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

FRONT MATTER

Abstract 1
Acknowledgements 3
Declaration 4
Contents 5
List of Figures 8
List of Tables 10
List of Algorithms 10
List of Abbreviations 11

1 INTRODUCTION 13
1.1 Probability theory 14
1.1.1 Random variables 14
1.1.2 Joint and conditional probability 15
1.1.3 Probability densities 16
1.1.4 Transforms of random variables 18
1.1.5 Expectations 21
1.1.6 Conditional expectations and densities 22
1.2 Graphical models 23
1.3 Inference 27
1.3.1 Hierarchical models 30
1.3.2 Simulator models 32
1.3.3 Undirected models 34
1.3.4 Model comparison 37
1.4 Summary 39

2 APPROXIMATE INFERENCE 41
2.1 Monte Carlo methods 42
2.1.1 Monte Carlo integration 42
2.1.2 Pseudo-random number generation 44
2.1.3 Transform sampling 45
2.1.4 Rejection sampling 46
2.1.5 Importance sampling 49
2.2 Markov chain Monte Carlo 51
2.2.1 Metropolis–Hastings 58

2.2.2 Gibbs sampling 63
2.3 Auxiliary variable methods 65
2.3.1 Slice sampling 67
2.3.2 Hamiltonian Monte Carlo 76
2.3.3 Simulated tempering 80
2.4 Summary 80

3 PSEUDO-MARGINAL METHODS 85
3.1 Problem definition 87
3.1.1 Example: hierarchical latent variable models 88
3.2 Pseudo-marginal Metropolis–Hastings 90
3.3 Reparametrising the density estimator 93
3.4 Auxiliary pseudo-marginal methods 96
3.5 Pseudo-marginal slice sampling 99
3.6 Numerical experiments 103
3.6.1 Gaussian latent variable model 104
3.6.2 Gaussian process probit regression 119
3.7 Discussion 128

4 IMPLICIT GENERATIVE MODELS 133
4.1 Differentiable generator networks 135
4.2 Generative models as transformations 138
4.3 Model parameterisation 141
4.4 Directed and undirected models 144
4.5 Approximate Bayesian Computation 145
4.6 ABC MCMC methods 153
4.7 Inference in the input space 156
4.8 Constrained Hamiltonian Monte Carlo 162
4.9 Method 165
4.10 Related work 170
4.11 Experiments 172
4.11.1 Lotka–Volterra parameter inference 172
4.11.2 Human pose and camera model inference 176
4.11.3 MNIST in-painting 181
4.12 Discussion 182

5 CONTINUOUS TEMPERING 185
5.1 Hamiltonian Monte Carlo 186
5.2 Thermodynamic methods 188
5.3 Continuous tempering 191

5.4 Proposed method 194
5.5 Choosing a base density 196
5.6 Experiments 199
5.6.1 Boltzmann machine relaxations 199
5.6.2 Hierarchical regression model 201
5.6.3 Generative image models 202
5.7 Discussion 204

A DISTRIBUTION DEFINITIONS 207

B COMPUTATION GRAPHS 211
B.1 Automatic differentiation 213

C OPTIMISATION-BASED APPROXIMATE INFERENCE 217
C.1 Laplace's method 218
C.2 Variational methods 220

BIBLIOGRAPHY 229

LIST OF FIGURES

Figure 1.1 Factor graph examples. 24
Figure 1.2 Hierarchical linear regression model factor graph. 25
Figure 1.3 Factor graph of iid observed variables. 27
Figure 1.4 Hierarchical latent variable model factor graph. 30
Figure 1.5 Simulator model example. 32
Figure 1.6 Example of implicit probabilistic model. 33
Figure 1.7 Boltzmann machine factor graph. 34
Figure 2.1 Example linear congruential generator sequence. 44
Figure 2.2 Visualisation of Box–Muller transform. 45
Figure 2.3 Visualisation of rejection sampling. 47
Figure 2.4 Factor graph of rejection sampling process. 48
Figure 2.5 Visualisation of importance sampling. 50
Figure 2.6 Markov chain factor graph. 52
Figure 2.7 Visualisation of Metropolis–Hastings algorithm. 58
Figure 2.8 Concentration of measure in high dimensions. 62
Figure 2.9 Gibbs sampling schematic. 63
Figure 2.10 Schematic of linear slice sampling. 67
Figure 2.11 Linear slice sampler comparison. 71
Figure 3.1 Hierarchical model factor graph. 88
Figure 3.2 Reflective linear slice sampling. 102
Figure 3.3 PM MH Gaussian model results. 106
Figure 3.4 PM MH Gaussian model traces. 108
Figure 3.5 APM MI+MH Gaussian model results. 111
Figure 3.6 APM MI+MH Gaussian model traces. 113
Figure 3.7 APM SS+MH Gaussian model results. 114
Figure 3.8 APM Gaussian model auxiliary variable traces. 116
Figure 3.9 APM SS Gaussian model results. 117
Figure 3.10 Gaussian process probit regression factor graph. 119
Figure 3.11 Gaussian process probit regression results. 125
Figure 3.12 Gaussian process probit regression traces. 127
Figure 4.1 Differentiable generator network factor graphs. 136
Figure 4.2 Unbounded unit variance densities. 142
Figure 4.3 Undirected and directed generative models. 144
Figure 4.4 Oscillatory Hamiltonian trajectory example. 164

Figure 4.5 Structured directed generative models. 168
Figure 4.6 Inference in Lotka–Volterra simulator model. 173
Figure 4.7 Human pose generative model factor graph. 176
Figure 4.8 Inference in human pose model. 178
Figure 4.9 MNIST in-painting samples. 181
Figure 5.1 Boltzmann machine relaxation results. 199
Figure 5.2 Factor graph of Radon hierarchical regression model. 201
Figure 5.3 Log marginal likelihood estimates. 202
Figure 5.4 Radon hierachical regression model marginal likelihood estimates 202
Figure 5.5 IWAE marginal likelihood estimates. 206
Figure B.1 Example computation graph. 212
Figure B.2 Reverse-mode automatic differentiation. 214
Figure C.1 Univariate example of Laplace's method. 219
Figure C.2 Variational objective comparison. 224

LIST OF TABLES

Table 4.1 Standardisation reparametrisations. 142
Table A.1 Standard discrete density definitions. 208
Table A.2 Standard unbounded density definitions. 208
Table A.3 Standard bounded density definitions. 209

LIST OF ALGORITHMS

Algorithm 1 Rejection sampling. 47
Algorithm 2 Metropolis–Hastings transition. 59
Algorithm 3 Sequential scan Gibbs transition. 63
Algorithm 4 Linear slice sampling transition. 68
Algorithm 5 Elliptical slice sampling transition. 74
Algorithm 6 Hamiltonian Monte Carlo transition. 76
Algorithm 7 Pseudo-marginal Metropolis–Hastings. 91
Algorithm 8 Auxiliary pseudo-marginal framework. 96
Algorithm 9 Auxiliary pseudo-marginal MI + MH. 96
Algorithm 10 ABC Pseudo–Marginal Metropolis–Hastings. 154
Algorithm 11 Constrained Hamiltonian Monte Carlo 166
Algorithm 12 Lotka–Volterra model generator functions 174
Algorithm 13 Human pose model generator functions 177
Algorithm 14 Reverse-mode automatic differentiation. 215

LIST OF ABBREVIATIONS

MCMC Markov chain Monte Carlo

MH Metropolis–Hastings

MI Metropolis independence

SS slice sampling

KL Kullback–Leibler

SDE stochastic differential equation

ODE ordinary differential equation

ABC approximate Bayesian computation

PM pseudo-marginal

APM auxiliary pseudo-marginal

HMC Hamiltonian Monte Carlo

SMC sequential Monte Carlo

BBVI black box variational inference

ADVI automatic differentiation variational inference

EP expectation propagation

AD automatic differentiation

AIS annealed importance sampling

VAE variational autoencoder

IWAE importance weighted autoencoder

GAN generative-adversarial network

CPU central processing unit

GPU graphics processing unit

RMSE root mean squared error

ELBO evidence lower bound

PRNG pseudo-random number generator

CDF cumulative distribution function

ESS effective sample size

AIS annealed importance sampling

BDMC bidirectional Monte Carlo

ST simulated tempering

PT parallel tempering

CT continuous tempering

NUTS no U-turn sampler

PSRF potential scale reduction factor

iid independently and identically distributed

wrt with respect to

iff if and only if

1 INTRODUCTION

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. While 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 uncertain. To make inferences under conditions of uncertainty, we must instead turn to probability theory, which both offers a consistent framework for quantifying our beliefs and making inferences given these beliefs.

The key computational task in inference is computing integrals with respect to probability distributions on the variables in a proposed model. Typically these integrals will not have analytic solutions and the large number of variables being integrated over mean numerical quadrature methods are impractically costly. In these cases we must resort to approximate methods which tradeoff an introduction of error for an increase in computational tractability. *Markov chain Monte Carlo* (MCMC) methods are a very generally applicable class of approximate inference techniques which estimate the integrals of interest by computing averages over the states of a Markov chain.

The topic of this thesis is the development of new MCMC methods. In particular we introduce several novel methods which exploit reparameterisations and augmentations of the state of a Markov chain to improve upon the computational efficiency, ease of use or degree of approximation error of existing approaches.

In this chapter we discuss the basic concepts of probabilistic modelling which underpin the inference methods discussed in later chapters. In particular we review the terminology and basic concepts of the measure-theoretic description of probability as some of the later results in the thesis are most clearly described within this framework. We also introduce graphical models as a compact way of visualising structure in probabilistic models. Finally we conclude with a discussion of the specific inference problems that the methods presented in the rest of this thesis are intended to help tackle.

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 [120] 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 $\mathsf{P}(A \cap B) = \mathsf{P}(A)\mathsf{P}(B)$.

The key advantage of the measure-theoretic approach to probability is that it provides a consistent definition of the probability of an event in any space we can define a measure on. This allows a unified treatment of the common cases of probability distributions of discrete and continuous random variables but also makes it possible to consider distributions on more general spaces. In Chapter 4 we will consider problems which involve distributions on implicitly-defined manifolds where this generality will be key to understanding the proposed methods.

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 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 refer to a *real random vector* and use the notation $\mathbf{x} : S \to X$. A final specific case is when X is countable and $\mathcal{F}$ is the power set $\mathscr{P}(X)$ in which case we refer to x as a *discrete random variable*. As we are most often concerned with real-valued random variables and vectors in this thesis, when it is unambiguous to do so we 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 typically 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 particularly 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 [50, 51] and de Finetti [74], 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 Section 1.2.

1.1.3 Probability densities

So far we have not specified 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

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

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

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 it.

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.

Real random variables will typically have a distribution P_x defined by a probability density $\mathsf{p}_\mathsf{x} : \mathbb{R} \to [0, \infty)$ with respect to the *Lebesgue measure*, λ, on $\mathbb{R}$,

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

The Lebesgue measure assigns a measure to subsets of a Euclidean space, and for $\mathbb{R}$, $\mathbb{R}^2$ and $\mathbb{R}^3$ formalises the intuitive concepts of length, area and volume of subsets respectively.

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 we have that

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

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 probability distribution of a discrete random variable can be defined via probability density $\mathsf{p}_\mathsf{x} : X \to [0, 1]$ with respect to the *counting measure* $\#$,

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

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

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, this avoiding difficulties with terminology and notation 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

$$\begin{aligned} P_{x,y}(A, B) &= \int_A \int_B p_{x,y}(x, y)\, \mu_x(dx)\, \mu_y(dy) \\ &= \int_B \int_A p_{x,y}(x, y)\, \mu_y(dy)\, \mu_x(dx) \quad \forall A \in \mathcal{F},\ B \in \mathcal{G}. \end{aligned} \tag{1.14}$$

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 $(S, \mathcal{E}, P)$ 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.

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 make use of in this thesis is defining transformations of 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

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

i.e. the probability of the event $y \in A$ 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

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

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 $\mathsf{p}_\mathsf{x} = \frac{\mathrm{d}\mathsf{P}_\mathsf{x}}{\mathrm{d}\#}$ we have that[1]

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

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

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

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

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

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

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.

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 infinitesimal 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 infinitesimal 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

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

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 to the 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 a more standard 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 $\mathsf{P}_{\mathsf{x}|\mathsf{y}}(\cdot \mid y)$ defines a valid probability measure on $(X, \mathcal{F})$ for P_y-almost all y and using (1.30) we have

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

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

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

If we define the *conditional density* $p_{x|y}$ as

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

then subsituting this definition in to (1.34) we have

$$\int_B P_{x|y}(A \mid y)\, P_y(dy) = \int_B \int_A p_{x|y}(x \mid y)\, \mu_x(dx)\, P_y(dy). \tag{1.36}$$

Therefore $p_{x|y}$ is the density of the regular conditional distribution $P_{x|y}$. We also have that if $p_{x,y}$ and p_y can be defined that

$$\mathbb{E}[x \mid y](y) = \int_X x\, p_{x|y}(x \mid y)\, \mu_x(dx) \quad \forall y \in Y : p_y(y) > 0. \tag{1.37}$$

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

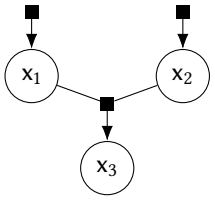
1.2 GRAPHICAL MODELS

Graphical models = statistics × graph theory × computer science
—Zoubin Ghahramani

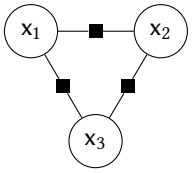
When working with probabilistic models involving large numbers of random variables, it will often be the case that not all the variables are jointly dependent on each other but that instead there are more local relationships between them. Graphical models, which use graphs to describe the dependencies between random variables, are a useful framework for visualising the structure of complex models.

Several graphical formalisms for representing dependency structure in probabilistic models have been proposed, with *directed graphical models* [175] (also known as *Bayesian networks*) and *undirected graphical models* [118] (also known as *Markov random fields*) both common in practice and each offering a more natural representation for certain model classes. In this thesis we will instead use *factor graphs* [75, 76] which combine the representational abilities of both directed and undirected graphical models while also offering a richer framework for representing fine-grained information about model structure.

Factor graphs are bipartite graphs consisting of two node types: *variable nodes*, displayed as labelled circles and representing individual (potentially non-scalar) random variables in a probabilistic model, and *factor nodes*, displayed as filled squares and representing individual



(a) Example directed factor graph.



(b) Example undirected factor graph.

Figure 1.1.: Examples of simple directed and undirected factor graphs. Square black nodes correspond to individual factors depending on the connected variables represented by circular nodes in the joint density.

factors in the joint density across the random variables in the model. 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.

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, 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 undirected edges then the factor instead corresponds to a marginal density. Graphs with directed factors must not contain any directed cycles, i.e. connected loops 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.

Figure 1.1a shows a simple example of fully directed factor graph for three random variables. The graph implies that the joint density for the model can be factorised as

$$\mathrm{p}_{\mathrm{x}_1,\mathrm{x}_2,\mathrm{x}_3}(x_1, x_2, x_3) = \mathrm{p}_{\mathrm{x}_3|\mathrm{x}_1,\mathrm{x}_2}(x_3 \mid x_1, x_2)\, \mathrm{p}_{\mathrm{x}_1}(x_1)\, \mathrm{p}_{\mathrm{x}_2}(x_2). \tag{1.38}$$

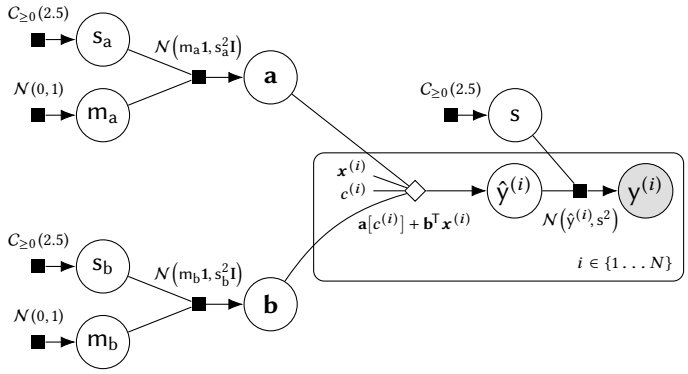


Figure 1.2.: Hierarchical linear regression model factor graph showing examples of extended factor graph notation.

Figure 1.1b shows a fully undirected factor graph on three random variables. If $\psi_{i,j}$ denotes the unnormalised density factor on the pair $(\mathsf{x}_i, \mathsf{x}_j)$ then the graph implies the joint density can be factorised as

$$\mathrm{p}_{\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_3}(x_1, x_2, x_3) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{1,3}(x_1, x_3)\psi_{2,3}(x_2, x_3) \tag{1.39}$$

with Z a normalising constant such that the density integrates to one.

Figure 1.2 shows examples of some additional useful factor graph notation we will use in this thesis. We use as an example a factor graph corresponding to a hierarchical linear regression model which will be discussed in Chapter 5.

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 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 1.2. A deterministic factor node can informally[2] be considered equivalent to a directed factor node with a degenerate Dirac delta conditional density on the

2 A Dirac delta is not strictly a density as it is not the Radon–Nikodyn derivative of an absolutely continuous measure, however informally we treat is as the density of a singular Dirac measure with respect to the Lebesgue measure $\int f(x)\,\delta(\mathrm{d}x) \simeq \int f(x)\delta(x)\,\mathrm{d}x$.

output variable which concentrates all the probability mass at the output of the function applied to the inputs variables.

The deterministic node notation allows generative models consisting of complex compositions of deterministic functions and probabilistic sampling operations to be represented in a unified framework. Subgraphs of a directed factor graph consisting entirely of determinstic nodes can be viewed as *computation graphs*, a graphical formalism typically used in numerical computing frameworks to support efficient *automatic differentiation* algorithms. We exploit this idea in later in the thesis to allow propagation of derivatives through complex probabilistic models and make extensive use of automatic differentiation implementations in frameworks such as *Theano* [218] in numerical experiments. In Appendix B we provide a short review of the basic concepts of computation graphs and automatic differentation and a discussion of their links to directed factor graphs.

In some cases constant values used in a model will be included in a factor graph as plain nodes indicated only by a label. The $\boldsymbol{x}^{(i)}$ and $c^{(i)}$ nodes in Figure 1.2 are an example of this notation.

A commonly used convention in factor graphs (and other graphical models) is *plate notation* [39], with an example of a plate shown by the rounded rectangle bounding some of the nodes in Figure 1.2. 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 1.2). 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.

Each of the factors in Figure 1.2 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 Appendix A. 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 1.2 is the use of a shaded variable node (corresponding to $\mathrm{y}^{(i)}$) to indicate a random variable corresponding to an observed quantity in the model.

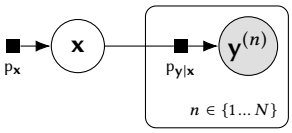


Figure 1.3.: Factor graph of N observations $\mathbf{y}^{(n)}$ independently and identically distributed according to a distribution with parameters $\mathbf{x}$.

1.3 INFERENCE

Having now introduced the tools and notation we use to define probabilistic models, we will now describe the inference problems we consider approximate approaches to solving in this thesis. We begin with a overview of *Bayesian inference*.

You cannot do inference without making assumptions
—David Mackay

The starting point for any inference problem is to define a model specifying proposed relationships between the observed data and unknown 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 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. We consider probabilistic model comparison in a subsequent subsection.

Amongst the simplest, but also most common, modelling assumptions made is that the observed data values are *independently and identically distributed* (iid) according to a parametric probability distribution. If we denote the collection of N observed variables $\{\mathbf{y}^{(n)}\}_{n=1}^{N}$ then we assume that each is independently generated from a distribution $\mathsf{P}_{\mathbf{y}^{(n)}|\mathbf{x}} = \mathsf{P}_{\mathbf{y}|\mathbf{x}}\ \forall n \in \{1 \dots N\}$ with density $\mathsf{p}_{\mathbf{y}|\mathbf{x}} = \frac{\partial \mathsf{P}_{\mathbf{y}|\mathbf{x}}}{\partial \mu_{\mathbf{y}}}$ and governed by a set of unknown parameters $\mathbf{x} \in X$.

Any beliefs we have about the plausible values for the parameters prior to observing data are integrated into the model by choosing an appropriate, typically parametric, marginal distribution $\mathsf{P}_{\mathbf{x}}$, with this distribution, and the corresponding density $\mathsf{p}_{\mathbf{x}} = \frac{\partial \mathsf{P}_{\mathbf{x}}}{\partial \mu_{\mathbf{x}}}$, referred to as the *prior*. The joint distribution on the model variables then factorises as

$$\mathsf{p}_{\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)},\mathbf{x}}(\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)}, \boldsymbol{x}) = \prod_{n=1}^{N} \mathsf{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x})\, \mathsf{p}_{\mathbf{x}}(\boldsymbol{x}) \tag{1.40}$$

with this structure illustrated as a directed factor graph in Figure 1.3. In analogy to the naming of the prior, the conditional distribution on the unknown model parameters after conditioning on observed data values is termed the *posterior* and from the definition of conditional density (1.35) we can express its density as

$$\mathrm{p}_{\mathbf{x}|\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(N)}}(\boldsymbol{x} \mid \boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}) = \frac{\prod_{n=1}^{N} \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})}{\mathrm{p}_{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(N)}}(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)})}. \tag{1.41}$$

Bayesian inference is named after Thomas Bayes, an 18th century Presbyterian minister, who proved a special case of what is now termed Bayes' theorem. Pierre-Simon Laplace later independently derived a more general statement of Bayes' theorem closer to the modern form.

This expression relating the posterior on the unknown parameters to the prior distribution and model of the observations is an example of *Bayes' theorem*. Typically the product of the conditional densities $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ is termed the *likelihood* and considered a function of the value $\boldsymbol{x}$ of the unknown parameters $\mathbf{x}$, with the observed data values $\{\boldsymbol{y}^{(n)}\}_{n=1}^{N}$ fixed. The denominator of the right-hand side (1.41), the marginal density on the observed variables, can be written as a integral marginalising out the parameters from the joint density

$$\mathrm{p}_{\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(N)}}(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}) = \int_X \prod_{n=1}^{N} \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^{(n)} \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mu_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}). \tag{1.42}$$

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

This term is often described as the *marginal likelihood* or the *model evidence*. Generally this integral will not have an analytic solution though there are exceptions to this in a few special cases. For example if the densities $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ and $\mathrm{p}_{\mathbf{x}}$ are both of *exponential family distributions* and form a conjugate pair such that the posterior density is in the same family as the prior density then (1.42) will have a closed-form solution. For models in which the parameters are discrete the integral in (1.42) corresponds to a summation and so is in theory exactly solvable, though if the total number of possible configurations of the parameters is very large this summation can be infeasible to compute in practice. If the parameters are instead real-valued but of a low-dimensionality it may be possible to use numerical quadrature methods [55] to compute the integral in (1.42) to a reasonable accuracy. Quadrature methods involve evaluating the integrand across a grid of points and then computing a weighted sum of these values. For a fixed grid resolution however the cost of quadrature scales exponentially with the dimension of the space being integrated over - if G points are used per dimension, for a D dimensional parameter space evaluating (1.42) would require summing the joint density over G^D parameter values.

For real-valued parameter spaces of a more than ~ 10 dimensions[3] evaluating the model evidence term (1.42) is therefore typically computationally intractable. We can therefore often only evalulate the posterior density (1.41) up to an unknown constant. The posterior density itself is usually not of direct interest as it is only a proxy for describing the posterior distribution and is dependent on the particular model parameterisation chosen. However most quantities of interest from an inference perspective involve integrating functions against the posterior distribution and as with the model evidence these integrals will typically be intractable to compute exactly.

For example under an iid assumption the density of the *predictive distribution* of a new data point $\mathbf{y}^*$ given the previously observed data is formed by integrating $\mathrm{p}_{\mathbf{y}|\mathbf{x}}$ against the posterior distribution

$$\begin{aligned}
&\mathrm{p}_{\mathbf{y}^*|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{y}^* \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)}) \\
&\quad = \int_X \mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^* \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{x} \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)})\, \mu_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \qquad (1.43) \\
&\quad = \mathbb{E}\left[\mathrm{p}_{\mathbf{y}|\mathbf{x}}(\boldsymbol{y}^* \mid \mathbf{x}) \mid \mathbf{y}^{(1)} = \boldsymbol{y}^{(1)}, \dots, \mathbf{y}^{(N)} = \boldsymbol{y}^{(N)}\right].
\end{aligned}$$

If we wish to for example minimise the expected prediction error under some loss function this will involve integrating against this predictive distribution and so as a sub-task integrating against the posterior distribution on the model parameters. Similarly evaluating statistics of the unknown parameters under the posterior such as their mean or covariance corresponds to computing conditional expectations. In general any inferential output which takes in to account all of the information available from the posterior distribution will involve integrating against the posterior and so the computation of integrals is the key computational task in inference.

As exact evaluation of the integrals of interest is usually intractable we must instead resort to *approximate inference* methods which tradeoff an introduction of some level of approximation for an increase in computational tractability.

3 The C-based implementation by Steven G. Johnson of an adaptive multi-dimensional quadrature algorithm [20] available at https://github.com/stevengj/cubature recommends using the package for integrals of up to around $D = 7$. Running a provided test cases for the integral of a Gaussian density across a D-dimensional space with a target error tolerance of 10^{-5} took around 2.5 seconds for $D = 5$, 50 seconds for $D = 6$ and 17 minutes for $D = 7$ on one core of a desktop CPU. Extrapolating the ~ 20-fold increase in run time per dimension, for $D = 10$ the run-time would be around 100 days.

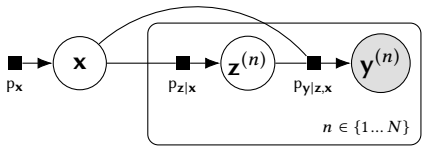


Figure 1.4.: Factor graph of a simple hierarchical latent variable model with N observed variables $\mathbf{y}^{(n)}$ each associated with a local latent variable $\mathbf{z}^{(n)}$, with both observed and latent variables dependent on a set of global latent variables (parameters) $\mathbf{x}$.

The iid assumption is widely made in inference problems and although it will not be always be entirely valid in practice, it will often be a reasonable approximation. For real-valued parameter spaces X and densities $p_{\mathbf{y}|\mathbf{x}}$ and $p_{\mathbf{x}}$ meeting certain regularity conditions, if an iid assumption is valid then the posterior distribution will asympotically tend to a multivariate normal distribution as the number of data points N tends to infinity [106]. For inference in models of large iid datasets where the conditions for asymptotic normality are met, while the dimensionality of the parameter space will often still require the use of approximate inference methods, the close to normal geometry of the posterior distribution will typically mean even relatively simple approximate inference methods can achieve good results.

In this thesis we will primarily be concerned with methods for performing inference in models which do not fit into this mould. In the following subsections we discuss some specific issues that can prove challenging to standard approximate inference approaches and which the methods contributed in this thesis are intended to help address.

1.3.1 Hierarchical models

In the preceding discussion of inference in a model of a iid dataset, it was assumed that the only unknown variables in the model were a set of parameters $\mathbf{x}$, the quantity of which did not depend on the number of data points N. This structure can be overly restrictive with it common that the process being modelled includes unknown quantities associated with each observed variable. Models will therefore often include local (per data point) latent variables in addition to a set of global latent variables (or parameters). This grouping structure in the observed and unobserved variables in a model can extend to multiple levels and such models often are termed *hierarchical* or *multilevel* models.

A simple example of a hierarchical model is shown as a factor graph in Figure 1.4. As in the factor graph in Figure 1.3 we assume there are N observed variables $\{\mathbf{y}^{(n)}\}_{n=1}^{N}$ and a vector of global latent variables $\mathbf{x}$. We further define N local latent variables $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$ paired with each observed variable. In Figure 1.4 we assume the local latent and observed variables are conditionally independent given the global latent variables. More complex structures are also common - for example dynamical state space models for time series data assume dependencies between the latent variables corresponding to adjacent time points.

Although powerful, the introduction of local latent variables in to models can significantly increase the complexity of inference. At a basic level, as the number of unobserved variables is now dependent on the data set size, the total dimensionality of the space which needs to be integrated over when performing inference will typically be much higher than for models with a fixed number of parameters. This means the need for inference methods which scale well with dimensionality is even more essential. The growth of the the number of unobserved variables with the data set size N will typically also mean that we can no longer expect asymptotic normality of the full posterior. Typically the posterior distribution on the local and global latent variables will have a complex geometry, with strong dependencies between the global and local latent variables that can limit the performance of many standard approximate inference approaches [26].

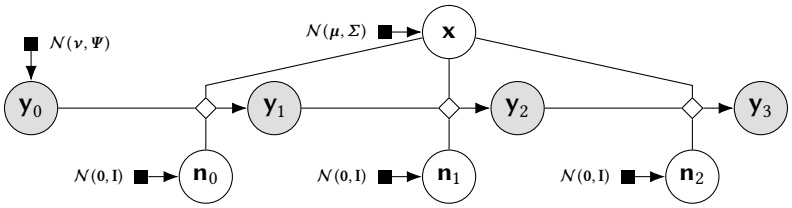
In some cases the posterior distributions of the local latent variables associated with the observed data will not be of direct interest to the downstream task. For example the conditional independence structure in Figure 1.4 means that the predictive distribution on a new unseen datapoint $\mathbf{y}^*$ given the observed data has density

$$\begin{aligned} &\mathrm{p}_{\mathbf{y}^*|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{y}^* \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)}) \\ &\quad = \int_Z \int_X \mathrm{p}_{\mathbf{y}|\mathbf{x},\mathbf{z}}(\boldsymbol{y}^* \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{p}_{\mathbf{z}|\mathbf{x}}(\boldsymbol{z} \mid \boldsymbol{x}) \\ &\qquad\qquad \mathrm{p}_{\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}(\boldsymbol{x} \mid \boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N)})\, \mu_{\mathbf{x}}(\mathrm{d}\boldsymbol{x})\, \mu_{\mathbf{z}}(\mathrm{d}\boldsymbol{z}). \end{aligned} \tag{1.44}$$

Predictions under the model will therefore not depend on the values of the local latent variables $\{\mathbf{z}^{(n)}\}_{n=1}^{N}$, and so ideally we would marginalise out these variables from the full posterior distribution on all unobserved variables $\mathrm{P}_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(N)},\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}$ to obtain the posterior dis-

$\boldsymbol{y}_0 \sim \mathcal{N}(\cdot \mid \boldsymbol{\nu}, \boldsymbol{\Psi})$
$\boldsymbol{x} \sim \mathcal{N}(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$
for $t \in \{1 \dots T\}$ **do**
  $\boldsymbol{n}_{t-1} \sim \mathcal{N}(\cdot \mid \mathbf{0}, \mathbf{I})$
  $\boldsymbol{y}_t \leftarrow \boldsymbol{y}_{t-1} + h\boldsymbol{m}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) - \frac{h}{2}\boldsymbol{s}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) \odot \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{y}}(\boldsymbol{y}_{t-1}, \boldsymbol{x})$
  $\boldsymbol{y}_t \leftarrow \boldsymbol{y}_t + \sqrt{h}\,\boldsymbol{s}(\boldsymbol{y}_{t-1}, \boldsymbol{z}) \odot \boldsymbol{n}_{t-1} + \frac{h}{2}\boldsymbol{s}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) \odot \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{y}}(\boldsymbol{y}_{t-1}, \boldsymbol{x}) \odot \boldsymbol{n}_{t-1}^2$

(a) Pseudo-code for Milstein method integration of SDE model.



(b) Directed factor graph of 3 time steps of SDE simulation.

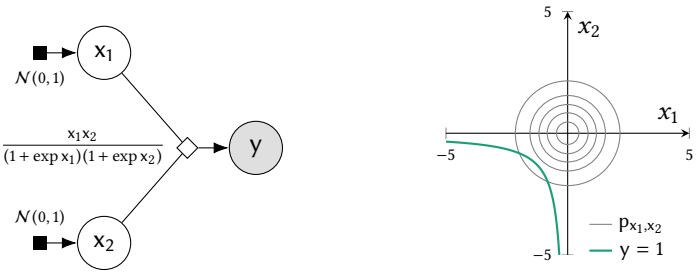
Figure 1.5.: Example of a simulator model corresponding to Milstein method integration of a set of SDEs, $\mathrm{d}\mathbf{y}(t) = \boldsymbol{m}(\mathbf{y}(t), \mathbf{x})\,\mathrm{d}t + \boldsymbol{s}(\mathbf{y}(t), \mathbf{x})\,\mathrm{d}\mathbf{n}(t)$, specified as pseudo-code in (a) and a directed factor graph in (b). The dynamics of model are governed by parameters $\mathbf{x}$. In the pseudo-code the notation $\sim$ followed by a distribution shorthand represents generating a value from the associated distribution.

tribution on just the global latent variables $\mathrm{P}_{\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}$. The distribution $\mathrm{P}_{\mathbf{x}|\mathbf{y}^{(1)},\dots,\mathbf{y}^{(N)}}$ is defined on a much lower dimensional space and will often have a simpler geometry which makes it more amenable to approximate inference methods, however generally the marginalisation over the local latent variables will not be analytically tractable. We can in some cases however approximately marginalise out the local latent variables - we discuss methods based on this idea in Chapter 3.

