unit

GPU graphics processing unit

FLOPS floating point operations per second

RMSE root mean squared error

ELBO evidence lower bound

PRNG pseudo-random number generator

CDF cumulative distribution function

ESS effective sample size

PSRF potential scale reduction factor

iid independently and identically distributed

wrt with respect to

iff if and only if

1 PROBABILISTIC MODELLING

The actual science of logic is conversant at present only with things either certain, impossible, or entirely doubtful, none of which (fortunately) we have to reason on. Therefore the true logic for this world is the calculus of probabilities
—James Clerk Maxwell

Inference is the process of drawing conclusions from evidence. Much of our lives are spent making inferences about the world given our observations of it; in particular inference is a central aspect of the scientific process. Although deductive logic offers a framework for inferring conclusions from absolute statements of truth, it does not apply to the more typical real-world setting where the information we receive is subject to uncertainty.

To make inferences under conditions of uncertainty, we must instead turn to probability theory. Probabilities offer a consistent framework for quantifying the uncertainty in our beliefs and making inferences given these beliefs. The output of the inference process is itself probabilistic, reflecting that the conclusions we make given uncertain information will themselves be subject to uncertainty.

The topic of this thesis will be methods for performing approximate inference in large complex probabilistic models, with we in particular concentrating on the class of Markov chain Monte Carlo methods which we will introduce in the following chapter. In this chapter we will discuss the basic concepts of probabilistic modelling which will underpin the inference methods discussed in later chapters.

In particular we will review some of the basic concepts of the measure-theoretic description of probability theory as some of the later results in the thesis are most clearly described within this framework which may be less familiar to some readers. We will also introduce graphical models as a compact way of visualising structure in probabilistic models and computation graphs as analagous tool for visualising the structure of functions. Finally we will give a concrete definition of the inference tasks that the methods presented in the rest of this thesis are aimed at computing approximate solutions to and motivate why such approximate computational methods are needed.

1.1 PROBABILITY THEORY

A σ-algebra, $\mathcal{E}$, on a set S is set of subsets of S with $S \in \mathcal{E}$, $\emptyset \in \mathcal{E}$ and which is closed under complement and countable unions and intersections.

A *probability space* is defined as a triplet $(S, \mathcal{E}, \mathsf{P})$ where

- S is the *sample space*, the set of all possible outcomes,
- $\mathcal{E}$ is the *event space*, a σ-algebra on S, defining all possible events (measurable subsets of S),
- P is the *probability measure*, a finite measure satisfying $\mathsf{P}(S) = 1$, which specifies the probabilities of events in $\mathcal{E}$.

Kolmogorov's axioms:

1. *Non-negativity:* $\mathsf{P}(E) \geq 0 \; \forall E \in \mathcal{E}$,
2. *Normalisation:* $\mathsf{P}(S) = 1$,
3. *Countable additivity: for any countable set of disjoint events $\{E_i\}_i : E_i \in \mathcal{F} \; \forall i$, $E_i \cap E_j = \emptyset \; \forall i \neq j$, $\mathsf{P}(\cup_i E_i) = \sum_i \mathsf{P}(E_i)$.*

Given this definition of a probability space, Kolmogorov's axioms [115] can be used to derive a measure-theoretic formulation of probability theory. The probability of an event $E \in \mathcal{E}$ is defined as its measure $\mathsf{P}(E)$. Two events $A, B \in \mathcal{E}$ are *independent* if $\mathsf{P}(A \cap B) = \mathsf{P}(A)\mathsf{P}(B)$.

A measure-theoretic approach has the advantage of providing a unified treatment for describing probabilities on both finite and infinite sample spaces. Although alternative derivations of the laws of probability from different premises such as Cox's theorem [46, 47] have been proposed, modern extensions of this work result in a calculus of probabilities that is equivalent to Kolmogorov's [218], with the differences mainly being in the philosophical interpretations of probabilities.

1.1.1 Random variables

If $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ are two measurable spaces, a function $f : X \to Y$ is measurable if $f^{-1}(E) \in \mathcal{F} \; \forall E \in \mathcal{G}$.

When modelling real-world processes, rather than considering events as subsets of an abstract sample space, it is usually more helpful to consider *random variables* which represent quantities in the model of interest. A random variable $\mathsf{x} : S \to X$ is defined as a measurable function from the sample space to a measurable space $(X, \mathcal{F})$.

The Borel σ-algebra $\mathscr{B}(\mathbb{R})$ is the smallest σ-algebra on $\mathbb{R}$ which contains all open real intervals.

Often X is the reals, $\mathbb{R}$, and $\mathcal{F}$ is the Borel σ-algebra on the reals, $\mathscr{B}(\mathbb{R})$, in which case we will refer to a *real random variable*. It is also common to consider cases where X is a real vector space, $\mathbb{R}^D$, and $\mathcal{F} = \mathscr{B}(\mathbb{R}^D)$ - in this case we will term the resulting random variable a *real random vector* and use the notation $\mathbf{x} : S \to X$. A final common special case is when X is countable and $\mathcal{F}$ is the power set $\mathscr{P}(X)$ in which case we will refer to x as a *discrete random variable*. As we will generally be concerned with real-valued random variables (and vectors) in this thesis, when it is unambiguous to do so we will drop the 'real' qualifier and simply refer to *random variables* and *random vectors*.

Due to the definition of a random variable as a measurable function, we can define a pushforward measure on a random variable x

$$\mathsf{P}_\mathsf{x}(A) = \mathsf{P} \circ \mathsf{x}^{-1}(A) = \mathsf{P}(\{s \in S : \mathsf{x}(s) \in A\}) \quad \forall A \in \mathcal{F}. \tag{1.1}$$

If $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ are two measurable spaces, μ a measure on these spaces and $f : X \to Y$ a measurable function, the pushforward measure μ_f satisfies $\mu_f(A) = \mu \circ f^{-1}(A)$ $\forall A \in \mathcal{G}$.

The measure P_x specifies that the probability of the event that the random variable x takes a value in a measurable set $A \in \mathcal{F}$ is $\mathsf{P}_\mathsf{x}(A)$. We will often describe P_x as the *distribution* of x.

1.1.2 Joint and conditional probability

Typically we will jointly define multiple random variables on the same probability space. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space and $\mathsf{x} : S \to X$, $\mathsf{y} : S \to Y$ be two random variables with corresponding σ-algebras $\mathcal{F}$ and $\mathcal{G}$. Then the *joint probability* of x and y is defined as

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \mathsf{P}\left(\mathsf{x}^{-1}(A) \cap \mathsf{y}^{-1}(B)\right) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.2}$$

The joint probability is related to P_x and P_y by

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, Y) = \mathsf{P}_\mathsf{x}(A), \; \mathsf{P}_{\mathsf{x},\mathsf{y}}(X, B) = \mathsf{P}_\mathsf{y}(B) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.3}$$

In this context P_x and P_y are referred to as *marginal distributions* of the joint distribution. Two random variables x and y are said to be independent if and only if

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \mathsf{P}_\mathsf{x}(A)\,\mathsf{P}_\mathsf{y}(B) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.4}$$

A particulary key concept for inference is the definition of *conditional probability*. The conditional probability of an event $A \in \mathcal{E}$ occuring given another event $B \in \mathcal{E}$ has occured is denoted $\mathsf{P}(A \mid B)$ and we have the definition

$$\mathsf{P}(A \mid B) = \frac{\mathsf{P}(A \cap B)}{\mathsf{P}(B)} \quad \forall A \in \mathcal{E}, B \in \mathcal{E} : \mathsf{P}(B) \neq 0. \tag{1.5}$$

In Kolmogorov's probability theory, (1.5) is given as an additional definition distinct from the basic axioms. In alternatives such as the work of Cox [46, 47] and de Finetti [68], conditional probabilities are instead viewed as a primitive.

Correspondingly we denote the conditional probability of the event of the random variable x taking a value in $A \in \mathcal{F}$ given the event that the random variable y takes a value in $B \in \mathcal{G}$ as $\mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid B)$. Using (1.5) and (1.2), $\mathsf{P}_{\mathsf{x}|\mathsf{y}}$ and $\mathsf{P}_{\mathsf{y}|\mathsf{x}}$ can be shown to satisfy

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid B)\,\mathsf{P}_\mathsf{y}(B) = \mathsf{P}_{\mathsf{y}|\mathsf{x}}(B \mid A)\,\mathsf{P}_\mathsf{x}(A) \quad \forall A \in \mathcal{F}, B \in \mathcal{G}. \tag{1.6}$$

This decomposition of a joint probability into a product of a conditional and marginal is sometimes referred to as the product rule. An implication of (1.6) is what is often termed *Bayes' theorem*

$$\mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid B) = \frac{\mathsf{P}_{\mathsf{y}|\mathsf{x}}(B \mid A)\,\mathsf{P}_{\mathsf{x}}(A)}{\mathsf{P}_{\mathsf{y}}(B)} \quad \forall A \in \mathcal{F},\, B \in \mathcal{G} : \mathsf{P}_{\mathsf{y}}(B) \neq 0, \tag{1.7}$$

which will be of key importance in the later discussion of inference.

The definition in (1.2) of the joint probability of a pair of random variables can be extended to arbitarily large collections of random variables. Similarly conditional probabilities can be defined for collections of multiple jointly dependent random variables, with the product rule given in (1.6) generalising to a combinatorial number of possible factorisations of the joint probability. Graphical models offer a convenient way of representing the dependencies between large collections of random variables and any resulting factorisation structure in their joint probability, and are discussed in Appendix B.

1.1.3 Probability densities

A measure on X is σ-finite if X is a countable union of finite measure sets.

So far we have ignored how the probability measure P is defined and by consequence the probability (distribution) of a random variable. The Radon–Nikodym theorem guarantees that for a pair of $\sigma-$finite measures μ and ν on a measurable space $(X, \mathcal{F})$ where ν is absolutely continuous with respect to μ, then there is a unique (up to μ-null sets) measurable function $f : X \to [0, \infty)$ termed a *density* such that

$$\nu(A) = \int_A f \,\mathrm{d}\mu = \int_A f(x)\,\mu(\mathrm{d}x) \quad \forall A \in \mathcal{F}. \tag{1.8}$$

If μ and ν are measures on a measurable space $(X, \mathcal{F})$ then ν has absolute continuity wrt to μ if $\forall A \in \mathcal{F}$, $\mu(A) = 0 \Rightarrow \nu(A) = 0$.

The two Lebesgue integral notations above are equivalent and we will use them interchangeably. The density function f is also termed the *Radon-Nikodym derivative* of ν with respect to μ, denoted $\frac{\mathrm{d}\nu}{\mathrm{d}\mu}$. Density functions therefore represent a convenient way to define a probability distribution with respect to a reference measure we will term the *base measure*. The key requirement defining what is an appropriate base measure to use it that the probability measure of interest is absolutely continuous with respect to the base measure.

It can also be shown that if $f = \frac{\mathrm{d}\nu}{\mathrm{d}\mu}$ and g is a measurable function

$$\int_X g(x)\,\nu(\mathrm{d}x) = \int_X g(x)\,f(x)\,\mu(\mathrm{d}x), \tag{1.9}$$

which we will use later when discussing calculation of expectations.

For real random variables, an appropriate base measure is the *Lebesgue measure*, λ, on $\mathbb{R}$. The probability P_x of a real random variable x can then be defined via a *probability density* $\mathsf{p}_\mathsf{x} : \mathbb{R} \to [0, \infty)$ by

$$\mathsf{P}_\mathsf{x}(A) = \int_A \mathsf{p}_\mathsf{x}(x)\, \lambda(\mathrm{d}x) = \int_A \mathsf{p}_\mathsf{x}(x)\, \mathrm{d}x \qquad \forall A \in \mathscr{B}(\mathbb{R}). \tag{1.10}$$

Analagously for a random vector $\mathbf{x}$ with density $\mathsf{p}_\mathbf{x} : \mathbb{R}^D \to [0, \infty)$ with respect to the D-dimensional Lebesgue measure λ^D

$$\mathsf{P}_\mathbf{x}(A) = \int_A \mathsf{p}_\mathbf{x}(\boldsymbol{x})\, \lambda^D(\mathrm{d}\boldsymbol{x}) = \int_A \mathsf{p}_\mathbf{x}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \qquad \forall A \in \mathscr{B}(\mathbb{R}^D). \tag{1.11}$$

The notation in the second equalities in (1.10) and (1.11) uses a convention that will be used throughout this thesis that integrals without an explicit measure are with respect to the Lebesgue measure.

The counting measure $\#$ is defined as $\#(A) = |A|$ for all finite A and $\#(A) = +\infty$ otherwise.

For discrete random variables, an appropriate base measure is instead the *counting measure*, $\#$. The probability of a discrete random variable is then defined via a probability density $\mathsf{p}_\mathsf{x} : X \to [0, 1]$ by

$$\mathsf{P}_\mathsf{x}(A) = \int_A \mathsf{p}_\mathsf{x}(x)\, \#(\mathrm{d}x) = \sum_{x \in A} \mathsf{p}_\mathsf{x}(x) \qquad \forall A \in \mathscr{P}(X). \tag{1.12}$$

The co-domain of a probability density p_x for a discrete random variable is restricted to $[0, 1]$ due to the non-negativity and normalisation requirements for the probability measure P_x, with $\sum_{x \in X} \mathsf{p}_\mathsf{x}(x) = 1$. Commonly for the case of a discrete random variable, the density p_x is instead referred to as a *probability mass function*, with density reserved for real random variables. We will however use *probability density* in both cases in keeping with the earlier definition of a density with respect to a base measure, this avoiding difficulties when defininining joint probabilities on a mixture of real and discrete random variables.

The joint probability $\mathsf{P}_{\mathsf{x},\mathsf{y}}$ of a pair of random variables x and y with co-domains the measurable spaces $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ respectively, can be defined via a joint probability density $\mathsf{p}_{\mathsf{x},\mathsf{y}} : X \times Y \to [0, \infty)$ with respect to a product measure $\mu_{\mathsf{x},\mathsf{y}} = \mu_\mathsf{x} \times \mu_\mathsf{y}$ by

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \int_{A \times B} \mathsf{p}_{\mathsf{x},\mathsf{y}}(x, y)\, \mu_{\mathsf{x},\mathsf{y}}(\mathrm{d}x, \mathrm{d}y) \tag{1.13}$$

If $(X_1, \mathcal{F}_1, \mu_1)$ and $(X_2, \mathcal{F}_2, \mu_2)$ are two measure spaces, the product measure $\mu_1 \times \mu_2$ on a measurable space $(X_1 \times X_2, \mathcal{F}_1 \otimes \mathcal{F}_2)$ is defined as satisfying $(\mu_1 \times \mu_2)(A_1 \times A_2) = \mu_1(A_1)\mu_2(A_2)$ $\forall A_1 \in \mathcal{F}_1,\, A_2 \in \mathcal{F}_2$.

As a consequence of Fubini's theorem, the integral over $\mu_{x,y}$ can be expressed as iterated integrals over μ_x and μ_y

$$\begin{aligned} \mathsf{P}_{x,y}(A, B) &= \int_A \int_B \mathsf{p}_{x,y}(x, y)\, \mu_x(\mathrm{d}x)\, \mu_y(\mathrm{d}y) \\ &= \int_B \int_A \mathsf{p}_{x,y}(x, y)\, \mu_y(\mathrm{d}y)\, \mu_x(\mathrm{d}x) \quad \forall A \in \mathcal{F},\, B \in \mathcal{G}. \end{aligned} \tag{1.14}$$

The two measures μ_x and μ_y can differ for example $\mu_x = \lambda$ and $\mu_y = \#$ if x is a real random variable and y is a discrete random variable.

When dealing with random variables, we will often only specify the co-domain of the random variable(s) and a (joint) probability density, with the base measure being implicitly defined as the Lebesgue measure for real random variables (or vectors), counting measure for discrete random variables and an appropriate product measure for a mix of random variables. Similarly we will usually neglect to explicitly define the probability space $(S, \mathcal{E}, \mathsf{P})$ which the random variable(s) map from. In this case we will typically use the loose notation $x \in X$ to mean a random variable x with co-domain X.

This less explicit but more succinct probability notation in terms of random variables and densities is common in the machine learning and computational statitistics literature and will generally be preferred to improve readability. Tables A.1, A.2 and A.3 in Appendix A give definitions of the densities and shorthand notation for some common parametric probability distributions that we use in this thesis.

1.1.4 Transforms of random variables

A key concept we will repeatedly make use of in this thesis is defining random variables via transformations of other random variables. Let x be a random variable with co-domain the measurable space $(X, \mathcal{F})$. Further let $(Y, \mathcal{G})$ be a second measurable space and $\phi : X \to Y$ a measurable function between the two spaces. If we define $y = \phi \circ x$ then analagously to our original definition of P_x as the pushforward measure of P under the measurable function defining x, we can define P_y in terms of P_x as

$$\mathsf{P}_y(A) = \mathsf{P}_x \circ \phi^{-1}(A) = \mathsf{P}_x(\{x \in X : \phi(x) \in A\}) \quad \forall A \in \mathcal{G}, \tag{1.15}$$

i.e. the probability of the event $\mathsf{y} \in A$ is equal to the probability of x taking a value in the pre-image under ϕ of A. To calculate probabilities of transformed random variables therefore we will therefore need to be able to find the pre-images of values of the transformed variable.

If the distribution P_x is defined by a density p_x with respect to a measure μ_x, we can also in some cases find a density p_y on the transformed variable $\mathsf{y} = \phi(\mathsf{x})$ with respect to a (potentially different) measure μ_y

$$\mathrm{P}_\mathsf{y}(A) = \int_{\phi^{-1}(A)} \mathrm{p}_\mathsf{x}(x)\, \mu_\mathsf{x}(\mathrm{d}x) = \int_A \mathrm{p}_\mathsf{y}(y)\, \mu_\mathsf{y}(\mathrm{d}y) \quad \forall A \in \mathcal{G}. \tag{1.16}$$

For random variables with countable co-domains where the integral in (1.16) corresponds to a sum, a p_y satisfying (1.16) is simple to identify. If x is a discrete random variable with probability density p_x with respect to the counting measure, then $\mathsf{y} = \phi(\mathsf{x})$ will necessarily also be a discrete random variable. Applying (1.16) for $\mathrm{p}_\mathsf{x} = \frac{\mathrm{dP}_\mathsf{x}}{\mathrm{d}\#}$ we have that[1]

$$\begin{aligned}\int_{\phi^{-1}(A)} \mathrm{p}_\mathsf{x}(x)\, \#(\mathrm{d}x) &= \sum_{x \in \phi^{-1}(A)} \mathrm{p}_\mathsf{x}(x) = \sum_{y \in A} \sum_{x \in \phi^{-1}(y)} \mathrm{p}_\mathsf{x}(x) \\ &= \int_A \sum_{x \in \phi^{-1}(y)} \mathrm{p}_\mathsf{x}(x)\, \#(\mathrm{d}y) \quad \forall A \in \mathcal{G}. \end{aligned} \tag{1.17}$$

We can therefore define $\mathrm{p}_\mathsf{y} = \frac{\mathrm{dP}_\mathsf{y}}{\mathrm{d}\#}$ in terms of p_x as

$$\mathrm{p}_\mathsf{y}(y) = \sum_{x \in \phi^{-1}(y)} \mathrm{p}_\mathsf{x}(x) \quad \forall y \in Y. \tag{1.18}$$

In the special case that ϕ is bijective with an inverse ϕ^{-1} we have that

$$\mathrm{p}_\mathsf{y}(y) = \mathrm{p}_\mathsf{x} \circ \phi^{-1}(y) \quad \forall y \in Y. \tag{1.19}$$

For transformations of real random variables and vectors, the situation is more complicated as we need to account for any local contraction or expansion of space by the transformation. We will consider here the special case where the transformation is a *diffeomorphism*: a differentiable bijection which has an inverse which is also differentiable. In Chapter 4 we will consider how this can be generalised to non-bijective

1 We use $\phi^{-1}(y)$ as a shorthand here for $\phi^{-1}(\{y\})$ i.e. the pre-image of a singleton set. In cases where an inverse function exists we will also use the same notation, which of the three meanings is intended should be clear from the context.

differentiable functions, including the case where the dimensionalities of the domain and co-domain of the function differ.

Let $X \subseteq \mathbb{R}^N$ and $Y \subseteq \mathbb{R}^N$ and $\boldsymbol{\phi} : X \to Y$ be a diffeomorphism. Then the *Jacobian* of $\boldsymbol{\phi}$ is defined as

$$\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial \phi_1}{\partial x_1} & \cdots & \frac{\partial \phi_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial \phi_N}{\partial x_1} & \cdots & \frac{\partial \phi_N}{\partial x_N} \end{bmatrix}. \tag{1.20}$$

The Jacobian matrix describes the local transformation of an infintesimal volume element $\mathrm{d}\boldsymbol{x}$ in X under the map $\boldsymbol{\phi}$. In particular the corresponding volume element in Y under the map will be an infintesimal parallelotope spanned by the columns of the Jacobian $\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x})$. The Jacobian matrix determinant $\left|\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x})\right|$ which corresponds to the volume of this parallelotope therefore determines hows the volume elements scales unders the map - a value more than one corresponds to a local expansion and less than one to a contraction. Informally we therefore have that $\mathrm{d}\boldsymbol{y} = \left|\mathbf{J}_{\boldsymbol{\phi}}(\boldsymbol{x})\right| \mathrm{d}\boldsymbol{x}$ and applying the same arguments to the inverse map $\boldsymbol{\phi}^{-1}$, $\mathrm{d}\boldsymbol{x} = \left|\mathbf{J}_{\boldsymbol{\phi}^{-1}}(\boldsymbol{y})\right| \mathrm{d}\boldsymbol{y}$.

Let $\mathbf{x}$ be a random vector taking values in the measurable space $(X, \mathscr{B}(X))$ and define $\mathbf{y} = \boldsymbol{\phi} \circ \mathbf{x}$ as a random vector taking values in $(Y, \mathscr{B}(Y))$. If $\mathsf{P}_{\mathbf{x}}$ has a density $\mathsf{p}_{\mathbf{x}}$ with respect to the Lebesgue measure

$$\begin{aligned} \mathsf{P}_{\mathbf{y}}(A) = \mathsf{P}_{\mathbf{x}} \circ \boldsymbol{\phi}^{-1}(A) &= \int_{\boldsymbol{\phi}^{-1}(A)} \mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ &= \int_A \mathsf{p}_{\mathbf{x}} \circ \boldsymbol{\phi}^{-1}(\boldsymbol{y}) \left|\mathbf{J}_{\boldsymbol{\phi}^{-1}}(\boldsymbol{y})\right| \mathrm{d}\boldsymbol{y}. \end{aligned} \tag{1.21}$$

Therefore $\mathsf{P}_{\mathbf{y}}$ has a density $\mathsf{p}_{\mathbf{y}}$ with respect to the Lebesgue measure

$$\mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) = \mathsf{p}_{\mathbf{x}} \circ \boldsymbol{\phi}^{-1}(\boldsymbol{y}) \left|\mathbf{J}_{\boldsymbol{\phi}^{-1}}(\boldsymbol{y})\right| \quad \forall \boldsymbol{y} \in Y. \tag{1.22}$$

In both the cases considered, we have seen that if the function ϕ the random variable x is mapped through is bijective, it is tractable to compute a density on the mapped random variable y as the pre-image $\phi^{-1}(y)$ of a point $y \in Y$ is itself a point. Bijectivity is a very limiting condition however, with many models involving non-bijective transformations of random variables. In Chapter 4 we will discuss methods for performing inference in generative models defined by complex, non-dimension preserving and non-bijective transformations of random variables.

1.1.5 Expectations

A key operation when working with probabilistic models is computing expectations. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space, and $\mathsf{x} : S \to X$ a random variable on this space. The *expected value of* x is defined as

$$\mathbb{E}[\mathsf{x}] = \int_S \mathsf{x}\, \mathrm{dP}. \tag{1.23}$$

Usually it will be more convenient to express expectations in terms of the probability P_x. If $g : X \to \mathbb{R}$ is an integrable function then

$$\int_X g(x)\, \mathsf{P}_\mathsf{x}(\mathrm{d}x) = \int_S g \circ \mathsf{x}(s)\, \mathsf{P}(\mathrm{d}s). \tag{1.24}$$

If we take g as the identity map we therefore have that

$$\mathbb{E}[\mathsf{x}] = \int_X x\, \mathsf{P}_\mathsf{x}(\mathrm{d}x). \tag{1.25}$$

If P_x is given by a density $\mathsf{p}_\mathsf{x} = \frac{\mathrm{dP}_\mathsf{x}}{\mathrm{d}\mu}$ then using (1.9) we also have

$$\mathbb{E}[\mathsf{x}] = \int_X x\, \mathsf{p}_\mathsf{x}(x)\, \mu(\mathrm{d}x), \tag{1.26}$$

which is often the form used for computation.

A further useful implication of (1.24) is what is sometimes termed the *Law of the unconscious statistician*. Let $\mathsf{x} : S \to X$ be a random variable, $\phi : X \to Y$ a measurable function and define $\mathsf{y} = \phi \circ \mathsf{x}$. Then

$$\mathbb{E}[\mathsf{y}] = \int_S \mathsf{y}(s)\, \mathsf{P}(\mathrm{d}s) = \int_S \phi \circ \mathsf{x}(s)\, \mathsf{P}(\mathrm{d}s) = \int_X \phi(x)\, \mathsf{P}_\mathsf{x}(\mathrm{d}x), \tag{1.27}$$

i.e. it can be calculated by integrating ϕ with respect to P_x. This means we can calculate the expected value of a transformed random variable $\mathsf{y} = \phi(\mathsf{x})$ without needing to use the change of variables formulae from Section 1.1.4 to explicitly calculate the probability P_y (or density p_y) and with a relatively weak condition of measurability on ϕ.

Closely related to the expected value are the *variance* and *covariance* of a random variable. The variance of a random variable x is defined

$$\mathbb{V}[\mathsf{x}] = \mathbb{E}\left[(\mathsf{x} - \mathbb{E}[\mathsf{x}])^2\right] = \mathbb{E}\left[\mathsf{x}^2\right] - \mathbb{E}[\mathsf{x}]^2. \tag{1.28}$$

For a pair of random variables x and y their covariance is defined

$$\mathbb{C}[\mathsf{x}, \mathsf{y}] = \mathbb{E}[(\mathsf{x} - \mathbb{E}[\mathsf{x}])(\mathsf{y} - \mathbb{E}[\mathsf{y}])]. \tag{1.29}$$

1.1.6 Conditional expectations and densities

A related concept, and one which will be key in our discussion of inference, is conditional expectation. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space, $(X, \mathcal{F})$ and $(Y, \mathcal{G})$ two measurable spaces and $\mathsf{x} : S \to X$ and $\mathsf{y} : S \to Y$ two random variables. Then the *conditional expectation of* x *given* y, is defined as a measurable function $\mathbb{E}[\mathsf{x} \mid \mathsf{y}] : Y \to X$ satisfying

$$\int_{\mathsf{y}^{-1}(B)} \mathsf{x}(s)\, \mathsf{P}(\mathrm{d}s) = \int_B \mathbb{E}[\mathsf{x} \mid \mathsf{y}](y)\, \mathsf{P}_\mathsf{y}(\mathrm{d}y) \quad \forall B \in \mathcal{G}. \tag{1.30}$$

$\mathbb{E}[\mathsf{x} \mid \mathsf{y}]$ is guaranteed to be uniquely defined almost everywhere in Y by (1.30), i.e. up to P_y-null sets. As a particular case where $B = Y$ we recover what is sometimes termed the *Law of total expectation*

$$\int_S \mathsf{x}\, \mathrm{d}\mathsf{P} = \int_S \mathbb{E}[\mathsf{x} \mid \mathsf{y}] \circ \mathsf{y}\, \mathrm{d}\mathsf{P} \implies \mathbb{E}[\mathsf{x}] = \mathbb{E}[\mathbb{E}[\mathsf{x} \mid \mathsf{y}] \circ \mathsf{y}]. \tag{1.31}$$

We will also use an alternative notation for the conditional expectation evaluated at point $\mathbb{E}[\mathsf{x} \mid \mathsf{y} = y] \equiv \mathbb{E}[\mathsf{x} \mid \mathsf{y}](y)$ but use the latter in this section to stress its definition as a measurable function.

Conditional expectation can be used to define the *regular conditional distribution* of a random variable conditioned on another random variable taking a particular value

$$\mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid y) = \mathbb{E}[\mathbb{1}_A \circ \mathsf{x} \mid \mathsf{y}](y) \quad \forall y \in Y,\, A \in \mathcal{F}. \tag{1.32}$$

We have reused our notation for conditional probability of random variables from Section 1.1.2 here however it should be clear from whether the value conditioned on is a point or a set which is being referred to. A regular conditional distribution $\mathsf{P}_{\mathsf{x}|\mathsf{y}}$ defines a valid probability measure on $(X, \mathcal{F})$ and using (1.30) we have that

$$\mathsf{P}_{\mathsf{x},\mathsf{y}}(A, B) = \int_B \mathsf{P}_{\mathsf{x}|\mathsf{y}}(A \mid y)\, \mathsf{P}_\mathsf{y}(\mathrm{d}y) \qquad \forall A \in \mathcal{F},\, B \in \mathcal{G}. \tag{1.33}$$

We can use this relationship to also motivate a definition of conditional density. We require that a joint density $\mathrm{p}_{\mathrm{x,y}} = \frac{\mathrm{dP}_{\mathrm{x,y}}}{\mathrm{d}(\mu_{\mathrm{x}} \times \mu_{\mathrm{y}})}$ exists and has marginal density $\mathrm{p}_{\mathrm{y}} = \frac{\mathrm{dP}_{\mathrm{y}}}{\mathrm{d}\mu_{\mathrm{y}}}$. Then for all $A \in \mathcal{F}$, $B \in \mathcal{G}$

$$\int_B \mathrm{P}_{\mathrm{x|y}}(A \mid y)\, \mathrm{P}_{\mathrm{y}}(\mathrm{d}y) = \int_B \int_A \mathrm{p}_{\mathrm{x,y}}(x, y)\, \mu_{\mathrm{x}}(\mathrm{d}x)\, \mu_{\mathrm{y}}(\mathrm{d}y) \tag{1.34}$$

If we define the *conditional density* $\mathrm{p}_{\mathrm{x|y}}$ as

$$\mathrm{p}_{\mathrm{x|y}}(x \mid y) = \begin{cases} \frac{\mathrm{p}_{\mathrm{x,y}}(x,y)}{\mathrm{p}_{\mathrm{y}}(y)} & \forall x \in X,\, y \in Y : \mathrm{p}_{\mathrm{y}}(y) > 0 \\ 0 & \forall x \in X,\, y \in Y : \mathrm{p}_{\mathrm{y}}(y) = 0 \end{cases} \tag{1.35}$$

then subsituting this definition in to (1.34) we have

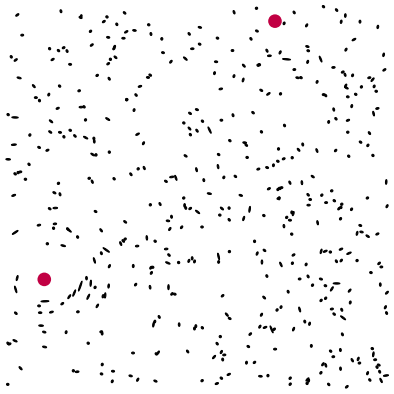
$$\int_B \mathrm{P}_{\mathrm{x|y}}(A \mid y)\, \mathrm{P}_{\mathrm{y}}(\mathrm{d}y) = \int_B \int_A \mathrm{p}_{\mathrm{x|y}}(x \mid y)\, \mu_{\mathrm{x}}(\mathrm{d}x)\, \mathrm{P}_{\mathrm{y}}(\mathrm{d}y). \tag{1.36}$$

Therefore $\mathrm{p}_{\mathrm{x|y}}$ is the density of the regular conditional distribution $\mathrm{P}_{\mathrm{x|y}}$. We also have that if $\mathrm{p}_{\mathrm{x,y}}$ and p_{y} can be defined that

$$\mathbb{E}[\mathrm{x} \mid \mathrm{y}](y) = \int_X x\, \mathrm{p}_{\mathrm{x|y}}(x \mid y)\, \mu_{\mathrm{x}}(\mathrm{d}x) \quad \forall y \in Y : \mathrm{p}_{\mathrm{y}}(y) > 0 \tag{1.37}$$

Note that the initial definition of conditional expectation in (1.30) was not dependent on a joint density $\mathrm{p}_{\mathrm{x,y}}$ being defined.

(a) Hubble Space Telescope image of galaxy cluster MACS J1206 showing visible distortion due to gravitational lensing. Image credit: ESA/Hubble.



(b) Simulated galaxy cluster image from *Observing Dark Worlds* data [107] showing ellipticity of galaxies (black ellipses) distorted by dark matter halos (red circles).

Figure 1.1.: Gravitational lensing in real and simulated galaxy cluster images.

1.2 INFERENCE

Having now introduced the tools we use to construct probabilistic models, we will now describe more concretely the task of inference. To help motivate our exposition we will discuss inference in the context of a specific problem: inferring the location of dark matter halos from the observed gravitational lensing of light emitted by surrounding galaxies. This task was the inspiration for a Kaggle[2] competition, *Observing Dark Worlds* [96, 107] and we will use the simplified formulation of the task used in the competition as the basis for our discussion here.

1.2.1 Example problem: Observing Dark Worlds

We will begin with a brief review of the motivation for the *Observing Dark Worlds* inference problem. Dark matter is a hypothesised form of matter which does not emit or interact with electromagnetic radiation. This prevents direct observation of dark matter by telescopes as it produces no signal in any part of the electromagnetic spectrum [137]. General relativity however predicts that all objects with mass locally distort spacetime. If a very large concentration of mass lies between an observer and a light source the strong distortion of spacetime by the mass significantly alters the paths of photons emitted by the source and causes the apparent image of the source to the observer to be vis-

2 An online platform for predictive modelling competitions `https://www.kaggle.com`.

ibly distorted [12]. This effect is analagous to placing a lens between the light source and observer and this motivates the term *gravitational lensing* to describe the phenonemon. It is believed that large concentrations of dark matter around galaxy clusters termed *dark matter halos*, cause gravitational lensing of the light emitted from background galaxies observed in telescope imagery of galaxy clusters.

Galaxies typically have approximately elliptical shapes in telescope images. It is assumed that the *ellipticities* of observed galaxy images not subject to gravitational lensing are isotropically distributed [12]: there is no preferred orientation of the galaxies to an observer on Earth. The presence of a large mass concentration such as a dark matter halo between background galaxies and a telescope however locally distorts the distribution of the ellipticities of the observed galaxy images. An example of this effect in a galaxy cluster image from the Hubble Space Telescope is shown in Figure 1.1a with the galaxies showing a bias towards being oriented tangentially to the bright region at the centre of the image. Local biases in the spatial distribution of galaxy ellipticities can therefore be used to infer the location of dark matter halos.

In the context of gravitational lensing ellipticity is defined as a complex quantity $\epsilon = \frac{a-b}{a+b}\exp(2i\phi)$ *where* a *is the length of the ellipse major axis,* b *the minor axis length and* ϕ *the orientation of the major axis. Here we will parameterise ellipticity terms of* $e_1 = \mathrm{Re}(\epsilon)$ *and* $e_2 = \mathrm{Im}(\epsilon)$.

The *Observing Dark Worlds* inference task is formulated as follows. The observed data consists of the sky co-ordinates $\{u_g, v_g\}_{g=1}^{G}$ and ellipticity components $\{e_{1,g}, e_{2,g}\}_{g=1}^{G}$ of a set of G galaxies. The task is given this data to infer the coordinates $\{\mathrm{x}_h, \mathrm{y}_h\}_{h=1}^{H}$ of the centers of a known number of dark matter halos present in the sky field of view. Galaxy coordinate and ellipticity data are provided for multiple cluster images each with a known number of halos present and the cluster image data are assumed to be *independently and identically distributed* (iid).

The *Observing Dark Worlds* data is simulated thus the ground truth positions of the dark matter halos are known in reality which was required for the purposes of evaluation in the competition. An example visualisation of one of the simulated galaxy cluster images in the data set is shown in Figure 1.1b. The known positions of the dark matter halos in this image are shown by red circles[3]. As can be seen the halo in the lower left of this image produces a strong visible distortion in the ellipt-

3 A *training set* of cluster image data was provided to competition participants for which the associated true dark matter halo positions were given. This data was distinct from the *test set* cluster data were the halo positions are unknown and which is the basis of the described inference problem and scoring for the competition, with the training data intended to aid initial model exploration and evaluation.

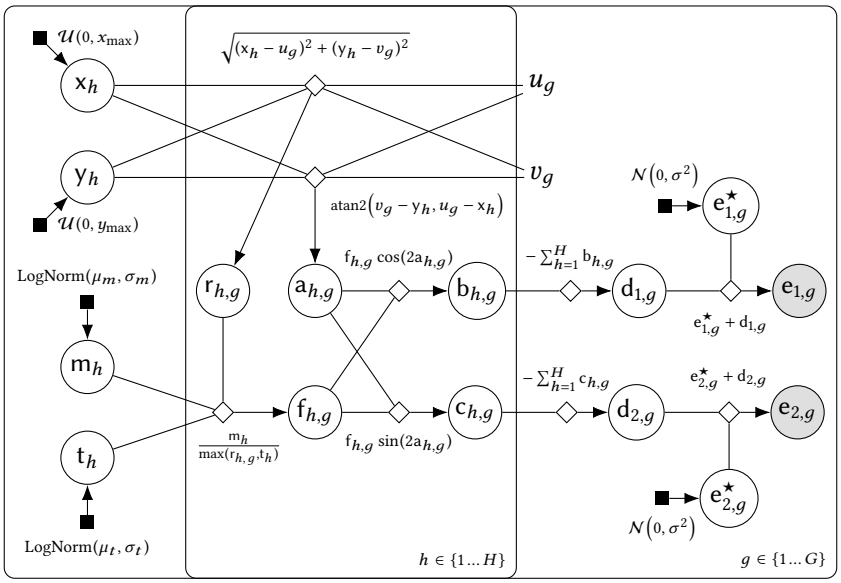


Figure 1.2.: Factor graph of proposed model for *Observing Dark Worlds* gravitational lensing inference problem for a single cluster image.

icities, however the second halo at the top right has a much less visible effect on the surrounding galaxies.

The use of simulated data reduces the difficulty of the *Observing Dark Worlds* inference task compared to working with real gravitational lensing data, however for the purposes of our discussion the task is still sufficiently complex to highlight the computational challenges involved in inference problems. Further the probabilistic approach described here is similar to that used (albeit with significantly more complex models) in methods and tools used in practice to analyse gravitational lensing data to infer dark matter properties [106, 136].

1.2.2 Defining a model

You cannot do inference without making assumptions
—David Mackay

Our starting point for tackling inference problems will be to define a probabilistic model specifying proposed relationships between the observed and non-observed quantities to be inferred. The model codifies the assumptions we make about the problem and any prior beliefs we have. In virtually all real inference problems the model will be a significantly simplified description of a much more complex underlying process, usually motivated by prior empirical observations that the behaviour proposed by the model is a reasonable description of reality. For now we will consider the model as a singular fixed object we

will perform inference with. In reality probabilistic modelling and inference are an iterative process with model criticism a key part of the loop [34, 76]. We will discuss some of the (computational) issues involved in probabilistic model evaluation and comparison at the end of this chapter.

For the *Observing Dark Worlds* problem, a proposed probabilistic model is shown as a directed factor graph in Figure 1.2[4]. This model assumes a simple, physically motivated relationship between the dark matter halo positions and the observed spatial distribution of galaxy ellipticities. As well as random variables for the halo coordinates $\{\mathsf{x}_h, \mathsf{y}_h\}_{h=1}^H$ and galaxy ellipticity components $\{\mathsf{e}_{1,g}, \mathsf{e}_{2,g}\}_{g=1}^G$, the model also introduces additional latent (non-observed) random variables $\{\mathsf{m}_h, \mathsf{t}_h\}_{h=1}^H$, which correpond respectively to the unknown total masses of each halo (multiplied by an arbitrary scaling factor) and an unknown *core radius* for each halo, which specifies a radial distance within which the distortion by the halo is of constant magnitude. The halo coordinates are assumed to be marginally uniformly distributed across the image extents. The halo mass and core radius latent variables are both positive quanties and are assumed to have log normal marginal distributions.

The model proposes that at sky coordinate (u, v) each halo produces a shear distortion of magnitude

$$f_h(u, v) = \frac{\mathsf{m}_h}{\max(r_h, \mathsf{t}_h)} \quad \text{where} \quad r_h = \sqrt{(\mathsf{x}_h - u)^2 + (\mathsf{y}_h - v)^2} \tag{1.38}$$

and acting in a tangential direction to the radial vector from the halo centre $(\mathsf{x}_h, \mathsf{y}_h)$ to (u, v). This functional relationship is just one possibility among many. The post-competition review article [96] revealed that the simulated data actually used dark matter halos with a mixture of two different radial density profiles, neither of which correspond to the distortion model assumed in (1.38). This mismatch between a model and the actual process by which observations were generated will virtually always be the case in real inference problems. We can still make inferences which are consistent with our modelling assumptions, however we should as far as possible also critically review those modelling assumptions to check the validity of the inferences made.

4 The factor graph in Figure 1.2 is based on the models used by the participants with the top-two winning entries in the Kaggle competition, Iain Murray and Tim Salimans, and described in their personal reports on their competition entries [153, 201] and an article evaluating the competition outcomes [96].

For galaxies where the variation of the magnitude of the distorting effect of a halo across the extent of the galaxy's image is small, a reasonable approximation of the gravitational lensing effect of a single halo on the observed ellipticity $(\mathrm{e}_{1,g}, \mathrm{e}_{2,g})$ of a galaxy image with intrinsic (prior to any gravitational lensing effect) ellipticity $(\mathrm{e}^{\star}_{1_g}, \mathrm{e}^{\star}_{2})$ is that

$$\mathrm{e}_{1,g} = \mathrm{e}^{\star}_{1,g} - \mathrm{f}_{h,g}\cos(2\mathrm{a}_{h,g}), \quad \mathrm{e}_{2,g} = \mathrm{e}^{\star}_{2,g} - \mathrm{f}_{h,g}\sin(2\mathrm{a}_{h,g}), \tag{1.39}$$

where $\mathrm{a}_{h,g} = \operatorname{atan2}\left(v_g - \mathrm{y}_h, u_g - \mathrm{x}_g\right)$ is the angle the line from the centre of halo h to galaxy g makes to the horizontal axis and $\mathrm{f}_{h,g} = f_h(u_g, v_g)$ is the magnitude of the shear distortion according to the proposed relationship in Equation (1.38) evaluated at the galaxy image centre [12, 136]. For clusters with multiple halos, a further simple linearity assumption is made, that the shear distortions due to the different halos act additively on the ellipticity components

$$\mathrm{e}_{1,g} = \mathrm{e}^{\star}_{1,g} - \sum_{h=1}^{H} \mathrm{f}_{h,g}\cos(2\mathrm{a}_{h,g}), \ \mathrm{e}_{2,g} = \mathrm{e}^{\star}_{2,g} - \sum_{h=1}^{H} \mathrm{f}_{h,g}\sin(2\mathrm{a}_{h,g}). \tag{1.40}$$

The intrinsic ellipticities of the galaxies are assumed to have a isotropic, zero-mean normal distribution, with standard deviation σ, with normal assumption corresponding well to empirical observations from large scale surveys [12, 136]. For now we will assume the standard deviation σ of the intrinsic ellipticities, and the parameters μ_m, σ_m, μ_t, σ_t of the halo mass and core radius marginal distributions have somehow been set to reasonable values, for example based on prior beliefs about the typical ranges of the variables. We will discuss possible strategies for choosing (or inferring) these parameters in more detail later.

1.2.3 Making predictions

Having now defined a model for the *Observing Dark Worlds* problem, we now consider how to use this model to make predictions about the unobserved dark matter halo positions. The downstream task we are using the inference output for will generally determine what the exact inferential query we wish to evaluate is. In general however, any prediction output which takes into account all of the information we have about the unobserved variables given the assumed model and observed data will be computed as a conditional expectation.

To motivate this statement for the *Observing Dark Worlds* example we will discuss some instances of outputs we might wish to compute. For notational convenience we define the following random vectors for the halo random variables

$$\mathbf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_H \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathsf{y}_1 \\ \vdots \\ \mathsf{y}_H \end{bmatrix}, \quad \mathbf{m} = \begin{bmatrix} \mathsf{m}_1 \\ \vdots \\ \mathsf{m}_H \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} \mathsf{t}_1 \\ \vdots \\ \mathsf{t}_H \end{bmatrix}, \tag{1.41}$$

and similarly for the observed and intrinsic galaxy ellipticities

$$\mathbf{e}_1 = \begin{bmatrix} \mathsf{e}_{1,1} \\ \vdots \\ \mathsf{e}_{1,G} \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} \mathsf{e}_{2,1} \\ \vdots \\ \mathsf{e}_{2,G} \end{bmatrix}, \quad \mathbf{e}_1^\star = \begin{bmatrix} \mathsf{e}^\star_{1,1} \\ \vdots \\ \mathsf{e}^\star_{1,G} \end{bmatrix}, \quad \mathbf{e}_2^\star = \begin{bmatrix} \mathsf{e}^\star_{2,1} \\ \vdots \\ \mathsf{e}^\star_{2,G} \end{bmatrix}. \tag{1.42}$$

An obvious output we may wish to compute is the expected (mean) values of the halo positions given the observed ellipticities. From a decision theoretic standpoint these values correspond to the position predictions which minimise the expected squared error loss from the true positions given the model and observed data. As conditonal expectations these are simply

$$\boldsymbol{m}_\mathbf{x} = \mathbb{E}\big[\mathbf{x} \,|\, \mathbf{e}_1 = \boldsymbol{e}_1,\, \mathbf{e}_2 = \boldsymbol{e}_2\big] \;\text{ and }\; \boldsymbol{m}_\mathbf{y} = \mathbb{E}\big[\mathbf{y} \,|\, \mathbf{e}_1 = \boldsymbol{e}_1,\, \mathbf{e}_2 = \boldsymbol{e}_2\big]. \tag{1.43}$$

We may also be interested in the covariances of the halo positions conditioned on the observations, these giving some idea of our remaining uncertainty in the positions and any correlations between them after conditioning on the observed ellipticities. Again these can be expressed as conditional expectations

$$\begin{aligned} \boldsymbol{C}_{\mathbf{x},\mathbf{x}} &= \mathbb{E}\Big[(\mathbf{x} - \boldsymbol{m}_\mathbf{x})(\mathbf{x} - \boldsymbol{m}_\mathbf{x})^\mathsf{T} \,|\, \mathbf{e}_1 = \boldsymbol{e}_1,\, \mathbf{e}_2 = \boldsymbol{e}_2\Big], \\ \boldsymbol{C}_{\mathbf{y},\mathbf{y}} &= \mathbb{E}\Big[(\mathbf{y} - \boldsymbol{m}_\mathbf{y})(\mathbf{y} - \boldsymbol{m}_\mathbf{y})^\mathsf{T} \,|\, \mathbf{e}_1 = \boldsymbol{e}_1,\, \mathbf{e}_2 = \boldsymbol{e}_2\Big], \\ \text{and} \quad \boldsymbol{C}_{\mathbf{x},\mathbf{y}} &= \mathbb{E}\Big[(\mathbf{x} - \boldsymbol{m}_\mathbf{x})(\mathbf{y} - \boldsymbol{m}_\mathbf{y})^\mathsf{T} \,|\, \mathbf{e}_1 = \boldsymbol{e}_1,\, \mathbf{e}_2 = \boldsymbol{e}_2\Big]. \end{aligned} \tag{1.44}$$

In the *Observing Dark Worlds* competition, participants' halo position predictions were evaluated by comparing to the known true halo positions using a metric provided as part of the competition instructions. If we denote the metric $\ell(\boldsymbol{x}, \boldsymbol{y}; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ where $(\boldsymbol{x}, \boldsymbol{y})$ are the true halo positions and $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ the predicted positions, then the optimal predictions

given the assumed model and observed data can be calculated by minimising the expected metric value conditioned on the observed data

$$\bar{\ell}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \mathbb{E}\big[\ell(\mathbf{x}, \mathbf{y}; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \,|\, \mathbf{e}_1 = \boldsymbol{e}_1, \mathbf{e}_2 = \boldsymbol{e}_2\big]. \tag{1.45}$$

Due to linearity of the expectation operator, we can also calculate the derivatives of the expected metric with respect to the predictions as

$$\begin{aligned} \frac{\partial \bar{\ell}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{x}}} &= \mathbb{E}\left[\frac{\partial \ell(\mathbf{x}, \mathbf{y}; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{x}}} \;\middle|\; \mathbf{e}_1 = \boldsymbol{e}_1, \mathbf{e}_2 = \boldsymbol{e}_2\right] \\ \text{and} \quad \frac{\partial \bar{\ell}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} &= \mathbb{E}\left[\frac{\partial \ell(\mathbf{x}, \mathbf{y}; \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} \;\middle|\; \mathbf{e}_1 = \boldsymbol{e}_1, \mathbf{e}_2 = \boldsymbol{e}_2\right]. \end{aligned} \tag{1.46}$$

For some metrics we will be able to analytically solve for the stationary points to find the predictions minimising $\bar{\ell}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and more generally we can use the derivatives in an iterative optimisation scheme.

Evaluating conditional expectations of functions of the unobserved variables in a model given observed data is therefore the key computational task in making inferences about the variables in a probabilistic model. The *Observing Dark Worlds* model factor graph in Figure 1.2 defines a joint density across all the variables in the model. We can therefore use Equation (1.37) to write the conditional expectation of a measurable function f of the halo position random variables $\mathbf{x}$ and $\mathbf{y}$ as[5]

$$\begin{aligned} \mathbb{E}\big[f(\mathbf{x}, \mathbf{y}) \,|\, \mathbf{e}_1 = \boldsymbol{e}_1, \mathbf{e}_2 = \boldsymbol{e}_2\big] = \\ \iint f(\boldsymbol{x}, \boldsymbol{y})\, \mathrm{p}_{\mathbf{x},\mathbf{y}|\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x}, \boldsymbol{y} \,|\, \boldsymbol{e}_1, \boldsymbol{e}_2)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}, \end{aligned} \tag{1.47}$$

where the conditional density $\mathrm{p}_{\mathbf{x},\mathbf{y}|\mathbf{e}_1,\mathbf{e}_2}$ is defined as

$$\mathrm{p}_{\mathbf{x},\mathbf{y}|\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x}, \boldsymbol{y} \,|\, \boldsymbol{e}_1, \boldsymbol{e}_2) = \frac{\mathrm{p}_{\mathbf{x},\boldsymbol{y},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{e}_1, \boldsymbol{e}_2)}{\mathrm{p}_{\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{e}_1, \boldsymbol{e}_2)}, \tag{1.48}$$

with $\mathrm{p}_{\mathbf{e}_1,\mathbf{e}_2}$ defined in terms of $\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{e}_1,\mathbf{e}_2}$ by marginalising out $\mathbf{x}$ and $\mathbf{y}$

$$\mathrm{p}_{\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{e}_1, \boldsymbol{e}_2) = \iint \mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{e}_1, \boldsymbol{e}_2)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}. \tag{1.49}$$

5 For brevity the sets on which integrals are evaluated have been omitted in this section and should be assumed to be the full co-domain of the corresponding random vector.

We can express $\mathrm{p}_{\mathbf{x},\boldsymbol{y},\mathbf{e}_1,\mathbf{e}_2}$ itself as a marginal of the joint density over all random variables in the model (which we can read off as a product of factors from Figure 1.2) by marginalising out $\mathbf{m}$, $\mathbf{t}$, $\mathbf{e}_1^\star$ and $\mathbf{e}_2^\star$

$$\begin{aligned}\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{e}_1,\boldsymbol{e}_2) = \iiiint & \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\,\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})\,\mathrm{p}_{\mathbf{m}}(\boldsymbol{m})\,\mathrm{p}_{\mathbf{t}}(\boldsymbol{t}) \\ & \mathrm{p}_{\mathbf{e}_1|\mathbf{e}_1^\star,\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}(\boldsymbol{e}_1 \mid \boldsymbol{e}_1^\star,\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}) \\ & \mathrm{p}_{\mathbf{e}_2|\mathbf{e}_2^\star,\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}(\boldsymbol{e}_2 \mid \boldsymbol{e}_2^\star,\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}) \\ & \mathrm{p}_{\mathbf{e}_1^\star}(\boldsymbol{e}_1^\star)\,\mathrm{p}_{\mathbf{e}_2^\star}(\boldsymbol{e}_2^\star)\,\mathrm{d}\boldsymbol{e}_1^\star\mathrm{d}\boldsymbol{e}_2^\star\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{m}.\end{aligned} \tag{1.50}$$

The integration with respect to $\boldsymbol{e}_1^\star$ and $\boldsymbol{e}_2^\star$ can be performed analytically as the conditional density factors $\mathrm{p}_{\mathbf{e}_2|\mathbf{e}_2^\star,\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}$ and $\mathrm{p}_{\mathbf{e}_2|\mathbf{e}_1^\star,\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}$ correspond to deterministic factors which depend linearly on $\mathbf{e}_1^\star$ and $\mathbf{e}_2^\star$. Expressing the deterministic factors as Dirac deltas and denoting the functions corresponding to the computational graphs in Figure 1.2 mapping from $\{\mathrm{x}_h, \mathrm{y}_h, \mathrm{m}_h, \mathrm{t}_h\}_{h=1}^H$ to $\{\mathrm{d}_{1,g}\}_{g=1}^G$ and $\{\mathrm{d}_{2,g}\}_{g=1}^G$, $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ respectively, we have that

$$\begin{aligned}&\mathrm{p}_{\mathbf{e}_1|\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}(\boldsymbol{e}_1 \mid \boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}) \\ &\quad= \int \mathrm{p}_{\mathbf{e}_1|\mathbf{e}_1^\star,\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}(\boldsymbol{e}_1 \mid \boldsymbol{e}_1^\star,\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t})\,\mathrm{p}_{\mathbf{e}_1^\star}(\boldsymbol{e}_1^\star)\,\mathrm{d}\boldsymbol{e}_1^\star \\ &\quad= \int \delta\Big(\boldsymbol{e}_1 - \boldsymbol{e}_1^\star - \boldsymbol{d}_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t})\Big)\,\mathcal{N}\Big(\boldsymbol{e}_1^\star \mid \mathbf{0}, \sigma^2\mathbf{I}\Big)\,\mathrm{d}\boldsymbol{e}_1^\star \\ &\quad= \mathcal{N}\Big(\boldsymbol{e}_1 \mid \boldsymbol{d}_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}), \sigma^2\mathbf{I}\Big),\end{aligned} \tag{1.51}$$

and likewise for $\mathrm{p}_{\mathbf{e}_2|\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}$

$$\mathrm{p}_{\mathbf{e}_2|\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t}}(\boldsymbol{e}_2 \mid \boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}) = \mathcal{N}\Big(\boldsymbol{e}_2 \mid \boldsymbol{d}_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}), \sigma^2\mathbf{I}\Big). \tag{1.52}$$

Rewriting (1.50) in terms of (1.51) and (1.52) and substituting the definitions for the factors from Figure 1.2 we have that

$$\begin{aligned}\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{e}_1,\boldsymbol{e}_2) &= \iint \mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t},\boldsymbol{e}_1,\boldsymbol{e}_2)\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{m} \\ &= \iint \prod_{h=1}^{H} \big(\mathcal{U}(x_h \mid 0, x_{\max})\,\mathcal{U}(y_h \mid 0, y_{\max})\big) \\ &\qquad \mathcal{N}\Big(\boldsymbol{e}_1 \mid \boldsymbol{d}_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}), \sigma^2\mathbf{I}\Big)\,\mathcal{N}\Big(\boldsymbol{e}_2 \mid \boldsymbol{d}_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{m},\boldsymbol{t}), \sigma^2\mathbf{I}\Big) \\ &\qquad \mathrm{LogNorm}\Big(\boldsymbol{m} \mid \mu_m\mathbf{1}, \sigma_m^2\mathbf{I}\Big)\,\mathrm{LogNorm}\Big(\boldsymbol{t} \mid \mu_m\mathbf{1}, \sigma_m^2\mathbf{I}\Big)\,\mathrm{d}\boldsymbol{t}\,\mathrm{d}\boldsymbol{m}.\end{aligned} \tag{1.53}$$

We cannot simplify this integral any further analytically. Thefore the conditional expectation we wish to compute in terms of integrals of functions we can evaluate pointwise is

$$\mathbb{E}[f(\mathbf{x}, \mathbf{y}) \mid \mathbf{e}_1 = \boldsymbol{e}_1, \mathbf{e}_2 = \boldsymbol{e}_2] = \frac{\iiiint f(\boldsymbol{x}, \boldsymbol{y})\, \mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m}, \boldsymbol{t}, \boldsymbol{e}_1, \boldsymbol{e}_2)\, \mathrm{d}\boldsymbol{t}\, \mathrm{d}\boldsymbol{m}\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}}{\iiiint \mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{m},\mathbf{t},\mathbf{e}_1,\mathbf{e}_2}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m}, \boldsymbol{t}, \boldsymbol{e}_1, \boldsymbol{e}_2)\, \mathrm{d}\boldsymbol{t}\, \mathrm{d}\boldsymbol{m}\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{y}}. \tag{1.54}$$

Each of the vector integration variables in (1.54) is of dimension H and so the overall dimension of the space being integrated over in both the numerator and denominator is $4H$. In the *Observing Dark Worlds* data the number of halos per cluster image H is between one and three so to evaluate conditional expectations of the halo positions we need to compute integrals over spaces with four, eight or twelve dimensions.

For four or eight dimensions it may be feasible to use quadrature methods [51], which involve evaluating the integrand across a fixed grid of points and then computing a weighted sum of these values, to numerically approximate the integrals to reasonable accuracy. For a fixed grid resolution however the cost of quadrature scales exponentially with the dimension of the space being integrated over - if N points are used per dimension, using quadrature to evaluate (1.54) will involve N^{4H} evaluations of the integrand.

Assuming the computational cost of the function $f(\mathbf{x}, \mathbf{y})$ the conditional expectation is being calculated of is negligible, the dominant cost in the evaluation of the integrands in (1.54) will be in evaluating $\boldsymbol{d}_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m}, \boldsymbol{t})$ and $\boldsymbol{d}_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{m}, \boldsymbol{t})$ which involve computing the distances and angles between all galaxy-halo pairs. Using a very simplistic (and conservative) assumption that each arithmetic, square root, comparison and trigonometric operation has a unit floating point operation cost, the computation graph in Figure 1.2 which is present on the overlap between the two plates will involve 15 floating point operations in a single forward pass and is evaluated GH times for an overall cost per integrand evaluation of $15GH$ floating point operations.

A rough but conservative lower bound on the floating point operation count of evaluating the integrals in (1.54) using quadrature is therefore $15GHN^{4H}$. The number of galaxies G per cluster image in the *Observing Dark Worlds* data varies between 300 and 740. If we assume $G = 500$

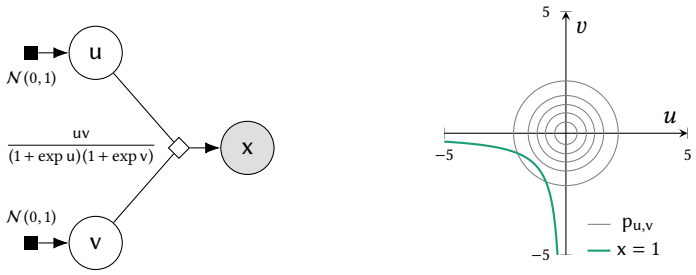
and that we use $N = 20$ grid points per dimension, for the $H = 2$ cases evaluating the conditional expectation will involve $\sim 1 \times 10^9$ floating point operations, for $H = 2$, $\sim 4 \times 10^{14}$ floating point operations and for $H = 3$, $\sim 9 \times 10^{19}$ floating point operations.

At the time of writing, the theoretical peak floating point operation performance of a top-end multi-core server *central processing unit* (CPU) is around 5×10^{11} *floating point operations per second* (FLOPS). Assuming this peak performance could be obtained when using quadrature to approximate (1.54) , our back-of-the-envelope calculation suggests a trivial two millisecond compute time for clusters with one halo, a more noticeable thirteen minute computation for clusters with two halos, and an impractical six year wait for clusters with three halos.

Current top-end *graphics processing units* (GPUs) have a peak theoretical (single-precision) floating point performance of around 10^{13} FLOPS which at best would cut the computation time for a $H = 3$ case by a factor of 20 to around 100 days. With a cluster of CPU or GPU nodes, or faster individual nodes due to future growth in processing power, the compute time could potentially be brought down further to more reasonable timescales. However the key point in this example is the exponential growth with dimension - even moving from eight to twelve dimensions made compute time impractical. Clearly therefore approximating the integrals in expectations like (1.54) using quadrature methods is not viable when performing inference in models with large numbers of unobserved variables, with the example here showing that even integrals over what might initially seem a low dimensionality of twelve can be problematic.

1.2.4 Implicit models

In the *Observing Dark Worlds* example it was possible to analytically integrate out the random vectors $\mathbf{e}_1^\star$ and $\mathbf{e}_2^\star$, which correspond to the intrinsic galaxy ellipticities, due to the observed ellipticities $\mathbf{e}_1$ and $\mathbf{e}_2$ being deterministic linear functions of $\mathbf{e}_1^\star$ and $\mathbf{e}_2^\star$ respectively (for fixed $\mathbf{x}$, $\mathbf{y}$, $\mathbf{m}$ and $\mathbf{t}$). Rather than calculating the conditional densities on $\mathbf{e}_1$ and $\mathbf{e}_2$ by integrating out $\mathbf{e}_1^\star$ and $\mathbf{e}_2^\star$, we could equivalently have applied the change of variables formula (1.22) for a bijective transformation; in this case the Jacobian determinant is simply one.



(a) Directed factor graph defining model with two latent variables (u, v), and an observed variable x.

(b) Plot of density on latent variables (contours) and set of values for which $x = 1$ (green curve).

Figure 1.3.: Simple example of an implicit probabilistic model where the observed variable is a non-bijective function of two latent variables.

If the observed variables had instead been the output of a deterministic factor where there was no bijective dependence on any of the parent variables (inputs to the deterministic factor), the change of variables formula (1.22) would no longer have applied. Generally in such cases it will not be possible to analytically marginalise out a parent variable to give an explicit conditional density on the observed variable(s). Models which do not admit explicit conditional densities on the observed variables are sometimes described as *implicit models* [57], with simulator models being a common case.

An illustration of such a case for a simple three variable model is shown in Figure 1.3. Here the observed variable x is a deterministic function of two latent (unobserved) variables u and v. There is no analytic solution in terms of elementary functions for u as a function of x and v or for v as a function of x and u. This means the Dirac delta term corresponding to the deterministic factor cannot be integrated out. Due to the presence of the Dirac delta the joint density $p_{x,u,v}$ is not well defined (the joint probability $P_{x,u,m}$ is not absolutely continuous with respect to any measure) which complicates evaluations of conditional expectations such as $\mathbb{E}[f(u, v) \mid x = 1]$.

In particular the set of u and v values corresponding to solutions to $x = 1$ (illustrated as the green curve in Figure B.1a) has zero Lebesgue measure. Therefore even though the dimensionality is low in this case we can not use simple quadrature to evaluate conditional expectations without some further form of approximation. We will revisit methods for performing inference in implicit models in Chapter 4.

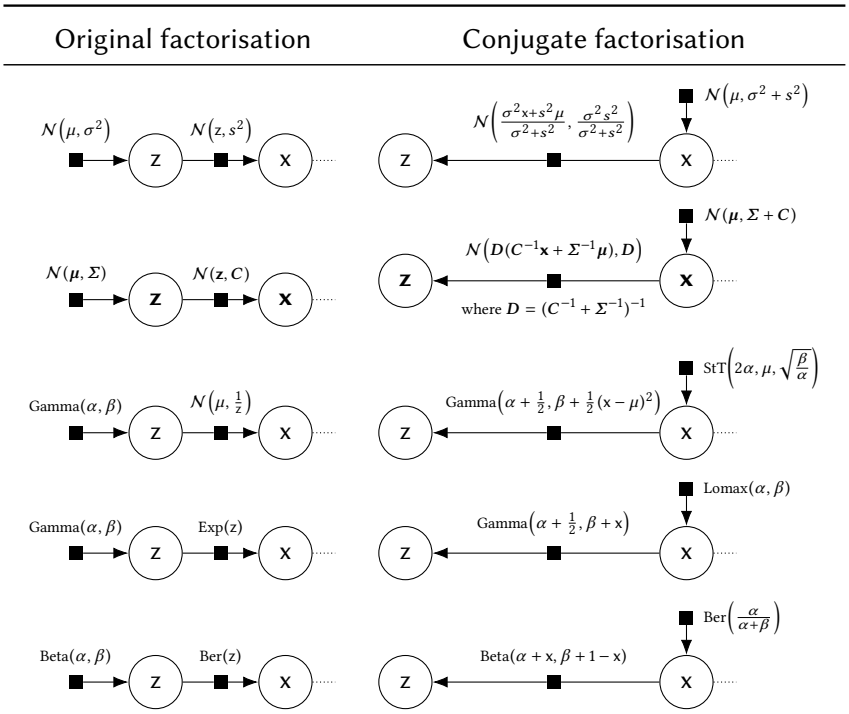


Table 1.1.: Factor graph illustrations of conjugate distributions.

1.2.5 Conjugacy and exact inference

Some densities have a *conjugacy* property that can simplify inference. If x and z are two random variables in a model then the joint density on the two variable can be factorised as

$$\mathsf{p}_{\mathsf{x},\mathsf{z}}(x, z) = \mathsf{p}_{\mathsf{x}|\mathsf{z}}(x \mid z)\mathsf{p}_{\mathsf{z}}(z) = \mathsf{p}_{\mathsf{z}|\mathsf{x}}(z \mid x)\mathsf{p}_{\mathsf{x}}(x). \tag{1.55}$$

For certain pairs of $\mathsf{p}_{\mathsf{x}|\mathsf{z}}$ and p_{z} the corresponding $\mathsf{p}_{\mathsf{z}|\mathsf{x}}$ and p_{x} have closed-form expressions. Some examples are shown in Table 1.1. In all of these examples $\mathsf{p}_{\mathsf{x}|\mathsf{z}}$ is a density for an *exponential family distribution*. For every exponential family distribution density $\mathsf{p}_{\mathsf{x}|\mathsf{z}}$ there exists a p_{z} such that $\mathsf{p}_{\mathsf{z}|\mathsf{x}}$ is of the same parametric family as p_{z}.