1.3.2 Simulator models

The probabilistic models considered so far have been defined by explicitly specifying a density over the all the variables in the model, for example via a factor graph. Rather than defining the density on the variables in a model an alternative approach is for a process for generating values for the variables in a model to be specified procedurally in code, with the resulting joint density on the model variables then only implicitly defined. Such models are sometimes termed *simulator* or *implicit* models [62].

A common setting in which such models occur is the simulation of a mechanistic model of a physical process for example described by a set



(a) Directed factor graph for model with two latent variables (x_1, x_2), and an observed variable y.

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

Figure 1.6.: Simple example of an implicit probabilistic model where the observed variable is a non-bijective function of two latent variables.

of *stochastic differential equations* (SDEs). In implementations of such simulator models, the stochasticity in the model will be introduced via draws from a pseudo-random number generator. Given these random inputs, the output of the simulator is then calculated as a series of deterministic operations 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. An example of a simulator model corresponding to approximate integration of a set of SDEs using the Milstein method [148] is shown as both pseudo-code and a directed factor graph in Figure 1.5.

The main complicating factor in performing inference in simulator models is the unavailability of an explicit density function on the model variables which is a prerequisite for most approximate inference methods. Computing a density function on the unobserved variables to be inferred (for example parameters of the dynamics of a SDE model) and simulated observed variables that are conditioned on requires that all other random variables used in the model are marginalised over. In some cases this marginalisation may technically be possible to exactly solve and so a density function possible to compute in theory but the complexity of the model structure means that the density is unavailable in practice.

In many cases however the density function may not be exactly evaluable even in theory. A key difference of simulator models from the probabilistic models considered previously is that the observed variables in

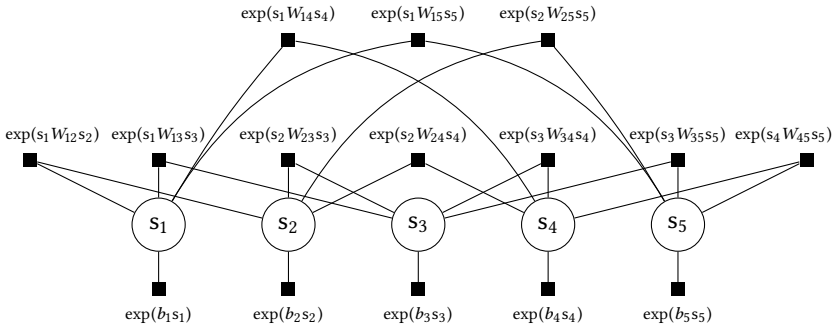


Figure 1.7.: Five unit Boltzmann machine factor graph.

the model are defined via deterministic transformations of other random variables. Using our above intuition that any simulator model can be expressed as a directed factor graph with deterministic factor nodes, this means that the observed variables in the graph correspond to the outputs of deterministic factors rather than the more usual case of the observed variables being connected to probabilistic factors.

An illustration of such a case for a simple three variable model is shown in Figure 1.6. Here the observed variable y is a deterministic function of two latent (unobserved) variables x_1 and x_2. There is no analytic solution in terms of elementary functions for x_1 as a function of y and x_2 or for x_2 as a function of x_1 and y. 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 $\mathsf{p}_{\mathsf{y},\mathsf{x}_1,\mathsf{x}_2}$ is not well defined (the joint distribution $\mathsf{P}_{\mathsf{y},\mathsf{x}_1,\mathsf{x}_2}$ is not absolutely continuous with respect to the Lebesgue measure) which complicates evaluations of conditional expectations such as $\mathbb{E}[f(\mathsf{x}_1, \mathsf{x}_2) \mid \mathsf{y} = 1]$. In particular the set of x_1 and x_2 values corresponding to solutions to $\mathsf{y} = y$ for an particular y (illustrated for $y = 1$ as the green curve in Figure 1.6b) is an implicitly defined manifold (here a one-dimensional curve) in the x_1–x_2 space with zero Lebesgue measure, and the conditional distribution $\mathsf{P}_{\mathsf{x}_1,\mathsf{x}_2|\mathsf{y}}$ has support only on this manifold. We explore methods for performing inference in implicit models in Chapter 4.

1.3.3 Undirected models

When introducing factor graphs we stated that factors can be both directed and undirected. In the preceding discussion we concentrated on directed models, both in the form of model explicitly specified via dir-

ected factor graphs as in the examples in Figure 1.3 and 1.4, and simulator models which as we argued in the previous subsection can be considered as implicitly defining a directed factor graph. A key defining feature of models corresponding to directed factor graphs is that they are natural descriptions of generative processes, with independent sampling from the joint distribution across model variables typically simple to perform via ancestral sampling (in the case of simulator models this being their defining feature).

Undirected models (which we will use here to mean models specified by factor graphs consisting solely of undirected factors) offer a complementary approach for defining a probabilistic model. Each undirected factor node is associated with a non-negative function defining a factor in the joint density across all model variables. Unlike a directed factor, this function does not correspond to a conditional or marginal density. Instead it describes a more general notion of 'compatibility' between the values of sets of variables in the model, defining a series of soft constraints as to which joint configurations are plausible (corresponding to a high value for the factor) or implausible (corresponding to a low model). This makes undirected models a natural representation for models of systems of mutually interacting components without a specific directivity in those interactions. For example they are commonly used in models of images to represent dependencies between pixel values, to model networks of stochastically spiking neurons in the brain and models of magnetic interactions in particle lattices. Unlike directed models, generating samples from the joint distribution on variables in an undirected model is typically a non-trivial task, with no general equivalent to ancestral sampling.

A particularly common form of undirected model is the *Boltzmann machine* [1] also known as a *pairwise binary Markov random field* [118] or in statistical physics settings an *Ising spin model* [115]. A Boltzmann machine consists of a set of binary random variables $\mathbf{s} = [\mathrm{s}_1 \cdots \mathrm{s}_D]^\mathsf{T}$; these are typically chosen to take values in $U = \{0, 1\}^D$ or $S = \{-1, +1\}^D$ - we will favour $S = \{-1, +1\}^D$. The joint distribution across the variables is parameterised by a symmetric weight matrix $\boldsymbol{W} \in \mathbb{R}^{D\times D}$ and a bias vector $\boldsymbol{b} \in \mathbb{R}^D$ and defined as

$$\mathrm{p}_{\mathbf{s}}(\boldsymbol{s}) = \frac{1}{Z}\exp\left(\frac{1}{2}\boldsymbol{s}^\mathsf{T}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^\mathsf{T}\boldsymbol{b}\right), \quad Z = \sum_{\boldsymbol{s}\in S}\exp\left(\frac{1}{2}\boldsymbol{s}^\mathsf{T}\boldsymbol{W}\boldsymbol{s} + \boldsymbol{s}^\mathsf{T}\boldsymbol{b}\right). \tag{1.45}$$

Evaluation of the normalising constant Z involves a summation over 2^D states and so for large D quickly become intractable to compute exactly. Evaluation of expectations with respect to the Boltzmann machine distribution also involves an exhaustive summation across S and so will also be intractable for high D.

If $\mathbf{s}_1$ and $\mathbf{s}_2$ are an arbitary partition of the variables in $\mathbf{s}$ then importantly the conditional distribution $\mathrm{P}_{\mathbf{s}_1|\mathbf{s}_2}$ will also be Boltzmann machine distributions. However unless the dimensionality of $\mathbf{s}_1$ is small enough that exhaustive summation over its possible states is feasible, then evaluating normalising constants of this conditional distribution and expectations with respect to it will also be intractable. Therefore inference in Boltzmann machines conditioned on observations of part of the state can be considered as a special case of computing expectations and the normalising constants of (non-conditioned) Boltzmann machine distributions, with the same challenges applying to both.

Figure 1.7 shows the factor graph for a Boltzmann machine distribution on five binary random variables $\{\mathrm{s}_i\}_{i=1}^5$. Each of the weights W_{ij} defines an undirected factor between a pair of variables $\mathrm{s}_i W_{ij} \mathrm{s}_j$. As the variables take on signed binary values, this factor is equal to W_{ij} when the variables are equal and so take the same sign and equal to $-W_{ij}$ when the variables take differing values. If W_{ij} is positive this factor therefore favours states where s_i and s_j are in the same configuration, while if W_{ij} is negative states with s_i and s_j in opposing configurations are preferred.

Boltzmann machine systems with a mixture of positive and negative weights will often be *frustrated* with no one global configuration which satisfies the preferences specified by each weight, and instead there being multiple states which each locally satisfy a subset of the soft constraints specified by the weights. This typically leads to a highly multi-modal distribution on the states of the system, with collections of nearby[4] states of high-probability separated sets of states with very low probability.

This multi-modality typically makes frustrated Boltzmann machines very challenging distributions to perform approximate inference with. In particular methods based on constructing Markov chains which explore the state of the system tend to converge very slowly as they will

4 Nearby here being in terms of the Hamming distance between the binary states.

typically remain confined to a particular high-probability region of the state space for many iterations. In Chapter 5 we will consider methods for constructing Markov chains with improved exploration of challenging multi-modal target distributions, including methods for estimating expectations and normalising constants of frustrated Boltzmann machine distributions.

1.3.4 Model comparison

So far we have discussed inferring the unobserved variables in a single fixed model. An important second level of inference is comparing competing models for the same observed data. This can be treated consistently within the probabilistic framework we have discussed.

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.

Given observed data, we would like to be able to make a judgement as to which of two (or more) 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 free, unobserved variables in a model, probabilistic model comparison automatically embodies Ockham's Razor [136].

A concrete structure for model comparison is to assume that there are a finite set of M models, indexed by an *indicator* variable $\mathsf{m} \in \{1 \ldots M\}$. All models share the same observed variables[5] $\mathbf{y}$, and there are a set of per model vectors of unobserved variables $\{\mathbf{x}_m\}_{m=1}^M$ 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

$$\begin{aligned} \mathsf{p}_{\mathbf{y},\mathsf{m},\mathbf{x}_1,\ldots,\mathbf{x}_M}(\boldsymbol{y}, m, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_M) = \\ \mathsf{p}_{\mathbf{x}|\mathsf{m},\mathbf{z}_m}(\boldsymbol{y} \mid m, \boldsymbol{x}_m)\, \mathsf{p}_{\mathsf{m}}(m) \prod_{n=1}^{M} \mathsf{p}_{\mathbf{x}_n}(\boldsymbol{x}_n). \end{aligned} \tag{1.46}$$

5 For notational simplicity here we assume all observed variables have been concatenated in to one vector and similarly for the unobserved variables, with any internal model factorisation structure such as discussed in the preceding sections omitted.

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 $\mathrm{p}_{\mathbf{y}|\mathrm{m},\mathbf{x}_m}$; 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 $\mathbf{y} \perp \{\mathbf{x}_n\}_{n \neq m} \mid \mathrm{m} = m$.

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 are given by

$$\mathrm{p}_{\mathrm{m}|\mathbf{y}}(m \mid \boldsymbol{y}) = \frac{\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid m)\, \mathrm{p}_{\mathrm{m}}(m)}{\sum_{n=1}^{M} \Big(\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid n)\, \mathrm{p}_{\mathrm{m}}(n) \Big)}, \tag{1.47}$$

which can be seen as a direct analogue to Bayes' theorem for the posterior density on unobserved random variables for a single model. The key quantities needed to evaluate the model posterior probabilities are the marginal densities $\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid m)$ evaluated at the observed data. Computing these values requires marginalising out the unobserved variables from the per model joint densities $\mathrm{p}_{\mathbf{y},\mathbf{x}_m|\mathrm{m}}$

$$\mathrm{p}_{\mathbf{y}|\mathrm{m}}(\boldsymbol{y} \mid m) = \int_{X_m} \mathrm{p}_{\mathbf{y}|\mathrm{m},\mathbf{x}_m}(\boldsymbol{y} \mid m, \boldsymbol{x}) \mathrm{p}_{\mathbf{x}_m}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}. \tag{1.48}$$

This value is equivalent to (1.42) for a single model, this explaining the naming of this term as the *model evidence*.

As described previously, evaluating the model evidence requires integrating across the space of all unobserved variables. 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. Unlike the integrals required for making predictions using a single model however, the model evidence integral cannot be naturally expressed as an expectation of a function with respect to the posterior distribution. This can complicate approximate computation of model evidence terms compared to other quantities involved in inference. In Chapter 5 we will consider extensions to a class of methods proposed for estimating model evidence terms.

1.4 SUMMARY

Probabilistic modelling offers a natural way to formalise our beliefs and assumptions about a problem and make inferences given those beliefs. Once a model has been defined the theoretical basis of the inference process is elegantly simple. Underlying this simplicity however are some very significant implementation challenges. The key computational task is the evaluation of integrals across high-dimensional spaces, which typically do not have closed form solutions and are intractable to compute using standard numerical integration approaches.

This intractability necessitates the use of approximate inference methods, the focus of this thesis. In particular we propose several novel extensions to MCMC methods, a class of approaches for drawing dependent samples from high-dimensional target distributions. In the next chapter we review the basic Monte Carlo method for integration and associated methods for generating and using independent pseudo-random variates to estimate the integrals. We then introduce the key Markov chain theory underlying MCMC methods and review some of the key existing MCMC algorithms. We will then conclude with an outline of the remainder of the thesis, in particular giving a a summary of the novel contributions made in this thesis and how these relate to the specific inference problems discussed in the last section of this chapter.

2 APPROXIMATE INFERENCE

In the previous chapter we argued 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. Generally these integrals will not have analytic solutions and for models with even moderate numbers of unobserved 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: 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 and 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. In this chapter we will concentrate sampling-based approaches to approximate inference, focussing in particular on Markov chain Monte Carlo methods, as these form the key basis for the contributions discussed in later chapters.

Although they are not the main focus of this thesis we will make use of several optimisation-based approximate inference methods within the algorithms discussed in the following chapters. We review the ideas underlying these methods in Appendix C.

2.1 MONTE CARLO METHODS

Inference at both the level of computing conditional expectations of unobserved variables in a model and in evaluating evidence terms to allow model comparison involves integrating functions against a probability distribution. Typically the distribution of interest will be defined by a probability density with respect to a base measure. Therefore 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 models, we would instead have that $\tilde{p}$ is the product of unnormalised factors and Z the corresponding normaliser.

2.1.1 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.

The framework that unifies all of the methods we will discuss in this section is the *Monte Carlo* method for integration [226]. Let $\mathbf{x}$ be a random vector distributed according to the target distribution i.e. $\mathrm{P}_{\mathbf{x}} = P$. Given an arbitrary measurable function $f : X \to \mathbb{R}$ we define a random variable $\mathrm{f} = f(\mathbf{x})$. Our task is to compute expectations $\mathbb{E}[\mathrm{f}] = \bar{f}$ corresponding to the integral (2.1). We assume that $\mathbb{E}[\mathrm{f}]$ exists and both $\mathbb{E}[\mathrm{f}]$ and $\mathbb{V}[\mathrm{f}]$ 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 distribution i.e. $\mathrm{P}_{\mathbf{x}_n} = P \;\forall n \in 1 \ldots N$ but potentially not independent of each other. We define random variables

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

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 $\hat{\mathsf{f}}_N$ we can show 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 $\{\mathbf{x}_n\}_{n=1}^N$ and so $\{\mathsf{f}_n\}_{n=1}^N$ are independent, then $\mathbb{C}[\mathsf{f}_n, \mathsf{f}_m] = 0\ \forall m \neq n$. In this case (2.4) reduces to $\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \mathbb{V}[\mathsf{f}]/N$, i.e. the variance of the *Monte Carlo estimate* $\hat{\mathsf{f}}_N$ for $\bar{f}$ is inversely proportional to the number of samples N. Importantly this scaling does not depend on the dimension of $\mathbf{x}$.

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 large constant factors in the Simpson's rule dependence can mean Monte Carlo performs better for practical N.

Note that computing Monte Carlo estimates from independent random variables is not optimal in terms of minimising $\mathbb{V}\left[\hat{\mathsf{f}}_N\right]$ 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 [121]. 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.

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.

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[1], PRNGs produce deterministic sequences of integers in a fixed range that nonetheless maintain many of the properties of a random sequence. 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* [127] which obeys the recurrent update

$$s_{n+1} = (as_n + c) \mod m \quad \text{with} \quad 0 < a < m,\ 0 \leq c < m, \tag{2.5}$$

with a, c and m integer parameters. If a, c and m are chosen appropriately, iterating the update (2.5) 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 [141], hence most modern numerical computing libraries use more robust variants such as the *Mersenne-Twister* [142], 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

1 We consider a true random process as one in which it is impossible to precisely predict the next value in the sequence given the previous values.

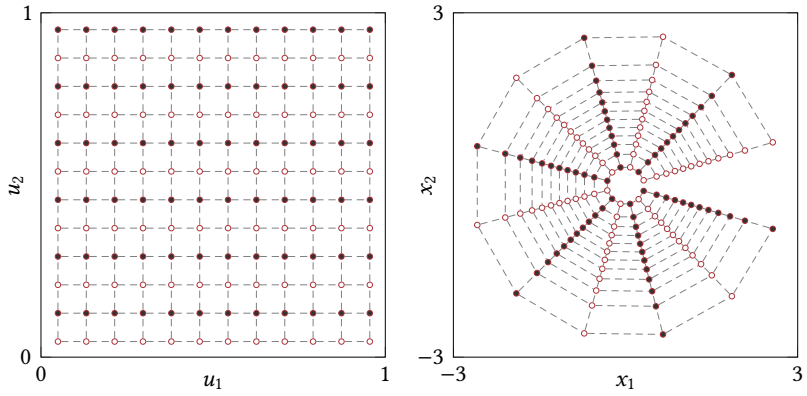


Figure 2.2.: Visualisation of Box–Muller transform. Left axis shows uniform grid on $U = [0,1]^2$ and right-axis shows grid points after mapping through $\boldsymbol{g}$ in transformed space $X = \mathbb{R}^2$.

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 arbitrary densities.

2.1.3 Transform sampling

Samples from many standard distributions can be generated by exploiting the transform of random variables relationships discussed in 1.1.4. Let $\mathbf{u}$ be a D-dimensional vector of independent random variables marginally distributed according to $\mathcal{U}(0,1)$ and $\boldsymbol{g} : [0,1)^D \to X$ be a bijective map to a vector space $X \subseteq \mathbb{R}^D$. If we define $\mathbf{x} = \boldsymbol{g}(\mathbf{u})$, then by (1.22) we have that

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \left| \frac{\partial \boldsymbol{g}^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|. \tag{2.6}$$

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

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

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

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_1^2}{2}\right)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x_2^2}{2}\right), \tag{2.7}$$

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* [37], 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.

A general method for sampling from univarite distributions 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

$$r(x) = \int_{-\infty}^{x} p(v)\,\mathrm{d}v \implies \frac{\partial r(x)}{\partial x} = p(x). \tag{2.8}$$

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

$$\mathrm{p}_{\mathrm{x}}(x) = \left|\frac{\partial r(x)}{\partial x}\right| = p(x). \tag{2.9}$$

To be able to use the inverse CDF transform method we need to be able to evaluate r^{-1}, sometimes termed the *quantile function*. Often neither the CDF or quantile function of a univariate distribution will have closed form solutions however we can use polynomial approximation methods and iterative solvers to evaluate both to arbitary precision [169]. For some distributions such as the standard normal $\mathcal{N}(0,1)$ even though the CDF and quantile function do not have analytic forms in terms of elementary functions it is common for numerical computing libraries to provide numerical approximations to both which are accurate to within small multiples of machine precision. 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 alternatives can be simpler to implement and 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 [166]. 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 distribution* Q which we can generate independent samples from and has

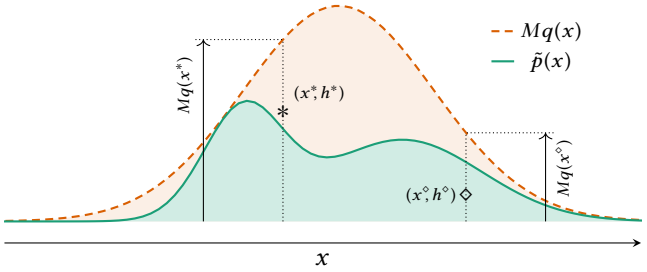


Figure 2.3.: Visualisation of rejection sampling. The green curve shows the (unnormalised) target density $\tilde{p}$, the green region underneath representing the area we wish to sample points uniformly from. The dashed orange curve shows the scaled proposal density Mq, with the orange (plus green) 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 density of proposal distribution Q, M : constant such that $\tilde{p}(\boldsymbol{x}) \leq Mq(\boldsymbol{x})\ \forall \boldsymbol{x} \in X$.
Output: Independent sample from distribution with density $p(\boldsymbol{x}) = \frac{1}{Z}\tilde{p}(\boldsymbol{x})$.

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

a density $q = \frac{\partial Q}{\partial \mu}$ that upper bounds 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 requirement to be able to generate independent samples from Q can be met for example by distributions amenable to transform sampling, e.g. the standard normal. The second requirement is generally more challenging and as we will see the efficiency of rejection sampling methods is very dependent on how tight the bound can be made.

Algorithm 1 describes the rejection sampling method to produce a single independent sample from a target distribution. A visualisation of how the algorithm works for a univariate target distribution 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 accepting only those which are below the green curve. To generate a point

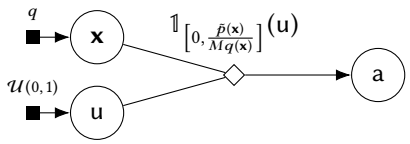


Figure 2.4.: Factor graph of rejection sampling process.

under the dashed orange curve we first generate an x from the proposal distribution 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 auxiliary 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

$$\mathrm{p}_{\mathbf{x},\mathrm{a}}(\boldsymbol{x}, a) = q(\boldsymbol{x})\left(\frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}\right)^{a}\left(1 - \frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}\right)^{1-a}, \tag{2.10}$$

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

$$\mathrm{p}_{\mathrm{a}}(a) = \left(\frac{Z}{M}\right)^{a}\left(1 - \frac{Z}{M}\right)^{1-a}. \tag{2.11}$$

Conditioning on the proposal being accepted we therefore have that

$$\mathrm{p}_{\mathbf{x}|\mathrm{a}}(\boldsymbol{x} \mid 1) = \frac{q(\boldsymbol{x})\frac{\tilde{p}(\boldsymbol{x})}{Mq(\boldsymbol{x})}}{\frac{Z}{M}} = \frac{\tilde{p}(\boldsymbol{x})}{Z} = p(\boldsymbol{x}). \tag{2.12}$$

Therefore the accepted proposals are distributed according to the target density as required. Further from (2.11) 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.

Although rejection sampling can be an efficient method of sampling from univariate target distributions (particularly for distributions with log-concave densities where adaptive variants areavailable [91]), it gen-

erally scales very poorly with the dimensionality of the target distribution. This is as the ratio of the volume under the target density to the volume under the scaled proposal density (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 an example of 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 a target distribution. Although samples can be of value in themselves for giving a representative set of plausible values from the target distribution (e.g. for visualisation purposes), usually the end goal is to estimate integrals of the form in (2.1).

Importance sampling [116] is a Monte Carlo method which allows arbitrary integrals to be estimated. If Q is a distribution, with density $q = \frac{\partial Q}{\partial \mu}$, which is absolutely continuous with respect to the target distribution (which requires that $p(\boldsymbol{x}) > 0 \Rightarrow q(\boldsymbol{x}) > 0$), then importance sampling is based on the identity

$$\bar{f} = \frac{\int_X f(\boldsymbol{x})\,\tilde{p}(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x})}{\int_X \tilde{p}(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x})} = \frac{\int_X f(\boldsymbol{x})\,\frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\,q(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x})}{\int_X \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}\,q(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x})}. \tag{2.13}$$

Each of the numerator and denominator in (2.13) take the form of an expectation of a measurable function of a random variable $\mathbf{x}$ with distribution Q. Further the denominator is exactly equal to $Z = \int_X \tilde{p}(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x})$. We therefore have that

$$Z\bar{f} = \mathbb{E}[w(\mathbf{x})f(\mathbf{x})] \;\text{ and }\; Z = \mathbb{E}[w(\mathbf{x})] \;\text{ with }\; w(\boldsymbol{x}) = \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})}. \tag{2.14}$$

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 $\hat{\mathrm{Z}}_N$ and $\hat{\mathrm{g}}_N$ as

$$\hat{\mathrm{Z}}_N = \frac{1}{N}\sum_{n=1}^{N} w(\mathbf{x}_n) \;\text{ and }\; \hat{\mathrm{g}}_N = \frac{1}{\hat{\mathrm{Z}}}\sum_{n=1}^{N} w(\mathbf{x}_n)f(\mathbf{x}_n). \tag{2.15}$$

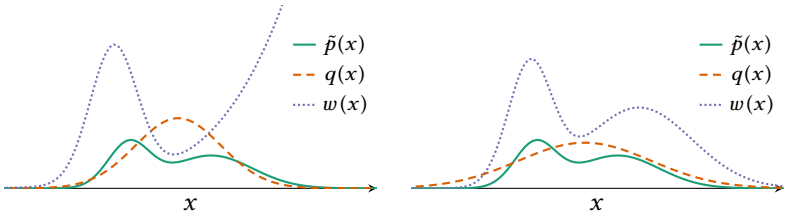


Figure 2.5.: Visualisation of importance sampling. On both axes the green curve shows the unnormalised target density $\tilde{p}$, the dashed orange curve the density q values are sampled from and the dotted violet curve the importance weighting function $w(x) = \frac{\tilde{p}(x)}{q(x)}$ to estimate 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.

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[2] 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 $w(x_n)$ values are typically termed the *importance weights*. If a few of the weights are very large, the weighted sums in (2.15) 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. 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 [136, 170]. Figure 2.5 shows a visualisation of the effect of the choice of q on the importance weights.

When previously discussing rejection sampling, we introduced an auxiliary binary accept indicator variable, a, associated with each proposed

2 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

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

$$\hat{\mathsf{f}}_N^{\text{RS}} = \frac{\sum_{n=1}^N \mathsf{a}_n f(\mathbf{x}_n)}{\sum_{m=1}^N \mathsf{a}_m}, \tag{2.16}$$

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

$$\hat{\mathsf{f}}_N^{\text{IS}} = \frac{\sum_{n=1}^N w(\mathbf{x}_n) f(\mathbf{x}_n)}{\sum_{m=1}^N w(\mathbf{x}_m)}. \tag{2.17}$$

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 $\hat{\mathsf{f}}_N^{\text{RS}}$ is unbiased unlike $\hat{\mathsf{f}}_N^{\text{IS}}$, 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 [136].

2.2 MARKOV CHAIN MONTE CARLO

The sampling methods considered in the previous section used independent random variables to form Monte Carlo estimates. When in-

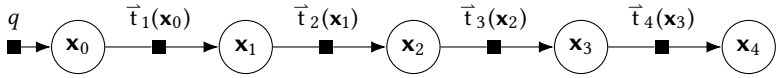


Figure 2.6.: Markov chain factor graph. The initial state $\mathbf{x}_0$ is sampled according to a density q and each subsequent state $\mathbf{x}_n$ is then generated from a transition density $\vec{\mathrm{t}}_n$ conditioned on the previous state $\mathbf{x}_{n-1}$.

troducing 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*, $\vec{\mathrm{T}}_n : \mathcal{F} \times X \to [0, 1]$. In particular the transition operators define a series of regular conditional distributions for each $n \in \{1 \dots N\}$

$$\mathrm{P}_{\mathbf{x}_n | \mathbf{x}_{n-1}}(A \mid \boldsymbol{x}) = \vec{\mathrm{T}}_n(A \mid \boldsymbol{x}) \quad \forall A \in \mathcal{F},\ \boldsymbol{x} \in X. \tag{2.18}$$

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

The key property required of a transition operator for use in MCMC methods is that the target distribution P is *invariant* under the transition, that is it satisfies

$$P(A) = \int_X \vec{\mathrm{T}}(A \mid \boldsymbol{x})\, \mathrm{d}P(\boldsymbol{x}) \quad \forall A \in \mathcal{F}, \tag{2.19}$$

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 de-

pendent 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 generally infeasible. Importantly however 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 if the target distribution is the *unique* invariant distribution of the chain.

To have a unique invariant distribution, a chain must be *irreducible* and *aperiodic* [220]. 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.

$$\forall \boldsymbol{x} \in X,\, A \in \mathcal{F} : \nu(A) > 0 \;\; \exists\, m \in \mathbb{Z}^+ : \mathsf{P}_{\mathbf{x}_m|\mathbf{x}_0}(A \,|\, \boldsymbol{x}) > 0. \tag{2.20}$$

A chain is periodic (and aperiodic otherwise) if disjoint regions of X 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 $\vec{\mathsf{T}}(A_j \,|\, \boldsymbol{x}) = 1\ \forall \boldsymbol{x} \in A_i,\ i \in \{1 \ldots r\},\ j = (i+1) \mod r$.

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 a MCMC estimator $\hat{\mathsf{f}}_N = \frac{1}{N}\sum_{n=1}^N f(\mathbf{x}_n)$ 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[3] [147]. 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* [29].

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

3 The 'except for a ν-null set' caveat can be removed by requiring the stronger property of *Harris recurrence* [105].

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

$$\|\mu - \nu\|_{\text{TV}} = \sup_{A \in \mathcal{F}} |\mu(A) - \nu(A)|. \tag{2.21}$$

For a ν-irreducible and aperiodic chain with invariant distribution P our earlier 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 $\mathbf{x}_0 = \boldsymbol{x}$, $\lim_{n\to\infty} \|\mathsf{P}_{\mathbf{x}_n|\mathbf{x}_0}(\cdot \mid \boldsymbol{x}) - P\|_{\text{TV}} = 0$. Geometric ergodicity makes a stronger statement that the convergence in total variation distance conditioned is geometric in n, i.e. that

$$\|\mathsf{P}_{\mathbf{x}_n|\mathbf{x}_0}(\cdot \mid \boldsymbol{x}) - P\|_{\text{TV}} \leq M(\boldsymbol{x}) r^n \tag{2.22}$$

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

$$\mathbb{V}\left[\hat{\mathsf{f}}_N\right] = \frac{\mathbb{V}[\mathsf{f}]}{N}\left(1 + 2\sum_{n=1}^{N-1}\left(\frac{N-n}{N}\frac{\mathbb{C}[\mathsf{f}_0, \mathsf{f}_n]}{\mathbb{V}[\mathsf{f}]}\right)\right). \tag{2.23}$$

If we multiply both sides of (2.23) 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

$$\lim_{N\to\infty}\left(N\,\mathbb{V}\left[\hat{\mathsf{f}}_N\right]\right) = \mathbb{V}[\mathsf{f}]\left(1 + 2\sum_{n=1}^{\infty}\rho_n\right). \tag{2.24}$$

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 [45, 86, 197] that (2.24) is also the asymptoptic variance for a MCMC estimator calculated using the chain states.

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

$$N_{\text{eff}} = \frac{N}{1 + 2\sum_{n=1}^{\infty} \rho_n}. \tag{2.25}$$

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 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 [219]. 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 [43, 180, 206].

So far we have not discussed how to 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 conditional density we term the *transition density* $\vec{\mathsf{t}} : X \times X \to [0, \infty)$ which is defined with respect to a base measure μ (which we 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

$$\vec{\mathsf{T}}(A \mid \boldsymbol{x}) = \int_A \vec{\mathsf{t}}(\boldsymbol{x}' \mid \boldsymbol{x}) \, \mathrm{d}\mu(\boldsymbol{x}') \quad \forall A \in \mathcal{F}, \ \boldsymbol{x} \in X. \tag{2.26}$$

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 $\vec{\mathsf{T}}$ is not absolutely continuous with respect to μ and a transition density is not strictly well 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

$$p(\boldsymbol{x}') = \int_X \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x}) \quad \forall \boldsymbol{x}' \in X. \tag{2.27}$$

Finding a transition density which leaves the target density invariant by satisfying (2.27) 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

$$\mathrm{p}_{\mathbf{x}_n, \mathbf{x}_{n-1}}(\boldsymbol{x}', \boldsymbol{x}) = \mathrm{p}_{\mathbf{x}_n \mid \mathbf{x}_{n-1}}(\boldsymbol{x}' \mid \boldsymbol{x})\, \mathrm{p}_{\mathbf{x}_{n-1}}(\boldsymbol{x}) = \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}). \tag{2.28}$$

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 $\mathrm{p}_{\mathbf{x}_{n-1} \mid \boldsymbol{x}_n}$. Due to stationarity $\mathrm{p}_{\mathbf{x}_n}$ is also equal to p and so we have that $\mathrm{p}_{\mathbf{x}_{n-1} \mid \boldsymbol{x}_n}$ 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[4]. If we therefore denote $\overleftarrow{\mathrm{t}} = \mathrm{p}_{\mathbf{x}_{n-1} \mid \boldsymbol{x}_n}$ (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

$$\vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) = \overleftarrow{\mathrm{t}}(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}') \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{2.29}$$

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

$$\int_X \vec{\mathrm{t}}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x}) = \int_X \overleftarrow{\mathrm{t}}(\boldsymbol{x} \mid \boldsymbol{x}')\, \mathrm{d}\mu(\boldsymbol{x})\, p(\boldsymbol{x}') = p(\boldsymbol{x}'), \tag{2.30}$$

and so that (2.27) 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 (2.29), then the transition operator specified by $\vec{\mathrm{t}}$ will leave the target distribution P

4 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.

invariant (and by an equivalent argument so will the transition operator specified by $\overleftarrow{\mathrm{t}}$). We can further simplify (2.29) by requiring that $\overrightarrow{\mathrm{t}} = \overleftarrow{\mathrm{t}} = \mathrm{t}$, 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

$$\mathrm{t}(\boldsymbol{x}' \,|\, \boldsymbol{x})\, p(\boldsymbol{x}) = \mathrm{t}(\boldsymbol{x} \,|\, \boldsymbol{x}')\, p(\boldsymbol{x}') \quad \forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X. \tag{2.31}$$

This is often termed the *detailed balance* condition. Importantly both the detailed balance (2.31) and *generalised balance* (2.29) conditions can 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 (2.27) 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 [59, 114, 164]. 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 [213, 214, 225] or chains with tractable invariant distributions such as multivariate normal [28].

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.

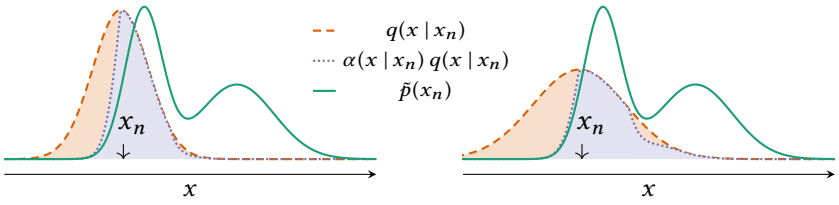


Figure 2.7.: Visualisation of Metropolis–Hastings algorithm in a univariate target density. The green curves shows the unnormalised target density. The arrows indicate the current chain state. The orange curves show the density of proposed moves from this state, with the left axis using a narrower proposal 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.

Having now introduced the key theory underlying MCMC methods, we will now discuss practical implementations of the approach. In the following sub-sections we will review two of the most popular frameworks for constructing reversible transition operators which leave a target distribution invariant: the *Metropolis–Hastings* algorithm and *Gibbs sampling*.

2.2.1 Metropolis–Hastings

Although the algorithm has come to be commonly known by Edward Metropolis' name as first author on the 1953 paper [146], it is believed that Arianna and Marshall Rosenbluth, two of the other co-authors, were the main contributors to the development of the algorithm [103].

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 [146] dates to the very beginnings of the Monte Carlo method, having being first implemented on Los Alamos' MANIAC[5] one of the earliest programmable computers. The method was generalised in a key paper by Hastings [110], and the optimality among several competing alternatives of the form now used demonstrated by Peskun [176]. An extension to Markov chains on trans-dimensional spaces was proposed by Green [101].

An outline of the method is given in Algorithm 2 and a visualisation of its application to a univariate target distribution shown in Figure 2.7.

5 *Mathematical Analyzer, Numerical Integrator and Computer.*

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

1: $\boldsymbol{x}^* \sim q(\cdot \mid \boldsymbol{x}_n)$ ▷ Generate proposed new state
2: $u \sim \mathcal{U}(\cdot \mid 0, 1)$
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: $\quad \boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}_n$ ▷ Proposed move rejected

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

$$\begin{aligned} \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x}) = \alpha(\boldsymbol{x}' \mid \boldsymbol{x})\, q(\boldsymbol{x}' \mid \boldsymbol{x}) + \\ \left(1 - \int_X \alpha(\boldsymbol{x}^* \mid \boldsymbol{x})\, q(\boldsymbol{x}^* \mid \boldsymbol{x})\, \mathrm{d}\mu(\boldsymbol{x}^*)\right) \delta(\boldsymbol{x}' - \boldsymbol{x}), \end{aligned} \tag{2.32}$$

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

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}\right\} = \min\left\{1, \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, \tilde{p}(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, \tilde{p}(\boldsymbol{x})}\right\}, \tag{2.33}$$

and $q : X \times X \to [0, \infty)$ the *proposal density*.

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 [110]), in which case the acceptance probability definition simplifies to

$$\alpha(\boldsymbol{x}' \mid \boldsymbol{x}) = \min\left\{1, \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})}\right\} = \min\left\{1, \frac{\tilde{p}(\boldsymbol{x}')}{\tilde{p}(\boldsymbol{x})}\right\}. \tag{2.34}$$

Note that in both (2.33) and (2.34) the target density only appears as a ratio and so only need be known up to a constant.

For the purposes of verifying the detailed balance condition (2.31), the density of *self-transitions*, i.e. a transition to the same state, can be ignored as (2.31) 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

component corresponding to rejected proposals can be neglected, we have $\forall \boldsymbol{x} \in X,\, \boldsymbol{x}' \in X : \boldsymbol{x} \neq \boldsymbol{x}$

$$\begin{aligned}
\mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) &= \min\left\{1,\, \frac{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')}{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}\right\} q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}) && (2.35)\\
&= \min\{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x}),\, q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')\} && (2.36)\\
&= \min\left\{\frac{q(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})}{q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}')},\, 1\right\} q(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}') && (2.37)\\
&= \mathrm{t}(\boldsymbol{x} \mid \boldsymbol{x}')\, p(\boldsymbol{x}'). && (2.38)
\end{aligned}$$

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

An important special case for chains on a Euclidean state space $X = \mathbb{R}^D$, 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

$$\begin{aligned}
\mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x}) &= \delta(\boldsymbol{x}' - \boldsymbol{\phi}(\boldsymbol{x}))\alpha(\boldsymbol{x}) + \delta(\boldsymbol{x}' - \boldsymbol{x})(1 - \alpha(\boldsymbol{x})),\\
\alpha(\boldsymbol{x}) &= \min\left\{1,\, \frac{p \circ \boldsymbol{\phi}(\boldsymbol{x})}{p(\boldsymbol{x})} D_{\boldsymbol{\phi}}(\boldsymbol{x})\right\},
\end{aligned} \tag{2.39}$$

then this transition operator will leave the target distribution P invariant. This deterministic transition operator variant is as a special case of the trans-dimensional Metropolis–Hastings extension introduced by Green [87, 101]. 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 (2.27)

$$\begin{aligned}
&\int_X \mathrm{t}(\boldsymbol{x}' \mid \boldsymbol{x})\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} && (2.40)\\
&= \int_X \delta(\boldsymbol{x}' - \boldsymbol{\phi}(\boldsymbol{x}))\, \alpha(\boldsymbol{x})\, p(\boldsymbol{x}) + \delta(\boldsymbol{x}' - \boldsymbol{x})(1 - \alpha(\boldsymbol{x}))\, p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} && (2.41)\\
&= \int_X \delta(\boldsymbol{x}' - \boldsymbol{y})\, \alpha \circ \boldsymbol{\phi}(\boldsymbol{y})\, p \circ \boldsymbol{\phi}(\boldsymbol{y})\, D_{\boldsymbol{\phi}}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y} + (1 - \alpha(\boldsymbol{x}'))\, p(\boldsymbol{x}') && (2.42)\\
&= p(\boldsymbol{x}') + \alpha \circ \boldsymbol{\phi}(\boldsymbol{x}')\, p \circ \boldsymbol{\phi}(\boldsymbol{x}')\, D_{\boldsymbol{\phi}}(\boldsymbol{x}') - \alpha(\boldsymbol{x}')\, p(\boldsymbol{x}'). && (2.43)
\end{aligned}$$

In going from (2.42) to (2.43) 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

$$\alpha \circ \boldsymbol{\phi}(\boldsymbol{x}')p \circ \boldsymbol{\phi}(\boldsymbol{x}')D_{\boldsymbol{\phi}}(\boldsymbol{x}') = \min\left\{p \circ \boldsymbol{\phi}(\boldsymbol{x}')D_{\boldsymbol{\phi}}(\boldsymbol{x}'), p(\boldsymbol{x}')\right\} = \alpha(\boldsymbol{x}')p(\boldsymbol{x}').$$

The last two terms in (2.43) therefore cancel and so (2.27) is satisfied by the transition operator defined by (2.39).

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 [220], 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 [197]. If the set of points with a non-zero probability of rejection has non-zero P-measure, then the transition operator is aperiodic [220].

A common choice of proposal density when the target distribution is defined on $\mathbb{R}^D$ is a multivariate normal density centred at the current state i.e. $q(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}' \mid \boldsymbol{x}, \Sigma)$ which satisfies the positivity condition for irreducibility. In general we would achieve optimal performance with a proposal density covariance Σ which is proportional to the covariance of the target distribution [200]. In practice we do not have access to the true covariance and so typically an isotropic proposal density is used with covariance $\Sigma = \sigma^2\mathbf{I}$ 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 (2.34) can be used, further the proposal density depends only on the difference $\boldsymbol{x}' - \boldsymbol{x}$ with Metropolis–Hastings methods having these properties often termed *random-walk Metropolis*.

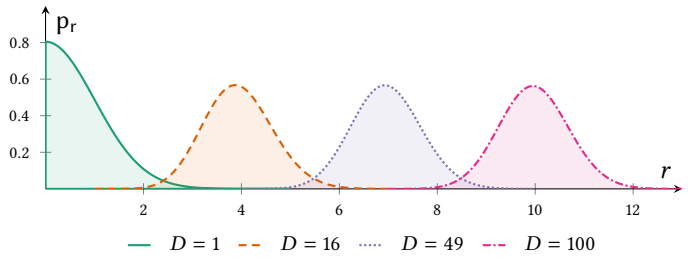


Figure 2.8.: Illustration of concentration of measure in a multivariate normal distribution. The plots shows the probability density of the distance from the origin $\mathrm{r} = \|\mathbf{x}\|_2$ of a D-dimensional multivariate normal random vector $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 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 $\boldsymbol{\mu}$.

Random walk Metropolis methods have been extensively theoretically studied, with sufficient conditions known in some cases to ensure geometric ergodicity of a chain [145, 199] 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 [79, 196] and adaptive variants which automatically estimate a non-isotropic proposal covariance [104, 200].

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 depencies 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 [10, 136]. An illustration of this phenomenon for the multivariate normal distribution 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 tar-

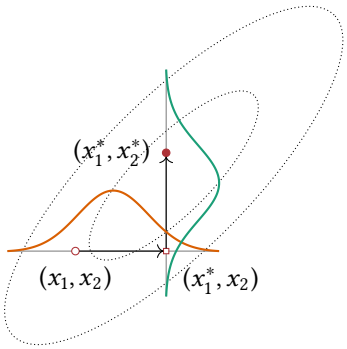


Figure 2.9.: Schematic of Gibbs sampling transition in a bivariate normal target distribution (ellipses indicate constant density contours). Given an initial state $\mathbf{x} = (x_1, x_2)$, the x_1 (horizontal) co-ordinate is first updated by independently sampling from the normal conditional $p_{x_1|x_2}(\cdot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

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 $\boldsymbol{x}_n \sim p(\cdot) \implies \boldsymbol{x}_{n+1} \sim p(\cdot)$.

1: $\boldsymbol{x} \leftarrow \boldsymbol{x}_n$
2: **for** $i \in I$ **do**
3: $\quad x_i \sim p_i(\cdot \mid \boldsymbol{x}_{\setminus i})$ ▷ Resample x_i from p_i given *current* $\boldsymbol{x}_{\setminus i}$.
4: $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}$

get 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 [24].

2.2.2 Gibbs sampling

Gibbs sampling [78, 84], originally proposed by Geman and Geman for image restoration using a Markov random field image model, is based 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 variables $\mathbf{x}_{\setminus i}$. 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.

$$p(\boldsymbol{x}) = p_i(x_i \mid \boldsymbol{x}_{\setminus i})\, p_{\setminus i}(\boldsymbol{x}_{\setminus i}), \tag{2.44}$$

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 [44, 198]. 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 sequential scan version this defines a reversible transition operator. The random update variant is more amenable to theoretical analysis, 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 [111]. A compromise between the completely random updates and a sequential scan is to randomly permute the update order after 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 [90] and JAGS [179]. A well-known issue with Gibbs sampling however is that its performance

is highly dependent on the parameterisation used for the target density [186], 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 terms such as *Metropolis-within-Gibbs* are sometimes used to refer to methods which sequentially apply Metropolis transition operators which each update only a subset of variables in the target distribution. We will 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.

2.3 AUXILIARY VARIABLE METHODS

Although Gibbs sampling and random-walk Metropolis are commonly used in practice, as discussed above both have drawbacks when applied to complex high-dimensional target distributions. One approach which has proven particularly successful for constructing alternative Markov transition operators which can overcome some of these shortcomings is the introduction of *auxiliary variables* in to the chain state. For concreteness of notation in the following discussion we let the variables of interest, which we term the target variables, be represented by the random vector $\mathbf{x} \in X$ and the introduced auxiliary variables by the random vector $\mathbf{a} \in A$. We assume for generality here multiple auxiliary variables are introduced, however methods using a single scalar auxiliary variable are a common special case.

One way of defining a joint distribution across the target and auxiliary variables is to specify an arbitrary conditional distribution $\mathrm{P}_{\mathbf{a}|\mathbf{x}}$

and choose the marginal distribution $\mathsf{P}_{\mathbf{x}}$ to be equal to the target distribution P. Given samples from a joint distribution we can estimate expectations with respect to the marginal distribution on a subset of the variables by simply ignoring the dimensions of the sampled state we wish to marginalise over. Therefore if we can construct a Markov chain with the resulting joint distribution $\mathsf{P}_{\mathbf{x},\mathbf{a}}$ as its unique invariant distribution, then we can use components of the sampled states corresponding to the target variables to estimate expectations with respect to the target distribution. We will consider two MCMC methods apply this approach, *slice sampling* and *Hamiltonian Monte Carlo*, in the follow subsections.

An alternative approach is to instead construct a joint distribution on the target and auxiliary variables such that the conditional distribution $\mathsf{P}_{\mathbf{x}|\mathbf{a}}$ is equal to the target distribution P across some set of values $A^* \subset A$ of the auxiliary variables, which can be expressed in terms of the density $\mathsf{p}_{\mathbf{x}|\mathbf{a}}$ as

$$\mathsf{p}_{\mathbf{x}|\mathbf{a}}(\boldsymbol{x} \mid \boldsymbol{a}) = p(\boldsymbol{x}) \qquad \forall \boldsymbol{x} \in X, \boldsymbol{a} \in A^*. \tag{2.45}$$

If the resulting marginal probability $\mathsf{P}_{\mathbf{a}}(A^*)$ is non-zero, then the sampled states $\{\boldsymbol{x}^{(s)}, \boldsymbol{a}^{(s)}\}_{s=1}^{S}$ of a Markov chain which has $\mathsf{P}_{\mathbf{x},\mathbf{a}}$ as its unique invariant distribution can be used to estimate expectations with respect to the target distribution by computing averages over only the sampled target variable values $\boldsymbol{x}^{(s)}$ for which the corresponding auxiliary variables $\boldsymbol{a}^{(s)}$ take values in A^* (these being at convergence samples from $\mathsf{P}_{\boldsymbol{x}|\boldsymbol{a}}(\cdot \mid \boldsymbol{a})$ and so the target distribution), i.e.

$$\int_X f(\boldsymbol{x}) \,\mathrm{d}P(\boldsymbol{x}) = \lim_{S \to \infty} \frac{\sum_{s=1}^{S} \mathbb{1}_{A^*}(\boldsymbol{a}^{(s)}) f(\boldsymbol{x}^{(s)})}{\sum_{s=1}^{S} \mathbb{1}_{A^*}(\boldsymbol{a}^{(s)})}. \tag{2.46}$$

We will discuss *simulated tempering*, an MCMC method which introduces an auxiliary variable in this manner in a subsequent subsection.

An issue with this approach is that if $\mathsf{P}_{\mathbf{a}}(A^*)$ is small, the number of sampled states with $\boldsymbol{a} \in A^*$ may be very small or even zero. This can require a large number of samples S for the sufficient samples with auxiliary variables in the required set A^* to be generated to allow the MCMC estimates computed using (2.46) to be reliable.

The estimator in (2.46) has a close resemblance to the formulation of the Monte Carlo estimator corresponding to rejection sampling given

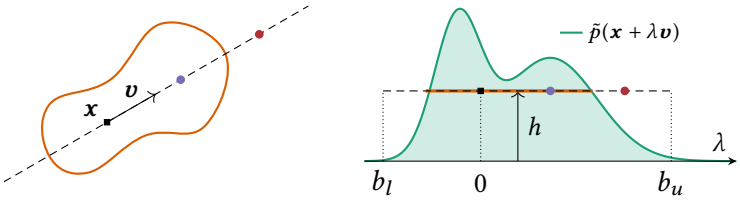


Figure 2.10.: Schematic of linear slice sampling, showing 'plan' (left) and 'cross-sectional' (right) views of a bivariate target density. Orange curve (left) and line (right) indicates a constant density slice S_h. The black square indicates current target state value $\boldsymbol{x}$ and the dashed line is *slice line*, the one-dimensional linear sub-space aligned 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 slice 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.

in (2.16), with averages computed over the subset of samples meeting an 'acceptance' criteria. As noted previously rejection sampling can in fact be considered as an auxiliary variable method, with binary accept indicator variables $a \in \{0, 1\}$ introduced such that the conditional density $p_{\mathbf{x}|a}(\boldsymbol{x} \mid 1)$ is equal to the target density $p(\boldsymbol{x})$ as shown in (2.12), i.e. exactly corresponding to the property in (2.45). Rejection sampling can therefore be seen to be another example of this construct, though in this case each pair of target – auxiliary variable samples are generated independently rather than by constructing a Markov chain.

When discussing the rejection sampling estimator (2.16) we saw there was a close link to importance sampling estimator (2.17), with the importance sampling estimator having the potential advantage however of using all of the generated samples in computing estimates. In Chapter 5 we will discuss a related alternative approach to constructing estimators for auxiliary variable methods based on conditioning like simulated tempering, which unlike the estimator in (2.46) allows using all of the samples in a Markov chain to compute estimates.

2.3.1 Slice sampling

Slice sampling is a family of auxiliary variable MCMC methods which exploit the same observation as used to motivate rejection sampling - to sample from a target distribution it is sufficient to uniformly sample

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 $\boldsymbol{x}_n \sim p(\cdot) \implies \boldsymbol{x}_{n+1} \sim p(\cdot)$.

$h \sim \mathcal{U}(\cdot \mid 0, \tilde{p}(\boldsymbol{x}_n))$ ▷ Sample slice height
$\boldsymbol{v} \sim q(\cdot)$ ▷ Sample vector setting slice line and initial bracket width
$b_u \sim \mathcal{U}(\cdot \mid 0, 1)$ ▷ Uniformly sample bracket around current state
$b_l \leftarrow b_u - 1$
if $M > 0$ **then** $(b_l,\, b_u) \leftarrow$ LinearStepOut$(\boldsymbol{x}_n,\, b_l,\, b_u,\, M)$
$\lambda \sim \mathcal{U}(\cdot \mid b_l, b_u)$
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
        $\lambda \sim \mathcal{U}(\cdot \mid b_l, b_u)$ ▷ Sample uniformly from new bracket
    else ▷ Proposed state on slice
        return $\boldsymbol{x}^*$
function LinearStepOut$(\boldsymbol{x}_n,\, b_l,\, b_u,\, M)$
    $L \sim \text{UniformInt}(\cdot \mid 0, M)$ ▷ 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 $b_l,\, b_u$

from the volume beneath a graph of the target density function. Rather than 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 [161, 163]. A related algorithm which uses per data-point auxiliary variables in Bayesian inference problems developed by Damien, Wakefield and Walker [54]. Murray, Adams and Mackay later proposed *elliptical slice sampling* [156], an extension of Neal's slice sampling method which is particularly effective for target distributions which are well approximated by a multivariate normal and which we will discuss at the end of this subsection.

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 $\mathsf{p}_{\mathsf{h}|\mathbf{x}}(h \mid \mathbf{x}) = \mathcal{U}(h \mid 0, \tilde{p}(\boldsymbol{x})) = \frac{1}{\tilde{p}(\boldsymbol{x})} \mathbb{1}_{[0, \tilde{p}(\boldsymbol{x}))}(h)$, i.e. uniform

over the interval between zero and the unnormalised target density value. The joint density on the augmented space is then

$$\mathrm{p}_{\mathbf{x},\mathsf{h}}(\boldsymbol{x}, h) = \frac{1}{\tilde{p}(\boldsymbol{x})} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h) \frac{\tilde{p}(\boldsymbol{x})}{Z} = \frac{1}{Z} \mathbb{1}_{[0,\tilde{p}(\boldsymbol{x}))}(h). \tag{2.47}$$

Marginalising (2.47) over $\mathbf{x}$ recovers the target density i.e. $\mathrm{p}_{\mathbf{x}} = p$.

The overall slice sampling transition is 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 $\mathrm{p}_{\mathsf{h}|\mathbf{x}}$ and $\mathrm{p}_{\mathbf{x}|\mathsf{h}}$ 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 $\mathrm{p}_{\mathsf{h}|\mathbf{x}}$ 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 $\mathrm{p}_{\mathbf{x}|\mathsf{h}}(\boldsymbol{x} \mid h)$ 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 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 multi-modal densities. The complexity of the slices generally prevents us therefore from being able to independently sample 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 $\mathrm{p}_{\mathsf{h}|\mathbf{x}}$ and $\mathrm{p}_{\mathbf{x}|\mathsf{h}}$.

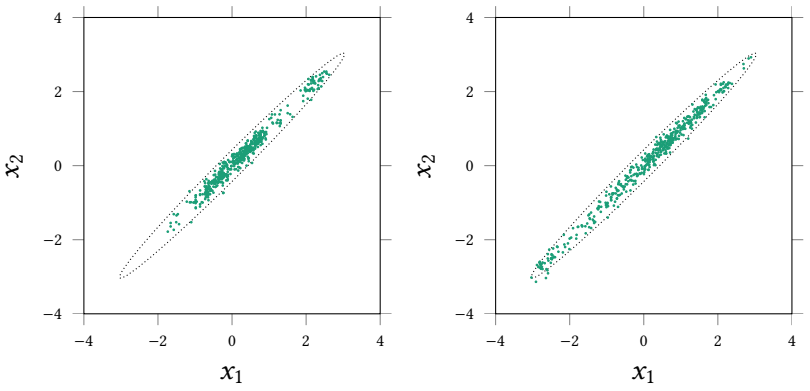
A key contribution of [163] was to introduce an elegant method for constructing a transition operator which leaves $\mathrm{p}_{\mathbf{x}|\mathsf{h}}$ invariant. In particular the method proposed has few free parameters to tune, has an efficiency which is relatively robust to the choices of the free 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 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 [163] the one-dimensional subspace is chosen to be axis-aligned, correponding to updating a single component of the target state.

In this case the restriction of the slice 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 which 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 [163] 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 simpler 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, which we will discuss shortly, will often be a better choice.



(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.

This random-direction slice sampling variant is discussed in comparison to elliptical slice sampling in [156]. It is also bears resemblance to the scheme proposed in [46] 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 applying 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 isotropic random-direction updates are able to more effectively explore the target density.

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 (2.31) 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 ideally chosen based on any knowledge of the typical 'scale' of the target density. Often 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 [163] 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 corresponds 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 [163]. 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.

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 $\boldsymbol{x}^*$

Once the initial bracket has been generated and potentially stepped out, the remainder of the algorithm consists of finding a point on the slice line 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 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 [156] 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 an optimisation based approximate inference scheme such as those discussed in Appendix C [167].

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

$$\boldsymbol{x}^*(\theta) = (\boldsymbol{x}_n - \boldsymbol{\mu}) \cos\theta + (\boldsymbol{v} - \boldsymbol{\mu}) \sin\theta + \boldsymbol{\mu}. \tag{2.48}$$

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

Algorithm 6 Hamiltonian Monte Carlo transition.

Input: $\boldsymbol{x}_n, \boldsymbol{p}_n$: current position–momentum state pair, $\tilde{p}$: differentiable unnormalised target density,
δt : leapfrog integrator step size
Output: $\boldsymbol{x}_{n+1}$: next chain state with $\boldsymbol{x}_n \sim p \implies \boldsymbol{x}_{n+1} \sim p$.

$\boldsymbol{x}^* \leftarrow \boldsymbol{x}_n + \frac{\delta t}{2}\boldsymbol{M}^{-1}\boldsymbol{p}_n$
$\boldsymbol{p}^* \leftarrow \boldsymbol{p}_n - \delta t \nabla\phi(\boldsymbol{x}^*)$
for $s \in \{1 \dots L-1\}$ **do**
    $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^* + \delta t \boldsymbol{L}^{-1}\boldsymbol{M}^{-1}\boldsymbol{p}^*$
    $\boldsymbol{p}^* \leftarrow \boldsymbol{p}^* - \delta t \nabla\phi(\boldsymbol{x}^*)$
$\boldsymbol{x}^* \leftarrow \boldsymbol{x}^* + \frac{\delta t}{2}\boldsymbol{M}^{-1}\boldsymbol{p}^*$
$u \sim \mathcal{U}(0,1)$
$\alpha \leftarrow \exp\left(\phi(\boldsymbol{x}_n) + \frac{1}{2}\boldsymbol{p}_n\boldsymbol{M}^{-1}\boldsymbol{p}_n - \phi(\boldsymbol{x}^*) - \frac{1}{2}\boldsymbol{p}^*\boldsymbol{M}^{-1}\boldsymbol{p}^*\right)$
if $u < \alpha$ **then**
    $\boldsymbol{x}_{n+1}, \boldsymbol{p}_{n+1} \leftarrow \boldsymbol{x}^*, \boldsymbol{p}^*$ ▷ Proposed move accepted
else
    $\boldsymbol{x}_{n+1}, \boldsymbol{p}_{n+1} \leftarrow \boldsymbol{x}_n, -\boldsymbol{p}_n$ ▷ Proposed move rejected
return $\boldsymbol{x}_{n+1}, \boldsymbol{p}_{n+1}$

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.

2.3.2 Hamiltonian Monte Carlo

The MCMC algorithms discussed so far have required only the ability to evaluate a (unnormalised) density function for the target distribution of interest. For distributions defined on real-valued variables the target density function $\tilde{p}$ will often be differentiable - that is the gradient

$\frac{\partial \tilde{p}}{\partial \boldsymbol{x}}$ exists P-almost everywhere. In these cases it is natural to consider using the gradient to help guide updates to the state.

A particularly powerful auxiliary variable MCMC method utilising gradient information is *Hamiltonian Monte Carlo* (HMC) [66, 165]. HMC introduces auxiliary *momentum* variables in to the chain state and then uses simulated Hamiltonian dynamics trajectories in the augmented space to generate proposed updates to the momentum–target variables state pair. The simulated Hamiltonian dynamics exhibit key geometric properties that make HMC well suited to performing MCMC in complex target distributions on high-dimensional spaces, with the method often scaling better to high-dimensional target distributions than simpler methods such as random-walk Metropolis and Gibbs sampling [27].

What we refer to as Hamiltonian Monte Carlo here was introduced in [66] as Hybrid Monte Carlo. *The alternative* Hamiltonian Monte Carlo *was suggested by MacKay [136] to emphasise the important role of Hamiltonian dynamics in the method while maintaining the same acronym [25].*

Most implementations of HMC require the target density p is defined with respect to the Lebesgue measure on a Euclidean space $X = \mathbb{R}^D$. Target densities with bounded support on $\mathbb{R}^D$ add complication to the algorithm by requiring checks that proposed updates to the state remain within the support of the target distribution, however often a change of variables can be performed with a bijective transformation that maps to a density with unbounded support, for example taking a log-transform of a positive variable, with the change of variables formula (1.22) used to compute the resulting density on the transformed space. We will consider extensions of the standard HMC approach to distributions defined on implicitly-defined manifolds embedded in a Euclidean space in Chapter 4.

Rather than working directly in terms of the target density p the HMC algorithm is more naturally described in terms of a *potential energy* function $\phi : \mathbb{R}^D \to \mathbb{R}$ which is related to the target density by

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp(-\phi(\boldsymbol{x})) \iff \phi(\boldsymbol{x}) = -\log p(\boldsymbol{x}) - \log Z. \tag{2.49}$$

The original *target variables* $\mathbf{x} \in \mathbb{R}^D$ are augmented with a vector of *momentum variables* $\mathbf{p} \in \mathbb{R}^D$. The conditional density on the momenta given the target variables $\mathrm{p}_{\mathbf{p}|\mathbf{x}}$ is defined in terms of a *kinetic energy* function $\tau : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ which is even in its first argument

$$\mathrm{p}_{\mathbf{p}|\mathbf{x}}(\boldsymbol{p}|\boldsymbol{x}) \propto \exp(-\tau(\boldsymbol{p} \mid \boldsymbol{x})). \tag{2.50}$$

The joint density on the momentum and target variables is then

$$\mathrm{p}_{\mathbf{x},\mathbf{p}}(\boldsymbol{x},\boldsymbol{p}) \propto \exp(-\phi(\boldsymbol{x}) - \tau(\boldsymbol{x} \,|\, \boldsymbol{p})) = \exp(-h(\boldsymbol{x},\boldsymbol{p})). \tag{2.51}$$

The function $h(\boldsymbol{x},\boldsymbol{p}) = \phi(\boldsymbol{x}) + \tau(\boldsymbol{p} \,|\, \boldsymbol{x})$ is termed the *Hamiltonian* for the system. A common simplification is for the momenta to be defined to be independent of the target variables with a marginal density

$$\mathrm{p}_{\mathbf{p}}(\boldsymbol{p}) \propto \exp(-\tau(\boldsymbol{p})). \tag{2.52}$$

In this case the Hamiltonian $h(\boldsymbol{x},\boldsymbol{p}) = \phi(\boldsymbol{x}) + \tau(\boldsymbol{p})$ is *separable* - there are no terms jointly dependent on both $\boldsymbol{x}$ and $\boldsymbol{p}$.

In classical mechanics, the Hamiltonian describes the total energy of a mechanical system, and can be used to define a *canonical Hamiltonian dynamic* via the set of *ordinary differential equations* (ODEs)

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial h^{\mathsf{T}}}{\partial \boldsymbol{p}}, \qquad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial h^{\mathsf{T}}}{\partial \boldsymbol{x}}. \tag{2.53}$$

We define the *flow map* corresponding to this dynamic as a family of mappings $\boldsymbol{\Psi}_t : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D \times \mathbb{R}^D$ parameterised by a time $t \in \mathbb{R}$ such that if $(\boldsymbol{x}(t), \boldsymbol{p}(t))$ is the solutions to the set of ODEs (2.53) at a time t given an initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0, \boldsymbol{p}(0) = \boldsymbol{p}_0$ then

$$(\boldsymbol{x}(t), \boldsymbol{p}(t)) = \boldsymbol{\Psi}_t(\boldsymbol{x}_0, \boldsymbol{p}_0). \tag{2.54}$$

The Hamiltonian flow map has several desirable properties as a proposal generating mechanism for a MCMC method. The Hamiltonian is exactly conserved along the trajectories generated by the flow map, i.e. $h(\boldsymbol{x},\boldsymbol{p}) = h \circ \boldsymbol{\Psi}_t(\boldsymbol{x},\boldsymbol{p})$ for all $t \in \mathbb{R}$ and for any initial $\boldsymbol{x},\boldsymbol{p}$ pair. As $\mathrm{p}_{\mathbf{x},\mathbf{p}}(\boldsymbol{x},\boldsymbol{p}) \propto \exp(-h(\boldsymbol{x},\boldsymbol{p}))$ this means Hamiltonian trajectories remain confined to constant density surfaces in the augmented state space. The Hamiltonian flow map is also *volume preserving* - the Jacobian of the flow map $\mathbf{J}_{\boldsymbol{\Psi}_t}$ has determinant one for all t and starting from any initial $(\boldsymbol{x},\boldsymbol{p})$. This volume preservation is a consequence of a stronger geometric property of the dynamic - that the flow map is *symplectic* [130]. Symplecity of the flow map is implied by the condition

$$\mathbf{J}_{\boldsymbol{\Psi}_t}^{\mathsf{T}} \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{J}_{\boldsymbol{\Psi}_t} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} \tag{2.55}$$

being satisfied. The symplectic nature of Hamiltonian dynamics have been argued to be key to the performance of HMC in high-dimensional target distributions [27].