A conditional density $\mathsf{p}_{\mathbf{u}|\mathbf{v}}$ is from the exponential family if it can be written as $\mathsf{p}_{\mathbf{u}|\mathbf{v}}(\boldsymbol{u} \mid \boldsymbol{v}) = \frac{h(\boldsymbol{u}) \exp(\boldsymbol{\eta}(\boldsymbol{v})^{\mathsf{T}} \boldsymbol{t}(\boldsymbol{u}))}{z(\boldsymbol{v})}$, with $\boldsymbol{\eta}(\boldsymbol{v})$ termed the natural parameters *and $\boldsymbol{t}(\boldsymbol{u})$ termed the* sufficient statistics.

If x corresponds to an observed variable in the model, then if evaluating conditional expectations $\mathbb{E}[f(\mathsf{z}) \mid \mathsf{x}]$ the conjugacy property means that conditional density $\mathsf{p}_{\mathsf{z}|\mathsf{x}}$ which f should be integrated against has a closed form expression. Often for simple f, e.g. $f(z) = z$ or $f(z) = z^2$, the conditional expectations will have closed form solutions in such cases (i.e. corresponding to moments of the distribution defined by $\mathsf{p}_{\mathsf{z}|\mathsf{x}}$). Even when f is more complex, typically generating independent random samples from $\mathsf{p}_{\mathsf{z}|\mathsf{x}}$ in such cases will be possible, simplifying

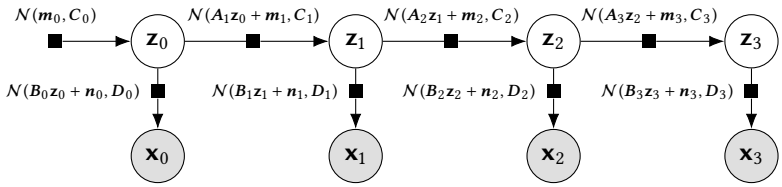


Figure 1.4.: Factor graph for a latent linear dynamical system model.

the use of Monte Carlo methods (which will be introduced in the next chapter). If both z and x are latent variables then the conjugacy property is also useful as in this case z can be analytically marginalised out of conditional expectations of functions which do not depend on z.

A Markov process is a stochastic process such that future states are conditionally independent of past states given the current state.

Various algorithms have been developed to exploit conjugacy in restricted classes of probabilistic models to perform efficient exact inference. Inference in *latent linear dynamical systems*, an example of which is shown as a factor graph in Figure 1.4, can be efficiently performed with the recursive Kalman filtering and smoothing algorithms [109]. Closely related are the forward and backward updates for inference in *hidden Markov models* [213] which have the same factorisation structure as latent linear dynamical systems but use a discrete rather than real-valued latent state. The *junction tree algorithm* [120] is a general purpose algorithm for performing exact inference in undirected graphical models of discrete random variables which exploits conditional independency structure to efficiently decompose the inference into local computations; the computational cost of the algorithm scales exponentially with the number of nodes in the largest clique in the model and so is mainly relevant for graphs with relatively small maximal cliques.

1.2.6 A note on Bayesian terminology

The system of inference that we have described would typically be identified as being *Bayesian*. This name arises because of the central importance of *Bayes' theorem* (1.7) in defining the conditional probability of unobserved variables given observations.

While often the inference problems we will discuss in this thesis will be motivated from a Bayesian standpoint, the use of probabilistic modelling and resulting need to deal with the computational challenges of computing integrals in high dimensional spaces are not unique to Bayesian statistics. For instance many of the approximate inference methods we will discuss in the next two chapters were originally de-

veloped for use in studying statistical physics problems such as the phase transitions of the *Ising spin model* (which we earlier encounted under the alternative name of the Boltzmann machine model). As our main interest in this thesis will be the computational aspects of inference rather than specific applications, we will therefore generally prefer to refer to probabilistic modelling and inference in general terms rather than Bayesian inference in particular.

For completeness we will briefly review the nomenclature typically used in Bayesian inference. For a model with observed variables $\mathbf{x} \in X$ and unobserved variables $\mathbf{z} \in Z$, often referred to as *model parameters* in a Bayesian setting, the standard presentation of Bayesian inference assumes that the joint probability is specified by a density $p_{x,z} = \frac{dP_{\mathbf{x},\mathbf{z}}}{d(\mu_{\mathbf{x}} \times \mu_{\mathbf{z}})}$ which naturally factorises as $p_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{z}) = p_{\mathbf{x}|\mathbf{z}}(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathbf{z}}(\boldsymbol{z})$.

In this case the probability distribution defined by the marginal density on the model parameters $p_{\mathbf{z}}$ is typically termed the *prior distribution*, with the intepretation that it captures our beliefs about the parameters before observing data. The conditional density on the observed variables given parameters $p_{\mathbf{x}|\mathbf{z}}$ is often referred to as the *likelihood* or sometimes the *statistical model* [192].

The distribution defined by the conditional density on the parameters given the observations $p_{\mathbf{z}|\mathbf{x}}$ is termed the *posterior distribution*, with this naming indicative of it representing our beliefs about the parameters after observing data. The posterior density can be calculated from the prior and likelihood using the definition for conditional density given in (1.35). Analagously to the definition for probabilities in (1.7) the resulting relationship is often also termed Bayes' theorem

$$p_{\mathbf{z}|\mathbf{x}}(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{p_{\mathbf{x}|\mathbf{z}}(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathbf{z}}(\boldsymbol{z})}{p_{\mathbf{x}}(\boldsymbol{x})} \quad \forall \boldsymbol{z} \in Z,\, \boldsymbol{x} \in X : p_{\mathbf{x}}(\boldsymbol{x}) > 0. \tag{1.56}$$

Inference is then typically posed as the problem of computing the posterior distribution. This is typically not available in a closed form as evaluating the denominator in (1.56), $p_{\mathbf{x}}$, sometimes termed the *model evidence*, requires integrating the joint density over the parameters

$$p_{\mathbf{x}}(\boldsymbol{x}) = \int_Z p_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} = \int_Z p_{\mathbf{x}|\mathbf{z}}(\boldsymbol{x} \mid \boldsymbol{z})\, p_{\mathbf{z}}(\boldsymbol{z})\, d\boldsymbol{z}. \tag{1.57}$$

Other than in special cases such as when $p_{\mathbf{z}}$ and $p_{\mathbf{x}|\mathbf{z}}$ are both in the exponential family and form a conjugate pair such as those shown in

Table 1.1, this integral does not have a closed form solution and typically Z will be of a dimensionality which means quadrature will be too expensive. As $p_{\mathbf{x}}(\boldsymbol{x})$ is found by marginalising out $\mathbf{z}$ from the product of the prior and likelihood, it is sometimes termed the *marginal likelihood*, though this name obscures that it is also dependent on the prior.

The usage of the term likelihood in Bayesian inference is overloaded: while it is often informally used to refer to the conditional density in $p_{\mathbf{x}|\mathbf{z}}$ in Bayes' theorem (which is often summarised as *posterior is proportional to likelihood times prior*), the likelihood is usually formally defined as a function of the parameters (given fixed values for the observed variables) with notation such as $\ell(\boldsymbol{z} \mid \boldsymbol{x}) = p_{\mathbf{x}|\mathbf{z}}(\boldsymbol{x} \mid \boldsymbol{z})$ sometimes used to emphasise this. This usage is particularly common when discussing *maximum likelihood* methods which find values of the parameters which maximise the likelihood (given data). In this latter interpretation it makes sense to refer to the likelihood of the parameters, but not the likelihood of observed variables (or observations / data). This leads to recommendations to refer to 'the likelihood of the parameters given the [observed] data' [130], which given the construct being discussed is actually a density on the observed data given parameter values is, in our opinion, not particularly clear.

In this thesis we will avoid using the terms *likelihood* and *marginal likelihood*. We will generally prefer to refer to individual conditional and marginal densities (or probabilities) explicitly using the notation introduced earlier in this chapter, though we will use the terms prior, posterior and model evidence as qualifiers where appropriate.

We will also prefer to consider a probabilistic model as defining a joint probability measure on observed and unobserved variables $P_{\mathbf{x},\mathbf{z}}$, without necessarily making further assumptions of how this joint probability is specified. Commonly the model will be defined via an explicit joint density $p_{\mathbf{x},\mathbf{z}}$ given for example by a directed factor graph, but for example in the case of simulator models, the model may instead by defined procedurally by code for sampling from the joint probability, with the joint density only implictly defined.

A further common special case is when the model is specified by a marginal density on the unobserved variables $p_{\mathbf{z}}$ (a prior in Bayesian terms) and a conditional density on observed variables given unobserved variables $p_{\mathbf{x}|\mathbf{z}}$ (a 'likelihood'), this factorisation naturally leading to the

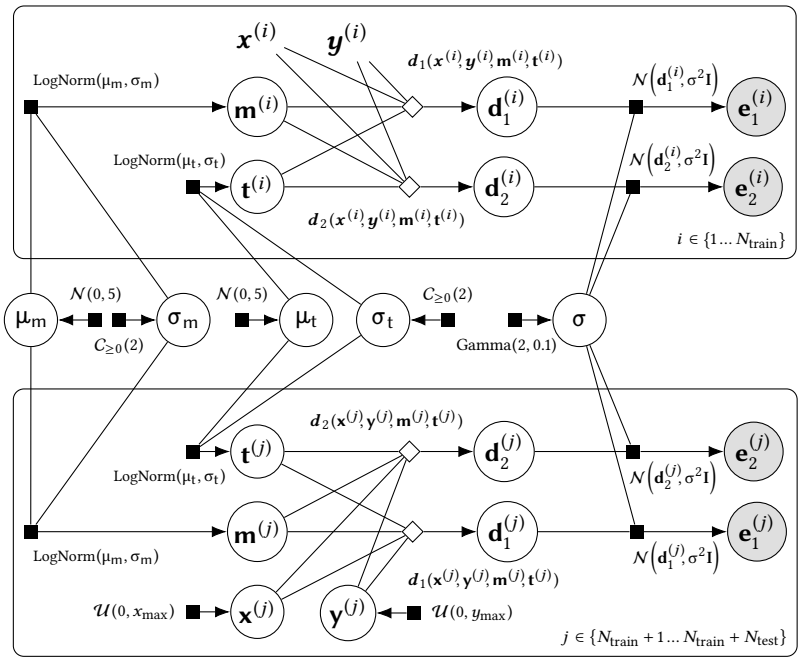


Figure 1.5.: Observing Dark Worlds hierarchical model.

form given for Bayes' theorem in (1.56). However the assumption that the joint density can be easily factorised into closed form densities $\mathrm{p}_{\mathbf{z}}$ and $\mathrm{p}_{\mathbf{x}|\mathbf{z}}$ is not a requirement to evaluate the posterior density $\mathrm{p}_{\mathbf{z}|\mathbf{x}}$, and there are inference problems where this will not apply.

As a final comment on terminology, we will prefer to refer to unobserved or latent variables (which we will use interchangeably) rather than parameters when these quantities are being inferred as opposed to set to fixed values. Parameters and latent variables are sometimes used distinctly, in particular defining parameters as unobserved variables that are of a fixed number and which all observed variables are dependent on, and latent variables as unobserved variables which only the observed variables associated with a particular data point are dependent upon and which increase in number with the quantity of observed datapoints [27]. When this distinction is helpful we will instead follow the terminology used in [29] and refer to the former as *global latent variables* and the latter as *local latent variables*.

1.2.7 Hierarchical modelling

When introducing the proposed model (Figure 1.2) for the *Observing Dark Worlds* problem we glossed over how the parameters σ, μ_m, σ_m, μ_t and σ_t were chosen, assuming for simplicity they had somehow been

set to reasonable values. Rather than considering these quantities as fixed parameters, a more Bayesian approach would be instead to also treat them as further unobserved variables to infer.

In particular if we now consider σ, μ_m, σ_m, μ_t and σ_t as (global) latent variables, we can encode our prior beliefs about what are plausible values for the variables by setting densities on each. For example, for the intrinsic ellipticity scale variable σ we might use Gamma$(2, 0.1)$. This has support only for positive values as required for a scale variable, and reflects the known properties of the intrinsic ellipticities - that they have non-zero variability (with the density tending to zero as $\sigma \to 0$) and that they are bounded to unit magnitude and hence their standard deviation would be expected to be $\ll 1$. We might have less strong prior beliefs about the range of plausible values for the halo mass and core radii distribution, and so choose to use concordantly less constraining prior distributions on the scale and location variables for these distributions. For example a prior of $\mathcal{N}(0, 5)$ on the location variables μ_m and μ_t and a prior of $C_{\geq 0}(2)$ on the scale variables σ_m and σ_t, supports a wide range of values for these variables as being reasonable while still making a weak assumption that extreme values are implausible.

In the *Observing Dark Worlds* competition, two distinct sets of data were provided — a training set of $N_{\text{train}} = 300$ of cluster images in which the observed galaxy ellipticities, galaxy coordinates and halo positions were all provided, and a test set of $N_{\text{test}} = 120$ images where only the observed galaxy ellipticities and galaxy coordinates are provided. Assuming both the training and test data are iid we can form a *hierarchical model* for the problem which specifies a joint density across the observed and local latent variables for all of the training and test set clusters as well as the global latent variables σ, μ_m, σ_m, μ_t and σ_t. A factor graph for the proposed hierarchical model is shown in Figure 1.5. Compared to the factor graph in Figure 1.2 for a single cluster image some detail in the model structure for each cluster has been hidden to make the higher level structure clearer, and the galaxy positions, which are all observed, have been omitted. The vector notation introduced when discussing inference in the model has also been used to remove the need for plates indexed across galaxies and halos.

When making predictions using the hierarchical model, the local latent variables $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{m}^{(i)}, \mathbf{t}^{(i)}\}$ are now jointly dependent when not conditioning on the values of the global latent variables σ, μ_m, σ_m, μ_t and σ_t.

It is therefore necessary to jointly integrate across all of the local latent variables associated with each training and test set cluster when evaluating conditional expectations given the observed data. The resulting integral is over a space of several thousand dimensions. Given the infeasibility of using quadrature to compute conditional expectations of the halo positions for even a single cluster image, clearly in this case the need for computational methods for inference which scale to high dimensionalities is even more stark.

1.2.8 Model comparison

So far we have discussed making inferences about the unobserved variables in a single fixed model. An important second level of inference is comparing between competing models for the same observed data. As we will see this can be treated consistently with a simple extension to the existing probabilistic framework we have discussed.

As a simple motivating example, we can consider comparing our proposed model for the *Observing Dark Worlds* problem, in which we assumed that the magnitude of the shear distortion had the functional dependence given in (1.38), to an alternative model which assumes

$$f_h(u, v) = \frac{\mathrm{m}_h}{r_h} \quad \text{where} \quad r_h = \sqrt{(\mathrm{x}_h - u)^2 + (\mathrm{y}_h - v)^2}. \tag{1.58}$$

Ockham's Razor is a philoshopical principle, commonly attributed to the 14th century Franciscan friar William of Ockham, that states if there exist multiple explanations for observations, all else being equal we should prefer the simplest.

This model is simpler in the sense of requiring only one additional latent variable, m_h, per halo (compared to both m_h and t_h), but produces a non-realistic infinite magnitude distortion at zero radial distances. Given the observed data, we would ideally like to be able to make a judgement as to which of the two proposed models better describes the data. To be useful this comparison must take into account the relative complexity of the models; a model with more free variables will generally be able to fit observed data more closely, however *Ockham's Razor* (and corresponding empirical evidence of the loss of predictive power of overly complex models) suggests we should prefer simpler models where possible. By marginalising over the unobserved variables in a model, the probabilistic model comparison framework we will describe automatically embodies Ockham's Razor [130].

The general model comparison set up we will assume is that we have a finite set of M models, indexed by an *indicator* variable $\mathrm{m} \in \{1 \dots M\}$. All models share the same observed variables $\mathbf{x}$, and there are a set

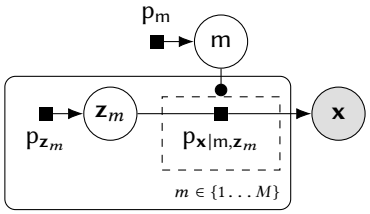


Figure 1.6.: Factor graph for inference over multiple models.

of per model vectors of unobserved variables $\{\mathbf{z}_m\}_{m=1}^M$ which are assumed to be independent (before conditioning on observations). More complex structures could be assumed such as the models sharing a set of common unobserved variables, however we only consider the case of independent models here. The joint density on the observations, model indicator and latent variables is then assumed to factorise as

$$\begin{aligned} p_{\mathbf{x},m,\mathbf{z}_1,\dots,\mathbf{z}_M}(\boldsymbol{x}, m, z_1, \dots, z_M) = \\ p_{\mathbf{x}|m,\mathbf{z}_m}(\boldsymbol{x} \mid m, z_m)\, p_m(m) \prod_{n=1}^{M} p_{\mathbf{z}_n}(z_n). \end{aligned} \tag{1.59}$$

The marginal density on the model indicator p_m represents our prior beliefs about the relative probabilities of the models before observing data. Importantly the value of the model indicator variable m selects the relevant per model conditional density on the observed variables given latent variables $p_{\mathbf{x}|m,\mathbf{z}_m}$; this represents the assumption that conditioned on the model indicator assuming a particular model index m the observed variables are conditionally independent of the latent variables of all other models $\mathbf{x} \perp \{\mathbf{z}_n\}_{n \neq m} \mid m = m$. An extension to factor graphs, *gates* [145], can be used to represent such context-dependent conditional independencies. Figure 1.6 shows a gated factor graph of equation 1.59, with the gate indicated by the dashed box.

Given this computational set up, the task in model comparison is then to compute the relative probabilities of each of the models given observed data. These probabilities[6] are given by

$$p_{m|\mathbf{x}}(m \mid \boldsymbol{x}) = \frac{p_{\mathbf{x}|m}(\boldsymbol{x} \mid m)\, p_m(m)}{\sum_{n=1}^{M} (p_{\mathbf{x}|m}(\boldsymbol{x} \mid n)\, p_m(n))}, \tag{1.60}$$

which can be seen as a direct analogue to Bayes' theorem for the posterior density on unobserved random variables for a single model. The

6 As m is a discrete random variable the probability of the event of m taking a value in the singleton set $\{m\}$ given that $\mathbf{x} = \boldsymbol{x}$ is equal to the density $p_{m|\mathbf{x}}(m \mid \boldsymbol{x})$.

key quantities needed to evaluate the model posterior probabilities are the marginal densities $\mathrm{p}_{\mathbf{x}|\mathrm{m}}(\boldsymbol{x} \mid m)$ evaluated at the observed data. Computing these values requires marginalising out the latent variables from the per model joint densities on observed and latent variables

$$\mathrm{p}_{\mathbf{x}|\mathrm{m}}(\boldsymbol{x} \mid m) = \int_{Z_m} \mathrm{p}_{\mathbf{x}|\mathrm{m},\mathbf{z}_m}(\boldsymbol{x} \mid m, z)\mathrm{p}_{\mathbf{z}_m}(z)\,\mathrm{d}z. \tag{1.61}$$

This value is equal to the denominator in Bayes' theorem (1.56), this explaining the naming of this term as the *model evidence*.

As with evaluating conditional expectations of functions of the latent variables in a model given observations, evaluating the model evidence values requires integrating across the space of all latent variables. In most cases this integral will not have an analytic solution and the number of latent variables will be sufficiently large to make numerical quadrature methods impractical. The key computational challenge in being able to perform probabilistic model comparison with complex high dimensional models is therefore again being able to efficiently to compute integrals in high dimensional spaces.

1.3 SUMMARY

In this chapter we reviewed the probabilistic modelling framework that will form the basis for the methods we will introduce in the rest of the thesis. We began by introducing some of the underlying concepts from probability theory, in particular focusing on the manipulation random variables which are a key abstraction for explaining much of theory behind the methods in this thesis. We then motivated the computational challenges of performing inference in complex probabilistic models involving large numbers of unobserved variables. In particular we identified the key computational requirement as being able to evaluate integrals across high-dimensional spaces. In the next chapter we will introduce some of the computational methods which have been proposed to address the challenges of finding solutions to inference problems. These approximate inference approaches will form the basis for the novel methods proposed in the later chapters in this thesis.

2 APPROXIMATE INFERENCE

In the previous chapter we motivated the claim that the key computational challenge in performing inference in probabilistic models is being able to evaluate integrals with respect to probability measures defined on high-dimensional spaces. Although in restricted cases such as conjugate exponential family models, the integrals involved in inference can be solved analytically, to exploit the full flexibility of probabilistic modelling we need to be able to compute such integrals in more general cases. As discussed in the previous chapter, for models with even moderate numbers of latent variables, numerical quadrature approaches to evaluating integrals are computationally infeasible due to the exponential scaling of computation cost with dimension.

Truth is much too complicated to allow anything but approximations.
—John von Neumann

In this chapter we will review some of the key algorithms proposed for computing *approximate* solutions to inference problems. A unifying aspect to all of these methods is trading off some loss of the accuracy of the answers provided to inferential queries, for a potentially significant increase in computational tractability. The literature on *approximate inference* methods is vast and so necessarily this chapter will only form a partial review of the available methods. We will therefore concentrate on those approaches which are directly relevant to this thesis.

Approximate inference methods can be coarsely divided in to two groups: those in which a more tractable approximation to the target probability measure of interest is found by optimising the approximation to be 'close' in some sense to the target measure; methods in which the integrals with respect to the target measure are estimated by computing weighted sums over points sampled from a probability measure over the target state space. As with any such binary classification of such a broad and diverse topic there are however methods which combine aspects of both these approaches. In this chapter we will begin with sections reviewing the key sampling and optimisation based approaches to approximate inference, before concluding with a discussion of how these methods relate to the contributions made in this thesis.

2.1 SAMPLING APPROACHES

A key observation in the previous chapter was that inference at both the level of computing conditional expectations of latent variables in a model and in evaluating evidence terms to allow model comparison, will for most models of interest correspond to being able to integrate (potentially constant) functions against a probability density defined with respect to a base measure[1]. In particular we wish to be able to compute integrals of the form

$$\int_X f(\boldsymbol{x})\,\mathrm{d}P(\boldsymbol{x}) = \int_X f(\boldsymbol{x})\,p(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x}) \tag{2.1}$$

where p is the density of a target distribution P on a space X with respect to a base measure μ and f is a measurable function. For instance in the case of computing the *posterior mean* in a Bayesian inference problem with observed variables $\mathbf{y}$ and latent variables $\mathbf{x}$ where the posterior density $\mathrm{p}_{\mathbf{x}|\mathbf{y}}$ is defined with respect to the D-dimensional Lebesgue measure, we would have $p(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y})$ for an observed $\boldsymbol{y}$, $\mu(\boldsymbol{x}) = \lambda^D(\boldsymbol{x})$ and $f(\boldsymbol{x}) = \boldsymbol{x}$. Often we will only be able to evaluate p up to an unknown unnormalising constant i.e. $p(\boldsymbol{x}) = \frac{1}{Z}\tilde{p}(\boldsymbol{x})$ with we able to evaluate $\tilde{p}$ pointwise but Z intractable to compute. For example in a Bayesian inference setting $\tilde{p}(\boldsymbol{x})$ would be the joint density $\mathrm{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y})$ and Z the model evidence $\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$. When peforming inference in undirected graphical models, we would instead have that $\tilde{p}$ is the product of clique potentials and Z the corresponding normaliser.

The key idea of the methods we will discuss in this section is that we can estimate (2.1) by generating a set of random samples from a probability distribution defined on X and then computing a (potentially weighted) average of the value of the function f evaluated at these sample points. The most obvious approach is to sample independently from the probability distribution defined by the target density p. As we will see this is not necessarily feasible to do for the complex target densities defined on high dimensional spaces, however a host of related methods for generating and using random samples to approximate integrals with respect to target densities arising from complex probabilistic models have been developed.

1 There are models for which the corresponding probability measure is not absolutely continuous with respect to another measure and so cannot be represented by a density, however we will concentrate for now on the common case were a density exists.

2.1.1 Monte Carlo method

The framework that unifies all of the methods we will discuss in this section is the *Monte Carlo method* for integration [227]. We will briefly describe the key ideas and properties of Monte Carlo integration.

The eponym of the Monte Carlo method is a Monocan casino, favoured haunt of the uncle of Stanisław Ulam, one of the method's inventors.

Let $\mathbf{x}$ be a random (vector) variable distributed according to the target density i.e. $\mathrm{p}_{\mathbf{x}} = \frac{\mathrm{dP}_{\mathbf{x}}}{\mathrm{d}\mu} = p$. Given an arbitrary measurable function $f : X \to \mathbb{R}$ we define a random variable $\mathsf{f} = f(\mathbf{x})$. Our task is to compute expectations $\mathbb{E}[\mathsf{f}] = \bar{f}$ corresponding to the integral (2.1). We assume that $\mathbb{E}[\mathsf{f}]$ exists and both $\mathbb{E}[\mathsf{f}]$ and $\mathbb{V}[\mathsf{f}]$ are finite.

For now we assume we have a way of generating values of N random variables $\{\mathbf{x}_n\}_{n=1}^N$, each marginally distributed according to the target density i.e. $\mathrm{p}_{\mathbf{x}_n} = p \; \forall n \in \{1 \dots N\}$. We will initially not require any further properties on the joint distribution across all N variables. We define random variables $\{\mathsf{f}_n\}_{n=1}^N$ and $\hat{\mathsf{f}}_N$ by

$$\mathsf{f}_n = f(\mathbf{x}_n) \quad \forall n \in \{1 \dots N\} \quad \text{and} \quad \hat{\mathsf{f}}_N = \frac{1}{N} \sum_{n=1}^{N} \mathsf{f}_n. \tag{2.2}$$

Due to linearity of the expectation operator, we have that

$$\mathbb{E}\left[\hat{\mathsf{f}}_N\right] = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[\mathsf{f}_n] = \frac{1}{N} \sum_{n=1}^{N} \bar{f} = \bar{f} \tag{2.3}$$

and so that in expectation $\hat{\mathsf{f}}_N$ is equal to $\bar{f}$, i.e. realisations of $\hat{\mathsf{f}}_N$ are unbiased estimators of $\bar{f}$. Note that this result does not require any independence assumptions about the generated random variables. Now considering the variance of $\hat{\mathsf{f}}_N$ it can be shown that

$$\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \frac{\mathbb{V}[\mathsf{f}]}{N}\left(1 + \frac{2}{N} \sum_{n=1}^{N-1} \sum_{m=1}^{n-1} \frac{\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m]}{\mathbb{V}[\mathsf{f}]}\right). \tag{2.4}$$

If the generated random variables $\{\mathbf{x}_n\}$ and so $\{\mathsf{f}_n\}$ are independent, then $\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m] = 0 \; \forall m \neq n$. In this case (2.4) reduces to

$$\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \frac{\mathbb{V}[\mathsf{f}]}{N}, \tag{2.5}$$

i.e. the variance of the *Monte Carlo estimate* $\hat{\mathsf{f}}_N$ for $\bar{f}$ is inversely proportional to the number of generated random samples N. Importantly this scaling does not depend on the dimension of $\mathbf{x}$.

Figure 2.1.: Binary representation of linear congruential generator sequence $s_{n+1} = 37s_n + 61 \bmod 128$. Columns left to right represents successive integer states in sequence. From least significant (bottom) to most significant (top), the bits in each column have patterns repeating with periods 2, 4, 8, 16, 32, 64, 128.

Therefore if we can generate a set of independent random variables from the target density, we can estimate expectations that asymptotically tend to the true value as N increases, with a typical deviation from the true value (as measured by the standard deviation, i.e. the square root of variance) that is $O\left(N^{-\frac{1}{2}}\right)$. In comparison a fourth-order quadrature method such as *Simpson's rule* has an error that is $O\left(N^{-\frac{4}{D}}\right)$ for a grid of N points uniformly spaced across a D dimensional space. Asymptotically for $D > 8$, Monte Carlo integration will therefore give better convergence than Simpson's rule, and even for smaller dimensions the large constant factors in the error dependence can sometimes favour Monte Carlo.

Note that computing Monte Carlo estimates from independent random variables is not optimal in terms of minimising $\mathbb{V}\left[\hat{f}_N\right]$ for a given f; the covariance terms in (2.4) can be negative which can reduce the overall variance. A wide range of *variance reduction* methods have been proposed to exploit this and produce lower variance of Monte Carlo estimates for a given f [117]. Although these methods can be very important in pratice for achieving an estimator with a practical variance for a specific f of interest, we will generally concentrate on the case where we do not necessarily know f in advance and so cannot easily exploit these methods.

2.1.2 Pseudo-random number generation

The generation of random numbers is too important to be left to chance.
—Robert R. Coveyou

Virtually all statistical computations involving random numbers in practice make use of *pseudo-random number generators* (PRNGs). Rather than generating samples from truly random processes[2], PRNGs produce deterministic[3] sequences of integers in a fixed range that nonetheless maintain many of the properties of a random sequence.

2 In a true random process it is impossible to precisely predict the next value in the sequence given the previous values.

3 The sequences are determnistic in the sense that if the generator internal state is known all values in the sequence can be reconstructed exactly.

In particular through careful choice of the updates, sequences with a very long period (number of iterations before the sequence begins repeating), a uniform distribution across the numbers in the sequence range and low correlation between successive states can be constructed. A very simple example of a class of PRNGs is the *linear congruential generator* [121] which obeys the recurrent update

$$s_{n+1} = (as_n + c) \mod m \quad \text{with} \quad 0 < a < m,\ 0 \leq c < m, \tag{2.6}$$

with a, c and m integer parameters. If a, c and m are chosen appropriately, iterating the update (2.6) from an initial seed $0 \leq s_0 < m$, will produce a sequence of states which visits all the integers in $[0, m)$ before repeating. An example state sequence with $m = 128$ is shown in Figure 2.1. In practice, linear congruential generators produce sequences with poor statistical properties, particularly when used to generate random points in high dimensional spaces [134], hence most modern numerical computing libraries use more robust variants such as the *Mersenne-Twister* [138], which is used in all experiments in this thesis.

The raw output of a PRNG is an integer sequence, with typically the sequence elements uniformly distributed over all integers in a range $[0, 2^n)$ for some $n \in \mathbb{N}$. All real values are represented at a finite precision on computers, typically using a floating point representation [8] of *single* (24-bit mantissa) or *double* (53-bit mantissa) precision. Through an approriate linear transformation, the integer outputs of a PRNG can be converted to floating-point values uniformly distributed across a finite interval. PRNG implementations typically provide a primitive to generate floating-point values uniformly distributed on $[0, 1)$.

Given the ability to generate sequences of (effectively) independent samples from a uniform distribution $\mathcal{U}(0, 1)$, the question is then how to use these values to produce random samples from arbitary densities. This will be the subject of the following sub-sections.

2.1.3 Transform sampling

Samples from a large class of distributions can be generated by directly exploiting the transform of random variables relationships discussed in 1.1.4. In particular if $\mathbf{u}$ is D-dimensional vector of independent random variables with $\mathcal{U}(0, 1)$ marginal densities, then if $\boldsymbol{g} : [0, 1)^D \to X$ is a

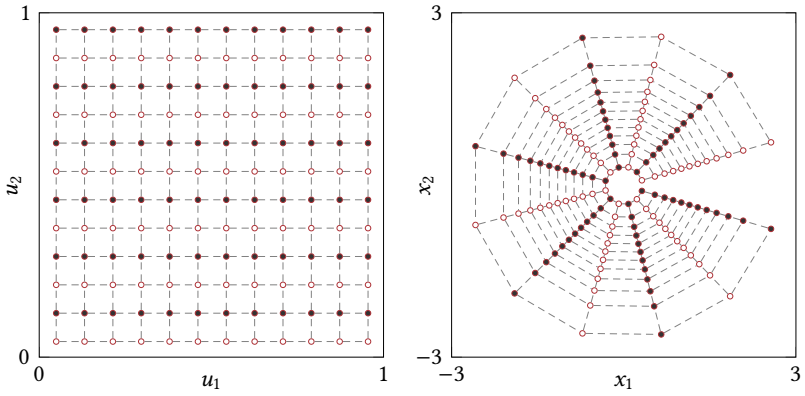


Figure 2.2.: Visualisation of Box–Muller transform. Left axis shows uniform grid on $U = [0,1]^2$ and right-axis shows grid points after mapping through $\boldsymbol{g}$ in transformed space $X = \mathbb{R}^2$.

bijective map to a vector space $X \subseteq \mathbb{R}^D$, then if we define $\mathbf{x} = \boldsymbol{g}(\mathbf{u})$ and $\boldsymbol{h} = \boldsymbol{g}^{-1}$, then by applying (1.22) we have that

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \left|\frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|. \tag{2.7}$$

For example for $D = 2$, $X = \mathbb{R}^2$ and a bijective map $\boldsymbol{g}$ defined by

$$\begin{aligned} \boldsymbol{g}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} &= \begin{bmatrix} \sqrt{-2\log u_1}\cos(2\pi u_1) \\ \sqrt{-2\log u_1}\sin(2\pi u_2) \end{bmatrix}, \\ \boldsymbol{h}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &= \begin{bmatrix} \exp\left(-\frac{1}{2}(x_1^2 + x_2^2)\right) \\ \frac{1}{2\pi}\arctan\left(\frac{x_1}{x_2}\right) \end{bmatrix}, \end{aligned} \tag{2.8}$$

then we have that the density of the transformed $\mathbf{x} = \boldsymbol{g}(\mathbf{u})$ is

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_1^2}{2}\right)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_2^2}{2}\right), \tag{2.9}$$

i.e. x_1 and x_2 are independent random variables with standard normal densities $\mathcal{N}(0,1)$. This is the *Box–Muller transform* [35], and allows generation of independent standard normal variables given a PRNG primitive for sampling from $\mathcal{U}(0,1)$. A visualisation of the transformation of space applied by the method is shown in Figure 2.2.

Due to the relatively high cost of the trigonometric function evaluations, more efficient alternatives to Box–Muller are usually used in practice to generate normal random variables such as a rejection sampling variant [134] (rejection sampling will be discussed in the next sub-section)

or the *Ziggurat algorithm* [135] (which also generalises to other symmetric univariate distributions).

A general method for sampling from univarite densities is to use an inverse *cumulative distribution function* (CDF) transform. For a probability density p on a scalar random variable, the corresponding CDF $r : \mathbb{R} \to [0, 1]$ is defined as

$$r(x) = \int_{-\infty}^{x} p(v)\,\mathrm{d}v \implies \frac{\partial r(x)}{\partial x} = p(x). \tag{2.10}$$

If u is a standard uniform random variable and $\mathsf{x} = r^{-1}(\mathsf{u})$ then

$$\mathsf{p}_{\mathsf{x}}(x) = \left|\frac{\partial r(x)}{\partial x}\right| = p(x). \tag{2.11}$$

To be able to use the inverse CDF transform method we need to be able to evaluate r^{-1}. For many univariate densities the CDF r itself does not have a closed form solution. For some densities such as the standard normal $\mathcal{N}(0, 1)$ even though the CDF does not have an analytic form in terms of elementary functions it is common for numerical computing libraries to provide numerical approximations to both r and r^{-1} which are accurate to within small multiples of machine precision. In densities where a standard library function for the CDF is not available, Chebyshev polynomial approximations to the density can be used to efficient compute an approximation to the CDF and an iterative solver used for the inversion [166].

Although the inverse CDF transform method gives a general recipe for sampling from univariate densities, it is not easy to generalise to multivariate densities and even for univariate densities, alternatives can be simpler to implement and in some cases more numerically stable.

2.1.4 Rejection sampling

An important class of generic sampling methods, particularly due their use as a building block in other algorithms, is rejection sampling [163]. Rejection sampling uses the observation that to sample from a probablity density $p : X \to [0, \infty]$ it is sufficient to uniformly sample from the volume under the graph of $(\boldsymbol{x}, p(\boldsymbol{x}))$.

The key requirement in rejection sampling is to identify a *proposal density* q which can be independently sampled from and that upper bounds

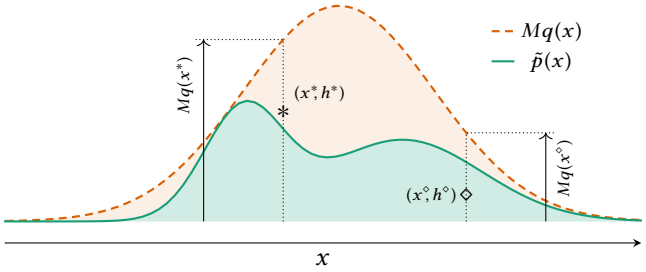


Figure 2.3.: Visualisation of rejection sampling. The green curve shows the (unnormalised) target density $\tilde{p}$, the green region underneath representing the area we wish to sample points uniformly from. The dashed orange curve shows the scaled proposal density Mq, with the orange (plus green) region representing the area we uniformly propose values from. Two example proposals are shown: $\diamond$ is under the target density and so accepted; $*$ is outside of the green region and so would be rejected.

Algorithm 1 Rejection sampling.

Input: $\tilde{p}$: unnormalised target density, q : normalised proposal density, M : constant such that $\tilde{p}(\boldsymbol{x}) \leq Mq(\boldsymbol{x})\ \forall \boldsymbol{x} \in X$.
Output: Independent sample from target density $p(\boldsymbol{x}) = \frac{1}{Z}\tilde{p}(\boldsymbol{x})$.

1: **repeat**
2: $\quad \boldsymbol{x} \sim q$
3: $\quad h \sim \mathcal{U}(0, Mq(\boldsymbol{x}))$
4: **until** $h \leq \tilde{p}(\boldsymbol{x})$
5: **return** $\boldsymbol{x}$

the potentially unnormalised target density $\tilde{p}$ across its full support X when multiplied by a known constant M, i.e. $\tilde{p}(x) \leq Mq(x)\ \forall x \in X$. The first requirement to be able to generate independent samples from q can be met for example by distributions we can sample from using a transform sampling method as described in the previous subsection, e.g. standard normal. The second requirement is general more challenging, both from the perspective of ensuring the target density is upper bounded everywhere but also because as we will see how efficient the method is very dependent on how tight the bound is.

Algorithm 1 describes the rejection sampling method to produce a single independent sample from the target density. A visualisation of how the algorithm works for a univariate target density is shown in Figure 2.3. The overall aim is to generate points uniformly distributed across the green area under the (unnormalised) target density curve. This is achieved by generating points uniformly under the dashed orange curve corresponding to the scaled proposal density and then ac-

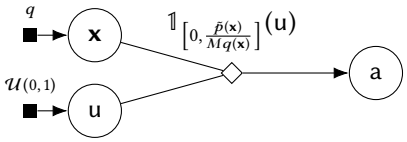


Figure 2.4.: Factor graph of rejection sampling process.

cepting only those which are below the green curve. To generate a point under the dashed orange curve we first generate an x from the proposal density and then generate an auxilliary 'height' variable by sampling uniformly from $[0, Mq(x)]$. If the sampled height is below the green curve we accept (as in the $\diamond$ example in Figure 2.3) else we reject the sample (as in the $*$ example in Figure 2.3).

Figure 2.4 shows the rejection sampling generative process as a directed factor graph, with $\mathbf{x}$ be a random variable representing the proposal, u the uniform auxilliary variable drawn to sample the 'height' and a a binary variable that indicates whether the proposal is accepted (a = 1) or not (a = 0). By marginalising out u we have that that

$$\mathrm{p}_{\mathbf{x},\mathrm{a}}(\boldsymbol{x}, a) = q(\boldsymbol{x})\left(\frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}\right)^{a}\left(1 - \frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}\right)^{1-a}, \tag{2.12}$$

and further marginalising over the proposal $\mathbf{x}$

$$\mathrm{p}_{\mathrm{a}}(a) = \left(\frac{Z}{M}\right)^{a}\left(1 - \frac{Z}{M}\right)^{1-a}. \tag{2.13}$$

Conditioning on the proposal being accepted we therefore have that

$$\mathrm{p}_{\mathbf{x}|\mathrm{a}}(\boldsymbol{x} \mid 1) = \frac{q(\boldsymbol{x})\frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}}{\frac{Z}{M}} = \frac{\tilde{p}(\boldsymbol{x})}{Z} = p(\boldsymbol{x}). \tag{2.14}$$

Therefore the accepted proposals are distributed according to the target density as required. Further from (2.13) we have that the $\mathrm{p}_{\mathrm{a}}(1) = \frac{Z}{M}$. This suggests we can use the accept rate to estimate Z but also hints at the difficulty in finding a M which guarantees the upper bound requirement as for $\frac{Z}{M}$ to be a valid probability $M \geq Z$ i.e. M needs to be an upper bound on the unknown normalising constant Z. This relationship also suggests it is desirable to set M as small as possible to maximise the acceptance rate; for a fixed proposal density q this will involve setting M to a value such that $Mq(\boldsymbol{x}) = \tilde{p}(\boldsymbol{x})$ for at least one $\boldsymbol{x}$ (as for example in Figure 2.3).

Although rejection sampling can be an efficient method of sampling from univariate target densities (particularly in the case of log-concave densities where an adaptive variant is available [82]), it generally scales very poorly with the dimensionality of the target density. This is as the ratio of the volume under the target density to the volume under the scaled proposal (in terms of Figure 2.3 the ratio of the green area to the green plus orange regions), and so the probability of accepting a proposal, will tend become exponentially smaller as the dimensionality increases. This is the so-called *curse of dimensionality*. Therefore although rejection sampling can be a useful subroutine for generating random variables from low-dimensional densities, in general it is not a viable option for generating samples directly for high-dimensional Monte Carlo integration.

2.1.5 Importance sampling

So far we have considered methods for generating samples directly from some target density. Although samples can be of value in themselves for giving a representative set of plausible values from the target density (e.g. for visualisation purposes), usually the end goal is to estimate integrals of the form in (2.1).

Importance sampling [108] is a Monte Carlo method which allows arbitrary integrals to be estimated. If q is density of a probability measure which is absolutely continuous to the probability measure defined by the target density p (which requires that $p(\boldsymbol{x}) > 0 \Rightarrow q(\boldsymbol{x}) > 0$), then importance sampling is based on the identity

$$\bar{f} = \int_X f(\boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x}) = \frac{\int_X f(\boldsymbol{x})\, \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x})}{\int_X \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\, q(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x})}. \tag{2.15}$$

Each of the numerator and denominator in (2.15) take the form of an expectation of a measurable function of a random variable $\mathbf{x}$ with probability density $\mathrm{p}_{\mathbf{x}} = q$. Further the denominator is exactly equal to $Z = \int_X \tilde{p}(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x})$. We therefore have that

$$Z\bar{f} = \mathbb{E}[w(\mathbf{x}) f(\mathbf{x})] \quad \text{and} \quad Z = \mathbb{E}[w(\mathbf{x})] \quad \text{with} \quad w(\boldsymbol{x}) = \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}. \tag{2.16}$$

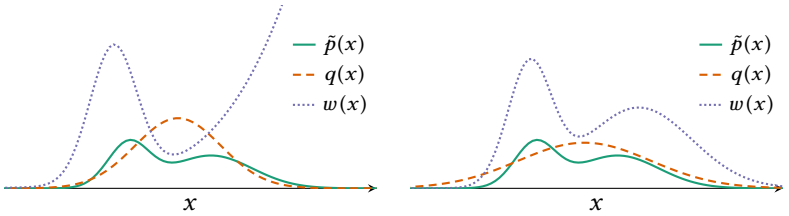


Figure 2.5.: Visualisation of importance sampling. On both axes the green curve shows the unnormalised target density $\tilde{p}$, the dashed orange curve the density q values are sampled from and the dotted violet curve the importance weighting function $w(x) = \frac{\tilde{p}(x)}{q(x)}$ to estimate expectations with respect to the target density using samples from q. In the left axis the q chosen is undispersed compared to $\tilde{p}$ leading to very large w values in the right tail. In constrast in the right axis, the broader q leads to less extreme variation in w.

If we can generate random variables $\{\mathbf{x}_n\}_{n=1}^N$ each with marginal density q we can therefore form Monte Carlo estimates of both the numerator and denominator. We define $\hat{Z}_N$ and $\hat{g}_N$ as

$$\hat{Z}_N = \frac{1}{N}\sum_{n=1}^{N} w(\mathbf{x}_n) \quad \text{and} \quad \hat{g}_N = \frac{1}{\hat{Z}}\sum_{n=1}^{N} w(\mathbf{x}_n) f(\mathbf{x}_n). \tag{2.17}$$

By the same argument as Section 2.1.1, $\mathbb{E}\left[\hat{Z}_N\right] = Z$ and $\mathbb{E}[\hat{g}_N] = Z\bar{f}$. We can therefore use importance sampling to form an unbiased estimate of the unknown normalising constant Z.

If we define $\hat{f}_N = \hat{g}_N / \hat{Z}_N$, then this is a biased[4] estimator for $\bar{f}$ as in general the expectation of the ratio of two random variables is not equal to the ratios of their expectations. However if both the numerator and denominator have finite variance, i.e. $\mathbb{V}\left[\hat{Z}_N\right] < \infty$ and $\mathbb{V}[\hat{g}_N] < \infty$, then $\hat{f}_N$ is a *consistent* estimator for $\bar{f}$ i.e. $\lim_{N\to\infty} \mathbb{E}\left[\hat{f}_N\right] = \bar{f}$.

The $w(\mathrm{x}_n)$ values are typically termed the *importance weights*. If a few of the weights are very large, the weighted sums in (2.17) will be dominated by those few values, reducing the effective number of samples in the Monte Carlo estimates. This can particularly be a problem if the are regions of X with low probability under q where $p(\boldsymbol{x}) \gg q(\boldsymbol{x})$. As sampling points in these regions will be a rare event, a large number of samples may be needed to diagnose the issue adding further difficulty.

4 In cases where the normalising constant Z is known, we can instead use $w(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$ in which case the ratio estimator is not required and an unbiased estimates can be calculated. As the problems we are interested in will generally have unknown Z we do not consider this case further

A general recommendation is to use densities q with tails as least as heavy of those of p, and in general the closer the match between q and p the better [130, 167]. Figure 2.5 shows a visualisation of the effect of the choice of q on the importance weights.

A heuristic that can be used to help assess the quality of importance sampling estimates is what is sometimes termed the *effective sample size* [116, 167]. It approximately quantifies how many independent samples from the target p would be required to get a Monte Carlo estimate with a similar variance to that achieved using an importance sampling estimator with weights $\{w(\mathbf{x}_n)\}_{n=1}^N$ given iid $\{\mathbf{x}_n\}_{n=1}^N$ generated from q. It can be calculated as

$$\mathsf{N}_{\text{eff}} = \left(\sum_{n=1}^{N} \bar{\mathsf{w}}_n^2\right)^{-1} \quad \text{where} \quad \bar{\mathsf{w}}_n = \frac{w(\mathbf{x}_n)}{\sum_{m=1}^{N} w(\mathbf{x}_m)}, \tag{2.18}$$

i.e. as the reciprocal of the sum of squares of the normalised importance weights. If $\mathsf{N}_{\text{eff}} \ll N$ this can suggest an issue with the choice of sampling density q. The diagnostic is not fool proof however as it is based on a finite sample size, and it may be that rare extreme importance weights due to e.g. a mode of the target p in the tails of q, are not encountered in a particular run giving a misleadingly high N_{eff}.

When previously discussing rejection sampling, we introduced an auxiliary binary accept indicator variable, a, associated with each proposed sample $\mathbf{x}$ (see Figure 2.4). If we generate N independent proposal – indicator pairs $\{\mathbf{x}_n, \mathsf{a}_n\}_{n=1}^N$ then the number of accepted proposals is $\mathsf{N}_{\text{acc}} = \sum_{n=1}^N \mathsf{a}_n$. Conditioned on N_{acc} being a value more than one, the generated rejection sampling variables $\{\mathbf{x}_n, \mathsf{a}_n\}_{n=1}^N$ can be used to form an *unbiased* Monte Carlo estimate of $\bar{f}$ using the estimator

$$\hat{\mathsf{f}}_N^{\text{RS}} = \frac{\sum_{n=1}^N \mathsf{a}_n f(\mathbf{x}_n)}{\sum_{m=1}^N \mathsf{a}_m}, \tag{2.19}$$

which just correponds to computing the empirical mean of the accepted proposals i.e. the standard Monte Carlo estimator. In comparison importance sampling forms a biased but consistent estimator for $\bar{f}$ from N samples $\{\mathbf{x}_n\}_{n=1}^N$ from a density q using the estimator

$$\hat{\mathsf{f}}_N^{\text{IS}} = \frac{\sum_{n=1}^N w(\mathbf{x}_n) f(\mathbf{x}_n)}{\sum_{m=1}^N w(\mathbf{x}_m)}. \tag{2.20}$$

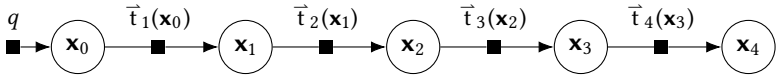


Figure 2.6.: Markov chain factor graph. The initial state $\mathbf{x}_0$ is sampled from a density q and each subsequent state $\mathbf{x}_n$ is then generated from a transition density $\vec{t}_n$ conditioned on the previous state $\mathbf{x}_{n-1}$.

From this perspective the accept indicators a_n in rejection sampling can be seen to act like binary importance weights, in contrast importance sampling using 'soft' weights which mean all sampled $\mathbf{x}_n$ make a contribution to the estimator (assuming $w(\boldsymbol{x}) \neq 0 \; \forall \boldsymbol{x} \in X$). However this correspondence is only loose. The rejection sampling estimator $\hat{f}_N^{\text{RS}}$ is unbiased unlike $\hat{f}_N^{\text{IS}}$, but this unbiasedness relies on conditioning on a non-zero value for N_{acc} (i.e. the number of accepted samples to generate) and continuing to propose points until this condition is met. In contrast importance sampling generates a fixed number of samples from q and does not use any auxiliary random variables.

Unlike rejection sampling, there is no need in importance sampling for q to upper-bound the target density (when multiplied by a constant). This allows more freedom in the choice of q however it is still important to choose q to be as close as possible to the target while remaining tractable to generate samples from. In general for target densities defined on high-dimensional spaces, it can be difficult to find an appropriate q such that the variation in importance weights is not too extreme [130]. As we will see later however the importance sampling framework can be combined with other methods we will discuss in the following sections to allow it to be scaled to high dimensional problems.

2.1.6 Markov chain Monte Carlo

The sampling methods we have considered so far have involved using independent random variables to form Monte Carlo estimates. However when introducing the Monte Carlo method we saw that is was not necessary for the random variables used in a Monte Carlo estimator to be independent. While it can be impractically computationally expensive to generate independent samples from complex high-dimensional target distributions, simulating a stochastic process which converges in distribution to the target and produces a sequence of *dependent* random variables is often a more tractable task. This is the idea exploited by *Markov chain Monte Carlo* (MCMC) methods.

A *Markov chain* is an ordered sequence of random variables $\{\mathbf{x}_n\}_{n=0}^N$ which have the *Markov property* — for all $n \in \{1 \dots N\}$, $\mathbf{x}_n$ is conditionally independent of $\{\mathbf{x}_n\}_{m<n-1}$ given $\mathbf{x}_{n-1}$. This conditional independence structure is visualised as a factor graph in Figure 2.6.

For a Markov chain defined on a general measurable state space $(X, \mathcal{F})$, the probability distribution of a state $\mathbf{x}_n$ given the state $\mathbf{x}_{n-1}$ is specified for each $n \in \{1 \dots N\}$ by a *transition operator*, $\vec{\mathsf{T}}_n : \mathcal{F} \times X \to [0, 1]$. In particular the transition operators define a series of regular conditional distributions for each $n \in \{1 \dots N\}$

$$\mathsf{P}_{\mathbf{x}_n|\mathbf{x}_{n-1}}(A \mid \boldsymbol{x}) = \vec{\mathsf{T}}_n(A \mid \boldsymbol{x}) \quad \forall A \in \mathcal{F},\ \boldsymbol{x} \in X. \tag{2.21}$$

We will generally assume that the chain is *homogeneous*, i.e. that the same transition operator is used for all steps $\vec{\mathsf{T}}_n = \vec{\mathsf{T}}\ \forall n \in \{1 \dots N\}$.

The key requirement of a transition operator for MCMC is that the target distribution P is *invariant* under the transition, that is it satisfies

$$P(A) = \int_X \vec{\mathsf{T}}(A \mid \boldsymbol{x})\, \mathrm{d}P(\boldsymbol{x}) \quad \forall A \in \mathcal{F}, \tag{2.22}$$

The invariance property means that if a chain state $\mathbf{x}_n$ is distributed according to the target P, all subsequent chain states $\mathbf{x}_{n+1}$, $\mathbf{x}_{n+2} \dots$ will also be marginally distributed according to the target. Therefore given a single random sample $\mathbf{x}_0$ from the target distribution, a series of dependent states marginally distributed according to the target could be generated and used to form Monte Carlo estimates of expectations.

Being able to generate even one exact sample from a complex high-dimensional target distribution is in general infeasible. Importantly however, subject to further conditions on the transition operator, the marginal distribution on the chain state $\mathsf{P}_{\mathbf{x}_n}$ of a Markov chain with a transition operator which leaves the target distribution invariant will converge to the target distribution irrespective of the distribution of the intial chain state. The requirements correspond to ensuring the target distribution is the *unique* invariant distribution of the chain.

To have a unique invariant distribution, a chain must be *irreducible* and *aperiodic* [221]. For a chain on a measurable space $(X, \mathcal{F})$, irreducibility is defined with respect to a measure ν, which could but does not necessarily need to be the target distribution P. A chain is ν-irreducible if

starting at any point in X there is a non-zero probability of moving to any set with positive ν-measure in a finite number of steps, i.e.

$$\forall \boldsymbol{x} \in X,\ A \in \mathcal{F} : \nu(A) > 0\ \ \exists m \in \mathbb{Z}^+ : \mathrm{P}_{\mathbf{x}_m|\mathbf{x}_0}(A \mid \boldsymbol{x}) > 0. \tag{2.23}$$

A chain with invariant distribution P is periodic if disjoint regions of the state space are visited cyclically, i.e. there exists an integer $r > 1$ and an ordered set of r disjoint P-positive subsets of X, $\{A_i\}_{i=1}^r$ such that $\vec{\mathrm{T}}(A_j \mid \boldsymbol{x}) = 1\ \forall \boldsymbol{x} \in A_i,\ i \in \{1 \ldots r\},\ j = (i+1) \mod r$.

If we can construct a ν-irreducible and aperiodic Markov chain $\{\mathbf{x}_n\}_{n=0}^N$ which has the target distribution P as its invariant distribution, then the *MCMC estimator* $\hat{\mathrm{f}}_N = \frac{1}{N}\sum_{n=1}^N f(\mathbf{x}_n)$ converges almost surely as $N \to \infty$ to $\bar{f} = \int_X f \mathrm{d}P$ for all starting states except for a ν-null set[5] [143]. This convergence of *time-averages* (i.e. over states at different steps of the Markov chain) to *space-averages* (i.e. with respect to the stationary distribution across the state space), is termed *ergodicity* and is a consequence of the *Birkhoff–Khinchin ergodic theorem* [26].

Although irreducibility and aperiodicity of a Markov chain which leaves the target distribution invariant are sufficient for convergence of MCMC estimators, this does not tell us anything about the rate of that convergence and so how to quantify the error introduced by computing estimates with a Markov chain simulated for only a finite number of steps. Stronger notions of ergodicity can be used to help quantify convergence; we will concentrate on *geometric ergodicity* here. We first define a notion of distance between two measures μ and ν on a measurable space $(X, \mathcal{F})$, the *total variation distance*, as

$$\|\mu - \nu\|_{\mathrm{TV}} = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|. \tag{2.24}$$

For a ν-irreducible and aperiodic chain with invariant distribution P our earlier vague statement that the distribution on the chain state converges to P can now be restated more precisely as that for ν-almost all initial states $\mathbf{x}_0 = \boldsymbol{x}$, $\lim_{n\to\infty} \|\mathrm{P}_{\mathbf{x}_n|\mathbf{x}_0}(\cdot \mid \boldsymbol{x}) - P\|_{\mathrm{TV}} = 0$. Geometric ergodicity makes a stronger statement that the convergence in total variation distance conditioned is geometric in n, i.e. that

$$\|\mathrm{P}_{\mathbf{x}_n|\mathbf{x}_0}(\cdot \mid \boldsymbol{x}) - P\|_{\mathrm{TV}} \leq M(\boldsymbol{x}) r^n \tag{2.25}$$

5 The 'except for a ν-null set' caveat can be removed by requiring the stronger property of *Harris recurrence* [94].

for a positive measurable function M which depends on the initial chain state $\boldsymbol{x}$ and rate constant $r \in [0, 1)$. For chains which are geometrically ergodic, we can derive an expression for the *asymptoptic variance* of an MCMC estimator $\hat{\mathsf{f}}_N$ related to the variance of a simple Monte Carlo estimator previously considered in Section 2.1.1.

A stochastic process is stationary if the joint distribution of the states at any set of time points does not change if all those times are shifted by a constant.

As in Section 2.1.1 we define $\mathsf{f}_n = f(\mathbf{x}_n)$ and $\hat{\mathsf{f}}_N = \frac{1}{N}\sum_{n=1}^{N} \mathsf{f}_n$, with here the $\{\mathbf{x}_n\}_{n=1}^{N}$ the states of a Markov chain. For a homogeneous Markov chain with a unique invariant distribution P which is *stationary*, the marginal density on the states $\mathsf{P}_{\mathbf{x}_n}$ is equal to P for all n and we can use the expression for the variance of a general Monte Carlo estimator (which did not assume independence of the random variables) stated earlier in (2.4). Further the stationarity of the chain means that the covariance $\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m]$ depends only on the difference $n - m$, and so the variance of the estimator simplifies to

$$\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \frac{\mathbb{V}[\mathsf{f}]}{N}\left(1 + 2\sum_{n=1}^{N-1}\left(\frac{N-n}{N}\frac{\mathbb{C}[\mathsf{f}_0, \mathsf{f}_n]}{\mathbb{V}[\mathsf{f}]}\right)\right). \tag{2.26}$$

If we multiply both sides of (2.26) by N and define $\rho_n = \frac{\mathbb{C}[\mathsf{f}_0, \mathsf{f}_n]}{\mathbb{V}[\mathsf{f}]}$ (the lag n autocorrelations of f), under the assumption that $\sum_{n=1}^{\infty} |\rho_n| < \infty$ in the limit of $N \to \infty$ we have that

$$\lim_{N\to\infty}\left(N\,\mathbb{V}\left[\hat{\mathsf{f}}_N\right]\right) = \mathbb{V}[\mathsf{f}]\left(1 + 2\sum_{n=1}^{\infty}\rho_n\right). \tag{2.27}$$

Now considering a chain which is geometrically ergodic from its initial state, if $\mathbb{E}\left[|\mathsf{f}|^{2+\delta}\right]$ is finite for some $\delta > 0$ then it can be shown [41, 78, 194] that (2.27) is also the asymptoptic variance for a *MCMC* estimator calculated using the chain states.

This motivates a definition of the *effective sample size* (ESS)[6] for an MCMC estimator $\hat{\mathsf{f}}_N$ computed using a geometrically ergodic chain as

$$N_{\text{eff}} = \frac{N}{1 + 2\sum_{n=1}^{\infty}\rho_n}. \tag{2.28}$$

The ESS quantifies the number of independent samples that would be required in a Monte Carlo estimator to give an equivalent variance to the MCMC estimator $\hat{\mathsf{f}}_N$ *in the asymptoptic limit* $N \to \infty$. In practice we cannot evaluate the exact autocorrelations and so we can only compute

6 Note this is unrelated to the previous definition for importance sampling.

an estimated ESS, $\hat{N}_{\text{eff}}$, from one or more simulated chains with the estimation method needing to be carefully chosen to ensure reasonable values [220]. Although the assumption of geometric ergodicity can often be hard to verify in practice and ESS estimates can give misleading results in chains far from convergence, when used appropriately estimated ESSs can still be a useful heuristic for evaluating and comparing the effiency of Markov chain estimators and are often available as a standard diagnostic in MCMC software packages [39, 177, 203].

So far we have not discussed how to actually construct a transition operator giving a chain with the required invariant distribution. As a notational convenience we will consider the transition operator as being specified by a *transition density* $\vec{\mathrm{t}} : X \times X \to [0, \infty)$ which is defined with respect to a base measure μ (which we will later assume to be the same as that which the target density we wish to integrate against is defined with respect to, hence the reuse of notation). The transition operator is then

$$\vec{\mathrm{T}}(A \mid \boldsymbol{x}) = \int_A \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}') \quad \forall A \in \mathcal{F}, \ \boldsymbol{x} \in X. \tag{2.29}$$

In practice the probability measure defined by a transition operator will often have a singular component, for example corresponding to a non-zero probability of the chain remaining in the current state. In this case $\vec{\mathrm{T}}$ is not absolutely continuous with respect to μ and the transition density is not strictly defined. As we did in the previous chapter however we will informally use Dirac deltas to represent a 'density' of singular measures, and so still consider a transition density as existing. The requirement that the transtion operator leaves the target distribution invariant, can then be expressed in terms of the target density p and transition density $\vec{\mathrm{t}}$ as

$$p(\boldsymbol{x}') = \int_X \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x}) \, p(\boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}) \quad \forall \boldsymbol{x}' \in X. \tag{2.30}$$

Finding a transition density which leaves the target density invariant by satisfying (2.30) seems difficult in general as it involves evaluating an integral against the target density - precisely the computational task which we have been forced to seek approximate solutions to. We can make progress by considering the joint density of a pair of successive

states for a chain with invariant distribution P that has converged to stationarity. Then we have that

$$\mathrm{p}_{\mathbf{x}_n,\mathbf{x}_{n-1}}(\boldsymbol{x}', \boldsymbol{x}) = \mathrm{p}_{\mathbf{x}_n|\mathbf{x}_{n-1}}(\boldsymbol{x}' \,|\, \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}_{n-1}}(\boldsymbol{x}) = \vec{\mathrm{t}}\,(\boldsymbol{x}' \,|\, \boldsymbol{x})\, p(\boldsymbol{x}). \tag{2.31}$$

We can also consider factorising this joint density into the product of the marginal density of the current state $\mathrm{p}_{\mathbf{x}_n}$ and the conditional density of the previous state given the current state $\mathrm{p}_{\mathbf{x}_{n-1}|\mathbf{x}_n}$. Due to stationarity $\mathrm{p}_{\mathbf{x}_n}$ is also equal to p and so we have that $\mathrm{p}_{\mathbf{x}_{n-1}|\mathbf{x}_n}$ must be the density of a transition operator which also leaves P invariant, corresponding to a time reversed version of the original (stationary) Markov chain[7]. If we therefore denote $\overleftarrow{\mathrm{t}} = \mathrm{p}_{\mathbf{x}_{n-1}|\mathbf{x}_n}$ (and which we will term the *backward transition density* in contrast to $\vec{\mathrm{t}}$ which in this context we will qualify as the *forward transition density*), we have that

$$\vec{\mathrm{t}}\,(\boldsymbol{x}' \,|\, \boldsymbol{x})\, p(\boldsymbol{x}) = \overleftarrow{\mathrm{t}}\,(\boldsymbol{x} \,|\, \boldsymbol{x}')\, p(\boldsymbol{x}') \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{2.32}$$

Integrating both sides with respect to $\boldsymbol{x}$, we have that $\forall \boldsymbol{x}' \in X$

$$\begin{aligned} \int_X \vec{\mathrm{t}}\,(\boldsymbol{x}' \,|\, \boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x}) &= \int_X \overleftarrow{\mathrm{t}}\,(\boldsymbol{x} \,|\, \boldsymbol{x}')\, p(\boldsymbol{x}')\, \mathrm{d}\mu(\boldsymbol{x}) \\ &= \int_X \overleftarrow{\mathrm{t}}\,(\boldsymbol{x} \,|\, \boldsymbol{x}')\, \mathrm{d}\mu(\boldsymbol{x})\, p(\boldsymbol{x}') = p(\boldsymbol{x}'), \end{aligned} \tag{2.33}$$

and so that (2.30) is satisfied, with the last inequality arising due to $\overleftarrow{\mathrm{t}}$ being a normalised density on its first argument. Therefore if we can find a pair of transition densities, $\vec{\mathrm{t}}$ and $\overleftarrow{\mathrm{t}}$, satisfying (2.32), then the transition operator specified by $\vec{\mathrm{t}}$ will leave the target distribution P invariant (and by an equivalent argument so will the transition operator specified by $\overleftarrow{\mathrm{t}}$). We can further simplify (2.32) by requiring that $\vec{\mathrm{t}} = \overleftarrow{\mathrm{t}} = \mathrm{t}$, i.e. that both forward and backward transition densities (and corresponding operators) take the same form and so that the chain at stationarity is *reversible*, in which case have that

$$\mathrm{t}(\boldsymbol{x}' \,|\, \boldsymbol{x})\, p(\boldsymbol{x}) = \mathrm{t}(\boldsymbol{x} \,|\, \boldsymbol{x}')\, p(\boldsymbol{x}') \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{2.34}$$

This is often termed the *detailed balance* condition. Importantly both the detailed balance (2.34) and *generalised balance* (2.32) conditions can

7 The time reversal of a Markov chain is always itself a a Markov chain irrespective of stationarity (as the defininining conditional independence structure is symmetric with respect to the direction of time), however the reverse of a homogeneous Markov chain which is not stationary will not in general itself be homogeneous.

also be written in terms of the unormalised density $\tilde{p}$ by multiplying both sides by Z, and so can be checked even when Z is unknown.

The restriction to reversible transition operators in detailed balance, while sufficient for (2.30) to hold is not necessary. Markov chains which satisfy the generalised balance condition but not detailed balance are termed *non-reversible*, and there are theoretical results suggesting that non-reversible Markov chains can sometimes achieve significantly improved convergence compared to related reversible chains [54, 105, 161] (a criticism made of some of these results is that choice of 'reference' reversible chain to compare to can be somewhat arbitrary [147]).

While there are several general purpose frameworks for specifying reversible transition operators which leave a target distribution invariant, developing methods for constructing irreversible transition operators with a desired invariant distribution has proven more challenging. The approaches proposed to date are generally limited in practice to special cases such as finite state spaces [214, 215, 226] or chains with tractable invariant distributions such as multivariate normal [25].

Nonetheless non-reversible Markov chains are still commonly used in MCMC applications. Given a set of transition operators which each individually leave a target distribution invariant, the sequential composition of the transition operators will necessarily by induction also leave the target distribution invariant. Even if the individual transition operators are all reversible, the overall sequential composition will generally not be (instead having an adjoint 'backward' operator corresponding to applying the individual transitions in the reversed order). Sequentially combining several reversible transition operators is common in MCMC implementations, though this is more often the result of each individual operator not meaning the requirements for ergodicity in isolation and so needing to be combined with other operators, rather than due to a specific aim of introducing irreversibility.