A final crucial property of the Hamiltonian flow map is that it exhibits a time-reversal symmetry under negation of the momenta - if $(\boldsymbol{x}', \boldsymbol{p}') = \boldsymbol{\Psi}_t(\boldsymbol{x}, \boldsymbol{p})$ then $(\boldsymbol{x}, -\boldsymbol{p}) = \boldsymbol{\Psi}_t(\boldsymbol{x}', -\boldsymbol{p}')$. If we define $\boldsymbol{F} : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D \times \mathbb{R}^D$ as a 'momentum-flip' operator such that $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{p}) = (\boldsymbol{x}, -\boldsymbol{p})$ then this time reversal symmetry means that the composition $\boldsymbol{F} \circ \boldsymbol{\Psi}_t$ is an *involution*. Further as $\boldsymbol{F}$ also has Jacobian determinant one, then the composition $\boldsymbol{F} \circ \boldsymbol{\Psi}_t$ also itself has a unit Jacobian determinant.

Using the previous result for the Metropolis–Hastings accept ratio for the special case of a deterministic proposal formed by an involution (2.39), we have that a proposal generated by applying $\boldsymbol{F} \circ \boldsymbol{\Psi}_t$ to the current state pair $(\boldsymbol{x}, \boldsymbol{p})$ for any t has an accept probability

$$\alpha(\boldsymbol{x}, \boldsymbol{p}) = \min\left\{1, \frac{\exp(-h \circ \boldsymbol{F} \circ \boldsymbol{\Psi}_t(\boldsymbol{x}, \boldsymbol{p}))}{\exp(-h(\boldsymbol{x}, \boldsymbol{p}))} \left|\mathbf{J}_{\boldsymbol{F} \circ \boldsymbol{\Psi}_t}(\boldsymbol{x}, \boldsymbol{p})\right|\right\} \tag{2.56}$$

$$= \min\{1, \exp(h(\boldsymbol{x}, \boldsymbol{p}) - h \circ \boldsymbol{F} \circ \boldsymbol{\Psi}_t(\boldsymbol{x}, \boldsymbol{p}))\} = 1. \tag{2.57}$$

This results from the Hamiltonian being conserved under the flow map (and momentum flip operator as τ is even in the momenta) and the composed map having unit Jacobian determinant. Therefore we will always accept proposals formed by integrating the ODEs forward by some length of time from the current state and then flipping the momentum. Further on its own the momentum flip operator $\boldsymbol{F}$ also forms an involution with unit Jacobian determinant which exactly conserves the Hamiltonian, and so can also be applied as a 'proposal' on its own with probability of acceptance of one. If we sequentially alternate updates using $\boldsymbol{F} \circ \boldsymbol{\Psi}_t$ and $\boldsymbol{F}$, each defines a valid Markov transition operator which leaves the (extended) target invariant, and in sequential composition the momentum flips cancel. We can therefore construct a Markov chain which leaves the target distribution with density (2.51) invariant by repeatedly generating new states by integrating the Hamiltonian dynamic forward by arbitary lengths of time.

There are two major problems with this scheme. Firstly for most ϕ and τ we will not be able to integrate the ODEs (2.53) exactly and so cannot evaluate the exact flow map $\boldsymbol{\Psi}_t$. Secondly the scheme as proposed would not be ergodic as the Hamiltonian is conserved by each applica-

tion of the flow map, and so all states generated in this way would be confined to a single constant Hamiltonian manifold in the joint target–momentum space.

The first issue can be resolved by instead approximately integrating the ODEs. Importantly by using a *symplectic integrator* we are able to form approximate Hamiltonian flow maps which maintain the key volume-preservation and time-reversibility properties of the exact flow map dynamic (and in fact, as the name suggests, are also symplectic). There is a large glass of such symplectic or geometric integrators [130] however for separable Hamiltonians a particularly popular and easily implementable scheme is the *Störmer–Verlet* or *leapfrog* integrator. A single leapfrog step of time step δt from a current state pair $(\boldsymbol{x}^{(n)}, \boldsymbol{p}^{(n)})$, corresponds to running the following updates

$$\boldsymbol{p}^{(n+1/2)} \leftarrow \boldsymbol{p}^{(n)} - \frac{\delta t}{2} \nabla \phi(\boldsymbol{x}^{(n)}) \tag{2.58}$$

$$\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}^{(n)} + \delta t \nabla \tau(\boldsymbol{p}^{(n+1/2)}) \tag{2.59}$$

$$\boldsymbol{p}^{(n+1)} \leftarrow \boldsymbol{p}^{(n+1/2)} - \frac{\delta t}{2} \nabla \phi(\boldsymbol{x}^{(n+1)}) \tag{2.60}$$

2.3.3 Simulated tempering

2.4 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 independent 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 *graphics processing units* (GPUs) and *central processing unit* (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 estimated 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 (2.15); 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 offers 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 [90] and JAGS [179] having supported their application to a wide range of models. Although requiring that we are able to decompose the target distribution into complete condition-

als 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 complexity may explain in part the seemingly less widespread use of slice sampling methods in practice[6], 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 [156] 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.

[To do]

6 As a very rough metric at the time of writing the original 1953 Metropolis et al. publication [146] has 35,288 citations recorded on Google Scholar, the 1984 Geman and Geman publication [84] commonly cited as the original sourcce of the Gibbs sampling algorithm 20,485 citations and the 2003 Neal [163] slice sampling publication, 1,444 citations.

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, 227] or *auxiliary variable* [69, 112] 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, 209]. 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 [52, 188].

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-

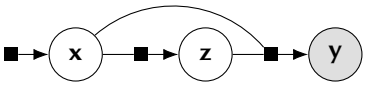


Figure 3.1.: Hierarchical model factor graph.

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

One application of pseudo-marginal methods 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 the unnormalised target density is 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 $\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}$, 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

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

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 tradeoff of an increased computational cost of each density estimate evaluation. The estimator variance will also be 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 $\mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}) \approx \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{\mu}_{\boldsymbol{x},\boldsymbol{y}}, \boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}\right)$ 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 [73].

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 [63, 97] for the marginal density of the observed state sequence given the model parameters [4, 48, 178], parameter inference in 'doubly-intractable' distributions [157] where an intractable normaliser depends on the variables of interest using density estimators based on exact sampling methods [151, 154, 185] and random series trunctation [135] and approximate inference in simulator models where the density on the simulator outputs is only implicitly defined [140].

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 [158]. 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 7. 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 [15] as the original

Algorithm 7 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 [117] and a MCMC method for doubly intractable distributions by Moller et al. [151].

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 a 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 7 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, 73, 209].

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 overall efficiency.

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 [36, 208] 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 [208] 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.

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 [208]. The result of [208] that using $N = 1$ is close to optimal (with N now the number of particles) is therefore not applicable in this case.

Under an alternative simplifying assumption relevant to the particle filtering setting 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 [64] that N should be chosen so as to make the standard deviation of the logarithm of the density estimator approximately equal to 1.2. In [209] a more specific case is considered of pseudo-marginal Metropolis–Hastings methods using a isotropic Gaussian random-walk Metropolis proposal $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$ and the same assumptions as [64] made 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 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 [64].

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 $\rho = \frac{\partial R}{\partial v}$ such that if $\boldsymbol{u}$ is an independent sample from R, then $\hat{p} = \varepsilon(\boldsymbol{x}, \boldsymbol{u})$ is an independent sample from $\mathrm{P}_{\hat{\mathrm{p}}|\mathbf{x}}(\cdot \mid \boldsymbol{x})$. Here R represents the known distribution of the auxiliary variables and ε the operations performed by the remain-

ing 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 $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{0}, \boldsymbol{I})$ and

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

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 7 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 7 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 7 leaves a distribu-

tion with density corresponding to (3.8) invariant is therefore sufficient for ensuring the correctness of the algorithm.

The transition operator corresponding to Algorithm 7 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 7 leaves the target distribution invariant.

We can equivalently consider Algorithm 7 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,

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

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

Algorithm 8 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 9 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 $(\boldsymbol{x}_{n+1}, \boldsymbol{u}_{n+1})$

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 [159]. 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 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 frame-

work 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 8.

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 9. Following the convention of [158] 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* (PM) 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 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 effect-

ive samples per computation time) when making perturbative moves in high-dimensions [79]. 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, 100].

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 [209]. 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 signal for adaptive algorithms to tune the step-size by, potentially requiring slow adaptation 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 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 [208] apply and a choice of $N = 1$ close to optimal, the additional signal provided by the accept rate of the

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.

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 can 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 to decide acceptances and so the accept probability of 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

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

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

$$\begin{aligned} &\pi\left(\boldsymbol{u}^{(m)} \mid \boldsymbol{x}, \{\boldsymbol{u}^{(n)}\}_{n \neq m}\right) \propto \\ &\qquad \mathrm{p}_{\mathbf{z}|\mathbf{x},\mathbf{y}}(\boldsymbol{L}\boldsymbol{u}^{(m)} + \boldsymbol{\mu} \mid \boldsymbol{x}, \boldsymbol{y}) + w\left(\boldsymbol{x}, \{\boldsymbol{u}^{(n)}\}_{n \neq m}\right) \mathcal{N}\left(\boldsymbol{u}^{(m)} \mid \mathbf{0}, \mathbf{I}\right) \end{aligned} \tag{3.13}$$

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

The sum of the importance weights in w 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 $(N-1)|\boldsymbol{L}|$) 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 $\mathcal{N}\left(\boldsymbol{u}^{(m)} \mid \mathbf{0}, \mathbf{I}\right)$ and independence from $\boldsymbol{x}$. Therefore as we increase N we would expect independently re-sampling the auxiliary variables from R in a MI step to have an increasing probability of acceptance.

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 $\{\boldsymbol{u}^{(n)}\}_{n=1}^{N}$, 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 [172] 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 could work well, it adds further tuning burden to the user which might outweigh any efficiency gains. For problems

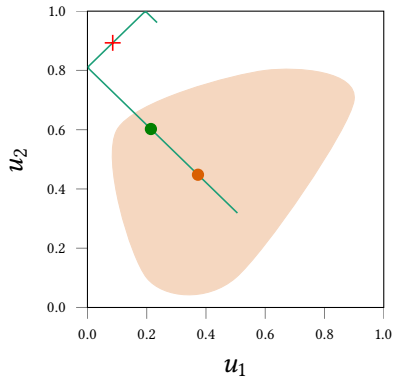


Figure 3.2.: Illustration of reflective linear slice sampling in two dimensions. The orange circular marker represents the current state and the light filled orange region the density slice at the sampled slice height (see explanation of Algorithm 4 in Chapter 2 for details). A random slice line direction 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).

in which we can reparametrise the density estimator as a deterministic function of a vector of standard normal draws so that $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$, an appealing option is to use elliptical slice sampling (Algorithm 5) to update the auxiliary variables. 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 [156] 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 also however able to exponentially back-off to smaller proposed moves around the current state if the bold initial proposal is not 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[3] i.e. $\rho(\boldsymbol{u}) = \prod_i \mathcal{U}(u_i \mid 0, 1)$, 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.

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 [163] and [65].

3.6 NUMERICAL EXPERIMENTS

We will now discuss the results of two 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 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.

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 $\mathsf{P}_{\mathbf{z}|\mathbf{x}}$ 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

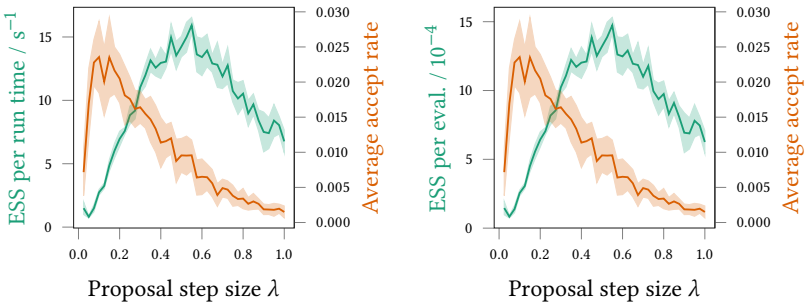
$$\varepsilon(\boldsymbol{x}, \boldsymbol{u}) = \frac{\mathcal{N}(\boldsymbol{x} \mid \mathbf{0}, \mathbf{I})}{N} \sum_{n=1}^{N} \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{y}^{(m)} \mid \sigma \boldsymbol{u}^{(n,m)} + \boldsymbol{x}, \epsilon^2 \mathbf{I}\right), \tag{3.17}$$

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

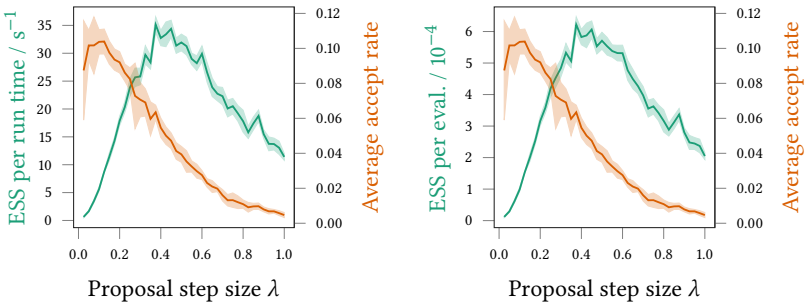
3.6.1.1 *Pseudo-marginal Metropolis–Hastings*

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 50 000 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 $\boldsymbol{y}$ having standard deviation 3.6 for $N = 1$, 1.8 for $N = 8$ and 1.2 for $N = 32$.

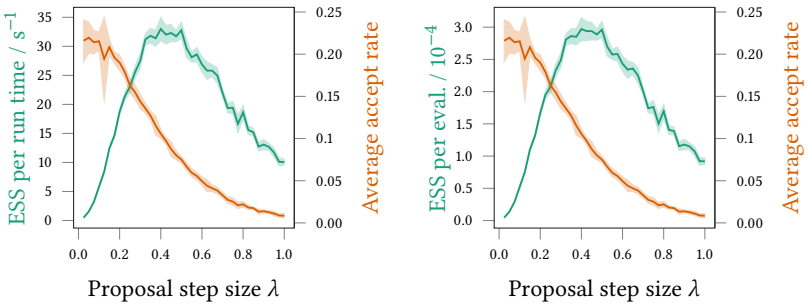
For all combinations of N and λ we estimated the *effective sample size* (as defined for a geometrically ergodic Markov chain in Equation 2.25 of Chapter 2) for the posterior mean of each chain using the `R CODA` package [180]. We then derived two overall measures of computational efficiency from these 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 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



(a) $N = 1$



(b) $N = 8$



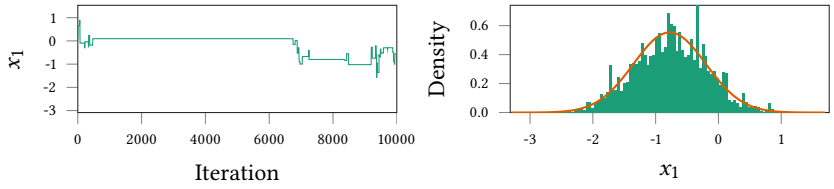
(c) $N = 32$

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.

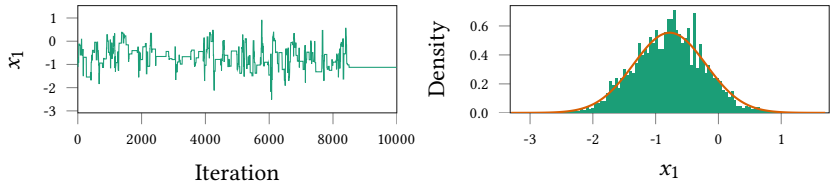
the plots the standard error was too small to be easily visible). On each axis as well as the normalised effective sample size, the average accept rate across the chains is also plotted in orange (with scale shown on the right vertical axis), with again the curves showing the mean value across the chains and the filled regions plus and minus one standard deviation.

The results of [208] suggest that asymptotically using $N = 1$ importance sample should be optimal in this case assuming a linear increase in the cost of generating each sample with N. The measure of computational efficiency used in [208] therefore 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.

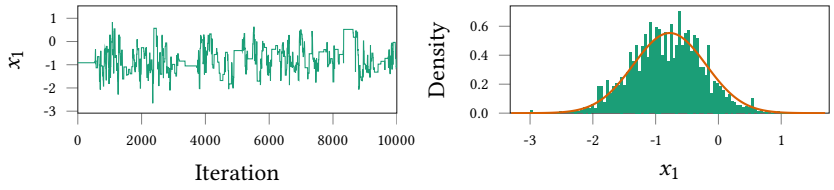
The plots in Figure 3.3a and to a lesser extent 3.3b show a spurious appearing behaviour for the smallest step sizes that the accept rate (orange curve) seems to initially *increase* as the step size is made larger, contrary to what we would reasonably expect. This anomaly can be ascribed to a lack of 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 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.



(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.

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 10 000 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 computational cost per update with the number of importance samples due to both overhead from the implementation of the rest of the operations in the transition and only partial utilisation of the parallel compute resource available (the CPU used in the experiments had 4 cores). Although the increase in efficiency per actual run time for $N \neq 1$ is 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.

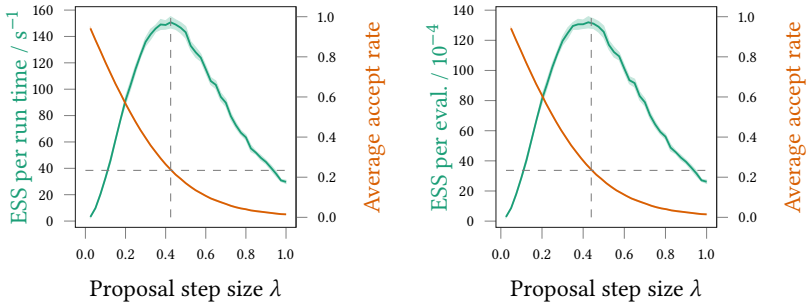
The results in Figure 3.3 also highlight the difficulty of tuning the proposal step size when using a random-walk Metropolis PM MH update. The optimal step size appears to possibly weakly depend on the number of importance samples used (though the noisiness of the curves make this difficult to determine). Further there is not a clear relationship between the average accept rate and optimal step size. As previously stated the result of [79] 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 smal-

lest 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.

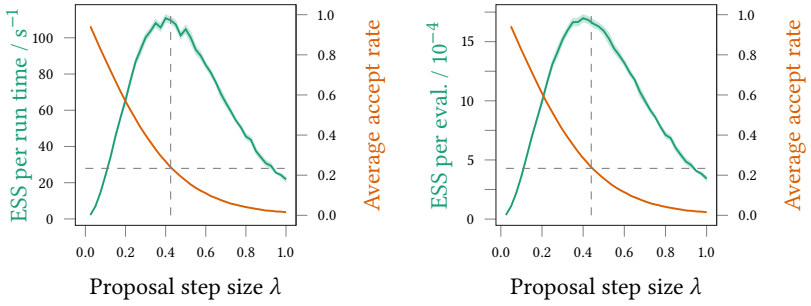
3.6.1.2 *Splitting the update*

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.

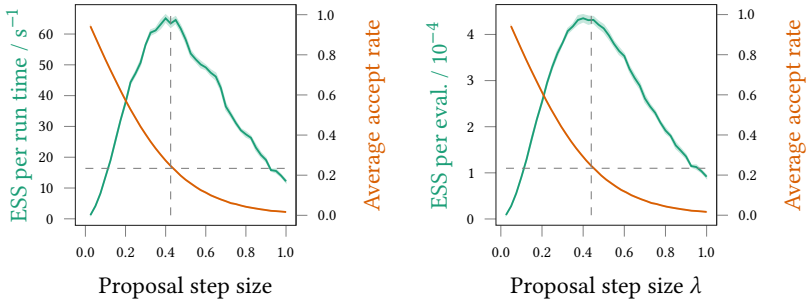
Figure 3.5 shows plots of results of an equivalent series of experiments as used to produce Figure 3.3. In this case the horizontal axes on the plots shows the proposal step size for the MH updates to the target variables which as previously use an random-walk Metropolis proposal $r(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}' \mid \boldsymbol{x}, \lambda^2 \mathbf{I}\right)$. Again 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 20 000 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 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 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



(a) $N = 1$



(b) $N = 8$



(c) $N = 32$

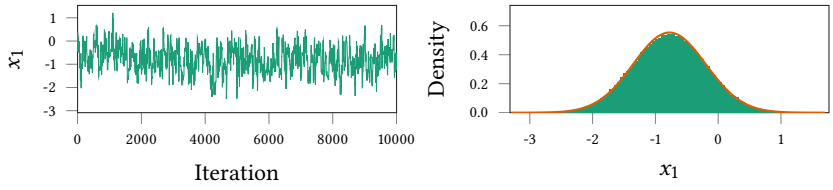
Figure 3.5.: Results of Gaussian latent variable model APM MI+MH chains. The plots in each row show both the estimated effective sample size (ESS) normalised by either the total compute time (green, left column) or number of density estimator evaluations (green, right column) and average acceptance rate 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.

an average accept rate of 0.23 for chains using $N = 32$ importance samples.

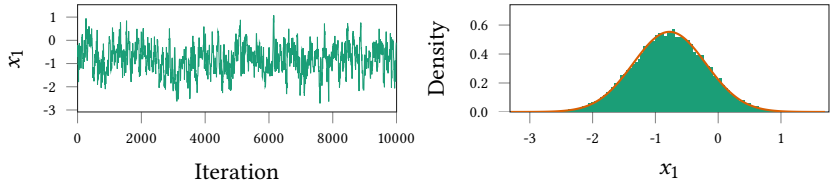
On both the time and density evaluation normalised measures of efficiency the APM MI+MH chains perform significantly better than the PM MH chains. The peak effective sample size per density evaluation value for the $N = 1$ and $\lambda = 0.425$ case is around a factor of ten higher than the corresponding 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 [79] 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.

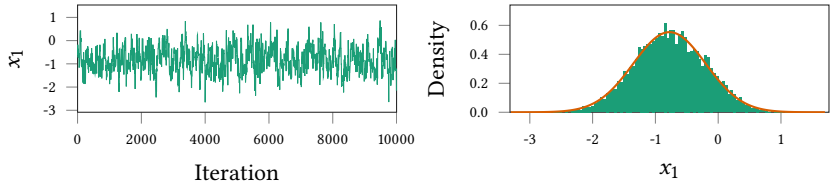
In further contrast to the PM MH results, the results for the APM MI+MH chains seem to unambiguously support using $N = 1$ importance sample. On both the computation time and density evaluation normalised measures of efficiency, the chains using one importance sample dominate over the $N = 8$ and $N = 32$ 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



(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 10 000 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.

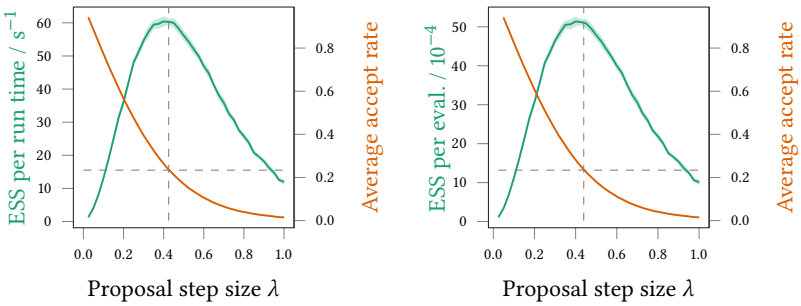


Figure 3.7.: Results of Gaussian latent variable model APM SS+MH chains (using $N = 1$ importance sample). The plots in each row show both the estimated effective sample size (ESS) normalised by either the total compute time (green, left column) or number of density estimator evaluations (green, right 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.

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 10 000 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 normalise 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 was reduced accordingly). This again suggests that 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*

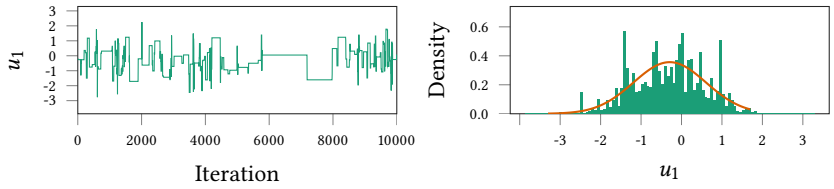
For the APM MI+MH chains discussed in the previous subsection, when using $N = 1$ importance sample the MI updates to the auxiliary variables were only accepted 2.5% of the time. Although this did not appear to impede convergence of the chain in this example, more generally low accept rates for the MI 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, 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, there is a diminishing returns behaviour to the increase of acceptance rate with the number of samples.

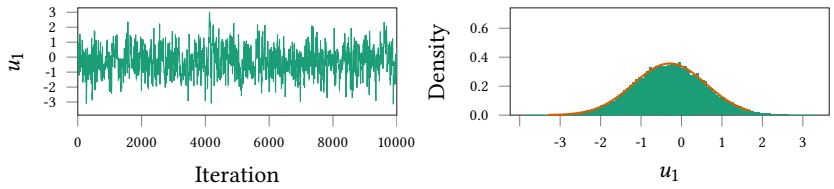
The earlier suggestion to use perturbative updates to the auxiliary variables provides an alternative approach to improve the auxiliary variable mixing. We test specifically here the proposal to use elliptical slice sampling updates to the auxiliary variables, which is a natural choice in this case due 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 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 their true posterior distribution (which again can be found analytically) compared to when using MI updates. Figure 3.8 shows traces and histograms of one of the auxiliary vari-

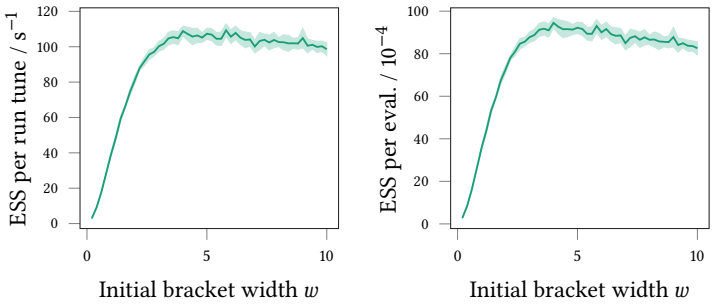


(a) APM MI+MH $N = 1, \lambda = 0.425$

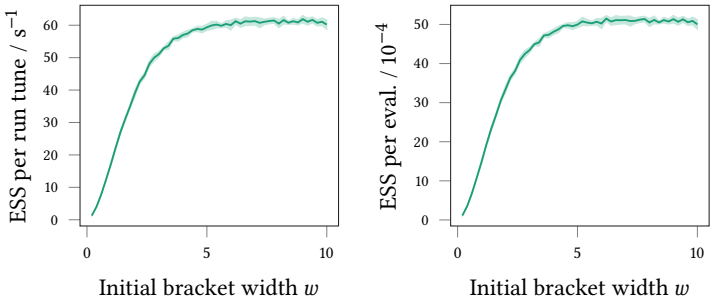


(b) APM SS+MH $N = 1, \lambda = 0.425$

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.



(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.

ables 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 of 20 000 iterations each. 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 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.

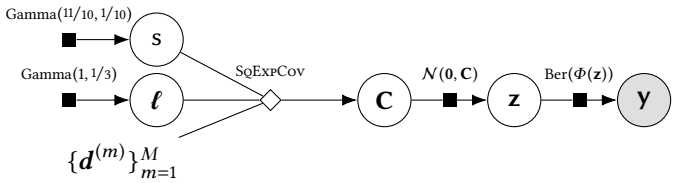


Figure 3.10.: Gaussian process probit regression factor graph.

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 explains 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 because of 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 Gaussian 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 [73] and shown to give signficant improvements over competing MCMC methods.

As an example data set we used the Wisconsin breast cancer prediction data set [137] from the UCI machine learning dataset repository [134] as also used for experiments in [73]. 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 [189] 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 $\boldsymbol{\ell} \in \mathbb{R}^K_{>0}$ 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 $i \in \{1 \dots M\}$ **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 [189]. 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 $\mathbf{z} = \boldsymbol{z}$ the binary outputs are modelled as having a Bernoulli distribution $\mathrm{Ber}(\Phi(\boldsymbol{z}))$ where Φ is the standard normal CDF function. Following [73] 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. $\mathbf{x} = [\log \mathrm{s}; \log \boldsymbol{\ell}]$ 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 $\mathrm{p}_{\mathbf{x}}$

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

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

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

One option would be to construct a Markov chain on the joint $(\mathbf{x}, \mathbf{z})$ space with an unnormalised target density $\mathrm{p}_{\mathbf{x},\mathbf{y},\mathbf{z}}$, 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 [73]. As an alternative [73] 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 $\mathrm{p}_{\mathbf{z}|\mathbf{x}}$, 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 [73] was that much lower variance density estimates can be computed 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 [73] both Laplace's method and *expectation propagation* (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 find the mode of the latent posterior for this probit regression Gaussian process model [189, §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 were 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. As $M = 682$ 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 $\boldsymbol{L}_{\boldsymbol{x},\boldsymbol{y}} = \text{chol}\,\boldsymbol{\Sigma}_{\boldsymbol{x},\boldsymbol{y}}$ 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 $\boldsymbol{u} = \left[\boldsymbol{u}^{(1)}; \boldsymbol{u}^{(2)} \ldots \boldsymbol{u}^{(N)}\right]$ 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 [208] that a choice of $N = 1$ is close to optimal does not apply here. In experiments in [208] 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 [73] 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 using $N \geq M$, 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 [73] $N = 1$ 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 perform 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 [73], 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 [73] 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

4 Although in the published version of [73] 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 in the experiments so we follow that instead.

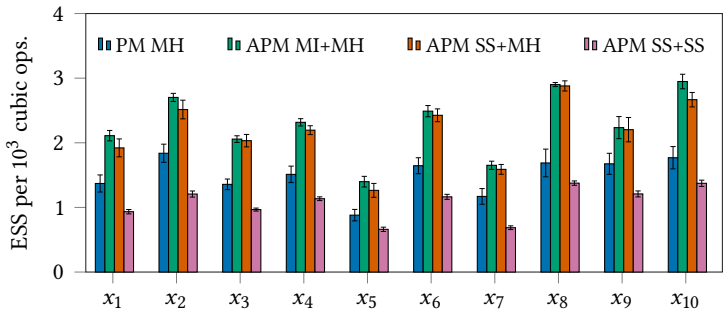
rate of 0.234 for the MH updates to the target variables in APM methods can be justified theoretically and empirically, it is not clear what the optimal choice is for PM MH updates, with this seeming to be dependent on the estimator variance and so number of importance samples N. While the $[0.15, 0.3]$ 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 [73] 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 runs. For 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.

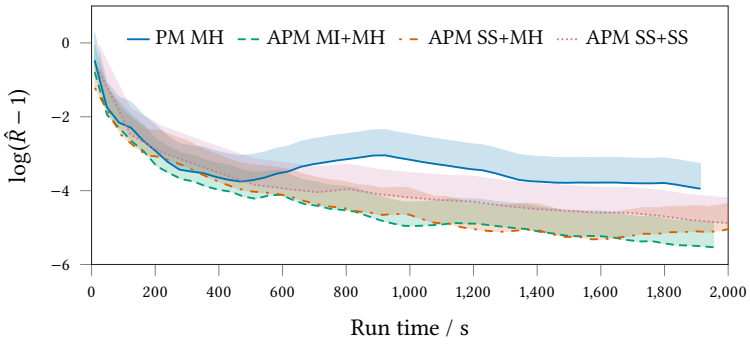
Results of the experiments are summarised in Figure 3.11. As a first measure of performance we consider the relative estimated sampling efficiency of the different methods. For each of the 10 target variables we estimated the effective sample size for the estimated mean of the variable using `R CODA` [180] and normalised these values by the total number of cubic operations performed in each chain[5]. The means of these values across the 10 chains per method (and standard errors) 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 opera-

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.



(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.



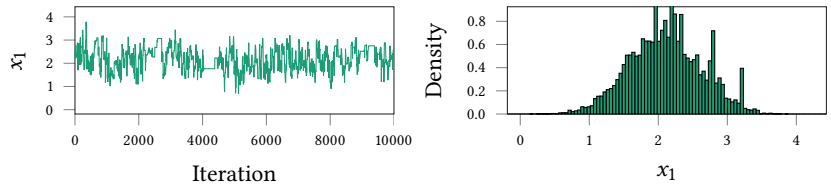
(b) Chain convergence. Plots of PSRF $\hat{R}$ statistic on a log scale computed across 10 independent chains initialised from the prior for increasing number of chain iterations (normalised by mean total run time for each method to adjust for different per iteration run times) for each of four transition 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.

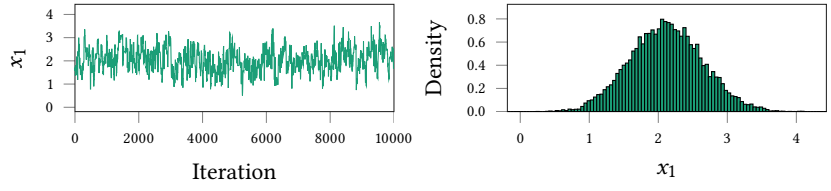
tions (providing the Cholesky factorisations of the Gaussian process prior covariance and importance distribution covariance at the current target variable values are cached from the target variable update), there is little effect on the overall run time from using elliptical SS updates to the auxiliary 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 SS updates to the auxiliary variables in this case as the MI updates are likely to be mixing the auxiliary variables sufficiently well.

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 [83], also often termed 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 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` 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 darker coloured curves show the median of the estimated $\hat{R}$ interval and the lighter 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 are 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.

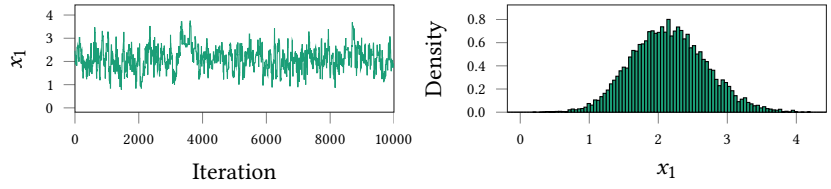
On this measure of performance the PM MH chains seem to perform more poorly, showing a slower convergence rate than the other methods, inclduing the APM SS+SS chains. The non-montonically decreasing behaviour seen in the $\hat{R}$ curve for the PM MH chains seems to be the result of the chains suffering 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 chains run using $N = 500$ importance samples (traces not shown here) also showed sticking behaviour



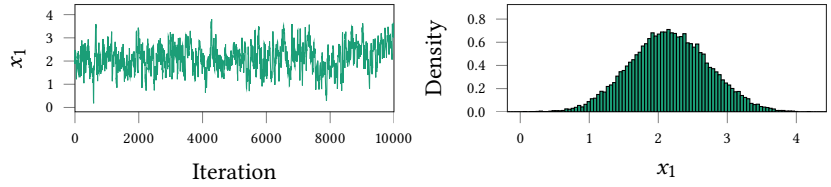
(a) PM MH



(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.

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 not shown.

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 each chain were 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, it seems reasonable to assume that the spurious peaks in the histogram for the PM MH chains are not a reflection of the true marginal density but instead a result 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 even with the run time adjustment 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 standing of the APM SS+SS chains on the effective sample size measure of performance is not an entirely accurate potrayal of the overall performance of the method.

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.

The simplest proposed approach of splitting the combined proposed update to both auxiliary and target variables in the standard PM MH algorithm into a separate Metropolis independence updates to the auxiliary variables and Metropolis–Hastings update to the target variables (APM MI+MH) involves changing only a few lines of code in most implementations and adds no further free parameters to tune. Despite involving only a minor change, in the empirical studies performed this adjusted update was found to give significantly better computational

cost normalised sampling efficiency over the standard pseudo-marginal Metropolis–Hastings update, despite in some cases doubling the computational effort per overall chain update. A simple intuition for understanding this improved performance is that for a fixed proposal distribution for the target variables, the accept rate of the MH update to the target variables in the APM MI+MH chains was typically more than double the corresponding accept rate for the overall PM MH update. Therefore the doubling of the number of density estimates needed per iteration was more than outweighed by more than double the number of proposed target variable updates *from the same proposal distribution* (e.g. same Gaussian random-walk step-size) being accepted

The size of the increase in the accept rates for a fixed proposal distribution is dependent on how high the variance of the density estimator is or equivalently how dependent the target and auxiliary variables are under the auxiliary joint target. For high variances cases e.g. when using $N = 1$ importance sample, the increase in accept rates for the target variable updates in APM MI+MH chains over the accept rate of the PM MH updates is higher due to poor performance of making independent proposed updates to the auxiliary variables in the PM MH having a strong deleterious effect on the PM MH accept rate. For example in the Gaussian latent variable model experiments when using $N = 1$ importance sample the accept rate of the MH updates in the APM MI+MH chains was typically around a factor of 20 higher than the accept rate for the corresponding PM MH chains using the same proposal step size. As the variance of the density estimator is decreased by increasing N, the difference in the accept rates for a fixed proposal step size becomes less marked with around a factor five difference for $N = 8$ and around a factor two difference for $N = 2$ between APM MI+MH and PM MH. So with a lower variance estimator the difference in performance between APM MI+MH and PM MH becomes less marked.

However the recommendation of [208] suggests that when the computational cost of each PM MH update scales linearly with N (and when using a density estimator formed as an average of unbiased Monte Carlo estimates) that using $N = 1$ is close to optimal for PM MH despite the higher estimator variance. As the low N, high density estimator variance cases are precisely when we expect to see the largest potential gains from using APM MI+MH over PM MH this suggests when this linear cost scaling argument is valid there will often be a computational

gain from using APM MI+MH. In some cases as we saw in the Gaussian process experiments we can form a much lower variance density estimate by expending some computational effort to fit a good importance distribution. In these cases due to the additional overhead of the fitting procedure the linear cost scaling argument no longer applies. Further the density estimates in this case may be sufficiently low variance for there to be little improvement in accept rates of updates to the target variables by splitting the PM MH in to separate MI and MH updates. However typically in these cases the overhead introduced by separately updating the auxiliary variables in an MI step will also be much less than the cost of the original PM MH update, as for fixed values of the target variables the importance distribution does not need to be refitted and any target variable dependent computations such as Cholesky factorisations of covariance matrices can be cached and reused. Therefore the overall cost per APM MI+MH update will be very close to that of each PM MH update and so even a small improvement in accept rate of the target variable updates can be worthwhile splitting the update.

Perhaps more important than the sampling efficiency gains seen from using APM MI+MH over PM MH in the experiments here was the significantly improved ability to tune the MH updates in the algorithm even when using a high-variance density estimator. By decoupling the dependency of the MH accept rate from the density estimator variance, theoretical guidelines for choosing a proposal step-size (or adapting other algorithm parameters) based on the average accept rate can be straightforwardly applied to tune APM MI+MH updates. The resulting increased ease of use of the algorithm and decreased requirement for user intervention to get good performance might often make APM MI+MH an attractive choice even when the extra runtime overhead per update negates any sampling efficiency gains. Further the separate MI step accept rate of the APM MI+MH update provide a diagnostic already computed as part of the chain updates which can alert users to issues with poor mixing of the auxiliary variables due to low accept rates of the auxiliary updates. In contrast it will not always be clear if a poor accept rate of a PM MH chain is due to poor choice of the target variables proposal distribution or due to a high density estimator variance, and separately monitoring the density estimator variance as part of the update adds overhead while not being as directly interpretable as the MI step accept rate.

If initial runs using an APM MI+MH method do show a very low accept rate for the updates to the auxiliary variables which might lead to convergence issues, the proposed APM SS+MH approaches offer a simple 'plug-in' solution to improve mixing of the auxiliary variables without having to tune a separate proposal distribution for a MH update to the auxiliary variables. If the auxiliary variables can be naturally represented as being marginally distributed according to the standard normal distribution, then elliptical slice sampling is a straightforward choice, having no free parameters to tune and still initially proposing bold moves to near independent points in the auxiliary space while able to back-off to more conservative updates to ensure a non-zero move to the auxiliary variables under weak smoothness conditions.

Another common case is auxiliary variables which are naturally parameterised as a vector of standard uniform draws, in which case reflective linear slice sampling offers analagous benefits. Although the linear slice sampling algorithm does have a free initial bracket width parameter to be chosen, in general (as seen in the experiments using this algorithm for updates to the target variables in the Gaussian latent variable model experiments) the efficiency of the algorithm is not strongly dependent on the choice of this parameter providing it is set large enough to cover most of the intersection of the slice with the sampled line as the exponential shrinking of the bracket on proposing an off slice point will quickly reduce the bracket to a more appropriate size if set initially too large. For reflective slice sampling in the unit hypercube a fixed initial bracket width of one and a direction vector $\boldsymbol{v}$ sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ was found to work well in experiments applying APM SS+MH methods to inference in a doubly-intractable Ising model problem in [158].

Independently of and concurrently with the original conference publication [158] related to this work, both Dahlin et al. [53] and Deligiannidis et al. [58] considered related frameworks in which the auxiliary random variables of a pseudo-marginal density estimator are updated using a Metropolis–Hastings update leaving the distribution defined by the density (3.8) on the joint auxiliary–target variable space invariant. Both assume a parameterisation in which the auxiliary variables have an isotropic standard normal marginal distribution $\rho(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u} \mid \mathbf{0}, \mathbf{I})$,

and consider a Metropolis–Hastings update to the auxiliary variables with proposal density

$$r^*(\boldsymbol{u}' \mid \boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{u}' \mid \sqrt{1-\lambda^2}\boldsymbol{u}, \lambda^2 \mathbf{I}\right) \tag{3.20}$$

which can be variously considered as a discretisation of a Ornstein-Uhlenbeck diffusion process, exact update of an AR(1) model or as an fixed step size update on an elliptical path that the elliptical slice sampling algorithm 5 generalises by adaptively setting the step size λ. This fixed step-size Metropolis–Hastings update is more amenable to analysis, with both [53] and [58] giving much more extensive theoretical justifications for using perturbative updates to the auxiliary variables (or equivalents introducing correlations in between the auxiliary variable samples) than the mainly intuition based and empirical arguments made here. These theoretical insights are important for informing future development of these ideas. In practical settings however, though the above MH update with an optimally tuned choice of λ may give better sampling efficiency performance compared to the elliptical slice sampling updates proposed here, we would suggest that the additional tuning burden placed on the user and loss of robustness in cases where the appropriate step size varies across the state space, would suggest that elliptical slice sampling updates to the auxiliary variables are still often a good default choice.

The empirical evidence for using slice sampling updates to the target variables as in the proposed APM MI+SS and APM SS+SS methods is less strong, with in both of the models considered in the experiments here these methods having poorer run-time adjusted efficiency than the well tuned APM MI+MH and APM SS+MH methods respectively. If adapting an existing PM MH algorithm where some effort has already been extended to identify an appropriate proposal distribution for updates to the target variables or other information is available to inform this choice, the additional overhead of the slice sampling updates might not be worthwhile. In cases however where we have less prior knowledge about appropriate scales for updates to the target variables or are more concerned with overall robustness and ease of use, slice sampling updates to the target variables are likely to be more attractive however.

4 IMPLICIT GENERATIVE MODELS

In the approximate inference methods considered in Chapter 2 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* [16, 62, 98] - 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) [109, 224] 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 [89] 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 [67, 95, 133], with there being significant gains in modelling flexibility by dropping the requirement to be able to compute an explicit density function [150, 222].

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 [98] and *approximate Bayesian computation* (ABC) [16].

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.

The contributions described in this chapter have previously appeared in the published conference paper

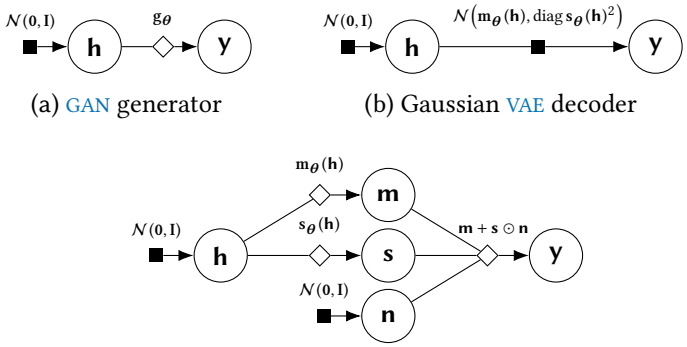
- Asymptotically exact inference in differentiable generative models. Matthew M. Graham and Amos J. Storkey. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR 54:499-508*, 2017.

I was the primary author of that work and contributed the main novel ideas, as well as performing and analysing the numerical experiments described in the paper, some of which are reproduced in this chapter in Section 4.11.

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) [95] and *variational autoencoders* (VAEs) [119, 192]. Although

1 We will follow the suggestion of [234] 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 $\mathbf{n}$.

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 [94]. 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.

GANs [95] 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 [119] or *deep latent Gaussian model* [192]. 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 in-

ference 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 $p_{\mathbf{h}|\mathbf{y}}$ 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 $\mathbf{y} = \mathbf{m}_{\boldsymbol{\theta}}(\mathbf{h}) + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{h}) \odot \mathbf{n}$ 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 $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{h}, \mathbf{n}) = \mathbf{m}_{\boldsymbol{\theta}}(\mathbf{h}) + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{h}) \odot \mathbf{n}$ 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.

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 $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x}] : X \to F$ 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 eval-

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.

uation 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 $\mathrm{P}_{\mathbf{x}}$ will not be absolutely continuous with respect to the Lebesgue measure. Instead $\mathrm{P}_{\mathbf{x}}$ 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 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 [14] 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* [108] can generate code for computing the function's derivatives. By applying the reverse-mode accumulation AD (Algorithm 14) 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

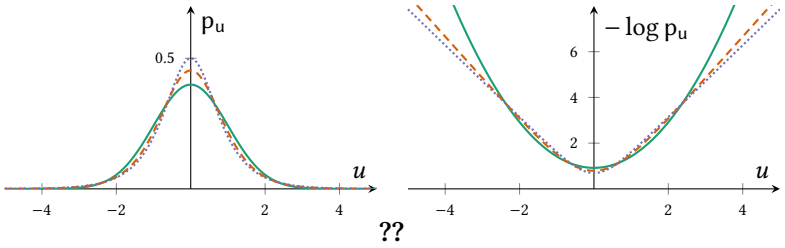


Figure 4.2.: Unit variance densities with unbounded support.

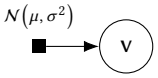
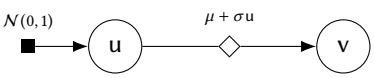
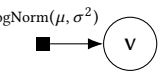
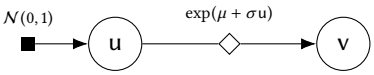
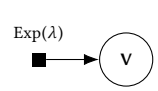
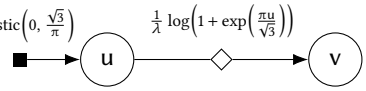
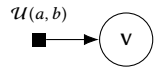
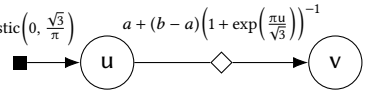
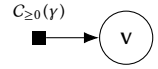
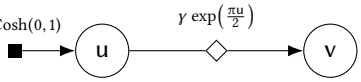
Original factor	Reparametrisation
$\mathcal{N}(\mu, \sigma^2)$ 	$\mathcal{N}(0,1)$ $\mu + \sigma u$ 
$\text{LogNorm}(\mu, \sigma^2)$ 	$\mathcal{N}(0,1)$ $\exp(\mu + \sigma u)$ 
$\text{Exp}(\lambda)$ 	$\text{Logistic}\left(0, \frac{\sqrt{3}}{\pi}\right)$ $\frac{1}{\lambda}\log\left(1+\exp\left(\frac{\pi u}{\sqrt{3}}\right)\right)$ 
$\mathcal{U}(a, b)$ 	$\text{Logistic}\left(0, \frac{\sqrt{3}}{\pi}\right)$ $a + (b-a)\left(1+\exp\left(\frac{\pi u}{\sqrt{3}}\right)\right)^{-1}$ 
$C_{\geq 0}(\gamma)$ 	$\text{InvCosh}(0,1)$ $\gamma \exp\left(\frac{\pi u}{2}\right)$ 

Table 4.1.: Reparameterisations of random variables with some common parametric distributions as deterministic transformations of unit-variance unbounded support random variables.

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 $\mathbf{z}$.

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 [81] and PyMC3 make use of a range of such transformations within their MCMC implementations [206].

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 InvCosh$(0, 1)$ and the logistic distribution 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

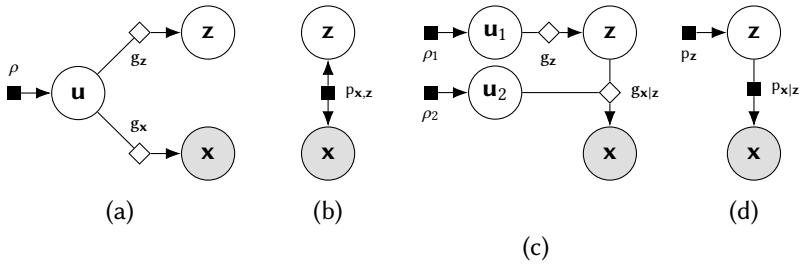


Figure 4.3.: Factor graphs of undirected and directed generative models. Panel (a) shows the more general undirected model case in which observed variables $\mathbf{x}$ and latent variables $\mathbf{z}$ are jointly generated from random inputs $\mathbf{u}$, with (b) showing equivalent factor graph after 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 $\mathbf{u}_2$.

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* [34, 173, 183] 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 literature to describe this idea [119]. 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 [26, 172, 173].

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 $\forall A \in \mathscr{B}(X)$

$$\begin{aligned} \lim_{\epsilon \to 0} \mathrm{P}_{\mathbf{y}}(A) &= \lim_{\epsilon \to 0} \int_X \mathrm{P}_{\mathbf{y}|\mathbf{x}}(A \mid \boldsymbol{x}) \, \mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \lim_{\epsilon \to 0} \int_X \int_X \mathbb{1}_A(\boldsymbol{y}) \, k_\epsilon(\boldsymbol{y}; \boldsymbol{x}) \, \mathrm{d}\boldsymbol{y} \, \mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \int_X \mathbb{1}_A(\boldsymbol{x}) \, \mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \mathrm{P}_{\mathbf{x}}(A), \end{aligned} \tag{4.6}$$

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

$$\mathrm{P}_{\mathbf{y}}(A) = \int_X \mathrm{P}_{\mathbf{y}|\mathbf{x}}(A \mid \boldsymbol{x}) \, \mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) = \int_A \int_X k_\epsilon(\boldsymbol{y}; \boldsymbol{x}) \, \mathrm{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \, \mathrm{d}\boldsymbol{y}. \tag{4.9}$$

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

$$\begin{aligned} \mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) &= \int_X k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x}}(\mathrm{d}\boldsymbol{x}) \\ &= \int_{X\times Z} k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \qquad \forall \boldsymbol{y} \in X. \end{aligned} \tag{4.10}$$

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

$$\begin{aligned} \int_{A\times Z} f(z)\, \mathsf{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}z) &= \int_{A\times X\times Z} f(z)\, \mathsf{P}_{\mathbf{y},\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \int_{X\times Z}\int_A f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathrm{d}\boldsymbol{y}\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \int_A \int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)\, \mathrm{d}\boldsymbol{y}. \end{aligned} \tag{4.11}$$

Using that $\mathsf{P}_{\mathbf{y}}$ has a density $\mathsf{p}_{\mathbf{y}}$ with respect to the Lebesgue measure, and that we can safely ignore the set for which $\mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) = 0$ when integrating against $\mathsf{P}_{\mathbf{y}}$ as it is zero-measure, we have that $\forall A \in \mathscr{B}(X)$

$$\begin{aligned} &\int_{A\times Z} f(z)\, \mathsf{P}_{\mathbf{y},\mathbf{z}}(\mathrm{d}\boldsymbol{y}, \mathrm{d}z) = \\ &\qquad \int_A \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)\, \mathsf{P}_{\mathbf{y}}(\mathrm{d}\boldsymbol{y}). \end{aligned} \tag{4.12}$$

Comparing this to the definition of the conditional expectation from Chapter 1 (1.30) therefore we have $\forall \boldsymbol{y} \in X : \mathsf{p}_{\mathbf{y}}(\boldsymbol{y}) > 0$

$$\begin{aligned} \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] &= \frac{1}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})} \int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z) \\ &= \frac{\int_{X\times Z} f(z)\, k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)}{\int_{X\times Z} k_\epsilon(\boldsymbol{y};\, \boldsymbol{x})\, \mathsf{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}z)}. \end{aligned} \tag{4.13}$$

For the case of a model in which $\mathsf{P}_{\mathbf{z}}$ has a density $\mathsf{p}_{\mathbf{z}}$ 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 $\mathsf{P}_{\mathbf{z}|\mathbf{y}}(A \mid \mathbf{y}) = \mathbb{E}[\mathbb{1}_A(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ we have that

$$\mathsf{P}_{\mathbf{z}|\mathbf{y}}(A \mid \boldsymbol{y}) = \int_A \frac{\int_X k_\epsilon(\boldsymbol{y};\boldsymbol{x})\, \mathsf{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \mid z)\, \mathsf{p}_{\mathbf{z}}(z)}{\mathsf{p}_{\mathbf{y}}(\boldsymbol{y})}\, \mathrm{d}z. \tag{4.14}$$

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 comput-

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.

ing 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 distribution. 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 $\hat{\mathrm{f}}_{S,\epsilon}$ will be consistent,

$$\lim_{S\to\infty} \mathbb{E}\left[\hat{\mathrm{f}}_{S,\epsilon}\right] = \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]. \tag{4.18}$$

If the kernel used is the uniform ball kernel (4.7), the estimator can be manipulated in to a particularly intuitive form

$$\hat{\mathrm{f}}_{S,\epsilon} = \frac{1}{|A|}\sum_{s\in A}(f(\mathbf{z}_s)) \;\text{ with }\; A = \{s \in \{1\ldots S\} : \|\boldsymbol{y} - \mathbf{x}_s\|_2 < \epsilon\} \tag{4.19}$$

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 [77, 184, 201, 217, 228] , 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

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 $\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ to the proposal distribution $\mathrm{P}_{\mathbf{x},\mathbf{z}}$ 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 [231] as allowing for example samples to be generated from the approximate posterior.

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 [16]. Kernels with bounded support but non-flat densities such as the *Epanechnikov kernel* [70] which has a 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

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.

which introduces significant approximation error. The ABC rejection method is well known to scale poorly with dimensionality due to curse of dimensionality effects [32, 138, 182]. 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 probability distribution on a high-dimensional space. If the proposal distribution ($\mathrm{P}_{\mathbf{x},\mathbf{z}}$ here) is significantly more diffuse than the target distribution ($\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}$ 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 [182]. 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

$$\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{s} = \mathbf{s}(\boldsymbol{y});\, \epsilon\big] = \frac{\int_{X\times Z} f(\boldsymbol{z})\, k_\epsilon(\mathbf{s}(\boldsymbol{y});\, \mathbf{s}(\boldsymbol{x}))\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}{\int_{X\times Z} k_\epsilon(\mathbf{s}(\boldsymbol{y});\, \mathbf{s}(\boldsymbol{x}))\, \mathrm{P}_{\mathbf{x},\mathbf{z}}(\mathrm{d}\boldsymbol{x}, \mathrm{d}\boldsymbol{z})}, \quad (4.20)$$

with now the variable conditioned on the T-valued variable $\mathbf{s}$ with

$$\mathrm{P}_{\mathbf{s}|\mathbf{x}}(A \,|\, \boldsymbol{x}) = \int_A k_\epsilon(\boldsymbol{s}; \mathbf{s}(\boldsymbol{x}))\, \mathrm{d}\boldsymbol{s} \qquad \forall A \in \mathscr{B}(T),\ \boldsymbol{x} \in X. \quad (4.21)$$

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 $\mathbf{x}$ [9]. 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 [182] and [33] 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 [232].

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 [231]. 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 $\mathrm{P}_{\mathbf{y}|\mathbf{x}}$ as representing the measurement noise model. We might also instead view the kernel $\mathrm{P}_{\mathbf{y}|\mathbf{x}}$ as accounting for the mismatch between our proposed model for how the observed values are generated and the true data generating process [190, 231]. 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 prac-

tice however the choice of summary statistics used and size of the ϵ tolerance are typically chosen more on grounds of computational tractability [138, 182, 194]. 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 [140, 210]. 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) [17, 57, 211, 221] 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 [12] and variational methods [153, 223]. 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

$$\mathrm{P}_{\mathbf{x},\mathbf{z}|\mathbf{y}}(A, B \,|\, \boldsymbol{y}) = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_B \int_A \mathrm{p}_{\mathbf{z}}(z)\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x} \,|\, z)\, \mathrm{d}z \qquad (4.22)$$

Algorithm 10 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.

$\boldsymbol{z}^* \sim Q(\cdot \mid \boldsymbol{z})$
$\boldsymbol{x}^* \sim \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\cdot \mid \boldsymbol{z}^*)$
$u \sim \mathcal{U}(0,1)$
$a \leftarrow \frac{q(\boldsymbol{z} \mid \boldsymbol{z}^*)\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z}^*)\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x}^*)}{q(\boldsymbol{z}^* \mid \boldsymbol{z})\, \mathrm{p}_{\mathbf{z}}(\boldsymbol{z})\, k_\epsilon(\boldsymbol{y}; \boldsymbol{x})}$
if $u < a$ **then**
    return $(\boldsymbol{x}^*, \boldsymbol{z}^*)$
else
    return $(\boldsymbol{x}, \boldsymbol{z})$

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 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 [140] 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 10. 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

$$\begin{aligned}
&\mathrm{T}(A, B \mid \boldsymbol{x}, \boldsymbol{z}) = \\
&\quad \int_B \int_A q(\boldsymbol{z}' \mid \boldsymbol{z}) \alpha(\boldsymbol{x}', \boldsymbol{z}' \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x}' \mid \boldsymbol{z}')\, \mathrm{d}\boldsymbol{z}' + \\
&\quad \mathbb{1}_A(\boldsymbol{x})\, \mathbb{1}_B(\boldsymbol{z}) \left(1 - \int_X \int_Y q(\boldsymbol{z}' \mid \boldsymbol{z}) \alpha(\boldsymbol{x}', \boldsymbol{z}' \mid \boldsymbol{x}, \boldsymbol{z})\, \mathrm{P}_{\mathbf{x}|\mathbf{z}}(\mathrm{d}\boldsymbol{x}' \mid \boldsymbol{z}')\, \mathrm{d}\boldsymbol{z}'\right).
\end{aligned}$$

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 [140, 210].

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 [210]. 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 $\mathrm{P}_{\mathbf{x}|\mathbf{z}=\boldsymbol{z}^*,\mathbf{y}=\boldsymbol{y}}$, 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, 16] where such sticking artifacts are also a well known problem [155].

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

$$\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathbf{x})] \tag{4.24}$$

$$= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\mathbb{E}[f(\mathrm{g}_{\mathbf{z}}(\mathbf{u}))\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}}(\mathbf{u}))] \tag{4.25}$$

$$= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})}\int_U f \circ \mathrm{g}_{\mathbf{z}}(\boldsymbol{u})\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}. \tag{4.26}$$

6 This is for concreteness of notation rather than a modelling restriction and the approach can easily be generalised to more general distributions.

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 model with a tractable marginal density on the unobserved variables $\mathrm{p}_{\mathbf{z}}$, using the notation introduced in Section (??) we have that

$$\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{y} = \boldsymbol{y};\, \epsilon\big] = \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \mathbb{E}\big[f(\mathbf{z})\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}|\mathbf{z}}(\mathbf{z},\, \mathbf{u}_2))\big] \tag{4.28}$$

$$= \frac{1}{\mathrm{p}_{\mathbf{y}}(\boldsymbol{y})} \int_Z \int_{U_2} f(z)\, k_\epsilon(\boldsymbol{y};\, \mathrm{g}_{\mathbf{x}|\mathbf{z}}(z,\, \boldsymbol{u}_2))\, \mathrm{p}_{\mathbf{z}}(z) \rho_2(\boldsymbol{u}_2)\, \mathrm{d}\boldsymbol{u}_2\, \mathrm{d}z \tag{4.29}$$

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 $\mathbb{E}\big[f(\mathbf{z}) \,|\, \mathbf{x}\big]$ 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 $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon]$ converges as $\epsilon \to 0$ to the conditional expectation of interest $\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}]$, providing that the implicit distribution of the observed variables in the generative model $\mathsf{P}_{\mathbf{x}}$ is absolutely continuous with respect to the Lebesgue measure with density $\mathsf{p}_{\mathbf{x}}$. 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

$$\mathbb{E}[f(\mathbf{z}) \mid \mathbf{x} = \boldsymbol{y}] = \lim_{\epsilon \to 0} \mathbb{E}[f(\mathbf{z}) \mid \mathbf{y} = \boldsymbol{y};\, \epsilon] \tag{4.31}$$

$$\simeq \frac{\int_U f \circ \mathbf{g}_{\mathbf{z}}(\boldsymbol{u})\, \delta(\boldsymbol{y} - \mathbf{g}_{\mathbf{x}}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}{\int_U \delta(\boldsymbol{y} - \mathbf{g}_{\mathbf{x}}(\boldsymbol{u}))\, \rho(\boldsymbol{u})\, \mathrm{d}\boldsymbol{u}}. \tag{4.32}$$

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 [60] 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 [60] 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* [72, §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 [66, 165] 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 [130].

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 [202] 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 [128].

The use of constrained dynamics in HMC has been proposed several times. In the molecular dynamics literature, both [107] and [131] suggest using a simulated constrained dynamic within a HMC framework to estimate free-energy profiles.

Most relevantly here [38] 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

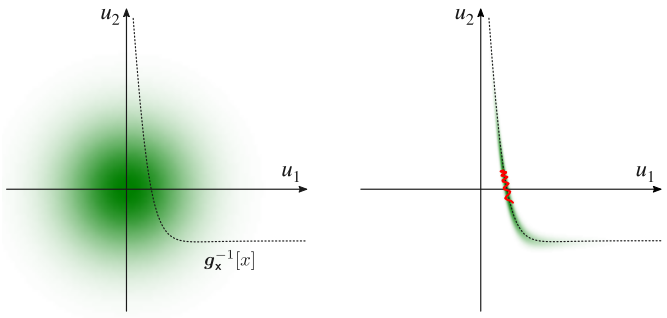


Figure 4.4.: Illustration of oscillatory behaviour in HMC trajectories when using an ABC target density (4.27) in the input space to a generative model. The left axis shows the two-dimensional input space U of a toy differentiable generative model with a Gaussian input density ρ (green shading). The 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 [92] 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 [22].