Having now introduced the key theoretical concepts underlying MCMC methods, we will now move on to discussing details of their implementation. In the following sub-sections we will review three widely applicable frameworks for constructing reversible transition operators which leave a target distribution invariant: the *Metropolis–Hastings* algorithm, *Gibbs sampling* and *slice sampling*. These three methods will form the basis for much of the work introduced in this thesis.

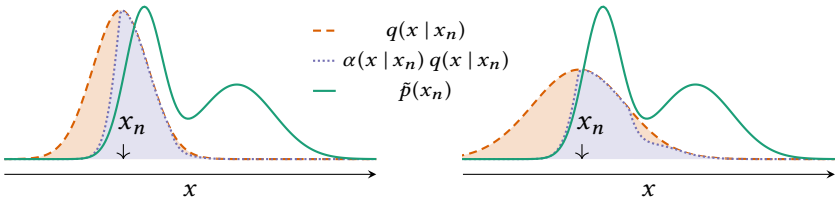


Figure 2.7.: Visualisation of Metropolis–Hastings algorithm in a univariate target density. The green curves shows the unnormalised target density. The arrows indicate the current chain state. The orange curves show the density of proposed moves from this state, with the left axis using a narrower proposal than the right. The violet curves show the proposal density scaled by the acceptance probability of the proposed move, this reducing the probability of transitions to states with lower density than the current state. The orange region between the violet and orange curves represents the probability mass reallocated to rejections by the downscaling by the acceptance function. The broader proposal in the right axis has an increased probability of making a move to the other mode in the target density but at a cost of an increased rejection probability.

Algorithm 2 Metropolis–Hastings transition.

Input: $\boldsymbol{x}_n$: current chain state, $\tilde{p}$: unnormalised target density,
q : normalised proposal density which we can sample from.
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

1: $\boldsymbol{x}^* \sim q(\boldsymbol{x}_n)$ ▷ Generate proposed new state
2: $u \sim \mathcal{U}(0,1)$
3: **if** $u < \frac{\tilde{p}(\boldsymbol{x}^*)\, q(\boldsymbol{x}_n \mid \boldsymbol{x}^*)}{\tilde{p}(\boldsymbol{x})\, q(\boldsymbol{x}^* \mid \boldsymbol{x}_n)}$ **then**
4: $\quad \boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}^*$ ▷ Proposed move accepted
5: **else**
6: $\quad \boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$ ▷ Propsed move rejected
7: **return** $\boldsymbol{x}_{n+1}$

2.1.6.1 *Metropolis–Hastings*

Although the algorithm has come to be commonly known by Edward Metropolis' name as first author on the 1953 paper [142], it is believed that Arianna and Marshall Rosenbluth, two of the other co-authors, were the main contributors to the development of the algorithm [92].

The seminal *Metropolis–Hastings* algorithm provides a general framework for constructing Markov chains with a desired invariant distribution and is ubitiqous in MCMC methodology. The original Rosenbluth–Teller–Metropolis variant of the algorithm [142] dates to the very beginnings of the Monte Carlo method, having being first implemented on Los Alamos' MANIAC[8] one of the earliest programmable computers. The method was generalised in a key paper by Hastings [99], and the optimality among several competing alternatives of the form now used demonstrated by Peskun [173]. An extension to Markov chains on trans-dimensional spaces was proposed by Green [91].

8 *Mathematical Analyzer, Numerical Integrator and Computer.*

An outline of the method is given in Algorithm 2 and a visualisation of terms involved in the transition operator in a univariate target density shown in Figure 2.7. The key idea is to propose updates to the state using an arbitrary transition operator, and then correct for this transition operator not necessarily leaving the target distribution invariant by stochastically accepting or rejecting the proposal. If a proposal is rejected the chain remains at the current state, otherwise the chain state takes on the proposed value.

The transition density corresponding to Algorithm 2 is

$$\begin{aligned} \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x}) = \alpha(\boldsymbol{x}' \mid \boldsymbol{x})\, q(\boldsymbol{x}' \mid \boldsymbol{x}) + \\ \left(1 - \int_X \alpha(\boldsymbol{x}^* \mid \boldsymbol{x})\, q(\boldsymbol{x}^* \mid \boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x}^*)\right) \delta(\boldsymbol{x}' - \boldsymbol{x}), \end{aligned} \tag{2.35}$$

with the *acceptance probability* $\alpha : X \times X \to [0, 1]$ defined as

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1,\ \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}\right\} = \min\left\{1,\ \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, \tilde{p}(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, \tilde{p}(\boldsymbol{x})}\right\}, \tag{2.36}$$

and $q : X \times X \to [0, \infty)$ the density of the proposal transition operator (from herein *proposal density*). This transition density corresponds to a reversible transition operator which leaves the target distribution P invariant as we will now show.

For the purposes of verifying the detailed balance condition (2.34), the density of *self-transitions*, i.e. a transition to the same state, can be ignored as (2.34) is trivially satisfied for $\boldsymbol{x}' = \boldsymbol{x}$. Considering therefore the cases $\boldsymbol{x} \neq \boldsymbol{x}'$ where the Dirac delta term representing the singular measure corresponding to rejected proposals can be neglected, we therefore have $\forall \boldsymbol{x} \in X,\ \boldsymbol{x}' \in X : \boldsymbol{x} \neq \boldsymbol{x}$

$$\begin{aligned} \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) &= \min\left\{1,\ \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}\right\} q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) && (2.37) \\ &= \min\{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}),\ q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')\} && (2.38) \\ &= \min\left\{\frac{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')},\ 1\right\} q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}') && (2.39) \\ &= \mathrm{t}(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}'). && (2.40) \end{aligned}$$

Therefore the detailed balance condition is satisfied, and the Metropolis–Hastings transition operator leaves the target distribution P invariant.

The original Rosenbluth–Teller–Metropolis algorithm used a symmetric proposal density $q(\boldsymbol{x}' \mid \boldsymbol{x}) = q(\boldsymbol{x} \mid \boldsymbol{x}') \; \forall \boldsymbol{x} \in X, \boldsymbol{x}' \in X$ (with the extension to the non-symmetric case being due to Hastings), in which case the acceptance probability definition simplifies to

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})}\right\} = \min\left\{1, \frac{\tilde{p}(\boldsymbol{x}')}{\tilde{p}(\boldsymbol{x})}\right\}. \tag{2.41}$$

Note that importantly in both (2.36) and (2.41) the target density only appears as a ratio and so the density need only be known up to a proportionality constant.

An important special case for chains on a Euclidean state space with a Borel σ-algebra i.e. $(X, \mathcal{F}) = (\mathbb{R}^D, \mathscr{B}(\mathbb{R}^D))$, is when the proposal transition operator is deterministic and corresponds to a differentiable involution of the current state. Let $\boldsymbol{\phi} : X \to X$ be an involution, i.e. $\boldsymbol{\phi} \circ \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x} \; \forall X$ with Jacobian determinant $D_{\boldsymbol{\phi}}(\boldsymbol{x}) = \left|\frac{\partial \boldsymbol{\phi}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|$ which is defined and non-zero P-almost everywhere. Then if we define a transition operator via the transition density

$$\begin{aligned} \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x}) &= \delta(\boldsymbol{x}' - \boldsymbol{\phi}(\boldsymbol{x}))\alpha(\boldsymbol{x}) + \delta(\boldsymbol{x}' - \boldsymbol{x})(1 - \alpha(\boldsymbol{x})), \\ \alpha(\boldsymbol{x}) &= \min\left\{1, \frac{p \circ \boldsymbol{\phi}(\boldsymbol{x})}{p(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x})\right\}, \end{aligned} \tag{2.42}$$

then this transition operator will leave the target distribution P invariant. This deterministic transition operator variant can be viewed as a special case of the trans-dimensional Metropolis–Hastings extension introduced by Green [79, 91]. To generate from this transition operator from a current state $\boldsymbol{x}$ we compute the proposed move $\boldsymbol{\phi}(\boldsymbol{x})$ and accept the move with probablity $\alpha(\boldsymbol{x})$. We can demonstrate that this transition operator leaves P invariant by directly verifying (2.30)

$$\int_X \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{2.43}$$

$$= \int_X \delta(\boldsymbol{x}' - \boldsymbol{\phi}(\boldsymbol{x}))\, \alpha(\boldsymbol{x})\, p(\boldsymbol{x}) + \delta(\boldsymbol{x}' - \boldsymbol{x})(1 - \alpha(\boldsymbol{x}))\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{2.44}$$

$$= \int_X \delta(\boldsymbol{x}' - \boldsymbol{y})\, \alpha \circ \boldsymbol{\phi}(\boldsymbol{y})\, p \circ \boldsymbol{\phi}(\boldsymbol{y})\, D_{\boldsymbol{\phi}}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y} + (1 - \alpha(\boldsymbol{x}'))\, p(\boldsymbol{x}') \tag{2.45}$$

$$= p(\boldsymbol{x}') + \alpha \circ \boldsymbol{\phi}(\boldsymbol{x}')\, p \circ \boldsymbol{\phi}(\boldsymbol{x}')\, D_{\boldsymbol{\phi}}(\boldsymbol{x}') - \alpha(\boldsymbol{x}')\, p(\boldsymbol{x}'). \tag{2.46}$$

In going from (2.44) to (2.45) we use a change of variables $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ in the integral. As $\boldsymbol{\phi}$ is an involution we have that $\boldsymbol{\phi} \circ \boldsymbol{\phi}(\boldsymbol{x}') = \boldsymbol{x}'$ and $D_{\boldsymbol{\phi}} \circ \boldsymbol{\phi}(\boldsymbol{x}') = D_{\boldsymbol{\phi}}(\boldsymbol{x}')^{-1}$ and so

$$\begin{aligned} \alpha \circ \boldsymbol{\phi}(\boldsymbol{x}')\, p \circ \boldsymbol{\phi}(\boldsymbol{x}')\, D_{\boldsymbol{\phi}}(\boldsymbol{x}') &= \min\left\{p \circ \boldsymbol{\phi}(\boldsymbol{x}')\, D_{\boldsymbol{\phi}}(\boldsymbol{x}'),\, p(\boldsymbol{x}')\right\} \\ &= \alpha(\boldsymbol{x}')\, p(\boldsymbol{x}'). \end{aligned} \tag{2.47}$$

The last two terms in (2.46) therefore cancel and we have that (2.30) is satisfied by the transition operator defined by (2.42). Although this transition operator leaves the target distribution P invariant, it is clear that it will not generate an ergodic Markov chain. Starting from a point $\boldsymbol{x}$ the next chain state will be either $\boldsymbol{\phi}(\boldsymbol{x})$ if the proposed move is accepted or $\boldsymbol{x}$ if rejected. In the former case the next proposed move will be to $\boldsymbol{\phi} \circ \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$ i.e. back to the original state. Therefore the chain will visit a maximum of two states. However as noted previously we can sequentially compose individual transition operators which all leave a target distribution invariant. Therefore a deterministic proposal Metropolis–Hastings transition can be combined with other transition operators to ensure the chain is irreducible and aperiodic.

In general for a Metropolis–Hastings transition operator to be irreducible, it is necessary that the proposal operator is irreducible [221], however this is not sufficient. For a target density which is positive everywhere on $X = \mathbb{R}^D$, then a sufficient but not necessary condition for irreducibility is that the proposal density is positive everywhere [194]. If the set of points with a non-zero probability of rejection has non-zero P-measure, then the transition operator is aperiodic [221].

A common choice of proposal density when the target distribution is defined on a $(\mathbb{R}^D, \mathscr{B}(\mathbb{R}^D))$ is a multivariate normal density centred at the current state i.e. $q(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{\Sigma})$ which satisfies the positivity condition for irreducibility. In general we would expect improved performance with a proposal density covariance $\boldsymbol{\Sigma}$ which is proportional to the true covariance of the target distribution [198], in practice we do not have access to the true covariance and so typically an isotropic proposal density is used with covariance $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$ controlled by a single scale parameter σ, often termed the *step size* or *proposal width*. This proposal density is symmetric so the simplified acceptance rule (2.41) can be used, further the prosal density depends only on the differ-

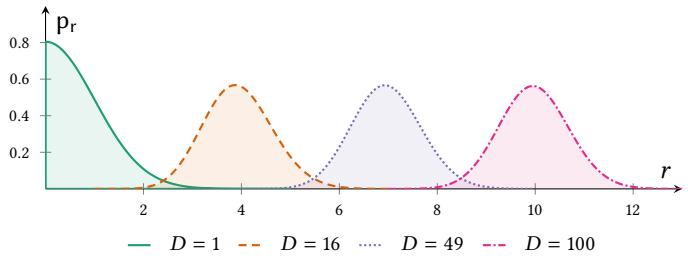


Figure 2.8.: Illustration of concentration of measure in a multivariate normal distribution. The plots shows the probability density of the distance from the origin $\mathrm{r} = \|\mathbf{x}\|_2$ of a D-dimensional multivariate normal random vector $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for different dimensionalities D. As the dimension increases most of the mass concentrates away from the origin around a spherical shell of radius $\sqrt{D}$. For a multivariate normal random vector with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ this generalises to the mass being mainly in an ellipsoidal shell aligned with the eigenvectors of $\boldsymbol{\Sigma}$ and centred at $\boldsymbol{\mu}$.

ence $\boldsymbol{x}' - \boldsymbol{x}$ with Metropolis–Hastings methods having these properties often termed *random-walk Metropolis*.

Random walk Metropolis methods have been extensively theoretically studied, with sufficient conditions known in some cases to ensure geometric ergodicity of a chain [141, 196] though these can be hard to verify in practical problems. There has also been much work on practical guidelines and methods for tuning the free parameters in the algorithm, including approaches for tuning the step-size using acceptance rates [73, 193] and adaptive variants which automatically estimate a non- isotropic proposal covariance [93, 198].

In general the choice of proposal density will be key in determining the efficiency of Metropolis–Hastings MCMC methods. Ideally we want to be able to propose large moves in the state space to reduce the dependencies between successive chain states and so increase the number of effective samples, however this needs to be balanced with maintaining a reasonable acceptance probability with large proposed moves often having a low acceptance probability. Figure 2.7 gives an illustration of this tradeoff in a one-dimensional example.

In high-dimensional spaces this issue is much more severe due to the phenomenon of *concentration of measure*: in probability distributions defined on high-dimensional spaces most of the probability mass will tend to be concentrated into a 'small' subset of the space [11, 130]. An illustration of this phenomenon for the multivariate normal distribution

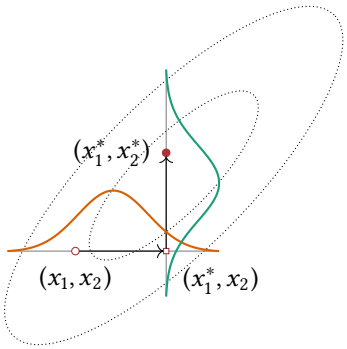


Figure 2.9.: Schematic of Gibbs sampling transition in a bivariate normal target distribution (ellipses indicate constant density contours). Given an initial state $\mathbf{x} = (x_1, x_2)$, the x_1 (horizontal) co-ordinate is first updated by independently sampling from the normal conditional $p_{x_1|x_2}(\cdot \mid x_2)$, represented by the orange curve. The new partially updated state is then $\mathbf{x} = (x_1^*, x_2)$. The second x_2 (vertical) co-ordinate is then independently resampled from the normal conditional $p_{x_2|x_1}(\cdot \mid x_1^*)$, shown by the green curve. The final updated state is then $\mathbf{x} = (x_1^*, x_2^*)$.

is shown in Figure 2.8, where the mass in high dimensions is mostly located in a thin ellipsoidal shell. The region where most of the mass concentrates, termed the *typical set* of the distribution, will for the target distributions of interest generally have a significantly more complex geometry. Finding proposals which can make large moves in such settings is challenging: moves in most directions will have a probability of acceptance which exponentially drops to zero as the distance away from the current state is increased and so simple proposal densities which ignore the geometry the typical set such as those used in random-walk Metropolis will need to make very small moves to have a reasonable probability of acceptance [23].

To tackle this issue methods have been proposed which exploit more information about the target distribution than just the point evaluations of the density in the Metropolis–Hasting acceptance probability term. *Hamiltonian Monte Carlo* (HMC) methods, which make use of the *gradient* of the target density, are a particularly important class of such methods and a central focus of this thesis. We will discuss HMC methods in detail in Chapter **??**.

2.1.6.2 Gibbs sampling

Gibbs sampling [72, 77], originally proposed by Geman and Geman for image restoration using a Markov random field image model, is based

Algorithm 3 Sequential scan Gibbs transition.

Input: $\boldsymbol{x}_n$: current chain state, I : ordered set of indices of all individual variables in chain state, $\{p_i\}_{i \in I}$: set of complete conditionals of target density p which can all be sampled from.

Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

$\boldsymbol{x} \leftarrow \boldsymbol{x}_n$
for $i \in I$ **do**
    $x_i \sim p_i(\boldsymbol{x}_{\setminus i})$ ▷ Resample x_i from p_i given *current* $\boldsymbol{x}_{\setminus i}$.
$\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}$
return $\boldsymbol{x}_{n+1}$

on the observation that a valid transition operator for a joint target distribution across many variables, is one which updates only a subset of the variables and leaves the conditional distribution on that subset given the rest invariant. Although if used in isolation a transition operator which only updates some components of the state will not give an ergodic chain, as discussed previously multiple transition operators can be combined together to achieve ergodicity.

More specifically the original formulation of Gibbs sampling defines a Markov chain by sequentially independently resampling each individual variable in the model from its conditional distribution given the current values of the remaining variables. If I is an index set over the individual variables in the vector target state $\mathbf{x}$, then for each $i \in I$ we partition the state $\mathbf{x}$ into the i^{th} variable x_i and a vector containing all the remaining variable values $\mathbf{x}_{\setminus i}$. For each $i \in I$ the target density can be factorised in to the marginal density $p_{\setminus i}$ on $\mathbf{x}_{\setminus i}$ and conditional density p_i on x_i given $\mathbf{x}_{\setminus i}$, i.e.

$$p(\boldsymbol{x}) = p_i(x_i \mid \boldsymbol{x}_{\setminus i})\, p_{\setminus i}(\boldsymbol{x}_{\setminus i}), \tag{2.48}$$

with the conditional densities $\{p_i\}_{i \in I}$ termed the *complete conditionals* of the target density. If each of these complete conditionals corresponds to a distribution we can generate samples from (for example using a transform method or rejection sampling) then we can apply the sequential Gibbs sampling transition operator defined in Algorithm 3 and visualised for a bivariate example in Figure 2.9.

The sequential Gibbs transition is irreducible and aperiodic under mild conditions [40, 195]. Rather than using a deterministic sequential scan through the variables, an alternative is to randomly sample without replacement the variable to update on each iteration; unlike the sequen-

tial scan version this defines a reversible transition operator and is in fact a valid way to construct a reversible transition from any set of individually reversible transition operators with a common invariant distribution. The random update variant is more amenable to theoretical analysis and comes with some stronger guarantees, however in practice the ease of implementation of the sequential scan variant and computational benefits in terms of memory access locality mean it seems to be more often used in practice [100]. A compromise between the completely random updates and a sequential scan is to randomly permute the update order have each complete scan.

A apparent advantage of Gibbs sampling over Metropolis–Hastings is the lack of a proposal density which needs to be tuned. This is has helped popularise 'black-box' implementations of Gibbs sampling such as the probabilistic modelling packages BUGS [81] and JAGS [176]. A well-known issue with Gibbs sampling however is that its performance is highly dependent on the parameterisation used for the target density [183], with strong correlations between variables leading to large dependencies between successive states and slow convergence to stationarity. This can be alleviated in some cases by using a suitable reparameterisation to reduce dependencies between variables, however this restores the difficulty of tuning free parameters.

The updates do not necessarily need to be performed by sampling from complete conditionals of single variables - in some cases the complete conditional of a vector variables has a tractable form which can be sampled from as a 'block'; this motivates the name *block Gibbs samling* for such variants. By accounting for the dependencies between the variables in a block this can help alleviate some of the issues with highly correlated targets where applicable.

Compound constructions such as *Metropolis-within-Gibbs* are sometimes used to refer to methods which sequentially apply *Metropolis–Hastings* transition operators which each update only a subset of variables in the target distribution. We will prefer to however consider the defining feature of Gibbs sampling as being exact sampling from one or more conditionals rather than sequentially applying transition operators which update only subsets of variables and so will only refer to 'Gibbs sampling' in that context.

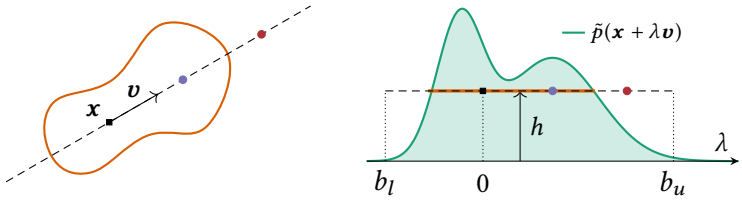


Figure 2.10.: Schematic of linear slice sampling, showing 'plan' (left) and 'cross-sectional' (right) views of a bivariate target density. Orange curve (left) and line (right) indicates a constant density slice S_h. Black square indicates current target state value $\boldsymbol{x}$ and the dashed line is the one-dimensional linear sub-space ligned with the vector $\boldsymbol{v}$ which a new value from the state will be sampled on. The extents of the dashed line segment represent the initial bracket new proposed states will be drawn from. Points are proposed on the line by drawing a value uniformly from the current bracket. The red circle represents an initial proposed point which is not in the slice and so the right bracket edge is shrunk to this point. The violet circle shows a second sampled point from the new reduced bracket, this point within the slice and so returned as the updated target state.

Algorithm 4 Linear slice sampling transition.

Input: $\boldsymbol{x}_n$: current chain state, $\tilde{p}$: unnormalised target density,
q : slice vector density, M : maximum number of step out iterations.
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

$h \sim \mathcal{U}(0, \tilde{p}(\boldsymbol{x}_n))$ ▷ Sample slice height
$\boldsymbol{v} \sim q$ ▷ Sample vector setting slice line and initial bracket width
$b_u \sim \mathcal{U}(0, 1)$ ▷ Uniformly sample bracket around current state
$b_l \leftarrow b_u - 1$
if $M > 0$ **then** $(b_l, b_u) \leftarrow$ LinearStepOut$(\boldsymbol{x}_n, b_l, b_u, M)$
$\lambda \sim \mathcal{U}(b_l, b_u)$
while True **do**
  $\boldsymbol{x}^* \leftarrow \boldsymbol{x}_n + \lambda \boldsymbol{v}$ ▷ Update proposed state
  if $\tilde{p}(\boldsymbol{x}^*) \leq h$ **then** ▷ Proposed point not on slice
    if $\lambda < 0$ **then** $b_l \leftarrow \lambda$ **else** $b_u \leftarrow \lambda$ ▷ Shrink slice bracket
    $\lambda \sim \mathcal{U}(b_l, b_u)$ ▷ Sample uniformly from new bracket
  else ▷ Proposed state on slice
    return $\boldsymbol{x}^*$
function LinearStepOut$(\boldsymbol{x}_n, b_l, b_u, M)$
  $L \sim$ UniInt$(0, M)$ ▷ Sample integer uniformly from $[0, M]$
  $U \leftarrow M - L$
  while $L > 0$ **and** $\tilde{p}(\boldsymbol{x}_n + b_l \boldsymbol{v}) > h$ **do** ▷ Step out lower bracket edge
    $b_l \leftarrow b_l - 1$
    $L \leftarrow L - 1$
  while $U > 0$ **and** $\tilde{p}(\boldsymbol{x}_n + b_u \boldsymbol{v}) > h$ **do** ▷ Step out upper bracket edge
    $b_u \leftarrow b_u + 1$
    $U \leftarrow U - 1$
  return b_l, b_u

2.1.6.3 *Slice sampling*

The final class of MCMC methods we will consider is *slice sampling*. Slice sampling is a family of auxiliary variable methods which exploit the same observation as used to motivate rejection sampling - to sample from a target distribution it is sufficient to uniformly sample from the volume beneath a graph of the target density function. Rather than attempt to generate independent points from this volume as in rejection sampling, slice sampling instead constructs a transition operator which leaves the uniform distribution on this volume invariant.

The method we will concentrate on here was proposed by Neal [159, 160] with a related algorithm which uses per data-point auxiliary variables in Bayesian inference problems developed concurrently by Damien, Wakefield and Walker [50]. Murray, Adams and Mackay later proposed *elliptical slice sampling* [155], a variant of the original slice sampling construct which leverages the ellipsoidal geometry of the typical set of multivariate normal distributions to construct a slice sampling transition operator which is particularly effective for target distributions which are well approximated by a multivariate normal.

Slice sampling defines a Markov chain on an augmented state space by introducing an auxiliary *height* variable $\mathsf{h} \in [0, \infty)$ in addition to the original target state $\mathbf{x} \in X$. The conditional density on the height variable is $\mathsf{p}_{\mathsf{h}|\mathbf{x}}(h \mid \mathbf{x}) = \frac{1}{\tilde{p}(\boldsymbol{x})} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h)$, i.e. uniform over the interval between zero and the unnormalised target density value. The joint density on the augmented space is then

$$\mathsf{p}_{\mathbf{x},\mathsf{h}}(\boldsymbol{x}, h) = \frac{1}{\tilde{p}(\boldsymbol{x})} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h) \frac{\tilde{p}(\boldsymbol{x})}{Z} = \frac{1}{Z} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h). \tag{2.49}$$

Marginalising this joint density over $\mathbf{x}$ recovers the original target density i.e. $\mathsf{p}_{\mathbf{x}} = p$. Therefore if we can construct a Markov chain with (2.49) as its unique invariant distribution, the $\mathbf{x}$ components of the chain state will be dependent samples from the target distribution for a chain at stationarity and so can be used to compute MCMC estimates. Several of the methods we will introduce later in this thesis will also exploit this idea of defining a Markov chain on an augmented state space by introducing auxiliary variables.

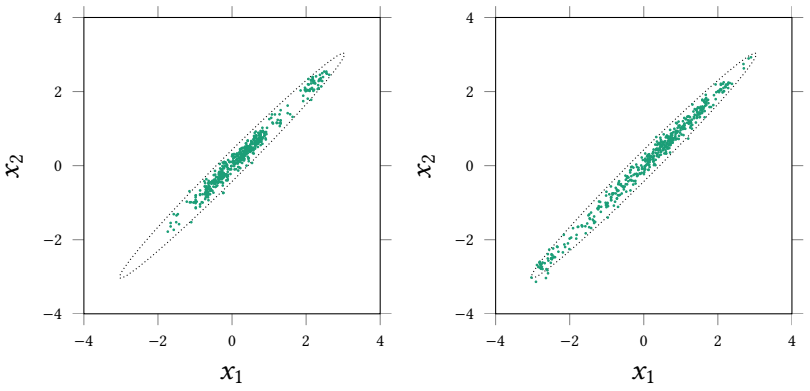
The overall slice sampling transition is then formed of the sequential composition of a transition operator which updates h given $\mathbf{x}$ and a

second operator which updates $\mathbf{x}$ given h, each leaving the distributions corresponding to the conditional densities $p_{h|\mathbf{x}}$ and $p_{\mathbf{x}|h}$ respectively invariant, and so by the same argument as for Gibbs sampling the overall transition leaving the target distribution invariant. By construction the conditional density $p_{h|\mathbf{x}}$ is a simple uniform density and so the first transition operator is a Gibbs sampling type update in which the height variable is independently resampled from $\mathcal{U}(0, \tilde{p}(\boldsymbol{x}))$, where $\boldsymbol{x}$ is the current value of the target state $\mathbf{x}$.

The conditional density $p_{\mathbf{x}|h}(\boldsymbol{x} \mid h)$ is also locally uniform, equal to a positive constant whenever $\tilde{p}(\boldsymbol{x}) > h$ and zero elsewhere. However we can only evaluate the density up to an unknown constant as we cannot compute the Lebesgue measure of the set $S_h = \{\boldsymbol{x} \in X : \tilde{p}(\boldsymbol{x}) > h\}$ that the density is non-zero over. In general S_h, which is the eponymous *slice* of slice sampling (so called as it represents a slice through the volume under the density curve at a fixed height h), will have a complex geometry including potentially consisting of several disconnected components in the case of multimodal densities. The complexity of the slices generally prevents us therefore from independently sampling a new value for $\mathbf{x}$ uniformly from S_h and so we cannot use a full Gibbs sampling scheme corresponding to sequentially independently sampling from $p_{h|\mathbf{x}}$ and $p_{\mathbf{x}|h}$.

A key contribution of [160] was to introduce an elegant method for constructing a transition operator which leaves $p_{\mathbf{x}|h}$ invariant. In particular the method proposed has few free parameters to tune, has an efficiency which is relatively robust to the choices of the free choices that are introduced, and will for smooth target densities always move the target state by some amount (in contrast to the potential for rejections in Metropolis–Hastings methods). This method is summarised in Algorithm 4 and a schematic visualisation of the process shown in Figure 2.10.

An important first step in the algorithm is reducing the problem of generating a point uniformly on the multidimensional slice S_h to making a move on a one-dimensional linear subspace of this slice (motivating our naming of this algorithm *linear slice sampling*) which includes the current $\mathbf{x}$ state. In the original description of the algorithm in [160] the one-dimensional subspace is chosen to be axis-aligned, correponding to updating a single component of the target state.



(a) Axis-aligned updates (b) Random direction updates

Figure 2.11.: Samples generated using (a) axis-aligned versus (b) random-direction linear slice sampling in a correlated bivariate normal distribution. In both cases 1000 transitions where performed (with random selection of axis to update on each iteration in (a)) with every second sampled state shown. the maximum number of step out iterations is $M = 4$ and the initial bracket width is fixed at $w = 1$. The dotted ellipse shows the contour of the target density which contains 0.99 of the mass. The random direction chain is able to explore the typical set of the target distribution more effectively in this case with the axis-aligned updates leading to slower diffusion along the major axis of the elliptical contour.

In this case the restriction of the slice on to the one-dimensional subspace is entirely specified by the conditional density on the chosen variable component given the current values of the remaining components in the state. Slice sampling transitions for each variable in the target state can then be applied sequentially akin to Gibb sampling, but with the advantage over Gibbs of not requiring the complete conditionals to be of a tractable form we can generate exact samples from. If conditional independency structure in the target density means the complete conditionals depend only on local subsets of variables in the target state using updates of this form has the advantage of exploiting this locality. As with Gibbs sampling however applying slice sampling in this manner makes performance strongly dependent on the parameterisation of the target density, with large magnitude correlations likely to lead to slow exploration of the space.

In [160] various specifically multivariate extensions of the algorithm are suggested which could help counter this issue, however they add significant implementation complexity compared to the basic algorithm. A simple alternative is to define the one-dimensional subspace as being the line defined by a randomly chosen vector and passing through the

current value of $\mathbf{x}$. If this vector is generated independently of the current state this is sufficient to ensure the overall transition retains the correct invariant distribution.

If little is known about the target distribution a reasonable default choice is to sample a unit vector of the required dimensionality by generating a random zero-mean isotropic covariance multivariate normal vector and then scaling it to unit norm; if an approximate covariance matrix $\hat{\Sigma}$ is known for the target density then instead generating the vector from $\mathcal{N}(\mathbf{0}, \Sigma)$ prior to normalising might be a better choice (as it favours moves aligned with the principle eigenvectors of Σ) however in this case elliptical slice sampling will often be a better choice.

This random-direction slice sampling variant is discussed in comparison to elliptical slice sampling in [155]. It is also bears resemblance to the scheme proposed in [42] which uses the same auxiliary variable formulation as slice sampling, but there the random direction is chosen in $X \times [0, \infty)$ i.e. to update both $\mathbf{x}$ and h and not used with the remainder of Neal's slice sampling algorithm. An example comparison of appying axis-aligned and random-direction linear slice sampling updates to a strongly positively correlated bivariate normal target distribution is shown in Figure 2.11. In this toy example the random-direction updates are able to more effectively explore the target distribution.

The generation of the vector $\boldsymbol{v}$ determining the one-dimensional subspace of the slice the update is performed on is represented in Algorithm 4 by Line 2 by $\boldsymbol{v}$ being generated from a density q. As well as specifying the *direction* of the slice line, the vector $\boldsymbol{v}$ also specifies a scale along this line. In Neal's description of the algorithm this is represented by the explicit *bracket width* parameter w. Here instead we assume this parameter is implicitly defined by the Euclidean norm of the vector $\boldsymbol{v}$, through suitable choice of q this allowing for direction dependent scales and also the possibility of randomisation of the scale; as we will see shortly however compared to for example random-walk Metropolis updates with a normal proposal, linear slice sampling is much less sensitive to the choice of scale parameters, therefore a single fixed scale will often be sufficient.

Once the slice line direction and scale has been chosen, the remainder of the algorithm can be split into two stages: selection of an initial bracket on the slice line and including the point corresponding to the

current state; iteratively uniformly sampling points within the current bracket, accepting the point if it is within the slice S_h otherwise shrinking the bracket and repeating. The algorithm proposed by Neal ensures both these stages are performed reversibly such that the detailed balance condition (2.34) is maintained.

The *slice bracket* defines a contiguous interval $\lambda \in [b_l, b_u]$ on the slice line $\boldsymbol{x}^*(\lambda) = \boldsymbol{x}_n + \lambda \boldsymbol{v}$ and always includes the point $\lambda = 0$ corresponding to the current state. The initial bracket is chosen by sampling a upper bound b_u uniformly from $[0, 1]$ and then setting $b_l \leftarrow b_u - 1$; in the λ slice line coordinate system this corresponds to a bracket width of one, however in general the slice line vector $\boldsymbol{v}$ can have non-unit length and so defines the initial bracket width in the target variable space. Randomising the positioning of the current state within the bracket ensures reversibility as the resulting bracket would have an equal probability (density) of being selected from any other point in the bracket (which the final accepted point will be within).

In general only a subset of the points in the current slice bracket will be within the slice S_h. As new states are proposed by sampling a point uniformly from the current bracket, the probability of such a proposal being in the slice and so accepted will be equal to the proportion of the bracket that intersects with the slice S_h. In general therefore it is desirable for the bracket to include as much of the slice as possible while not making the proportion of the bracket intersecting with the slice too small such that many points need to be proposed before one on the slice is found. The magnitude of $\boldsymbol{v}$ determines the initial bracket extents and so should generally chosen based on any knowledge of the typical 'scale' of the target density. Often however we will have little prior knowledge about such scaling however and the typical scale will often vary signifcantly across the target space, and so we may choose an initial bracket which includes only a small proportion of the intersection of the slice with the slice line.

The stepping out routine proposed by [160] and detailed in Lines 14 to 23 in Algorithm 4 is designed to counter this issue. The initial slice bracket $[b_l, b_u]$ is iteratively 'stepped-out' by incrementing / decrementing the upper / lower bracket bounds until the corresponding endpoint of the bracket lies outside the slice or a pre-determined maximum number of steps out have been performed. Ideally the step out routine will return a bracket which contains all of the intersection of the slice with

slice line while not also including too great a proportion of off slice points; in general the slice may be non-convex or consist of multiple disconnected components and so the intersection of the slice line with the slice may consist of multiple disconnected intervals in which case the stepping out routine will likely only expand the slice to include a subset of these intervals. The adaptivity provided by the stepping out routine will still however generally help to make the performance of the sampler much less sensitive to the choice of the bracket scale in contrast to for example random-walk Metropolis algorithms which typically use a single fixed scale.

Analagously to the randomisation of the initial bracket positioning, in the stepping out routine if a maximum number of step out iterations M is set, the resulting step 'budget' is randomly allocated between increments of the upper bound b_u and decrements of the lower bound b_l such that final extended bracket generated by the step out routine would have an equal probability of being generated from any point within the generated bracket interval. If M is set to zero this correponds to not performing any stepping out and simply using the initial sampled bracket; although reducing the robustness of the algorithm to the choice of the initial bracket width this option has the advantage of minimising the number of target density evaluations by not requiring additional density evaluations at the bracket endpoints during the step-out routine. An alternative 'doubling' step-out routine was also proposed in [160]. This has the advantage of exponentially expanding the slice bracket compared to the linear growth of the step-out routine described in Algorithm 4 and so can be more efficient in target distributions where the typical scales of the density varies across several orders of magnitude. The doubling procedure requires a more complex subsequent procedure for sampling points in the resulting bracket however to ensure reversibility.

Once the initial bracket has been generated and potentially stepped out, the remainder of the algorithm consists of sampling a point within the slice bracket which is within the slice S_h. This is done in an iterative manner by first sampling a point uniformly from the current bracket and checking if it is within the slice or not. If the proposed point is in the slice, the corresponding value for the target variables is returned at the new state. Otherwise the proposed point is set as the new upper or lower bound of the bracket such that the point corresponding to the

Algorithm 5 Elliptical slice sampling transition.

Input: $\boldsymbol{x}_n$: current chain state, $\tilde{p}$: unnormalised target density,
$\boldsymbol{\mu}, \boldsymbol{\Sigma}$: mean and covariance of normal approximation to target.
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

$h \sim \mathcal{U}(0, \tilde{p}(\boldsymbol{x}_n) / \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}))$ ▷ Sample slice height
$\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ ▷ Sample vector setting slice ellipse
$\theta_u \sim \mathcal{U}(0, 2\pi)$ ▷ Uniformly sample bracket around current state
$\theta_l \leftarrow \theta_u - 2\pi$
$\theta \leftarrow \theta_u$
while TRUE **do**
    $\boldsymbol{x}^* \leftarrow (\boldsymbol{x}_n - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{v} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}$ ▷ Update proposed state
    if $\tilde{p}(\boldsymbol{x}^*) / \mathcal{N}(\boldsymbol{x}^* \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \leq h$ **then** ▷ Proposed point not on slice
        if $\theta < 0$ **then** $\theta_l \leftarrow \theta$ **else** $\theta_u \leftarrow \theta$ ▷ Shrink slice bracket
        $\theta \sim \mathcal{U}(\theta_l, \theta_u)$ ▷ Sample uniformly from new bracket
    else
        return $\boldsymbol{x}^*$

current state remains within the bracket. This shrinks the bracket by removing an interval where it is known at least some regions of are not within the slice. A new point is then sampled uniformly from the new smaller bracket and the procedure repeats until an acceptable point in the slice is found.

The iterative shrinking of the slice bracket implemented by this procedure introduces a further level of adaptivity in to the slice sampling algorithm, meaning that even if only a small proportion of the initial bracket lies within the slice only relatively few iterations will be needed still till the bracket is shrunk sufficiently for there to be a high probability of proposing a point within the bracket. By ensuring the point corresponding to the current state always remains within the current bracket, reversibility is maintained.

An alternative to the linear slice sampling procedure just described, is the *elliptical slice sampling* method proposed in [155] and described in Algorithm 5. As suggested by the name, in elliptical slice sampling rather than proposing points on a line instead an elliptical path in the target space is defined and new points proposed on this ellipse.

Elliptical slice sampling is intended for use in target distributions which are reasonably well approximated by a known multivariate normal distribution with density $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. This Gaussian density might correspond to a multivariate normal prior distribution on model latent variables where the dependence between the latent and observed variables is only weak and so the posterior remains well approximated by the

prior or a Gaussian approximation fitted directly to the target distribution using one of the optimisation based approximate inference strategies we will discuss in the following section [164].

In each elliptical slice sampling transition an auxiliary vector $\boldsymbol{v}$ is independently sampled from the distribution with density $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. If the target distribution was exactly described by the multivariate normal density we could use this independent draw directly as the new chain state (though obviously in this case there would be no advantage in formulating as an MCMC method). In reality the target distribution will only approximately described by the multivariate normal density $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and so we wish to instead use this independent draw to define a Markov transition operator that will potentially move the state to a point nearly independent of the current state, but is also able to back off to more conservative proposals closer to the current chain state. This is achieved by defining an elliptical path in target space centred at $\boldsymbol{\mu}$, passing through the current chain state $\boldsymbol{x}_n$ and the auxiliary vector $\boldsymbol{v}$ and parameterised by an angular variable θ

$$\boldsymbol{x}^*(\theta) = (\boldsymbol{x}_n - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{v} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu} \tag{2.50}$$

If we generated θ uniformly from $\mathcal{U}(0, 2\pi)$ then the corresponding proposed transition $\boldsymbol{x}^*(\theta)$ would exactly leave a distribution with density $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ invariant. As we instead wish to leave the target distribution invariant, a slice sampling algorithm is used to find a θ which accounts for the difference between the target distribution and multivariate normal approximation. An auxiliary slice height variable h is sampled uniformly from $\mathcal{U}(0, \tilde{p}(\boldsymbol{x}_n) / \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}))$ and used to define a slice $S_h = \{\boldsymbol{x} \in X : \tilde{p}(\boldsymbol{x}_n) / \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) < h\}$. Similar to the linear slice sampling algorithm, a bracket $[\theta_l, \theta_u]$ on the elliptical path is randomly placed around $\theta = 0$ corresponding to the current state $\boldsymbol{x}_n$. Unlike the requirement to choose a suitable initial bracket width in linear slice sampling however, we can define the initial bracket in elliptical slice sampling to include the entire elliptical path i.e. $\theta_l = \theta_u - 2\pi$; we only need to randomise the 'cut-point' defining the initial end-points of the bracket to ensure reversibility. This removes the need to choose an initial bracket width (defined by $|\boldsymbol{v}|$ in our description of the linear slice algorithm) and for any step out procedure, and so beyond choosing the multivariate normal approximation elliptical slice sampling does not have any free settings which need to be tuned.

Once the initial bracket is defined, a directly analagous iterative procedure to that used in the linear slice sampling algorithm is used to find a θ value corresponding to a point in the slice while using rejected proposed points to shrink the bracket. As with linear slice sampling, providing the target density is a smooth function and so the intersection of the elliptical path with the slice is a non-zero measure set, then the state moved to by the elliptical slice sampling transition operator will never be equal to the previous state.

2.1.7 Summary

The sampling approaches to approximate inference described in this section allow tractable estimation of the integrals involved in many inference problems. In cases where we can tractably generate inpedendent samples from the target distribution, the $\frac{1}{N}$ scaling of the variance of Monte Carlo estimates of expectations with the number of samples N, independent of the dimensionality of the space being integrated over, allows computation of estimates which are sufficiently accurate for many practical purposes without the infeasible exponential blow-up in computation of quadrature methods. Further simple Monte Carlo methods are trivially parallelisable meaning even if generation of each independent sample is relatively expensive, parallel compute devices such as GPUs and CPU clusters can easily be exploited if available.

Generating independent samples from distributions on high-dimensional spaces with complex dependencies between the variables is generally non-tractable however. Transform sampling methods offer a scalable approach for generating independent samples for a few special cases such as the multivariate normal distribution. Rejection sampling is more generally applicable however still requires idendifying a proposal distribution with a density which (scaled by a constant) strictly upper bounds the target distribution density. In high-dimensional distributions it will generally be infeasible to find a suitable proposal distribution which is a sufficiently tight bound, with in general the proportion of accepted samples becoming exponentially small as the dimensionality is increased, making rejection sampling only applicable to low-dimensional distributions in practice.

Importance sampling methods may seem initially to offer a solution to this issue, allowing consistent estimates of the values of arbitrary integrals (including marginal density estimations that may not be directly es-

timated with standard Monte Carlo approaches) providing we can generate independent sample from a distribution which meets the weak condition of having a density which is non-zero everywhere the integrand is non-zero. The general importance sampling estimator is formed as a ratio of two Monte Carlo estimates (2.17); providing these estimators have finite variance each will show the typical $\frac{1}{N}$ scaling of the estimator variance with the number of samples used. In general however unless the importance distribution used is very closely matched to the target distribution, simple importance sampling methods are also impractical in high-dimensional spaces as the constant factors in the variances of the Monte Carlo estimates can grow exponentially with dimension meaning an infeasibly large number of samples are needed to compute estimates of a useful level of accuracy.

Markov chain Monte Carlo methods offer a more a scalable alternative to approximate inference in complex high-dimensional probabilistic models. MCMC methods exploit the intuition that finding a good 'local' approximation to the target distribution — for example within a small region around a poin or varying along only one direction in the target space — is usually a much more tractable task than finding an approximation which matches the target distribution globally, particularly in high dimensional spaces where concentration of measure will mean the typical set of a distribution is usually concentrated in to small region of the space and even seemingly small mismatches between an approximation and target can lead to very little overlap between the target and approximation typical sets. Markov chain theory shows how we can exploit such local approximations to make perturbative updates to a Markov chain state such that the realisation of the chain converge to generating dependent samples from the target distribution of interest. Although theory can guarantee MCMC estimates will eventually converge to the correct values it is usually difficult to assess the rate of that convergence in practical problems making it difficult to diagnose chain convergence or a lack thereof. This can make application of MCMC methods more challenging from a user-perspective than simple Monte Carlo methods as some level of expertise is usually needed to diagnose and find solutions to convergence issues.

We discussed three general methods for constructing Markov chains which leave a target distribution invariant — the Metropolis–Hastings method, Gibbs sampling and slice sampling. Each of the constructs of-

fers its own advantages and disadvantages and no one method dominates the others in all aspects. It is also common to combine transition operators of different types, for exampling using different methods to update distinct subset of the variables in a model given the remaining variables, or use multiple updates to the same variables to potentially combine the good properties of the individual operators.

Due in part probably to their ease of implementation, Metropolis–Hastings methods based on simple proposal distribution such as Gaussian random-walk Metropolis methods are very commonly used in practice. Performance of such methods is however usually very dependent on the good choice of the algorithm free parameters such as the proposal width / step size, with poor choices leading to very slow mixing chains. In target distributions with a complex geometry, for example where the appropriate scale for state updates may differ signficiantly across the target space, Metropolis–Hastings methods using simple fixed proposals may be unable to mix well even when optimally tuned. One solution to this issue is to use proposals which exploit more information about the local geometry of the distribution such as the gradient-based Hamiltonian Monte Carlo methods we will discuss in Chapter **??**.

Gibbs sampling methods are also very popular with general purpose implementations such as BUGS [81] and JAGS [176] having supported their application to a wide range of models. Although requiring that we are able to decompose the target distribution into complete conditionals we can tractably generate independent samples from, in the models where it can be applied Gibbs sampling offers the advantage over Metropolis–Hastings methods of not requiring choosing and tuning the free parameters of the proposal distribution. As noted previously however the performance of Gibbs sampling methods is very dependent on the parameterisation of the target distribution, with parameterisations with strong dependencies between the variables tending to lead to very slowly mixing chains. Although this can be alleviated by reparameterisation or using block-updates to coupled variables in some cases, the resulting extra implementation and tuning requirements erode the simplicity advantage compared to competing methods.

The slice sampling algorithms introduced at the end of the last section are more complex than the corresponding algorithms for Metropolis–Hastings and Gibbs sampling, and this added implementation complex-

ity may explain in part the seemingly less widespread use of slice sampling methods in practice[9], although the relatively much more recent introduction of the algorithm is likely also a factor. The tradeoff achieved for this increased implementation complexity however are algorithms which usually require much less tuning than random-walk Metropolis methods to achieve reasonable performance and are more robust than both Gibbs sampling and random-walk Metropolis methods to target distributions with complex geometries. As noted in [155] due to the multiple target density evaluations per chain state update in slice sampling methods, often a well-tuned Metropolis–Hastings method will be able to achieve a greater efficiency in terms of effective samples per run time. However the sometimes significant user time required for tuning can often outweigh the gains in efficiency over slice sampling, and even if automatic adaptive tuning methods are used these will still often struggle to cope with distributions with locally varying geometry. One of the contributions of this thesis will be illustrating how slice sampling methods can often be applied to inference problems where Metropolis–Hastings methods are more commonly used with sometimes significant improvements in robustness and efficiency.

2.2 OPTIMISATION APPROACHES

The sampling-based approaches to approximate inference discussed in the previous section although a significant improvement in terms of computational complexity over quadrature methods can still be computationally demanding. In particular the MCMC methods which were identified as most suitable for inference in large, complex probabilistic models, will involve a minimum of one evaluation of the target distribution density per generated MCMC sample if using for example a simple random-walk Metropolis method and potentially tens or hundreds of density evaluation per sample if using more complex schemes such as slice sampling or the Hamiltonian Monte Carlo methods discussed in the next chapter. Typically chains will need to be run on the order of 10^2 to 10^4 iterations to ensure adequate converge of to the target distribution and to give a sufficient number of effective samples to

9 As a very rough metric at the time of writing the original 1953 Metropolis et al. publication [142] has 35,288 citations recorded on Google Scholar, the 1984 Geman and Geman publication [77] commonly cited as the original sourcce of the Gibbs sampling algorithm 20,485 citations and the 2003 Neal [160] slice sampling publication, 1,444 citations.

get reasonable estimates of the expectations of interest, with generally running multiple chains preferred to give additional robustness and to allow convergence diagnostics.

In large complex models each target density evalulation may be computationally expensive. In particular when performing inference conditioned on large sets of observed data the target density will typically factorise into a product (or sum in log space) of per datapoint factors. This means the cost of each target density evaluation will scale with the number of datapoints and so can become appreciable for large datasets. Alongside the increase in computational demands for large (in the sense of number of datapoints) datasets, for common forms of probabilistic models such as observed variables which are iid given a set of fixed dimension unobserved variables (parameters), local asymptotic normality results means that the target (posterior) distribution will become increasingly well approximated by a multivariate normal distribution as the number of observed data points increases.

In this section we will review an alternative class of approximate inference methods to the sampling approaches reviewed so far which tradeoff a generally lower computational cost than MCMC approaches for a loss of the ability to represent integrals across complex distributions with arbitary accuracy and so asymptoptic exactness guarantees. The central idea of these methods is to try to find a normalised probability density $q(\boldsymbol{x})$ from a 'simple' family that in some sense approximates the target density, i.e. $p(\boldsymbol{x}) \approx q(\boldsymbol{x})$. Depending on the family chosen for q, integrals of some functions f against the target density p, can be approximated by analytic solutions to integrals of f against q e.g. if $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ then we can approximate the mean of the target density as $\boldsymbol{\mu}$ and the covariance as $\boldsymbol{\Sigma}$. To compute integrals of more general functions f we will typically still need to resort to using a Monte Carlo approach; generally it will be possible to directly generate independent samples from q however while usually this will not be the case for p hence this two-step approach still offers (computational) advantages over directly applying a Monte Carlo approach. Often the approaches we will discuss also allow estimation of the normalising constant Z which may be needed for model comparison.

2.2.1 Laplace's method

For target densities p defined with respect to a D-dimensional Lebesgue measure λ^D, a simple approach for computing a multivariate normal approximation q to p is *Laplace's method*. Although not always strictly required, in general the method will work better for target densities with unbounded support, and more generally for targets which are as 'close to normal' as possible. Therefore a useful initial step will often be to apply a change of variables to the target density, such that the density on the transformed space has unbounded support, for example working with the density on the logarithm of a random variable with support only on positive values.

The key idea in Laplace's method is to form a truncated Taylor series approximation to the logarithm of the unnormalised target density

$$\begin{aligned}\log \tilde{p}(\boldsymbol{x}) \approx \log \tilde{p}(\boldsymbol{x}^*) &+ \boldsymbol{g}(\boldsymbol{x}^*)^\mathsf{T}(\boldsymbol{x}-\boldsymbol{x}^*) \\ &+ \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^\mathsf{T}\boldsymbol{H}(\boldsymbol{x}^*)(\boldsymbol{x}-\boldsymbol{x}^*),\end{aligned} \tag{2.51}$$

where the *gradient* and *Hessian* of $\log \tilde{p}$ are defined respectively as

$$\boldsymbol{g}(\boldsymbol{x}) = \frac{\partial \log \tilde{p}(\boldsymbol{x})^\mathsf{T}}{\partial \boldsymbol{x}} \quad \text{and} \quad \boldsymbol{H}(\boldsymbol{x}) = \frac{\partial^2 \log \tilde{p}(\boldsymbol{x})}{\partial \boldsymbol{x} \partial \boldsymbol{x}^\mathsf{T}}. \tag{2.52}$$

A matrix $\boldsymbol{M} \in \mathbb{R}^{D\times D}$ is positive semi definite*, denoted $\boldsymbol{M} \succeq 0$, iff $\boldsymbol{x}^\mathsf{T}\boldsymbol{M}\boldsymbol{x} \geq 0$ $\forall \boldsymbol{x} \in \mathbb{R}^D$ and* positive definite*, denoted $\boldsymbol{M} \succ 0$, if the inequality is made strict. Corresponding definitions for a* negative semi definite *matrices, $\boldsymbol{M} \preceq 0$, and* negative definite *matrices, $\boldsymbol{M} \prec 0$, are formed by reversing the sign of the inequality.*

If the point $\boldsymbol{x}^*$ the expansion is formed around is chosen to be a (local) maxima of $\log \tilde{p}$, which necessarily means that the gradient is zero, $\boldsymbol{g}(\boldsymbol{x}^*) = \boldsymbol{0}$, and the Hessian is negative definite, $\boldsymbol{H}(\boldsymbol{x}^*) \prec 0$, then

$$\log \tilde{p}(\boldsymbol{x}) \approx \log \tilde{p}(\boldsymbol{x}^*) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^\mathsf{T}\boldsymbol{H}(\boldsymbol{x}^*)(\boldsymbol{x}-\boldsymbol{x}^*). \tag{2.53}$$

Taking the exponential of both sides we therefore have that

$$\tilde{p}(\boldsymbol{x}) \approx \tilde{p}(\boldsymbol{x}^*) \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^\mathsf{T}(-\boldsymbol{H}(\boldsymbol{x}^*))(\boldsymbol{x}-\boldsymbol{x}^*)\Big). \tag{2.54}$$

This has the form of an unnormalised multivariate normal density with mean $\boldsymbol{x}^*$ and inverse covariance (precision) $-\boldsymbol{H}(\boldsymbol{x}^*)$.

This suggests setting the approximate density q to a multivariate normal density $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{x}^*, \boldsymbol{C})$ with $\boldsymbol{C} = -\boldsymbol{H}(\boldsymbol{x}^*)^{-1}$, i.e.

$$q(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{D}{2}}|\boldsymbol{C}|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^\mathsf{T}\boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{x}^*)\Big). \tag{2.55}$$

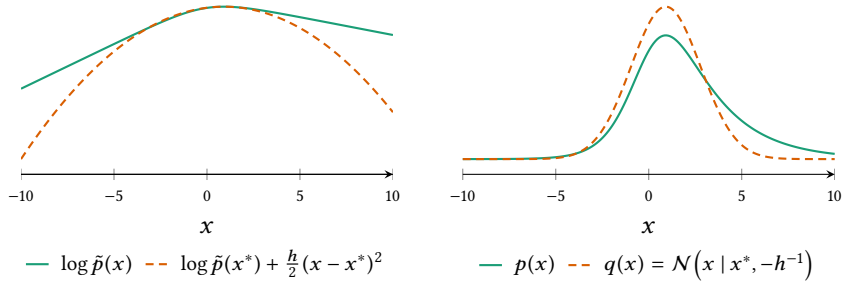


Figure 2.12.: Univariate example of Laplace's method. Left axis shows the logarithm of the unnormalised target density $\log \tilde{p}(x)$ (green curve) and the corresponding quadratic Taylor series approximation $\log \tilde{p}(x^*) + \frac{h}{2}(x - x^*)^2$ (dashed orange curve) around the maxima x^* with $h = (\partial^2 \log \tilde{p}/\partial x^2)|_{x^*}$. The right axis shows the corresponding normalised target density $p(x)$ (green curve) and approximate density $q(x) = \mathcal{N}\left(x \mid x^*, -h^{-1}\right)$ (dashed orange curve).

An example of applying Laplace's method to fit a normal approximation to a univariate generalised logistic target is shown in Figure 2.12.

As $q(\boldsymbol{x}^*) \approx p(\boldsymbol{x}^*) = \tilde{p}(\boldsymbol{x}^*)/Z$ we can also form an approximation $\tilde{Z}$ to the normalising constant Z for the target density

$$Z \approx \tilde{Z} = (2\pi)^{\frac{D}{2}} |\boldsymbol{C}|^{\frac{1}{2}} \tilde{p}(\boldsymbol{x}^*). \tag{2.56}$$

To use Laplace's method we need to be able to find a maxima of $\log \tilde{p}$ and evaluate the Hessian at this point. For simple unimodal target densities it may be possible to find the maxima and corresponding Hessian analytically. More generally if the gradient of $\log \tilde{p}$ can be calculated (using for example reverse-mode automatic differentation), then a maxima can be found by performing iterative gradient ascent. The Hessian can then be evaluated at this point using analytic expressions for the second partial derivatives or again by using automatic differentiation (by computing the Jacobian of the gradient of $\log \tilde{p}$).

Though relatively simple to calculate, Laplace's method will often result in an approximate density which fits poorly to the target. As it only uses local information about the curvature of the (log) target density at the mode, away from the mode the approximate density can behave very differently from the target density, for instance observe the poor fit to the tails of the target of the example shown in Figure 2.12. For multimodal densities, several different Laplace approximations can be calculated, each likely to at best capture a single mode well. For target

densities which are well approximated by a normal distribution, for instance due to asymptotic convergence to normality of a posterior for iid data, Laplace's method can give reasonable results however.

2.2.2 Variational inference

Laplace's method is limited by using information about the target density evaluated at only one point to fit the approximation. An alternative approach is to instead try to fit the approximate density based on minimising a global measure of 'goodness of fit' to the target; this is the strategy employed in *variational inference*.

The naming of variational inference arises from its roots in the *calculus of variations*, which is concerned with functionals (loosely a function of a function, often defined by a definite integral) and their derivatives. In particular it is natural to define the measure of the 'goodness of fit' of the approximate density to the target as a functional of the approximate density. The value of this functional is then minimised with respect to the approximate density function.

The most common functional used to define goodness of fit in variational inference is the *Kullback–Leibler* (KL) divergence [119]. The KL divergence in its most general form is defined for a pair of probability measures P and Q on a space X with P absolutely continuous with respect to Q as

$$\mathbb{D}_{\mathrm{KL}}[P \parallel Q] = \int_X \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right) \mathrm{d}P, \tag{2.57}$$

which is read as the KL divergence from P to Q. The KL divergence is always non-negative $\mathbb{D}_{\mathrm{KL}}[P \parallel Q] \geq 0$, with equality if and only if $P = Q$ almost everywhere. Intutively the KL divergence gives a measure of how 'close' two measures are[10], however it is not a true distance as it is asymmetric: in general $\mathbb{D}_{\mathrm{KL}}[P \parallel Q] \neq \mathbb{D}_{\mathrm{KL}}[Q \parallel P]$.

Generally we will work with probability densities rather than underlying probability measures. If p and q are the densities of two probability measures P and Q defined with respect to the same base measure μ on a space X, i.e. $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$ and $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$, then we will denote the KL divergence

10 From an information theory perspective $\mathbb{D}_{\mathrm{KL}}[P \parallel Q]$ is typically termed the *relative entropy of P with respect to Q* and measures the expected information loss (in *nats* for base-e logarithms or *bits* for base-2 logarithms) of using Q to model samples from P.

from P to Q in terms of the densities p and q by $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q] = \mathbb{D}_{\mathrm{KL}}[P \,\|\, Q]$, and from the definition (2.57) we have that

$$\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q] = \int_X p(x) \, \log \frac{p(x)}{q(x)} \, \mathrm{d}\mu(x), \tag{2.58}$$

with absolute continuity of P with respect to Q corresponding to a requirement that $p(x) = 0 \; \forall x \in X : q(x) = 0$. Somewhat loosely, we will refer to $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$ as the KL divergence from the (density) p to the (density) q rather than refering to the underlying measures.

When used without further qualification, variational inference is generally intended to mean inference performed by minimising a variational objective corresponding to the KL divergence from an approximate density q to the target density p. More specifically using the decomposition of the target density into an unnormalised density $\tilde{p}$ and normalising constant Z we have that

$$\mathbb{L}[q] = \log Z - \mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p] = \int_X q(\boldsymbol{x}) \, \log \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})} \, \mathrm{d}\mu(\boldsymbol{x}), \tag{2.59}$$

with $\mathbb{L}[q]$ the specific objective usually maximised in variational inference problems, with all terms in the integrand being evaluable pointwise. As $\log Z$ is constant with respect to the approximate density, maximising $\mathbb{L}$ with respect to q is directly equivalent to minimising $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$. Due to the non-negativity of the KL divergence we have that the following inequality holds

$$\mathbb{L}[q] \leq \log Z. \tag{2.60}$$

When the target density p corresponds to a posterior $\mathrm{p}_{\mathbf{x}|\mathbf{y}}$ on latent variables $\mathbf{x}$ given observed variables $\mathbf{y}$ and $\tilde{p}$ the corresponding joint density $\mathrm{p}_{\mathbf{x},\mathbf{y}}$, the normalising constant Z is equal to the model evidence term $\mathrm{p}_{\mathbf{x}}$ in Bayes' theorem. As $\mathbb{L}$ is a lower bound on $\log Z$ and so the (log) model evidence, the variational objective $\mathbb{L}$ is therefore sometimes termed the *evidence lower bound* (ELBO) in this context.

Using the KL divergence from the approximate to target density as the variational objective is not the only choice avaialable. One obvious alternative is the reversed form of the KL divergence, $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$ from the target density to the approximate density. In general as this form of the divergence involves evaluating an integral with respect to the target density, precisely the intractable computational task we are hoping to find an approximate solution, direct applications of this approach

are limited to toy problems were this integral can be solved exactly or efficiently approximated.

An approach called *expectation propagation* (EP) [144] however locally optimises an objective closely related to $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$. EP is generally applied to target distributions with a density which factorise in to a product of (often per-datapoint) factors

$$\tilde{p}(\boldsymbol{x}) = \prod_{i \in I} \tilde{p}_i(\boldsymbol{x}). \tag{2.61}$$

An approximate density is defined with an equivalent factorisation

$$q(\boldsymbol{x}) = \prod_{i \in I} q_i(\boldsymbol{x}), \tag{2.62}$$

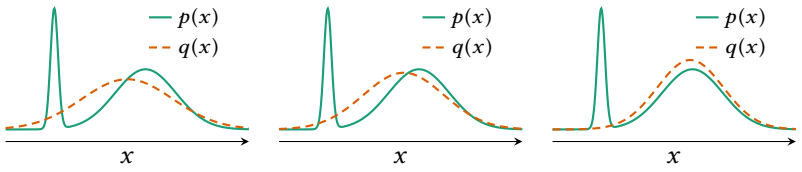
with each q_i factor restricted to be the density of an exponential family distribution. EP then fits the individual approximate factors by iteratively for each $j \in I$ minimising

$$\min_{q_j} \mathbb{D}^{\mu}_{\mathrm{KL}}\left[\tilde{p}_j(\boldsymbol{x}) \prod_{i \in I \setminus \{j\}} q_i(\boldsymbol{x}) \,\middle\|\, q_j(\boldsymbol{x}) \prod_{i \in I \setminus \{j\}} q_i(\boldsymbol{x})\right]. \tag{2.63}$$

This is similar to minimising the KL divergence from the individual target factor $\tilde{p}_j$ to the corresponding approximate factor q_j, i.e. $\mathbb{D}^{\mu}_{\mathrm{KL}}[\tilde{p}_j \,\|\, q_j]$, but (2.63) instead weights the integral by the density of the 'cavity distribution' formed by current approximation of the product of the remaining target factors. Ideally as training proceeds the cavity distribution becomes an increasingly good approximation to the product of the true remaining factors and so EP locally minimises an objective increasingly close to $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$. The additional context provided by weighting by the cavity distribution density favours approximate factors q_j which fit well to the true factor $\tilde{p}_j$ where the mass of the current global approximation is concentrated. This is usually a significant improvement over simply fitting each q_j individually by minimsing $\mathbb{D}^{\mu}_{\mathrm{KL}}[\tilde{p}_j \,\|\, q_j]$ which will often fit a very poor global approximation.

The KL divergence can be considered as a special case of a broader class of α-divergences. In particular the *Rényi divergence* [65, 188] of order $\alpha > 0, \alpha \neq 1$ between two probability measures P and Q with probability densities $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$ and $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$ on a space X is defined as

$$\mathbb{D}_{\alpha}[P \,\|\, Q] = \mathbb{D}^{\mu}_{\alpha}[p \,\|\, q] = \frac{1}{\alpha - 1} \log\left(\int_X p(\boldsymbol{x})^{\alpha} \, q(\boldsymbol{x})^{1-\alpha} \, \mathrm{d}\mu(\boldsymbol{x})\right). \tag{2.64}$$



(a) $\mathbb{D}_{\mathrm{KL}}^{\lambda}[p \,\|\, q]$ (b) $\mathbb{D}_{\alpha}^{\lambda}[p \,\|\, q],\ \alpha = \frac{1}{2}$ (c) $\mathbb{D}_{\mathrm{KL}}^{\lambda}[q \,\|\, p]$

Figure 2.13.: Comparison of approximate densities fitted under different variational objectives. Each plot shows a bimodal target density $p(x)$ and a normal approximate density $q(x) = \mathcal{N}\left(x \,|\, \mu, \sigma^2\right)$ where μ and σ have been set to values which minimise the variational objective shown in the caption.

For $\alpha > 0$, $\mathbb{D}_{\alpha}[P \,\|\, Q]$ is a valid divergence, that is $\mathbb{D}_{\alpha}[P \,\|\, Q] \geq 0$ with equality if and only if $P = Q$ almost everywhere. The definition can also be extended to the cases $\alpha = 1$ and $\alpha = 0$ by considering limits of (2.64). Using L'Hôpital's rule it can be shown that $\lim_{\alpha \to 1} \mathbb{D}_{\alpha}[P \,\|\, Q] = \mathbb{D}_{\mathrm{KL}}[P \,\|\, Q]$. For $\alpha \to 0$, we have that $\mathbb{D}_{\alpha}[P \,\|\, Q] \to -\log P(\mathrm{supp}(Q))$ where $\mathrm{supp}(Q)$ represents the support of the probability measure Q; in this case $\mathbb{D}_{\alpha}[P \,\|\, Q]$ is no longer a valid divergence as it is equal to zero whenever $\mathrm{supp}(P) = \mathrm{supp}(Q)$. It can also be shown that for $\alpha \notin \{0, 1\}$ that $\mathbb{D}_{\alpha}[P \,\|\, Q] = \frac{\alpha}{1-\alpha}\mathbb{D}_{1-\alpha}[Q \,\|\, P]$. This motivates extending the definition in (2.64) for $\alpha < 0$, in which case we have that $\mathbb{D}_{\alpha}[P \,\|\, Q] = \frac{\alpha}{1-\alpha}\mathbb{D}_{1-\alpha}[Q \,\|\, P] \leq 0$ [126].