4.9 METHOD

Our constrained HMC implementation is shown in Algorithm 11. 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 [129]. 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 11. Here we use

Algorithm 11 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

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^\mathsf{T}\left(\mathbf{J}_{\mathbf{g_x}}(\boldsymbol{u}')\boldsymbol{J}^\mathsf{T}\right)^{-1}(\mathbf{g_x}(\boldsymbol{u}') - \boldsymbol{x}). \tag{4.46}$$

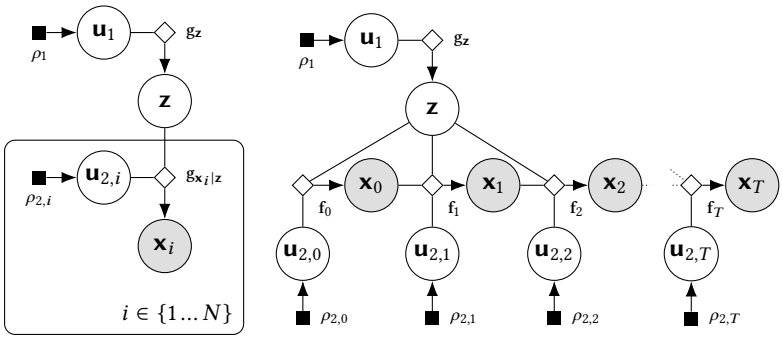
This requires recalculating the Jacobian and solving a dense linear system within the optimisation loop. Instead we use a symmetric quasi-Newton update,

$$\boldsymbol{u}' \leftarrow \boldsymbol{u}' - \boldsymbol{J}^\mathsf{T}\left(\boldsymbol{J}\boldsymbol{J}^\mathsf{T}\right)^{-1}(\mathrm{g}_{\boldsymbol{x}}(\boldsymbol{u}') - \boldsymbol{x}). \tag{4.47}$$

as proposed in [11]. 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 $\|\mathrm{g}_{\boldsymbol{x}}(\boldsymbol{u}) - \boldsymbol{x}\|_\infty < \epsilon$, 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 11) 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 [235]. The square root of the tolerance ϵ used for the initial Newton convergence check *in the output space of generator* (line 29 in Algorithm 11) is used for the reverse-step check on *the inputs* (line 48 in Algorithm 11) based on standard recommendations for checking convergence in optimisation routines [49]. In the implementation we used in the experiments, we fall back to a MINPACK [152] implementation of the robust Powell's Hybrid method [181] 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 num-



(a) Independent $\mathbf{x}_i$ (b) Markovian $\mathbf{x}_i$

Figure 4.5.: Factor graphs of examples of structured directed generative models.

ber of 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 [38], which demonstrates the validity of the constrained HMC framework theoretically and experimentally. The proposed method in [38] uses a RATTLE-based integrator rather than the geodesic scheme used here. The focus in [38] 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 [41] also considers applying a HMC scheme to sample from non-linear manifolds embedded in a Euclidean space. Similarly to [38] however the motivation is performing inference with respect to distributions explicitly defined on a manifold such as directional statistics. The method presented in [41] 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 [129], 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 [235]. 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 [60], 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 [60] 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 [143], 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 [47, 229]. 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 [23]. 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* [144]. 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 [218] 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. [144]. 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 $\sigma = 1$,

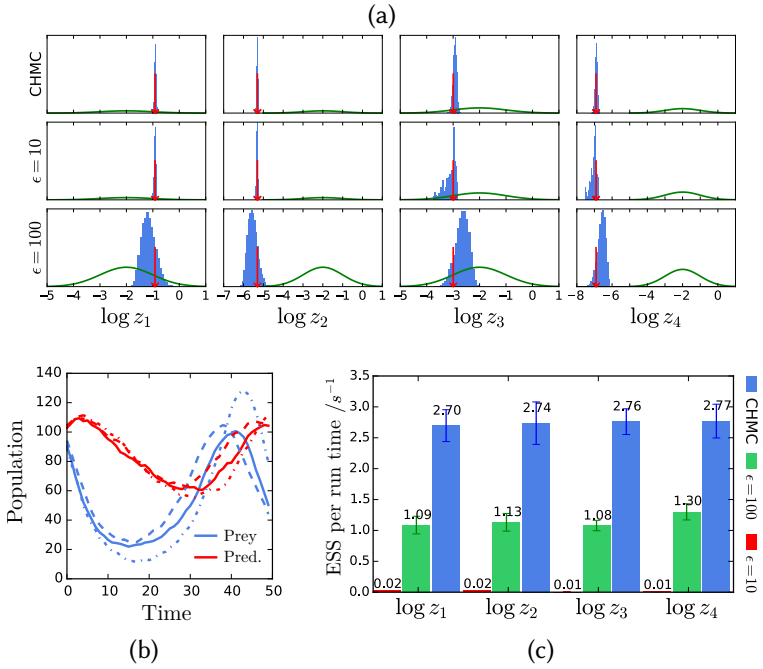


Figure 4.6.: Lotka–Volterra (a) Marginal empirical histograms for the (logarithm of the) four parameters (columns) from constrained HMC samples (top) and ABC samples with $\epsilon = 10$ (middle) and $\epsilon = 100$ (bottom). Horizontal axes shared across columns. Red arrows indicate true parameter 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 12. 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 12 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 [155] 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 [180]) 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

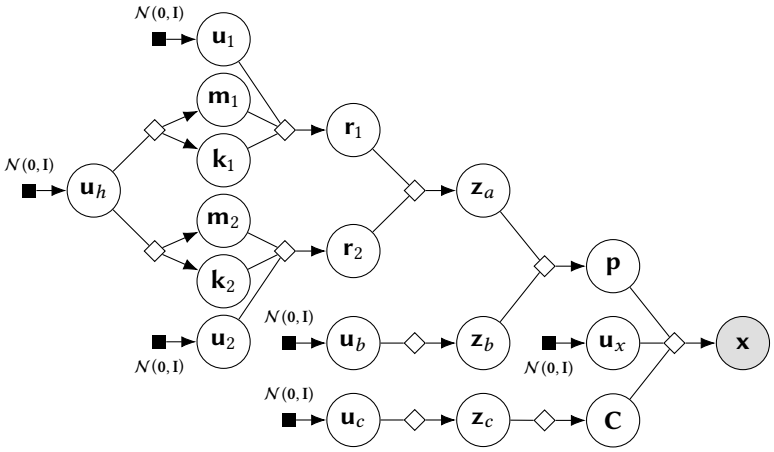


Figure 4.7.: Factor graph of human pose differentiable generative model. The operations corresponding to the deterministic nodes ($\diamond$) in the graph are described in Algorithm 13.

observed 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* [113] 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 [2] with a Gaussian VAE [119] 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 13 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 $\textsc{DifferentiableNetwork}(\mathbf{u}_h)$
  $\mathbf{h}_0 \leftarrow \tanh(\boldsymbol{W}_0 \mathbf{u}_h + \boldsymbol{b}_0)$
  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

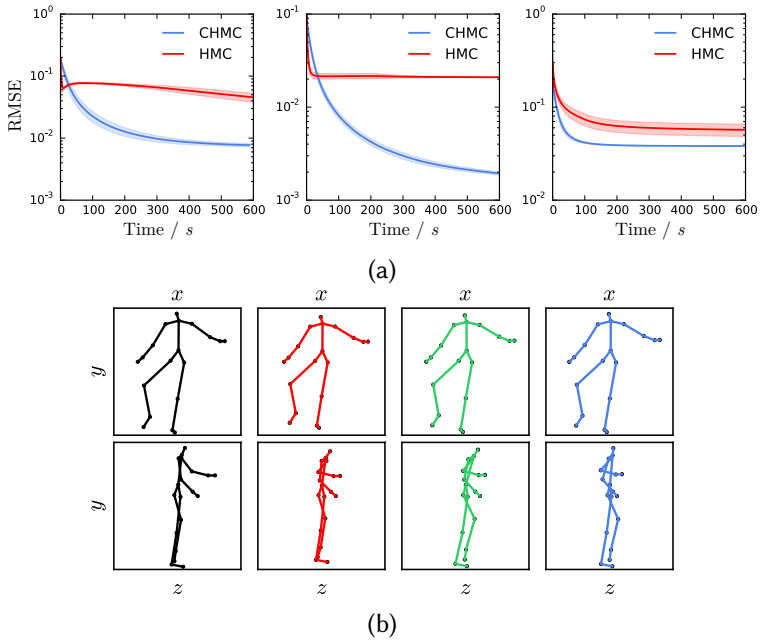


Figure 4.8.: Human pose (a) RMSEs of 3D pose posterior mean estimates given binocular projections, using samples from our method (blue) versus running HMC in hierarchical model (red) for three different scenes sampled from the prior. Horizontal axes show computation time to produce number of samples 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 [96], 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 [129]. 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 [138, 182]. 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 $\mathbf{g}_{\mathbf{x}}$.

5 CONTINUOUS TEMPERING

Applications of MCMC methods to perform inference in challenging statistical physics problems were among the earliest uses of the first modern electronic computers [195]. In the years since, MCMC methods have become a mainstay for performing approximate inference in complex probabilistic models in most fields of computational science. Statistical physics in particular has continued to contribute to key developments of MCMC methodology, such as tempering methods [85, 139, 215] and gradient-based dynamics [66].

The aim in MCMC methods is to construct an aperiodic and irreducible Markov chain which leaves a target distribution invariant, so that samples from the chain can be used to form Monte Carlo estimates of expectations with respect to the target distribution. Several general purpose constructions have been developed for specifying valid Markov transition operators for arbitrary target distributions, including Metropolis–Hastings methods [110, 146], slice sampling [163] and Gibbs sampling [78, 84].

While these methods give a basis for constructing chains that will asymptotically give correct inferences, producing algorithms that will give reasonable results in a practical amount of time is still a major challenge. For the restricted case of target distributions defined by densities that are differentiable functions of a real-valued vector variable, HMC[1] [66, 165] provides a framework for defining efficient Markov chains that exploit the gradient of the target density.

Although HMC is able to efficiently explore contiguous regions of high probability density in a target distribution, as with most MCMC methods using local moves it struggles to move between isolated modes in multimodal target distributions [165]. Tempering methods [85, 139, 215] which augment the state space with an temperature variable are often used to improve exploration in multimodal distributions. As the tem-

1 Originally termed *Hybrid Monte Carlo* in [66].

perature variable is typically discrete however it cannot be included in HMC updates.

In this paper we present an alternative *continuous tempering* approach which instead augments the state with a continuous temperature variable. This allows use of HMC to jointly update both the temperature and original state, making the method simple to use with existing HMC implementations. The proposed approach both improves exploration of multimodal densities and also provides estimates of the typically unknown normalising constant of the target density, which is often an important inferential quantity in its own right for model comparison [82].

5.1 HAMILTONIAN MONTE CARLO

HMC can be used to perform inference whenever the target distribution of interest is defined on a random vector $\mathbf{x} \in \mathbb{R}^D = X$, by a density function which is differentiable and has support almost everywhere in X. Distributions with bounded support can often be mapped to an equivalent unbounded distribution by performing a change of variables via a smooth bijective map [43, 122]. We will follow the common convention of defining the target density in a Boltzmann-Gibbs form in terms of a real-valued *potential energy* function $\phi : X \to \mathbb{R}$ and a (typically unknown) normalising constant Z,

$$\mathrm{p}_{\mathbf{x}}(\boldsymbol{x}) = \frac{1}{Z} \exp(-\phi(\boldsymbol{x})). \tag{5.1}$$

The key idea of HMC is to simulate a Hamiltonian dynamic in an extended state space to form long-range proposals for a Metropolis accept-reject step with a high probability of acceptance. The *target vector* $\mathbf{x}$ is augmented with a *momentum vector* $\mathbf{p} \in \mathbb{R}^D$. Typically the momentum is chosen to be independent of $\mathbf{x}$ with marginal $\mathrm{p}_{\mathbf{p}}(\boldsymbol{p}) \propto \exp(-\tau(\boldsymbol{p}))$, with a joint density

$$\mathrm{p}_{\mathbf{x},\mathbf{p}}(\boldsymbol{x}, \boldsymbol{p}) = \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{p}_{\mathbf{p}}(\boldsymbol{p}) \propto \exp(-\phi(\boldsymbol{x}) - \tau(\boldsymbol{p})). \tag{5.2}$$

With analogy to classical dynamics, $\tau(\boldsymbol{x})$ is referred to as the *kinetic energy* and $h(\boldsymbol{x}, \boldsymbol{p}) = \phi(\boldsymbol{x}) + \tau(\boldsymbol{p})$ is termed the *Hamiltonian*. By construction, marginalising the joint density over the momenta recovers $\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})$.

Each HMC update involves a discrete-time simulation of the canonical Hamiltonian dynamic

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial h}{\partial \boldsymbol{p}}^{\mathsf{T}} = \frac{\partial \tau}{\partial \boldsymbol{p}}^{\mathsf{T}} \qquad \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = -\frac{\partial h}{\partial \boldsymbol{x}}^{\mathsf{T}} = -\frac{\partial \phi}{\partial \boldsymbol{x}}^{\mathsf{T}} \tag{5.3}$$

which conserves the Hamiltonian and is time-reversible and volume-preserving. Through choice of an appropriate symplectic integrator such as the popular leapfrog (Störmer-Verlet) scheme, the simulated discrete time dynamic remains exactly time-reversible and volume-preserving and also approximately conserves the Hamiltonian even over long simulated trajectories [130].

The discrete-time dynamic is used to generate a new proposed state $(\boldsymbol{x}', \boldsymbol{p}')$ given the current state $(\boldsymbol{x}, \boldsymbol{p})$ and this proposal accepted or rejected in a Metropolis step with acceptance probability $\min\{1, \exp(h(\boldsymbol{x}, \boldsymbol{p}) - h(\boldsymbol{x}', \boldsymbol{p}'))\}$. Due to the approximate Hamiltonian conservation the acceptance probability is typically close to one, and so HMC is able to make long-range moves in high-dimensional state-spaces while still maintaining high acceptance rates, a significant improvement over the behaviour typical of simpler Metropolis–Hastings methods in high-dimensions.

The energy conservation property which gives this desirable behaviour also however suggests that standard HMC updates are unlikely to move between isolated modes in a target distribution. The Hamiltonian is approximately conserved over a trajectory therefore $\phi(\boldsymbol{x}') - \phi(\boldsymbol{x}) \approx \tau(\boldsymbol{p}) - \tau(\boldsymbol{p}')$. Typically a quadratic kinetic energy $\tau(\boldsymbol{p}) = \boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p} \,/\, 2$ is used corresponding to a Gaussian marginal density on the momentum. As this kinetic energy is bounded below by zero, the maximum change in potential energy over a trajectory is approximately equal to the initial kinetic energy.

At equilibrium the momenta will have a Gaussian distribution and so the kinetic energy a χ^2 distribution with mean $D/2$ and variance D [22, 165]. If potential energy barriers significantly larger than $\sim D$ separate regions of the configuration state space the HMC updates are unlikely to move across the barriers meaning impractically long sampling runs will be needed for effective ergodicity.

5.2 THERMODYNAMIC METHODS

A common approach in MCMC methods for dealing with multimodal target distributions is to introduce a concept of temperature. In statistical mechanics, the Boltzmann distribution on a configuration $\mathbf{x}$ of a mechanical system with energy function ϕ and in thermal equilibrium with a heat bath at temperature T is defined by a probability density $\exp(-\beta\phi(\boldsymbol{x}))/z(\beta)$ where $\beta = (k_B T)^{-1}$ is the *inverse temperature*, k_B is Boltzmann's constant and $z(\beta)$ is the *partition function*. At high temperatures ($\beta \to 0$) the density function becomes increasingly flat across the target state space and, correspondingly, energy barriers between different regions of the state space become lower.

In the above statistical mechanics formulation, if $\mathbf{x} \in \mathbb{R}^D$ the distribution in the limit $\beta \to 0$ has an improper flat density across the target state space. More usefully from a statistical perspective we can use an inverse temperature variable $\beta \in [0, 1]$ to geometrically bridge between a simple *base distribution* with normalised density $\exp(-\psi(\boldsymbol{x}))$ at $\beta = 0$ and the target distribution at $\beta = 1$

$$\pi(\boldsymbol{x} \mid \beta) = \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x})), \tag{5.4}$$

$$z(\beta) = \int_X \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{x}. \tag{5.5}$$

Several approaches to introducing a system temperature in to MCMC methods have been proposed. In *simulated tempering* (ST) [139] an ordered set of inverse temperatures are chosen

$$\{\beta_n\}_{n=0}^N : 0 = \beta_0 < \beta_1 < \cdots < \beta_N = 1, \tag{5.6}$$

and a joint distribution defined with density

$$\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta_n) \propto \pi(\boldsymbol{x} \mid \beta_n) \exp(w_n), \tag{5.7}$$

where $\{w_n\}_{n=0}^N$ are a set of prior weights associated with each inverse temperature. Alternating MCMC updates of $\beta \mid \mathbf{x}$ and $\mathbf{x} \mid \beta$ are performed, with $\mathbf{x}$ samples for which $\beta = 1$ converging in distribution to the target. Updates of the inverse temperature variable can propose moves to a limited set of neighbouring inverse temperatures or sample β independently from its conditional $\mathrm{p}_{\beta|\mathbf{x}}(\beta_n \mid \boldsymbol{x})$.

The ratio of the marginal densities of $\mathrm{p}_\beta(1)$ and $\mathrm{p}_\beta(0)$ can be related to the usually unknown normalising constant Z for the target density by

$$Z = \exp(w_0 - w_N)\, \mathrm{p}_\beta(1) / \mathrm{p}_\beta(0), \tag{5.8}$$

allowing estimation of Z from a ST chain by computing the ratio of counts of samples with $\beta = 1$ and $\beta = 0$. As an alternative [42] proposes to instead use a '*Rao-Blackwellised*' estimator for Z based on the identity

$$\mathrm{P}_\beta(\beta_n) = \int_X \mathrm{P}_{\beta|\mathbf{x}}(\beta_n \mid \boldsymbol{x}) \mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \tag{5.9}$$

which indicates $\mathrm{P}_\beta(\beta_n)$ can be estimated by averaging the conditional probabilities $\mathrm{P}_{\beta|\mathbf{x}}(\beta_n \mid \boldsymbol{x})$ across samples of $\mathbf{x}$ from the joint in (5.7). The authors empirically demonstrate this estimator can give significantly improved estimation accuracy over the simpler count-based estimator.

A problem for ST methods is that as $\mathrm{P}_\beta(\beta_n) \propto z(\beta_n) \exp(w_n)$ and the partition function can vary across several orders of magnitude, the chain can mix poorly between inverse temperatures. This is often tackled with an iterative approach in which initial pilot runs are used to estimate $\mathrm{P}_\beta(\beta_n)$ and the weights $\{w_n\}_{n=0}^N$ set so as to try to flatten out this marginal distribution [88].

An alternative is *parallel tempering* (PT) [68, 85, 215], where multiple Markov chains on the target state are run in parallel, with the n^{th} chain having an associated inverse temperature β_n defined as in (5.6). MCMC updates are performed on each chain which leave a distribution with unnormalised density $\pi(\boldsymbol{x} \mid \beta_n)$ invariant. Interleaved with these updates, exchanges of the states of the chains at adjacent inverse temperatures (β_n, β_{n+1}) are proposed in a Metropolis–Hastings step. Parallel tempering sidesteps the issue with mixing between inverse temperatures in ST by keeping the inverse temperatures for the chains fixed, at a cost of a significantly increased state size.

Tempered transitions [160] uses a different approach, deterministically altering the inverse temperature rather than including it as part of the Markov chain state. Again an ordered set of inverse temperatures are specified as in (5.6). For each inverse temperature $\beta_n : n > 0$ a pair of transition operators $\hat{\mathrm{T}}_n, \check{\mathrm{T}}_n$ are defined which satisfy a reversibility condition $\hat{\mathrm{T}}_n[\boldsymbol{x}' \mid \boldsymbol{x}]\pi(\boldsymbol{x} \mid \beta_n) = \check{\mathrm{T}}_n[\boldsymbol{x} \mid \boldsymbol{x}']\pi(\boldsymbol{x}' \mid \beta_n)$ (if $\hat{\mathrm{T}}_n = \check{\mathrm{T}}_n$ this is

just detailed balance). The transition operators associated with lower inverse-temperatures will typically be able to make larger moves in the state space.

These transition operators are applied to the current state in a fixed sequence $\hat{\mathsf{T}}_1 \dots \hat{\mathsf{T}}_N \check{\mathsf{T}}_N \dots \check{\mathsf{T}}_1$ to generate a chain of intermediate states, with the idea that in the initial 'heating' stage $\hat{\mathsf{T}}_1 \dots \hat{\mathsf{T}}_N$ the state will be roughly distributed after each transition according to the target distribution at an increasing series of temperatures before being 'cooled' back down with the transitions $\check{\mathsf{T}}_N \dots \check{\mathsf{T}}_1$ so that the final state is approximately distributed according to the target (with $\beta_0 = 1$) again. The overall transition from the initial state to the final state in the chain is then accepted or rejected in a Metropolis–Hastings step. A HMC specific variation of this idea in which the tempering is applied within a simulated trajectory by scaling the momenta in a deterministic sequence has also been proposed [165].

annealed importance sampling (AIS) [162] is another thermodynamic ensemble method, closely related to *Tempered Transitions* [160] and often the 'go-to' method for normalising constant estimation in the machine learning literature e.g [203, 233]. In AIS an ordered set of inverse temperatures are defined as in (5.6) and a corresponding series of transition operators $\{\mathsf{T}_n\}_{n=1}^{N-1}$. Each T_n leaves a distribution with unnormalised density $\pi(\boldsymbol{x} \mid \beta_n)$ invariant. Assuming the base distribution corresponding to $\beta_0 = 0$ can be sampled from independently, an independent state $\boldsymbol{x}^{(0)}$ from this distribution is generated and then the transition operators applied in a fixed sequence $\mathsf{T}_1 \dots \mathsf{T}_{N-1}$ to generate a chain of intermediate states $\{\boldsymbol{x}^{(n)}\}_{n=1}^{N-1}$. The product of ratios

$$r = \prod_{n=0}^{N-1} \left(\pi\left(\boldsymbol{x}^{(n)} \mid \beta_{n+1}\right) / \pi\left(\boldsymbol{x}^{(n)} \mid \beta_n\right) \right) \tag{5.10}$$

is an importance weight for the final state $\boldsymbol{x}^{(N-1)}$ with respect to the target distribution (5.1) and also an unbiased estimator for Z. By simulating multiple independent runs of this process, a set of importance weighted samples can be computed to estimate expectations with respect to the target (5.1) and the weights r used to unbiasedly estimate Z. Due to Jensen's inequality the unbiased estimate of Z corresponds to a stochastic lower bound on $\log Z$ [102].

In cases where an exact sample from the target distribution can be generated, running reversed AIS from the target to base distribution allow calculating an unbiased estimate of Z^{-1} and so a stochastic upper bound on $\log Z$ [102]. This combination of stochastically lower bounding $\log Z$ by forward AIS runs and stochastically upper bounding by reverse AIS runs is termed *bidirectional Monte Carlo*.

5.3 CONTINUOUS TEMPERING

In all three of AIS, PT and ST the choice of the discrete inverse temperature schedule (5.6) is key to getting the methods to perform well in complex high dimensional distributions [19, 21, 160]. To get reasonable performance it may be necessary to do preliminary pilot runs to guide the number of inverse temperatures and spacing between them to use, adding to the computational burden and difficulty of using these methods in a black-box fashion. A natural question is therefore whether it is possible to use a continuously varying inverse temperature variable.

Path sampling [82] proposes this approach, defining a general *path* as a function parametrised by β which continuously maps between the target density $\exp(-\phi(\boldsymbol{x}))/Z$ at $\beta = 1$ and a base density $\exp(-\psi(\boldsymbol{x}))$ at $\beta = 0$, with the geometric bridge in (5.4) a particular example. A joint target $\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta) \propto \pi(\boldsymbol{x} \mid \beta)\, \rho(\beta)$ is defined with $\rho(\beta)$ a chosen prior on the inverse temperature variable analogous to the weights $\{w_n\}_{n=0}^N$ in simulated tempering. In [82] it is proposed to construct a Markov chain leaving the joint density invariant by alternating updates of $\mathbf{x} \mid \beta$ and $\beta \mid \mathbf{x}$. Samples from the joint system can be used to estimate Z via the *thermodynamic integration* identity

$$\log Z = \int_0^1 \int_X \frac{\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta)}{\mathrm{p}_{\beta}(\beta)} \frac{\partial \log \pi(\boldsymbol{x} \mid \beta)}{\partial \beta} \,\mathrm{d}\boldsymbol{x}\, \mathrm{d}\beta. \tag{5.11}$$

Adiabatic Monte Carlo [21] also proposes using a continuously varying inverse temperature variable, here specifically in the context of HMC. The original Hamiltonian system $(\mathbf{x}, \mathbf{p})$ is further augmented with a

continuous inverse temperature coordinate $\beta \in [0, 1]$. A *contact Hamiltonian* is defined on the augmented system,

$$h_c(\boldsymbol{x}, \boldsymbol{p}, \beta) = \beta\phi(\boldsymbol{x}) + (1-\beta)\psi(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{p}^\mathsf{T}\boldsymbol{M}^{-1}\boldsymbol{p} + \log z(\beta) + h_0 \tag{5.12}$$

this defining a corresponding *contact Hamiltonian flow*,

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial h_c}{\partial \boldsymbol{p}}^\mathsf{T}, \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = \frac{\partial h_c}{\partial \beta}\boldsymbol{p} - \frac{\partial h_c}{\partial \boldsymbol{x}}^\mathsf{T}, \frac{\mathrm{d}\beta}{\mathrm{d}t} = h_c - \frac{\partial h_c}{\partial \boldsymbol{p}}\boldsymbol{p}. \tag{5.13}$$

The contact Hamiltonian flow restricted to the zero level-set of the contact Hamiltonian (which the initial state can always be arranged to lie in by appropriately choosing the arbitrary constant h_0) generates trajectories which exactly conserve the contact Hamiltonian and extended state space volume element, and correspond to the thermodynamical concept of a reversible adiabatic process.

For a quadratic kinetic energy, $\frac{\mathrm{d}\beta}{\mathrm{d}t}$ is always non-positive and so forward simulation of the contact Hamiltonian flow generates non-increasing trajectories in β (and backwards simulation generates non-decreasing trajectories in β). In the ideal case this allows the inverse temperature range $[0, 1]$ to be coherently traversed without the random-walk exploration inherent to simulated tempering.

Simulating the contact Hamiltonian flow is non-trivial in practice however: the contact Hamiltonian (5.12) depends on the log partition function $\log z(\beta)$, the partial derivatives of which require computing expectations with respect to $\pi(\boldsymbol{x} \mid \beta)$ which for most problems is intractable to do exactly. Moreover the contact flow can encounter meta-stabilities whereby $\frac{\mathrm{d}\beta}{\mathrm{d}t}$ becomes zero and the flow halts at an intermediate β meaning the flow no longer defines a bijection between $\beta = 0$ and $\beta = 1$. This can be ameliorated by regular resampling of the momenta however this potentially increases random-walk behaviour.

An alternative *extended Hamiltonian approach* for simulating a system with a continuously varying inverse temperature was proposed recently in the statistical physics literature [93]. The inverse temperature of the system is indirectly set via an auxiliary variable, which we will term a *temperature control variable* $\mathsf{u} \in \mathbb{R}$. This control variable is mapped to an interval $[s, 1]$, $0 < s < 1$ via a smooth piecewise defined function $\beta : \mathbb{R} \to [s, 1]$, with the conditions that for a pair of thresholds

(θ_1, θ_2) with $0 < \theta_1 < \theta_2$, $\beta(u) = 1 \; \forall \; |u| \leq \theta_1$, $\beta(u) = s \; \forall \; |u| \geq \theta_2$ and $s < \beta(u) < 1 \; \forall \; \theta_1 < |u| < \theta_2$.

Unlike Adiabatic Monte Carlo, an additional momentum variable v corresponding to u is also introduced. Although seemingly a minor difference this simplifies the implementation of the approach significantly as the system retains a symplectic structure and can continue to be viewed within the usual Hamiltonian dynamics framework. An *extended Hamiltonian* is defined on the augmented system

$$\tilde{h}(\boldsymbol{x}, u, \boldsymbol{p}, v) = \beta(u)\phi(\boldsymbol{x}) + \omega(u) + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p} + \frac{v^2}{2m} \tag{5.14}$$

where ω is a 'confining potential' on u and m is the mass (marginal variance) associated with v. This extended Hamiltonian is separable with respect to the extended configuration ($\mathbf{x}$, u) and extended momentum ($\mathbf{p}$, v) and so can be efficiently simulated using a standard leapfrog integrator. In [93] the extended Hamiltonian dynamics are integrated using a Langevin scheme without Metropolis adjustment and shown to improve mixing in several molecular dynamics problems.

Due to the condition $\beta(u) = 1 \; \forall \; |u| < \theta_1$, the set of sampled configuration states $\mathbf{x}$ which have associated $|\mathsf{u}| < \theta_1$ will (assuming the dynamic is ergodic and Metropolis adjustment were used) asymptotically converge in distribution to the target, and so can be used to estimate expectations without any importance re-weighting. The β function is required to be bounded below by a $s > 0$ in [93] due to the base density being bridged to being an improper uniform density on X. The partition function $z(\beta) \to \infty$ as $\beta \to 0$ in this case, which would imply an infinite density for regions in the extended state space where $\beta(\mathsf{u}) = 0$. Even with a non-zero lower bound on β, the large variations in $z(\beta(\mathsf{u}))$ across different u values can lead to the dynamic poorly exploring the u dimension.

In [93] this issue is tackled by introducing an adaptive history-dependent biasing potential on u to try to achieve a flat density across a bounded interval $|\mathsf{u}| < \theta_2$, using for example metadynamics [124]. The resulting non-Markovian updates bias the invariant distribution of the target state however this can be accounted for either by a re-weighting scheme [35], or using a vanishing adaptation.

5.4 PROPOSED METHOD

As in [93] we define an extended Hamiltonian on an augmented state ($\mathbf{x}$, u) with associated momenta ($\mathbf{p}$, v),

$$\begin{aligned}\tilde{h}(\boldsymbol{x}, u, \boldsymbol{p}, v) =& \beta(u)(\phi(\boldsymbol{x}) + \log\zeta) + (1-\beta(u))\psi(\boldsymbol{x}) \\ &- \log\left|\frac{\partial\beta}{\partial u}\right| + \frac{1}{2}\boldsymbol{p}^{\mathsf{T}}\boldsymbol{M}^{-1}\boldsymbol{p} + \frac{v^2}{2m}.\end{aligned} \tag{5.15}$$

As with the approach of [93], this Hamiltonian is separable and the corresponding dynamic can be efficiently simulated with a leapfrog integrator. The reversible and volume-preserving simulated dynamic can then be used as a proposal generating mechanism on the joint space ($\mathbf{x}$, u, $\mathbf{p}$, v) for a Metropolis–Hastings step as in standard HMC. We will term this approach of running HMC in the extended joint space *joint continuous tempering*.

In contrast to [93] we propose to use a smooth monotonically increasing map $\beta : \mathbb{R} \to [0,1]$ as the inverse temperature function, with our default choice in all experiments being the logistic sigmoid $\beta(u) = (1+\exp(-u))^{-1}$.

As previously ψ is the negative logarithm of a simple *normalised* base density, which (as we will motivate in the next section) we will usually choose to be an approximation to the target density (5.1). Similarly the $\log\zeta$ term will be chosen to be an approximation to $\log Z$.

We can marginalise out the momenta from the joint distribution defined by this Hamiltonian, giving a joint density

$$\begin{aligned}\mathrm{p}_{\mathbf{x},\mathrm{u}}(\boldsymbol{x}, u) \propto \left|\frac{\partial\beta}{\partial u}\right| \exp\Big(&- \beta(u)(\phi(\boldsymbol{x}) + \log\zeta) \\ &- (1-\beta(u))\psi(\boldsymbol{x})\Big).\end{aligned} \tag{5.16}$$

If we define a variable $\beta = \beta(\mathrm{u})$ and use the change of variables formula for a density, we further have that

$$\mathrm{p}_{\mathbf{x},\beta}(\boldsymbol{x}, \beta) \propto \frac{1}{\zeta^{\beta}} \exp(-\beta\phi(\boldsymbol{x}) - (1-\beta)\psi(\boldsymbol{x})) \tag{5.17}$$

The bijectivity between u and β is useful as although if simulating a Hamiltonian dynamic we will generally wish to work with u as it is

unbounded, the conditional density on β given $\mathbf{x}$ has a tractable normalised form

$$\mathrm{p}_{\beta|\mathbf{x}}(\beta \mid \boldsymbol{x}) = \frac{\exp(-\beta\Delta(\boldsymbol{x}))\Delta(\boldsymbol{x})}{1-\exp(-\Delta(\boldsymbol{x}))}, \tag{5.18}$$

with $\Delta(\boldsymbol{x}) = \phi(\boldsymbol{x}) + \log\zeta - \psi(\boldsymbol{x})$. This corresponds to an exponential distribution with rate parameter $\Delta(\boldsymbol{x})$ truncated to $[0,1]$. As an alternative to the joint updates, another option is therefore to form a Markov chain which leaves (5.17) invariant by alternating independently sampling β from its conditional given $\mathbf{x}$ and performing a transition which leaves the conditional on $\mathbf{x}$ given β invariant, similar to the suggested approach in [82]. We will term this Gibbs sampling type procedure as *Gibbs continuous tempering*.