Analogously to using the decomposition of the target density p in to an unnormalised density $\tilde{p}$ and unknown normaliser Z when defining the previous variational objective in (2.59), it is observed in [126] that a *variational Rényi bound*, $\mathbb{L}_{\alpha}$, can be defined as

$$\mathbb{L}_{\alpha}[q] = \log Z - \mathbb{D}_{\alpha}^{\mu}[q \,\|\, p] = \frac{1}{1-\alpha} \log \int_X q(\boldsymbol{x}) \left(\frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\right)^{1-\alpha} \mathrm{d}\mu(\boldsymbol{x}). \quad (2.65)$$

For $\alpha > 0$, we have that $\mathbb{D}_{\alpha}^{\mu}[q \,\|\, p] \geq 0$ and so $\mathbb{L}_{\alpha}$ is a lower bound on the $\log Z$, analogously to the ELBO, and we should maximise $\mathbb{L}_{\alpha}$ with respect to q to minimise $\mathbb{D}_{\alpha}^{\mu}[q \,\|\, p]$. For $\alpha < 0$ we have instead that $\mathbb{D}_{\alpha}^{\mu}[q \,\|\, p] \leq 0$ and so $\mathbb{L}_{\alpha}$ is an upper bound on $\log Z$ and that we should minimise $\mathbb{L}_{\alpha}$ to minimise $\mathbb{D}_{1-\alpha}^{\mu}[p \,\|\, q]$ (note the swapped order of the density arguments). An equivalent observation of the possibility of upper bounding $\log Z$ is made in [56] with a reparameterised version of (2.65) in terms of $n = 1 - \alpha > 1$.

As generally the family chosen for the approximate density q will not include the target density as a member, the choice of variational objective is important in determining the properties of how q approximates the target density [27]. The standard variational objective corresponding to $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$ strongly penalises regions in X where $\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \ll 1$, therefore the approximate densities fitted using this objective tend to be undispersed compared to the target density, and in the case of target densities with multiple separated modes fitted with a unimodal approximate density, the approximate density will tend to fit only one mode well (with fits to the different modes corresponding to different local optima in the objective). Conversely using the reversed KL divergence $\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q]$ as the variational objective penalises approximate densities where $\frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \ll 1$ in regions with significant mass under the target density, therefore the approximate densities fitted using this objective tend to be overdispersed compared to the target density, and in the case of multimodal target densities, the approximate densities will tend to 'cover' multiple modes. Using a variational objective corresponding to a Rényi divergence with $0 < \alpha < 1$, allows interpolating between these two behaviours (with α close to one favouring undispersed approximate densities similar to $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$, with the solutions becoming increasingly dispersed as α becomes lower).

Figure 2.13 gives examples of normal approximate densities fitted to a bimodal target with three variational objectives to illustrate the effect of the different objectives on the fitted approximation. In Figure 2.13a the approximate density q was fitted by minimising $\mathbb{D}^{\lambda}_{\mathrm{KL}}[p \,\|\, q]$, the resulting q putting mass on both modes in the target (and significant mass on the region of low density between the two target modes). The approximate density q in Figure 2.13c was instead fitted by minimising $\mathbb{D}^{\lambda}_{\mathrm{KL}}[q \,\|\, p]$, with the result that q concentrates its mass around one of the modes. Finally Figure 2.13b shows an approximate density fitted by minimising the Rényi divergence (2.64) with $\alpha = \frac{1}{2}$ for which $\mathbb{D}^{\lambda}_{\alpha}[p \,\|\, q] = \mathbb{D}^{\lambda}_{\alpha}[q \,\|\, p]$ and which interpolates between the behaviours of the two objectives used in Figures 2.13a and 2.13c. The approximate density here is less dispersed than in the $\mathbb{D}^{\lambda}_{\mathrm{KL}}[p \,\|\, q]$ case, but still places more mass on the minor mode than the $\mathbb{D}^{\lambda}_{\mathrm{KL}}[q \,\|\, p]$ case.

Once the variational objective has been defined, it still remains to choose the family of the approximate density q and optimisation scheme. A very common choice is to use an approximate density in the *mean-field*

variational family; this assumes that the variables the target density is defined on can be grouped in to a set of mutually independent vectors $\{\mathbf{x}_i\}_{i\in I}$ and so the approximate density can be factorised as

$$q(\boldsymbol{x}) = \prod_{i\in I} q_i(\boldsymbol{x}_i). \tag{2.66}$$

This assumption can signficantly reduce the computational demands of variational inference and facilitates simple evaluation of the approximate marginal density q_i of each variable group once fitted. However the mutual independence assumption prevents the approximate density q from being able to represent any of the dependencies between the variable groups in the target density. The early development of variational inference was largely based around mean-field family approximations [174, 205], with the naming arising from its origins in *mean-field theory*, used to study the behaviour of systems such as the Ising spin model in statistical physics [171]. Despite the limitations in representational capacity imposed by the independence assumption, because of its computational tractability variational inference using mean-field family approximate densities remains very popular [29].

The mean-field family supports a particularly simple algorithm for optimising the standard variational objective (2.59), *coordinate ascent variational inference* (CAVI) [27, 29]. If we define for each variable group vector $\boldsymbol{x}_i$ a corresponding vector $\boldsymbol{x}_{\setminus i} = [\boldsymbol{x}_j]_{j\in I\setminus i}$ concatenating all the remaining variables, then it can be shown that the optimal factors of a mean-field family approximate density satisfy

$$q_i(\boldsymbol{x}_i) \propto \exp\left(\int \prod_{j\in I\setminus i} \big(q_j(\boldsymbol{x}_j)\big) \log \tilde{p}(\boldsymbol{x}_i, \boldsymbol{x}_{\setminus i})\, \mathrm{d}\boldsymbol{x}_{\setminus i}\right). \tag{2.67}$$

The optimal value for each factor is coupled to the values of all of the other factors and so cannot be explicitly solved for even when the integral in (2.67) has an analytic solution. CAVI therefore uses a fixed point iteration approach, sequentially updating each of the factors q_i according to (2.67) given the current values of the remaining factors. This iterative update scheme is guaranteed to eventually converge to a local optimum with all factors satisfying (2.67).

The key computation in CAVI is computing the integral in (2.67) for the updates to each factor. For models with target densities where the con-

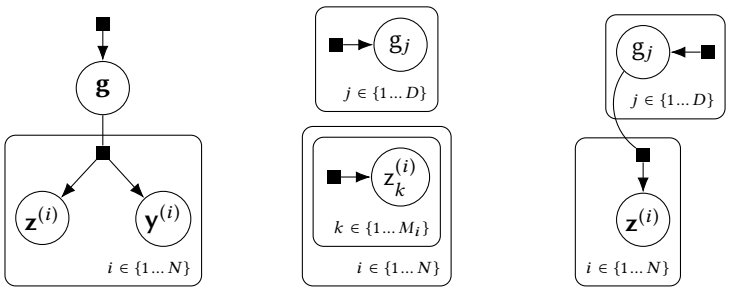
ditional densities on each variable group $\boldsymbol{x}_i$ given the remaining variables $\boldsymbol{x}_{\setminus i}$ (termed the *complete conditionals*) are all exponential family densities, an optimal parameteric form for each of the q_i factors can be analytically derived. The optimal factors have a density from the same exponential family as the corresponding complete conditional, with the integral in (2.67) having a closed form solution in this case.

For a model defined by a directed factor graph, a sufficient condition for the complete conditionals to all be exponential family densities is that all factors correspond to exponential family densities and that the factors specifying the (conditional) densities on any parent nodes to a factor are conjugate to the density on the child of the factor (in the sense of Section 1.2.5); such models are termed *conjugate exponential*.

Variational message passing [235], is a CAVI algorithm for performing inference in conjugate exponential models. It exploits factorisation structure in the target density, typically described by a directed graphical model or factor graph, to efficiently update the factors. General purpose implementations are available in software frameworks such as VIBES [28] and Infer.NET [146] which can automatically perform inference given a model specification. The conjugate exponential assumptions can be partially relaxed to also allow deterministic nodes which are multilinear functions of their parents and truncated forms of some exponential family densities which still admit analytic solutions to the factor updates [235].

A more recent alternative to CAVI for mean-field variational inference is *stochastic variational inference* (SVI) [103, 204]. SVI is designed for a common class of models consisting of a set of global latent variables $\mathbf{g}$ plus a set of local latent variables $\{\mathbf{z}^{(i)}\}_{i=1}^N$ each associated with one of N observed data points $\{\mathbf{y}^{(i)}\}_{i=1}^N$. Each pair of observed and local latent variable $(\mathbf{y}^{(i)}, \mathbf{z}^{(i)})$ are conditionally independent from all the others given the global latent variables; this factorisation is visualised in Figure 2.14c. The hierarchical model for the *Observing Dark Worlds* problem encountered earlier for example matches this structure.

CAVI requires a complete pass through all local latent variables for each update to the global latent variables. As the data set size N grows large this can become onerous computationally. Intuitively we might expect that redundancy in the data should mean that a subset of the data points should contain sufficient information to update the approximate dens-



(a) Target p (b) Mean-field q (c) Structured q

Figure 2.14.: (a) Factor graph of a model with global latent variables $\mathbf{g}$, per-datapoint local latent variables $\{\mathbf{z}^{(i)}\}_{i=1}^{N}$ and observed variables $\{\mathbf{y}^{(i)}\}_{i=1}^{N}$. (b) and (c) Factor graphs for mean-field and structured variational approximate densities for model shown in (a).

ity factors for the global variables, particularly early on in the optimisation when far from convergence and so even noisy information can allow significant improvements. The fixed-point iteration of CAVI does not however easily lend itself to exploiting this intuition.

If we instead consider using gradient ascent to maximise the objective, then the gradient of the ELBO objective with respect to the natural parameters of the global variable approximate density factors takes the form of a sum of N terms each dependent on a local latent variable and observation pair $(\mathbf{z}^{(i)}, \mathbf{y}^{(i)})$. We can form an unbiased estimate of this gradient by sampling a subset of M of the local latent variable and observation pairs (commonly $M = 1$). We can then leverage stochastic optimisation methods [190] which are designed precisely to work in this setting, of optimising an objective given a noisy but unbiased estimate of it gradient with respect to parameters.

It is assumed in SVI that the model is conjugate exponential, this meaning the gradients of the variational objective with respect to the natural parameters of the approximate density factors on both local and global latent variables can be computed in closed form. Further in this case of conjugate exponential models, the *natural gradients* [2] with respect to the variational parameters can be efficiently computed; the natural gradient exploits the differential geometry of the natural parameter space (i.e. that it is a Riemannian manifold) by rescaling the standard (Euclidean) gradient by the inverse of a Riemannian metric for the natural parameter manifold. SVI uses stochastic gradient ascent with noisy

estimates of the natural parameter natural gradients to allow efficient mean-field variational inference with large datasets.

The methods discussed so far have only applied when using an approximate density in the mean-field family. An alternative is to use a more structured factorisation which reflects some or all of the known dependencies between variables in the target distribution [9, 104, 206, 212]. These *structured variational inference* approaches use known dependency information such as from a factor graph of the target density, to inform the choice of approximate density factorisation. In general structured variational inference methods will still put some constraints on the approximate density factorisation to maintain tractability.

For example *structured stochastic variational inference* [104] applies to the same class of conjugate exponential models as SVI, i.e. with the factorisation structure shown in Figure 2.14c. It extends on SVI by allowing the approximate density to account for dependencies between the global latent variables and local latent variables, assuming a structured factorsiation corresponding to that shown in the factor graph in Figure 2.14c as opposed to the typical mean-field factorisation shown in 2.14b which would be used in standard SVI. This improves on the mean-field approximate density by including the dependencies between the local latent variables and on the global latent variables, but still requires an assumption of independence between the global latent variables.

The variational inference methods considered so far have made the strong assumption that the model being approximated is conjugate exponential. Although the analytic updates to factors made possibly by this assumption offers significant advatantages in terms of the computational tractability and stability of the optimisation of the approximate density (factors), conjugacy is a restrictive assumption which excludes many useful models. For instance the model proposed in the previous chapter for the *Observing Dark Worlds* problem is not conjugate exponential and even seemingly 'simple' models such as a logistic regression model for binary classification with a Gaussian prior on the regression weights breaks conjugacy assumptions.

Various extensions have been proposed for applying mean-field CAVI to non-conjugate exponential models where exact analytic solutions to the factor updates are no longer available. *Non-conjugate variational message passing* [114] describes an extension of the variational message

passing framework to models with non-conjugate factors, and provides a concrete algorithm for logistic regression models using model specific bounds on the factor update integrals for which analytic solutions are not available. In [230] an alternative approach with less model specific derivation is proposed. Laplace's method is used to approximate integrals with respect to non-conjugate factors, with a further option of using a Taylor series approximation to the underlying variational objective. It has also been proposed to use quadrature and tractable mixture density approximations to individual factors to approximate solutions to intractable integrals in non-conjugate factor updates [229].

A proposed unifying view of many of the mean-field methods developed for dealing with non-conjugate models, is that they relax the assumption that the approximate factors take the 'non-parameteric' optimal form given by the solution to (2.67), which is derived using a variational approach, and instead assume a fixed parameteric form for some or all of the approximate density factors [197]. This links these methods to other variational inference approaches which assume a fixed parametric form for the whole approximate density, i.e. $q(\boldsymbol{x}) = q_{\boldsymbol{\theta}}(\boldsymbol{x})$, where $q_{\boldsymbol{\theta}}$ is a density with respect to the measure μ of a fixed parametric family with a vector of parameters $\boldsymbol{\theta}$ [89, 118, 168, 184, 202].

Under this parametric assumption, rather than a variational optimisation problem we can now consider the variational objective functional $\mathbb{L}[q]$ as instead a function of the parameters $\ell(\boldsymbol{\theta}) = \mathbb{L}[q_{\boldsymbol{\theta}}]$. For the standard variational objective (2.59) we have that

$$\ell(\boldsymbol{\theta}) = \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \log \frac{\tilde{p}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \, \mu(\mathrm{d}\boldsymbol{x}). \tag{2.68}$$

Using the identities that for any $q_{\boldsymbol{\theta}}$ which is differentiable with respect to $\boldsymbol{\theta}$ we have that

$$\frac{\partial q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} = q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \tag{2.69}$$

$$\text{and} \qquad \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \, \mu(\mathrm{d}\boldsymbol{x}) = 0, \tag{2.70}$$

the gradient of (2.68) with respect $\boldsymbol{\theta}$ can be expressed as

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \log \frac{\tilde{p}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \, \mu(\mathrm{d}\boldsymbol{x}). \tag{2.71}$$

Typically both of the integrals in (2.68) and (2.71) defining the variational objective and its gradient will not have analytic solutions. However both take the forms of expectations of a random vector with distribution defined by the approximate density $q_{\boldsymbol{\theta}}$. If we can generate independent samples from $q_{\boldsymbol{\theta}}$ we can therefore form unbiased Monte Carlo estimates of the objective and its gradient.

The unbiased gradient estimates can then be used in a stochastic gradient ascent method [190] to maximise $\ell(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. This basic framework is applicable to a much broader class of target distributions than the previously discussed variational inference approaches, requiring only that we can pointwise evaluate a, potentially unnormalised, density function $\tilde{p}$ for the target distribution. Likewise the only restrictions on the approximating distribtion are that we can evaluate a density function $q_{\boldsymbol{\theta}}$ which is differentiable with respect to its parameters $\boldsymbol{\theta}$ and that we can generate independent samples from this distribution to form the Monte Carlo estimates.

For target distributions on a real-valued space, a simple choice for $q_{\boldsymbol{\theta}}$ meeting these requirements is a multivariate normal density $q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma)$ with the mean $\boldsymbol{\mu}$ and covariance Σ forming the parameters $\boldsymbol{\theta}$ maximised with respect to. Using a diagonal covariance Σ would correspond to a mean-field factorisation assumption for the approximate density, however we can also use more general covariances including a full dense matrix allowing for arbitrary covariance structure in the approximate density, or a sparse covariance matrix which exploits known conditional independencies in the target distribution.

Although appealingly simple and flexible, the basic scheme as described so far has a major pitfall which is that the variance of the gradients estimate computed by forming the obvious Monte Carlo estimator from (2.71) typically has a very high variance for the complex target distributions of interest. This necessitates either taking a very large number of Monte Carlo samples to estimate the gradient for each parameter update with sufficient accuracy or taking very small gradient steps to allow stable optimisation. Therefore practical schemes based on this idea generally require the use of variance reduction methods to estimate the variational objective parameter gradient.

The *black box variational inference* (BBVI) algorithm of [184] proposes using two forms of variance reduction to compute more efficient gradi-

ent estimtates - Rao-Blackwellisation and control variates. The Rao-Blackwellisation method relies on being able to decompose the approximate density $q_{\boldsymbol{\theta}}$ into a product of factors with per factor variational parameters and is so restricted to cases such as mean-field approximations where this is the case. The control variate method is more general and can be applied to non mean-field approximate densities.

An alternative variance reduction approach is proposed in the *automatic differentiation variational inference* (ADVI) algorithm of [118] which instead uses a reparameterisation of the approximating distribution to produce a lower variance gradient estimator for a more restricted class of target distributions which have a differentiable density with respect to the Lebesgue measure. It is assumed that the samples from the approximating distribution can be generating using a transform sampling method, more specifically that there exists a differentiable bijective function $\mathrm{g}_{\boldsymbol{\theta}} : U \to X$ and a distribution R on U which has a density ρ which does not depend on $\boldsymbol{\theta}$ such that

$$q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \rho\Big(\mathrm{g}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x})\Big)\left|\frac{\partial \mathrm{g}_{\boldsymbol{\theta}}^{-1}}{\partial \boldsymbol{x}}\right|. \tag{2.72}$$

In this case by the change of variables formula (1.22) discussed in Chapter 1 if $\boldsymbol{u}$ is an independent sample from R then $\boldsymbol{x} = \mathrm{g}_{\boldsymbol{\theta}}(\boldsymbol{u})$ will be an independent sample from the approximate distribution with density $q_{\boldsymbol{\theta}}$. This transformation can be used to reparameterise the variational objective integral as

$$\ell(\boldsymbol{\theta}) = \int_U \rho(\boldsymbol{u})\Big(\log(\tilde{p} \circ \mathrm{g}_{\boldsymbol{\theta}}(\boldsymbol{u})) + \log\left|\frac{\partial \mathrm{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{u}}\right| - \log \rho(\boldsymbol{u})\Big)\,\mathrm{d}\boldsymbol{u} \tag{2.73}$$

with a corresponding gradient expression

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \int_U \rho(\boldsymbol{u})\left(\frac{\partial}{\partial \boldsymbol{\theta}} \log(\tilde{p} \circ \mathrm{g}_{\boldsymbol{\theta}}(\boldsymbol{u})) + \frac{\partial}{\partial \boldsymbol{\theta}} \log\left|\frac{\partial \mathrm{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{u}}\right|\right)\mathrm{d}\boldsymbol{u}. \tag{2.74}$$

As suggested by the name ADVI uses automatic differentiation to calculate the gradient expression inside the parentheses in (??), with importantly this requiring propagation of derivatives through the target density function $\tilde{p}$ as well as the transformation $\mathrm{g}_{\boldsymbol{\theta}}$. To form a Monte Carlo estimate of the gradient using this reparameterisation we therefore require the target model density to be differentiable and so it is less general than the method used in BBVI, however as shown empiric-

ally in [118] the resulting gradient estimator will tend to be signficantly more efficient requiring fewer samples to bring the variance to a reasonable level, with often gradients computed with a single sample being sufficient for stable optimisation.

2.3 DISCUSSION

The approximate inference methods we have reviewed in this chapter provide

3 PSEUDO-MARGINAL METHODS

The MCMC methods considered in Chapter 2 provide a widely applicable set of tools for performing inference in probabilistic models where we can evaluate a, potentially unnormalised, density of the target distribution of interest. In some models we may not be able to directly evaluate such a function however but instead have access to an unbiased estimator of the target density. The *pseudo-marginal* MCMC framework [5] allows MCMC methods to be extended to such problems.

The typical setting for pseudo-marginal methods is that a distribution on an extended set of variables is constructed which has the target distribution as a marginal. Values of a density function for the target distribution are then estimated by using a Monte Carlo method such as importance sampling to approximately marginalise out the additional variables. The variables which are marginalised out may correspond to latent variables specified in the model but that are not of direct interest for the inference task or variables introduced solely for computational reasons. In both cases it will usually be possible to specify a Markov transition operator which leaves the distribution on the extended set of variables invariant, with such schemes often being described as *data augmentation* [216, 228] or *auxiliary variable* [63, 101] methods. Here we will refer to any variables which are marginalised over as auxiliary variables and the variables of interest we wish to infer plausible values for as the target variables.

The density of the joint distribution on auxiliary and target variables will often have a complex geometry with strong dependencies between the variables and potentially multimodality, i.e. multiple separated regions of high probability density. This can lead to poor exploration of the extended space by simple MCMC schemes such as random-walk Metropolis–Hastings and Gibbs sampling [5]. The motivation for pseudo-marginal methods is that in some cases the density of the marginal distribution on the target variables will have a simpler geometry than the density of the joint distribution on the extended space and therefore be more amenable to exploration by standard MCMC methods.

Although in general we cannot analytically integrate out the auxiliary variables, the pseudo-marginal framework shows how an unbiased estimator of the marginal density can be used within a Metropolis–Hastings update while maintaining the asymptoptic exactness of standard MCMC methods. Intuitively the lower the variance of the density estimator the closer the behaviour of the algorithm to the case where the auxiliary variables are analytically marginalised out. We can control the variance of the estimator both by varying the number of auxiliary variable samples used in the Monte Carlo estimate and by using variance reduction methods to increase the estimator efficiency.

By posing the problem of specifying an MCMC algorithm in terms of designing an efficient[1] unbiased estimator of the density of interest, the large literature on methods for constructing low-variance unbiased estimators can be exploited. For example comparatively cheap but biased optimisation-based inference approaches such as Laplace's method can be combined with an importance sampling 'debiasing' step to produce an unbiased estimator which can then be used in a pseudo-marginal MCMC update. This provides a way of exploiting cheap but biased approximate inference methods within a MCMC method which still gives guarantees of asymptotically exact results.

The pseudo-marginal framework has been applied to a wide range of probabilistic models were inference might otherwise be intractable. However the standard pseudo-marginal method, which is based on a Metropolis–Hastings transition operator, is suspectible to 'sticking' behaviour where proposed moves are repeatedly rejected for many iterations [5, 208]. The method can also be difficult to tune as it breaks some of the assumptions underlying standard heuristics for adapting the parameters of Metropolis–Hastings methods.

In this chapter we will discuss an alternative formulation of the pseudo-marginal framework which bridges between the approach of directly specifying a Markov transition operator on the extended state space which includes the auxiliary variables and the pseudo-marginal method where the auxiliary variables are approximately marginalised out. This *auxiliary pseudo-marginal* framework still allows the intuitive design of pseudo-marginal algorithms in terms of identifying low-variance unbiased estimators, while overcoming some of the issues of the pseudo-

1 We use 'efficient' in a general sense here rather than the notion of a minimum-variance unbiased estimator satisfying the Cramér-Rao lower bound [48, 185].

marginal Metropolis–Hastings method. In particular it shows how more flexible adaptive MCMC algorithms such as slice-sampling can be used within the pseudo-marginal setting, which can improve the robustness and ease of application of the approach by minimising the amount of user-tuning of free parameters required.

The work summarised in this chapter is based on a collaboration with Iain Murray which resulted in the published conference paper

- Pseudo-marginal slice sampling. Iain Murray and Matthew M. Graham. *The Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, JMLR W&CP 51:911-919*, 2016.

Iain Murray was the main contributor of the ideas proposed in that publication and responsible for the 'doubly-intractable' Gaussian and Ising model experiments in Sections 5.1 and 5.2 of the paper. My contribution was implementing and analysing the Gaussian process classification experiments summarised in Section 5.3 of that work, an extended version of which is reproduced in Section 3.6.2 of this Chapter. The Gaussian latent variable model experiments discussed in Section 3.6.1 were directly inspired by the Gaussian model experiments in Section 5.1 of the above paper, but we use a different latent variable model formulation for the model here and conduct additional empirical studies of the effect of the variance of the estimator on the relative performance of the algorithms and the sensitivity of the performance of the pseudo-marginal slice sampling algorithms to their free parameters. The text and figures in this chapter are all my own work, though inevitably some of the discussion and analysis is similar to sections of the *Pseudo-marginal slice sampling* publication.

3.1 PROBLEM DEFINITION

As in the previous chapter our goal is to be able to compute estimates of expectations with respect to a *target distribution* of interest, that is integrals of the form

$$\bar{f} = \int_X f(\boldsymbol{x})\, P(\mathrm{d}\boldsymbol{x}) = \int_X f(\boldsymbol{x})\, p(\boldsymbol{x})\, \mu(\mathrm{d}\boldsymbol{x}) \tag{3.1}$$

where $f : X \to \mathbb{R}$ is an arbitrary Lebesgue integrable function and P is a probability distribution on a space X with density $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$. We as-

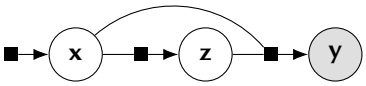


Figure 3.1.: Hierarchical model factor graph.

sume as previously that density p may have an intractable normalising constant C that we cannot evaluate i.e. $p(\boldsymbol{x}) = \tilde{p}(\boldsymbol{x})/C$. We make the further assumption here that we cannot directly evaluate $\tilde{p}$ either but only compute an unbiased, non-negative estimate of it. More explicitly we assume we can generate values of a non-negative random variable $\hat{\mathrm{p}}$ from a regular conditional distribution $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}$ such that

$$\tilde{p}(\boldsymbol{x}) = \mathbb{E}\left[\hat{\mathrm{p}} \mid \mathbf{x} = \boldsymbol{x}\right] = \int_0^\infty \hat{p}\, \mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}(\mathrm{d}\hat{p} \mid \boldsymbol{x}) \quad \forall \boldsymbol{x} \in X. \tag{3.2}$$

Note that we only require that we can generate independent $\hat{\mathrm{p}}$ values for a given $\mathbf{x}$, not that we can evaluate a density function for $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}$. For concreteness throughout the rest of this chapter we will assume that the target variables take values in a real-valued space $X = \mathbb{R}^D$ and that any density on these variables is defined with respect to the Lebesgue measure $\mu = \lambda^D$.

3.1.1 Example: hierarchical latent variable models

A common application of pseudo-marginal methodsd is inference in hierarchical probabilistic models where the unobserved variables are split into global latent variables we are interested in inferring and local (per datapoint) latent variables that we wish to marginalise over the values of. For notational simplicity we assume all observed variables are concatenated in a single vector $\mathbf{y}$ and likewise all associated local latent variables in a vector $\mathbf{z}$. The global latent variables, i.e. the target variables for inference, are then $\mathbf{x}$. A factor graph representing the factorisation across the model variables is shown in Figure 3.1.

The target distribution P is then the posterior distribution $\mathrm{P}_{\mathbf{x}|\mathbf{y}}$ given fixed observed values $\boldsymbol{y}$ and a corresponding unnormalised target density can be chosen as the joint density $\tilde{p}(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y})$. We can express $\tilde{p}$ as a marginal of the joint density $\mathrm{p}_{\mathbf{x},\mathbf{z},\mathbf{y}}$, which assuming the latent variables $\mathbf{z}$ being marginalised over are real-valued and have a density with respect to the Lebesgue measure can be written

$$\tilde{p}(\boldsymbol{x}) = \int_Z \mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{y}, z)\, \mathrm{d}z. \tag{3.3}$$

Generally this integral will not have an analytic solution. We can however form an unbiased estimate of (3.3) using importance sampling. We define a *importance distribution* Q which we can generate independent samples from and with a known density q which in general may depend on the values of the target variables $\boldsymbol{x}$ and observations $\boldsymbol{y}$. If $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$ are a set of independent variables distributed according to Q then we can define a unbiased density estimator $\hat{\mathrm{p}}$ as

$$\hat{\mathrm{p}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\boldsymbol{y},\mathbf{z}^{(n)})}{q(\mathbf{z}^{(n)} \mid \mathbf{x},\boldsymbol{y})} \implies \mathbb{E}[\hat{\mathrm{p}} \mid \mathbf{x} = \boldsymbol{x}] = \tilde{p}(\boldsymbol{x}). \tag{3.4}$$

The variance $\mathbb{V}[\hat{\mathrm{p}}]$ is proportional to $\frac{1}{N}$ and so one method of decreasing the estimator variance is to increase the number of importance samples used, however this comes with the obvious tradeoff of an increased computational cost of each density estimate evaluation. The estimator variance will also be strongly dependent on the importance distribution used. The optimal choice in terms of minimsing variance would be the conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$. Under this choice the density 'estimate' takes the form

$$\hat{\mathrm{p}} = \frac{1}{N}\sum_{n=1}^{N}\frac{\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\boldsymbol{y},\mathbf{z}^{(n)})}{\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\mathbf{z}^{(n)} \mid \mathbf{x},\boldsymbol{y})} = \frac{1}{N}\sum_{n=1}^{N}\mathrm{p}_{\mathbf{x},\mathbf{y}}(\mathbf{x},\boldsymbol{y}) = \tilde{p}(\mathbf{x}) \tag{3.5}$$

and so is equal to the unnormalised target density independent of the sampled $\mathbf{z}^{(n)}$ values with zero variance. In reality however we will not be able to evaluate the density of $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ nor sample from it as this is equivalent to being able to analytically solve the integral (3.3).

The conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ will often be tractable to sample from and to evaluate the density of and so is a possible choice for the importance distribution. Typically however $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ will be much less concentrated than $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$. This will mean samples from $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ will tend to fall in low density regions of $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$, with only occassionally sampled values being in regions with high density under $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ leading to a high variance estimator, with the problem becoming more severe as the dimension of $\mathbf{z}$ increases. This can mean a large number of importance samples are needed to achieve an estimator with a reasonable variance.

An alternative is to fit an approximation to $\mathrm{P}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ to use as the importance distribution using for example one of the optimisation-based approximate inference approaches discussed in Chapter 2. For example

we could use Laplace's method to fit a multivariate normal approximation $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{z} \,|\, \boldsymbol{x}, \boldsymbol{y}) \approx \mathcal{N}\left(\boldsymbol{z} \,|\, \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ and use this as the importance distribution. As $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ depends on $\boldsymbol{x}$ this involves fitting an approximation for each $\boldsymbol{x}$ value we wish to evaluate the density at. Although computationally costly the significant variance reduction brought by this approach can make this overhead worthwhile in pratice [67].

Inference in hierarchical latent variable models using an importance sampling estimator for the marginal density is just one setting in which pseudo-marginal methods are applied. Other applications of the framework have included inference methods for dynamical state space models using a particle filter estimator [58, 87] for the marginal density of the observed state sequence given the model parameters (target variables) [4, 44, 175], parameter inference in 'doubly-intractable' distributions [156] where an intractable normaliser depends on the variables of interest using density estimators based on exact sampling methods [149, 152, 182] and random series trunctation [129] and approximate inference in simulator models where the density on the simulator outputs is only implicitly defined [133].

In the discussion and experiments in this chapter we will concentrate on latent variable models and importance sampling density estimators of the form described in this section. Examples of applying the methods discussed here to inference in a doubly intractable distribution were discussed in the associated conference paper [157]. Although particle filtering based methods are a major use case of the pseudo-marginal framework, the associated models and estimators tend to be more complex and we have chosen to avoid further expanding the theoretical background material in this thesis by concentrating on simpler cases here. The use of pseudo-marginal MCMC methods to perform inference in simulator models will be a major topic of the next chapter which specifically considers inference methods applicable in this setting so we will delay discussion of models of this form till then.

3.2 PSEUDO-MARGINAL METROPOLIS–HASTINGS

The pseudo-marginal Metropolis–Hastings method is summarised in Algorithm 6. The term *pseudo-marginal* was proposed by Andrieu and Roberts in [5], with they also giving an extensive theoretical analysis of the framework. Andrieu and Roberts cite Beaumont [17] as the original

Algorithm 6 Pseudo-marginal Metropolis–Hastings.

Input: $(\boldsymbol{x}_n, \hat{p}_n)$: current target variables – density estimate state pair, $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}$: density estimate conditional distribution, r : proposal density for updates to target variables.

Output: $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1})$: new target variables – density estimate state pair.

$\boldsymbol{x}^* \sim r(\cdot \mid \boldsymbol{x}_n)$ ▷ Propose new values for target variables.
$\hat{p}^* \sim \mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}(\cdot \mid \boldsymbol{x}^*)$ ▷ Estimate density at proposed $\boldsymbol{x}^*$.
$u \sim \mathcal{U}(\cdot \mid 0, 1)$
if $u < \frac{r(\boldsymbol{x}_n \mid \boldsymbol{x}^*)\,\hat{p}^*}{r(\boldsymbol{x}^* \mid \boldsymbol{x}_n)\,\hat{p}_n}$ **then**
    $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1}) \leftarrow (\boldsymbol{x}^*, \hat{p}^*)$ ▷ Accept proposal.
else
    $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1}) \leftarrow (\boldsymbol{x}_n, \hat{p}_n)$ ▷ Reject proposal.
return $(\boldsymbol{x}_{n+1}, \hat{p}_{n+1})$

source of the algorithm. Special cases of the algorithm have also been independently proposed, for example in the statistical physics literature by Kennedy and Kuti [110] and a MCMC method for doubly intractable distributions by Moller et al. [149].

The algorithm takes an intuitive form, with a very similar structure to the standard Metropolis–Hastings method (Algorithm 2) except for the ratio of densities in the accept probability calculation being replaced with the ratio of the density estimates. Importantly the stochastic density estimates are maintained as part of the chain state: if we reject a proposed update on the next iteration of the algorithm we reuse the same density estimate for the current state as in the previous iteration. This is required for the correctness of the algorithm, but also helps explain the sticking behaviour sometimes encountered with pseudo-marginal Metropolis–Hastings chains. If the density estimator distribution is heavy-tailed occassionally a estimate $\hat{p}_n$ will be sampled for the current target state $\boldsymbol{x}_n$ which is much higher than the expected value $\tilde{p}(\boldsymbol{x}_n)$. Assuming for simplicity a symmetric proposal density r is used such that the accept probability ratio in Algorithm 6 reduces to $\hat{p}^*/\hat{p}_n$, for subsequent proposed $(\boldsymbol{x}^*, \hat{p}^*)$ pairs the $\hat{p}^*$ values will typically be much smaller than the outlier $\hat{p}_n$ value and so the accept probability low. This can cause a long sequence of proposed moves being rejected until a move is proposed to an $\boldsymbol{x}^*$ where the density is similar to $\hat{p}_n$ or another atypically high density estimate is proposed [5, 67, 208].

The efficiency of the pseudo-marginal Metropolis–Hastings update depends on how noisy the density estimates are and so the choice of the number of Monte Carlo samples N in the density estimate, for example

the number of importance samples in (3.4). As N increase, the variance decreases and the algorithm becomes increasingly similar to performing standard Metropolis–Hastings updates under the (marginal) target distribution. Generally a chain will therefore mix better for larger N, with fewer sticking events. Typically however the computational cost of density estimate and so Metropolis–Hastings updates also increases with N and so there is a tradeoff between this improved mixing and increased per-update cost. Several theoretical studies have suggested guidelines for how to tune the parameters of the algorithm to optimise this tradeoff.

For estimators formed as a Monte Carlo average of unbiased estimators (such as the importance sampling estimator discussed above) and under an assumption of that the computational cost of each density estimate scales linearly with the number of Monte Carlo samples N, it has been shown [33, 207] that it is close to optimal to choose $N = 1$. Although the variance reduction in the density estimates for larger N generally gives higher acceptance rates and improved mixing, the gain in the number effective samples in this case is usually smaller than the increased computational cost per update.

As noted in [207] in many practical settings cases the assumption of a linear increase in cost with the number of importance samples N will not be valid, particularly for small N. For example most modern CPUs have some degree of parallel compute capability through multiple cores so (assuming the parallelism can be exploited) there will usually be a non-linear increase in cost until all cores are at full utilisation: a rough guideline in this case is to use one sample per core. Another situation in which the linear cost assumption may not hold is when there is a high fixed computational overhead in each density estimate independent of the number of samples. For example if a importance distribution is used which is dependent on the target variables there may be computational operations such as matrix decompositions that can be performed once and then their cost amortised over generation of multiple importance samples. In an empirical investigation in [208] it is found in a Gaussian process inference problem with this property that the optimal computational efficiency was achieved roughly when the marginal cost of generating N importance samples was roughly equal to the fixed overhead.

Particle filtering estimators do not take the form of a simple Monte Carlo average of independent unbiased estimates but are instead are formed as a product of (dependent) Monte Carlo estimates [207]. The result of [207] that using $N = 1$ is close to optimal (with N now representing the number of particles) is therefore not applicable in this case. Under an alternative simplifying assumption that the noise in the logarithm of the density estimator is normally distributed and independent of the value of the target variables $\boldsymbol{x}$ and that the computational cost of each density estimate scales linearly with N, it is argued in [59] that N should be chosen so as to make the standard deviation of the logarithm of the density estimator equal to approximately 1.2. For the case of pseudo-marginal Metropolis–Hastings methods using a isotropic normal random-walk Metropolis proposal density $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$ and the same assumptions of additive normal noise in the logarithm of the density estimator which is independent of $\boldsymbol{x}$ and a computational cost for each density estimate which scales linearly with N, it is shown in [208] that for target distributions on a D dimensional space which obey certain regularity assumptions as $D \to \infty$ that computational efficiency is maximised for a choice of λ and N which gives an average accept rate of approximately 0.07 and a noise standard deviation for the logarithm of the density estimator of approximately 1.8.

3.3 REPARAMETRISING THE DENSITY ESTIMATOR

As a first step in considering how to apply alternative transition operators to pseudo-marginal inference problems, we define a reparameterisation of the density estimator in terms of a deterministic function of the auxiliary random variables used in computing the estimate. An equivalent reparameterisation has also been used in other work analysing the pseudo-marginal framework, for example [59].

In general the computation of a density estimate will involve sampling values from known distributions using a pseudo-random number generator and then applying a series of deterministic operations to these auxiliary random variables. Under the simplifying assumption that the estimator uses a fixed number of auxiliary random variables, we can therefore define a non-negative deterministic function $\varepsilon : X \times U \to [0, \infty)$ and a distribution R with known density $\rho = \frac{\partial R}{\partial \nu}$ such that if

$\boldsymbol{u}$ is an independent sample from R, then $\hat{p} = \varepsilon(\boldsymbol{x}, \boldsymbol{u})$ is an independent sample from $\mathsf{P}_{\hat{\mathsf{p}}|\mathbf{x}}$. Here R represents the known distribution of the auxiliary variables and ϵ the operations performed by the remaining estimator code given values for the target and auxiliary variables. We can use this to reparameterise (3.2) as

$$\tilde{p}(\boldsymbol{x}) = \int_U \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, R(\mathrm{d}\boldsymbol{u}) = \int_U \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, \rho(\boldsymbol{u})\, \nu(\mathrm{d}\boldsymbol{u}) \quad \forall \boldsymbol{x} \in X. \tag{3.6}$$

For example considering the importance-sampling density estimator for a hierachical latent variable model defined in (3.4), if we assume the importance distribution is chosen to be a multivariate normal with density $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ then defining $\boldsymbol{u} = \left[\boldsymbol{u}^{(1)}; \cdots; \boldsymbol{u}^{(n)}\right]$ as the concatenated vector of standard normal variables used to generate the importance distribution samples, we have $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \boldsymbol{I})$ and

$$\varepsilon(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{N} \sum_{n=1}^{N} \frac{\mathsf{p}_{\mathbf{x},\mathsf{y},\mathsf{z}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{u}^{(n)} + \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}})}{\mathcal{N}\left(\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{u}^{(n)} + \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}} \mid \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)}, \tag{3.7}$$

where $\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}}$ is the lower triangular Cholesky factor of $\boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}$.

Rather than defining the chain state in the pseudo-marginal Metropolis–Hastings update as the target state – density estimate pair $(\boldsymbol{x}, \hat{p})$, we can instead replace the density estimate $\hat{p}$ with the auxiliary random variables $\boldsymbol{u}$ drawn from R used to compute the estimate. As $\hat{p}$ is a deterministic function of $\boldsymbol{x}$ and $\boldsymbol{u}$ these two parameterisations are equivalent. The implementation in Algorithm 6 can be considered a practically motivated variant that avoids the $\boldsymbol{u}$ values needing to be stored in memory and in fact means they do not need to be explicitly defined in the algorithm at all.

While the formulation of the update in Algorithm 6 is the more useful for implementation purposes, showing the correctness of the update is simpler when considering the chain state as $(\boldsymbol{x}, \boldsymbol{u})$. We will briefly go through this derivation now as it provides some useful insights in to the pseudo-marginal Metropolis–Hastings algorithm that will help motivate our alternative proposed approaches.

From (3.6) we known that a distribution on $X \times U$ with density

$$\pi(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{C}\, \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, \rho(\boldsymbol{u}) \tag{3.8}$$

will have the target distribution on X as its marginal distribution. Showing that the transition operator defined by Algorithm 6 leaves a distribution with density corresponding to (3.8) invariant is therefore sufficient for ensuring the correctness of the algorithm.

The transition operator corresponding to Algorithm 6 has a density

$$\mathrm{t}(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u}) + \delta(\boldsymbol{x} - \boldsymbol{x}')\,\delta(\boldsymbol{u} - \boldsymbol{u}') \left(1 - \int_U \int_X r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u})\,\mu(\mathrm{d}\boldsymbol{x})\nu(\mathrm{d}\boldsymbol{u})\right),$$

with the accept probability α being defined here as

$$\alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u}) = \min\left\{1, \frac{r(\boldsymbol{x} \mid \boldsymbol{x}')\varepsilon(\boldsymbol{x}', \boldsymbol{u}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})\varepsilon(\boldsymbol{x}, \boldsymbol{u})}\right\}. \tag{3.9}$$

As in Chapter 2 it is sufficient to show the non self-transition term in this transition density satisfies detailed balance with respect to the target density (3.8) as self-transitions leave any distribution invariant. We have that for $\boldsymbol{x} \neq \boldsymbol{x}'$, $\boldsymbol{u} \neq \boldsymbol{u}'$

$$\begin{aligned}
&\mathrm{t}(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u})\,\pi(\boldsymbol{x}, \boldsymbol{u}) \\
&\quad = \frac{1}{C}\, r(\boldsymbol{x}' \mid \boldsymbol{x})\, \rho(\boldsymbol{u}')\, \alpha(\boldsymbol{x}', \boldsymbol{u}' \mid \boldsymbol{x}, \boldsymbol{u})\, \varepsilon(\boldsymbol{x}, \boldsymbol{u})\, \rho(\boldsymbol{u}) \\
&\quad = \frac{1}{C}\, \rho(\boldsymbol{u}')\, \rho(\boldsymbol{u})\, \min\{r(\boldsymbol{x}' \mid \boldsymbol{x})\, \varepsilon(\boldsymbol{x}, \boldsymbol{u}), r(\boldsymbol{x} \mid \boldsymbol{x}')\, \varepsilon(\boldsymbol{x}', \boldsymbol{u}')\} \\
&\quad = \frac{1}{C}\, r(\boldsymbol{x} \mid \boldsymbol{x}')\, \rho(\boldsymbol{u})\, \alpha(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{x}', \boldsymbol{u}')\, \varepsilon(\boldsymbol{x}', \boldsymbol{u}')\, \rho(\boldsymbol{u}') \\
&\quad = \mathrm{t}(\boldsymbol{x}, \boldsymbol{u} \mid \boldsymbol{x}', \boldsymbol{u}')\, \pi(\boldsymbol{x}', \boldsymbol{u}'),
\end{aligned} \tag{3.10}$$

and so the transition operator corresponding to Algorithm 6 leaves the target distribution invariant.

We can equivalently consider Algorithm 6 as a standard Metropolis–Hastings transition operator on a target distribution with density (3.8) using a proposal $r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')$ i.e. perturbatively updating the $\boldsymbol{x}$ values and indepedently resampling the $\boldsymbol{u}$ values. Substituting this proposal density and target density into the standard Metropolis–Hastings accept ratio recovers the form used in the pseudo-marginal variant,

$$\frac{r(\boldsymbol{x} \mid \boldsymbol{x}')\rho(\boldsymbol{u})\frac{1}{C}\,\varepsilon(\boldsymbol{x}', \boldsymbol{u}')\,\rho(\boldsymbol{u}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})\rho(\boldsymbol{u}')\frac{1}{C}\,\varepsilon(\boldsymbol{x}, \boldsymbol{u})\,\rho(\boldsymbol{u})} = \frac{r(\boldsymbol{x} \mid \boldsymbol{x}')\varepsilon(\boldsymbol{x}', \boldsymbol{u}')}{r(\boldsymbol{x}' \mid \boldsymbol{x})\varepsilon(\boldsymbol{x}, \boldsymbol{u})}. \tag{3.11}$$

Algorithm 7 Auxiliary pseudo-marginal framework.

Input: $(\boldsymbol{x}_n, \boldsymbol{u}_n)$: current target variables – auxiliary variables pair, T_1 : transition operator updating only auxiliary variables $\boldsymbol{u}$ and leaving distribution with density in (3.8) invariant, T_2 : transition operator updating only target variables $\boldsymbol{x}$ and leaving distribution with density in (3.8) invariant.

Output: $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$: new target state – auxiliary variables pair.

$\boldsymbol{u}_{n+1} \sim \mathsf{T}_1(\cdot \mid \boldsymbol{x}_n, \boldsymbol{u}_n)$
$\boldsymbol{x}_{n+1} \sim \mathsf{T}_2(\cdot \mid \boldsymbol{x}_n, \boldsymbol{u}_{n+1})$
return $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$

Algorithm 8 Auxiliary pseudo-marginal MI + MH.

Input: $(\boldsymbol{x}_n, \boldsymbol{u}_n)$: current target – auxiliary variables state pair, ε : estimator function for density of target distribution, ρ : density of estimator's auxiliary variable distribution, r : proposal density for updates to target state.

Output: $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$: new target – auxiliary variables state pair.

$\boldsymbol{u}^* \sim \rho(\cdot)$ ▷ T_1: MI update to auxiliary variables.
$v \sim \mathcal{U}(\cdot \mid 0, 1)$
if $v < \frac{\varepsilon(\boldsymbol{x}_n, \boldsymbol{u}^*)}{\varepsilon(\boldsymbol{x}_n, \boldsymbol{u}_n)}$ **then**
    $\boldsymbol{u}_{n+1} \leftarrow \boldsymbol{u}^*$
else
    $\boldsymbol{u}_{n+1} \leftarrow \boldsymbol{u}_n$
$\boldsymbol{x}^* \sim r(\cdot \mid \boldsymbol{x}_n)$ ▷ T_2: MH update to target variables.
$w \sim \mathcal{U}(\cdot \mid 0, 1)$
if $w < \frac{r(\boldsymbol{x}_n \mid \boldsymbol{x}^*)\, \varepsilon(\boldsymbol{x}^*, \boldsymbol{u}_{n+1})}{r(\boldsymbol{x}^* \mid \boldsymbol{x}_n)\, \varepsilon(\boldsymbol{x}_n, \boldsymbol{u}_{n+1})}$ **then**
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}^*$
else
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$
return $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$

This formulation highlights a potential source of some of the computational issues with the pseudo-marginal Metropolis–Hastings algorithm. In high-dimensional spaces generally we would expect independent resampling of a subset of the variables in a Markov chain state from their marginal distribution for a proposed Metropolis–Hastings move to perform poorly [158]. Unless the variables being independently resampled have little or no dependency on the rest of the chain state, the marginal distribution will be significantly different from the conditional distribution given the remaining variables and proposed values from the marginal will be often be highly atypical under the conditional and so have a low probability of acceptance.

3.4 AUXILIARY PSEUDO-MARGINAL METHODS

The observation that the pseudo-marginal Metropolis–Hastings update just corresponds to a special case of the standard Metropolis–Hastings

algorithm with independent proposed updates to the auxiliary random variables suggests the possibility of using alternative transition operators within a pseudo-marginal context. A particularly simple framework is to alternate updates to the target state $\boldsymbol{x}$ given the auxiliary variables $\boldsymbol{u}$ and to the auxiliary variables $\boldsymbol{u}$ given the target state $\boldsymbol{x}$. We refer to this scheme as the *auxiliary pseudo-marginal* (APM) framework and summarise it in Algorithm 7.

A simple example of an APM method is formed by alternating *Metropolis independence* (MI) updates to the auxiliary variables given the target variables using R as the proposal distribution with *Metropolis–Hastings* (MH) updates to the target variables given the current auxiliary variables; this variant is described in Algorithm 8. Following the convention of [157] we name this method APM MI+MH for short and will in general use the form APM [T1]+[T2] to name APM methods where [T1] and [T2] are abbreviations for the types of the transition operators T_1 and T_2 respectively.

The APM MI+MH method retains the black-box nature of the original pseudo-marginal MH algorithm by requiring no explicit knowledge of the auxiliary random variables used in the density estimate providing we can read and write the internal state of the PRNG used by the estimator. This can be achieved for example using the `.Random.seed` attribute in `R` and the `get_state` and `set_state` methods of a `NumPy` `RandomState` object. We then only need to store the PRNG state associated with each target density estimator evaluation and restore a previous state if we wish to estimate the density at a new target state with the same set of auxiliary variables as used for a previous evaluation.

Any *pseudo-marginal* (PM) MH implementation can easily be converted in to a APM MI+MH method as the two algorithms require exactly the same input objects with the APM MI+MH method simply splitting the original single MH step into two separate propose-accept steps. The APM MI+MH method introduces some overhead by requiring two new evaluations of the target density estimator per overall update (once for the new proposed auxiliary variables and once for the new proposed target variables) compared to the single evaluation required for the PM MH algorithm.

Importantly however the updates to the target variables in APM MI+MH take the form of a standard perturbative MH update. If we use a random-

walk Metropolis update then this means we can automatically tune the step size of the updates by for example appealing to theoretical results suggesting tuning the step size to achieve an average acceptance rate of 0.234 is optimal (in terms of maximising the number of effective samples per computation time) when making perturbative moves in high-dimensions [73]. The tuning can either be done in an initial warm-up phase of the chain with the samples from this initial phase not included in the final Monte Carlo estimates or by using online approaches which use vanishing adaptation [6, 90].

As discussed earlier for particle filtering estimators, under certain simplifying assumptions an alternative average acceptance rate of 0.07 has shown to be optimal for PM MH with a isotropic normal random-walk proposal in high-dimensional target distributions [208]. While this does provide a target for tuning the step-size of a standard PM MH update in the cases where it is relevant, the APM MI+MH update may often be easier to tune in practice. The 0.07 target accept rate is predicated on the variance of the density estimator having been tuned, via the number of Monte Carlo samples, such that log density estimates have a standard deviation of approximately 1.8. In general tuning the density estimator variance can be non-straightforward as in real problems it will typically vary depending on $\boldsymbol{x}$ and it is not clear which value or values to use to measure the variance at, potentially requiring an additional preliminary run to find a suitable $\boldsymbol{x}$ value to tune at. Further the non-constant estimator variances found in practice will tend to give an accept rate which varies in mean and variance across the target space. This gives a noisy accept rate signal for adaptive algorithms to tune the step-size by, potentially requiring a slower adaptation rate for stability.

In contrast the APM MI+MH method decouples the MI auxiliary updates, which have an acceptance rate controlled by the variance of the density estimate[2] and so N, and the MH target variables updates which have an acceptance rate which is controlled by the proposal step-size λ. The two distinct accept rates provide independent signals to tune the two free

2 During the MI update to the auxiliary variables the target variables $\boldsymbol{x}$ are held fixed and a proposed new set of auxiliary variable values $\boldsymbol{u}^*$ and so density estimate $\hat{p}^* = \varepsilon(\boldsymbol{x}, \boldsymbol{u}^*)$ independently sampled. If the variance of the density estimate tends to zero the ratio of $\hat{p}^*$ to the previous estimate $\hat{p}$ which determines the accept probability of the MI step tends to one.

parameters N and λ by, and which individually will generally be less noisy than the single combined accept rate of the PM MH update.

In density estimators which are simple Monte Carlo averages and the cost of the estimator scales linearly with the number of Monte Carlo samples N such that the results of [207] apply and a choice of $N = 1$ close to optimal, the additional signal provided by the accept rate of the MI updates to the auxiliary variables is of less direct relevance. However as noted previously, in practice often the linear estimator cost assumption will not hold for small N, due to utilisation of parallel computation or high fixed costs. In these cases we may still wish to use the MI accept rate to adjust N so that the accept rate is above some lower threshold: although a low N (and so high estimator variance and low MI step accept probability) may be preferable in the asymptotic regime as the number of samples tends to infinity, in practical settings with finite length chains it can be that an overly high density estimator variance can lead to very low accept rates for the auxiliary variable updates such that in a finite length chain the number of updates to the auxiliary variables is very low (or even zero), potentially leading to biases in the marginal distributions of the sampled target variables.

3.5 PSEUDO-MARGINAL SLICE SAMPLING

Rather than using a MH update to the target variables, the APM framework also makes it simple to apply alternative transition operators to pseudo-marginal inference problems. A particularly appealing option are the linear and elliptical *slice sampling* (SS) algorithms discussed in Chapter 2 (Algorithms 4 and 5); when combined with MI updates to the auxiliary variables we term such methods APM MI+SS. Slice sampling algorithms automatically adapt the scale of proposed moves and so will generally require less tuning than random-walk Metropolis to achieve reasonable performance and also cope better in target distributions where the geometry of the density and so appropriate scale for proposed updates varies across the target variable space.

Slice sampling updates will always lead to a non-zero move of the target variables on each update providing for fixed values of the auxiliary variables the estimator function ϵ is a smooth function of the target variables. In such cases APM MI+SS chains will not show the'sticking' artifacts in the traces of the target variables common to PM MH chains.

As the auxiliary variables are still being updated using Metropolis independence transitions however they will still be susceptible to having proposed moves rejected so the accept rate (and traces if available) of the auxiliary variables updates should also be monitored to check for convergence issues.

The APM MI+MH and MI+SS methods although offering advantages over the standard PM MH method do not address the issue that proposing new auxiliary variable values for fixed values of the target variables independent of the previous auxiliary variable values may sometimes perform poorly in high dimensions. Even weak dependence between the auxiliary variables and target variables will mean that in high-dimensions the typical set of the marginal auxiliary variable distribution R used as the proposal distribution will differ significantly from the typical set of the conditional distribution on the auxiliary variables given the current target variables values used in the Metropolis accept step and so the probability of accepting proposed updates to the auxiliary variables will be small.

One way of increasing the probability of proposed updates to the auxiliary variables from R being accepted is to increase the number of Monte Carlo samples N used in the estimator. For concreteness we will assume we use the importance sampling estimator (3.4) for inference in a hierarchical latent variable model with a multivariate normal importance distribution $q(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right)$ (in general $\boldsymbol{\mu}$ and $\boldsymbol{L}$ will depend on $\boldsymbol{x}$ and $\boldsymbol{y}$ but we leave this dependence implicit for notational simplicity). Using the reparametrisation of the estimator in (3.7), the target density (3.8) on the auxiliary and target variables takes the form

$$\pi(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{NC} \sum_{n=1}^{N} \frac{\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu})}{\mathcal{N}\left(\boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu} \mid \boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right)} \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right). \tag{3.12}$$

Using that $C = \mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$ and $\mathcal{N}\left(\boldsymbol{L}\boldsymbol{u} + \boldsymbol{\mu} \mid \boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}\right) = |\boldsymbol{L}|^{-1}\mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$ this can be manipulated into the form

$$\pi(\boldsymbol{x}, \boldsymbol{u}) = \frac{\mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y})}{N} \sum_{n=1}^{N} \frac{\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y})}{|\boldsymbol{L}|^{-1}\mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right)} \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right).$$

By separating out the terms involving a single auxiliary variable sample $\boldsymbol{u}^{(m)}$, the conditional density on $\boldsymbol{u}^{(m)}$ given the remaining auxiliary variable samples can be shown to take the form of a mixture

$$\pi\left(\boldsymbol{u}^{(m)} \mid \boldsymbol{x}, \{\boldsymbol{u}^{(n)}\}_{n\neq m}\right) \propto |\boldsymbol{L}| \mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(m)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y}) + \sum_{n\neq m}\left(\frac{\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(n)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y})}{|\boldsymbol{L}|^{-1}\mathcal{N}\left(\boldsymbol{u}^{(n)} \mid \mathbf{0}, \mathbf{I}\right)}\right)\mathcal{N}\left(\boldsymbol{u}^{(m)} \mid \mathbf{0}, \mathbf{I}\right) \tag{3.13}$$

The sum of the importance weights in the second term will grow with N (for independent $\boldsymbol{u}^{(n)} \sim \mathcal{N}(\cdot \mid \mathbf{0}, \mathbf{I})\ \forall n \neq m$ it would have an expected value $(N-1)$) and so for large N the second term in the mixture will increasingly dominate and the conditional density on $\boldsymbol{u}^{(m)}$ will tend to $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and therefore be independent of $\boldsymbol{x}$. Therefore as we increase N we expect independently re-sampling the auxiliary variables from R in a MI step to have an increasing accept probability.

Although non-rigorous, this analysis also gives an intution to why the pseudo-marginal method can provide an advantage over directly performing MCMC in the joint space of $\mathbf{x}$ and $\mathbf{z}$ in hierachical latent variable models: if the conditional density on the local latent variables $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ has a challenging geometry, for example it is multimodal, then MCMC transition operators based on local moves working in the $(\mathbf{x}, \mathbf{z})$ are likely to mix poorly for example by getting stuck in a single mode or only being able to make very small moves per update. If we instead reparameterise in terms of a set of auxiliary variables $\left\{\boldsymbol{u}^{(n)}\right\}_{n=1}^{N}$, then we are able to maintain the correct marginal distribution on the target variables $\mathbf{x}$ while working with a distribution on an extended space which becomes increasingly tractable to sample from as we increase N, with the individual auxiliary variable samples $\boldsymbol{u}^{(n)}$ individually having conditional densities which only weakly depend on $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$.

While we can always increase N to the point where independently proposing updates to the auxiliary variables from R will have a reasonable probability of acceptance, this will also increase the computational expense of each update. Rather than proposing new values for the auxiliary variables independently of their previous values, an obvious idea is to take a more standard MCMC approach by using local perturbative updates which leave the overall target distribution (3.8) invariant. For $N = 1$ this equivalent to performing MCMC directly in a non-centred reparameterisation [169] of the joint $(\mathbf{x}, \mathbf{z})$ space by alternating updates

of the target and auxiliary (latent) variables. For $N > 1$ we potentially gain from the conditional distribution on the auxiliary variables being easier for MCMC algorithms to explore though with an increased computational cost per update.

One option is to use a MH method such as random-walk Metropolis to update the auxiliary variables. While with a well tuned proposal distribution this approach can work well, it adds further tuning burden to the user which might outweigh any efficiency gains. For problems in which we can reparametrise the density estimator as a deterministic function of a vector of standard normal draws so that $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$ (such as shown for the importance sampling estimator (3.7)), a particularly appealing option is elliptical slice sampling (Algorithm 5). The elliptical slice sampling algorithm has no free parameters for the user to choose and initially proposes moves to points nearly independent of the current values [155] so if the conditional distribution of the auxiliary variables is well approximated by the normal marginal distribution R, elliptical slice sampling should perform similarly to a MI update. Using the adaptive bracket shrinking procedure discussed in Chapter 2 the elliptical slice sampler is able to exponentially back-off to smaller proposed moves around the current state until a point is found on the slice. Providing for fixed values of the target variables the target density (3.8) is a smooth function of the auxiliary variables, the slice sampling procedure will always lead to a non-zero update of the auxiliary variables.

If the auxiliary variables are instead marginally distributed as independent standard uniform variables i.e. $\rho(\boldsymbol{u}) = \prod_i \mathcal{U}(u_i \mid 0, 1)$[3], one option is to reparameterise these as independent standard normal variables which are then mapped through the normal CDF. We can then run elliptical slice sampling in the transformed normal space. In general evaluation of the normal CDF is a relatively expensive operation and the distortion induced by pushing through the CDF may in some case map a distribution with a relatively simple geometry in the uniform space to a more complex geometry in the normal space. An alternative is to therefore perform linear slice sampling directly in the uniform auxiliary variable space.

3 The auxiliary variables in this case could for example represent all the standard uniform draws from the PRNG that are used to generate random variables in the estimator using the rejection and transform sampling routines discussed in Chapter 2.

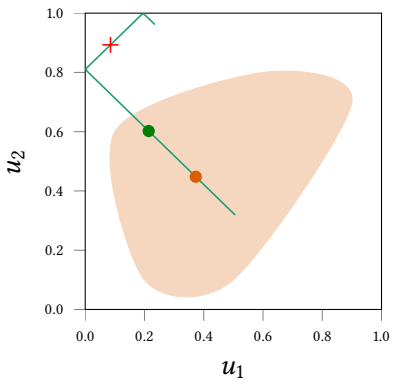


Figure 3.2.: Illustration of reflective linear slice sampling in two dimensions. The orange circle represents the current state, and the light filled orange region the density slice at the sampled slice height (see explanation of Algorithm 4 in Chapter 2 for details). A random slice line direction vector $\boldsymbol{v}$ is sampled from some distribution as in Algorithm 4, for example with elements independently sampled from $\mathcal{N}(0,1)$ or $\mathcal{U}(-1,1)$. This defines a line passing through the current point (green-blue line in Figure), with importantly in this case the line *reflected* at the boundaries of the hypercube (square in this two-dimensional case). An initial bracket of a specified width is randomly placed around the current point on the line. The algorithm then proceeds as in the standard linear slice sampling algorithm by repeatedly proposing a point in the current bracket and accepting if on the slice (in orange region, for example the green circle) or rejecting and shrinking the bracket if off the slice (outside orange region, for example the red cross).

A small subtlety is that the target distribution on the auxiliary variables will only have support on the unit hypercube in this case. We can adjust Algorithm 4 for this setting by replacing Line 8 in the Algorithm with $\boldsymbol{x}^* \leftarrow \textsc{Reflect}(\boldsymbol{x}_n + \lambda \boldsymbol{v})$ (and the likewise the corresponding equivalent expressions in the step-out routine in Lines 17 and 20), where the Reflect function is defined elementwise by

---

function Reflect(u)
  $v \leftarrow u \bmod 2$
  return $v\, \mathbb{1}_{[0,1)}(v) + (2 - v)\, \mathbb{1}_{[1,2)}(v)$

The reflection transformation defined by this function has a unit Jacobian determinant and maintains reversibility and so the reflective slice sampling transition leaves the uniform distribution on the slice invariant. An illustrative schematic of a reflective linear slice sampling transition in two dimension is shown in Figure 3.2. Reflective variants of slice sampling are discussed in [160] and [60].

3.6 NUMERICAL EXPERIMENTS

We will now discuss the results of empirical studies in to the performance of the proposed auxiliary pseudo-marginal methods. Further experiments applying some of the proposed methods in a simulator model inference setting will be discussed in Chapter 4.

3.6.1 Gaussian latent variable model

As a first numerical example we consider inference in a hierarchical Gaussian latent variable model. In particular we assume a model with the factorisation structure shown in Figure 3.1 with

$$\begin{aligned} \mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \mathbf{0}, \mathbf{I}), \quad \mathrm{p}_{\mathbf{z}|\mathbf{x}}(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{z}^{(m)} \mid \boldsymbol{x}, \sigma^2 \mathbf{I}\right), \\ \text{and} \quad \mathrm{p}_{\mathbf{y}|\mathbf{x},\mathbf{z}}(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{y}^{(m)} \mid \boldsymbol{z}^{(m)}, \epsilon^2 \mathbf{I}\right). \end{aligned} \tag{3.14}$$

We used $\sigma = 1$ and $\epsilon = 2$ in the experiments and generate $M = 10$ simulated observed values $\{\boldsymbol{y}^{(m)}\}_{m=1}^{M}$, each of dimensionality $D = 10$. We assume we wish to infer plausible values for the D-dimensional vector $\mathbf{x}$ consistent with the observed $\mathbf{y}$ and so the target distribution for inference has density $p(\boldsymbol{x}) = \mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y})$. Here because of the self-conjugacy of the Gaussian distribution, the marginalisation over the local latent variables $\mathbf{z}$ can be performed analytically to give

$$\mathrm{p}_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x} \mid \boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x} \,\middle|\, \frac{1}{M + \sigma^2 + \epsilon^2} \sum_{m=1}^{M} \boldsymbol{y}^{(m)}, \frac{\sigma^2 + \epsilon^2}{N + \sigma^2 + \epsilon^2} \mathbf{I}\right). \tag{3.15}$$

Although exact inference is therefore tractable in this case, we apply pseudo-marginal MCMC methods to allow us to study the performance of the methods in a case where we have a ground-truth for the inferences to check convergence against.

We use an importance sampling estimator of the form given in (3.4) using $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ as the importance distribution i.e.

$$q\left(\{\boldsymbol{z}^{(m)}\}_{m=1}^{M} \mid \boldsymbol{x}, \{\boldsymbol{y}^{(m)}\}_{m=1}^{M}\right) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{z}^{(m)} \mid \boldsymbol{x}, \sigma^2 \mathbf{I}\right). \tag{3.16}$$

As this importance distribution does not take in to account the observed values $\{\boldsymbol{y}^{(m)}\}_{m=1}^{M}$ it results in a relatively high-variance importance

sampling estimator of the density with a variance which depends on the values of the target variables $\boldsymbol{x}$. Therefore although exact inference in this example is tractable and the target distribution has a simple isotropic geometry, in this pseudo-marginal formulation the model still has some of the key features which can pose challenges to pseudo-marginal inference algorithms.

For the auxiliary pseudo-marginal methods, we use a reparameterisation of the estimator equivalent to (3.7), using the standard normal variables used to generate samples from $\mathrm{P}_{\mathbf{z}|\mathbf{x}}$ as the auxiliary variables, resulting in an auxiliary variable marginal distribution with density $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$ and an estimator function ε

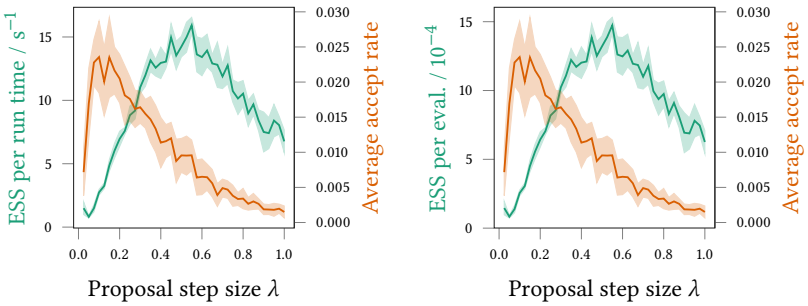
$$\varepsilon(\boldsymbol{x}, \boldsymbol{u}) = \frac{\mathcal{N}(\boldsymbol{x} \mid \mathbf{0}, \mathbf{I})}{N} \sum_{n=1}^{N} \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{y}^{(m)} \mid \sigma \boldsymbol{u}^{(n,m)} + \boldsymbol{x}, \epsilon^2 \mathbf{I}\right), \tag{3.17}$$

with $\boldsymbol{u} = \left[\boldsymbol{u}^{(1,1)}; \ldots \boldsymbol{u}^{(1,M)}; \boldsymbol{u}^{(2,1)} \ldots \boldsymbol{u}^{(N,M)}\right] \in \mathbb{R}^{NM}$.

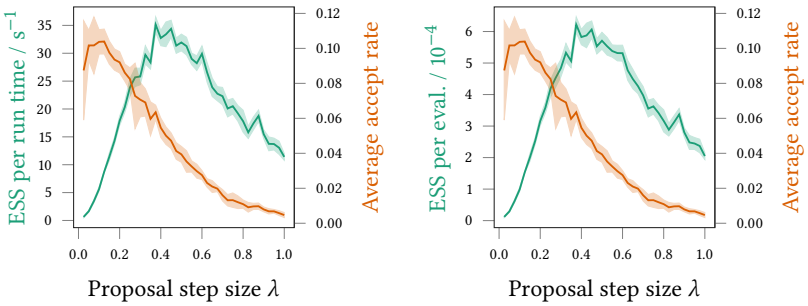
3.6.1.1 *Pseudo-marginal Metropolis–Hastings*

We first applied the PM MH algorithm to perform inference in this model, using an isotropic normal random-walk proposal distribution for the updates to the target variables, i.e. $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. To assess the impact of the choice of the proposal step size parameter λ on sampling efficiency, we ran 10 independent chains initialised from the prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for λ values on a equispaced grid of 40 points between 0.025 and 1, running each chain for 50000 iterations. We ran all experiments for the cases of density estimators using $N = 1$, $N = 8$ and $N = 32$ importance samples, with the logarithm of the density estimate at the value of the target variables $\boldsymbol{x}$ used to generate the observed values $\boldsymbol{y}$ conditioned on having standard deviation 3.6 for $N = 1$, 1.8 for $N = 8$ and 1.2 for $N = 32$.

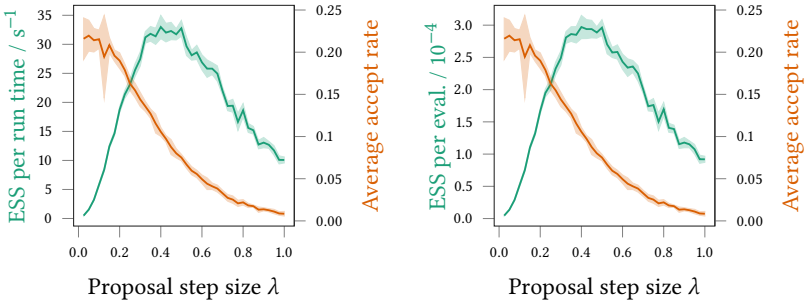
For all combinations of N and λ we estimated the *effective sample size* (as defined for a geometrically ergodic Markov chain in Equation 2.28 of Chapter 2) for the posterior mean of each chain using the R CODA package [177]. We then derived two overall measures of computational efficiency from these effective sample size estimates by normalising either by the number of joint density evaluations in the density estimator (which increases per iteration with the number of importance samples N) or the wall clock run time of the chains in seconds. The



(a) $N = 1$



(b) $N = 8$



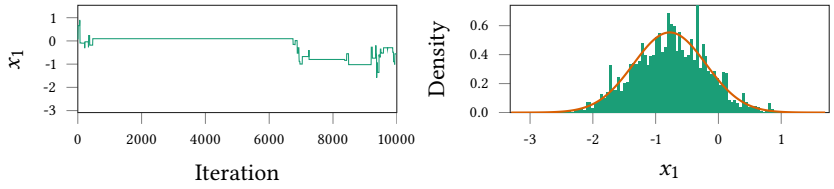
(c) $N = 32$

Figure 3.3.: Results of Gaussian latent variable model PM MH chains. The plots in each row show both the estimated effective sample size (ESS) normalised by either the compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate of MH updates (orange), versus the isotropic random-walk proposal step-size λ for the MH updates to the target variables. The top row shows the case for a density estimator using $N = 1$ importance sample, middle row for $N = 8$ and the bottom row for $N = 32$. In all cases the curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation.

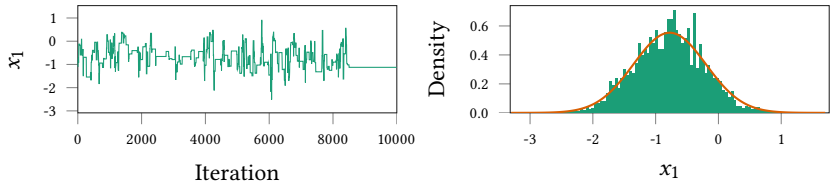
results are plotted in Figure 3.3. Each pair of plots in a row corresponds to a particular number of importance samples. In each row the left column shows the effective sample sizes normalised by the run time and the right column by the number of density evaluations, with the green curves representing the mean of these values across all the chains and the filled region plus and minus one standard deviation (note the standard deviation rather than standard error of mean was used as in some of the plots the standard error was too small to be easily visible). On each axis as well as the normalised effective sample size, the average accept rate across the chains is also plotted in orange (with scale shown on the right vertical axis), with again the curves showing the mean value across the chains and the filled regions plus and minus one standard deviation.

The results of [207] suggest that asymptotically using $N = 1$ importance sample should be optimal in this case assuming a linear increase in the cost of generating each samples with N. The measure of computational efficiency used in [207] most closely corresponds to the estimated effective sample size normalised by the number of density evaluations (which scales linearly with N), and indeed on this measure (green curves in right column of Figure 3.3) we see that the chains using $N = 1$ outperforms the $N = 8$ and $N = 32$ cases.

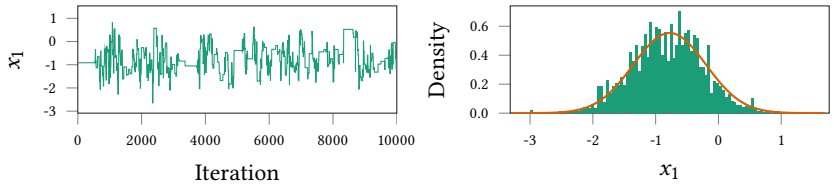
The plots in Figure 3.3a and to a lesser extent 3.3b show a spurious appearing behaviour for the smallest step sizes that the accept rate (orange curve) seems to initially *increase* as the step size is made larger, contrary to what we would reasonably expect. This anomaly can be ascribed to a lack of convergence in the chains with small step sizes due to the sticking behaviour discussed previously. For the $N = 1$ case, because of the relatively high density estimator variance, the chains are prone to getting stuck for thousands of iterations at a time. The estimator variance is dependent on the values of the target variables $\boldsymbol{x}$ and generally seems to be lower for values typical under the posterior. As the chains are initialised from the prior, they tend to therefore initialised in regions in which the estimator variance is higher than typical often leading to long sticking periods near the start of the chain. For the chains with small step sizes the chain is slower to 'warm-up' and converge towards the typical set of the posterior distribution on the target variables and so this propensity for sticking during the initial warm-up period has a larger effect, leading to some chains rejecting



(a) $N = 1, \lambda = 0.550$



(b) $N = 8, \lambda = 0.375$



(c) $N = 32, \lambda = 0.400$

Figure 3.4.: Example traces and histograms of PM MH chains in Gaussian latent variable model inference task. In each row a trace of the sampled values for the x_1 target variable in the last 10000 iterations of a PM MH Markov chain using the optimal step size for the relevant N found from Figure 3.3 is shown in the left plot, while the right plot shows a histogram of the samples values from the full chain (green filled region) against the exact marginal posterior density (orange curve). In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the increased number of density evaluations for higher N, so the $N = 1$ plot is of a chain of 3.2×10^6 samples, the $N = 8$ plot is of 8×10^5 samples as the $N = 32$ plot of 1×10^5 samples.

nearly all updates even though the step size is very small. This counter intuitive behaviour of the empirical accept rates for small step sizes and general noisiness of the dependency of the accept rate on the step size, particularly for small N, highlights the difficulty of tuning the PM MH updates: the low accept rates here would intuitively indicate the step size should be made smaller but in some cases this would actually make the measured accept rate even worse.

Figure 3.4 shows example traces of the x_1 variable samples for chains using density estimates with $N = 1$, $N = 8$ and $N = 32$ importance samples. In each case the step size suggested to be optimal by the results in Figure 3.3 (in terms of effective samples per density evaluation) has been used, and the traces shown are the last 10000 iterations of a longer run. Also shown are histogram estimates of the posterior marginal densities on the x_1 variable using the sampled states from the whole chain, with the total number of samples in each chain adjusted to account for the extra computational cost of using more importance samples, along with a curve showing the true posterior marginal density. The propensity of the chains to stick is clearly visible in the traces particularly for the $N = 1$ case, with long series of thousands of rejected updates at a time. This is also reflected in the noisiness of the marginal density estimates with spurious peaks appearing around the states where the chain gets stuck.

When comparing instead in terms of the estimated effective sample size normalised by actual chain run time (green curves in left column of Figure 3.3) the results no longer suggest $N = 1$ is optimal, with the $N = 8$ and $N = 32$ cases both performing better on this measure for all step sizes. This can be explained by the non-linear scaling of the overall computational cost per update with the number of importance samples with both overhead from the implementation of the rest of the operations in the transition and only partial utilisation of all parallel compute resource (the CPU used in the experiments had 4 cores). Rather than the increase in efficiency per actual run time which is fairly implementation and device dependent, a possibly stronger reason suggested by the results to use $N > 1$ is the less brittle nature of the chains behaviour, with the very low accept rates in the $N = 1$ case needing long runs to smooth out the effects of long series of rejections.

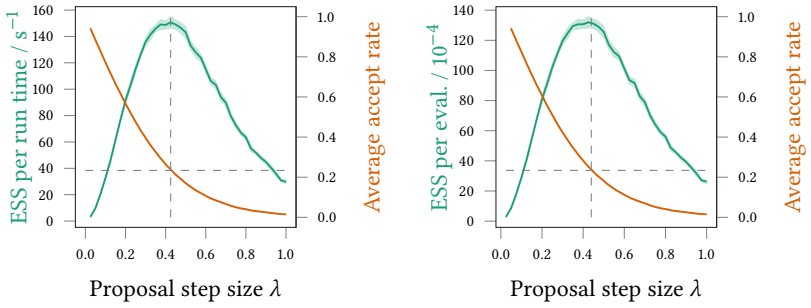
The results in Figure 3.3 also highlight the difficulty of tuning the proposal step size when using a random-walk Metropolis PM MH update.

The optimal step size appears to possibly weakly depend on the number of importance samples used (though the noisiness of the curves make this difficult to determine). Further there is not a clear relationship between the average accept rate and optimal step size. As previously stated the result of [73] that a step size giving an accept rate of 0.234 is close to optimal is not applicable to the update here, with this confirmed empirically by the fact that only the chains with the smallest step sizes for the $N = 32$ case are even able to achieve an accept rate close to 0.234 (and are far from optimal in efficiency). In practice we therefore we do not have an obvious signal to tune the step size by beyond running pilot chains and computing effective sample sizes which is likely to add too much cost to justify any gain in efficiency from choosing a better step size for subsequent chains.

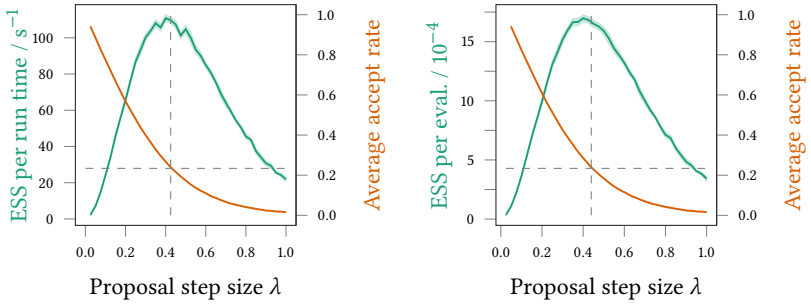
3.6.1.2 *Splitting the update*

We next applied the proposed APM MI+MH algorithm to perform inference in the Gaussian latent variable model. From an implementation perspective this simply requires the original combined update to the auxiliary and target variables in the PM MH case to be split in to separate MI updates of the auxiliary variables given fixed target variables and MH updates of the target variables for fixed auxiliary variables. Despite the seemingly minor change to the form of the update, the difference in the results is dramatic.

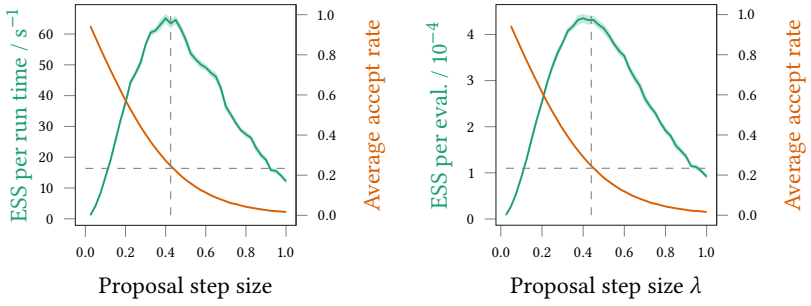
Figure 3.5 shows plots of results of an equivalent series of experiments as used to produce Figure 3.3. In this case the horizontal axes on the plots shows the proposal step size for the MH updates to the target variables which again use an random-walk Metropolis proposal $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. As previously 10 independent chains initialised from the prior were run for each step size λ and number of importance samples N pair, with in this case shorter chains of 20000 iterations used (with the known posterior means and standard deviations used to establish that the chains had adequately converged). Again the estimated effective sample sizes for estimates of the (known) posterior mean were computed for each chain, with the green curves in the left column of plots in Figure 3.5 showing the mean of these estimated effective sample sizes across the chains normalised by the total wall clock run time for the chain, and the right column the effective sample sizes normalised by the number of joint density evaluations. The average accept rate shown



(a) $N = 1$



(b) $N = 8$



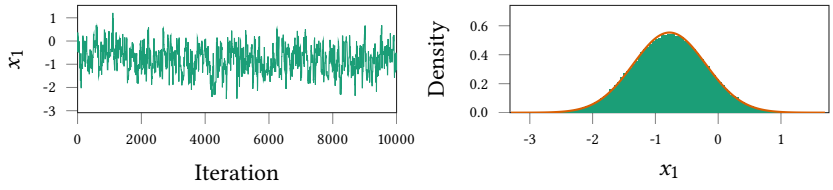
(c) $N = 32$

Figure 3.5.: Results of Gaussian latent variable model APM MI+MH chains. The plots in each row show both the estimated effective sample size (ESS) normalised by either the total compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate for the MH updates (orange), versus the isotropic random-walk proposal step-size for the MH updates to the target variables. The top row shows the case for a density estimator using $N = 1$ importance sample and the bottom row for $N = 8$. The top row shows the case for a density estimator using $N = 1$ importance sample, middle row for $N = 8$ and the bottom row for $N = 32$. In all cases the curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation. The horizontal dashed lines indicate an accept rate of 0.234 and the vertical dashed lines the corresponding proposal step size.

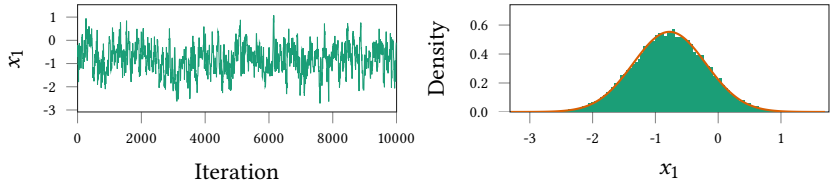
by the orange curves in Figure 3.5 is for the MH update to the target variables. A separate average accept rate was recorded for the MI updates to the auxiliary variables and was found to not show any obvious dependency on the target variable proposal step size, with an average accept rate of approximately 0.025 for chains with $N = 1$ importance sample in the density estimates, an average accept rate of 0.11 for chains with $N = 8$ importance samples in the density estimate and an average accept rate of 0.23 for chains using $N = 32$ importance samples.

On both the time and density evaluation normalised measures of efficiency the APM MI+MH chains perform significantly better than the PM MH chains. The peak effective sample size per density evaluation value for the $N = 1$ and $\lambda = 0.425$ case is around a factor of ten higher than the corresponding peak value for the PM MH chains, while in terms of the effective sample size per run time metric the best APM MI+MH chains show around a factor four improvement over the PM MH chains. While other experiments have suggested this level of improvement is atypical, it seems reasonable to conclude that at least in some cases the extra overhead introduced by requiring two density estimates per overall update is worthwhile.

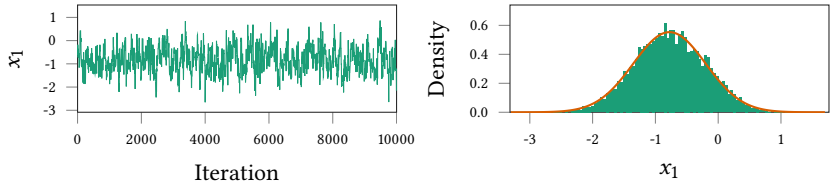
More importantly perhaps the curves in Figure 3.5 suggest the APM MI+MH update is signicantly easier to tune. The average accept rate of the MH updates to the target variables shows the expected monotonically decreasing behaviour as the step size is increased and in general the measured accept rates are significantly less noisy than the corresponding accept rates for the PM MH updates. The horizontal dashed lines in Figure 3.5 indicate an average accept rate of 0.234 with the corresponding vertical dashed lines showing the estimated proposal step size corresponding to this acceptance rate. As can be seen by both the compute time and density evaluation normalised measures of sampling efficiency, the chains with proposal step sizes giving accept rates near to 0.234 are close to optimal in efficiency, suggesting the theoretical result of [73] holds here as suggested earlier. Further in this model at least, this relationship seems to hold for a range of different numbers of importance samples and so density estimator variances. This suggests it is valid to use standard adaptive approaches which use the average accept rate as a control signal to tune the step size of the target variables MH proposal distribution when using the APM MI+MH update.



(a) $N = 1, \lambda = 0.425$



(b) $N = 8, \lambda = 0.425$



(c) $N = 32, \lambda = 0.425$

Figure 3.6.: Example traces and empirical histograms of APM MI+MH chains in Gaussian latent variable model inference task. In each row a trace of the sampled values for the x_1 target variable in the last 10000 iterations of a APM MI+MH Markov chain using the optimal step size for the relevant N found from Figure 3.5 is shown in the left plot, while the right plot shows an empirical histogram of the samples values from the full chain (green filled region) against the exact marginal posterior density (orange curve). In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the increased number of density evaluations for higher N and to account for the 2 evaluations per update compared to PM MH to allow fair comparison with Figure 3.4, so the $N = 1$ plot is of a chain of 1.6×10^6 samples, the $N = 8$ plot is of 2×10^5 samples as the $N = 32$ plot of 5×10^4 samples.

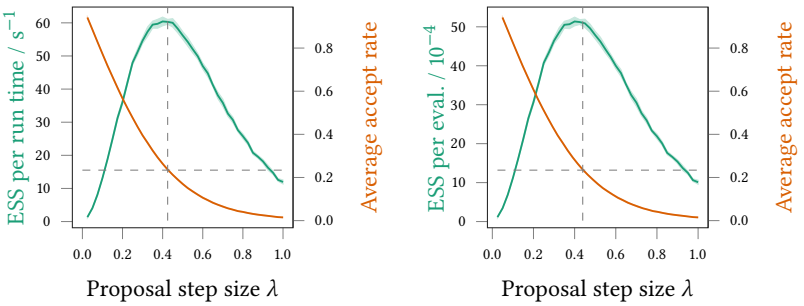


Figure 3.7.: Results of Gaussian latent variable model APM SS+MH chains (using $N = 1$ importance sample). The plots in each row show both the estimated effective sample size (ESS) normalised by either the total compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate for the MH updates (orange), versus the isotropic random-walk proposal step-size λ for the MH updates to the target variables. The curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation.

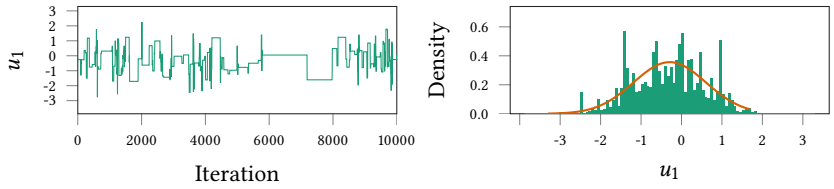
In further contrast to the PM MH results, the results for the APM MI+MH chains seem to unambiguously support using $N = 1$ importance sample. On both the computation time and density evaluation normalised measures of efficiency, the chains using one importance sample dominate over the $N = 8$ and $N = 32$ cases. The APM MI+MH chains using a single importance sample do not show the pathological sticking behaviour evident in the PM MH chains, with an example trace shown for a step size of $\lambda = 0.425$ (which Figure 3.5 suggests is close to optimal) in Figure 3.6a. Unlike the $N = 1$ PM MH trace, over the 10000 iterations shown the APM MI+MH seems to mix well with no obvious sticking periods. The example traces for the $N = 8$ and $N = 32$ APM MI+MH chains in Figure 3.6 also seem to follow this pattern. Comparing the posterior marginal density estimates for the x_1 target variable shown in the right column of Figure 3.6, the marginal estimates for the $N = 1$ case appear the smoothest, almost indistinguishable from the curve of the true density (to normalised for the additional density evaluations required for the $N = 8$ and $N = 32$ cases the number of samples in the chains used to produce the histograms is reduced accordingly). This again suggests that in this case any improvement in mixing by using $N > 1$ in this case is outweighed by the cost of the additional density evaluations.

3.6.1.3 *Slice sampling the auxiliary variables*

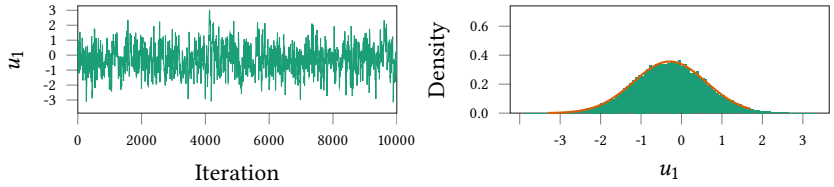
For the APM MI+MH chains discussed in the previous subsection, when using $N = 1$ importance sample the MI updates to the auxiliary variables were only accepted 2.5% of the time. Although this did not appear to impede convergence of the chain in this case, more generally low accept rates for the MI updates to the auxiliary variables may be a cause for concern as in shorter chains this will mean the auxiliary variables are only updated a small number of times across the chain. As convergence of the distribution on the target variables in the chain state to their marginal target distribution is reliant on the distribution of the auxiliary variables in the chain state also converging to the corresponding target distribution, very infrequent updates of the auxiliary variables could potentially lead to difficult to diagnose convergence issues in the target variable chains. Although increasing the number of importance samples in the estimator can increase the MI step accept rate as seen in the APM MI+MH experiments above, the increase in acceptance rate with the number of samples is relatively small - going from $N = 1$ to $N = 8$ samples gives a roughly four-fold improvement in acceptance rate, while going from $N = 8$ to $N = 32$ approximately doubles the accept rate again.

The earlier suggestion to use perturbative updates to the auxiliary variables provides an alternative approach. We test specifically here the proposal to use elliptical slice sampling updates to the auxiliary variables, which is a natural choice in this case due to their Gaussian marginal distribution. We use the same MH update to the target variables as in the experiments in the previous two subsections, and again measure sampling efficiency for different proposal step sizes λ. We only run chains using an density estimator taking $N = 1$ importance sample in this case as we are mainly interested in using perturbative updates to the auxiliary variables as an alternative to having to increase the number of importance samples to achieve reasonable acceptance rates for MI updates to the auxiliary variables.

Results for an equivalent series of experiments as discussed in the previous two subsections for APM SS+MH chains using elliptical slice sampling updates to the auxiliary variables are shown in Figure 3.7. In this case as the MI updates to the auxiliary variables for the $N = 1$ case seemed to be sufficient to achieve convergence, the elliptical slice sampling updates do not seem to significantly improve mixing of the target variables. The



(a) APM MI+MH $N = 1, \lambda = 0.425$



(b) APM SS+MH $N = 1, \lambda = 0.425$

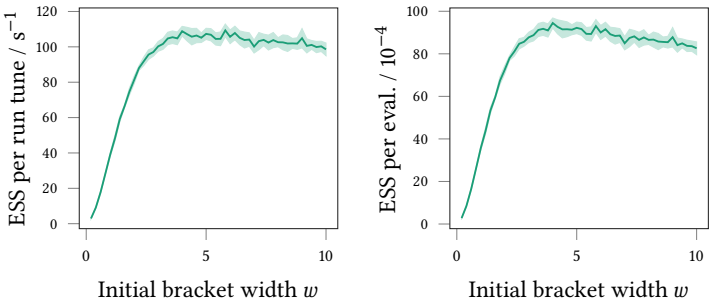
Figure 3.8.: Example traces and histograms of an auxiliary variable in APM MI+MH and APM SS+MH chains in Gaussian latent variable model inference task. In each row a trace of the sampled values for the u_1 auxiliary variable in the last 10000 iterations of a Markov chain using the optimal step size $\lambda = 0.425$ and $N = 1$ is shown in the left plot, while the right plot shows a histogram of the sample values from the full chain (green filled region) against the exact marginal posterior density (orange curve). In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the roughly two times increase in the number of density evaluations per sample for the APM SS+MH updates compared to APM MI+MH, so the APM MI+MH plot is of a chain of 10^5 samples and the APM SS+MH plot is of 5×10^4 samples.

extra overhead from the adaptive slice sampling updates means overall computational efficiency decreases by roughly a factor of two across all proposal step sizes λ compared to the corresponding APM MI+MH results for $N = 1$ in Figure 3.5a, with this consistent across both the density evaulation normalised efficiency metric and run time normalised measure.

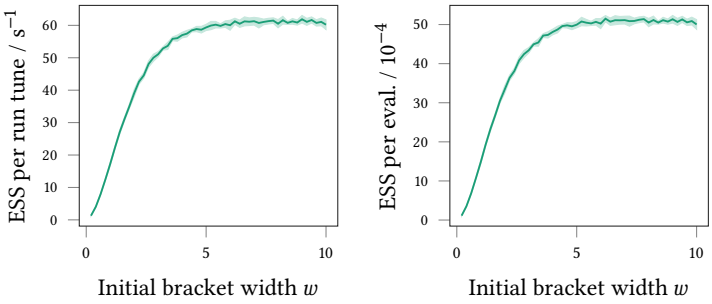
Although the slice sampling updates do not help improve the sampling of the target variables here, the resulting auxiliary variables samples are much more representative of the true posterior distribution on the auxiliary variables (which again can be found analytically) compared to when using MI updates. Figure 3.8 shows traces and histograms of one of the auxiliary variables for chains computed using both the APM MI+MH and APM SS+MH updates. The slice sampling updates give significantly better mixing of the auxiliary variables than the MI updates which due to the low accept rate remain fixed for many iterations. Although in this case this does seem to translate to an obvious improvement in convergence of the target variables, more generally a factor two increase in run time for the added robustness of significantly improved mixing of the auxiliary variables seems like it will often be a worthwhile tradeoff to avoid possible convergence issues.

3.6.1.4 Slice sampling the target variables

As a final set of experiment for this model, we explored the use of slice sampling updates to the target variables with an auxiliary pseudo-marginal framework, specifically linear slice sampling updates along an isotropically sampled direction. To test the claim that the efficiency of slice sampling updates is less sensitive to the choice of the free initial bracket width parameter w of the algorithm than random-walk Metropolis updates are to the choice of the proposal step size parameter λ, we ran a similar series of experiments as in the previous sub-sections to analyse the dependency of sampling efficiency on λ by instead varying the initial bracket width w. For each of 50 initial bracket width w values on an equi-spaced grid between 0.2 and 10, we ran 10 independent APM MI+SS and APM SS+SS chains (with elliptical slice sampling updates to the auxiliary variables) initialised from the prior. As previously for each set of chains for a particular w value we computed the effective sample sizes of the chains for the estimate of the posterior mean and normalised this value by both the total wall-clock run time in seconds



(a) APM MI+SS ($N = 1$)



(b) APM SS+SS ($N = 1$)

Figure 3.9.: Results of Gaussian latent variable model APM chains using SS to target variables an either MI updates to auxiliary variables (top row) or elliptical SS updates (bottom row). The plots in each row show both the estimated effective sample size (ESS) normalised by either the total compute time (left column) or number of density estimator evaluations (right column), versus the slice sampler initial bracket width for the linear SS updates to the target variables. The curves show mean values across 10 independent chains initialised from the prior and filled region show ±1 standard deviation.

and total number of joint density evaluations to give two measures of overall efficiency. The means and one standard deviation intervals of these values across the 10 chains are shown for the APM MI+SS chains in Figure 3.9a and for the APM SS+SS chains in Figure 3.9b. In all cases zero linear step-out iterations were used in the slice sampling updates to the target variables.

The peak efficiency achieved by the APM MI+SS chains on this problem is less than that for the best APM MI+MH chains by a factor of around 1.5 on both measures of efficiency. As the slice sampling updates do more work per iteration than the MH updates this is not unexpected as a well-tuned MH update will generally perform better than a slice sampling update when the geometry of the target distribution is simple (as is the case here). Importantly however the slice sampling updates maintain a computational efficiency that is within around 10% of the optimal efficiency across a wide range of intial bracket width values, with values from $w = 2$ to $w = 10$ all seeming to perfom reasonably well in this problem. This is in contrast to the much tighter range of proposal step size values required to get good performance with MH updates to the target variables. The exponential back-off to smaller proposals provided by the adaptive bracket shrinking procedure in the slice sampling transition means that the penalty for using an overly large scale parameter w is much less severe than the corresponding situation for using an overly large λ in a MH update.

The results for the APM SS+SS show a similar pattern compared to the APM SS+MH results, except for that the best APM SS+MH chains perform almost identically to the best APM SS+SS chains. This is due to the extra overhead introduced by the elliptical slice sampling updates to the auxiliary variables having a larger effect than the slightly more efficient updates to the target variables by the optimally tuned MH updates in this case. This also seems to explain the even slower drop-off in efficiency for the APM SS+SS for large values of the initial bracket width w, with the extra density evaluations this requires on average having a smaller overall effect on efficiency due to the higher baseline number of evaluations due to the elliptical slice sampling updates.

3.6.2 Gaussian process probit regression

As a second experiment we consider a more challenging problem of inferring the parameters of the covariance function of a latent Gaus-

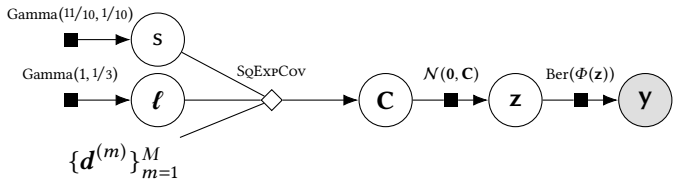


Figure 3.10.: Gaussian process probit regression factor graph.

sian process used to model the relationship between pairs of feature vectors and binary target outputs. The use of PM MH for this task was considered in [67] and shown to give signficant improvements over competing MCMC methods.

As an example data set we used the Wisconsin breast cancer prediction data set [131] from the UCI machine learning dataset repository [128] as also used for experiments in [67]. The data $\{\boldsymbol{d}^{(m)}, y_m\}_{m=1}^M$ consists of pairs of vectors $\boldsymbol{d}^{(m)}$ of $K = 9$ integer descriptors of individual cells found in a fine needle aspiration biopsy of suspect breast lumps, and a binary class y_m indicating whether the lump was later found to malignant or benign. The original dataset contains 699 data-points, however 17 data-points have missing atrributes so $M = 682$ data-points were used in the experiments here.

To model the unknown relationship between the input descriptors and binary class label output, a zero-mean Gaussian process prior [186] was placed on a set of latent real-valued function values $\mathbf{z} \in \mathbb{R}^M$. A squared exponential covariance function was used with per-feature length scales $\boldsymbol{\ell} \in \mathbb{R}_{>0}^K$ and output scale $\mathrm{s} \in \mathbb{R}_{>0}$, with the covariance function specifically defined as

---

function SqExpCov($\{\boldsymbol{d}^{(m)}\}_{m=1}^M, \mathrm{s}, \boldsymbol{\ell}, \epsilon = 10^{-8}$)
  for $i \in \{1 \ldots M\}$ **do**
    $\mathrm{C}_{i,i} \leftarrow \mathrm{s} + \epsilon$
    for $j \in \{1 .. j-1\}$ **do**
      $\mathrm{C}_{i,j} \leftarrow \mathrm{s} \exp\left(-\frac{1}{2} \sum_{k=1}^D \left(\frac{d_k^{(i)} - d_k^{(j)}}{\ell_d}\right)^2\right)$
      $\mathrm{C}_{j,i} \leftarrow \mathrm{C}_{i,j}$
  return C

The ϵ value is a 'jitter' parameter to improve numerical stability [186]. This covariance functions represents an assumption that nearby $\boldsymbol{d}^{(m)}$ points correspond to similar $\mathbf{z}$ values, with the typical length-scales in the feature space over which correlations are high determined by

the elements of $\boldsymbol{\ell}$. The latent variables $\mathbf{z}$ are assumed to determine the probability of the observed binary class outputs $\mathbf{y}$ being one or zero by a probit link function i.e. given $\mathbf{z} = z$ the binary outputs are modelled as having a Bernoulli distribution $\text{Ber}(\Phi(z))$ where Φ is the standard normal CDF function. Following [67] Gamma prior distributions were placed on both the length-scale $\boldsymbol{\ell}$ and output scale s covariance function parameters. The overall model is shown as a directed factor graph in Figure 3.10.

For inference we assume we are interested in inferring the posterior distribution on the $\boldsymbol{\ell}$ and s covariance function parameters given the observed input-output pairs, such that we could then use the inferred plausible $\boldsymbol{\ell}$ and s values to make predictions of the outputs corresponding to unlabelled inputs. We define the target variables for inference $\mathbf{x}$ as the logarithms of $\boldsymbol{\ell}$ and s so that the target distribution has support on an unbounded space i.e. $\mathbf{x} = [\log \text{s}; \log \boldsymbol{\ell}]$ and $\mathbf{x} \in \mathbb{R}^{10}$, with a Jacobian determinant factor accounting for the change of variables being included in the transformed prior (marginal) density $\text{p}_{\mathbf{x}}$

$$\text{p}_{\boldsymbol{x}}(\boldsymbol{x}) \propto \exp\left(\frac{11x_1}{10} - \frac{\exp(x_1)}{10}\right) \prod_{i=2}^{10} \exp\left(x_i - \frac{\exp(x_i)}{3}\right). \tag{3.18}$$

The unnormalised target density is then $\tilde{p}(\boldsymbol{x}) = \text{p}_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y}) = \text{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y} \mid \boldsymbol{x})$. We cannot evaluate $\text{p}_{\mathbf{y}|\mathbf{x}}$ as it involves an intractable marginalisation over the latent function values $\mathbf{z}$

$$\text{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}|\boldsymbol{x}) = \int_Z \prod_{m=1}^{M} \left(\Phi(z_m)^{y_m}(1 - \Phi(z_m))^{1-y_m}\right) \mathcal{N}(z \mid \mathbf{0}, \mathbf{C})\, \text{d}z. \tag{3.19}$$

One option would be to construct a Markov chain on the joint $(\mathbf{x}, \mathbf{z})$ space with a target density $\text{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}$, however strong dependencies between the (transformed) covariance function parameters $\mathbf{x}$ and the latent variables $\mathbf{z}$ makes the joint distribution difficult for MCMC dynamics to explore effectively [67]. As an alternative [67] proposes to use the pseudo-marginal framework to construct a Markov chain using an unbiased importance sampling estimator of $\tilde{p}$.

Though a Monte Carlo estimate of (3.19) can be formed by sampling latent values $\mathbf{z}$ from the Gaussian proccess prior $\text{p}_{\mathbf{z}|\mathbf{x}}$, as this ignores the observed output values $\boldsymbol{y}$ this will tend to lead to a density estimator with unusably high variance for the purposes of use in a pseudo-

marginal update. A key insight in [67] was that much lower variance density estimate can be formed by using an optimisation-based approximate inference method to fit a Gaussian approximation $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ to $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}$ (which as discussed previously is the optimal choice for the importance distribution in terms of minimising variance) to use as the importance distribution. In [67] both Laplace's method and EP are considered within this context; we concentrate on Laplace's method here for simplicity.

As discussed in Chapter 2, Laplace's method involves finding the mode of the density being approximated and then evaluating the Hessian matrix of the log density at this point. An efficient and numerically stable implementation of a Newton–Raphson method can be used to perform to find the mode of the latent posterior for this probit regression Gaussian process model [186, §3.4] with the latent posterior density guaranteed to have a unique mode. Each Newton–Raphson step involves computing a Cholesky factorisation of the Hessian of the log density at the current point which has a $O(M^3)$ computational cost. In the experiments typically around 10 Newton steps where needed to achieve convergence when finding the mode. Evaluating the density of the Gaussian process prior on the latent function values $\mathbf{z}$ also requires computing a Cholesky decomposition of the Gaussian process covariance matrix which again has $O(M^3)$ cost. For $M = 682$ as we have here these cubic cost operations will tend to be the dominant contributor to the overall run time. As the Gaussian process covariance and Laplace approximation to the latent posterior both depend on the value of the covariance function parameters and so target variables, the cubic operations have to be performed each time a density estimate is computed at a new value for the target variables.

Once an approximate Gaussian latent posterior $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ has been fitted using Laplace's method, it can then be used as the importance distribution in an importance sampling estimator of the form shown in (3.4). The Cholesky factorisation $\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} = \operatorname{chol} \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}$ is computed as part of the Laplace's method iteration, and so can be reused to efficiently evaluate the importance distribution density at a $O(M^2)$ cost for each importance sample and to generate samples from the importance distribution using $\boldsymbol{z}^{(n)} = \boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}}\boldsymbol{u}^{(n)} + \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}$ where $\boldsymbol{u}^{(n)}$ is a sampled standard normal vector from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, this again having a $O(M^2)$ cost. This same expression can also be used to reparameterise the estimator as a

deterministic function of a set of independent standard normal values $\boldsymbol{u} = [\boldsymbol{u}^{(1)}; \boldsymbol{u}^{(2)} \ldots \boldsymbol{u}^{(N)}]$ as shown previously in (3.7) and as required for the proposed auxiliary pseudo-marginal methods.

Due to the high overhead of the cubic operations the result of [207] that a choice of $N = 1$ is close to optimal does not apply here. In experiments in [207] with a similar Gaussian process classifier model (in their case using a logistic link function and using a dataset with $M = 144$) they found computational efficiency was approximately maximised by using $N = 200$ importance samples with they noting this is around the number required for the $O(M^2N)$ cost of sample generation to be of comparable magnitude to the cubic operation cost. In their example a non-iterative approach is used to find a Gaussian importance distribution hence only a single Cholesky decomposition of the importance distribution covariance matrix is required. The use of an iterative Laplace method approximation for the importance distribution here as proposed in [67] makes it unclear whether a similar choice of the number of importance samples is reasonable here: while the even higher overhead of the multiple cubic operations per estimator evaluation supports possibly even using $N > M$, part of the justification of using an expensive procedure to fit the importance distribution is that it means fewer importance samples are needed to achieve a low-variance density estimator. In the experiments with the same dataset in [67] $N = 1$ importance sample was used and found to work well, though in that case an isotropic covariance function was used with a single length scale parameter such that the dimensionality of the target space was two rather than ten as here.

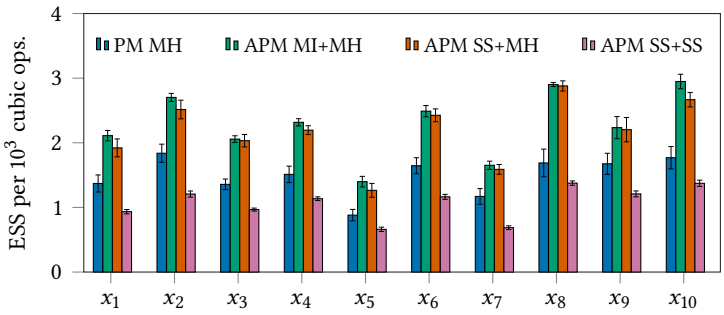
In preliminary runs we found that the PM MH update had very low accept rates however small we set the proposal step-size when using $N = 1$ importance sample in the density estimator. Increasing the number of importance samples to $N = 50$ gave a signficant improvement in performance and overall stability with a negligible increase in run time per update. Increasing the number of importance samples further to $N = 500$ gave a further increase in efficiency but also increased the run time in our implmentation by around one third which outweighed the per iteration sampling efficiency gains made. We therefore used $N = 50$ importance samples for the main experiments with all methods; given the limited number of values tested this is unlikely to be optimal but in most practical situations we would be unlikely to per-

form an exhaustive search for the optimal N. Interestingly the auxiliary pseudo-marginal methods appeared to still be able to mix when using $N = 1$ importance sample with APM MI+MH chains still able to achieve a target accept rate of $\sim 0.15 - 0.3$ for the MH updates to the target variables. Due to the negligible increase in run time however when using $N = 50$ importance samples we performed the experiments for the APM methods with $N = 50$ also.

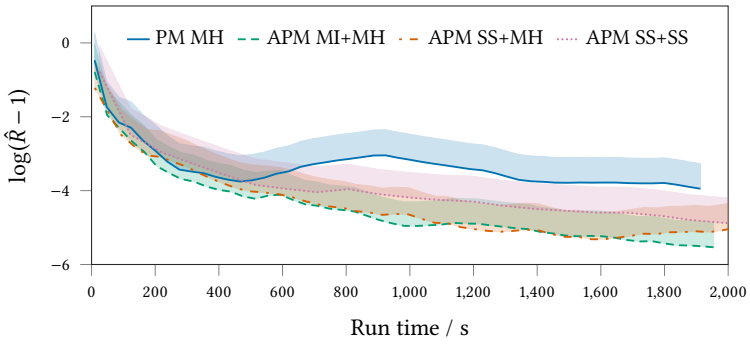
We generated Markov chains for the model using each of PM MH, APM MI+MH, APM SS+MH and APM SS+SS for the updates. For the MH updates to the target variables in the first three methods we used a Gaussian random-walk Metropolis proposal distribution $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. To set the proposal step size λ we followed the adaptive approach used in [67], with the step size adjusted over an initial warm-up phase of 2000 iterations, with the average accept rate over every 100 iterations used as a control signal to decide whether to increase or decrease the step size. Also following [67] a target average accept rate range of $[0.15, 0.3]$ was used[4], with the step size made smaller or larger, if the average accept rate is below or above this range respectively during the adaptive phase. As noted in the previous experiments, while a target rate of 0.234 for the MH updates to the target variables in APM methods can be justified theoretically and empirically, it is not clear what the optimal choice is for PM MH updates, with this seeming to be dependent on the estimator variance and so number of importance samples N. While the $[0.15, 0.3]$ target accept rate range therefore seems reasonable for the APM methods it is unclear whether it is a good choice for the pseudo-marginal method, however as it was used with some success in [67] and given a lack of obvious alternative methods for choosing the target rate, we use it for the PM MH updates here.

For the APM SS+MH and APM SS+SS methods we used elliptical slice sampling for the updates to the auxiliary variables. For the slice sampling update to the target variables in the APM SS+SS chains we used linear slice sampling along a random direction vector (sampled isotropically) with a fixed initial bracket width of $w = 4$ and no linear step out iterations. To account for the 2000 adaptive warm up iterations performed before the main PM MH, APM MI+MH and APM SS+MH chains, for the APM SS+SS chains we ran 1000 warm up iterations before the main chain

4 Although in the published version of [67] it is stated a target range of $[0.2, 0.3]$ was used, the code accompanying the paper suggests a range of $[0.15, 0.3]$ was used so we follow that instead.



(a) Sampling efficiency. Effective sample size (ESS) estimates for each of 10 target variables normalised by the number of cubic cost operations performed per chain. The bars show the means values across 10 independent chains with markers for ±1 standard error of mean.



(b) Chain convergence. Plots of PSRF $\hat{R}$ statistic on a log scale computed across 10 independent chains initialised from the prior for increasing number of chain iterations (normalised by mean total run time for each method to adjust for different per iteration run times) for each of four transition operators tested. Curves show median value and filled regions indicate confidence interval to upper 95th percentile of computed estimate. A $\hat{R}$ value of unity is indicative of chains having converged to stationarity, so for the plotted $\log(\hat{R}-1)$ values, more negative values indicate approaching convergence.

Figure 3.11.: Gaussian process probit regression results.

runs. Four all four methods, 10 chains independently initialised from the prior were run for 10 000 iterations, with both the total number of cubic operations performed and overall run time recorded to allow for adjustment for different per iteration costs in the results.

Results of the experiments are summarised in Figure **??**. As a first measure of performance we consider the relative estimated sampling efficiency of the different methods. For each of the 10 target variables we estimated the effective sample size for the estimated mean of the variable using `R CODA` [177] and normalised these values by the total

number of cubic operations performed in each chain[5]. The averages of these values across the 10 chains per method (and standard errors of mean) are shown for each of the target variables in the bar plot in Figure 3.11a.

By this effective sample size measure of efficiency, the APM MI+MH and APM SS+MH chains both consistently perform better than the PM MH chains, with they performing very similarly to each other, and the APM SS+SS chains perform worse than all other methods. Note that as the updates to the auxiliary variables to do not require any cubic operations (providing the Cholesky factorisations of the Gaussian process prior covariance and importance distribution covariance at the current target variable values are cached from the target variable update), there is very effect on the overall run time from using elliptical SS updates to the auxiliary variables as opposed to MI updates, hence the much closer performance here of APM MI+MH and APM SS+MH compared to the previous Gaussian latent variable experiments. The average accept rate of the MI updates to the auxiliary variables in the APM MI+MH chains was 0.24 here suggesting there is probably a limited gain from using elliptical slice sampling updates to the auxiliary variables here as the MI updates are likely to be mixing the auxiliary variables sufficiently well here. If using a smaller number of importance samples, in which case we would expect the MI accept rate to be lower, the increased robustness from using elliptical slice sampling would seem to be preferable given the negligible effect on run times.

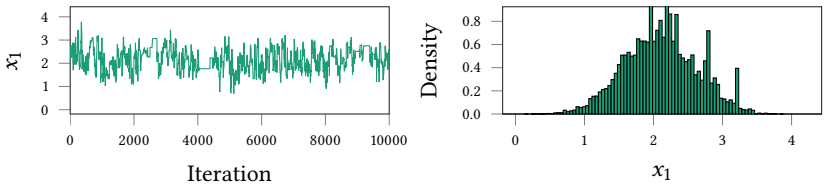
Although PM MH seems to outperform the APM SS+SS method here, other results suggest the estimated effective sample size measures of performance should be treated with some caution, with in general estimated effective sample sizes being susceptible to giving misleading results when chains have poorly converged. Figure 3.11b shows plots of the *potential scale reduction factor* (PSRF) convergence diagnostic proposed by Gelman and Rubin in [75], also often denote the $\hat{R}$ statistic. This is a heuristic measure of Markov chain convergence computed from multiple independent chains initialised from a distribution which should be over dispersed compared to the (common) target distribution (we use

5 The mean chain run time per cubic operation performed was 0.0184 ± 0.00007 s for PM MH, 0.0187 ± 0.00014 s for APM MI+MH, 0.0196 ± 0.00012 s for APM SS+MH and 0.0184 ± 0.00013 s for APM SS+SS so using the cubic operation count as a proxy for overall computational cost seems reasonable here and removes the effect of any variable background system processes on the wall-clock run times.

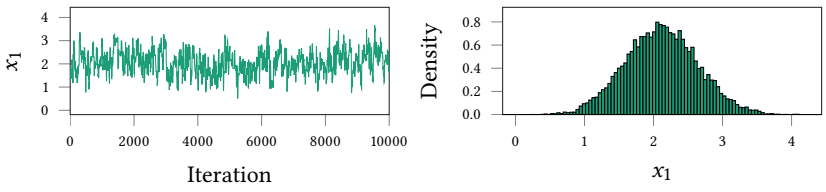
the prior here). The diagnostic compares the between-chain and within-chain variance of each variable in the chain state, with a necessary but not sufficient condition for convergence being that these converge to being equal, corresponding to a $\hat{R}$ value of one. We used ***CODA!*** (CODA!) to estimate the $\hat{R}$ values from the 10 independent chains run for each method as a function of an increasing number of iterations in the chain sequences used to compute the $\hat{R}$ estimates. We then accounted for the different per iteration run time of the different methods (in particular the APM SS+SS chains took on average $\sim 2.5\times$ longer per iteration than the other methods) by plotting these $\hat{R}$ values for increasing chain iterations against the estimated run time to complete that number of iterations, the resulting curves shown in Figure 3.11b. The bold curves show the median of the estimated $\hat{R}$ interval and the light filled regions of the same colour show the 50th–95th percentile range of the estimate. To allow the curves to be more clearly distinguished, the $\hat{R}$ values where plotted on a shifted log scale i.e. $\log(\hat{R} - 1)$, with more negative values therefore corresponding to $\hat{R}$ values closer to one and so indicative of the chains being closer to convergence.

On this measure of performance the PM MH chains seem to perform more poorly, showing a slower convergence rate than the other methods, inclduing the APM SS+SS chains. The non-montonically decreasing behaviour seen in the $\hat{R}$ curve for the PM MH chains seems to be the result of the chains suffering the earlier discussed sticking behaviour, with one of the 10 chains found to have stuck for a run of over 2000 iterations and multiple incidents of sticking periods of hundreds of iterations in all of the chains. An example trace of one of the chains for the x_1 target variable is shown in Figure 3.12a where these sticking periods are clearly visible. The preliminary chains run using $N = 500$ importance samples also showed sticking behaviour though somewhat less frequently, suggesting that while increasing the number of importance samples can lessen the impact of these events, it does not seem to necessarily eliminate them. Figure 3.12 also shows example chain traces for the x_1 variable for each of the three other APM methods; in all cases here there are no visible long sticking periods and this was also reflected across the other chains.

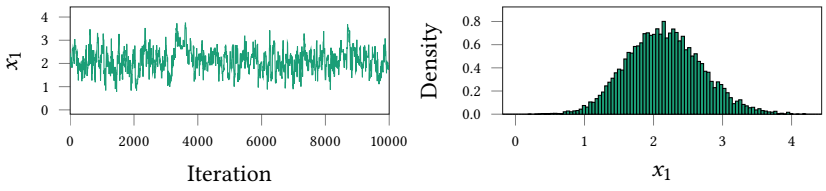
The right column of Figure 3.12 shows histograms for the x_1 target variable computed from the samples from all 10 chains for each method (in the case of the APM SS+SS chains only the first 4000 iterations from



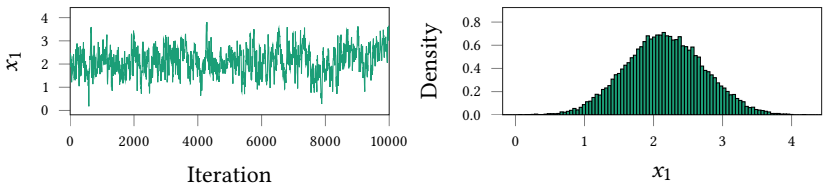
(a) PM MH



(b) APM MI+MH



(c) APM SS+MH



(d) APM SS+SS

Figure 3.12.: Example traces and histograms of target variable $x_1 = \log s$ from chains sampled using pseudo-marginal and auxiliary pseudo-marginal approaches in Gaussian process probit regression model inference task. In each row a trace of the sampled values for the x_1 variable for a single 10000 iteration Markov chain is shown in the left plot, while the right plot shows a histogram of the sampled values from all 10 chains. In the histogram plots the number of samples in the chain used to produce the plot have been adjusted to account for the roughly 2.5 times increase in run time for the APM SS+SS chains compared to the other methods.

each chain where included to account for the roughly 2.5 times slower run time per chain in this case). Although we do not have a ground truth for the marginal posterior density here to compare against, the spurious peaks in the histogram for the PM MH chains can be reasonably assumed to be not a reflection of the actual marginal density but instead of the long sticking artifacts in the chains causing the states that the chain remains stuck at to be overly represented in the histograms. The APM methods produced much smoother marginal density estimates, with the APM SS+SS chains seeming to give a particularly smooth result here despite even with the run time adjustment here meaning this histogram is computed from less than half the number of samples as used in the other methods. Although by no means conclusive, this provides a further hint that the relatively poor performance of the APM SS+SS chains on the effective sample size measure of performance is not a an entirely accurate potrayal.

3.7 DISCUSSION

The auxiliary pseudo-marginal methods discussed in this Chapter are a relatively simple extension to the existing pseudo-marginal MCMC framework which nonetheless offer some important benefits.

Auxiliary variable methods have been extensively studied within the context of particle MCMC methods,

Independently of and concurrently with the original conference publication related to this work, both Dahlin et al. [49] and Deligiannidis et al. [53] considered related frameworks in which the auxiliary random variables of a pseudo-marginal density estimator are updated using a Metropolis–Hastings update which leaves the distribution defined by the density (3.8) on the joint auxiliary–target variable space invariant. Both in particular assume a parameterisation in which the auxiliary variables have a isotropic standard normal marginal distribution, and

4 IMPLICIT GENERATIVE MODELS

In the approximate inference methods considered in Chapters 2 and **??** a unifying element was the requirement to be able to evaluate an explicit probability density function p for the target distribution of interest P. In many inference problems p is only evaluable up to to an unknown normalising constant Z however both sampling and optimisation based inference approaches are generally able to cope with this ambiguity, and in some cases such as importance sampling are able to estimate Z itself. There are however many probabilistic models specified by a generative process in which the density of the model variables is defined only *implicitly* [18, 57, 88] - that is we can generate sample values for the variables in the model, but we cannot tractably evaluate the probability distribution of those variables or more specifically its density with respect to an appropriate base measure.

Although models without an explicit density function are challenging to work with from an inferential perspective, they are ubiquitous in science and engineering in the form of probabilistic models defined by the computational simulation of a physical system. Typically simulator models are specified procedurally in code with any stochasticity introduced by drawing values from a pseudo-random number generator. The complexity of the function mapping from random inputs to simulated outputs typically makes calculating an explicity density on the outputs at best non-trivial. In the common case where the simulator outputs cannot be expressed as an injective function of (potentially a subset of) the random inputs the density on the model variables will usually not have a closed form expression.

There has also been a long history in statistics of using distributions defined by their *quantile function* (i.e. the inverse of their CDF) [98, 225] from which we can easily generate independent samples using the inverse transform sampling method discussed in Chapter 2. Although these *quantile distributions* are often able to offer very flexible descriptions of shape of a distribution [80] often the quantile functions will not have an analytic inverse meaning their CDF and so density func-

tion cannot be evaluated analytically. Generative models in which the density of the model variables is only defined implicitly have also been the subject of substantial recent interest in the machine learning community due to the development of effective training approaches which do not require evaluation of a density on the model variables [62, 85, 127], with there being significant gains in modelling flexibility by dropping the requirement to be able to compute an explicit density function [148, 223].

The focus of this chapter will therefore be methods for performing approximate inference in generative models where we do not necessarily have access to an explicit density on the model variables. A lack of an explicit density function makes it non-trivial to directly apply the approximate inference approaches that have been discussed so far in this thesis. This has spurred the development of inference approaches specifically targeted at implicit generative models such as indirect inference [88] and *approximate Bayesian computation* (ABC) [18].

In both indirect inference and ABC, inferences about plausible values of the unobserved variables are made by computing distances between simulated observed variables and observed data. At a qualitative level, values of the unobserved variables associated with simulated observations that are 'near' to the data are viewed to be more plausible. This approximation that the simulated observations are only close but not equal to the observed data makes the inference problem more tractable, but also biases the inference output. Further simple distance measures tend to become increasingly less informative as the dimensionality of a space increases, making it challenging to use these approaches to perform inference in models with large numbers of unobserved variables. This motivates the use of dimensionality reduction techniques to project the observations to a set of lower-dimensional summary statistics. Although through careful choice of summaries this approach can yield good results, identifying informative summaries is challenging and except for rare cases where sufficient statistics are available any reduction to summary statistics will entail a loss of information about the unobserved variables compared to conditioning on all observations.

We make two main contributions in this chapter. First we show that by reparameterising the approximate conditional expectations estimated in ABC approaches to inference in generative models it is possible to express them in the form of an expectation of a function of a ran-

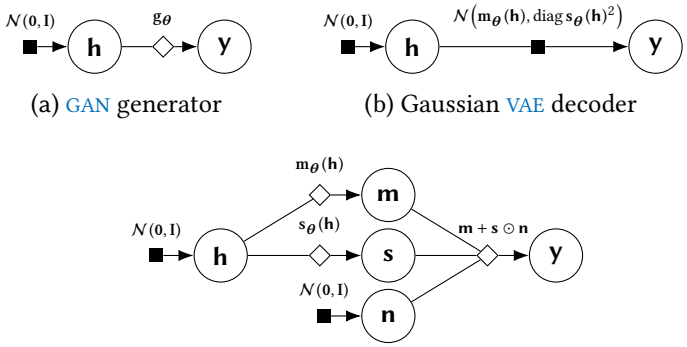
dom vector variable distributed according to a density which we can evaluate up to a normalising constant. This makes it possible to apply efficient general purpose approximate inference methods such as slice sampling and Hamiltonian Monte Carlo to implicit generative models without the need to develop dedicated ABC variants. It is often feasible to apply these methods when conditioning on all observations without the need to reduce dimensionality using summary statistics.

Secondly for a restricted class of generative models we term differentiable generative models and which we define in a following section, we show that it is possible to express exact conditional expectations under the model as integrals against a density we can evaluate pointwise across an implicitly defined manifold. We use this to propose a novel constrained HMC method for performing inference in differentiable generative models. Unlike ABC approaches, this method allows inference to be performed by conditioning the observed variables in the model to be within arbitary small distances of the data values while remaining computationally tractable.

4.1 DIFFERENTIABLE GENERATOR NETWORKS

We will first briefly review two common approaches to specifying generative models using differentiable networks[1], *generative-adversarial networks* (GANs) [85] and *variational autoencoders* (VAEs) [112, 189]. Although the methods using for training these models differ significantly, their generative component have the same form of a function, specified by a differentiable network, which takes as input a vector of random variables from a known distribution and outputs a generated sample from an implicitly defined distribution. The overarching term *differentiable generator networks* has been suggested for generative models with this form [84]. We will use VAE models in some of the later experiments in this chapter so this material is partly to provide the necessary background for our description of the models used in those experiments, however more broadly the structure of the generative models described here was a key inspiration for the ideas described in this chapter.

1 We will follow the suggestion of [237] and refer to what is typically termed a neural network as a differentiable network- i.e. a differentiable parametric function formed by interleaving 'layers' of affine transformations and elementwise non-linearities. This highlights the key property of differentiability and avoids conflation with biological neural networks.



(a) GAN generator (b) Gaussian VAE decoder

(c) Reparameterised Gaussian VAE decoder

Figure 4.1.: Example factor graphs for the generator of a GAN and decoder of a VAE. (a) The generator for a GAN with a standard normal distribution on the hidden code $\mathbf{h}$, this mapped through a differentiable network $\mathbf{g}_{\boldsymbol{\theta}}$, to generate the simulated output vector $\mathbf{y}$. (b) The decoder of a Gaussian VAE. Again a hidden code vector $\mathbf{h}$ with a standard normal distribution is used, differentiable network functions $\mathbf{m}_{\boldsymbol{\theta}}$ and $\mathbf{s}_{\boldsymbol{\theta}}$ then mapping from this code vector to mean and diagonal covariance parameters of a multivariate normal distribution on the output vector $\mathbf{y}$. (c) The same VAE decoder model as (b), with in this case the conditional factor on the outputs $\mathbf{y}$ given hidden code $\mathbf{h}$ reparameterised in terms of a deterministic transformation of a standard normal vector $\mathbf{n}$.

GANs [85] have become a popular approach in unsupervised machine learning for training models which can generate plausible simulated data points, typically images, given a large collection of data to learn from. The training procedure for GANs is posed as a minimax game between a *generator function*, a differentiable network $\mathbf{g}_{\boldsymbol{\theta}}$ which receives as input a vector of values $\mathbf{h}$ drawn from a simple known distribution such as the standard normal and outputs a simulated data point $\mathbf{y} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{h})$, and an adversarial *discriminator function* $\mathbf{d}_{\boldsymbol{\phi}}$, which predicts whether a presented vector input is a simulated or real data point drawn from the training data. Training proceeds by updating the generator parameters $\boldsymbol{\theta}$ to maximise the expected discriminator uncertainty, while the discriminator parameters $\boldsymbol{\phi}$ are updated to minimise the expected discriminator uncertainty.

Although there are many variants of this basic outline of the training procedure, for our purposes the main relevant factor is that most GAN models retain the same basic structure for the generator, which is visualised as a factor graph in Figure 4.1a. While $\mathrm{p}_{\mathbf{h}}$ is known, as $\mathbf{y}$ is a deterministic transformation of $\mathbf{h}$ there is not a well-defined joint density on $\mathbf{h}$ and $\mathbf{y}$. If the Jacobian $\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{h}}$ is full row-rank $\mathrm{P}_{\mathbf{h}}$-almost everywhere

we can in theory apply the change of variable formulae discussed in Chapter 1 to express a density $\mathrm{p}_{\mathbf{y}}$ for the generated outputs in terms of $\mathrm{p}_{\mathbf{h}}$ and $\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{h}}$. The generator function will typically be non-injective however and so the generalised change of variables formula (??) is required which involves finding the pre-image of an output point in the input space and integrating across this set, which will typically be a complex implicitly-defined non-linear manifold. This means even if it exists, analytically computing $\mathrm{p}_{\mathbf{y}}$ is usually intractable and often in practice the requirement on the rank of $\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{h}}$ is not met as typically the dimension of $\mathbf{h}$ is less than that of $\mathbf{y}$.

An alternative generative modelling approach using differentiable networks is the Gaussian VAE [112] or *deep latent Gaussian model* [189]. In a Gaussian VAE differentiable networks $\mathbf{m}_{\boldsymbol{\theta}}$ and $\mathbf{s}_{\boldsymbol{\theta}}$ are used to generate respectively the mean and per-dimension standard deviations, corresponding to a diagonal covariance matrix, of a conditional normal distribution on the outputs given a hidden code vector $\mathbf{h}$ drawn from a known distribution. The simulated output $\mathbf{y}$ can then be generated by sampling from the conditional distribution $\mathcal{N}\left(\mathbf{m}_{\boldsymbol{\theta}}(\mathbf{h}), \operatorname{diag} \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{h})^2\right)$ given a sampled code vector $\mathbf{h}$. Unlike a GAN, in a Gaussian VAE the joint density on $\mathbf{y}$ and $\mathbf{h}$ is tractable to evaluate, for the case of a normally distributed code vector $\mathbf{h}$ corresponding to

$$\mathrm{p}_{\mathbf{y},\mathbf{h}}(\boldsymbol{y}, \boldsymbol{h}) = \mathcal{N}\left(\boldsymbol{y} \mid \mathbf{m}_{\boldsymbol{\theta}}(\boldsymbol{h}),\, \operatorname{diag} \mathbf{s}_{\boldsymbol{\theta}}(\boldsymbol{h})^2\right) \mathcal{N}(\boldsymbol{h} \mid \mathbf{0},\, \mathbf{I}). \tag{4.1}$$

Although typically we cannot marginalise out the hidden code vector $\mathbf{h}$ to get the marginal density on the generated outputs $\mathbf{y}$, having access to the joint density allows the use of standard approximate inference methods when training the model. In particular as suggested by their name variational autoencoders are trained using a parametric variational inference approach which uses a second *encoder* differentiable network to encode the parameters of a variational approximation to the posterior density $\mathrm{p}_{\mathbf{h}|\mathbf{y}}$ given a data point $\boldsymbol{y}$, with a lower bound on the log joint density of the data points then maximised with respect to the encoder and decoder network parameters. Once a VAE model is trained, the joint density (4.1) also allows direct application of approximate inference methods such as MCMC to infer plausible values for a subset $\mathbf{y}_1$ of the decoder generated outputs $\mathbf{y}$ given observations of the remaining values $\mathbf{y}_2$ by jointly inferring $\mathbf{y}_1$ and $\mathbf{h}$ given $\mathbf{y}_2$.

By reparameterising the normal conditional factor $p_{\mathbf{y}|\mathbf{h}}$ in (4.1) as a deterministic transformation $\mathbf{y} = \mathbf{m}_\theta(\mathbf{h}) + \mathbf{s}_\theta(\mathbf{h}) \odot \mathbf{n}$ where $\mathbf{n}$ is a vector of standard normal variables we can express the generative process specified by the decoder of a VAE similarly to that of a GAN by considering the generator to be $\mathbf{g}_\theta(\mathbf{h}, \mathbf{n}) = \mathbf{m}_\theta(\mathbf{h}) + \mathbf{s}_\theta(\mathbf{h}) \odot \mathbf{n}$ with both $\mathbf{h}$ and $\mathbf{n}$ as inputs. These two parameterisations of a VAE decoder are shown as factor graphs in Figures 4.1b and 4.1c.

This definition of the 'generator' corresponding to a VAE decoder is helpful when using it as a building block in a larger generative model where it is composed with other functions. When composing together several generator modules like this, even if we are able to evaluate a density on the variables in an individual module it may not be possible to evaluate a density on the variables of interest in the overall model. However by defining each module in the standard form of a differentiable function from input variables to generated outputs, the overall model retains the same form allowing us to build up more complex models and still be able to apply the same inference methods.

4.2 GENERATIVE MODELS AS TRANSFORMATIONS

In the preceding section we saw that the generative process of both GAN and VAE models can be described as a transformation of a vector of random variables drawn from a known distribution. This formulation of a generative model in fact extends beyond these machine learning examples. Any probabilistic model that we can programmatically generate values from in a finite time can be expressed in the form of a deterministic function which takes as input a vector of random variables sampled from a known distribution. This observation just corresponds to stating that we can track all of the calls to a random number generator in a program, and that given the values sampled from the random number generator all of the operations then performed by the program are deterministic[2]. The key idea we will exploit in this chapter is that we can perform inference in generative models by considering the distribution induced on the random inputs to the model when conditioning on partial observations of the generated output.