Further we can use (5.18) to write the marginal density p_β

$$\mathrm{p}_\beta(\beta) = \mathbb{E}\left[\frac{\exp(-\beta\Delta(\mathbf{x}))\Delta(\mathbf{x})}{1-\exp(-\Delta(\mathbf{x}))}\right]. \tag{5.19}$$

By integrating the joint (5.17) over X we also have that

$$\begin{aligned} \mathrm{p}_\beta(\beta) = \frac{1}{C\zeta^\beta}\int_X \pi(\boldsymbol{x} \mid \beta)\,\mathrm{d}\boldsymbol{x} = \frac{z(\beta)}{C\zeta^\beta} \\ \Rightarrow\ \mathrm{p}_\beta(0) = \frac{1}{C},\ \mathrm{p}_\beta(1) = \frac{Z}{C\zeta}, \end{aligned} \tag{5.20}$$

for an unknown normalising constant C of the joint, and so for MCMC samples $\left\{\boldsymbol{x}^{(s)},\ \beta^{(s)}\right\}_{s=1}^S$ from the joint (5.17)

$$Z = \frac{\mathrm{p}_\beta(1)}{\mathrm{p}_\beta(0)}\zeta = \lim_{S\to\infty}\frac{\sum_{s=1}^S\left(w_1\left(\boldsymbol{x}^{(s)}\right)\right)}{\sum_{s=1}^S\left(w_0\left(\boldsymbol{x}^{(s)}\right)\right)}\zeta, \tag{5.21}$$

$$\text{with } w_0(\boldsymbol{x}) = \frac{\Delta(\boldsymbol{x})}{1-\exp(-\Delta(\boldsymbol{x}))},\ w_1(\boldsymbol{x}) = \frac{\Delta(\boldsymbol{x})}{\exp(\Delta(\boldsymbol{x}))-1}.$$

This can be seen to be a continuous analogue of the *Rao-Blackwellised* estimator (5.9) used in [42]. Similarly we can calculate consistent estimates of expectations with respect to the target density $\mathrm{p}_{\mathbf{x}|\beta}(\boldsymbol{x} \mid 1) = \exp(-\phi(\boldsymbol{x}))/Z$ as importance weighted sums

$$\begin{aligned} \mathbb{E}[f(\mathbf{x}) \mid \beta = 1] &= \frac{\int_X f(\boldsymbol{x})\,\mathrm{p}_{\beta|\mathbf{x}}(1 \mid \boldsymbol{x})\,\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}{\int_X \mathrm{p}_{\beta|\mathbf{x}}(1 \mid \boldsymbol{x})\,\mathrm{p}_{\mathbf{x}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}} \\ &\approx \frac{\sum_{s=1}^S\left(w_1\left(\boldsymbol{x}^{(s)}\right)f\left(\boldsymbol{x}^{(s)}\right)\right)}{\sum_{s=1}^S\left(w_1\left(\boldsymbol{x}^{(s)}\right)\right)}. \end{aligned} \tag{5.22}$$

We can also estimate expectations with respect to the base density $\mathrm{p}_{\mathbf{x}|\beta}(\boldsymbol{x} \mid 0) = \exp(-\psi(\boldsymbol{x}))$ using

$$\mathbb{E}[f(\mathbf{x}) \mid \beta = 0] \approx \frac{\sum_{s=1}^{S}\left(w_0\left(\boldsymbol{x}^{(s)}\right) f\left(\boldsymbol{x}^{(s)}\right)\right)}{\sum_{s=1}^{S}\left(w_0\left(\boldsymbol{x}^{(s)}\right)\right)}. \tag{5.23}$$

Often the base density will have known moments (e.g. mean and covariance of a Gaussian base density) which can be compared to the estimates calculated using (5.23) to check for convergence problems. Convergence of the estimates to the true moments is not a sufficient condition for convergence of the chain to the target joint density (5.17) but is necessary.

5.5 CHOOSING A BASE DENSITY

By applying Hölder's and Jensen's inequalities we can bound $\mathrm{p}_\beta(\beta)$ (see Appendix A for details)

$$\frac{1}{C}\left(\frac{Z}{\zeta}\right)^{\beta} \exp\left(-\beta d^{b\to t}\right) \leq \mathrm{p}_\beta(\beta) \leq \frac{1}{C}\left(\frac{Z}{\zeta}\right)^{\beta}, \tag{5.24}$$

where $d^{b\to t}$ indicates the *Kullback–Leibler* (KL) divergence from the base to target distribution

$$d^{b\to t} = \int_X \exp(-\psi(\boldsymbol{x})) \log\left(\frac{\exp(-\psi(\boldsymbol{x}))}{\exp(-\phi(\boldsymbol{x}))/Z}\right) \mathrm{d}\boldsymbol{x}. \tag{5.25}$$

If $\zeta = Z$ the upper-bound is constant. If additionally we had $d^{b\to t} = 0$, the bound becomes tight and we would have a flat marginal density on β, which we should improve mixing in the β dimension. In reality we do not know Z and cannot choose a base distribution such that $d^{b\to t} = 0$ as we wish to use a simple distribution amenable to exploration. However under the constraint of the base distribution allowing exploration, a reasonable heuristic is to minimise the KL divergence to the target distribution. Further we want ζ as close to Z as possible.

Variational inference is an obvious route for tackling both problems, allowing us to fit a base density in a simple parametric family (e.g. Gaussian) by directly minimising the KL divergence (5.25) while also giving a lower bound on $\log Z$. In some cases we can use variational methods specifically aimed at the target distribution family. More generally

methods such as *automatic differentiation variational inference* (ADVI) [122] provide a black-box framework for fitting variational approximations to differentiable target densities. In models such as Variational Autoencoders [119, 192] a parametric variational approximation to the target density of interest (e.g. posterior on latent space) is fitted during training of the original model, in which case it provides a natural choice for the base distribution as observed in [233].

A potential problem is that the classes of target distribution that we are particularly interested in applying our approach to — those with multiple isolated modes — are precisely the same distributions that simple variational approximations will tend to fit poorly, the divergence (5.25) being minimised favouring 'mode-seeking' solutions [30], which usually fit only one mode well. This both limits how small the divergence from the base to target can be made, but also crucially is undesirable as we wish to use the base distribution to move between modes in the target. One option would be to instead to minimise the reversed form of the KL divergence from the target to base distribution, this tending to produce 'mode-covering' solutions that match global moments (and can also be used to produce an analogous lower bound on the marginal density to that for $d^{b\to t}$ in (5.24) as shown in Appendix A). Methods such as EP [149] do allow moment-matching approximations to be found and may be a good option in some cases. Another possibility would be to use an alternative divergence in a variational framework that favours mode-covering solutions, with methods using the χ-divergence [61] and Rényi divergence [132] having been recently proposed in this context.

A further alternative is to fit multiple local variational approximations $\{q_i(\boldsymbol{x})\}_{i=1}^L$ by minimising the variational objective from multiple random parameter initialisations (discarding any duplicate solutions as measured by some distance tolerance between the variational parameters), each approximating a single mode well. We can then combine these local approximations into a global approximation $q(\boldsymbol{x})$, for example using a mixture model

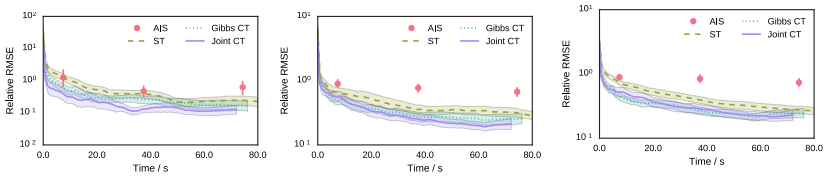
$$q(\boldsymbol{x}) = \tfrac{1}{\zeta}\textstyle\sum_{i=1}^L(\exp(\ell_i)q_i(\boldsymbol{x})), \quad \zeta = \sum_{i=1}^L \exp(\ell_i), \tag{5.26}$$

with ℓ_i the final variational objective value for q_i ($\log Z$ minus the KL divergence from q_i to target distribution). If the local approximations

had non-overlapping support this would lead to a global approximation which is guaranteed to be at least as close in KL divergence as any of the local approximations and a $\log \zeta$ which is at least as tight a lower bound on $\log Z$ as any of the individual ℓ_i [237]. A mixture distribution is unlikely to itself be a good choice of base distribution however, as it will tend to be multimodal. We can therefore instead use a base distribution with moments matched to the fitted mixture, e.g. a Gaussian $\exp[-\psi(\boldsymbol{x})]$ with mean and covariance matched to the mean and covariance of the mixture $q(\boldsymbol{x})$.

For complex target distributions it will sometimes remain difficult to find a reasonable ψ and $\log \zeta$ which well approximate ϕ and $\log Z$. If the KL divergence from the base to target distribution is high, the bounds on the marginal in (5.24) become increasingly loose and this tends to lead to low densities on intermediate inverse temperatures. This reduces the ability of the MCMC dynamic to move between the inverse temperatures corresponding to the target and base distributions and so the mixing gain from the augmentation. Similarly from (5.20) the density ratio $\mathrm{p}_\beta(1) \,/\, \mathrm{p}_\beta(0)$ is $\exp(\log Z - \log \zeta)$, and so for large $|\log Z - \log \zeta|$ the dynamic will tend to spend most of the time close to $\beta = 1$ if $\log Z > \log \zeta$ or $\beta = 0$ if $\log Z < \log \zeta$. In the former case there will be limited gain from using the extended space while in the latter the variance of the estimator in (5.22) will be comparable to directly using an importance sampling estimator for the target using samples from the base distribution.

A natural approach to try to ameliorate these effects is to use an iterative 'bootstrap' method. An initial ψ and $\log \zeta$ are chosen for example using a Gaussian variational approximation to the target density as described above. A Markov chain which leaves the joint density (5.17) invariant is then simulated. The samples from this chain can then be used to both compute an estimate for $\log Z$ using (5.21) and updated estimates of the target density mean and covariance using (5.22). A new Gaussian base density can then be specified with the updated mean and covariance estimates and $\log \zeta$ chosen to be the updated $\log Z$ estimate. Samples of the new joint density can then be used to update $\log \zeta$ and ψ again and so on. This is analogous to the iterative approaches often used in ST to choose the prior weights on the inverse temperatures [42, 88].



(a) $\log Z$ (b) $\mathbb{E}[\mathbf{x}]$ (c) $\mathbb{E}\left[\mathbf{x}\mathbf{x}^{\mathsf{T}}\right] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^{\mathsf{T}}$

Figure 5.1.: RMSEs in empirical moments estimated from MCMC samples against run time for various thermodynamic ensemble MCMC methods run on Gaussian Boltzmann machine relaxation target distributions. All RMSEs are relative to the RMSE of the corresponding approximate moments calculated using the moment-matched variational mixtures, so values below 1 represent improvement on deterministic approximation. For AIS points across time axis represent increasing number of inverse temperatures: $(1,\ 5,\ 10) \times 10^3$. For ST, Gibbs CT and joint CT curves show RMSEs for expectations calculated with increasing number of samples from chains. All curves / points show mean across 10 runs for each of 10 generated parameter sets. Filled regions / error bars show ±3 standard errors of mean.

5.6 EXPERIMENTS

We performed a series of experiments comparing our proposed approaches to existing methods. Code for running all experiments is available at `https://git.io/cthmc`. Appendix **??** shows an additional illustrative example.

5.6.1 Boltzmann machine relaxations

As a first experiment we performed inference in Gaussian mixture relaxations of a set of ten synthetic Boltzmann machine distributions [236]. The parameters of the Boltzmann machine distributions were randomly generated so that the corresponding relaxations are highly multimodal and so challenging to explore well. A 2D projection of samples from one generated distribution illustrating this multimodality is shown in Figure **??** in Appendix **??**.

The moments of the relaxation distributions can be calculated from the moments of the original discrete Boltzmann machine distribution, which for models with a small number of binary units D_B (30 in our experiments) can be computed exactly by exhaustive iteration across the 2^{D_B} discrete states. This allows ground truth moments to be calculated against which convergence can be checked. The parametrisation used is described in in Appendix B. A Gaussian base density and approximate normalising constant ζ was fit to each the 10 relaxation target

densities by matching moments to a mixture of variational Gaussian approximations (individually fitted using a mean-field approach based on the underlying Boltzmann machine distribution) as described in section 5.5.

Plots showing the RMSE in estimates of $\log Z$ and the mean and covariance of the relaxation distribution against computational run time for different sampling methods are shown in Figure 5.1. The RMSE values are normalised by the RMSEs of the corresponding estimated moments used in the base density (and $\log \zeta$) such that values below unity indicate an improvement in accuracy over the variational approximation. The curves shown are RMSEs averaged over 10 independent runs for each of the 10 generated parameter sets, with the filled regions indicating ± 3 standard errors of the mean. The free parameters of all methods were tuned on one parameter set and these values then fixed across all runs. All methods used a shared Theano [218] implementation running on a Intel Core i5-2400 quad-core CPU for the HMC updates and so run times are roughly comparable.

For ST, *Rao Blackwellised* estimators were used as described in [42], with HMC-based updates of the target state $\mathbf{x} \,|\, \beta$ interleaved with independent sampling of $\beta \,|\, \mathbf{x}$, and 1000 β_n values used. For AIS, HMC updates were used for the transition operators and separate runs with 1000, 5000 and 10000 β_n values used to obtain estimates at different run times. For the tempering approaches run times correspond to increasing numbers of MCMC samples.

The two CT approaches, Gibbs CT and joint CT, both dominate in terms of having lower average ***RMSEs!*** (**RMSEs!**) in all three moment estimates across all run times, with joint CT showing marginally better performance on estimates of $\log Z$ and $\mathbb{E}[\mathbf{x}]$. The tempering approaches seem to outperform AIS here, possibly as the highly multimodal nature of the target densities favours the ability of tempered dynamics to move the inverse temperature both up and down and so in and out of modes in the target density, unlike AIS where the fixed temperature updates are more likely to end up with chains confined to a single mode after the initial transitions for low β_n.

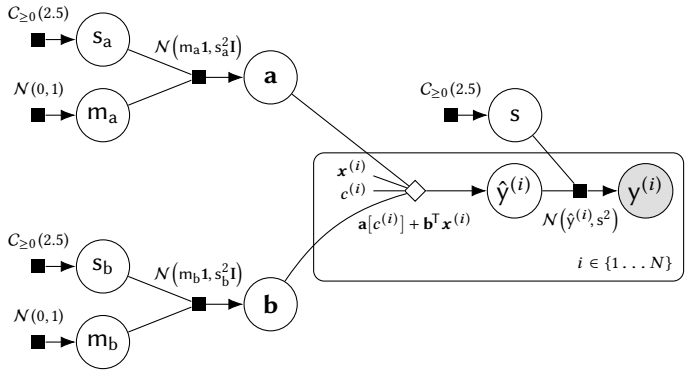


Figure 5.2.: Factor graph of Radon hierarchical regression model.

5.6.2 Hierarchical regression model

As a second experiment, we apply our joint continuous tempering approach to perform Bayesian inference in a hierarchical regression model for predicting indoor radon measurements [80]. To illustrate the ease of integrating our approach in existing HMC-based inference software, this experiment was performed with the Python package PyMC3 [206], with its ADVI feature used to fit the base density and its implementation of the adaptive NUTS [113] HMC variant used to sample from the extended space.

The regression target in the dataset is measurements of the amount of radon gas $y^{(i)}$ in $N = 919$ households. Two continuous regressors $\boldsymbol{x}^{(i)}$ and one categorical regressor $c^{(i)}$ are provided per household. A multilevel regression model defined by the factor graph in 5.2 was used. The model includes five scalar parameters ($\sigma_{\mathbf{a}}$, $\mu_{\mathbf{a}}$, $\sigma_{\mathbf{b}}$, $\mu_{\mathbf{b}}$, ϵ), an 85-dimensional intercept vector $\mathbf{a}$ and a two-dimensional regressor coefficients vector $\mathbf{b}$, giving 92 parameters in total. As an example task, we consider inferring the marginal likelihood of the data under the model. Estimation of the marginal likelihood from MCMC samples of the target density alone is non-trivial, with approaches such as the harmonic mean estimator having high variance. Here we try to establish if our approach can be used in a black-box fashion to compute a reasonable estimate of the marginal likelihood.

As our 'ground truth' we use a large batch of long AIS runs (average across 100 runs of 10000 inverse temperatures) on a separate Theano implementation of the model. We use ADVI to fit a diagonal covariance Gaussian variational approximation to the target density and use this

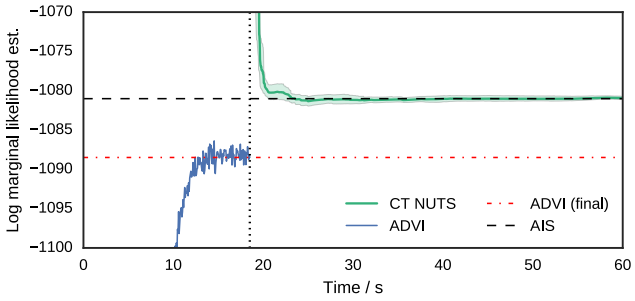


Figure 5.3.: Log marginal likelihood estimates.

Figure 5.4.: Log marginal likelihood estimates against run time for hierarchical regression model. Black dashed line shows estimated log marginal likelihood from a long AIS run which is used as a proxy ground truth. The noisy blue curve shows the *evidence lower bound* ADVI objective over training and the red dot-dashed line the final converged value used for $\log \zeta$. The green curve shows log marginal likelihood estimates using samples from NUTS chains running on the extended joint density in the estimator (5.21), with the run time corresponding to increasing samples being included in the estimator (offset by initial ADVI run time). Curve shows mean over 10 runs and filled region ±3 standard errors of mean.

as the base density. NUTS chains, initialised at samples from the base density, were then run on the extended space for 2500 iterations. The samples from these chain were used to compute estimates of the normalising constant (marginal likelihood) using the estimator (5.21). The results are shown in Figure 5.4. It can be seen that estimates from the NUTS chains in the extended continuously tempered space quickly converge to a marginal likelihood estimate very close to the AIS estimate, and significantly improve over the final lower bound on the marginal likelihood that ADVI converges to.

5.6.3 Generative image models

For our final experiments, we compared the efficiency of our CT approaches to ST and AIS for marginal likelihood estimation in decoder-based generative models for images. Use of AIS in this context was recently proposed in [233]. Specifically we estimate the joint marginal likelihood of 1000 generated binary images under the Bernoulli decoder distribution of two IWAE [40] models. Each IWAE has one stochastic hidden layer and a 50-dimensional latent space, with the two models trained on binarised versions of the MNIST [126] and Omniglot [125]

datasets using the code at https://github.com/yburda/iwae. The generated images used are shown in Appendix **??**.

By performing inference on the per-image posterior densities on the latent representation given image, the joint marginal likelihood of the images can be estimated as the product of estimates of the normalising constants of the individual posterior densities. The use of generated images allows BDMC [102] to be used to 'sandwich' the marginal likelihood with stochastic upper and lower bounds formed with long forward and backward AIS runs (averages over 16 independent runs with 10000 inverse temperatures as used in [233]).

As the per-image latent representations are conditionally independent given the images, chains on all the posterior densities can be run in parallel, with the experiments in this section run on a NVIDIA Tesla K40 GPU to exploit this inherent parallelism. The encoder of the trained IWAE models is an inference network which outputs the mean and diagonal covariance of a Gaussian variational approximation to the posterior density on the latent representation given an image and so was used to define per-image Gaussian base densities as suggested in [233]. Similarly the per-image $\log \zeta$ values were set using importance-weighted variational lower bound estimates for the per-image marginal likelihoods.

The results are shown in Figure 5.5, with the curves / points showing average results across 10 independent runs and filled regions / bars ± 3 standard error of means for the estimates. Here Gibbs CT and AIS perform similarly, with joint CT converging less quickly and simulated tempering significantly less efficient. The quick convergence of AIS and Gibbs CT here suggests the posterior densities are relatively easy for the dynamics to explore and well matched by the Gaussian base densities, limiting the gains from any more coherent exploration of the extended space by the joint CT updates. The higher per-leapfrog-step costs of the HMC updates in the extended space therefore mean the joint CT approach is less efficient overall here. The poorer performance of simulated tempering here is in part due to the generation of the discrete random indices becoming a bottleneck in the GPU implementation.

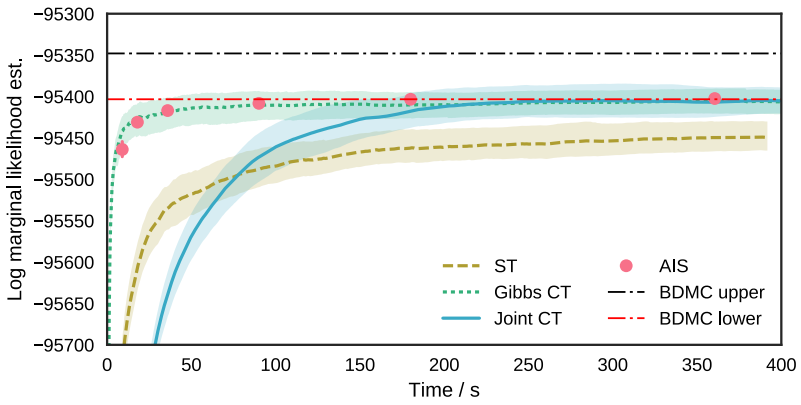
5.7 DISCUSSION

The approach we have presented is a simple but powerful extension to existing tempering methods which can both help exploration of distributions with multiple isolated modes and allow estimation of the normalisation constant of the target distribution. A key advantage of the joint CT method is its ease of implementation - it simply requires running HMC in an extended state space and so can easily be used for example within existing probabilistic programming software such as PyMC3 [206] and Stan [43]. By updating the temperature jointly with the original target state, it is also possible to leverage adaptive HMC variants such as NUTS [113] to perform tempered inference in a 'black-box' manner without the need to separately tune the updates of the inverse temperature variable.

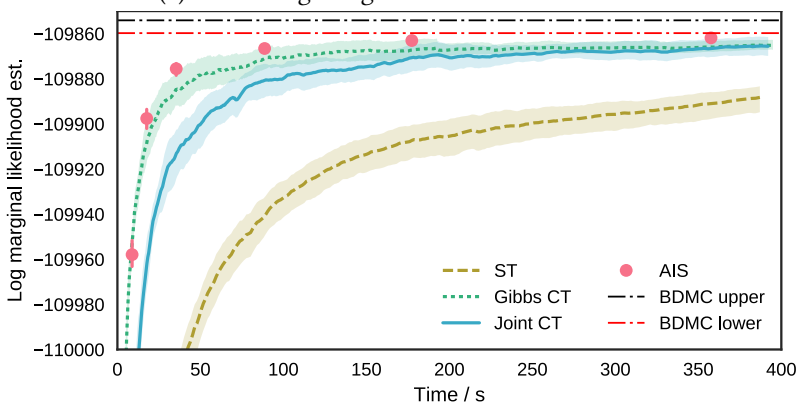
The Gibbs CT method also provides a relatively black-box framework for tempering. Compared to ST it removes the need to choose the number and spacing of discrete inverse temperatures and also replaces generation of a discrete random variate from a categorical distribution when updating β given $\mathbf{x}$ (which as seen in Section 5.6.3 can become a computational bottleneck) with generation of a truncated exponential variate (which can be performed efficiently by inverse transform sampling). Compared to the joint CT approach, the Gibbs approach is less simple to integrate in to existing HMC code due to the separate β updates, but eliminates the need to tune the temperature control mass value m and achieved similar or better sampling efficiency in the experiments in Section 5.6.

Our proposal to use variational approximations within an MCMC framework can be viewed within the context of several existing approaches which suggest combining variational and MCMC inference methods. *Variational MCMC* [56] proposes using a variational approximation as the basis for a proposal distribution in a Metropolis-Hastings MCMC method. *MCMC and Variational Inference: Bridging the Gap* [204] includes parametrised MCMC transitions within a (stochastic) variational approximation and optimises the variational bound over these (and a base distribution's) parameters. Here we exploit cheap (relative to running a long MCMC chain) but biased variational approximations to a target distribution and its normalising constant, and propose using them within an

MCMC method which gives asymptotically exact results to help improve sampling efficiency.



(a) MNIST log marginal likelihood estimates.



(b) Omniglot log marginal likelihood estimates.

Figure 5.5.: Estimates of the log joint marginal likelihood of 1000 generated images under the Bernoulli decoder distributions of two IWAE models trained on the MNIST and Omniglot datasets against computation time. The black / red dashed lines show stochastic upper / lower bounds calculated using long BDMC runs. For AIS points across time axis represent increasing number of inverse temperatures: (50, 100, 200, 500, 1000, 2000). For ST, Gibbs CT and joint CT curves show estimates calculated with an increasing number of samples from chains. All curves / points show mean across 10 runs. Filled regions / error bars show ±3 standard errors of mean.

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	$\mathrm{LogNorm}(x \mid \mu, \sigma^2)$	$\frac{1}{x\sqrt{2\pi}\sigma} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right)$	$x > 0$
Multivariate log-normal	$\boldsymbol{\mu} \in \mathbb{R}^D$: log mean $\boldsymbol{\Sigma} \in \mathcal{S}^D_{++}$: log covariance	$\mathrm{LogNorm}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\frac{\exp\left(-\frac{1}{2}(\log \boldsymbol{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1} (\log \boldsymbol{x} - \boldsymbol{\mu})\right)}{\prod_{d=1}^D (x_d)\sqrt{(2\pi)^D \lvert\boldsymbol{\Sigma}\rvert}}$	$\boldsymbol{x} \in [0, \infty)^D$
Exponential	$\lambda > 0$: rate	$\mathrm{Exp}(x \mid \lambda)$	$\lambda \exp(-\lambda x)$	$x \geq 0$
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	$\mathrm{Gamma}(x \mid \alpha, \beta)$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x)$	$x \geq 0$
Beta	$\alpha > 0$: shape $\beta > 0$: shape	$\mathrm{Beta}(x \mid \alpha, \beta)$	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$	$0 \leq x \leq 1$
Dirichlet	$\boldsymbol{\alpha} \in (0, \infty)^K$: concentration	$\mathrm{Dir}(\boldsymbol{x} \mid \boldsymbol{\alpha})$	$\frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K x_i^{\alpha_k - 1}$	$\boldsymbol{x} \in \mathbb{S}^K$
Lomax	$\alpha > 0$: shape $\beta > 0$: scale	$\mathrm{Lomax}(x \mid \alpha, \beta)$	$\frac{\alpha\beta^\alpha}{(\beta+x)^{\alpha+1}}$	$x \geq 0$

Table A.3.: Definitions of densities of parameteric distributions for bounded real random variables used in this thesis.

B COMPUTATION GRAPHS

In Chapter 1 we introduced graphical models as a compact way of representing the structure in probabilistic models. Directed factor graphs in particular offer a natural approach for representing generative models. A directed graph can be used to specify a generative process via *ancestral sampling*, with values for the variables in the graph successively calculated in a forward pass consisting of a combination of deterministic operations, represented by deterministic factor nodes, and stochastic sampling operations, represented by probabilistic factor nodes.

Computation graphs [13] are a directed graph based representation of the operations involved in evaluating a mathematical expression. Similarly to a directed factor graph, a computation graph can be consided as specifying a generative process - generation of the expression outputs given inputs - computed via forward pass through the graph. The main difference of a forward pass through a computation graph compared to an ancestral sampling pass through a directed factor graph is that the inputs to a computation graph are assumed to be given rather than sampled from marginal densities and the intermediate operations are all deterministic. In this appendix we briefly review the key concepts of computation graphs and in particular their application to perform automatic differentiation.

Here we will distinguish between two types of nodes in a computation graph. *Variable nodes* correspond 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.

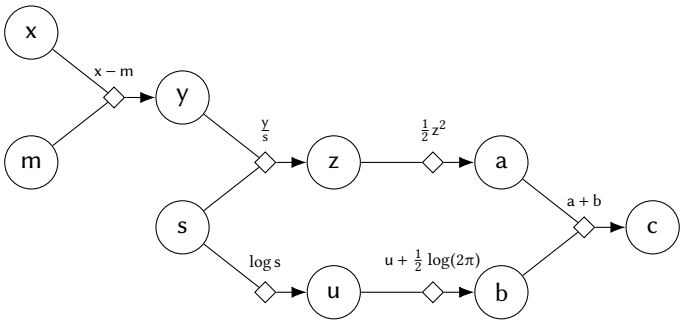


Figure B.1.: Example computation graph corresponding to calculation of the negative log density of a univariate normal distribution.

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. 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.

As with 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.

$$c = \frac{1}{2}\left(\frac{x - m}{s}\right)^2 + \log s + \frac{1}{2}\log(2\pi) \tag{B.1}$$

is shown in Figure B.1. 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.1 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.1 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 [18, 168]. 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* [212], 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.1) with respect to x, m and s can be

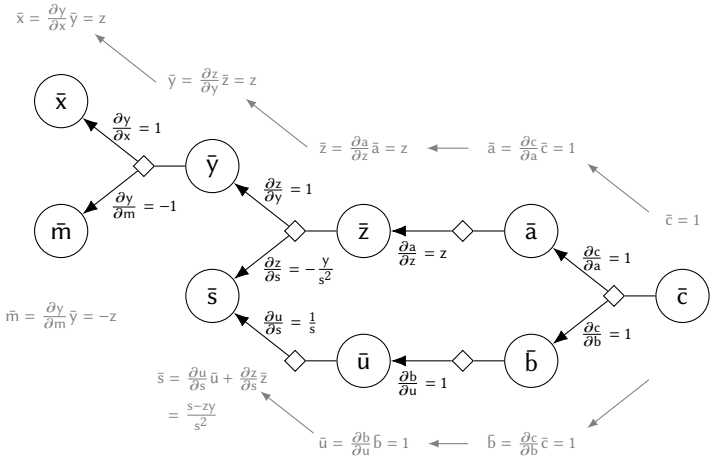


Figure B.2.: Visualisation of applying reverse-mode automatic differentiation to the computation graph in Figure B.1 to calculate the derivatives of the negative log density of a univariate normal distribution.

decomposed using the chain rule in terms of the intermediate variables in the computation graph shown in Figure B.1 as

$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial x}, \tag{B.2}$$

$$\frac{\partial c}{\partial m} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial m}, \tag{B.3}$$

$$\frac{\partial c}{\partial s} = \frac{\partial c}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial s} + \frac{\partial c}{\partial b}\frac{\partial b}{\partial u}\frac{\partial u}{\partial s}. \tag{B.4}$$

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.2 shows a visualisation of reverse-mode accumulation applied to the computation graph in Figure B.1. 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 14 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.

1: **for** $i \in \{M+1 \dots N\}$ **do** ▷ Forward pass
2: $\quad x_i \leftarrow f_i\left(\{x_j\}_{j\in \mathrm{Pa}(i)}\right)$
3: $\bar{x}_N \leftarrow 1$ ▷ Reverse pass
4: **for** $i \in \{N-1 \dots 1\}$ **do**
5: $\quad \bar{x}_i \leftarrow \sum_{j\in \mathrm{Ch}(i)} \bar{x}_j \partial_i f_j(x_i)$
6: **return** $x_N,\ \{\bar{x}_i\}_{i=1}^{M}$

denoted with barred symbols in Figure B.2 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.2).

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.2. 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 14.

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 [14]. 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 $\{\mathrm{x}_i\}_{i=1}^{M}$ and N outputs $\{\mathrm{y}_i\}_{i=1}^{N}$, i.e. the $N \times M$ matrix $\boldsymbol{J}$ with entries $J_{i,j} = \frac{\partial \mathrm{y}_i}{\partial \mathrm{x}_j}$, 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* [230], 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).

C OPTIMISATION-BASED APPROXIMATE INFERENCE

The sampling-based approaches to approximate inference discussed in Chapter 2 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. 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 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 *independently and identically distributed* (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 appendix we will review an alternative class of approximate inference methods 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 $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma)$ then we can approximate the mean of the target density as $\boldsymbol{\mu}$ and the covariance as Σ. 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.

C.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

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

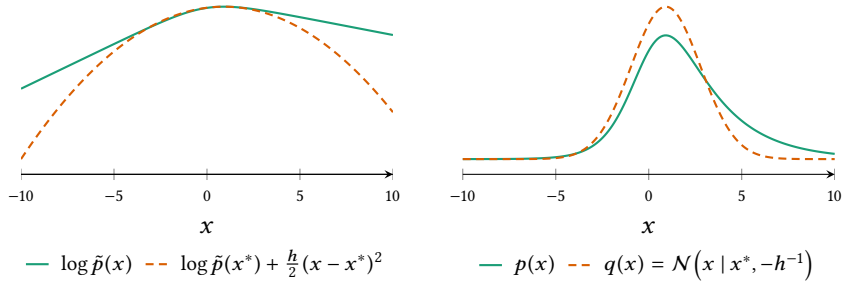


Figure C.1.: Univariate example of Laplace's method. Left axis shows the logarithm of the unnormalised target density $\log \tilde{p}(x)$ (green curve) and the corresponding quadratic Taylor series approximation $\log \tilde{p}(x^*) + \frac{h}{2}(x - x^*)^2$ (dashed orange curve) around the maxima x^* 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).