2 For the purposes of clarity of exposition here we consider the outputs of a pseudo-random number generator as truly random, even though in reality as we saw in Chapter 2 they are deterministically computed.

To formalise this idea we first introduce some notation. Let $(S, \mathcal{E}, \mathsf{P})$ be a probability space, and $(X, \mathcal{G})$, $(Z, \mathcal{H})$ be two measurable spaces. We denote the vector of observed random variables in the model of interest as $\mathbf{x} : S \to X$ and the vector of unobserved random variables that we wish to infer $\mathbf{z} : S \to Z$. Our objective is to be able to compute conditional expectations $\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{x}\big] : X \to F$ of arbitrary measurable functions $f : Z \to F$ of the unobserved variables given known values for the observed variables $\mathbf{x}$. We now give a concrete definition for what we will consider as constituting a generative model for $\mathbf{x}$ and $\mathbf{z}$.

DEFINITION 4.1 (Generative model): *Let $(U, \mathcal{F})$ be a measurable space and $\mathbf{u} : S \to U$ a random vector taking on values in this space. We require that $\mathsf{P}_{\mathbf{u}}$ has a density ρ that we can evaluate with respect to a base measure μ and that we can generate independent samples from $\mathsf{P}_{\mathbf{u}}$. Then if $\mathbf{g}_{\mathbf{x}} : U \to X$ and $\mathbf{g}_{\mathbf{z}} : U \to Z$ are measurable functions such that*

$$\mathbf{x}(s) = \mathbf{g}_{\mathbf{x}} \circ \mathbf{u}(s) \quad \textit{and} \quad \mathbf{z}(s) = \mathbf{g}_{\mathbf{z}} \circ \mathbf{u}(s) \quad \forall s \in S. \tag{4.2}$$

we define $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_{\mathbf{x}}, \mathbf{g}_{\mathbf{z}})$ as a generative model for $\mathbf{x}$ and $\mathbf{z}$. We call $(U, \mathcal{F})$ the input space *of the generative model, $(X, \mathcal{G})$ the* observed output space *and $(Z, \mathcal{G})$ the* unobserved output space. *Further we will refer to $\mathbf{g}_{\mathbf{x}}$ as the* generator of $\mathbf{x}$ *and likewise $\mathbf{g}_{\mathbf{z}}$ the* generator of $\mathbf{z}$. *The random vector $\mathbf{u}$ is the* random inputs *and the density ρ the* input density.

Intuitively the input vector $\mathbf{u}$ represents all of the values drawn from a random number generator in the code of a generative model and the generator functions $\mathbf{g}_{\mathbf{x}}$ and $\mathbf{g}_{\mathbf{z}}$ represent the operations used to generate values for $\mathbf{x}$ and $\mathbf{z}$ respectively given values for the random inputs $\mathbf{u}$. In some cases the number of random inputs used in a generator evaluation will depend on the values of the random inputs themselves, for example if there is a branching statement which depends on a random input and the operations in each branch use different random inputs. Although implementationally more challenging, we can still consider this case within the above framework by enumerating the random inputs required in all possible control flow paths through the generator code and mapping each to a different element in $\mathbf{u}$. In interpreted languages, this can be done lazily by detecting if a call to a random number generator object has occured at the same point in a execution trace previously and if so matching to same element in $\mathbf{u}$ as used previously otherwise matching to a new $\mathbf{u}$ element.

In this chapter we will concentrate on a restricted class of generative models in which we term *differentiable generative models*.

DEFINITION 4.2 (Differentiable generative model): *Let* $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ *be a generative model for* $\mathbf{x}$ *and* $\mathbf{z}$ *as specified in Definition 4.1. Then if the following conditions are satisfied*

1. $U \subseteq \mathbb{R}^M$, $\mathcal{F} = \mathscr{B}(U)$ *and* $X \subseteq \mathbb{R}^{N_\mathbf{x}}$, $\mathcal{G} = \mathscr{B}(X)$,
2. $\mathsf{P}_\mathbf{u}$ *has density* ρ *with respect to* $\mu = \lambda^M$,
3. *the input density gradient* $\frac{\partial \rho}{\partial \boldsymbol{u}}$ *exists* $\mathsf{P}_\mathbf{u}$*-almost everywhere,*
4. *the* $\mathbf{x}$ *generator Jacobian* $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ *exists* $\mathsf{P}_\mathbf{u}$*-almost everywhere.*

we describe $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ *as a differentiable generative model.*

These requirements are quite severe: for example they exclude any models with discrete random inputs and those in which branch statements in the generator code introduce discontinuities. However there are still a large class of interesting generative models which do meet these conditions: for example models based on approximate integration of partial or ordinary differential equations combined with a stochastic observation model or *stochastic differential equation* (SDE) models without a jump-process component. As differentiability with respect to model parameters is a requirement for training models such as GANs and VAEs using stochastic gradient descent, the corresponding generators will also usually be differentiable with respect to the random inputs and so fall in to this class.

A further restriction we will require in some cases is that the Jacobian $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ is full row-rank $\mathsf{P}_\mathbf{u}$-almost everywhere, which also necessarily means that $M \geq N_\mathbf{x}$ i.e the number of random inputs is at least as many as the number of observed variables that will be conditioned on. In cases where this does not hold the implicitly defined probability distribution $\mathsf{P}_\mathbf{x}$ will not be absolutely continuous with respect to the Lebesgue measure. Instead $\mathsf{P}_\mathbf{x}$ will only have support on a sub-manifold of dimension locally equal to the rank of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ and conditioning on arbitrary $\boldsymbol{x} \in X$ is not a well-defined operation. The GAN generator models trained in practice often do not meet this condition as it is typical to use a lower dimensional hidden input than the generated output dimension [7]. There is no fundamental requirement in adversarial training to use generators of this form however and theoretical results [7] suggest that the lack

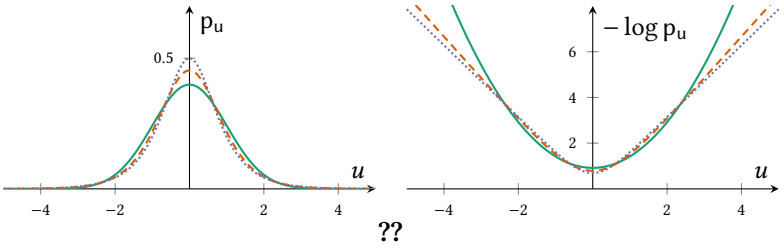


Figure 4.2.: Unit variance densities with unbounded support.

of absolute continuity of the implicit distribution on the generator outputs with respect to $\lambda^{N_\mathbf{x}}$ may contribute to the often unstable behaviour of GAN training.

Although we only required the existence of the input density gradient $\frac{\partial \rho}{\partial \boldsymbol{u}}$ and generator Jacobian $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ in Definition 4.2, unsurprisingly this is motivated by the need to evaluate these terms in the proposed inference methods for differentiable generative models. Although this may seem a limiting requirement for complex models, the availability of efficient general-purpose *automatic differentiation* (AD) libraries [16] means it is possible to automatically calculate the necessary derivatives given just the code defining the forward functions ρ and $\mathbf{g}_\mathbf{x}$. For generative models implemented in existing code this will often require re-coding using an appropriate AD framework. In some cases however it is possible to use AD tools which automatically transform existing source code — for example given C or Fortran code for a function *Tapenade* [97] can generate code for computing the function's derivatives. By applying the reverse-mode accumulation AD (Algorithm 13) the gradient $\frac{\partial \rho}{\partial \boldsymbol{u}}$ can be evaluated at a $O(1)$ cost relative to evaluating the density itself and the Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ can be evaluated at a $O(N_\mathbf{x})$ factor of the cost of a single evaluation of the generator $\mathbf{g}_\mathbf{x}$.

4.3 MODEL PARAMETERISATION

A generative model $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ for $\mathbf{x}$ and $\mathbf{z}$ will not uniquely define the resulting joint distribution $\mathrm{P}_{\mathbf{x},\mathbf{z}}$. As a simple example if $F = \mathscr{B}(U)$, $\mu = \lambda^M$ and $\mathbf{f} : V \to U$ is a diffeomorphism, then we can reparameterise the random inputs as $\mathbf{v} = \mathbf{f}^{-1}(\mathbf{u})$, and if we define an input density $\tilde{\rho}(\boldsymbol{v}) = \left|\frac{\partial \mathbf{f}}{\partial \boldsymbol{v}}\right| \rho(\mathbf{f}(\boldsymbol{v}))$ using the change of variables formula for a diffeomorphism (1.22) then $(V, \mathscr{B}(V), \tilde{\rho}, \mu, \mathbf{g}_\mathbf{x} \circ \mathbf{f}, \mathbf{g}_\mathbf{z} \circ \mathbf{f})$ is also a generative model for $\mathbf{x}$ and $\mathbf{z}$.

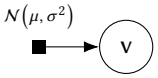
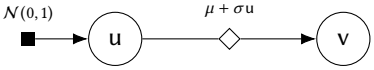
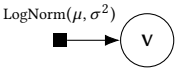
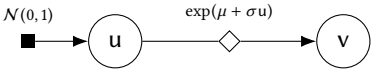
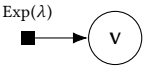
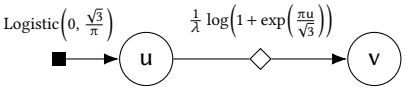
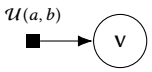
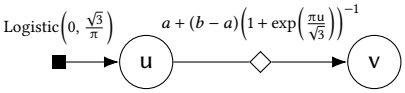
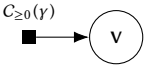
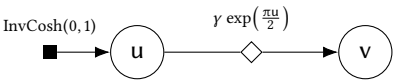
Original factor	Reparametrisation
$\mathcal{N}\left(\mu, \sigma^2\right)$ 	$\mathcal{N}(0,1)$ 
$\text{LogNorm}(\mu, \sigma^2)$ 	$\mathcal{N}(0,1)$ 
$\text{Exp}(\lambda)$ 	$\text{Logistic}\left(0, \frac{\sqrt{3}}{\pi}\right)$ 
$\mathcal{U}(a, b)$ 	$\text{Logistic}\left(0, \frac{\sqrt{3}}{\pi}\right)$ 
$C_{\geq 0}(\gamma)$ 	$\text{InvCosh}(0,1)$ 

Table 4.1.: Reparameterisations of random variables with some common parametric distributions as deterministic transformations of unit-variance unbounded support random variables.

As the inference methods we propose work in the generator input space, we can exploit this ability to reparameterise the input space to endow it with a favourable form for inference. For example we will generally reparametrise input variables with bounded support to transformed variables with unbounded support, for example reparameterising in terms of the logarithm of a strictly positive variable. In general working with unbounded variables will simplify MCMC inference by preventing the need to check transitions respect bounding constraints. Probabilistic programming frameworks such as Stan [74] and PyMC3 make use of a range of such transformations within their MCMC implementations [203].

As well as transforming to variables with unbounded support, another useful heuristic is to parameterise to model as far as possible in terms of inputs variables which have unit variance. Three examples of potentially suitable distributions with unit variance and unbounded support to parameterise the model in terms of are the standard normal $\mathcal{N}(0,1)$, inverse hyperbolic cosine (or hyperbolic secant) distribution $\text{InvCosh}(0,1)$ and the logistic distribution $\text{Logistic}(0, \sqrt{3}/\pi)$. The densities for all three shown for comparison in Figure 4.2 and Table 4.1 shows reparameterisations for some common distributions in terms of variables distributed according to these standard densities. Normalising the scale of variables in ρ typically makes it easier to choose

an appropriate step size parameters for the MCMC transitions. The distribution of the input variables $\mathbf{u}$ once conditioning on the output of the generator may differ significantly from the prior distribution and so normalising the scale of variables in the prior is no guarantee of similar scaling in the posterior, however we have found empirically it is still a useful guideline.

Also note that although we motivated our definition of $\mathbf{u}$ by saying it could be constructed by tracking all the draws from a random number generator, in general we will not want to parameterise $\mathbf{u}$ in terms of low-level uniform draws, but instead use the output of higher-level functions for producing samples from standard densities using the transform and rejection sampling methods discussed in Chapter 2. This is important as if for example we defined as inputs the uniform draws used in the rejection sampling routines used to generate Gamma random variables, we would both require dealing with the complications involved with generators using variable numbers of random inputs as decribed earlier and also have that $\mathbf{g}_{\mathbf{x}}$ would be non-differentiable with respect to the rejection sampling inputs even if the all of the operations performed with the Gamma variable to produce the generated outputs are themselves differentiable. If we instead use the generated Gamma variable itself as the input by including an appropriate Gamma density factor in ρ we side step these issues.

In some cases using the outputs of higher-level random number generator routines as the input variables will introduce dependencies between the variables in the input density ρ. In particular if u_i is drawn from a distribution with parameters depending on one or more previous random inputs $\{\mathrm{u}_j\}_{j\in\mathcal{J}}$, then an appropriate conditional density factor on u_i given $\{\mathrm{u}_j\}_{j\in\mathcal{J}}$ will need to be included in ρ. By using alternative parameterisations it may be possible to avoid introducing such dependencies; for example a random input v_i generated from a normal distribution with mean μ and standard deviation σ which depend on previous random inputs $\{\mathrm{u}_j\}_{j\in\mathcal{J}}$ can instead be parameterised in terms of an independent random variable u_i distributed with a standard normal density $\mathcal{N}(0,1)$ and v_i computed as $\sigma\mathrm{u}_i + \mu$ in the generator. Such *non-centred parameterisations* [32, 170, 180] are available for example for all location-scale family distributions. The reparametrisation of the Gaussian VAE decoder discussed above also used this same identity, and the term 'reparameterisation trick' is often used in the machine learning

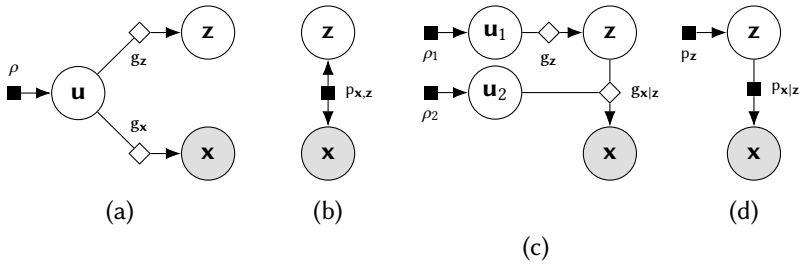


Figure 4.3.: Factor graphs of undirected and directed generative models. Panel (a) shows the more general undirected model case in which observed variables $\mathbf{x}$ and latent variables $\mathbf{z}$ are jointly generated from random inputs $\mathbf{u}$, with (b) showing equivalent factor graph after marginalising out the random inputs. Panel (c) shows the directed model case in which we first generate the latent variables $\mathbf{z}$ from a subset of the random inputs $\mathbf{u}_1$ then generate the observed variables $\mathbf{x}$ from $\mathbf{z}$ and the remaining random inputs $\mathbf{u}_2$, with (d) showing resulting natural directed factorisation of joint distribution when marginalising out $\mathbf{u}_1$ and $\mathbf{u}_2$.

literature to describe this idea [112]. Whether it is necessarily helpful to remove dependencies in ρ like this for the methods discussed in this chapter is an open question and will likely be model specific; it has previously been found that non-centred parameterisations can be beneficial when performing MCMC inference in hierarchical models when the unobserved variables are only weakly identified by observations [24, 169, 170].

4.4 DIRECTED AND UNDIRECTED MODELS

So far we have considered generative models where both the observed and unobserved variables are jointly generated from $\mathbf{u}$ without assuming any particular relationship between $\mathbf{z}$ and $\mathbf{x}$. This structure is shown as a factor graph in Figure 4.3a and a corresponding factor graph for just $\mathbf{x}$ and $\mathbf{z}$ with $\mathbf{u}$ marginalised out shown in Figure 4.3b.

A common special case is when the input space is partitioned $U = U_1 \times U_2$ and the unobserved variables $\mathbf{z}$ are generated from a subset of the random inputs $\mathbf{u}_1 : S \to U_1$ (e.g. corresponding to sampling from a prior distribution over the parameters of a simulator model), with the observed variables $\mathbf{x}$ then generated from a function $\mathbf{g}_{\mathbf{x}|\mathbf{z}} : Z \times U_2 \to X$ which takes as input both the generated unobserved variables $\mathbf{z}$ and the remaining random variables $\mathbf{u}_2 : S \to U_2$, i.e. $\mathbf{x} = \mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2) = \mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{g}_{\mathbf{z}}(\mathbf{u}_1), \mathbf{u}_2)$. This is illustrated as a factor graph

in Figure 4.3c. Again a corresponding factor graph with $\mathbf{u}$ marginalised out is shown in Figure 4.3d, with in this case the structure of the generator making a directed factorisation in terms $\mathrm{p}_{\mathbf{z}}$ and $\mathrm{p}_{\mathbf{x}|\mathbf{z}}$ natural.

We will therefore term models with this structure as *directed generative models* (with the more general case termed *undirected* for symmetry). The method we propose are equally applicable to undirected and directed generative models, though often the extra structure present in the directed case can allow computational gains. Most ABC inference methods concentrate on directed generative models. Typically the marginal density $\mathrm{p}_{\mathbf{z}}$ will be tractable to explicitly compute in such cases, such that it is only the conditional density $\mathrm{p}_{\mathbf{x}|\mathbf{z}}$ which we cannot evaluate. As this conditional density is often referred to as the *likelihood*, this motivates the alternative designation of *likelihood-free inference* for ABC and related methods.

4.5 APPROXIMATE BAYESIAN COMPUTATION

We will now review the ABC approach to inference in generative models. We will assume here that the observed variables in the generative model of interest are real-valued, i.e. that $X \subseteq \mathbb{R}^{N_{\mathbf{x}}}$, with inference in generative models with discrete observations being in general simpler from a theoretical perspective (though not necessarily computationally). The auxiliary-variable description we give of ABC is non-standard, but is consistent with the algorithms used in practice and will help illustrate the relation of our approach to existing ABC methods.

We introduce an auxiliary X-valued random vector $\mathbf{y}$ which depends on the observed random vector $\mathbf{x}$ via a regular conditional distribution $\mathrm{P}_{\mathbf{y}|\mathbf{x}}$ we term the *kernel* which has a conditional density $k_\epsilon : X \times X \to [0, \infty)$ with respect to the Lebesgue measure,

$$\mathrm{P}_{\mathbf{y}|\mathbf{x}}(A \mid \boldsymbol{x}) = \int_A k_\epsilon(\boldsymbol{y}; \boldsymbol{x}) \, \mathrm{d}\boldsymbol{y} \qquad \forall A \in \mathscr{B}(X), \, \boldsymbol{x} \in X. \tag{4.3}$$

The kernel density k_ϵ is parameterised by a *tolerance* ϵ and chosen such that the following conditions holds for arbitrary Lebesgue measurable functions $f : X \to \mathbb{R}$

$$\lim_{\epsilon \to 0} \int_X f(\boldsymbol{y})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{d}\boldsymbol{y} = f(\boldsymbol{x}) \tag{4.4}$$

$$\text{and} \quad \lim_{\epsilon \to 0} \int_X f(\boldsymbol{x})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{d}\boldsymbol{x} = f(\boldsymbol{y}). \tag{4.5}$$

Intuitively these requirements correspond to kernels which collapse to a Dirac delta in the limit of $\epsilon \to 0$. For kernels meeting these condition (4.4) we have that $\forall A \in \mathscr{B}(X)$

$$\begin{aligned} \lim_{\epsilon \to 0} \mathsf{P}_{\mathbf{y}}(A) &= \lim_{\epsilon \to 0} \int_X \mathsf{P}_{\mathbf{y}|\mathbf{x}}(A \mid \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \lim_{\epsilon \to 0} \int_X \int_X \mathbb{1}_A(\boldsymbol{y})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{d}\boldsymbol{y}\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \int_X \mathbb{1}_A(\boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \mathsf{P}_{\mathbf{x}}(A), \end{aligned} \tag{4.6}$$

i.e. that in the limit $\epsilon \to 0$, $\mathbf{y}$ has the same distribution as $\mathbf{x}$. Intuitively, as we decrease the tolerance ϵ we increasingly tightly constrain $\mathbf{y}$ and $\mathbf{x}$ to have similar distributions. Two common choices of kernels satisfying (4.4) and (4.5) are the *uniform ball* and *Gaussian* kernels which respectively have densities

$$k_\epsilon(\boldsymbol{y};\, \boldsymbol{x}) = \frac{\Gamma(\frac{N_{\mathbf{x}}}{2} + 1)}{\pi^{\frac{N_{\mathbf{x}}}{2}} \epsilon^{N_{\mathbf{x}}}}\, \mathbb{1}_{[0,\epsilon]}(\|\boldsymbol{y} - \boldsymbol{x}\|_2) \qquad \text{(uniform ball)}, \tag{4.7}$$

$$\text{and} \quad k_\epsilon(\boldsymbol{y};\, \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{x},\, \epsilon^2 \mathbf{I}\right) \qquad \text{(Gaussian)}. \tag{4.8}$$

The marginal distribution of $\mathbf{y}$ can be written $\forall A \in \mathscr{B}(X)$ as

$$\mathsf{P}_{\mathbf{y}}(A) = \int_X \mathsf{P}_{\mathbf{y}|\mathbf{x}}(A \mid \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \int_A \int_X k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x})\, \mathrm{d}\boldsymbol{y}. \tag{4.9}$$

Therefore $\mathsf{P}_{\mathbf{y}}$ has a density with respect to the Lebesgue measure

$$\begin{aligned} \mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) &= \int_X k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \int_{X \times Z} k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z}) \qquad \forall \boldsymbol{y} \in X. \end{aligned} \tag{4.10}$$

The density $\mathsf{p}_{\mathbf{y}}$ exists irrespective of whether $\mathsf{P}_{\mathbf{x}}$ has a density with respect to the Lebesgue measure (it may not for example if $\mathsf{P}_{\mathbf{x}}$ only has support on a sub-manifold of X). Using this definition of the density $\mathsf{p}_{\mathbf{y}}$

we have that for any measurable function $f : Z \to F$ of the unobserved variables and $\forall A \in \mathscr{B}(X)$ that

$$\begin{aligned}\int_{A\times Z} f(z)\, \mathrm{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}z) &= \int_{A\times X\times Z} f(z)\, \mathrm{P}_{\mathbf{y},\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \int_{X\times Z}\int_{A} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{d}\boldsymbol{y}\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \int_{A}\int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)\, \mathrm{d}\boldsymbol{y}.\end{aligned} \tag{4.11}$$

Using that $\mathrm{P}_{\mathbf{y}}$ has a density $\mathrm{p}_{\mathbf{y}}$ with respect to the Lebesgue measure, and that we can safely ignore the set for which $\mathrm{p}_{\mathbf{y}}(\boldsymbol{y}) = 0$ when integrating against $\mathrm{P}_{\mathbf{y}}$ as it is zero-measure, we have that $\forall A \in \mathscr{B}(X)$

$$\begin{aligned}&\int_{A\times Z} f(z)\, \mathrm{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}z) = \\ &\qquad \int_{A} \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)\, \mathrm{P}_{\mathbf{y}}(\mathrm{d}\boldsymbol{y}).\end{aligned} \tag{4.12}$$

Comparing this to the definition of the conditional expectation from Chapter 1 (1.30) therefore we have $\forall \boldsymbol{y} \in X : \mathrm{p}_{\mathbf{y}}(\boldsymbol{y}) > 0$

$$\begin{aligned}\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \frac{\int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)}{\int_{X\times Z} k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)}.\end{aligned} \tag{4.13}$$

For the case of a model in which $\mathrm{P}_{\mathbf{z}}$ has a density $\mathrm{p}_{\mathbf{z}}$ with respect to the Lebesgue measure, then if we use $f = \mathbb{1}_A$ for $A \in \mathcal{H}$ in (4.13) and the definition of a regular conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{y}}(A \mid \mathbf{y}) = \mathbb{E}[\mathbb{1}_A(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ we have that

$$\mathrm{P}_{\mathbf{z}|\mathbf{y}}(A \mid \boldsymbol{y}) = \int_A \frac{\int_X k_\epsilon(\boldsymbol{y};\boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid z)\, \mathrm{p}_{\mathbf{z}}(z)}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\, \mathrm{d}z. \tag{4.14}$$

In this case the regular conditional distribution $\mathrm{P}_{\mathbf{z}|\mathbf{y}}$ has a conditional density $\mathrm{p}_{\mathbf{z}|\mathbf{y}}$ with respect to the Lebesgue measure,

$$\mathrm{p}_{\mathbf{z}|\mathbf{y}}(z \mid \boldsymbol{y}) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_X k_\epsilon(\boldsymbol{y};\boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid z)\, \mathrm{p}_{\mathbf{z}}(z). \tag{4.15}$$

In reference to typical terminology of Bayesian inference, the density $\mathrm{p}_{\mathbf{z}|\mathbf{y}}$ is termed the ABC posterior, and therefore conditional expectations of the form of (4.13) which correspond to an integral with respect to this ABC posterior, are termed ABC posterior expectations.

We now consider how $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ is related to the conditional expectation we are interested in evaluating $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x}]$. If we assume that $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathrm{p}_{\mathbf{x},\mathbf{z}}$, using (4.5) we have that $\forall \boldsymbol{y} \in X : \mathrm{p}_{\mathbf{x}}(\boldsymbol{y}) > 0$

$$\begin{aligned}\lim_{\epsilon \to 0} \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \lim_{\epsilon \to 0} \frac{\int_Z f(z) \int_X k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, z)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}z}{\int_Z \int_X k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, z)\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}z} \\ &= \frac{\int_Z f(z)\, \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{y}, z)\, \mathrm{d}z}{\int_Z \mathrm{p}_{\mathbf{x},\mathbf{z}}(\boldsymbol{y}, z)\, \mathrm{d}z} = \mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}].\end{aligned}$$

We therefore have that ABC posterior expectations $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y};\, \epsilon]$ converge as $\epsilon \to 0$ to the exact posterior expectations we wish to be able to estimate $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x}]$. Note this result requires that $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ is absolutely continuous with respect to the Lebesgue measure.

Crucially from a computational perspective the numerator and denominator of (4.13) both take the forms of expectations of known functions of $\mathbf{x}$ and $\mathbf{z}$, i.e.

$$\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] = \frac{\mathbb{E}[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathbf{x})]}{\mathbb{E}[k_\epsilon(\boldsymbol{y};\, \mathbf{x})]}. \tag{4.16}$$

Generating Monte Carlo estimates of these expectations only requires us to be able to generate samples from $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ without any requirement to be able to evaluate densities and therefore can be achieved in the implicit generative models of interest.

We can therefore estimate $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ by generating a set of independent pairs of random vectors $\{\mathbf{x}_s,\, \mathbf{z}_s\}_{s=1}^S$ from $\mathrm{P}_{\mathbf{x},\mathbf{z}}$[3] and computing Monte Carlo estimates of the numerator and denominator in (4.16), which gives the following estimator

$$\hat{\mathrm{f}}_{S,\epsilon} = \frac{\sum_{s=1}^S (f(\mathbf{z}_s)\, k_\epsilon(\boldsymbol{y};\, \mathbf{x}_s))}{\sum_{s=1}^S (k_\epsilon(\boldsymbol{y};\, \mathbf{x}_s))}. \tag{4.17}$$

This is directly corresponds to an importance sampling estimator for expectations with respect to $\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ using $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ as the proposal distribu-

3 As ABC is usually applied to directed models this is usually considered as generating $\mathbf{z}$ from a prior then simulating $\mathbf{x}$ given $\mathbf{z}$ however more generally we can sample from the joint.

tion. Therefore if both $f(\mathbf{z})\, k_\epsilon(\boldsymbol{y}; \mathbf{x})$ and $k_\epsilon(\boldsymbol{y}; \mathbf{x})$ have finite variance, then the estimator $\hat{\mathsf{f}}_{S,\epsilon}$ will be consistent,

$$\lim_{S\to\infty} \mathbb{E}\left[\hat{\mathsf{f}}_{S,\epsilon}\right] = \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]. \tag{4.18}$$

If the kernel used is the uniform ball kernel (4.7), the estimator can be manipulated in to a particularly intuitive form

$$\hat{\mathsf{f}}_{S,\epsilon} = \frac{1}{|A|}\sum_{s\in A}(f(\mathbf{z}_s)) \;\text{ with }\; A = \{s \in \{1\dots S\} : \|\boldsymbol{y} - \mathbf{x}_s\|_2 < \epsilon\} \tag{4.19}$$

which corresponds to averaging the values of sampled unobserved variables $\mathbf{z}_s$ where the corresponding samples of model observed variables $\mathbf{x}_s$ are within a distance ϵ of the observed data $\boldsymbol{y}$. The is the standard ABC rejection algorithm [71, 181, 199, 217, 231] , with A corresponding to the indices of the set of accepted samples, with the other samples being 'rejected' as the simulated observations $\mathbf{x}_s$ are more than a distance ϵ from the observed data $\boldsymbol{y}$. As an instance of a rejection sampler[4], conditioned on the acceptance set containing at least one sample, i.e. $|A| > 0$, the (4.19) is an unbiased estimator for $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$.

If we instead use a Gaussian (or other smoothly varying) kernel (4.8), then as for the general case for importance sampling, the estimator (4.17) is no longer unbiased. In the Gaussian kernel case we more highly weight samples if the simulated observed variables are closer to the data which may be viewed as preferable to equally weighting all values within a fixed tolerance as in ABC reject. However as it has support on all of X the Gaussian kernel also gives non-zero weights to all of the samples, with typically most making very little contribution to the expectation which may be considered somewhat wasteful of computation versus the rejection scheme which creates a sparse set of samples to compute expectations over [18]. Kernels with bounded support but non-flat densities such as the *Epanechnikov kernel* [64] which has a

4 Compared to the general rejection sampling scheme described in Algorithm 1 it may seem that we are missing the probabilistic accept step. However the ratio of the target distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ to the proposal distribution $\mathsf{P}_{\mathbf{x},\mathbf{z}}$ here is always equal to exactly zero or a constant $c(\epsilon)$ corresponding to the ratio of the volume of the ϵ radius ball, thus if we choose the bounding constant M in the rejection sampler as $c(\epsilon)$ the acceptance probabilities will always be zero or one and so no auxiliary u values are needed to perform a probabilistic accept. For more general kernels a rejection sampler with probabilistic accept is discussed in [234] as allowing for example samples to be generated from the approximate posterior.

parabolic density in a bounded region, offer a tradeoff between these behaviours of the uniform ball and Gaussian kernels.

Irrespective of the kernel chosen, the estimate formed is only consistent for the ABC posterior expectation $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y}; \epsilon]$ rather than the actual posterior expectation $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$ we are directly interested in. As $\epsilon \to 0$, $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y}; \epsilon]$ converges to $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$, however for reject ABC we also have that as $\epsilon \to 0$ the proportion of accepted samples will tend to zero meaning that we need to expend increasing computational effort to get an estimator for $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y}; \epsilon]$ with a similar variance (which by a standard Monte Carlo argument is inversely proportional to the number of accepted samples).

In the more general importance sampling case, although we do not explicity reject any samples if using a kernel with unbounded support, we instead have that as $\epsilon \to 0$ that the kernel weightings in (4.17) will becoming increasingly dominated by the few samples closest to the observed data and so the contribution from to the estimator (4.17) from all but a few will be negligible, again leading to an increasing number of samples being needed to keep the variance of the estimator reasonable - i.e. the same issues which we discussed in the context of more general importance samplers in Chapter 2. For the exact $\epsilon = 0$ case we would only accept (or equivalently put non-zero weight on) samples for which $\mathbf{x}_s$ is exactly equal to $\boldsymbol{y}$. For $X \subseteq \mathbb{R}^{N_\mathbf{x}}$ if $\mathrm{P}_\mathbf{x}$ is absolutely continuous with respect to the Lebesgue measure, the event $\mathbf{x} = \boldsymbol{y}$ has zero measure under $\mathrm{P}_{\mathbf{x},\mathbf{z}}$[5] and so some degree of approximation due to non-zero ϵ is always required in practice in these simple Monte Carlo ABC schemes.

When the dimensionality of the observed variable vector $\mathbf{x}$ is high it quickly becomes impractical to reduce the variance of these naive Monte Carlo estimators for (**??**) to reasonable levels without using large ϵ which introduces significant approximation error. The ABC rejection method is well known to scale poorly with dimensionality due to curse of dimensionality effects [30, 132, 179]. Although often discussed specifically in the context of ABC, the issues faced are much the same as encountered when trying to use any simple rejection or importance sampling scheme to approximate expectations with respect to a probab-

5 In reality due the use of finite floating-point precision arithmetic the probability of generating observed values exactly consistent with data though vanishingly small is non-zero.

ility distribution on a high-dimensional space. If the proposal distribution ($\mathsf{P}_{\mathbf{x},\mathbf{z}}$ here) is significantly more diffuse than the target distribution ($\mathsf{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ here) an exponentially small proportion of the probability mass of the proposal distribution will lie in the typical set of the target distribution and so very few samples will be accepted / have non-negligible importance weights.

Rather than conditioning on the full observed data most ABC methods used in practice therefore instead use *summary statistics* to extract lower dimensional representations of the observed data [179]. That is a function $\mathbf{s} : X \to T$ is defined which computes summary statistics from simulated observed outputs $\mathbf{x}$ and observed data $\boldsymbol{y}$ with the dimensionality of the summaries, $\dim(T)$, typically much smaller than $N_{\mathbf{x}}$. The ABC posterior expectation is then computed using

$$\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{s} = \mathbf{s}(\boldsymbol{y});\, \epsilon\big] = \frac{\int_{X\times Z} f(\boldsymbol{z})\, k_\epsilon(\mathbf{s}(\boldsymbol{y});\, \mathbf{s}(\boldsymbol{x}))\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}{\int_{X\times Z} k_\epsilon(\mathbf{s}(\boldsymbol{y});\, \mathbf{s}(\boldsymbol{x}))\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}, \quad (4.20)$$

with now the variable conditioned on the T-valued variable $\mathbf{s}$ with

$$\mathsf{P}_{\mathbf{s}|\mathbf{x}}(A \,|\, \boldsymbol{x}) = \int_A k_\epsilon(\boldsymbol{s}; \mathbf{s}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{s} \qquad \forall A \in \mathscr{B}(T),\ \boldsymbol{x} \in X. \quad (4.21)$$

In general the statistics used will not be *sufficient* - the posterior distribution on $\mathbf{z}$ will differ when conditioning on $\mathbf{s}(\mathbf{x})$ compared to conditioning on $\mathbf{x}$ directly. By a data processing inequality argument we know that the mutual information between $\mathbf{z}$ and $\mathbf{s}(\mathbf{x})$ will be less than or equal to the mutual information between $\mathbf{z}$ and $\mathbf{x}$ therefore we would expect for the posterior distribution on $\mathbf{z}$ given $\mathbf{s}(\mathbf{x})$ to be less informative about $\mathbf{z}$ that the posterior distribution given $\mathbf{x}$ [10]. This means that even in the limit of $\epsilon \to 0$ estimates of the ABC summary statistics posterior expectation will generally not converge to the true posterior expectations of interest.

If a, b *and* c *are random variables and* $\mathbb{I}[\mathrm{a}, \mathrm{b}]$ *denotes the mutual information between* a *and* b *the data processing inequality states that if* $\mathrm{a} \perp \mathrm{c} \,|\, \mathrm{b}$ *then* $\mathbb{I}[\mathrm{a}, \mathrm{b}] \geq \mathbb{I}[\mathrm{a}, \mathrm{c}]$.

ABC methods therefore tradeoff between the approximation errors introduced due to using summary statistics and a non-zero tolerance ϵ, and the Monte Carlo error from using a finite number of samples in the estimates. If informative summary statistics can be found then typically the approximation error can be kept to a more reasonable level compared to the conditioning on the full data without the Monte Carlo error becoming impractically large by allowing a smaller ϵ to be used while maintaining a reasonable accept rate. Finding informative low-

dimensional summaries is often critical to getting existing ABC methods to work well in practice and their is a wide literature on developing effective methods for choosing summary statistics - see [179] and [31] for reviews.

In some cases use of summary statistics might not be viewed just as a computational convenience, but as a purposeful excercise in removing 'irrelevant' information from the data. For example if inferring plausible parameter values for a dynamic model of a system given observed sequences of the system state showing quasi-periodic behaviour, then we might view the phase of observed state sequences as an irrelevant artifact of the arbitrary point at which observations were started. In this case conditioning on the exact observed data could be viewed as over constraining the model to reproduce features of the data which are only incidental, and therefore using summary statistics which for example are invariant to phase could be preferable to conditioning on the full data [236].

Similarly the introduction of a kernel in ABC need not be viewed as simply a method for making inference tractable, but instead as part of the modelling process [234]. In general we will expect any observed data to be subject to some amount of measurement noise (at the very least it will include some quantification noise) and so conditioning the model to reproduce the exact values of the data is not necessarily desirable. In this context we can consider $\mathbf{y}$ the noisy measured version of an underlying observed state $\mathbf{x}$ and the kernel $P_{\mathbf{y}|\mathbf{x}}$ as representing the measurement noise model. We might also instead view the kernel $P_{\mathbf{y}|\mathbf{x}}$ as accounting for the mismatch between our proposed model for how the observed values are generated and the true data generating process [187, 234]. In both these cases we could then consider ϵ as a further unobserved variable to be inferred.

These examples demonstrate that in some cases there may be a modelling motivation for introducing summary statistics and / or a 'noise' kernel rather than exactly conditioning on the observed data. In practice however the choice of summary statistics used and size of the ϵ tolerance are typically chosen more on grounds of computational tractability [132, 179, 191]. Therefore inference methods which are able to maintain computational tractability when conditioning on higher-dimensional summaries or in some cases all observations, and when using smaller tolerance ϵ values are of significant practical interest.

4.6 ABC MCMC METHODS

The ABC inference methods considered so far correspond to simple Monte Carlo inference approaches that we previously claimed in Chapter 2 scale poorly to large complex probabilistic models. It is natural to consider therefore whether more scalable approximate inference methods can be applied instead. In this section we will discuss an approach for using MCMC within an ABC framework [133, 209]. The framework we propose in the following section is intended to address some of the shortcomings of this method.

There has also been a significant amount of work on developing more complex ABC inference schemes, with in particular methods based on *sequential Monte Carlo* (SMC) [19, 52, 210, 222] having achieved significant empirical success. Typically however ABC SMC approaches make use of ABC MCMC moves as part of the overall algorithm therefore improved MCMC methods are also of direct relevance to those frameworks. More recently there has also been several approaches proposed for using optimisation based approximate inference schemes in an ABC setting, including expectation propagation [14] and variational methods [151, 224]. These offer an interesting alternative to the standard Monte Carlo based approaches, and the variational methods in particular share significant aspects with some of the ideas proposed here. We will discuss these links in more detail in a later section.

As is standard in ABC methods, ABC MCMC approaches are generally targeted at directed generative models where the unobserved variables has a known marginal density $\mathrm{p}_{\mathbf{z}}$ but where we can only generate samples from the conditional distribution $\mathrm{P}_{\mathbf{x}|\mathbf{z}}$. If a Markov chain is constructed with unique stationary distribution

$$\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}(A,\, B \,|\, \boldsymbol{y}) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_B \int_A \mathrm{p}_{\mathbf{z}}(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \,|\, z)\, \mathrm{d}z \qquad (4.22)$$

then by the standard MCMC convergence theory discused in Chapter 2 we can compute consistent MCMC estimators for (??) by computing averages over the chain states.

An apparent difficulty is that unlike the more typical inference problems considered previously in the context of MCMC methods, the target stationary distribution for the chain (4.22) does not have a closed form

Algorithm 9 ABC Pseudo–Marginal Metropolis–Hastings.

Input: $(\boldsymbol{x}, z)$: current chain state, $\mathrm{p}_{\mathbf{z}}$: marginal density of unobserved variables $\mathbf{z}$, k_ϵ : ABC kernel density, $\boldsymbol{y}$: observed data values, q : density of proposal kernel Q for $\mathbf{z}$ updates.
Output: $(\boldsymbol{x}', z')$: new chain state.

$z^* \sim Q(\cdot \mid z)$
$\boldsymbol{x}^* \sim \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\cdot \mid z^*)$
$u \sim \mathcal{U}(0,1)$
$a \leftarrow \frac{q(z \mid z^*)\, \mathrm{p}_{\mathbf{z}}(z^*)\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x}^*)}{q(z^* \mid z)\, \mathrm{p}_{\mathbf{z}}(z)\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})}$
if $u < a$ **then**
    return $(\boldsymbol{x}^*, z^*)$
else
    return $(\boldsymbol{x}, z)$

density that we can evaluate. It is therefore not clear how to apply any of the standard approaches discussed for constructing a transition operator which leaves a target distribution invariant: Metropolis–Hastings and slice sampling algorithms both involve evaluating the density of the target distribution, while Gibbs sampling requires being able to sample from the per-variable complete conditionals of the target which it seems unlikely we will be able to derive given the lack of a density for $\mathrm{P}_{\mathbf{x}|\mathbf{z}}$.

The key idea of the original ABC MCMC [133] approach is to construct a Metropolis–Hastings proposal kernel in such a way that the unknown (and potentially non-existing) density of the conditional distribution $\mathrm{P}_{\mathbf{x}|\mathbf{z}}$ does not appear in the accept ratio. This can be achieved by perturbatively updating the unobserved variables $\mathbf{z}$ but then independently resampling the observed variables $\mathbf{x}$ given the new proposed $\mathbf{z}$ values from $\mathrm{P}_{\mathbf{x}|\mathbf{z}}$ i.e. generating a new $\mathbf{x}$ value using the model. The method is summarised in Algorithm 9. Here we assume the full observations are conditioned on, i.e. no summary statistics are used; the adjustments for the case where summaries are used are simple. A proposal kernel $Q : \mathcal{H} \times Z \to [0,1]$ needs to be chosen for the updates to the unobserved variables which we can both draw independent samples from and evaluate the density $q : Z \times Z \to [0, \infty)$ of. This proposal distribution can be chosen similarly to the standard Metropolis–Hastings case, with a common choice for $Z \subseteq \mathbb{R}^{N_{\mathbf{z}}}$ being an isotropic Gaussian random-walk proposal density $\mathcal{N}\left(z' | z,\, \sigma^2 \mathbf{I}\right)$. In this case free step-size parameter σ needs to be chosen to trade off between decreasing dependence between successive $\mathbf{z}$ samples (achieved by making σ larger) and maintaining a reasonable accept rate (by not making σ too

large). Although this tuning problem is common to all random-walk Metropolis–Hastings methods we will see later that in this case the compound proposal with $\mathbf{x}$ also being independently resampled from the model $\mathrm{P}_{\mathbf{x}|\mathbf{z}}$ makes the tuning problem more difficult here.

. A proposed $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ pair is accepted with probability

$$\alpha(\boldsymbol{x}^*, \boldsymbol{z}^* \mid \boldsymbol{x}, \boldsymbol{z}) = \min\left\{1, \frac{q(\boldsymbol{z} \mid \boldsymbol{z}^*)\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z}^*)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x}^*)}{q(\boldsymbol{z}^* \mid \boldsymbol{z})\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})}\right\}. \tag{4.23}$$

The transition operator defined by this process is

$$\begin{aligned}
&\mathsf{T}(A, B \mid \boldsymbol{x}, \boldsymbol{z}) = \\
&\quad \int_B \int_A q(\boldsymbol{z}' \mid \boldsymbol{z}) \alpha(\boldsymbol{x}', \boldsymbol{z}' \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x}' \mid \boldsymbol{z}')\, \mathrm{d}\boldsymbol{z}' + \\
&\quad \mathbb{1}_A(\boldsymbol{x})\, \mathbb{1}_B(\boldsymbol{z}) \left(1 - \int_X \int_Y q(\boldsymbol{z}' \mid \boldsymbol{z}) \alpha(\boldsymbol{x}', \boldsymbol{z}' \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x}' \mid \boldsymbol{z}')\, \mathrm{d}\boldsymbol{z}'\right).
\end{aligned}$$

As in the previous discussion of the validity of the Metropolis–Hastings transition operator, for the purposes of showing the transition satisfies the detailed balance condition we can ignore the component of the transition operator corresponding to rejections as staying in the same state will leave any distribution invariant. Consider just the first term we therefore have that

$$\begin{aligned}
&\int_Z \int_X \int_B \int_A q(\boldsymbol{z}' \mid \boldsymbol{z}) \alpha(\boldsymbol{x}', \boldsymbol{z}' \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x}' \mid \boldsymbol{z}')\, \mathrm{d}\boldsymbol{z}'\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid \boldsymbol{z})\, \mathrm{d}\boldsymbol{z} = \\
&\int_Z \int_X \int_B \int_A \min\{q(\boldsymbol{z}' \mid \boldsymbol{z})\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x}), q(\boldsymbol{z} \mid \boldsymbol{z}')\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z}')\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x}')\}\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x}' \mid \boldsymbol{z}')\, \mathrm{d}\boldsymbol{z}'\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid \boldsymbol{z})\, \mathrm{d}\boldsymbol{z}
\end{aligned}$$

leaves (4.22) invariant, and under a suitable choice of proposal density for the $\mathbf{z}$ updates will be aperiodic and irreducible and so have (4.22) as its unique stationary distribution [133, 209].

By making small changes to the unobserved variables $\mathbf{z}$ and so making use of information from the previous state about plausible values for $\mathbf{z}$ under $\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}=\boldsymbol{y}}$ rather than independently sampling them from $\mathrm{P}_{\mathbf{z}}$ as in the simpler Monte Carlo schemes, ABC MCMC can often increase efficiency in generative models with large numbers of unobserved variables to infer [209]. This potential improved efficiency comes at a cost of introducing the usual difficulties associated with MCMC methods such as high dependence between successive samples and difficulty monitoring convergence. Further ABC MCMC chains can be prone to 'sticking'

pathologies - suffering large series of rejections visible as the variables being stuck at a fixed value in traces of the chain state. Though we propose small updates to $\mathbf{z}$ we independently sample proposed simulated observations $\mathbf{x}$ in each transition; often the conditional distribution $\mathrm{P}_{\mathbf{x}|\mathbf{z}=z^*,\mathbf{y}=\boldsymbol{y}}$, i.e. describing the plausible values for $\mathbf{x}$ given the observed data *and* proposed $\mathbf{z}$ values, will be much more concentrated than the distribution $\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}=\boldsymbol{y}}$ and so proposing updates to $\mathbf{x}$ from the latter will often lead to proposed values for $(\mathbf{x}, \mathbf{z})$ with a very low acceptance probability. The ABC MCMC Metropolis–Hastings scheme can also be considered an instance of a pseudo-marginal MCMC method [5, 18] where such sticking artifacts are also a well known problem [154].

4.7 INFERENCE IN THE INPUT SPACE

We now consider reposing the inference problem in terms of the input variables to the generative models, introduce in Section **??**. Assuming for now only that the generative model has inputs with a distribution P with a known density ρ with respect to the Lebesgue measure[6], but not yet requiring any of the other conditions specified for differentiable generative models, we have from (**??**) and basic properties of the expectation that

$$\mathbb{E}\left[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\right] = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}\left[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathbf{x})\right] \tag{4.24}$$

$$= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}\left[f(\mathrm{g}_{\mathbf{z}}(\mathbf{u}))\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}}(\mathbf{u}))\right] \tag{4.25}$$

$$= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\int_U f \circ \mathrm{g}_{\mathbf{z}}(\boldsymbol{u})\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}. \tag{4.26}$$

Crucially this reparameterisation takes the form of an integral of a function $f \circ \mathrm{g}_{\mathbf{z}}$ against an *explicit* probability density

$$\pi_\epsilon(\boldsymbol{u}) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}}(\boldsymbol{u}))\, \rho(\boldsymbol{u}), \tag{4.27}$$

that we can evaluate up to an unknown normalising constant $\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})$. This is the typical setting for approximate inference in explicit probabilistic models, and so is straight away amenable to applying standard variants of methods such as MCMC and variational inference. In the common special case (and typical ABC setting) of a directed generative

6 This is for concreteness of notation rather than a modelling restriction and the approach can easily be generalised to more general distributions.

model with a tractable marginal density on the unobserved variables $\mathrm{p}_{\mathbf{z}}$, using the notation introduced in Section (??) we have that

$$\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big] = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}\big[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z},\, \mathbf{u}_2))\big] \tag{4.28}$$

$$= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_Z \int_{U_2} f(z)\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}|\mathbf{z}}(z,\, \boldsymbol{u}_2))\, \mathrm{p}_{\mathbf{z}}(z) \rho_2(\boldsymbol{u}_2)\, \mathrm{d}\boldsymbol{u}_2\, \mathrm{d}z \tag{4.29}$$

with now the explicit target density for inference being

$$\pi_\epsilon(z, \boldsymbol{u}_2) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}|\mathbf{z}}(z,\, \boldsymbol{u}_2))\, \mathrm{p}_{\mathbf{z}}(z)\, \rho_2(\boldsymbol{u}_2). \tag{4.30}$$

Rather than defining a MCMC chain jointly updating all of the random inputs $\mathbf{u}$ this formulation suggests a possible alternative approach of defining a chain on $(\mathbf{z}, \mathbf{u}_2)$ and alternating transition operators updating $\mathbf{z}$ while leaving $\mathbf{u}_2$ fixed and updating $\mathbf{u}_2$ while leaving $\mathbf{z}$ fixed. We will revisit this reparameterisation in the context of a proposed slice-sampling approach for inference in the next chapter.

In the reparameterising inference in terms of evaluating an integral over the input space we have still so far required the definition of a kernel k_ϵ and tolerance ϵ and the integral being estimated is the approximate expectation (??) rather than target conditional expectation $\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{x}\big]$ we are directly interested in. We now consider in the specific case of differentiable generative models how to perform inference without introducing an ABC kernel.

We begin an initial intuition for the approach, by considering taking the limit of $\epsilon \to 0$ in the integral (4.24) corresponding to evaluating the ABC conditional expectation in the generator input space. We previously showed in (4.5) that the approximate expectation $\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon\big]$ converges as $\epsilon \to 0$ to the conditional expectation of interest $\mathbb{E}\big[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}\big]$, providing that the implicit distribution of the observed variables in the generative model $\mathrm{P}_{\mathbf{x}}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathrm{p}_{\mathbf{x}}$. Informally we can consider that for kernels meeting the conditions (4.4) and (4.5), in the limit of $\epsilon \to 0$ the

kernel density terms $k_\epsilon(\boldsymbol{y};\, \mathbf{g}_\mathbf{x}(\boldsymbol{u}))$ tend to Dirac deltas $\delta(\boldsymbol{y} - \mathbf{g}_\mathbf{x}(\boldsymbol{u}))$ and so

$$\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{y}\big] = \lim_{\epsilon \to 0} \mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon\big] \tag{4.31}$$

$$\simeq \frac{\int_U f \circ \mathbf{g}_\mathbf{z}(\boldsymbol{u})\, \delta(\boldsymbol{y} - \mathbf{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}{\int_U \delta(\boldsymbol{y} - \mathbf{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}. \tag{4.32}$$

The Dirac delta term restricts the integral across the input space U to an embedded, $M - N_\mathbf{x}$ dimensional, implicitly-defined manifold corresponding to the pre-image under $\mathbf{g}_\mathbf{x}$ of $\boldsymbol{y}$, $\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{y}] \equiv \{\boldsymbol{u} \in U : \mathbf{g}_\mathbf{x}(\boldsymbol{u}) = \boldsymbol{y}\}$. It is not necessarily immediately clear however how to define the required density on that manifold for abitrary non-injective $\mathbf{g}_\mathbf{x}$.

In differentiable generative models (i.e. a model meeting the conditions stated in Section **??**) we can however use a derivation similar to that given by Diaconis, Holmes and Shahshahani in [55] for the conditional density on a manifold to find an expression for the conditional expectation consistent with definition given earlier in (**??**). Our derivation is largely a restatement of that given in [55] except for the minor difference of working in terms of conditional expectations rather densities. The key result we will use is a formula from geometric measure-theory, Federer's *co-area formula* [66, §3.2.12].

The K-dimensional Hausdorff measure $\hbar^K$ on $\mathbb{R}^N$ for $K \in \mathbb{N}$, $0 < K < N$ formalises a measure of the 'volume' of K-dimensional submanifolds of $\mathbb{R}^N$ - e.g. for $K = 1$ it corresponds to the length of a curve in $\mathbb{R}^N$. Additionally $\hbar^N = \lambda^N$ and $\hbar^0 = \#$.

THEOREM 4.1 (Co-area formula): *Let $V \subseteq \mathbb{R}^L$ and $W \subseteq \mathbb{R}^K$ with $L \geq K$, $\mathbf{m} : V \to W$ be Lipschitz and $\mathrm{h} : V \to \mathbb{R}$ be Lebesgue measurable. Then*

$$\int_V \mathrm{h}(\boldsymbol{v})\, D_\mathbf{m}(\boldsymbol{v})\, \mathrm{d}\lambda^L(\boldsymbol{v}) = \int_W \int_{\mathbf{m}^{-1}[\boldsymbol{w}]} \mathrm{h}(\boldsymbol{v})\, \mathrm{d}\hbar^{L-K}(\boldsymbol{v})\, \mathrm{d}\lambda^K(\boldsymbol{w}) \tag{4.33}$$

with $\hbar^{L-K}$ denoting the $L-K$-dimensional Hausdorff measure and $D_\mathbf{m}(\boldsymbol{v})$ denoting the generalised Jacobian determinant

$$D_\mathbf{m}(\boldsymbol{v}) \equiv \left| \frac{\partial \mathbf{m}}{\partial \boldsymbol{u}} \frac{\partial \mathbf{m}}{\partial \boldsymbol{u}}^\mathsf{T} \right|^{\frac{1}{2}}. \tag{4.34}$$

More immediately applicable in our case is the following corollary.

COROLLARY 4.1: *If Q is a probability measure on V with density q with respect to the Lebesgue measure λ^L and $\mathbf{J}_\mathbf{m}$ is full row-rank Q-almost everywhere, then for Lebesgue measurable* $\mathrm{h}' : V \to \mathbb{R}$

$$\int_V \mathrm{h}'(\boldsymbol{v})\, q(\boldsymbol{v})\, \mathrm{d}\lambda^L(\boldsymbol{v}) = \int_W \int_{\mathbf{m}^{-1}[\boldsymbol{w}]} \mathrm{h}'(\boldsymbol{v})\, q(\boldsymbol{v})\, D_\mathbf{m}(\boldsymbol{v})^{-1}\, \mathrm{d}\hbar^{L-K}(\boldsymbol{v})\, \mathrm{d}\lambda^K(\boldsymbol{w}). \tag{4.35}$$

This can be shown by setting $h(\boldsymbol{v}) = h'(\boldsymbol{v})\, q(\boldsymbol{v}) D_\mathbf{f}(\boldsymbol{v})^{-1}$ in (4.33) and using the equivalence of Lebesgue integrals in which the integrand differs only zero-measure sets.

We first show that $\mathsf{P}_\mathbf{x}$ has a density $\mathrm{p}_\mathbf{x} = \frac{\mathrm{d}\mathsf{P}_\mathbf{x}}{\mathrm{d}\lambda^{N_\mathbf{x}}}$.

PROPOSITION 4.1 (Change of variables in a differentiable generative model): *For a differentiable generative model $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_\mathbf{x}, \mathbf{g}_\mathbf{z})$ as defined in Definition 4.2, then if the generator Jacobian $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ is Lipschitz and has full row-rank $\mathsf{P}_\mathbf{u}$-almost everywhere, the observed vector $\mathbf{x}$ has a density with respect to the Lebesgue measure satisfying*

$$\mathrm{p}_\mathbf{x}(\boldsymbol{x}) = \int_{\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]} \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_\mathbf{x}}(\boldsymbol{u}) \qquad \forall \boldsymbol{x} \in X. \tag{4.36}$$

Proof. From Definition 4.2 we have that $\mathbf{x} = \mathbf{g}_\mathbf{x}(\mathbf{u})$ and $\frac{\mathrm{d}\mathsf{P}_\mathbf{u}}{\mathrm{d}\lambda^M} = \rho$ and so

$$\mathsf{P}_\mathbf{x}(A) = \int_U \mathbb{1}_A \circ \mathbf{g}_\mathbf{x}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, \mathrm{d}\lambda^M(\boldsymbol{u}) \qquad \forall A \in \mathcal{G}.$$

As the generator Jacobian $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}}$ is Lipschitz and has full row-rank $\mathsf{P}_\mathbf{u}$-almost everywhere we can apply Corollary 4.1, and so we have that $\forall A \in \mathcal{G}$

$$\mathsf{P}_\mathbf{x}(A) = \int_X \int_{\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]} \mathbb{1}_A \circ \mathbf{g}_\mathbf{x}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_\mathbf{x}}(\boldsymbol{u})\, \mathrm{d}\lambda^{N_\mathbf{x}}(\boldsymbol{x}).$$

The term $\mathbb{1}_A \circ \mathbf{g}_\mathbf{x}(\boldsymbol{u})$ inside the inner integral is equal to $\mathbb{1}_A(\boldsymbol{x})$ across all points in the space $\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]$ being integrated across and so can be taken outside the inner integral to give

$$\begin{aligned}\mathsf{P}_\mathbf{x}(A) &= \int_X \mathbb{1}_A(\boldsymbol{x}) \int_{\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]} \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_\mathbf{x}}(\boldsymbol{u})\, \mathrm{d}\lambda^{N_\mathbf{x}}(\boldsymbol{x}) \\ &= \int_A \int_{\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]} \rho(\boldsymbol{u})\, D_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_\mathbf{x}}(\boldsymbol{u})\, \mathrm{d}\lambda^{N_\mathbf{x}}(\boldsymbol{x}).\end{aligned}$$

By definition the density $\mathrm{p}_{\mathbf{x}}$ of a probability measure $\mathrm{P}_{\mathbf{x}}$ with respect to the Lebesgue measure $\lambda^{N_{\mathbf{x}}}$ satisfies

$$\mathrm{P}_{\mathbf{x}}(A) = \int_A \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{d}\lambda^{N_{\mathbf{x}}}(\boldsymbol{x}) \qquad \forall A \in \mathcal{G}$$

$\therefore$ $\mathrm{P}_{\mathbf{x}}$ has a density corresponding to (4.36) with respect to $\lambda^{N_{\mathbf{x}}}$. □

This is a generalisation of the change of variables formula under a diffeomorphism encountered previously in Chapter 1. We now derive a result for the conditional expectation.

PROPOSITION 4.2 (Conditional expectations in a differentiable generative model): *For a differentiable generative model* $(U, \mathcal{F}, \rho, \mu, \mathbf{g}_{\mathbf{x}}, \mathbf{g}_{\mathbf{z}})$ *as defined in Definition 4.2, then if the generator Jacobian* $\frac{\partial \mathbf{g}_{\mathbf{x}}}{\partial \boldsymbol{u}}$ *is Lipschitz and has full row-rank* $\mathrm{P}_{\mathbf{u}}$*-almost everywhere, then for Lebesgue measurable functions* $f : X \to \mathbb{R}$ *and* $\boldsymbol{x} \in X$ *such that* $\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) > 0$ *we have that*

$$\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{x}\big] = \frac{1}{\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})} \int_{\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_{\mathbf{x}}}(\boldsymbol{u}). \tag{4.37}$$

Proof. Restating the general definition for a conditional expectation from Chapter 1, we need to find a measurable function $\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{x}\big] : X \to \mathbb{R}$ which $\forall A \in \mathcal{G}$ satisfies

$$\int_A \mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{x} = \boldsymbol{x}\big]\, \mathrm{dP}_{\mathbf{x}}(\boldsymbol{x}) = \int_{A \times Z} f(z)\, \mathrm{dP}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, z),$$

with this uniquely defining the conditional expectation up to $\mathrm{P}_{\mathbf{x}}$-null sets. Using $\mathbf{x} = \mathbf{g}_{\mathbf{x}}(\mathbf{u})$, $\mathbf{z} = \mathbf{g}_{\mathbf{z}}(\mathbf{u})$ and $\mathrm{p}_{\mathbf{u}} = \rho$ we have that $\forall A \in \mathcal{G}$

$$\int_{A \times Z} f(z)\, \mathrm{dP}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x}, z) = \int_U \mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})\, f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, \mathrm{d}\lambda^M(\boldsymbol{u}).$$

Applying the co-area corollary (4.35) to the right-hand side and again noting the indicator term $\mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})$ is constant across the space being integrated on, we have that $\forall A \in \mathcal{G}$

$$\begin{aligned}
&\int_{A\times Z} f(\boldsymbol{z})\, \mathrm{dP}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x},\boldsymbol{z}) \\
&= \int_X \int_{\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} \mathbb{1}_A \circ \mathbf{g}_{\mathbf{x}}(\boldsymbol{u})\, f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_{\mathbf{x}}}(\boldsymbol{u})\, \mathrm{d}\lambda^{N_{\mathbf{x}}}(\boldsymbol{x}) \\
&= \int_A \int_{\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_{\mathbf{x}}}(\boldsymbol{u})\, \mathrm{d}\lambda^{N_{\mathbf{x}}}(\boldsymbol{x}).
\end{aligned}$$

Finally using that $\mathrm{P}_{\mathbf{x}}$ has a density $\mathrm{p}_{\mathbf{x}}$ with respect to the Lebesgue measure as shown in the previous proposition, we have that

$$\begin{aligned}
&\int_{A\times Z} f(\boldsymbol{z})\, \mathrm{dP}_{\mathbf{x},\mathbf{z}}(\boldsymbol{x},\boldsymbol{z}) = \\
&\qquad \int_A \frac{1}{\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})} \int_{\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]} f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \rho(\boldsymbol{u})\, D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \mathrm{d}\hbar^{M-N_{\mathbf{x}}}(\boldsymbol{u})\, \mathrm{dP}_{\mathbf{x}}(\boldsymbol{x}).
\end{aligned}$$

Note that as we are integrating against the probability measure $\mathrm{P}_{\mathbf{x}}$ we can safely ignore the points for which $\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = 0$ as the set of all such points naturally has zero measure under $\mathrm{P}_{\mathbf{x}}$ and so does not contribute to integral. Comparing to the definition of the conditional expectation we have that (4.37) satisfies the definition. □

The expression derived for the conditional expectation has the form of an integral of function $f \circ \mathbf{g}_{\mathbf{z}}$ integrated against a density

$$\pi(\boldsymbol{u}) = \frac{1}{\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})} D_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{-1}\, \rho(\boldsymbol{u}) \tag{4.38}$$

which we can evaluate upto an unknown normalising constant $\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})$. The key complicating factor is that the integral is now not across a Euclidean space, but an implictly defined manifold corresponding to the pre-image $\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$. However if we can construct a Markov transition operator which has an invariant distribution with density (4.38) with respect to the Hausdorff measure on the manifold, then we can use samples of the chain states $\{\mathbf{u}^{(s)}\}_{s=1}^{S}$ to compute an estimate

$$\hat{\mathrm{f}}_S = \frac{1}{S} \sum_{s=1}^{S} \left(\mathrm{f} \circ \mathbf{g}_{\mathbf{z}}\left(\mathbf{u}^{(s)}\right) \right) \tag{4.39}$$

which providing the chain is also aperiodic and irreducible by the standard MCMC law of large numbers argument will be a consistent estimator for $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{x}]$.

Although constructing a Markov transition operator with the required properies is non-trivial, there is a significant body of existing work on methods for defining Markov chains on manifolds. We propose here to use a constrained Hamiltonian Monte Carlo method.

4.8 CONSTRAINED HAMILTONIAN MONTE CARLO

HMC [61, 162] is an auxiliary variable MCMC method which uses the gradient of the density of the target distribution of interest within a simulated Hamiltonian dynamic. The vector variable of interest $\mathbf{u}$ is augmented with a momentum variable $\mathbf{p} \in \mathbb{R}^M$. The momentum is taken to be independently Gaussian distributed with zero mean and covariance $\boldsymbol{M}$, often called the mass matrix. The negative logarithm of the density π of the target distribution is termed the potential energy $\phi(\boldsymbol{u}) = -\log \pi(\boldsymbol{u})$. The joint distribution on $\mathbf{u}$ and $\mathbf{p}$ then has a density which is proportional to $\exp(-H(\boldsymbol{u}, \boldsymbol{p}))$ where the Hamiltonian $H(\boldsymbol{u}, \boldsymbol{p})$ is defined as

$$H(\boldsymbol{u}, \boldsymbol{p}) = \phi(\boldsymbol{u}) + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p}. \tag{4.40}$$

The canonical Hamiltonian dynamic is described by the system of ordinary differential equations

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{p}} = \boldsymbol{M}^{-1}\boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial H}{\partial \boldsymbol{u}} = -\nabla\phi(\boldsymbol{u}). \tag{4.41}$$

This dynamic is time-reversible, volume-preserving and exactly conserves the Hamiltonian. Symplectic integrators allow approximate integration of the Hamiltonian flow while maintaining the time-reversibility and volume-preservation properties. Subject to stability bounds on the time-step, such integrators will exactly conserve some 'nearby' Hamiltonian, and so the change in the Hamiltonian will tend to remain small even over long simulated trajectories [124].

These properties make simulated Hamiltonian dynamics an ideal proposal mechanism for a Metropolis MCMC method. The Metropolis accept ratio for a proposal $(\boldsymbol{u}_{\mathrm{p}}, \boldsymbol{p}_{\mathrm{p}})$ generated by simulating the dynamic N_s time steps forward from $(\boldsymbol{u}, \boldsymbol{p})$ with a symplectic integrator and

then negating the momentum, is simply $\exp\Big(H(\boldsymbol{u}, \boldsymbol{p}) - H(\boldsymbol{u}_\mathrm{p}, \boldsymbol{p}_\mathrm{p})\Big)$. Typically the change in the Hamiltonian will be small and so the probability of acceptance high. To ensure ergodicity, dynamic moves can be interspersed with updates independently sampling a new momentum from $\mathcal{N}(\mathbf{0}, \boldsymbol{M})$.

In our case the system is subject to a constraint of the form

$$\mathbf{g}_\mathbf{x}(\boldsymbol{u}) - \boldsymbol{x} = \mathbf{0}. \tag{4.42}$$

By introducing Lagrangian multipliers λ_i for each of the $N_\mathbf{x}$ constraints, the Hamiltonian for a constrained system can be written as

$$H(\boldsymbol{u}, \boldsymbol{p}) = \phi(\boldsymbol{u}) + \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{M}^{-1}\boldsymbol{p} + (\mathbf{g}_\mathbf{x}(\boldsymbol{u}) - \boldsymbol{x})^\mathsf{T}\boldsymbol{\lambda}, \tag{4.43}$$

and a corresponding constrained Hamiltonian dynamic

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{M}^{-1}\boldsymbol{p}, \quad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\nabla\phi(\boldsymbol{u}) - \mathbf{J}_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})^\mathsf{T}\boldsymbol{\lambda}, \tag{4.44}$$

$$\text{subject to } \mathbf{g}_\mathbf{x}(\boldsymbol{u}) - \boldsymbol{x} = \mathbf{0}, \quad \mathbf{J}_{\mathbf{g}_\mathbf{x}}(\boldsymbol{u})\boldsymbol{M}^{-1}\boldsymbol{p} = \mathbf{0}. \tag{4.45}$$

A popular numerical integrator for simulating constrained Hamiltonian dynamics is RATTLE [3] (and the algebraically equivalent SHAKE [200] scheme). This a natural generalisation of the Störmer-Verlet (leapfrog) integrator typically used in standard HMC with additional projection steps in which the Lagrange multipliers $\boldsymbol{\lambda}$ are solved for to satisfy the conditions (4.45). RATTLE and SHAKE maintain the properties of being time-reversible, volume-preserving and symplectic [122].

The use of constrained dynamics in HMC has been proposed several times. In the molecular dynamics literature, both [95] and [125] suggest using a simulated constrained dynamic within a HMC framework to estimate free-energy profiles.

Most relevantly here [36] proposes using a constrained HMC variant to perform inference in distributions defined on implicitly defined embedded non-linear manifolds. This gives sufficient conditions on ρ, $\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]$ and $\mathbf{g}_\mathbf{x}$ for the transition operator have the distribution corresponding to the target density as an invariant distribution and to be aperiodic and irreducible. In particular in our case it is sufficient for ρ to be C^2 continuous, $\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]$ to be a connected smooth and differentiable manifold and $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ has full row-rank everywhere. These are stricter than our

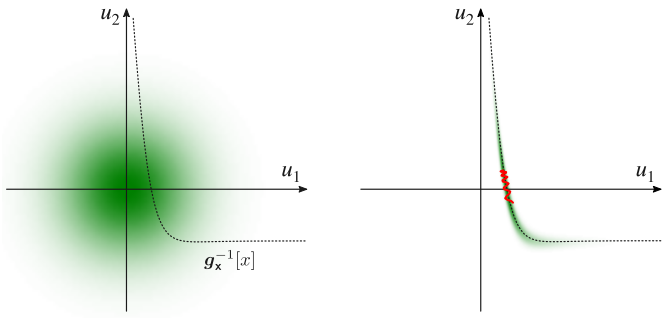


Figure 4.4.: Illustration of oscillatory behaviour in HMC trajectories when using an ABC target density (4.27) in the input space to a generative model. The left axis shows the two-dimensional input space U of a toy differentiable generative model with a Gaussian input density ρ (green shading). The dashed curve shows the one-dimensional manifold corresponding to the pre-image under the generator function $\mathbf{g}_\mathbf{x}$ of an observed output x. The right axis shows the same input space with now the green shading showing the density proportional to $k_\epsilon(x; \mathbf{g}_\mathbf{x}(\boldsymbol{u}))\, \rho(\boldsymbol{u})$ with a Gaussian k_ϵ. The red curve shows a simulated HMC trajectory using this density as the target distribution: the large magnitude density gradients normal to the manifold cause high-frequency oscillations and slows movement along the manifold (which corresponds to variation in the latent variable z).

initial definition of a differentiable generative model. The requirement for C^2 continuity of ρ requires the second-derivatives of the generator function $\mathbf{g}_\mathbf{x}$ to also exist and be continuous, which should generally be feasible to check by analysing the computation graph of the generator.

The requirement for $\mathbf{g}_\mathbf{x}^{-1}[\boldsymbol{x}]$ to be connected will generally be much more difficult to verify. If the pre-image consists of multiple disconnected components then the dynamic will generally remain confined to just one of them. Although problematic, this issue is similar to that faced by most MCMC methods in target distributions which potentially have multiple separated modes. Defining an augmented generator $\mathbf{g}_\mathbf{y}(\mathbf{u}, \mathbf{n}) = \mathbf{g}_\mathbf{x}(\mathbf{u}) + \epsilon\mathbf{n}$ with $\mathbf{n}$ a vector of $N_\mathbf{x}$ independent zero-mean unit-variance Gaussian random variables and ϵ a small constant and then performing constrained HMC on the augmented pair $(\mathbf{u}, \mathbf{n})$ will guarantee that the manifold $\mathbf{g}_\mathbf{y}^{-1}[\boldsymbol{x}]$ is connected and $\mathbf{J}_{\mathbf{g}_\mathbf{y}}$ is full row-rank everywhere. Of course this is equivalent to the ABC approach with a ϵ tolerance Gaussian kernel, and using our earlier observation we could alternatively perform standard HMC in the input space using (4.27) as the target density.

If ϵ is small however the high gradients in the target density normal to the tangent space of the manifold $\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$ will tend to lead to a small integrator step size needing to be used to maintain reasonable accept rates and the simulated trajectories tend to exhibit high frequency oscillations as illustrated in Figure 4.4. We have found in some cases that applying constrained HMC with the Gaussian augmented generator $\mathbf{g}_{\mathbf{y}}$ can therefore still be more efficient than running standard HMC in the ABC target density, despite the much higher per-step costs, as constrained HMC updates are able to make much larger steps, particularly when using small ϵ. The constrained HMC dynamic exploits more information about the geometry of the target distribution by using the Jacobian $\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}$ which describes the tangent space of the manifold. A related approach would be to use a Riemannian-manifold HMC [83] method with a position-dependent metric $\boldsymbol{M}(\boldsymbol{u}) = \mathbf{J}_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})\mathbf{J}_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})^{\mathsf{T}} + \epsilon^2\mathbf{I}$ when using a Gaussian ABC kernel in the input space (equation (4.27)); this should dynamically adjust the momentum scaling so as to reduce the inefficient oscillatory behaviour seen in the standard (fixed metric) HMC trajectories [21].

4.9 METHOD

Our constrained HMC implementation is shown in Algorithm 10. We use a generalisation of the RATTLE scheme to simulate the dynamic. The inner updates of the state to solve for the geodesic motion on the constraint manifold are split into multiple smaller steps, which is a special case of the scheme described in [123]. This allows more flexibility in choosing an appropriately small step-size to ensure convergence of the iterative solution of the equations projecting on to the constraint manifold while still allowing a more efficient larger step size for updates to the momentum due to the negative log density gradient. We have assumed $\boldsymbol{M} = \mathbf{I}$ here; other mass matrix choices can be equivalently implemented by adding an initial linear transformation stage in the generator.

Each inner geodesic time-step involves stepping along the current momentum $\tilde{\boldsymbol{u}} \leftarrow \boldsymbol{u} + (\delta t/N_g)\boldsymbol{p}$ and then projecting $\tilde{\boldsymbol{u}}$ back on to $\mathbf{g}_{\mathbf{x}}^{-1}[\boldsymbol{x}]$ by solving for $\boldsymbol{\lambda}$ which satisfy $\mathbf{g}_{\mathbf{x}}(\tilde{\boldsymbol{u}} - \boldsymbol{J}^{\mathsf{T}}\boldsymbol{\lambda}) = \boldsymbol{x}$ where $\boldsymbol{J} = \mathbf{J}_{\mathbf{g}_{\mathbf{x}}}(\boldsymbol{u})$. This is performed in the function ProjectPos in Algorithm 10. Here we use

Algorithm 10 Constrained Hamiltonian Monte Carlo

Input:

$\mathbf{g_x}$: observed variable generator function;
ϕ : potential energy function $\phi(\boldsymbol{u}) = -\log \rho(\boldsymbol{u}) + \frac{1}{2}\log|\mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u})\mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u})|$;
$\boldsymbol{x}$: observed data values being conditioned on;
$\boldsymbol{u}$: current chain state (model inputs) with $\|\mathbf{g_x}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$;
$(\varphi, \boldsymbol{J}, \boldsymbol{L})$: cached values of ϕ, $\mathbf{J}_{\mathbf{g_x}}$ and $\text{chol}\left(\mathbf{J}_{\mathbf{g_x}}\mathbf{J}_{\mathbf{g_x}}^\mathsf{T}\right)$ evaluated at $\boldsymbol{u}$;
ϵ : convergence tolerance for Newton iteration;
M : number of Newton iterations to try before rejecting for non-convergence;
δt : integrator time step;
N_s : number of time steps to simulate;
N_g : number of geodesic steps per time step.

Output:

$\boldsymbol{u}_\text{n}$: new chain state with $\|\mathbf{g_x}(\boldsymbol{u}_\text{n}) - \boldsymbol{x}\|_\infty < \epsilon$;
$(\varphi_\text{n}, \boldsymbol{J}_\text{n}, \boldsymbol{L}_\text{n})$: values of ϕ, $\mathbf{J}_{\mathbf{g_x}}$ and $\text{chol}\left(\mathbf{J}_{\mathbf{g_x}}\mathbf{J}_{\mathbf{g_x}}^\mathsf{T}\right)$ evaluated at new $\boldsymbol{u}_\text{n}$.

$\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
$\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\boldsymbol{n}, \boldsymbol{J}, \boldsymbol{L})$
$\boldsymbol{u}_\text{p}, \boldsymbol{p}_\text{p}, \boldsymbol{J}_\text{p}, \boldsymbol{L}_\text{p} \leftarrow \textsc{SimDyn}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
$\varphi_\text{p} \leftarrow \phi(\boldsymbol{u})$
$r \sim \mathcal{U}(0, 1)$
$p_a \leftarrow \exp\left(\varphi + \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{p} - \varphi_\text{p} - \frac{1}{2}\boldsymbol{p}_\text{p}^\mathsf{T}\boldsymbol{p}_\text{p}\right)$
if $r < p_a$ **then**
  $\boldsymbol{u}_\text{n}, \varphi_\text{n}, \boldsymbol{J}_\text{n}, \boldsymbol{L}_\text{n} \leftarrow \boldsymbol{u}_\text{p}, \varphi_\text{p}, \boldsymbol{J}_\text{p}, \boldsymbol{L}_\text{p}$
else
  $\boldsymbol{u}_\text{n}, \varphi_\text{n}, \boldsymbol{J}_\text{n}, \boldsymbol{L}_\text{n} \leftarrow \boldsymbol{u}, \varphi, \boldsymbol{J}, \boldsymbol{L}$

function $\textsc{SimDyn}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \frac{\delta t}{2}\nabla\phi(\boldsymbol{u})$
  $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
  $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L} \leftarrow \textsc{SimGeo}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  for $s \in \{2 \ldots N_s\}$ **do**
    $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \delta t\nabla\phi(\boldsymbol{u})$
    $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L} \leftarrow \textsc{SimGeo}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  $\tilde{\boldsymbol{p}} \leftarrow \boldsymbol{p} - \frac{\delta t}{2}\nabla\phi(\boldsymbol{u})$
  $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
  return $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L}$

function $\textsc{ProjectMom}(\boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  return $\boldsymbol{p} - \boldsymbol{J}^\mathsf{T}\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{J}\boldsymbol{p}$

function $\textsc{ProjectPos}(\boldsymbol{u}, \boldsymbol{J}, \boldsymbol{L})$
  $\boldsymbol{\delta} \leftarrow \mathbf{g_x}(\boldsymbol{u}) - \boldsymbol{x}$
  $i \leftarrow 0$
  while $\|\boldsymbol{\delta}\|_\infty > \epsilon \wedge i < M$ **do**
    $\boldsymbol{u} \leftarrow \boldsymbol{u} - \boldsymbol{J}^\mathsf{T}\boldsymbol{L}^{-\mathsf{T}}\boldsymbol{L}^{-1}\boldsymbol{\delta}$
    $\boldsymbol{\delta} \leftarrow \mathbf{g_x}(\boldsymbol{u}) - \boldsymbol{x}$
    $i \leftarrow i + 1$
  if $i = M$ **then**
    raise RejectMove
  return $\boldsymbol{u}$

function $\textsc{SimGeo}(\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L})$
  for $i \in \left\{1 \ldots N_g\right\}$ **do**
    $\tilde{\boldsymbol{u}} \leftarrow \boldsymbol{u} + \frac{\delta t}{N_g}\boldsymbol{p}$
    $\boldsymbol{u}' \leftarrow \textsc{ProjectPos}(\tilde{\boldsymbol{u}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{J} \leftarrow \mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u}')$
    $\boldsymbol{L} \leftarrow \text{chol}\left(\boldsymbol{J}\boldsymbol{J}^\mathsf{T}\right)$
    $\tilde{\boldsymbol{p}} \leftarrow \frac{N_g}{\delta t}(\boldsymbol{u}' - \boldsymbol{u})$
    $\boldsymbol{p} \leftarrow \textsc{ProjectMom}(\tilde{\boldsymbol{p}}, \boldsymbol{J}, \boldsymbol{L})$
    $\boldsymbol{u}_r \leftarrow \boldsymbol{u}' - \frac{\delta t}{N_g}\boldsymbol{p}$
    $\boldsymbol{u}_r \leftarrow \textsc{ProjectPos}(\boldsymbol{u}_r, \boldsymbol{J}, \boldsymbol{L})$
    if $\|\boldsymbol{u} - \boldsymbol{u}_r\|_\infty > \sqrt{\epsilon}$ **then**
      raise RejectMove
    $\boldsymbol{u} \leftarrow \boldsymbol{u}'$
  return $\boldsymbol{u}, \boldsymbol{p}, \boldsymbol{J}, \boldsymbol{L}$

a quasi-Newton method for solving the system of equations. The true Newton update would be

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^\mathsf{T}\left(\mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u}')\boldsymbol{J}^\mathsf{T}\right)^{-1}(\mathbf{g_x}(\boldsymbol{u}') - \boldsymbol{x}). \tag{4.46}$$

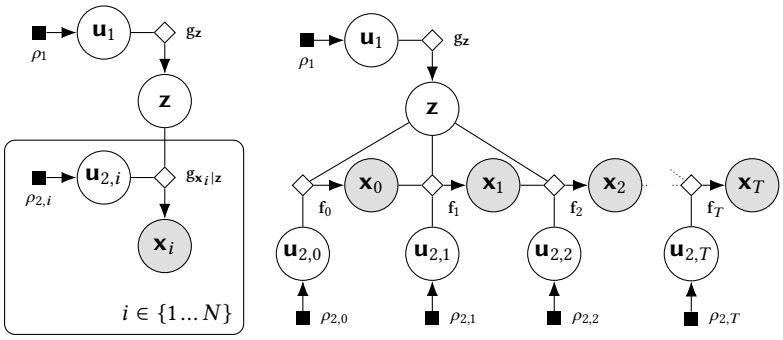
This requires recalculating the Jacobian and solving a dense linear system within the optimisation loop. Instead we use a symmetric quasi-Newton update,

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^{\mathsf{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}\right)^{-1}(\mathrm{g}_{\mathbf{x}}(\boldsymbol{u}') - \boldsymbol{x}). \tag{4.47}$$

as proposed in [13]. The Jacobian product $\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}$ evaluated at the previous state is used to condition the moves. This matrix is positive-definite and a Cholesky decomposition can be calculated outside the optimisation loop allowing cheaper quadratic cost solves within the loop.

Convergence of the quasi-Newton iteration is signalled when $\|\mathrm{g}_{\mathbf{x}}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$, i.e. the elementwise maximum absolute difference between the model observed output and the observed data is below a tolerance ϵ. The tolerance is analogous to the ϵ parameter in ABC methods, however here we can set this value close to machine precision (with $\epsilon = 10^{-8}$ in the experiments) and so the error introduced is comparable to that otherwise incurred for using non-exact arithmetic.

In some cases the quasi-Newton iteration will fail to converge. We use a fixed upper limit on the number of iterations and reject the move (line 34 in Algorithm 10) if convergence is not achieved within this limit. To ensure reversibility, once we have solved for a forward geodesic step on the manifold in SimGeo, we then check if the corresponding reverse step (with the momentum negated) returns to the original position. This involves running a second Newton iteration, though as it reuses the same Jacobian $\boldsymbol{J}$ and Cholesky decomposition $\boldsymbol{L}$, the evaluation of which tend to be the dominant costs in the algorithm, in the experiments we found the overhead introduced tended to be quite small (around a 20% increase in run-time compared to only performing the forward step). A similar scheme for ensuring reversibility is proposed in [238]. The square root of the tolerance ϵ used for the initial Newton convergence check *in the output space of generator* (line 29 in Algorithm 10) is used for the reverse-step check on *the inputs* (line 48 in Algorithm 10) based on standard recommendations for checking convergence in optimisation routines [45]. In the implementation we used in the experiments, we fall back to a MINPACK [150] implementation of the robust Powell's Hybrid method [178] if the quasi-Newton iteration diverges or fails to converge, with a rejection then only occurring if both iterative solvers fail. In practice we found if the step size δt and number of



(a) Independent $\mathbf{x}_i$ (b) Markovian $\mathbf{x}_i$

Figure 4.5.: Factor graphs of examples of structured directed generative models.

geodesic steps N_g is chosen appropriately then rejections due to non-convergence / non-reversible steps occur very rarely.

For larger systems, the Cholesky decomposition of the constraint Jacobian matrix product $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}$ (line 42) will become a dominant cost, generally scaling cubically with $N_\mathbf{x}$. In many models however conditional independency structure will mean that not all observed variables $\mathbf{x}$ are dependent on all of the input variables $\mathbf{u}$ and so the Jacobian $\mathbf{J}_{\mathbf{g}_\mathbf{x}}$ has sparsity structure which can be exploited to reduce this worst-case cost.

In particular two common cases are directed generative models in which the observed variables $\mathbf{x}$ can be split into groups $\{\mathbf{x}_i\}_{i=1}^G$ such that all of the $\mathbf{x}_i$ are either conditionally independent given the latent variables $\mathbf{z} = \mathbf{g}_\mathbf{z}(\mathbf{u}_1)$ (for example independent and identically distributed observations), or each $\mathbf{x}_i$ is conditionally independent of all $\{\mathbf{x}_j\}_{j<i-1}$ given $\mathbf{x}_{i-1}$ and $\mathbf{z}$ (most commonly Markov chains for example from a SDE model though observations with more general tree structured dependencies can also be ordered into this form). Figure 4.5 shows factor graphs for directed generative models with these two structures, with the conditional independencies corresponding to each $\mathbf{x}_i$ being generated as a function of only a subset $\mathbf{u}_{2,i}$ of the random input variables $\mathbf{u}_2$. Equivalently these structures correspond to generator functions $\mathbf{g}_\mathbf{x}$ which can be expressed in one of the two forms

$$\mathbf{x}_i = \mathbf{g}_{\mathbf{x}_i|\mathbf{z}}(\mathbf{z}, \mathbf{u}_{2,i}) \qquad \text{(independent)} \qquad (4.48)$$

$$\mathbf{x}_i = \mathbf{f}_i(\mathbf{z}, \mathbf{x}_{i-1}, \mathbf{u}_{2,i}) = \mathbf{g}_{\mathbf{x}_i|\mathbf{z}}\left(\mathbf{z}, \{\mathbf{u}_{2,j}\}_{j=1}^i\right) \qquad \text{(Markov).} \qquad (4.49)$$

For models with these structures the generator Jacobian

$$\mathbf{J}_{\mathbf{g}_\mathbf{x}} = \left[\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1} \middle| \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}\right] \tag{4.50}$$

has a component $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$ which is either block-diagonal (independent) or block-triangular (Markovian). Considering first the simplest case where each $(\mathbf{x}_i, \mathbf{u}_{2,i})$ pair are single dimensional, the Cholesky decomposition of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T} = \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}{}^\mathsf{T} + \frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}{}^\mathsf{T}$ can then be computed by low-rank Cholesky updates of the triangular / diagonal matrix $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$ with each of the columns of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}$. As $\dim(\boldsymbol{u}_1) = L$ is often significantly less than the number of observations being conditioned on $N_\mathbf{x}$, the resulting $O(LN_\mathbf{x}^2)$ cost of the low-rank Cholesky updates is a significant improvement over the original $O(N_\mathbf{x}^3)$. For cases in which each $(\mathbf{x}_i, \mathbf{u}_{2,i})$ pair are both vectors of dimension D (i.e. $N_\mathbf{x} = GD$) and so $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$ is block diagonal / triangular, then the Cholesky factorisation of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}{}^\mathsf{T}$ can be computed at a cost $O(GD^3)$ for block diagonal, and $O(G^2D^3)$ for block triangular $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_2}$, with then again $O(LN_\mathbf{x}^2)$ cost low-rank updates of this Cholesky factor by the columns of $\frac{\partial \mathbf{g}_\mathbf{x}}{\partial \boldsymbol{u}_1}$ performed. When $\mathbf{x}_i$ and $\mathbf{u}_{2,i}$ are vectors of differing dimensions, with generally in this case $\dim(\mathbf{u}_{2,i}) > \dim(\mathbf{x}_i)$ due to the requirement the total number of random inputs M is at least $N_\mathbf{x}$, then though we could choose a subset of each $\mathbf{u}_{2,i}$ of equal dimension to $\mathbf{x}_i$ so as to identify a block-triangular component, generally any gain from exploiting this structure will be minimal and in practice it seems likely to be more efficient to compute the Cholesky of $\mathbf{J}_{\mathbf{g}_\mathbf{x}}\mathbf{J}_{\mathbf{g}_\mathbf{x}}{}^\mathsf{T}$ directly.

The Metropolis accept step and momentum updates in the SimDyn routine require evaluating the logarithm of the target density (4.38) and its gradient respectively. Although this can by achieved by directly using the expression given in (4.38) (and applying reverse-mode AD to get the gradient), both the log-density and gradient can be more efficiently calculated by reusing the Cholesky decomposition of the constraint Jacobian Gram matrix computed in line 42. Details are given in Appendix ??.

A final implementation detail is the requirement to find an initial $\boldsymbol{u}$ satisfying $\mathbf{g}_\mathbf{x}(\boldsymbol{u}) = \boldsymbol{x}$. In directed generative models with one of the structures just described, one method we found worked well in the experiments was to sample a $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ pair from P and then keeping the $\boldsymbol{u}_1$ values fixed, solve $\mathbf{g}_\mathbf{x}(\mathbf{g}_\mathbf{z}(\boldsymbol{u}_1), \boldsymbol{u}_2) = \boldsymbol{x}$ for $\boldsymbol{u}_2$ using for example Newton's

method or by directly minimising the Euclidean norm $\|\mathbf{g}_{\mathbf{x}}(\mathbf{g}_{\mathbf{z}}(\boldsymbol{u}_1), \boldsymbol{u}_2) - \boldsymbol{x}\|_2^2$ with respect to $\boldsymbol{u}_2$ by gradient descent. In more general cases one possible strategy is to randomly sample affine subspaces by generating a $M \times N_{\mathbf{x}}$ matrix $\boldsymbol{P}$ and M dimensional vector $\boldsymbol{b}$ and then attempting to find any intersections with the manifold by iteratively solving $\mathbf{g}_{\boldsymbol{x}}(\boldsymbol{P}\boldsymbol{v} + \boldsymbol{b})$ for $\boldsymbol{v}$ (and sampling a new subspace if no roots are found).

4.10 RELATED WORK

Closely related is the *Constrained HMC* method of [36], which demonstrates the validity of the constrained HMC framework theoretically and experimentally. The proposed method in [36] uses a RATTLE-based integrator rather than the geodesic scheme used here. The focus in [36] is also on performing inference in distributions inherently defined on a fixed non-Euclidean manifold such as the unit sphere or space of orthogonal matrices, rather than performing inference in differentiable generative models.

Geodesic Monte Carlo [38] also considers applying a HMC scheme to sample from non-linear manifolds embedded in a Euclidean space. Similarly to [36] however the motivation is performing inference with respect to distributions explicitly defined on a manifold such as directional statistics. The method presented in [38] uses an exact solution for the geodesic flow on the manifold. Our use of constrained Hamiltonian dynamics, and in particular the geodesic integration scheme of [123], can be considered an extension for cases when an exact geodesic solution is not available. Instead the geodesic flow is approximately simulated while still maintaining the required volume-preservation and reversibility properties for validity of the overall HMC scheme.

An alternative Metropolis method for sampling from densities defined on manifolds embedded in a Euclidean space is proposed in [238]. Compared to constrained HMC this alleviates the requirements to calculate the gradient of (the logarithm of) the target density on the manifold, though still requiring evaluation of the constraint function Jacobian. As discussed earlier in Section **??** (and in Appendix **??**), using reverse-mode AD the gradient of the target density can be computer at a constant factor overhead of evaluating the target density itself. In general we would expect exploiting the gradient of the target density on the manifold within a simulated Hamiltonian dynamic to lead to more

coherent exploration of the target distribution, instead of the more random-walk behaviour of a non-gradient based Metropolis update, and so for the gradient evaluation overhead to be worthwhile.

There is extensive theoretical discussion of the issues involved in sampling from distributions defined on manifolds in [55], including a derivation of the conditional density on a manifold using the co-area formula which directly motivated our earlier derivation of the target density for our constrained HMC method. The experiments in [55] are mainly concentrated on expository examples using simple parameterised manifolds such as a torus embedded in $\mathbb{R}^3$ and conditional testing in exponential family distributions.

Hamiltonian ABC [139], also proposes applying HMC to perform inference in simulator models as considered here. An ABC set-up is used with a Gaussian synthetic-likelihood formed by estimating moments from simulated data. Rather than using automatic differentiation to exactly calculate gradients of the generator function, *Hamiltonian ABC* uses a stochastic gradient estimator. This is based on previous work considering methods for using a stochastic gradients within HMC [43, 232]. It has been suggested however that the use of stochastic gradients can destroy the favourable properties of Hamiltonian dynamics which enable coherent exploration of high dimensional state spaces [22]. In *Hamiltonian ABC* it is also observed that representing the generative model as a deterministic function by fixing the random inputs to the generator is a useful method for improving exploration of the state space. This is achieved by including the seed of the pseudo-random number generator in the chain state rather than the set of random inputs.

Also related is *Optimisation Monte Carlo* [140]. The authors propose using an optimiser to find parameters of a simulator model consistent with observed data (to within some tolerance ϵ) given fixed random inputs sampled independently. The optimisation is not volume-preserving and so the Jacobian of the map is approximated with finite differences to weight the samples. Our method also uses an optimiser to find inputs consistent with the observations, however by using a volume-preserving dynamic we avoid having to re-weight samples which can scale poorly with dimensionality.

Our method also differs in treating all inputs to a generator equivalently; while the *Optimisation Monte Carlo* authors similarly identify the simulator models as deterministic functions they distinguish between parameters and random inputs, optimising the first and independently sampling the latter. This can lead to random inputs being sampled for which no parameters can be found consistent with the observations (even with a within ϵ constraint). Although optimisation failure is also potentially an issue for our method, we found this occurred rarely in practice if an appropriate step size is chosen. Our method can also be applied in cases were the number of unobserved variables is greater than the number of observed variables unlike *Optimization Monte Carlo*.

4.11 EXPERIMENTS

To illustrate the applicability of the proposed method we performed inference tasks in three diverse settings: parameter inference in a stochastic Lotka-Volterra predator-prey model simulation, 3D human pose and camera parameter inference given 2D joint position information and finally in-painting of missing regions of digit images using a generative model trained on MNIST. In all three experiments Theano [219] was used to specify the generator function and calculate the required derivatives. All experiments were run on an Intel Core i5-2400 quad-core CPU.

4.11.1 Lotka–Volterra parameter inference

As a first demonstration we considered a stochastic continuous state variant of the Lotka–Volterra model, a common example problem for ABC methods e.g. [140]. In particular we consider parameter inference given a simulated solution of the following stochastic differential equations

$$\mathrm{d}r = (z_1 r - z_2 r f)\mathrm{d}t + \mathrm{d}n_r, \quad \mathrm{d}f = (z_4 r f - z_3 f)\mathrm{d}t + \mathrm{d}n_f, \tag{4.51}$$

where r represents the prey population, f the predator population, $\{z_i\}_{i=1}^4$ the system parameters and n_r and n_f zero-mean white noise processes.

The observed data was generated with an Euler-Maruyama discretisation, time-step $\delta t = 1$, white noise process standard deviation $\sigma = 1$,

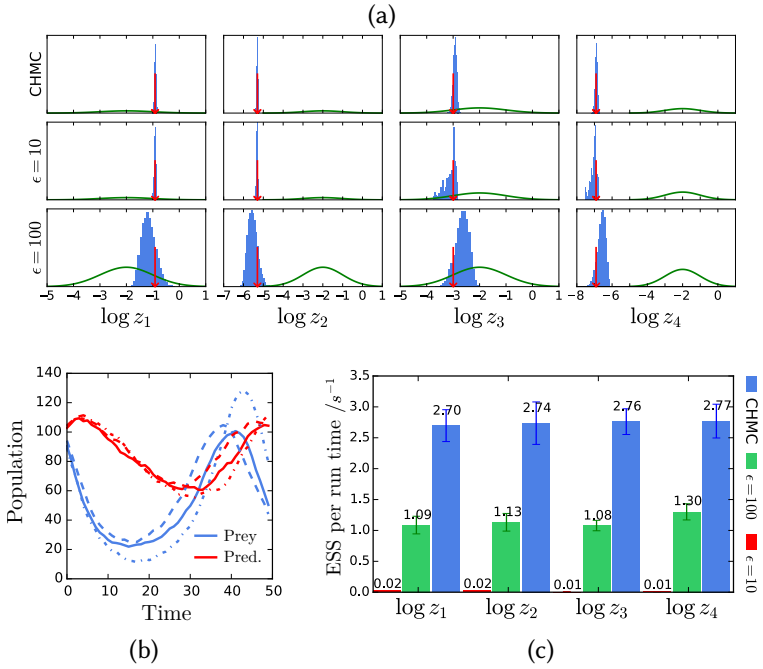


Figure 4.6.: Lotka–Volterra (a) Marginal empirical histograms for the (logarithm of the) four parameters (columns) from constrained HMC samples (top) and ABC samples with $\epsilon = 10$ (middle) and $\epsilon = 100$ (bottom). Horizontal axes shared across columns. Red arrows indicate true parameter values. Green curves show the log-normal prior densities for comparison. (b) Observed predator-prey populations (solid) and ABC sample trajectories with $\epsilon = 10$ (dashed) and $\epsilon = 100$ (dot-dashed). (c) Mean ESS normalised by compute time for each of four parameters for ABC with $\epsilon = 10$ (red), $\epsilon = 100$ (green) and our method (blue). Error bars show ±3 standard errors of mean.

initial condition $r_0 = f_0 = 100$ and $z_1 = 0.4$, $z_2 = 0.005$, $z_3 = 0.05$, $z_4 = 0.001$ (chosen to give stable dynamics). The simulation was run for $N_s = 50$ time-steps with the observed outputs defined as the concatenated vector $\mathbf{x} = [r_1\ f_1\ \ldots\ r_{50}\ f_{50}]$. Log-normal priors $z_i \sim \log \mathcal{N}(-2, 1)$ were place on the system parameters. Pseudo-code for the corresponding generator functions $\mathbf{g_z}$ and $\mathbf{g_{x|z}}$ is given in Algorithm 11. The generator in this case has the Markovian structure discussed in Section 4.9 allowing efficient computation of the Cholesky factor of the Jacobian matrix product $\mathbf{J_{g_x}}\mathbf{J_{g_x}}^\mathsf{T}$.

We compared our method to various ABC approaches using a uniform ball kernel with radius ϵ. ABC rejection failed catastrophically, with no acceptances in 10^6 samples even with a large $\epsilon = 1000$. ABC MCMC with a Gaussian proposal density q also performed very poorly with the dynamic having zero acceptances over multiple runs of 10^5 updates for

Algorithm 11 Lotka–Volterra model generator functions

Input:
δt : Euler–Maryuma integrator time step;
N_s : number of integrator steps to perform;
f, r : initial predator and prey populations;
σ : white noise process standard deviation;
$\boldsymbol{m}$, $\boldsymbol{s}$: location and scale parameters of log-normal prior.

function $\mathbf{g}_{\mathbf{z}}(\mathbf{u}_1)$
  $\mathbf{z} \leftarrow \exp(\boldsymbol{s} \odot \mathbf{u}_1 + \boldsymbol{m})$
  return z
function $\mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2)$
  $\mathrm{r}_0 \leftarrow r$
  $\mathrm{f}_0 \leftarrow f$
  for $s \in \{1 \dots N_s\}$ **do**
    $\mathrm{r}_s \leftarrow \mathrm{r}_{s-1} + \delta t(\mathrm{z}_1 \mathrm{r}_{s-1} - \mathrm{z}_2 \mathrm{r}_{s-1} \mathrm{f}_{s-1}) + \sqrt{\delta t}\sigma \mathrm{u}_{2,2s}$
    $\mathrm{f}_s \leftarrow \mathrm{f}_{s-1} + \delta t(\mathrm{z}_4 \mathrm{r}_{s-1} \mathrm{f}_{s-1} - \mathrm{z}_3 \mathrm{f}_{s-1}) + \sqrt{\delta t}\sigma \mathrm{u}_{2,2s+1}$
  $\mathbf{x} \leftarrow [\mathrm{r}_1, \mathrm{f}_1, \dots \mathrm{r}_{N_s}, \mathrm{f}_{N_s}]$
  return x

$\epsilon = 100$ and getting stuck at points in parameter space over many updates for larger $\epsilon = 1000$, even with small proposal steps. The same issues were also observed when using a Gaussian kernel. As we are conditioning on all of the observed data without use of summary statistics this poor performance is not unexpected.

We found that however by constructing a chain in the generator inputs space with target distribution (4.27), we can afford to condition on all of the observed outputs even when using relatively simple non-gradient based MCMC methods. In particular based on the pseudo-marginal slice sampling method [154] discussed earlier, we tried using alternating elliptical slice sampling updates of the random inputs $\mathbf{u}_1$ used to generate the parameters, i.e. $\mathbf{z} = \mathbf{g}_{\mathbf{z}}(\mathbf{u}_1)$, and remaining random inputs $\mathbf{u}_2$ used to generate the simulated observations given parameters, i.e. $\mathbf{x} = \mathbf{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z}, \mathbf{u}_2)$. The slice sampling updates locally adapt the size of steps made to ensure a move can always be made. Using this method we were able to obtain reasonable convergence over long runs for both $\epsilon = 100$ and $\epsilon = 10$. We therefore used this as our base-line method to compare the gains (in terms of how well the parameters are identified in the posterior) from using an $\epsilon \approx 0$ in the proposed constrained HMC method compared to non-zero ϵ values.

The results are summarised in Figure 4.6. Figure 4.6b shows the simulated data used as observations (solid curves) and ABC sample tra-

jectories for $\epsilon = 10$ (dashed) and $\epsilon = 100$ (dot-dashed). Though the ABC sampled trajectories follow the general trends of the observed data there are large discrepancies particularly for $\epsilon = 100$. Our method in contrast samples parameters generating trajectories in which the discrepancy between the simulated and observed trajectories is effectively zero (with the convergence tolerance used corresponding to a maximum elementwise difference of 10^{-8}). Figure 4.6a shows the marginal histograms for the parameters. The inferred posterior on the parameters are significantly more tightly distributed about the true values used to generate the observations for our approach and the $\epsilon = 10$ case compared to the results for $\epsilon = 100$. In all cases, as should be expected, the empirical posterior marginals are significantly more tightly distributed than the priors (green curves).

Figure 4.6c shows the relative sampling efficiency of our approach against the slice-sampling based ABC methods, as measured by the ESS (computed with R-CODA [177]) normalised by chain run time averaged across 10 sampling runs for each method. Despite the significantly higher per-update cost in our method, the longer range moves made by the Hamiltonian dynamic gave significantly better performance even over the very approximate $\epsilon = 100$ case. Compared to the $\epsilon = 10$ case the speed-up is around a factor of 100, with the slice sampling updates although performing much better than a standard Metropolis–Hastings based ABC MCMC approach, only able to make small moves in the input space on each iteration due to the (relatively) tight ϵ constraint. This both produces very slowly decaying auto-correlations between successive states and thus a much reduced effective sample size, but also leads to a slow initial 'warm-up' convergence to the posterior typical set. The spurious appearing peaks in the distributions for z_3 and z_4 for the $\epsilon = 10$ case in Figure 4.6a seem likely to be an artefact of some of the chains used having not fully converged even after the 10000 transitions used in each chain.

Although using effective sample sizes as a measure of performance can give misleading results, particularly in the face of convergence issues such as those just discussed, the large difference in the computed values and consistent improvement over multiple independent chains, gives some credence to there being a real gain in performance from the proposed constrained HMC approach. Further by eliminating the need to choose an appropriate ϵ tolerance and allowing use of all the observed

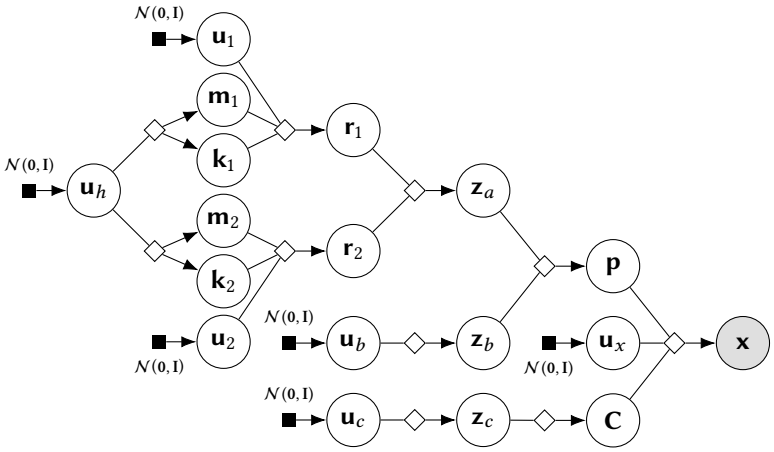


Figure 4.7.: Factor graph of human pose differentiable generative model. The operations corresponding to the deterministic nodes ($\diamond$) in the graph are described in Algorithm 12.

data rather than needing to find appropriate summary statistics, the method also removes some of the complexities of standard ABC MCMC approaches. This does however come at the cost of requiring the HMC algorithm free parameters to be tuned, though methods such as the *No U-Turns Sampler* [102] have been proposed for automating these choices.

4.11.2 Human pose and camera model inference

For our next experiment we considered inferring a three-dimensional human pose and camera model from two-dimensional projections of joint positions. We used a 19 joint skeleton model, with a learnt prior distribution over poses parametrised by 47 local joint angles $\mathbf{z}_a$. The pose model was learnt from the *PosePrior* motion capture data-set [1] with a Gaussian VAE [112] trained to match the distribution of the motion capture joint angle data. The circular topology of the angular data is poorly matched by the Euclidean space a Gaussian VAE typically learns a distribution on, and simply 'unwrapping' the angles to e.g. $[-\pi, \pi)$ leads to unnatural discontinuities at the $\pm\pi$ cut-point, this both making the initial learning problem challenging (as there is no in-built prior knowledge of continuity across the cut-point) and tending to lead to a learned latent space less amenable to MCMC inference as 'nearby' poses with one or more joint angles on opposite sides of the cut-point will likely end up corresponding to points far apart in the latent space.

Algorithm 12 Human pose model generator functions

Input:

$\{\boldsymbol{W}_\ell, \boldsymbol{b}_\ell\}_{\ell=0}^{L}$: parameters of pose angle differentiable network;
$\boldsymbol{\mu}_b$, $\boldsymbol{\Sigma}$: mean and covariance of skeleton bone lengths;
$\boldsymbol{\mu}_{c,:2}$, $\boldsymbol{\sigma}_{c,:2}$: camera x, y coordinates normal prior parameters;
$\mu_{c,2}$, $\sigma_{c,2}$: camera z coordinate log-normal prior parameters;
ϵ : image joint position observation noise standard deviation;
JointPositions : maps pose angles and bone lengths to joint positions;
CameraMatrix : maps camera parameters to projective camera matrix;
Project : uses camera matrix to map world to image coordinates;
Partition : partitions a vector in a specified number of equal length parts;
Flatten : flattens a multidimensional array to a vector.

function $\mathrm{g}_{\mathbf{z}}([\mathbf{u}_h; \mathbf{u}_1; \mathbf{u}_2; \mathbf{u}_b; \mathbf{u}_c])$
  $\mathbf{h}_L \leftarrow \textsc{DifferentiableNetwork}(\mathbf{u}_h)$
  $\mathbf{m}_1, \mathbf{k}_1, \mathbf{m}_2, \mathbf{k}_2 \leftarrow \textsc{Partition}(\mathbf{h}_L, 4)$
  $\mathbf{r}_1 \leftarrow \exp(\mathbf{k}_1) \odot \mathbf{u}_1 + \mathbf{m}_1$
  $\mathbf{r}_2 \leftarrow \exp(\mathbf{k}_2) \odot \mathbf{u}_2 + \mathbf{m}_2$
  $\mathbf{z}_a \leftarrow \mathrm{atan2}(\mathbf{r}_2, \mathbf{r}_1)$
  $\mathbf{z}_b \leftarrow \exp(\boldsymbol{\mu}_b + \boldsymbol{\Sigma}_b \mathbf{u}_b)$
  $\mathbf{z}_{c,:2} \leftarrow \boldsymbol{\sigma}_{c,:2} \odot \mathbf{u}_{c,:2} + \boldsymbol{\mu}_{c,:2}$
  $\mathrm{z}_{c,2} \leftarrow \exp(\sigma_{c,2} \mathrm{u}_{c,2} + \mu_{c,2})$
  return $[\mathbf{z}_a; \mathbf{z}_b; \mathbf{z}_c]$
function DifferentiableNetwork($\mathbf{u}_h$)
  $\mathbf{h}_0 \leftarrow \tanh(\boldsymbol{W}_0 \mathbf{u}_h + \boldsymbol{b}_0)$
  for $\ell \in \{1 \ldots L-1\}$ **do**
    $\mathbf{h}_\ell \leftarrow \tanh(\boldsymbol{W}_\ell \mathbf{h}_{\ell-1} + \boldsymbol{b}_\ell) + \mathbf{h}_{\ell-1}$
  return $\boldsymbol{W}_L \mathbf{h}_{L-1} + \boldsymbol{b}_L$
function $\mathrm{g}_{\mathbf{x}|\mathbf{z}}([\mathbf{z}_a; \mathbf{z}_b; \mathbf{z}_c], \mathbf{u}_x)$
  $\mathbf{P} \leftarrow \textsc{JointPositions}(\mathbf{z}_a, \mathbf{z}_b)$
  $\mathbf{C} \leftarrow \textsc{CameraMatrix}(\mathbf{z}_c)$
  $\mathbf{X} \leftarrow \textsc{Project}(\mathbf{C}, \mathbf{P})$
  return $\textsc{Flatten}(\mathbf{X}) + \epsilon \mathbf{u}_x$

During training we therefore mapped each vector of 47 joint angles $\boldsymbol{z}_a^{(i)}$ (corresponding to a single motion capture datapoint) to a pair of 47-dimensional vectors $(\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)})$ by sampling a Gaussian random vector $\boldsymbol{n}^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then computing $\boldsymbol{r}_1^{(i)} = \exp \boldsymbol{n}^{(i)} \odot \cos \boldsymbol{z}_a^{(i)}$ and $\boldsymbol{r}_2^{(i)} = \exp \boldsymbol{n}^{(i)} \odot \sin \boldsymbol{z}_a^{(i)}$ and training the VAE to maximise (a variational lower bound) on the joint marginal density of the $\{\boldsymbol{r}_1^{(i)}, \boldsymbol{r}_2^{(i)}\}_i$ pairs. At the cost of doubling the dimension, this leads to an embedding in a Euclidean space which does not introduce any arbitary cut-points and empirically seemed to lead to better sample quality from the learned generative model compared to learning the angles directly. Given the trained model we can generate a vector of angles $\mathbf{z}_a$ using the model by sampling a Gaussian code (latent representation) vector $\mathbf{u}_h$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ then sampling a pair of 47-dimensional vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ from

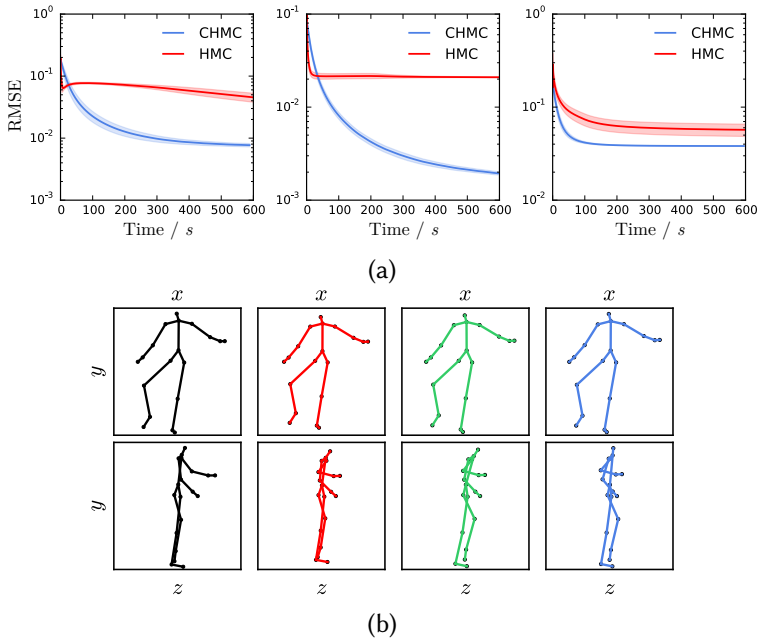


Figure 4.8.: Human pose (a) RMSEs of 3D pose posterior mean estimates given binocular projections, using samples from our method (blue) versus running HMC in hierarchical model (red) for three different scenes sampled from the prior. Horizontal axes show computation time to produce number of samples in estimate. Solid curves are average RMSE over 10 runs with different seeds and shaded regions show ±3 standard errors of mean. (b) Orthographic projections (top: front view, bottom: side view) of 3D poses consistent with monocular projections. Left most pair (black) shows pose used to generate observations, right three show constrained HMC samples.

the learnt Gaussian decoder model given $\mathbf{u}_h$ (and further Gaussian random input vectors $\mathbf{u}_1$ and $\mathbf{u}_2$), and finally recovering an angle by computing $\mathbf{z}_a = \text{atan2}(\mathbf{r}_2, \mathbf{r}_1)$. The resulting distribution on $\mathbf{z}_a$ is only implicitly defined, but the overall generative model is differentiable with respect to the input vectors $\mathbf{u}_h$, $\mathbf{u}_1$ and $\mathbf{u}_2$.

The *PosePrior* motion capture data includes recordings from only a relatively small number of distinct actors and so limited variation in the 'bone-lengths' of the skeleton model. Therefore a serparate log-normal model for the bone lengths $\mathbf{z}_b$ was fitted using data from the *ANSUR* anthropometric data-set [86], due to symmetry in the skeleton thirteen independent lengths being specified. A simple pin-hole projective camera model with three position parameters $\mathbf{z}_c$ and fixed focal-length

was used[7]. A log-normal prior distribution was placed on the depth co-ordinate $z_{c,2}$ to enforce positivity with normal priors on the other two co-ordinates $z_{c,0}$ and $z_{c,1}$.

Given a generated triplet of joint-angles, bone length and camera parameters $\mathbf{z}_a$, $\mathbf{z}_b$ and $\mathbf{z}_c$, a simulated two-dimensional projection of the skeleton $\mathbf{x}$ is produced by first mapping the joint-angles and bone-lengths to a 4×19 matrix of joint positions $\mathbf{P}$ in (homegeneous) world-coordinates by recursing through the skeleton tree. A 3×4 projective camera matrix $\mathbf{C}$ is generated from $\mathbf{z}_c$ and then used to project the world-coordinate joint positions to a 2×19 matrix $\mathbf{X}$ of joint positions in two-dimensional image-coordinates. The projected positions matrix $\mathbf{X}$ is flattened to a vector and a Gaussian vector with standard deviation ϵ added to the projected position vector to give the $19 \times 2 = 38$ dimensional observed vector $\mathbf{x}$. The noise standard deviation ϵ is chosen so that the noise in the projected joint positions is non-obvious in generated projections. The overall corresponding model generator functions $\mathbf{g}_{\mathbf{x}|\mathbf{z}}$ and $\mathbf{g}_{\mathbf{z}}$ are described procedurally in Algorithm **??** and a factor graph summarising the relationships between the variables in the model shown in Figure 4.7.

Although the Gaussian observed output noise is necessarily not needed to apply our proposed constrained HMC method as the generator without the final additive noise still defines a valid differentiable generative model, using the noisy observation model means that an explicit hierarchical joint density on is defined on $\{\mathbf{x}, \mathbf{u}_h, \mathbf{r}_1, \mathbf{r}_2, \mathbf{z}_b, \mathbf{z}_c\}$ allowing comparison of our constrained HMC method with (non-constrained) HMC as a baseline. Further as discussed previously the adding noise to the output ensures the generator Jacobian is full-rank everywhere and also significantly simplifies the process of finding an initial $\mathbf{u}$ such that the generated $\mathbf{x}$ matches observations.

We first considered binocular pose estimation, with the variables defining the three-dimensional scene information $\mathbf{z}_a$, $\mathbf{z}_b$ and $\mathbf{z}_c$, inferred given a pair of two-dimensional projections from two simulated cameras with a known offset in their positions (in this case the generator function is adjusted accordingly to output an $19 \times 2 \times 2 = 76$ dimensional observed vector $\mathbf{x}$ corresponding to the concatenation of the

7 The camera orientation was assumed fixed to avoid replicating the degrees of freedom specified by the angular orientation of the root joint of the skeleton: only the relative camera–skeleton orientation is important.

flattened projected joint positions from both 'cameras'). In this binocular case, the disparity in projected joint positions between the two projections gives information about the distances of the correspondings joints from the image plane in the depth direction and so we would expect the posterior distribution on the three-dimensional pose to be tightly distributed around the true values used to generate the observations. We compared our constrained HMC method to running standard HMC on the conditional density of $\{\mathbf{u}_h, \mathbf{r}_1, \mathbf{r}_2, \mathbf{z}_b, \mathbf{z}_c\}$ given $\mathbf{x}$.

Figure 4.8a shows the RMSE between the posterior mean estimate of the three-dimensional joint positions and the true positions used to generate the observations as the number of samples included in the estimate increases for three test scenes. For both methods the horizontal axis has been scaled by run time. The constrained HMC method (blue curves) tends to give position estimates which converge more quickly to the true position. In this case standard HMC performs relatively poorly despite the signficantly cheaper cost of each integrator step compared to the constrained dynamics. This is at leastin part due to the small output noise standard deviation ϵ used which requires a small integrator step to be used to maintain reasonable accept rates. Relaxing too larger ϵ values makes the non-constrained approach more competitive but with an associated cost in loss of

Visually inspecting the sampled poses and individual run traces (not shown) it seems that the HMC runs tended to often get stuck in local modes corresponding to a subset of joints being 'incorrectly' positioned while still broadly matching the (noisy) projections. The complex dependencies of the joint positions on the angle parameters mean the dynamic struggles to find an update which brings the 'incorrect' joints closer to their true positions without moving other joints out of line. The constrained HMC method seemed to be less susceptible to this issue.

We also considered inferring 3D scene information from a single 2D projection. Monocular projection is inherently information destroying with significant uncertainty to the true pose and camera parameters which generated the observations. Figure 4.8b shows pairs of orthographic projections of 3D poses: the left most column is the pose used to generate the projection conditioned on and the right three columns are poses sampled using constrained HMC consistent with the observations. The top row shows front x–y views, corresponding to the camera

(a) Constrained HMC samples (b) HMC samples

Figure 4.9.: MNIST In-painting samples. The top black-on-white quarter of each image is the fixed observed region and the remaining white-on-black region the proposed in-painting. In left-right, top-bottom scan order the images in (a) are consecutive samples from a constrained HMC run; in (b) the images are every 40^{th} sample from a HMC run to account for the $\sim 40\times$ shorter run-time per sample. All images in this run are show in Figure **??** in the Appendix.

view though with a orthographic rather than perspective projection, the bottom row shows side z–y views with the z axis the depth from the camera. The dynamic is able to move between a range of plausible poses consistent with the observations while reflecting the inherent depth ambiguity from the monocular projection.

4.11.3 MNIST in-painting

As a final task we considered inferring in-paintings for a missing region of a digit image $\mathbf{z}$ given knowledge of the rest of the pixel values $\mathbf{x}$. A Gaussian VAE trained on MNIST was used as the generative model, with a 50-dimensional hidden code $\mathbf{h}$. We compared our method to running HMC in the known conditional distribution on $\mathbf{h}$ given $\mathbf{x}$ ($\mathbf{z}$ can then be directly sampled from its Gaussian conditional distribution given $\mathbf{h}$).

Example samples are shown in Figure 4.9. In this case the constrained and standard HMC approaches appear to be performing similarly, with both able to find a range of plausible in-paintings given the observed pixels. Without cost adjustment the standard HMC samples show greater correlation between subsequent updates, however for a fairer comparison the samples shown were thinned to account for the approximately $40\times$ larger run-time per constrained HMC sample. Although the constrained dynamic does not improve efficiency here neither does it seem to hurt it.

4.12 DISCUSSION

We have presented a generally applicable framework for performing inference in differentiable generative models. Though simulating the constrained Hamiltonian dynamic is computationally costly, the resulting coherent exploration of the state space can lead to significantly improved sampling efficiency over alternative methods.

Further our approach allows asymptotically exact inference in differentiable generative models where ABC methods might otherwise be used. We suggest an approach for dealing with two of the key issues in ABC methods — enabling inference in continuous spaces as ϵ collapses to zero and allowing efficient inference when conditioning on high-dimensional observations without the need for dimensionality reduction with summary statistics (and the resulting task of choosing appropriate summary statistics). As well as being of practical importance itself, this approach should be useful in providing 'ground truth' inferences in more complex models to assess the affect of the approximations used in ABC methods on the quality of the inferences.

In molecular simulations, constrained dynamics are often used to improve efficiency. Intra-molecular motion is removed by fixing bond lengths. This allows a larger time-step to be used due to the removal of high-frequency bond oscillations [123]. An analogous effect is present when performing inference in an ABC setting with a ϵ kernel 'soft-constraint' to enforce consistency between the inputs and observed outputs. As $\epsilon \to 0$ the scales over which the inputs density changes value in directions orthogonal to the constraint manifold and along directions tangential to the manifold increasingly differ. To stay within the soft constraint a very small step-size needs to be used. Using a constrained dynamic decouples the motion on the constraint manifold from the steps to project on to it, allowing more efficient larger steps to be used for moving on the manifold.

A limitation of our method is the requirement of differentiability of the generator. This prevents applying our approach to generative models which use discontinuous operations or discrete random inputs. In some cases conditioned on fixed values of discrete random inputs the generator may still be differentiable and so the proposed method can be used to update the continuous random inputs given values of the

discrete inputs. This would need to be alternated with updates to the discrete inputs, which would require devising methods for updating the discrete inputs to the generator while constraining its output to exactly match observations.

As discussed in Section 4.5, a common approach in ABC methods is to define the kernel or distance measure in terms of summary statistics of the observed data [132, 179]. This is necessary in standard ABC approaches to cope with the 'curse of dimensionality' with the probability of accepting samples / moves for a fixed tolerance ϵ exponentially decreasing as the dimensionality of the observations conditioned on increases. Although as already noted the proposed method is better able to cope with high observation dimensions, if appropriate informative statistics are available (e.g. based on expert knowledge) and these are differentiable functions of the generator outputs, they can be easily integrated in to the proposed method by absorbing the statistic computation in to the definition of $\mathbf{g}_{\mathbf{x}}$.

A DISTRIBUTION DEFINITIONS

Name	Parameters	Shorthand	Density	Support
Bernoulli	$\pi \in [0,1]$	$\mathrm{Ber}(x \mid \pi)$	$\pi^x(1-\pi)^{(1-x)}$	$x \in \{0,1\}$
Categorical	$\boldsymbol{\pi} \in \mathbb{S}^K$	$\mathrm{Cat}(x \mid \boldsymbol{\pi})$	$\sum_{k=1}^{K}(\mathbb{1}_{\{k\}}(x)\pi_k)$	$x \in \{1 \dots K\}$

Table A.1.: Definitions of densities of parameteric distributions for discrete random variables used in this thesis.

Name	Parameters	Shorthand	Density
Normal	$\mu \in \mathbb{R}$: mean $\sigma > 0$: standard deviation	$\mathcal{N}\left(x \mid \mu, \sigma^2\right)$	$\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
Multivariate normal	$\boldsymbol{\mu} \in \mathbb{R}^D$: mean vector $\boldsymbol{\Sigma} \in \mathcal{S}_{++}^D$: covariance matrix	$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\frac{1}{\sqrt{(2\pi)^D\lvert\boldsymbol{\Sigma}\rvert}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$
Student's t	$\nu > 0$: degrees of freedom $\mu \in \mathbb{R}$: location $\sigma > 0$: scale	$\mathrm{StT}(x \mid \nu, \mu, \sigma)$	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu}\sigma}\left(1+\frac{1}{\nu}\left(\frac{x-\mu}{\sigma}\right)^2\right)^{-\frac{\nu+1}{2}}$
Logistic	$\mu \in \mathbb{R}$: location $\sigma > 0$: scale	$\mathrm{Logistic}(x \mid \mu, \sigma)$	$\frac{1}{4\sigma}\cosh\left(\frac{x-\mu}{2\sigma}\right)^{-2}$
Inverse cosh	$\mu \in \mathbb{R}$: location $\sigma > 0$: scale	$\mathrm{InvCosh}(x \mid \mu, \sigma)$	$\frac{1}{2\sigma}\cosh\left(\frac{\pi(x-\mu)}{2\sigma}\right)^{-1}$

Table A.2.: Definitions of densities of parameteric distributions for unbounded real random variables used in this thesis.

Name	Parameters	Shorthand	Density	Support
Log-normal	$\mu \in \mathbb{R}$: log mean $\sigma > 0$: log standard deviation	$\text{LogNorm}(x \mid \mu, \sigma^2)$	$\frac{1}{x\sqrt{2\pi}\sigma} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right)$	$x > 0$
Multivariate log-normal	$\boldsymbol{\mu} \in \mathbb{R}^D$: log mean $\boldsymbol{\Sigma} \in \mathcal{S}^D_{++}$: log covariance	$\text{LogNorm}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\frac{\exp\left(-\frac{1}{2}(\log \boldsymbol{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1} (\log \boldsymbol{x} - \boldsymbol{\mu})\right)}{\prod_{d=1}^D (x_d)\sqrt{(2\pi)^D \lvert\boldsymbol{\Sigma}\rvert}}$	$\boldsymbol{x} \in [0, \infty)^D$
Exponential	$\lambda > 0$: rate	$\text{Exp}(x \mid \lambda)$	$\lambda \exp(-\lambda x)$	$x \geq 0$
Uniform	$a \in \mathbb{R}$: minimum $b \in \mathbb{R}$: maximum, $b > a$	$\mathcal{U}(x \mid a, b)$	$\frac{1}{b-a} \mathbb{1}_{[a,b]}(x)$	$a \leq x \leq b$
Half-Cauchy	$\gamma > 0$: scale	$C_{\geq 0}(x \mid \gamma)$	$\frac{2}{\pi\gamma}\left(1 + \frac{x^2}{\gamma^2}\right)^{-1}$	$x \geq 0$
Gamma	$\alpha > 0$: shape $\beta > 0$: rate	$\text{Gamma}(x \mid \alpha, \beta)$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x)$	$x \geq 0$
Beta	$\alpha > 0$: shape $\beta > 0$: shape	$\text{Beta}(x \mid \alpha, \beta)$	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$	$0 \leq x \leq 1$
Dirichlet	$\boldsymbol{\alpha} \in (0, \infty)^K$: concentration	$\text{Dir}(\boldsymbol{x} \mid \boldsymbol{\alpha})$	$\frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K x_i^{\alpha_k - 1}$	$\boldsymbol{x} \in \mathbb{S}^K$
Lomax	$\alpha > 0$: shape $\beta > 0$: scale	$\text{Lomax}(x \mid \alpha, \beta)$	$\frac{\alpha\beta^\alpha}{(\beta+x)^{\alpha+1}}$	$x \geq 0$

Table A.3.: Definitions of densities of parameteric distributions for bounded real random variables used in this thesis.

B GRAPHICAL MODELS

Graphical models = statistics × graph theory × computer science
—Zoubin Ghahramani

When working with probabilistic models involving large numbers of random variables, it will often be the case that not all the variables are jointly dependent on each other but that instead there are more local conditional relationships between them. Graphical models, which use graphs to describe the dependencies between random variables, are a useful framework for visualising the structure in complex probabilistic models and for giving a graph-theoretic basis for establishing the dependence between sets of random variables. Central to graphical models is the concept of conditional independence. Let $\mathsf{x} : S \to X$, $\mathsf{y} : S \to Y$ and $\mathsf{z} : S \to Z$ be three random variables with corresponding σ-algebras, $\mathcal{F}_{\mathsf{x}}$, $\mathcal{F}_{\mathsf{y}}$ and $\mathcal{F}_{\mathsf{z}}$ respectively. Following from our earlier definition of (unconditional) independence of random variables in (1.4), we say that x *and* y *are conditionally independent given* z, denoted $\mathsf{x} \perp \mathsf{y} \mid \mathsf{z}$, if for P_{z}-almost all $z \in Z$

$$\mathsf{P}_{\mathsf{x},\mathsf{y}|\mathsf{z}}(A, B \mid z) = \mathsf{P}_{\mathsf{x}|\mathsf{z}}(A \mid z)\, \mathsf{P}_{\mathsf{y}|\mathsf{z}}(B \mid z) \;\; \forall A \in \mathcal{F}_{\mathsf{x}},\, B \in \mathcal{F}_{\mathsf{y}}. \tag{B.1}$$

If a joint density on the random variables exists, a sufficient condition for $\mathsf{x} \perp \mathsf{y} \mid \mathsf{z}$ is that the conditional density $\mathsf{p}_{\mathsf{x},\mathsf{y}|\mathsf{z}}$ factorises as

$$\mathsf{p}_{\mathsf{x},\mathsf{y}|\mathsf{z}}(x, y \mid z) = \mathsf{p}_{\mathsf{x}|\mathsf{z}}(x \mid z)\mathsf{p}_{\mathsf{y}|\mathsf{z}}(y \mid z) \quad \forall x \in X,\, y \in Y,\, z \in Z. \tag{B.2}$$

This definition can be naturally extended to conditional independence when conditioning on more than one random variable, for example

$$\mathsf{v} \perp \mathsf{x} \mid \mathsf{y},\, \mathsf{z} \implies \mathsf{p}_{\mathsf{v},\mathsf{x}|\mathsf{y},\mathsf{z}}(v, x \mid y, z) = \mathsf{p}_{\mathsf{v}|\mathsf{y},\mathsf{z}}(v \mid y, z)\mathsf{p}_{\mathsf{x}|\mathsf{y},\mathsf{z}}(x \mid y, z). \tag{B.3}$$

B.1 DIRECTED AND UNDIRECTED MODELS

Several different graphical frameworks have been proposed for representing conditional independency relationships (and other information) in probabilistic models.

(a) Directed graphical model. (b) Undirected graphical model.

Figure B.1.: Examples of directed and undirected graphical models. Circular nodes represent random variables in the model, with edges between them indicating dependencies between variables.

Directed graphical models [172], also known as *Bayesian networks*, represent probabilistic models as *directed acyclic graphs* (i.e. a directed graph in which there are no directed cycles), with the nodes in the graph representing random variables in the model and the edges of the graph defining a factorisation of the joint density over these variables into a product of conditional and marginal densities. In particular a conditional density factor is included for each node with parents (on the node random variable value given the parent variable values) and a marginal density factor for each root node without any parents.

An example directed graphical model for three random variables, x_1, x_2 and x_3, is shown in Figure B.1a. The graph implies that the joint density can be factorised as

$$\mathrm{p}_{\mathrm{x}_1,\mathrm{x}_2,\mathrm{x}_3}(x_1, x_2, x_3) = \mathrm{p}_{\mathrm{x}_3|\mathrm{x}_1,\mathrm{x}_2}(x_3 \mid x_1, x_2)\, \mathrm{p}_{\mathrm{x}_1}(x_1)\, \mathrm{p}_{\mathrm{x}_2}(x_2). \tag{B.4}$$

Ancestral sampling in a directed graphical model corresponds to first sampling values from all the root nodes from their marginal densities, then iteratively sampling from the conditional densities on each node for which all the parents nodes already have sampled values to condition on.

Note that this factorisation would not be valid for all joint densities on the three variables; in particular we have that x_1 and x_2 are (unconditionally) independent and so that the joint density $\mathrm{p}_{\mathrm{x}_1,\mathrm{x}_2}$ can be written as the product of the two marginals $\mathrm{p}_{\mathrm{x}_1}$ and $\mathrm{p}_{\mathrm{x}_2}$.

Directed graphical models are a natural way of specifying *generative models* - i.e. probabilistic models which can be used to generate simulated observable quantities. Typically the factorisation specified by a directed graphical model gives a straightforward method to generate values from the joint density via *ancestral sampling*.

An alternative formalism for graphically representing probabilistic models is that of *undirected graphical models* [111], which are also known as *Markov random fields*. As with directed graphical models, each node in the graph represents a random variable, but here the edges connecting nodes are undirected. Rather than describing a factorisation of a joint

density into conditional and marginal densities, an undirected graphical model indicates the factorisation of a joint density into a product of clique potentials on each of the maximal cliques in the graph. A *clique* is a fully connected component of the graph - i.e. a subset of nodes in the graph such that all pairs of nodes in the subset are connected by an edge. A *maximal clique* is a clique which is not a strict subset of any other clique. A *clique potential* is a non-negative function of the values of the random variables in the clique; it does necessarily correspond to any conditional or marginal probabilty density.

An example undirected graphical model on four random variables, x_1, x_2, x_3 and x_4, is shown in Figure B.1b. Here the (maximal) cliques correspond to all the connected pairs of nodes. If $\psi_{a,b}$ denotes the clique potential on the pair (a, b) then the graphical model implies the joint density can be factorised as

$$\frac{1}{Z}\psi_{\mathsf{x}_1,\mathsf{x}_2}(x_1, x_2)\psi_{\mathsf{x}_1,\mathsf{x}_3}(x_1, x_3)\psi_{\mathsf{x}_2,\mathsf{x}_4}(x_2, x_4)\psi_{\mathsf{x}_3,\mathsf{x}_4}(x_3, x_4), \tag{B.5}$$

with Z a normalising constant such that the density integrates to 1 and so defines a valid probability measure.