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

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

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

$$\log \tilde{p}(\boldsymbol{x}) \approx \log \tilde{p}(\boldsymbol{x}^*) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^{\mathsf{T}} \boldsymbol{H}(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*). \tag{C.3}$$

Taking the exponential of both sides we therefore have that

$$\tilde{p}(\boldsymbol{x}) \approx \tilde{p}(\boldsymbol{x}^*) \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^{\mathsf{T}}(-\boldsymbol{H}(\boldsymbol{x}^*))(\boldsymbol{x} - \boldsymbol{x}^*)\right). \tag{C.4}$$

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.

$$q(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{C}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^{\mathsf{T}} \boldsymbol{C}^{-1}(\boldsymbol{x} - \boldsymbol{x}^*)\right). \tag{C.5}$$

An example of applying Laplace's method to fit a normal approximation to a univariate generalised logistic target is shown in Figure C.1.

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.*

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

$$Z \approx \tilde{Z} = (2\pi)^{\frac{D}{2}} |\boldsymbol{C}|^{\frac{1}{2}} \tilde{p}(\boldsymbol{x}^*). \tag{C.6}$$

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 C.1. 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.

C.2 VARIATIONAL METHODS

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 KL divergence [123]. 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

$$\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] = \int_X \log\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right) \mathrm{d}P, \tag{C.7}$$

which is read as the KL divergence from P to Q. The KL divergence is always non-negative $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] \geq 0$, with equality if and only if $P = Q$ almost everywhere. Intutively the KL divergence gives a measure of how 'close' two measures are[1], however it is not a true distance as it is asymmetric: in general $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q] \neq \mathbb{D}_{\mathrm{KL}}[Q \,\|\, P]$.

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 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 (C.7) we have that

$$\mathbb{D}^{\mu}_{\mathrm{KL}}[p \,\|\, q] = \int_X p(x) \, \log \frac{p(x)}{q(x)} \, \mathrm{d}\mu(x), \tag{C.8}$$

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

$$\mathbb{L}[q] = \log Z - \mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p] = \int_X q(\boldsymbol{x}) \, \log \frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})} \, \mathrm{d}\mu(\boldsymbol{x}), \tag{C.9}$$

1 From an information theory perspective $\mathbb{D}_{\mathrm{KL}}[P \,\|\, Q]$ 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.

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

$$\mathbb{L}[q] \leq \log Z. \tag{C.10}$$

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 EP [149] 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

$$\tilde{p}(\boldsymbol{x}) = \prod_{i \in I} \tilde{p}_i(\boldsymbol{x}). \tag{C.11}$$

An approximate density is defined with an equivalent factorisation

$$q(\boldsymbol{x}) = \prod_{i \in I} q_i(\boldsymbol{x}), \tag{C.12}$$

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

$$\min_{q_j} \mathbb{D}^{\mu}_{\mathrm{KL}}\left[\tilde{p}_j(\boldsymbol{x}) \prod_{i \in I \setminus \{j\}} q_i(\boldsymbol{x}) \,\middle\|\, q_j(\boldsymbol{x}) \prod_{i \in I \setminus \{j\}} q_i(\boldsymbol{x})\right]. \tag{C.13}$$

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 (C.13) 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.

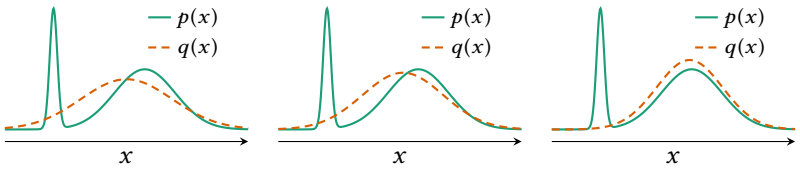
The KL divergence can be considered as a special case of a broader class of α-divergences. In particular the *Rényi divergence* [71, 191] of order $\alpha > 0, \alpha \neq 1$ between two probability measures P and Q with probability densities $p = \frac{\mathrm{d}P}{\mathrm{d}\mu}$ and $q = \frac{\mathrm{d}Q}{\mathrm{d}\mu}$ on a space X is defined as

$$\mathbb{D}_{\alpha}[P \,\|\, Q] = \mathbb{D}^{\mu}_{\alpha}[p \,\|\, q] = \frac{1}{\alpha - 1} \log\left(\int_X p(\boldsymbol{x})^{\alpha} \, q(\boldsymbol{x})^{1-\alpha} \, \mathrm{d}\mu(\boldsymbol{x}) \right). \quad \text{(C.14)}$$

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 (C.14). 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 (C.14) for $\alpha < 0$, in which case we have that $\mathbb{D}_{\alpha}[P \,\|\, Q] = \frac{\alpha}{1-\alpha} \mathbb{D}_{1-\alpha}[Q \,\|\, P] \leq 0$ [132].

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 (C.9), it is observed in [132] that a *variational Rényi bound*, $\mathbb{L}_{\alpha}$, can be defined as

$$\mathbb{L}_{\alpha}[q] = \log Z - \mathbb{D}^{\mu}_{\alpha}[q \,\|\, p] = \frac{1}{1 - \alpha} \log \int_X q(\boldsymbol{x}) \left(\frac{\tilde{p}(\boldsymbol{x})}{q(\boldsymbol{x})} \right)^{1-\alpha} \mathrm{d}\mu(\boldsymbol{x}). \quad \text{(C.15)}$$



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

Figure C.2.: 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 $\alpha > 0$, we have that $\mathbb{D}^{\mu}_{\alpha}[q \,\|\, p] \geq 0$ 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 $\mathbb{D}^{\mu}_{\alpha}[q \,\|\, p]$. For $\alpha < 0$ we have instead that $\mathbb{D}^{\mu}_{\alpha}[q \,\|\, p] \leq 0$ and so $\mathbb{L}_{\alpha}$ is an upper bound on $\log Z$ and that we should minimise $\mathbb{L}_{\alpha}$ to minimise $\mathbb{D}^{\mu}_{1-\alpha}[p \,\|\, q]$ (note the swapped order of the density arguments). An equivalent observation of the possibility of upper bounding $\log Z$ is made in [61] with a reparameterised version of (C.15) 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 [30]. 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 approxim-

ate densities similar to $\mathbb{D}^{\mu}_{\mathrm{KL}}[q \,\|\, p]$, with the solutions becoming increasingly dispersed as α becomes lower).

Figure C.2 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 C.2a 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 C.2c 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 C.2b shows an approximate density fitted by minimising the Rényi divergence (C.14) 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 C.2a and C.2c. 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

$$q(\boldsymbol{x}) = \prod_{i \in I} q_i(\boldsymbol{x}_i). \tag{C.16}$$

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 [177, 207], 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 [174]. Despite the limitations in representational capacity imposed by the independence assumption, because of its computational tractability mean-field variational inference methods remain very popular [31], with mean-field approximations allowing use of a

particularly simple algorithm for optimising the standard variational objective (C.9), *co-ordinate ascent variational inference* [30, 31].

A more recent alternative to traditional mean-field variational methods, is to assume a fixed parametric form for the 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 $\boldsymbol{\theta}$ [99, 122, 171, 187, 205]. 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 (C.9) we have that

$$\ell(\boldsymbol{\theta}) = \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \log \frac{\tilde{p}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \, \mu(\mathrm{d}\boldsymbol{x}). \tag{C.17}$$

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

$$\frac{\partial q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} = q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \tag{C.18}$$

$$\text{and} \qquad \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \, \mu(\mathrm{d}\boldsymbol{x}) = 0, \tag{C.19}$$

the gradient of (C.17) with respect $\boldsymbol{\theta}$ can be expressed as

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \int_X q_{\boldsymbol{\theta}}(\boldsymbol{x}) \frac{\partial \log q_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}} \log \frac{\tilde{p}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{x})} \, \mu(\mathrm{d}\boldsymbol{x}). \tag{C.20}$$

Typically both of the integrals in (C.17) and (C.20) 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 [193] 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 (C.20) 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 [187] proposes using two forms of variance reduction to compute more efficient gradient 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 ADVI algorithm of [122] 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 $\mathbf{g}_{\boldsymbol{\theta}} : U \to X$ and a distribution R on U which has a density ρ which does not depend on $\boldsymbol{\theta}$ such that

$$q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \rho\Big(\mathbf{g}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x})\Big)\left|\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}^{-1}}{\partial \boldsymbol{x}}\right|. \tag{C.21}$$

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 $\boldsymbol{x} = \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{u})$ 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

$$\ell(\boldsymbol{\theta}) = \int_U \rho(\boldsymbol{u})\Big(\log(\tilde{p} \circ \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{u})) + \log\left|\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{u}}\right| - \log \rho(\boldsymbol{u})\Big)\, \mathrm{d}\boldsymbol{u} \tag{C.22}$$

with a corresponding gradient expression

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}} = \int_U \rho(\boldsymbol{u})\left(\frac{\partial}{\partial \boldsymbol{\theta}} \log(\tilde{p} \circ \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{u})) + \frac{\partial}{\partial \boldsymbol{\theta}} \log\left|\frac{\partial \mathbf{g}_{\boldsymbol{\theta}}}{\partial \boldsymbol{u}}\right|\right) \mathrm{d}\boldsymbol{u}. \tag{C.23}$$

As suggested by the name ADVI uses automatic differentiation to calculate the gradient expression inside the parentheses in (C.23), with importantly this requiring propagation of derivatives through the target density function $\tilde{p}$ as well as the transformation $\mathbf{g}_{\boldsymbol{\theta}}$. 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 empirically in [122] 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.

BIBLIOGRAPHY

[1] David H Ackley, Geoffrey E Hinton and Terrence J Sejnowski. 'A learning algorithm for Boltzmann machines'. In: *Cognitive science* 9.1 (1985), pp. 147–169.

[2] 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.

[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] 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.

[10] 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).

[11] Eric Barth, Krzysztof Kuczera, Benedict Leimkuhler and Robert D Skeel. 'Algorithms for constrained molecular dynamics'. In: *Journal of computational chemistry* (1995).

[12] Simon Barthelmé and Nicolas Chopin. 'Expectation propagation for likelihood-free inference'. In: *Journal of the American Statistical Association* 109.505 (2014), pp. 315–333.

[13] Friedrich L Bauer. 'Computational graphs and rounding error'. In: *SIAM Journal on Numerical Analysis* 11.1 (1974), pp. 87–96.

[14] 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).

[15] Mark A Beaumont. 'Estimation of population growth or decline in genetically monitored populations'. In: *Genetics* 164.3 (2003), pp. 1139–1160.

[16] Mark A Beaumont, Wenyang Zhang and David J Balding. 'Approximate Bayesian computation in population genetics'. In: *Genetics* (2002).

[17] 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.

[18] 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.

[19] Gundula Behrens, Nial Friel and Merrilee Hurn. 'Tuning tempered transitions'. In: *Statistics and computing* (2012).

[20] Jarle Berntsen, Terje O Espelid and Alan Genz. 'An adaptive algorithm for the approximate calculation of multiple integrals'. In: *ACM Transactions on Mathematical Software (TOMS)* 17.4 (1991), pp. 437–451.

[21] MJ Betancourt. 'Adiabatic Monte Carlo'. In: *arXiv preprint arXiv:1405.3489* (2014).

[22] Michael Betancourt. 'A general metric for Riemannian manifold Hamiltonian Monte Carlo'. In: *Geometric science of information*. Springer, 2013.

[23] 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.

[24] Michael Betancourt. 'A Conceptual Introduction to Hamiltonian Monte Carlo'. In: *arXiv preprint arXiv:1701.02434* (2017).

[25] Michael Betancourt. 'The Convergence of Markov chain Monte Carlo Methods: From the Metropolis method to Hamiltonian Monte Carlo'. In: *arXiv preprint arXiv:1706.01520* (2017).

[26] Michael Betancourt and Mark Girolami. 'Hamiltonian Monte Carlo for hierarchical models'. In: *Current trends in Bayesian methodology with applications* 79 (2015), p. 30.

[27] Michael Betancourt, Simon Byrne, Sam Livingstone, Mark Girolami et al. 'The geometric foundations of Hamiltonian Monte Carlo'. In: *Bernoulli* 23.4A (2017), pp. 2257–2298.

[28] Joris Bierkens. 'Non-reversible Metropolis–Hastings'. In: *Statistics and Computing* 26.6 (2016), pp. 1213–1228.

[29] George D Birkhoff. 'Proof of the ergodic theorem'. In: *Proceedings of the National Academy of Sciences* 17.12 (1931), pp. 656–660.

[30] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006. ISBN: 9780387310732.

[31] 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).

[32] Michael GB Blum. 'Approximate Bayesian computation: a nonparametric perspective'. In: *Journal of the American Statistical Association* 105.491 (2010), pp. 1178–1187.

[33] 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.

[34] Georges Bonnet. 'Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire'. In: *Annals of Telecommunications* 19.9 (1964), pp. 203–220.

[35] Massimiliano Bonomi, Alessandro Barducci and Michele Parrinello. 'Reconstructing the equilibrium Boltzmann distribution from well-tempered metadynamics'. In: *Journal of computational chemistry* (2009).

[36] 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.

[37] 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.

[38] 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.

[39] Wray L Buntine. 'Operations for learning with graphical models'. In: *Journal of artificial intelligence research* (1994).

[40] Yuri Burda, Roger Grosse and Ruslan Salakhutdinov. 'Importance weighted autoencoders'. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[41] Simon Byrne and Mark Girolami. 'Geodesic Monte Carlo on embedded manifolds'. In: *Scandinavian Journal of Statistics* (2013).

[42] David Carlson, Patrick Stinson, Ari Pakman and Liam Paninski. 'Partition Functions from Rao-Blackwellized Tempered Sampling'. In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016.

[43] 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).

[44] KS Chan. 'Asymptotic behavior of the Gibbs sampler'. In: *Journal of the American Statistical Association* 88.421 (1993), pp. 320–326.

[45] 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.

[46] 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.

[47] Tianqi Chen, Emily Fox and Carlos Guestrin. 'Stochastic Gradient Hamiltonian Monte Carlo'. In: *Proceedings of the 31st International Conference on Machine Learning*. 2014.

[48] Nicolas Chopin and Sumeetpal S Singh. 'On particle Gibbs sampling'. In: *Bernoulli* 21.3 (2015), pp. 1855–1883.

[49] 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.

[50] 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.

[51] 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.

[52] H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1946.

[53] 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).

[54] 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.

[55] Philip J. Davis and Philip Rabinowitz. *Numerical Integration*. Blaisdell Publishing Company, 1967.

[56] Nando De Freitas, Pedro Højen-Sørensen, Michael I Jordan and Stuart Russell. 'Variational MCMC'. In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. 2001.

[57] 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.

[58] George Deligiannidis, Arnaud Doucet, Michael K Pitt and Robert Kohn. 'The Correlated Pseudo-Marginal Method'. In: *arXiv preprint arXiv:1511.04992* (2015).

[59] Persi Diaconis, Susan Holmes and Radford M Neal. 'Analysis of a nonreversible Markov chain sampler'. In: *Annals of Applied Probability* (2000), pp. 726–752.

[60] 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.

[61] 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).

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] Simon Duane, Anthony D Kennedy, Brian J Pendleton and Duncan Roweth. 'Hybrid Monte Carlo'. In: *Physics Letters B* (1987).

[67] 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.

[68] David J Earl and Michael W Deem. 'Parallel tempering: theory, applications, and new perspectives'. In: *Physical Chemistry Chemical Physics* (2005).

[69] 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.

[70] Vassiliy A Epanechnikov. 'Non-parametric estimation of a multivariate probability density'. In: *Theory of Probability & Its Applications* 14.1 (1969), pp. 153–158.

[71] Tim van Erven and Peter Harremos. 'Rényi divergence and Kullback-Leibler divergence'. In: *IEEE Transactions on Information Theory* 60.7 (2014), pp. 3797–3820.

[72] Herbert Federer. *Geometric measure theory*. Springer, 2014.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] Alan E Gelfand and Adrian FM Smith. 'Sampling-based approaches to calculating marginal densities'. In: *Journal of the American Statistical Association* (1990).

[79] 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.

[80] Andrew Gelman and Jennifer Hill. 'Data analysis using regression and multilevel/hierarchical models'. In: Cambridge University Press, 2006. Chap. 12: Multilevel Linear Models: the Basics.

[81] 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.

[82] Andrew Gelman and Xiao-Li Meng. 'Simulating normalizing constants: From importance sampling to bridge sampling to path sampling'. In: *Statistical science* (1998).

[83] Andrew Gelman and Donald B Rubin. 'Inference from iterative simulation using multiple sequences'. In: *Statistical science* (1992), pp. 457–472.

[84] 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).

[85] Charles J Geyer. 'Markov chain Monte Carlo maximum likelihood'. In: *Computer Science and Statistics* (1991).

[86] Charles J Geyer. *Markov chain Monte Carlo lecture notes (Statistics 8931)*. Tech. rep. University of Minnesota, 1998.

[87] Charles J Geyer. 'The Metropolis-Hastings-Green Algorithm'. 2003. URL: http://www.stat.umn.edu/geyer/f05/8931/bmhg.pdf.

[88] Charles J Geyer and Elizabeth A Thompson. 'Annealing Markov chain Monte Carlo with applications to ancestral inference'. In: *Journal of the American Statistical Association* (1995).

[89] W. Gilchrist. *Statistical Modelling with Quantile Functions*. CRC Press, 2000.

[90] Wally R Gilks, Andrew Thomas and David J Spiegelhalter. 'A language and program for complex Bayesian modelling'. In: *The Statistician* (1994), pp. 169–177.

[91] Walter R Gilks and Pascal Wild. 'Adaptive rejection sampling for Gibbs sampling'. In: *Applied Statistics* (1992), pp. 337–348.

[92] 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.

[93] Gianpaolo Gobbo and Benedict J Leimkuhler. 'Extended Hamiltonian approach to continuous tempering'. In: *Physical Review E* (2015).

[94] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[95] 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.

[96] 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.

[97] 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.

[98] Christian Gourieroux, Alain Monfort and Eric Renault. 'Indirect inference'. In: *Journal of applied econometrics* 8.S1 (1993), S85–S118.

[99] Alex Graves. 'Practical Variational Inference for Neural Networks'. In: *Advances in Neural Information Processing Systems 24*. 2011, pp. 2348–2356.

[100] Todd L Graves. 'Automatic step size selection in random walk Metropolis algorithms'. In: *arXiv preprint arXiv:1103.5986* (2011).

[101] Peter J Green. 'Reversible jump Markov chain Monte Carlo computation and Bayesian model determination'. In: *Biometrika* (1995), pp. 711–732.

[102] Roger Grosse, Zoubin Ghahramani and Ryan P Adams. 'Sandwiching the marginal likelihood using bidirectional Monte Carlo'. In: *arXiv preprint arXiv:1511.02543* (2015).

[103] 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.

[104] Heikki Haario, Eero Saksman and Johanna Tamminen. 'An adaptive Metropolis algorithm'. In: *Bernoulli* (2001), pp. 223–242.

[105] 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.

[106] J.A. Hartigan. *Bayes theory*. Springer series in statistics. Springer-Verlag, 1983. ISBN: 9783540908838.

[107] 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).

[108] 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.

[109] 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.

[110] W Keith Hastings. 'Monte Carlo sampling methods using Markov chains and their applications'. In: *Biometrika* (1970).

[111] 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.

[112] 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.

[113] 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).

[114] Akihisa Ichiki and Masayuki Ohzeki. 'Violation of detailed balance accelerates relaxation'. In: *Physical Review E* 88.2 (2013), p. 020101.

[115] Ernst Ising. 'Beitrag zur theorie des ferromagnetismus'. In: *Zeitschrift für Physik A Hadrons and Nuclei* 31.1 (1925), pp. 253–258.

[116] 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.

[117] AD Kennedy and Julius Kuti. 'Noise without noise: a new Monte Carlo method'. In: *Physical review letters* 54.23 (1985), p. 2473.

[118] Ross Kindermann and Laurie Snell. *Markov random fields and their applications*. American Mathematical Society, 1980.

[119] Diederik P Kingma and Max Welling. 'Auto-Encoding Variational Bayes'. In: *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*. 2013.

[120] 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.

[121] 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.

[122] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman and David M Blei. 'Automatic Differentiation Variational Inference'. In: *arXiv preprint arXiv:1603.00788* (2016).

[123] Solomon Kullback and Richard A Leibler. 'On information and sufficiency'. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86.

[124] Alessandro Laio and Michele Parrinello. 'Escaping free-energy minima'. In: *Proceedings of the National Academy of Sciences* (2002).

[125] Brenden M Lake, Ruslan Salakhutdinov and Joshua B Tenenbaum. 'Human-level concept learning through probabilistic program induction'. In: *Science* (2015).

[126] Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* (1998).

[127] 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.

[128] Benedict J Leimkuhler and Robert D Skeel. 'Symplectic numerical integrators in constrained Hamiltonian systems'. In: *Journal of Computational Physics* (1994).

[129] Benedict Leimkuhler and Charles Matthews. 'Efficient molecular dynamics using geodesic integration and solvent–solute splitting'. In: *Proc. R. Soc. A*. The Royal Society. 2016.

[130] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian dynamics*. Cambridge University Press, 2004.

[131] Tony Lelièvre, Mathias Rousset and Gabriel Stoltz. 'Langevin dynamics with constraints and computation of free energy differences'. In: *Mathematics of computation* (2012).

[132] Yingzhen Li and Richard E Turner. 'Rényi divergence variational inference'. In: *Advances in Neural Information Processing Systems*. 2016.

[133] 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.

[134] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[135] 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.

[136] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[137] 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.

[138] Jean-Michel Marin, Pierre Pudlo, Christian P Robert and Robin J Ryder. 'Approximate Bayesian computational methods'. In: *Statistics and Computing* (2012).

[139] Enzo Marinari and Giorgio Parisi. 'Simulated tempering: a new Monte Carlo scheme'. In: *Europhysics Letters* (1992).

[140] Paul Marjoram, John Molitor, Vincent Plagnol and Simon Tavaré. 'Markov chain Monte Carlo without likelihoods'. In: *Proceedings of the National Academy of Sciences* (2003).

[141] George Marsaglia. 'Random numbers fall mainly in the planes'. In: *Proceedings of the National Academy of Sciences* 61.1 (1968), pp. 25–28.

[142] 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.

[143] Edward Meeds, Robert Leenders and Max Welling. 'Hamiltonian ABC'. In: *Proceedings of 31st Conference of Uncertainty in Artificial Intelligence*. 2015.

[144] Ted Meeds and Max Welling. 'Optimization Monte Carlo: Efficient and Embarrassingly Parallel Likelihood-Free Inference'. In: *Advances in Neural Information Processing Systems*. 2015.

[145] 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.

[146] 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).

[147] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 1993.

[148] GN Mil'shtejn. 'Approximate integration of stochastic differential equations'. In: *Theory of Probability & Its Applications* 19.3 (1975), pp. 557–562.

[149] Thomas P Minka. 'Expectation propagation for approximate Bayesian inference'. In: *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence* (2001).

[150] Shakir Mohamed and Balaji Lakshminarayanan. 'Learning in Implicit Generative Models'. In: *Proceedings of the International Conference on Learning Representations*. 2017.

[151] 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.

[152] J. J. Moré, B. S. Garbow and K. E. Hillstrom. *User Guide for MINPACK-1*. ANL-80-74, Argonne National Laboratory. 1980.

[153] Alexander Moreno, Tameem Adel, Edward Meeds, James M Rehg and Max Welling. 'Automatic Variational ABC'. In: *arXiv preprint arXiv:1606.08549* (2016).

[154] Iain Murray. 'Advances in Markov chain Monte Carlo methods'. PhD thesis. University College London, University of London, 2007.

[155] Iain Murray and Ryan P Adams. 'Slice sampling covariance hyperparameters of latent Gaussian models'. In: *Advances in Neural Information Processing Systems*. 2010.

[156] Iain Murray, Ryan Prescott Adams and David J.C. MacKay. 'Elliptical slice sampling'. In: JMLR: W&CP 9 (2010), pp. 541–548.

[157] 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.

[158] 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.

[159] Peter Neal and Clement Lee. 'Optimal scaling of the independence sampler: Theory and Practice'. In: *arXiv preprint arXiv:1511.04334* (2015).

[160] Radford M Neal. 'Sampling from multimodal distributions using tempered transitions'. In: *Statistics and Computing* (1996).

[161] Radford M Neal. *Markov chain Monte Carlo methods based on 'slicing' the density function*. Tech. rep. 9722. Department of Statistics, University of Toronto, 1997.

[162] Radford M Neal. 'Annealed importance sampling'. In: *Statistics and Computing* (2001).

[163] Radford M Neal. 'Slice sampling'. In: *Annals of statistics* (2003).

[164] Radford M Neal. 'Improving asymptotic variance of MCMC estimators: Non-reversible chains are better'. In: *arXiv preprint math/0407281* (2004).

[165] Radford M Neal. 'MCMC using Hamiltonian dynamics'. In: *Handbook of Markov Chain Monte Carlo*. Chapman & Hall/CRC, 2011. Chap. 5, pp. 113–162.

[166] John von Neumann. 'Various techniques used in connection with random digits'. In: *National Bureau of Standards applied mathematics series* 3 (1951), pp. 36–38.

[167] 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.

[168] John F Nolan. 'Analytical differentiation on a digital computer'. PhD thesis. Massachusetts Institute of Technology, 1953.

[169] Sheehan Olver and Alex Townsend. 'Fast inverse transform sampling in one and two dimensions'. In: *arXiv preprint arXiv:1307.1223* (2013).

[170] Art B. Owen. 'Importance sampling'. In: *Monte Carlo theory, methods and examples*. 2013. URL: http://statweb.stanford.edu/~owen/mc/Ch-var-is.pdf.

[171] 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.

[172] 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.

[173] Omiros Papaspiliopoulos, Gareth O Roberts and Martin Sköld. 'A general framework for the parametrization of hierarchical models'. In: *Statistical Science* (2007), pp. 59–73.

[174] G. Parisi. *Statistical Field Theory*. Advanced book classics. Avalon Publishing, 1998. ISBN: 9780738200514. URL: https://books.google.co.uk/books?id=y0-8xQOw6FcC.

[175] Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.

[176] Peter H Peskun. 'Optimum Monte-Carlo sampling using Markov chains'. In: *Biometrika* 60.3 (1973), pp. 607–612.

[177] Carsten Peterson and James R Anderson. 'A mean field theory learning algorithm for neural networks'. In: *Complex systems* (1987).

[178] Michael Pitt, Ralph Silva, Paolo Giordani and Robert Kohn. 'Auxiliary particle filtering within adaptive Metropolis-Hastings sampling'. In: *arXiv preprint arXiv:1006.1914* (2010).

[179] 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.

[180] 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.

[181] M. J. D. Powell. 'Numerical Methods for Nonlinear Algebraic Equations'. In: Gordon and Breach, 1970. Chap. A Hybrid Method for Nonlinear Equations.

[182] Dennis Prangle. 'Summary statistics in approximate Bayesian computation'. In: *arXiv preprint arXiv:1512.05633* (2015).

[183] Robert Price. 'A useful theorem for nonlinear devices having Gaussian inputs'. In: *IRE Transactions on Information Theory* 4.2 (1958), pp. 69–72.

[184] 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.

[185] 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.

[186] Adrian E Raftery and Steven Lewis. *How many iterations in the Gibbs sampler?* Tech. rep. Washington University, Seattle, Department of Statistics, 1991.

[187] Rajesh Ranganath, Sean Gerrish and David Blei. 'Black Box Variational Inference'. In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. 2014.

[188] C Radhakrishna Rao. 'Information and the accuracy attainable in the estimation of statistical parameters'. In: *Breakthroughs in statistics*. Springer, 1992, pp. 235–247.

[189] 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.

[190] 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).

[191] 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.

[192] 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.

[193] Herbert Robbins and Sutton Monro. 'A stochastic approximation method'. In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[194] 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).

[195] Christian Robert and George Casella. 'A short history of Markov Chain Monte Carlo: subjective recollections from incomplete data'. In: *Statistical Science* (2011).

[196] Gareth O Roberts and Jeffrey S Rosenthal. 'Optimal scaling for various Metropolis-Hastings algorithms'. In: *Statistical science* 16.4 (2001), pp. 351–367.

[197] Gareth O Roberts and Jeffrey S Rosenthal. 'General state space Markov chains and MCMC algorithms'. In: *Probability Surveys* 1 (2004), pp. 20–71.

[198] 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.

[199] 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.

[200] Jeffrey S Rosenthal et al. 'Optimal proposal distributions and adaptive MCMC'. In: *Handbook of Markov Chain Monte Carlo* (2011), pp. 93–112.

[201] 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.

[202] 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).

[203] Ruslan Salakhutdinov and Iain Murray. 'On the quantitative analysis of deep belief networks'. In: *Proceedings of the 25th International Conference on Machine learning*. 2008.

[204] Tim Salimans, Diederik P Kingma and Max Welling. 'Markov chain Monte Carlo and variational inference: Bridging the gap'. In: *International Conference on Machine Learning*. 2015.

[205] 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.

[206] John Salvatier, Thomas V Wiecki and Christopher Fonnesbeck. 'Probabilistic programming in Python using PyMC3'. In: *PeerJ Computer Science* (2016).

[207] Lawrence K Saul, Tommi Jaakkola and Michael I Jordan. 'Mean field theory for sigmoid belief networks'. In: *Journal of Artificial Intelligence Research* (1996).

[208] Chris Sherlock, Alexandre Thiery and Anthony Lee. 'Pseudo-marginal Metropolis–Hastings using averages of unbiased estimators'. In: *arXiv preprint arXiv:1610.09788* (2016).

[209] 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.

[210] 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.

[211] 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.

[212] Bert Speelpenning. 'Compiling Fast Partial Derivatives of Functions Given by Algorithms'. PhD thesis. University of Illinois at Urbana-Champaign, 1980.

[213] 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.

[214] Hidemaro Suwa and Synge Todo. 'Markov chain Monte Carlo method without detailed balance'. In: *Physical review letters* 105.12 (2010), p. 120603.

[215] Robert H Swendsen and Jian-Sheng Wang. 'Replica Monte Carlo simulation of spin-glasses'. In: *Physical Review Letters* (1986).

[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] 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.

[219] Madeleine B Thompson. 'A comparison of methods for computing autocorrelation time'. In: *arXiv preprint arXiv:1011.0175* (2010).

[220] Luke Tierney. 'Markov chains for exploring posterior distributions'. In: *The Annals of Statistics* (1994), pp. 1701–1728.

[221] 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.

[222] Dustin Tran, Rajesh Ranganath and David M Blei. 'Deep and Hierarchical Implicit Models'. In: *arXiv preprint arXiv:1702.08896* (2017).

[223] Minh-Ngoc Tran, David J Nott and Robert Kohn. 'Variational Bayes with intractable likelihood'. In: *Journal of Computational and Graphical Statistics* (2017).

[224] John W. Tukey. *Practical Relationship Between the Common Transformations of Percentages or Fractions and of Amounts*. Technical Report 36. Statistical Research Group,Princeton, 1960.

[225] 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.

[226] Stanislaw Ulam and Nicholas Metropolis. 'The Monte Carlo method'. In: *Journal of the American Statistical Association* 44.247 (1949), pp. 335–341.

[227] 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.

[228] Gunter Weiss and Arndt von Haeseler. 'Inference of population history using a likelihood approach'. In: *Genetics* 149.3 (1998), pp. 1539–1546.

[229] Max Welling and Yee W Teh. 'Bayesian learning via stochastic gradient Langevin dynamics'. In: *Proceedings of the 28th International Conference on Machine Learning*. 2011.

[230] Robert Edwin Wengert. 'A simple automatic derivative evaluation program'. In: *Communications of the ACM* 7.8 (1964), pp. 463–464.

[231] 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).

[232] Simon N Wood. 'Statistical inference for noisy nonlinear ecological dynamic systems'. In: *Nature* 466.7310 (2010), pp. 1102–1104.

[233] Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov and Roger Grosse. 'On the Quantitative Analysis of Decoder-Based Generative Models'. In: *arXiv preprint arXiv:1611.04273* (2016).

[234] Reza Zadeh. *Twitter status*. Dec. 2016. URL: https://twitter.com/Reza_Zadeh/status/811130294291963904.

[235] Emilio Zappa, Miranda Holmes-Cerfon and Jonathan Goodman. 'Monte Carlo on manifolds: sampling densities and integrating functions'. In: *arXiv preprint arXiv:1702.08446* (2017).

[236] Yichuan Zhang, Zoubin Ghahramani, Amos J Storkey and Charles A Sutton. 'Continuous relaxations for discrete Hamiltonian Monte Carlo'. In: *Advances in Neural Information Processing Systems*. 2012.

[237] Oliver Zobay. 'Mean field inference for the Dirichlet process mixture model'. In: *Electronic Journal of Statistics* (2009).