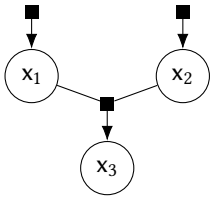
Undirected graphical models are a natural representation for models of systems of mutually interacting components. For example they are commonly used in models of images to represent dependencies between pixel values and models of magnetic interactions in particle lattices.

Unlike directed models, generating joint configurations of the random variables in an undirected graphical model from the implied joint distribution is typically a non-trivial task, with no general equivalent to ancestral sampling. Further the joint density can typically only be evaluated up to an unknown normalising constant, with the integral needed to evaluate this constant often intractable for models involving a large number of variables or complex potentials. As we will see, these properties mean that inference in distributions defined by undirected graphical models is often particularly challenging.

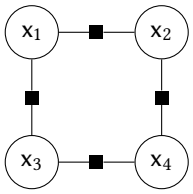
As suggested at the start of this section, both directed and undirected graphical models encode conditional independence properties of probabilistic models. In particular the rules of *D-separation* for directed graphical models and *U-separation* for undirected model give graph-based algorithmic descriptions of how to determine whether a pair of

D-separation: $\mathsf{x} \perp \mathsf{y} \mid C \iff$ *all paths in the graph between* x *and* y *are blocked. A path is blocked if at least one of the following holds:*
1. The path includes a $\rightarrow\bigcirc\rightarrow$ *node or a* $\leftarrow\bigcirc\rightarrow$ *node in* C.
2. The path includes a $\rightarrow\bigcirc\leftarrow$ *node and neither the node or its descendants are in* C.

U-separation: x *and* y *in the model and a conditioning set of random variables*



(a) A factor graph equivalent of the directed model in Figure B.1a.



(b) A factor graph equivalent of the undirected model in Figure B.1b.

Figure B.2.: Examples of factor graphs corresponding to the directed and undirected graphical models in Figure B.1. Square black nodes correspond to individual factors depending on the connected variables (represented by circular nodes) in the joint density.

random variables are conditionally independent for a given conditioning set of random variables.

For example the directed graphical model in Figure B.1a encodes the (un)conditional independence property $x_1 \perp x_2 \,|\, \emptyset = x_1 \perp x_2$ i.e. that x_1 and x_2 are independent if the value of x_3 is *not* conditioned on. The undirected graphical model in Figure B.1b encodes the conditional independence properties $x_1 \perp x_4 \,|\, x_2, x_3$ and $x_2 \perp x_3 \,|\, x_1, x_4$.

Although there are methods to convert a directed graphical model to an undirected one and vice versa, in general these transformations are lossy - not all of the conditional independence relationships encoded in the original graph will necessarily be maintained in the transformed graph. For example there is no undirected graphical model which will represent the exact set of conditional independence properties represented by the directed graphical model in Figure B.1a. Likewise there is no directed graphical model which will represent the exact set of conditional independence properties represented by the undirected graphical model in Figure B.1b. Further there are distributions with dependency structures and factorisations which cannot be uniquely represented by either directed or undirected graphical models [69].

B.2 FACTOR GRAPHS

An alternative graphical model formalism which overcomes some of the limitations of directed and undirected graphical models is that of factor graphs [69, 70]. In factor graphs, in addition to nodes representing random variables, represented as in directed and undirected graph-

ical models by circular nodes, a second class of nodes, denoted by filled squares (■), are introduced which represent individual factors in the joint density across the random variables represented in the model.

Factors may be either directed or undirected. Undirected factors, denoted by factor nodes in which all edges connecting to variable nodes are undirected, correspond to a factor in the joint density which depends on all of the variables with nodes connected to the factor, but without any requirement that the factor corresponds to a conditional or marginal probability density. Directed factors, denoted by factor nodes in which at least one edge from the factor node to a variable node is directed, correspond to a conditional density on the variables pointed to by directed edges given the values of the variables connected to the the factor node by undirected edges (if there are no such variables then the factor instead corresponds to a marginal density).

Edges between nodes in a factor graph are always between nodes of disparate types i.e. between factor and variable nodes, but never between factor and factor or variable and variable nodes. As with directed graphical models, factor graphs with directed factors must not contain any directed cycles (i.e. a connected loop of edges in which one of every pair of edges connected to a factor on the loop is directed and all of the directed edges point in the same sense around the loop).

In the original extension of undirected factor graphs [70] to include directivity [69], it was proposed to allow multiple directed factors to connect via directed edges to the same variable node, representing multiple factors in a conditional density on that varaiable. This generalisation introduces extra normalisation requirements and looses the interpretation of a directed factor as directly representing a conditional density, and so we will here only use directed factor graphs in which there is at most one directed edge connecting from a factor to a node.

Whether two variables are conditionally independent given a set of other variables can be checked from a factor graph by checking if all paths (i.e. connected series of edges and nodes) between the two corresponding variables nodes in the factor graph are *blocked*. A path is blocked if at least one of the following conditions is satisfied [69]

1. One of the variable nodes in the path is in the conditioning set.

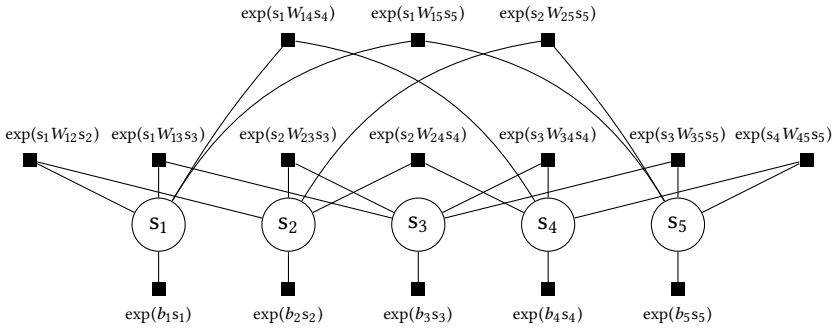


Figure B.3.: Five unit Boltzmann machine factor graph showing explicit factorisation of distribution into pairwise and single variable potentials.

2. One of the directed factor nodes in the path has two connected undirected edges in the path and there is no second directed path from the node to a variable node in the conditioning set.

Both directed and undirected graphical models can always be losslessly converted to a factor graph, i.e. such that by applying the above blocking rules after the transformation we obtain exactly the same set of conditional independency properties as present in the original graph, and thus they have a superset of the capacity to represent conditional indendence properties as either of these two alternative frameworks. For example, factor graph equivalents of the directed and undirected graphical model examples in Figure B.1 are shown in Figure B.2.

As well as allowing representations of mixed graphs with both directed and undirected factors which cannot be represented with either directed or undirected graphical models, factor graphs are also able to include finer-grained information about the factorisation of the joint density than either of the other two model types by explicitly indicating the presence of individual factors. For instance Figure B.3 shows the factor graph for a *Boltzmann machine* distribution, sometimes called a *pairwise binary Markov random field* or *Ising model*, on five binary random variables $\{\mathrm{s}_i\}_{i=1}^5$. A Boltzmann machine distribution can be factored in to a product of pairwise weighted interactions $\exp(\mathrm{s}_i W_{ij} \mathrm{s}_j)$ and single variable bias potentials $\exp(b_i \mathrm{s}_i)$, each of which are explicitly represented by labelled factors in Figure B.3. A corresponding undirected graphical model representation would have a single clique involving all five variables, and so would not indicate any information about the factorisation of the joint density.

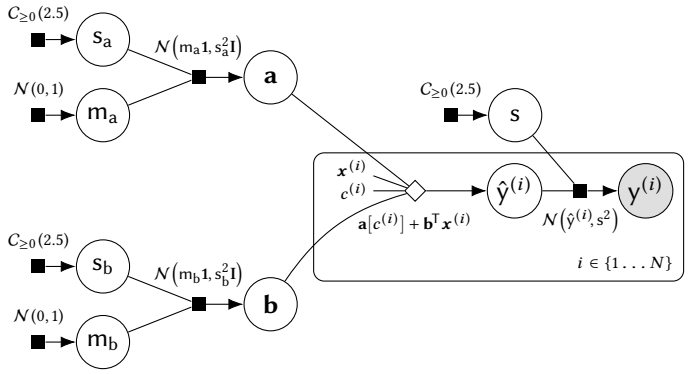


Figure B.4.: Hierarchical linear regression model factor graph showing examples of extended factor graph notation.

In Figure B.4 we illustrate some additional useful factor graph notation we will use in this thesis. We use a factor graph corresponding to a hierarchical linear regression model which will be discussed in more detail later in the thesis as a motivating example. The exact meaning of the model and its various factors are unimportant to the discussion of notation here so will be skipped for now.

It will often be useful to be able to explicitly represent deterministic functions applied to the random variables in a factor graph. For this purpose we introduce an additional node type denoted by an unfilled diamond (◇). The semantics of this node type are very similar to standard directed factor nodes. Variables acting as inputs to the function are connected to the node by undirected edges and the variable corresponding to the function output indicated by a directed edge from the node to the relevant variable. Like standard factor nodes, the deterministic factor nodes only ever connect to variable nodes. The operations performed by the function on the inputs will usually be included as a label adjacent to the node as illustrated by the example in Figure B.4.

A deterministic factor node can informally[1] be considered equivalent to a directed factor node with a degenerate Dirac delta conditional density on the output variable which concentrates all the probability mass at the output of the function applied to the inputs variables. The previously discussed rules for evaluating conditional independency properties in factor graphs can be directly extended to account for the new node type by just considering it as a directed factor node.

1 A Dirac delta cannot strictly define a density as it is not the Radon–Nikodyn derivative of an absolutely continuous measure, however it can be informally treated as the density of a singular Dirac measure $f(0) = \int f(x)\,\mathrm{d}\delta(x) \simeq \int f(x)\delta(x)\,\mathrm{d}x$.

Optionally constant values used in a model may be included in a factor graph as plain nodes indicated only by a label. The $\boldsymbol{x}^{(i)}$ and $c^{(i)}$ nodes in Figure B.4 are an example of this notation.

A commonly used convention in factor graphs (and other graphical models) is *plate notation* [37], with an example of a plate shown by the rounded rectangle bounding some of the nodes in Figure B.4. Plates are used to indicate a subgraph in the model which is replicated multiple times (with the replications being indexed over a set which is typically indicated in the lower right corner of the plate as in Figure B.4). The subgraph entirely contained on the plate is assumed to be replicated the relevant number of times, with any edges crossing into the plate from variable nodes outside of the plate being repeated once for each subgraph replication. Plates are commonly used to represent a model component repeated across multiple data items.

Each of the factors in Figure B.4 is labelled with a shorthand for a probability density function corresponding to the conditional or marginal density factor associated with the node. Definitions for the shorthand notations that are used for densities in this thesis are given in Tables A.2 and A.3. The dependence of the factors on the value of the random variable the density is defined on is omitted in the labels for brevity.

A final additional notation used in Figure B.4 is the use of a shaded variable node (corresponding to $\mathsf{y}^{(i)}$) to indicate a random variable corresponding to an observable quantity in the model.

B.3 COMPUTATION GRAPHS

A final graph based tool we will make use of in this theis is that of *computation graphs* [15]. In particular computation graphs (via associated software frameworks [219]) will be used to allow automatic differentiation of complex probabilistic models used in later chapters. Computation graphs are not typically considered in the context of probabilistic graphical models, but they share many of the same features and as we will see are closely related to directed factor graphs.

A *computation graph*, sometimes insead termed a *computational graph* or *data flow graph*, represents the computations involved in evaluating a mathematical expression. In this thesis we will distinguish between two types of nodes in a computation graph. *Variable nodes* correspond

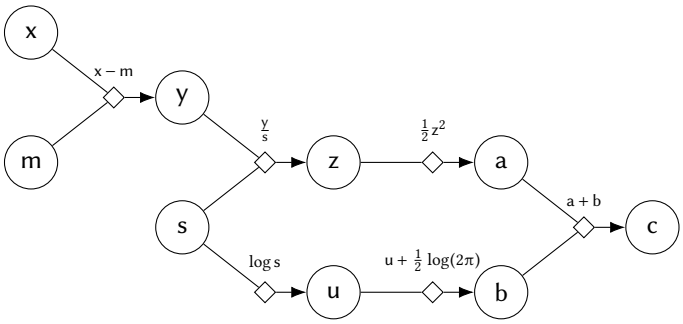


Figure B.5.: Example computation graph corresponding to calculation of the negative log density of a univariate normal distribution.

to variables which hold either inputs to the computation or intermediate results corresponding to the outputs of sub-expressions. *Operation nodes* describe how non-input variable nodes are computed as functions of other variable nodes. In other presentations of computation graphs often the operation nodes are instead implicitly represented by directed edges between variable nodes. However analagously to the more explicit factorisation afforded by directed factor graphs compared to directed graphical models, directly representing operations as nodes allows finer grained information about the decomposition of the operations associated with a computation graph to be included.

As with directed graphical models and directed factor graphs, computation graphs cannot contain directed cycles. This does not preclude recursive and recurrent computations however as these can always be unrolled to form a directed acyclic graph. The 'mathematical expressions' a computation graph is constructed to evaluate can be arbitarily complex - a computation graph corresponding to the evaluation of any numerical algorithm can always be constructed including use of arbitrary nested flow control and branching statements.

An example of a computation graph representing the calculation of the negative log density of a univariate normal distribution, i.e.

$$c = \frac{1}{2}\left(\frac{x - m}{s}\right)^2 + \log s + \frac{1}{2}\log(2\pi) \tag{B.6}$$

is shown in Figure B.5. The graph inputs have chosen to be the value of the random variable (x) to evaluate the density at and the mean (m) and the standard deviation (s) parameters of the density.

Variable nodes in the computation graph have been represented by labelled circles and operation nodes with labelled diamonds. Undirected edges connecting from a variable node to an operation node correspond to the inputs to the operation, and directed edges from an operation node to variable nodes to the outputs of the operation.

The computation graph associated with a given expression is not uniquely defined. There will usually be multiple possible orderings in which operations can be applied to achieve the same result (up to differences due to non-exact floating point computation). Similarly what should be considered a single operation to be represented by a node in the computation graph as opposed to being split up into a sub-graph of multiple operations is a matter of choice. For example in Figure B.5 the addition of the constant $\frac{1}{2}\log(2\pi)$ could have been included at various other points in the graph and the operation $\frac{1}{2}z^2$ could have been split in to separate multiplication and exponentation operations.

B.4 AUTOMATIC DIFFERENTIATION

The main motivation for representing expressions as computation graphs is to formalise an efficient general procedure termed automatic differentiation for automatically calculating derivatives of the output of an expression with respect to its inputs [20, 165]. The key ideas in automatic differentiation are to use the chain rule to decompose the derivatives into products and sums of the partial derivatives of the output of each individual operation in the expression with respect to its input, and to use an efficient recursive accumulation of these partial derivative sum-products corresponding to a traversal of the computation graph such that multiple derivatives can be efficiently calculated together.

Depending on how the computation graph is traversed to accumulate the derivative terms, different modes of automatic differentiation are possible. Of most use in this thesis will be *reverse-mode accumulation* [211], in which the derivatives of an output node with respect to all input nodes are accumulated by a reverse pass through the computation graph from the output node to inputs.

As an example the partial derivatives of the expression for univariate normal log density given in (B.6) with respect to x, m and s can be

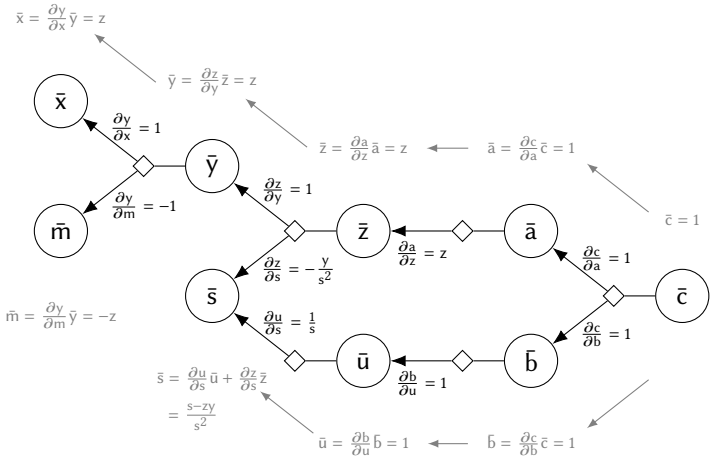


Figure B.6.: Visualisation of applying reverse-mode automatic differentiation to the computation graph in Figure B.5 to calculate the derivatives of the negative log density of a univariate normal distribution.

decomposed using the chain rule in terms of the intermediate variables in the computation graph shown in Figure B.5 as

$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial x}, \tag{B.7}$$

$$\frac{\partial c}{\partial m} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial m}, \tag{B.8}$$

$$\frac{\partial c}{\partial s} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s} + \frac{\partial c}{\partial b}\frac{\partial b}{\partial u}\frac{\partial u}{\partial s}. \tag{B.9}$$

We can immediately see that some of the chains of products of partial derivatives are repeated in the different derivative expressions - for example $\frac{\partial c}{\partial a}\frac{\partial a}{\partial z}$ appears in the expressions for all three derivatives. Reverse-mode accumulation is effectively an automatic way of exploiting these possibilities for reusing calculations.

Figure B.6 shows a visualisation of reverse-mode accumulation applied to the computation graph in Figure B.5. The first step is for a *forward pass* through the graph to be performed, i.e. values are provided for each of the input variables and then each of the intermediate and output variables calculated from the incoming operation applied to their parent values. Importantly the values of all variables in the graph calculated during the forward pass must be maintained in memory.

The *reverse pass* recursively calculates the values of the partial derivatives of the relevant output node with respect to each variable node in the graph - we will term these intermediate derivatives *accumulators*

Algorithm 13 Reverse-mode automatic differentiation.

Input: $\{x_i\}_{i=1}^{M}$: computation graph input variables,
Pa : indices of parent variables to an operation given its index,
Ch : indices to child operations of a variable given its index,
$\{f_i\}_{i=M+1}^{N}$: computation graph operations in topological order,
$\left\{\{\partial_j f_i\}_{j\in \mathrm{Pa}(i)}\right\}_{i=M+1}^{N}$: operation partial derivatives wrt parent variables.
Output: x_N : function output,
$\{\bar{x}_i\}_{i=1}^{M}$: partial derivatives of function output wrt input variables.

for $i \in \{M+1 \dots N\}$ **do** ▷ Forward pass
  $x_i \leftarrow f_i\left(\{x_j\}_{j\in \mathrm{Pa}(i)}\right)$
$\bar{x}_N \leftarrow 1$ ▷ Reverse pass
for $i \in \{N-1 \dots 1\}$ **do**
  $\bar{x}_i \leftarrow \sum_{j\in \mathrm{Ch}(i)} \bar{x}_j \partial_i f_j(x_i)$
return x_N, $\{\bar{x}_i\}_{i=1}^{M}$

denoted with barred symbols in Figure B.6 e.g. $\bar{a} = \frac{\partial c}{\partial a}$. The reverse pass begins by seeding an accumulator for the output node to one (i.e. $\bar{c} = \frac{\partial c}{\partial c} = 1$ in Figure B.6).

Accumulators for the input variables of an operation are calculated by multiplying the accumulator for the operation output by the partial derivatives of the operation output with respect to each input variable. For non-linear operations multiplying by the operator partial derivatives will require access to the value of the input variables calculated in the forward pass. If a variable is an input to multiple operations, the derivative terms from each operation are added together in the relevant accumulator, as for example shown for $\bar{s}$ in Figure B.6. By recursively applying these product and sum operations, the derivatives of the output with respect to all variables in the graph can be calculated. A general description of the method for computation graphs with a single output node and multiple inputs is given in Algorithm 13.

This reverse accumulation method allows computation of numerically exact (up to floating point error) derivatives of a single output variable in a computation graph with respect to *all input variables* with a computational cost, in terms of the number of atomic operations which need to be performed, that is a constant factor of the cost of the evaluation of the original expression represented by the computation graph in the forward pass. The constant factor is typically two to three and at most six [16]. This efficient computational cost is balanced by the requirement that the values of all intermediate variables in the computation graph evaluated in the forward pass through the graph must be stored

in memory for the derivative accumulation in a reverse pass, which for large computational graphs can become a bottleneck.

To calculate the full Jacobian from a computation graph representing a function with M inputs $\{\mathsf{x}_i\}_{i=1}^{M}$ and N outputs $\{\mathsf{y}_i\}_{i=1}^{N}$, i.e. the $N \times M$ matrix $\boldsymbol{J}$ with entries $J_{i,j} = \frac{\partial \mathsf{y}_i}{\partial \mathsf{x}_j}$, we can do a single forward pass and N reverse passes each time accumulating the derivatives of one output variable with respect to all inputs. This leads to an overall computational cost that is $O(N)$ times the cost of a single (forward) function evalaution to evaluate the full Jacobian. As each of the reverse passes can trivially be run in parallel (in addition to any parallelisation of the operations in the forward and reverse passes themselves), this $O(N)$ factor in the operation count need not corresponds to an equivalent increase in compute time.

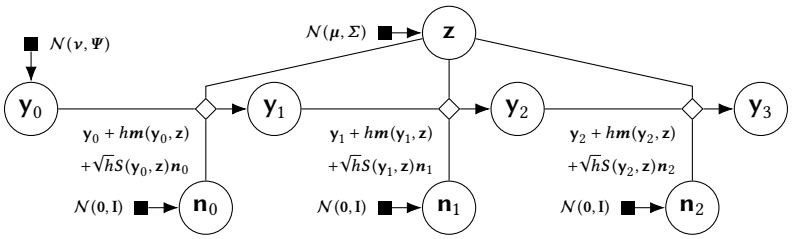
An alternative to reverse-mode accumulation is *forward-mode accumulation* [233], which insteads accumulates partial derivatives with respect to a single input variable alongside the forward pass through the graph. In contrast to reverse-mode, this allows calculation of the partial derivatives of all output variables with respect to a single input variable at a computational cost that is a constant factor of the cost of the evaluation of the original expression in the forward pass. Forward-mode accumulation therefore allows evaluation of the Jacobian of a function with M inputs and N outputs at an overall computational cost that is $O(M)$ times the cost of a single function evaluation.

For functions with $M \gg N$, e.g. scalar valued functions of multiple inputs, reverse-mode accumulation is generally therefore signficantly more efficient at computing the Jacobian. Forward-mode accumulation is however useful for evaluating the Jacobian of functions with $N \gg M$, and also has the advantage over reverse-mode accumulation of avoiding the requirement to store the values of intermediate variables from the forward pass for the reverse pass(es).

The direct overlap in our notation to represent variable and operation nodes in computation graphs and that used to represent (random) variable nodes and deterministic factor nodes in factor graphs is intentional. Although often the operations associated with a deterministic node in a factor graph will be more complex than the operations usually represented by nodes in a computation graph, this is only a matter of granularity of reprensentation - fundamentally they perform the same role.

$\mathbf{y}_0 \sim \mathcal{N}(\boldsymbol{\nu}, \boldsymbol{\Psi})$
$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
for $t \in \{1 \dots T\}$ **do**
  $\mathbf{n}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{y}_t \leftarrow \mathbf{y}_{t-1} + h\,\boldsymbol{m}(\mathbf{y}_{t-1}, \mathbf{z}) + \sqrt{h}\,\boldsymbol{S}(\mathbf{y}_{t-1}, \mathbf{z})\mathbf{n}_{t-1}$

(a) Pseudo-code for Euler–Maruyama simulation of SDE model.



(b) Directed factor graph of 3 time steps of SDE simulation.

Figure B.7.: Example of a simulator model corresponding to Euler–Maruyma integration of a set of *stochastic differential equations* (SDEs), $\mathrm{d}\mathbf{y}(t) = \boldsymbol{m}(\mathbf{y}(t), \mathbf{z})\,\mathrm{d}t + \boldsymbol{S}(\mathbf{y}(t), \mathbf{z})\,\mathrm{d}\mathbf{n}(t)$, specified as pseudo-code in (a) and a directed factor graph in (b). In the pseudo-code the notation $\sim$ followed by a distribution shorthand represents generating a value from the associated distribution and assigning it to a variable.

Importantly this means we can treat subgraphs of a factor graphs consisting of only variable and deterministic factor nodes as computation graphs and if the operations performed by the deterministic nodes are differentiable, use reverse-mode automatic differentiation to efficiently propagate derivatives through these sub-graphs.

Like directed graphical models, a directed factor graph naturally specifies a generative process via ancestral sampling, with values for the random variables in the graph successively calculated in a forward pass consisting of a combination of deterministic and stochastic operations on the values of parent variables. A computation graph likewise specifies a generative process, how to compute the expression outputs given inputs, computed via a forward pass through the graph with the main differences being here that the inputs to that process are assumed to be given rather than sampled from marginal densities and the intermediate operations are all deterministic.

B.5 SIMULATORS

Rather than specifying a generative model via a directed factor graph (or graphical model), it is common for complex models to instead be specified procedurally in code as a *simulator*. Often such simulators may involve a mechanistic model of a physical process for example described by a set of *stochastic differential equations* (SDEs). Any stochasticity in a simulator model will be introduced via draws from a pseudo-random number generator in the programming language used to specify the model. Given these random inputs, the output of the simulator is then calculated as a series of determinstic operations performed to the inputs and so can be described by a computation graph. The overall composition of directed factor nodes specifying the generation of random inputs from known densities by the random number generator and computation graph describing the operations performed by the simulator code together therefore define a directed factor graph from which we can extract a joint density on all the variables in the models as the product of all factors (with implicit Dirac delta terms on the outputs of deterministic factors). An example of a simulator model corresponding to Euler–Maruyma approximate integration [113] of a set of SDEs is shown as both pseudo-code and a factor graph in Figure B.7.

A key difference of simulator models from more typical probabilistic models is that the variables corresponding to observables in the factor graph of a simulator model may be the output of deterministic factors rather than probabilistic directed factors.

BIBLIOGRAPHY

[1] Ijaz Akhter and Michael J. Black. 'Pose-Conditioned Joint Angle Limits for 3D Human Pose Reconstruction'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[2] Shun-Ichi Amari. 'Differential geometry of curved exponential families-curvatures and information loss'. In: *The Annals of Statistics* (1982), pp. 357–385.

[3] Hans C Andersen. 'RATTLE: A velocity version of the SHAKE algorithm for molecular dynamics calculations'. In: *Journal of Computational Physics* (1983).

[4] Christophe Andrieu, Arnaud Doucet and Roman Holenstein. 'Particle Markov chain Monte Carlo methods'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3 (2010), pp. 269–342.

[5] Christophe Andrieu and Gareth O Roberts. 'The pseudo-marginal approach for efficient Monte Carlo computations'. In: *The Annals of Statistics* (2009).

[6] Christophe Andrieu and Johannes Thoms. 'A tutorial on adaptive MCMC'. In: *Statistics and computing* 18.4 (2008), pp. 343–373.

[7] Martín Arjovsky and Léon Bottou. 'Towards Principled Methods for Training Generative Adversarial Networks'. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2017.

[8] IEEE Standards Association. 'Standard for Floating-Point Arithmetic'. In: *IEEE 754-2008* (2008).

[9] David Barber and Wim Wiegerinck. 'Tractable variational structures for approximating graphical models'. In: *Advances in Neural Information Processing Systems*. 1999, pp. 183–189.

[10] Chris P. Barnes, Sarah Filippi, Michael P. H. Stumpf and Thomas Thorne. 'Considerate approaches to constructing summary statistics for ABC model selection'. In: *Statistics and Computing* 22.6 (2012), pp. 1181–1197. URL: http://dx.doi.org/10.1007/s11222-012-9335-7.

[11] Alessandro Barp, Francois-Xavier Briol, Anthony D Kennedy and Mark Girolami. 'Geometry and Dynamics for Markov Chain Monte Carlo'. In: *arXiv preprint arXiv:1705.02891* (2017).

[12] Matthias Bartelmann and Peter Schneider. 'Weak gravitational lensing'. In: *Physics Reports* 340.4 (2001), pp. 291–472.

[13] Eric Barth, Krzysztof Kuczera, Benedict Leimkuhler and Robert D Skeel. 'Algorithms for constrained molecular dynamics'. In: *Journal of computational chemistry* (1995).

[14] Simon Barthelmé and Nicolas Chopin. 'Expectation propagation for likelihood-free inference'. In: *Journal of the American Statistical Association* 109.505 (2014), pp. 315–333.

[15] Friedrich L Bauer. 'Computational graphs and rounding error'. In: *SIAM Journal on Numerical Analysis* 11.1 (1974), pp. 87–96.

[16] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul and Jeffrey Mark Siskind. 'Automatic differentiation in machine learning: a survey'. In: *arXiv preprint arXiv:1502.05767* (2015).

[17] Mark A Beaumont. 'Estimation of population growth or decline in genetically monitored populations'. In: *Genetics* 164.3 (2003), pp. 1139–1160.

[18] Mark A Beaumont, Wenyang Zhang and David J Balding. 'Approximate Bayesian computation in population genetics'. In: *Genetics* (2002).

[19] Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin and Christian P Robert. 'Adaptive approximate Bayesian computation'. In: *Biometrika* 96.4 (2009), pp. 983–990.

[20] L. M. Beda, L. N. Korolev, N. V. Sukkikh and T. S. Frolova. *Programs for automatic differentiation for the machine BESM*. Technical Report. (In Russian). Moscow, USSR: Institute for Precise Mechanics and Computation Techniques, Academy of Science, 1959.

[21] Michael Betancourt. 'A general metric for Riemannian manifold Hamiltonian Monte Carlo'. In: *Geometric science of information*. Springer, 2013.

[22] Michael Betancourt. 'The fundamental incompatibility of scalable Hamiltonian Monte Carlo and naive data subsampling'. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

[23] Michael Betancourt. 'A Conceptual Introduction to Hamiltonian Monte Carlo'. In: *arXiv preprint arXiv:1701.02434* (2017).

[24] Michael Betancourt and Mark Girolami. 'Hamiltonian Monte Carlo for hierarchical models'. In: *Current trends in Bayesian methodology with applications* 79 (2015), p. 30.

[25] Joris Bierkens. 'Non-reversible Metropolis–Hastings'. In: *Statistics and Computing* 26.6 (2016), pp. 1213–1228.

[26] George D Birkhoff. 'Proof of the ergodic theorem'. In: *Proceedings of the National Academy of Sciences* 17.12 (1931), pp. 656–660.

[27] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN: 9780387310732.

[28] Christopher M Bishop, David Spiegelhalter and John Winn. 'VIBES: A variational inference engine for Bayesian networks'. In: *Advances in Neural Information Processing Systems*. Vol. 15. 2002, pp. 777–784.

[29] David M Blei, Alp Kucukelbir and Jon D McAuliffe. 'Variational inference: A review for statisticians'. In: *Journal of the American Statistical Association* just-accepted (2017).

[30] Michael GB Blum. 'Approximate Bayesian computation: a nonparametric perspective'. In: *Journal of the American Statistical Association* 105.491 (2010), pp. 1178–1187.

[31] Michael GB Blum, Maria Antonieta Nunes, Dennis Prangle, Scott A Sisson et al. 'A comparative review of dimension reduction methods in approximate Bayesian computation'. In: *Statistical Science* 28.2 (2013), pp. 189–208.

[32] Georges Bonnet. 'Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire'. In: *Annals of Telecommunications* 19.9 (1964), pp. 203–220.

[33] Luke Bornn, Natesh S Pillai, Aaron Smith and Dawn Woodard. 'The use of a single pseudo-sample in approximate Bayesian computation'. In: *Statistics and Computing* 27.3 (2017), pp. 583–590.

[34] George EP Box. 'Sampling and Bayes' inference in scientific modelling and robustness'. In: *Journal of the Royal Statistical Society. Series A (General)* (1980), pp. 383–430.

[35] George EP Box, Mervin E Muller et al. 'A note on the generation of random normal deviates'. In: *The Annals of Mathematical Statistics* 29.2 (1958), pp. 610–611.

[36] Marcus A Brubaker, Mathieu Salzmann and Raquel Urtasun. 'A Family of MCMC Methods on Implicitly Defined Manifolds.' In: *International Conference on Artificial Intelligence and Statistics*. 2012.

[37] Wray L Buntine. 'Operations for learning with graphical models'. In: *Journal of artificial intelligence research* (1994).

[38] Simon Byrne and Mark Girolami. 'Geodesic Monte Carlo on embedded manifolds'. In: *Scandinavian Journal of Statistics* (2013).

[39] Bob Carpenter, Andrew Gelman, Matt Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Michael A Brubaker, Jiqiang Guo, Peter Li and Allen Riddell. 'Stan: A probabilistic programming language'. In: *Journal of Statistical Software* (2016).

[40] KS Chan. 'Asymptotic behavior of the Gibbs sampler'. In: *Journal of the American Statistical Association* 88.421 (1993), pp. 320–326.

[41] Kung Sik Chan and Charles J Geyer. 'Discussion: Markov chains for exploring posterior distributions'. In: *The Annals of Statistics* 22.4 (1994), pp. 1747–1758.

[42] Ming-Hui Chen and Bruce Schmeiser. 'Toward black-box sampling: A random-direction interior-point Markov chain approach'. In: *Journal of Computational and Graphical Statistics* 7.1 (1998), pp. 1–22.

[43] Tianqi Chen, Emily Fox and Carlos Guestrin. 'Stochastic Gradient Hamiltonian Monte Carlo'. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014.

[44] Nicolas Chopin and Sumeetpal S Singh. 'On particle Gibbs sampling'. In: *Bernoulli* 21.3 (2015), pp. 1855–1883.

[45] TM Christensen, AS Hurn and KA Lindsay. 'The devil is in the detail: hints for practical optimisation'. In: *Economic Analysis and Policy* 38.2 (2008), pp. 345–368.

[46] Richard T Cox. 'Probability, frequency and reasonable expectation'. In: *American Journal of Physics* 14.1 (1946), pp. 1–13. URL: http://dx.doi.org/10.1119/1.1990764.

[47] Richard T Cox. 'The algebra of probable inference'. In: *American Journal of Physics* 31.1 (1963), pp. 66–67. URL: http://dx.doi.org/10.1119/1.1969248.

[48] H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1946.

[49] Johan Dahlin, Fredrik Lindsten, Joel Kronander and Thomas B Schön. 'Accelerating pseudo-marginal Metropolis-Hastings by correlating auxiliary variables'. In: *arXiv preprint arXiv:1511.05483* (2015).

[50] P Damien, John Wakefield and Stephen Walker. 'Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.2 (1999), pp. 331–344.

[51] Philip J. Davis and Philip Rabinowitz. *Numerical Integration*. Blaisdell Publishing Company, 1967.

[52] Pierre Del Moral, Arnaud Doucet and Ajay Jasra. 'An adaptive sequential Monte Carlo method for approximate Bayesian computation'. In: *Statistics and Computing* 22.5 (2012), pp. 1009–1020.

[53] George Deligiannidis, Arnaud Doucet, Michael K Pitt and Robert Kohn. 'The Correlated Pseudo-Marginal Method'. In: *arXiv preprint arXiv:1511.04992* (2015).

[54] Persi Diaconis, Susan Holmes and Radford M Neal. 'Analysis of a nonreversible Markov chain sampler'. In: *Annals of Applied Probability* (2000), pp. 726–752.

[55] Persi Diaconis, Susan Holmes and Mehrdad Shahshahani. 'Sampling from a manifold'. In: *Advances in Modern Statistical Theory and Applications*. Institute of Mathematical Statistics, 2013, pp. 102–125.

[56] Adji B Dieng, Dustin Tran, Rajesh Ranganath, John Paisley and David M Blei. 'The χ-Divergence for Approximate Inference'. In: *arXiv preprint arXiv:1611.00328* (2016).

[57] Peter J Diggle and Richard J Gratton. 'Monte Carlo methods of inference for implicit statistical models'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1984), pp. 193–227.

[58] A. Doucet, A. Smith, N. de Freitas and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Information Science and Statistics. Springer New York, 2001. ISBN: 9780387951461. URL: https://books.google.co.uk/books?id=uxX-koqKtMMC.

[59] Arnaud Doucet, MK Pitt, George Deligiannidis and Robert Kohn. 'Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator'. In: *Biometrika* 102.2 (2015), pp. 295–313.

[60] Oliver B Downs, David JC MacKay and Daniel D Lee. 'The nonnegative Boltzmann machine'. In: *Advances in Neural Information Processing Systems*. 2000, pp. 428–434.

[61] Simon Duane, Anthony D Kennedy, Brian J Pendleton and Duncan Roweth. 'Hybrid Monte Carlo'. In: *Physics Letters B* (1987).

[62] Gintare Karolina Dziugaite, Daniel M Roy and Zoubin Ghahramani. 'Training generative neural networks via maximum mean discrepancy optimization'. In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. AUAI Press. 2015, pp. 258–267.

[63] Robert G Edwards and Alan D Sokal. 'Generalization of the Fortuin–Kasteleyn–Swendsen–Wang representation and Monte Carlo algorithm'. In: *Physical review D* 38.6 (1988), p. 2009.

[64] Vassiliy A Epanechnikov. 'Non-parametric estimation of a multivariate probability density'. In: *Theory of Probability & Its Applications* 14.1 (1969), pp. 153–158.

[65] Tim van Erven and Peter Harremos. 'Rényi divergence and Kullback-Leibler divergence'. In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 3797–3820.

[66] Herbert Federer. *Geometric measure theory*. Springer, 2014.

[67] Maurizio Filippone and Mark Girolami. 'Pseudo-marginal Bayesian inference for Gaussian processes'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2214–2226.

[68] Bruno de Finetti. 'Foresight: its logical laws, its subjective sources'. In: *Studies in Subjective Probability*. Ed. by H. E. Kyburg. English translation of original 1937 French article *La Prévision: ses lois logiques, ses sources subjectives*. Springer, 1992, pp. 134–174.

[69] Brendan J Frey. 'Extending factor graphs so as to unify directed and undirected graphical models'. In: *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 2002, pp. 257–264. URL: https://arxiv.org/abs/1212.2486.

[70] Brendan J Frey, Frank R Kschischang, Hans-Andrea Loeliger and Niclas Wiberg. 'Factor graphs and algorithms'. In: *Proceedings of the 35th Annual Allerton Conference on Communication Control and Computing*. 1997.

[71] Yun-Xin Fu and Wen-Hsiung Li. 'Estimating the age of the common ancestor of a sample of DNA sequences.' In: *Molecular biology and evolution* 14.2 (1997), pp. 195–199.

[72] Alan E Gelfand and Adrian FM Smith. 'Sampling-based approaches to calculating marginal densities'. In: *Journal of the American Statistical Association* (1990).

[73] Andrew Gelman, Walter R Gilks and Gareth O Roberts. 'Weak convergence and optimal scaling of random walk Metropolis algorithms'. In: *The annals of applied probability* 7.1 (1997), pp. 110–120.

[74] Andrew Gelman, Daniel Lee and Jiqiang Guo. 'Stan: A probabilistic programming language for Bayesian inference and optimization.' In: *Journal of Educational and Behavioral Statistics* 40.5 (2015), pp. 530–543.

[75] Andrew Gelman and Donald B Rubin. 'Inference from iterative simulation using multiple sequences'. In: *Statistical science* (1992), pp. 457–472.

[76] Andrew Gelman and Cosma Rohilla Shalizi. 'Philosophy and the practice of Bayesian statistics'. In: *British Journal of Mathematical and Statistical Psychology* 66.1 (2013), pp. 8–38.

[77] Stuart Geman and Donald Geman. 'Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images'. In: *IEEE Transactions on pattern analysis and machine intelligence* (1984).

[78] Charles J Geyer. *Markov chain Monte Carlo lecture notes (Statistics 8931)*. Tech. rep. University of Minnesota, 1998.

[79] Charles J Geyer. 'The Metropolis-Hastings-Green Algorithm'. 2003. URL: http://www.stat.umn.edu/geyer/f05/8931/bmhg.pdf.

[80] W. Gilchrist. *Statistical Modelling with Quantile Functions*. CRC Press, 2000.

[81] Wally R Gilks, Andrew Thomas and David J Spiegelhalter. 'A language and program for complex Bayesian modelling'. In: *The Statistician* (1994), pp. 169–177.

[82] Walter R Gilks and Pascal Wild. 'Adaptive rejection sampling for Gibbs sampling'. In: *Applied Statistics* (1992), pp. 337–348.

[83] Mark Girolami and Ben Calderhead. 'Riemann-manifold Langevin and Hamiltonian Monte Carlo methods'. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214.

[84] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[85] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. 'Generative adversarial nets'. In: *Advances in Neural Information Processing Systems*. 2014.

[86] Claire C. Gordon, Thomas Churchill, Charles E. Clauser, Bruce Bradtmiller, John T. McConville, Ilse Tebbets and Robert A. Walker. *Anthropometric Survey of US Army Personell: Final Report*. Tech. rep. United States Army, 1988.

[87] Neil J Gordon, David J Salmond and Adrian FM Smith. 'Novel approach to nonlinear/non-Gaussian Bayesian state estimation'. In: *IEE Proceedings F (Radar and Signal Processing)*. Vol. 140. 2. IET. 1993, pp. 107–113.

[88] Christian Gourieroux, Alain Monfort and Eric Renault. 'Indirect inference'. In: *Journal of applied econometrics* 8.S1 (1993), S85–S118.

[89] Alex Graves. 'Practical Variational Inference for Neural Networks'. In: *Advances in Neural Information Processing Systems 24*. 2011, pp. 2348–2356.

[90] Todd L Graves. 'Automatic step size selection in random walk Metropolis algorithms'. In: *arXiv preprint arXiv:1103.5986* (2011).

[91] Peter J Green. 'Reversible jump Markov chain Monte Carlo computation and Bayesian model determination'. In: *Biometrika* (1995), pp. 711–732.

[92] J. E. Gubernatis. 'Marshall Rosenbluth and the Metropolis algorithm'. In: *Physics of Plasmas* 12.5 (2005), p. 057303. URL: http://dx.doi.org/10.1063/1.1887186.

[93] Heikki Haario, Eero Saksman and Johanna Tamminen. 'An adaptive Metropolis algorithm'. In: *Bernoulli* (2001), pp. 223–242.

[94] Theodore E Harris. 'The existence of stationary measures for certain Markov processes'. In: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 2. 1956, pp. 113–124.

[95] Carsten Hartmann and Christof Schutte. 'A constrained hybrid Monte-Carlo algorithm and the problem of calculating the free energy in several variables'. In: *ZAMM-Zeitschrift fur Angewandte Mathematik und Mechanik* (2005).

[96] David Harvey, Thomas D Kitching, Joyce Noah-Vanhoucke, Ben Hamner, Tim Salimans and AM Pires. 'Observing Dark Worlds: A crowdsourcing experiment for dark matter mapping'. In: *Astronomy and Computing* 5 (2014), pp. 35–44.

[97] L. Hascoët and V. Pascual. 'The Tapenade Automatic Differentiation tool: Principles, Model, and Specification'. In: *ACM Transactions On Mathematical Software* 39.3 (2013). URL: http://dx.doi.org/10.1145/2450153.2450158.

[98] Cecil Hastings Jr, Frederick Mosteller, John W Tukey and Charles P Winsor. 'Low moments for small samples: a comparative study of order statistics'. In: *The Annals of Mathematical Statistics* (1947), pp. 413–426.

[99] W Keith Hastings. 'Monte Carlo sampling methods using Markov chains and their applications'. In: *Biometrika* (1970).

[100] Bryan D He, Christopher M De Sa, Ioannis Mitliagkas and Christopher Ré. 'Scan Order in Gibbs Sampling: Models in Which it Matters and Bounds on How Much'. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1–9.

[101] David M Higdon. 'Auxiliary variable methods for Markov chain Monte Carlo with applications'. In: *Journal of the American Statistical Association* 93.442 (1998), pp. 585–595.

[102] Matthew D Hoffman and Andrew Gelman. 'The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.' In: *Journal of Machine Learning Research* (2014).

[103] Matthew D Hoffman, David M Blei, Chong Wang and John William Paisley. 'Stochastic variational inference.' In: *Journal of Machine Learning Research* (2013).

[104] Matthew Hoffman and David Blei. 'Structured stochastic variational inference'. In: *Artificial Intelligence and Statistics*. 2015, pp. 361–369.

[105] Akihisa Ichiki and Masayuki Ohzeki. 'Violation of detailed balance accelerates relaxation'. In: *Physical Review E* 88.2 (2013), p. 020101.

[106] Eric Jullo, Jean-Paul Kneib, Marceau Limousin, Ardis Eliasdottir, PJ Marshall and Tomas Verdugo. 'A Bayesian approach to strong lensing modelling of galaxy clusters'. In: *New Journal of Physics* 9.12 (2007), p. 447. URL: https://arxiv.org/abs/0706.0048.

[107] Kaggle. *Observing Dark Worlds*. https://www.kaggle.com/c/DarkWorlds. 2012.

[108] Herman Kahn and Theodore E Harris. 'Estimation of particle transmission by random sampling'. In: *National Bureau of Standards applied mathematics series* 12 (1951), pp. 27–30.

[109] Rudolph Emil Kalman. 'A new approach to linear filtering and prediction problems'. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

[110] AD Kennedy and Julius Kuti. 'Noise without noise: a new Monte Carlo method'. In: *Physical review letters* 54.23 (1985), p. 2473.

[111] Ross Kindermann and Laurie Snell. *Markov random fields and their applications*. American Mathematical Society, 1980.

[112] Diederik P Kingma and Max Welling. 'Auto-Encoding Variational Bayes'. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. 2013.

[113] P.E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*. Applications of Mathematics. Springer-Verlag, 1992. ISBN: 9783540540625.

[114] David A Knowles and Tom Minka. 'Non-conjugate variational message passing for multinomial and binary regression'. In: *Advances in Neural Information Processing Systems*. 2011, pp. 1701–1709.

[115] Andreĭ Nikolaevich Kolmogorov. *Foundations of the Theory of Probability*. Ed. by Nathan Morrison. 2nd English Edition. English translation of original 1933 German monograph, *Grundbegriffe der Wahrscheinlichkeitrechnung*. Chelsea Publishing Company, 1956. URL: https://pdfs.semanticscholar.org/c3e1/51f71168a5f348bdebfde11752ca603fa6d0.pdf.

[116] Augustine Kong. *A note on importance sampling using standardized weights*. Technical Report 348. Department of Statistics, University of Chicago, 1992.

[117] Dirk P. Kroese, Thomas Taimre and Zdravko I. Botev. 'Variance Reduction'. In: *Handbook of Monte Carlo Methods*. John Wiley & Sons, Inc., 2011, pp. 347–380. ISBN: 9781118014967. DOI: 10.1002/9781118014967.ch9. URL: http://dx.doi.org/10.1002/9781118014967.ch9.

[118] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman and David M Blei. 'Automatic Differentiation Variational Inference'. In: *arXiv preprint arXiv:1603.00788* (2016).

[119] Solomon Kullback and Richard A Leibler. 'On information and sufficiency'. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[120] Steffen L Lauritzen and David J Spiegelhalter. 'Local computations with probabilities on graphical structures and their application to expert systems'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1988), pp. 157–224. DOI: 10.2307/2345762.

[121] Derrick H Lehmer. 'Mathematical methods in large-scale computing units'. In: *Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery (1949)*. 1951, pp. 141–146.

[122] Benedict J Leimkuhler and Robert D Skeel. 'Symplectic numerical integrators in constrained Hamiltonian systems'. In: *Journal of Computational Physics* (1994).

[123] Benedict Leimkuhler and Charles Matthews. 'Efficient molecular dynamics using geodesic integration and solvent–solute splitting'. In: *Proc. R. Soc. A*. The Royal Society. 2016.

[124] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*. Cambridge University Press, 2004.

[125] Tony Lelièvre, Mathias Rousset and Gabriel Stoltz. 'Langevin dynamics with constraints and computation of free energy differences'. In: *Mathematics of computation* (2012).

[126] Yingzhen Li and Richard E Turner. 'Rényi divergence variational inference'. In: *Advances in Neural Information Processing Systems*. 2016.

[127] Yujia Li, Kevin Swersky and Rich Zemel. 'Generative moment matching networks'. In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 1718–1727.

[128] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[129] Anne-Marie Lyne, Mark Girolami, Yves Atchadé, Heiko Strathmann, Daniel Simpson et al. 'On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods'. In: *Statistical science* 30.4 (2015), pp. 443–467.

[130] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[131] Olvi L Mangasarian, W Nick Street and William H Wolberg. 'Breast cancer diagnosis and prognosis via linear programming'. In: *Operations Research* 43.4 (1995), pp. 570–577.

[132] Jean-Michel Marin, Pierre Pudlo, Christian P Robert and Robin J Ryder. 'Approximate Bayesian computational methods'. In: *Statistics and Computing* (2012).

[133] Paul Marjoram, John Molitor, Vincent Plagnol and Simon Tavaré. 'Markov chain Monte Carlo without likelihoods'. In: *Proceedings of the National Academy of Sciences* (2003).

[134] George Marsaglia. 'Random numbers fall mainly in the planes'. In: *Proceedings of the National Academy of Sciences* 61.1 (1968), pp. 25–28.

[135] George Marsaglia, Wai Wan Tsang et al. 'The Ziggurat Method for Generating Random Variables'. In: *Journal of Statistical Software* 5.i08 (2000).

[136] Phillip James Marshall, Michael Paul Hobson and Anže Slosar. 'Bayesian joint analysis of cluster weak lensing and Sunyaev–Zel'dovich effect data'. In: *Monthly Notices of the Royal Astronomical Society* 346.2 (2003), pp. 489–500.

[137] Richard Massey, Thomas Kitching and Johan Richard. 'The dark matter of gravitational lensing'. In: *Reports on Progress in Physics* 73.8 (2010), p. 086901.

[138] Makoto Matsumoto and Takuji Nishimura. 'Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator'. In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[139] Edward Meeds, Robert Leenders and Max Welling. 'Hamiltonian ABC'. In: *Proceedings of 31st Conference of Uncertainty in Artificial Intelligence*. 2015.

[140] Ted Meeds and Max Welling. 'Optimization Monte Carlo: Efficient and Embarrassingly Parallel Likelihood-Free Inference'. In: *Advances in Neural Information Processing Systems*. 2015.

[141] Kerrie L Mengersen and Richard L Tweedie. 'Rates of convergence of the Hastings and Metropolis algorithms'. In: *The annals of Statistics* 24.1 (1996), pp. 101–121.

[142] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller and Edward Teller. 'Equation of state calculations by fast computing machines'. In: *The Journal of Chemical Physics* (1953).

[143] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 1993.

[144] Thomas P Minka. 'Expectation propagation for approximate Bayesian inference'. In: *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence* (2001).

[145] Tom Minka and John Winn. 'Gates: a graphical notation for mixture models'. In: *Advances in Neural Information Processing Systems*. 2009, pp. 1073–1080.

[146] Tom Minka, John Winn, John Guiver, David Knowles, John Bronksill, Sam Webster and Yordan Zakyov. *Infer.NET*. 2014.

[147] Antonietta Mira and Charles J Geyer. 'On non-reversible Markov chains'. In: *Monte Carlo Methods, Fields Institute/AMS* (2000), pp. 95–110.

[148] Shakir Mohamed and Balaji Lakshminarayanan. 'Learning in Implicit Generative Models'. In: *Proceedings of the International Conference on Learning Representations*. 2017.

[149] Jesper Møller, Anthony N Pettitt, R Reeves and Kasper K Berthelsen. 'An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants'. In: *Biometrika* 93.2 (2006), pp. 451–458.

[150] J. J. Moré, B. S. Garbow and K. E. Hillstrom. *User Guide for MINPACK-1*. ANL-80-74, Argonne National Laboratory. 1980.

[151] Alexander Moreno, Tameem Adel, Edward Meeds, James M Rehg and Max Welling. 'Automatic Variational ABC'. In: *arXiv preprint arXiv:1606.08549* (2016).

[152] Iain Murray. 'Advances in Markov chain Monte Carlo methods'. PhD thesis. University College London, University of London, 2007.

[153] Iain Murray. *A Bayesian approach to Observing Dark Worlds*. http://homepages.inf.ed.ac.uk/imurray2/pub/. 2012.

[154] Iain Murray and Ryan P Adams. 'Slice sampling covariance hyperparameters of latent Gaussian models'. In: *Advances in Neural Information Processing Systems*. 2010.

[155] Iain Murray, Ryan Prescott Adams and David J.C. MacKay. 'Elliptical slice sampling'. In: JMLR: W&CP 9 (2010), pp. 541–548.

[156] Iain Murray, Zoubin Ghahramani and David J. C. MacKay. 'MCMC for doubly-intractable distributions'. In: *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press, 2006, pp. 359–366.

[157] Iain Murray and Matthew Graham. 'Pseudo-marginal slice sampling'. In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. 2016, pp. 911–919.

[158] Peter Neal and Clement Lee. 'Optimal scaling of the independence sampler: Theory and Practice'. In: *arXiv preprint arXiv:1511.04334* (2015).

[159] Radford M Neal. *Markov chain Monte Carlo methods based on 'slicing' the density function*. Tech. rep. 9722. Department of Statistics, University of Toronto, 1997.

[160] Radford M Neal. 'Slice sampling'. In: *Annals of statistics* (2003).

[161] Radford M Neal. 'Improving asymptotic variance of MCMC estimators: Non-reversible chains are better'. In: *arXiv preprint math/0407281* (2004).

[162] Radford M Neal. 'MCMC using Hamiltonian dynamics'. In: *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011. Chap. 5, pp. 113–162.

[163] John von Neumann. 'Various techniques used in connection with random digits'. In: *National Bureau of Standards applied mathematics series* 3 (1951), pp. 36–38.

[164] Robert Nishihara, Iain Murray and Ryan P Adams. 'Parallel MCMC with generalized elliptical slice sampling.' In: *Journal of Machine Learning Research* 15.1 (2014), pp. 2087–2112.

[165] John F Nolan. 'Analytical differentiation on a digital computer'. PhD thesis. Massachusetts Institute of Technology, 1953.

[166] Sheehan Olver and Alex Townsend. 'Fast inverse transform sampling in one and two dimensions'. In: *arXiv preprint arXiv:1307.1223* (2013).

[167] Art B. Owen. 'Importance sampling'. In: *Monte Carlo theory, methods and examples*. 2013. URL: http://statweb.stanford.edu/~owen/mc/Ch-var-is.pdf.

[168] John W Paisley, David M Blei and Michael I Jordan. 'Variational Bayesian Inference with Stochastic Search'. In: *Proceedings of the 29th International Conference on Machine Learning*. 2012, pp. 1367–1374.

[169] Omiros Papaspiliopoulos, Gareth O Roberts and Martin Sköld. 'Non-centered parameterisations for hierarchical models and data augmentation'. In: *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*. Vol. 307. Oxford University Press, USA. 2003.

[170] Omiros Papaspiliopoulos, Gareth O Roberts and Martin Sköld. 'A general framework for the parametrization of hierarchical models'. In: *Statistical Science* (2007), pp. 59–73.

[171] G. Parisi. *Statistical Field Theory*. Advanced book classics. Avalon Publishing, 1998. ISBN: 9780738200514. URL: https://books.google.co.uk/books?id=y0-8xQOw6FcC.

[172] Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

[173] Peter H Peskun. 'Optimum Monte-Carlo sampling using Markov chains'. In: *Biometrika* 60.3 (1973), pp. 607–612.

[174] Carsten Peterson and James R Anderson. 'A mean field theory learning algorithm for neural networks'. In: *Complex systems* (1987).

[175] Michael Pitt, Ralph Silva, Paolo Giordani and Robert Kohn. 'Auxiliary particle filtering within adaptive Metropolis-Hastings sampling'. In: *arXiv preprint arXiv:1006.1914* (2010).

[176] Martyn Plummer et al. 'JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling'. In: *Proceedings of the 3rd international workshop on distributed statistical computing*. Vol. 124. Vienna. 2003, p. 125.

[177] Martyn Plummer, Nicky Best, Kate Cowles and Karen Vines. 'CODA: Convergence Diagnosis and Output Analysis for MCMC'. In: *R News* 6.1 (2006), pp. 7–11. URL: http://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

[178] M. J. D. Powell. 'Numerical Methods for Nonlinear Algebraic Equations'. In: Gordon and Breach, 1970. Chap. A Hybrid Method for Nonlinear Equations.

[179] Dennis Prangle. 'Summary statistics in approximate Bayesian computation'. In: *arXiv preprint arXiv:1512.05633* (2015).

[180] Robert Price. 'A useful theorem for nonlinear devices having Gaussian inputs'. In: *IRE Transactions on Information Theory* 4.2 (1958), pp. 69–72.

[181] Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun and Marcus W Feldman. 'Population growth of human Y chromosomes: a study of Y chromosome microsatellites.' In: *Molecular biology and evolution* 16.12 (1999), pp. 1791–1798.

[182] James Gary Propp and David Bruce Wilson. 'Exact sampling with coupled Markov chains and applications to statistical mechanics'. In: *Random structures and Algorithms* 9.1-2 (1996), pp. 223–252.

[183] Adrian E Raftery and Steven Lewis. *How many iterations in the Gibbs sampler?* Tech. rep. Washington University, Seattle, Department of Statistics, 1991.

[184] Rajesh Ranganath, Sean Gerrish and David Blei. 'Black Box Variational Inference'. In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. 2014.

[185] C Radhakrishna Rao. 'Information and the accuracy attainable in the estimation of statistical parameters'. In: *Breakthroughs in statistics*. Springer, 1992, pp. 235–247.

[186] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited, 2006. ISBN: 9780262182539.

[187] Oliver Ratmann, Christophe Andrieu, Carsten Wiuf and Sylvia Richardson. 'Model criticism based on likelihood-free inference, with an application to protein network evolution'. In: *Proceedings of the National Academy of Sciences* (2009).

[188] Alfréd Rényi. 'On measures of entropy and information'. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. 1961, pp. 547–561.

[189] Danilo Jimenez Rezende, Shakir Mohamed and Daan Wierstra. 'Stochastic Backpropagation and Approximate Inference in Deep Generative Models'. In: *Proceedings of The 31st International Conference on Machine Learning*. 2014, pp. 1278–1286.

[190] Herbert Robbins and Sutton Monro. 'A stochastic approximation method'. In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[191] Christian P Robert, Kerrie Mengersen and Carla Chen. 'Model choice versus model criticism'. In: *Proceedings of the National Academy of Sciences of the United States of America* (2010).

[192] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[193] Gareth O Roberts and Jeffrey S Rosenthal. 'Optimal scaling for various Metropolis-Hastings algorithms'. In: *Statistical science* 16.4 (2001), pp. 351–367.

[194] Gareth O Roberts and Jeffrey S Rosenthal. 'General state space Markov chains and MCMC algorithms'. In: *Probability Surveys* 1 (2004), pp. 20–71.

[195] Gareth O Roberts and Adrian FM Smith. 'Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms'. In: *Stochastic processes and their applications* 49.2 (1994), pp. 207–216.

[196] Gareth O Roberts and Richard L Tweedie. 'Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms'. In: *Biometrika* (1996), pp. 95–110.

[197] David Rohde and Matt P Wand. 'Semiparametric mean field variational Bayes: General principles and numerical issues'. In: *Journal of Machine Learning Research* 17.172 (2016), pp. 1–47.

[198] Jeffrey S Rosenthal et al. 'Optimal proposal distributions and adaptive MCMC'. In: *Handbook of Markov Chain Monte Carlo* (2011), pp. 93–112.

[199] Donald B Rubin et al. 'Bayesianly justifiable and relevant frequency calculations for the applied statistician'. In: *The Annals of Statistics* 12.4 (1984), pp. 1151–1172.

[200] Jean-Paul Ryckaert, Giovanni Ciccotti and Herman JC Berendsen. 'Numerical integration of the Cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes'. In: *Journal of Computational Physics* (1977).

[201] Tim Salimans. *Observing Dark Worlds*. http://timsalimans.com/observing-dark-worlds/. 2012.

[202] Tim Salimans, David A Knowles et al. 'Fixed-form variational posterior approximation through stochastic linear regression'. In: *Bayesian Analysis* 8.4 (2013), pp. 837–882.

[203] John Salvatier, Thomas V Wiecki and Christopher Fonnesbeck. 'Probabilistic programming in Python using PyMC3'. In: *PeerJ Computer Science* (2016).

[204] Masa-Aki Sato. 'Online model selection based on the variational Bayes'. In: *Neural Computation* 13.7 (2001), pp. 1649–1681.

[205] Lawrence K Saul, Tommi Jaakkola and Michael I Jordan. 'Mean field theory for sigmoid belief networks'. In: *Journal of Artificial Intelligence Research* (1996).

[206] Lawrence K Saul and Michael I Jordan. 'Exploiting tractable substructures in intractable networks'. In: *Advances in Neural Information Processing Systems*. 1996, pp. 486–492.

[207] Chris Sherlock, Alexandre Thiery and Anthony Lee. 'Pseudo-marginal Metropolis–Hastings using averages of unbiased estimators'. In: *arXiv preprint arXiv:1610.09788* (2016).

[208] Chris Sherlock, Alexandre H Thiery, Gareth O Roberts and Jeffrey S Rosenthal. 'On the efficiency of pseudo-marginal random walk Metropolis algorithms'. In: *The Annals of Statistics* 43.1 (2015), pp. 238–275.

[209] Scott A Sisson and Yanan Fan. 'Likelihood-free MCMC'. In: *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011. Chap. 12, pp. 313–333.

[210] Scott A Sisson, Yanan Fan and Mark M Tanaka. 'Sequential Monte Carlo without likelihoods'. In: *Proceedings of the National Academy of Sciences* 104.6 (2007), pp. 1760–1765.

[211] Bert Speelpenning. 'Compiling Fast Partial Derivatives of Functions Given by Algorithms'. PhD thesis. University of Illinois at Urbana-Champaign, 1980.

[212] Amos J Storkey. 'Dynamic trees: A structured variational method giving efficient propagation rules'. In: *Proceedings of the Sixteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc. 2000, pp. 566–573.

[213] R. L. Stratonovich. 'Conditional Markov Processes'. In: *Theory of Probability & Its Applications* 5.2 (1960), pp. 156–178. DOI: 10.1137/1105015. URL: http://dx.doi.org/10.1137/1105015.

[214] Yi Sun, Jürgen Schmidhuber and Faustino J Gomez. 'Improving the asymptotic performance of Markov chain Monte-Carlo by inserting vortices'. In: *Advances in Neural Information Processing Systems*. 2010, pp. 2235–2243.

[215] Hidemaro Suwa and Synge Todo. 'Markov chain Monte Carlo method without detailed balance'. In: *Physical review letters* 105.12 (2010), p. 120603.

[216] Martin A Tanner and Wing Hung Wong. 'The calculation of posterior distributions by data augmentation'. In: *Journal of the American statistical Association* 82.398 (1987), pp. 528–540.

[217] Simon Tavaré, David J Balding, Robert C Griffiths and Peter Donnelly. 'Inferring coalescence times from DNA sequence data'. In: *Genetics* 145.2 (1997), pp. 505–518.

[218] Alexander Terenin and David Draper. 'Cox's Theorem and the Jaynesian Interpretation of Probability'. arXiv preprint. 2015. URL: https://arxiv.org/abs/1507.06597v2.

[219] Theano development team. 'Theano: A Python framework for fast computation of mathematical expressions'. In: *arXiv e-prints* abs/1605.02688 (2016). URL: http://arxiv.org/abs/1605.02688.

[220] Madeleine B Thompson. 'A comparison of methods for computing autocorrelation time'. In: *arXiv preprint arXiv:1011.0175* (2010).

[221] Luke Tierney. 'Markov chains for exploring posterior distributions'. In: *The Annals of Statistics* (1994), pp. 1701–1728.

[222] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen and Michael P.H. Stumpf. 'Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems'. In: *Journal of the Royal Society Interface* 6.31 (2009), pp. 187–202.

[223] Dustin Tran, Rajesh Ranganath and David M Blei. 'Deep and Hierarchical Implicit Models'. In: *arXiv preprint arXiv:1702.08896* (2017).

[224] Minh-Ngoc Tran, David J Nott and Robert Kohn. 'Variational Bayes with intractable likelihood'. In: *Journal of Computational and Graphical Statistics* (2017).

[225] John W. Tukey. *Practical Relationship Between the Common Transformations of Percentages or Fractions and of Amounts*. Technical Report 36. Statistical Research Group,Princeton, 1960.

[226] Konstantin S Turitsyn, Michael Chertkov and Marija Vucelja. 'Irreversible Monte Carlo algorithms for efficient sampling'. In: *Physica D: Nonlinear Phenomena* 240.4 (2011), pp. 410–414.

[227] Stanislaw Ulam and Nicholas Metropolis. 'The Monte Carlo method'. In: *Journal of the American Statistical Association* 44.247 (1949), pp. 335–341.

[228] David A Van Dyk and Xiao-Li Meng. 'The art of data augmentation'. In: *Journal of Computational and Graphical Statistics* 10.1 (2001), pp. 1–50.

[229] Matthew P Wand, John T Ormerod, Simone A Padoan, Rudolf Fuhrwirth et al. 'Mean field variational Bayes for elaborate distributions'. In: *Bayesian Analysis* 6.4 (2011), pp. 847–900.

[230] Chong Wang and David M Blei. 'Variational inference in nonconjugate models'. In: *Journal of Machine Learning Research* 14.Apr (2013), pp. 1005–1031.

[231] Gunter Weiss and Arndt von Haeseler. 'Inference of population history using a likelihood approach'. In: *Genetics* 149.3 (1998), pp. 1539–1546.

[232] Max Welling and Yee W Teh. 'Bayesian learning via stochastic gradient Langevin dynamics'. In: *Proceedings of the 28th International Conference on Machine Learning*. 2011.

[233] Robert Edwin Wengert. 'A simple automatic derivative evaluation program'. In: *Communications of the ACM* 7.8 (1964), pp. 463–464.

[234] Richard David Wilkinson. 'Approximate Bayesian computation (ABC) gives exact results under the assumption of model error'. In: *Statistical applications in genetics and molecular biology* (2013).

[235] John Winn and Christopher M Bishop. 'Variational message passing'. In: *Journal of Machine Learning Research* 6.Apr (2005), pp. 661–694.

[236] Simon N Wood. 'Statistical inference for noisy nonlinear ecological dynamic systems'. In: *Nature* 466.7310 (2010), pp. 1102–1104.

[237] Reza Zadeh. *Twitter status*. Dec. 2016. URL: https://twitter.com/Reza_Zadeh/status/811130294291963904.

[238] Emilio Zappa, Miranda Holmes-Cerfon and Jonathan Goodman. 'Monte Carlo on manifolds: sampling densities and integrating functions'. In: *arXiv preprint arXiv:1702.08446* (2017).