Bayesian Inference for Challenging Scientific Models

James A. Ritchie

Doctor of Philosophy
Centre for Doctoral Training in Data Science
School of Informatics
University of Edinburgh
2023

Abstract

Advances in technology and computation have led to ever more complicated scientific models of phenomena across a wide variety of fields. Many of these models present challenges for Bayesian inference, as a result of computationally intensive likelihoods, high-dimensional parameter spaces or large dataset sizes. In this thesis we show how we can apply developments in probabilistic machine learning and statistics to do inference with examples of these types of models. As a demonstration of an applied inference problem involving a non-trivial likelihood computation, we show how a combination of optimisation and MCMC methods along with careful consideration of priors can be used to infer the parameters of an ODE model of the cardiac action potential.

We then consider the problem of pileup, a phenomenon that occurs in astronomy when using CCD detectors to observe bright sources. It complicates the fitting of even simple spectral models by introducing an observation model with a large number of continuous and discrete latent variables that scales with the size of the dataset. We develop an MCMC-based method that can work in the presence of pileup by explicitly marginalising out discrete variables and using adaptive HMC on the remaining continuous variables. We show with synthetic experiments that it allows us to fit spectral models in the presence of pileup without biasing the results. We also compare it to neural Simulation-Based Inference approaches, and find that they perform comparably to the MCMC-based approach whilst being able to scale to larger datasets.

As an example of a problem where we wish to do inference with extremely large datasets, we consider the Extreme Deconvolution method. The method fits a probability density to a dataset where each observation has Gaussian noise added with a known sample-specific covariance, originally intended for use with astronomical datasets. The existing fitting method is batch EM, which would not normally be applied to large datasets such as the Gaia catalog containing noisy observations of a billion stars. In this thesis we propose two minibatch variants of extreme deconvolution, based on an online variation of the EM algorithm, and direct gradient-based optimisation of the log-likelihood, both of which can run on GPUs. We demonstrate that these methods provide faster fitting, whilst being able to scale to much larger models for use with larger datasets.

We then extend the extreme deconvolution approach to work with non-Gaussian noise, and to use more flexible density estimators such as normalizing flows. Since both adjustments lead to an intractable likelihood, we resort to amortized variational inference in order to fit them. We show that for some datasets that flows can outperform Gaussian mixtures for extreme deconvolution, and that fitting with non-Gaussian noise is now possible.

Lay Summary

Modern scientific research often involves making computer models of natural phenomena of interest. These models are used to check scientific theories, and to make informed predictions about the world. Often these models have a number of variables or parameters within them that need to be set. We can infer the values of these parameters by comparing the output of the model to data that we observe, then finding the values of the parameters that make the output look most similar to the real data. This is referred to as fitting the model to data. Bayesian inference is a particular branch of inference that is very useful for certain modelling problems. It allows us to measure the degree of uncertainty in both our inferences about the values of parameters and in the predictions we make with these models.

It also allows us to make use of any prior beliefs we may have about the values of particular parameters even before we see any data. For example, a scientist may know that a particular variable must be limited to a certain range of values, or that it must not stray too far from a particular central value. Bayesian inference allows us to incorporate this knowledge into our inferences in a principled manner.

Despite this, many scientific models can cause problems for Bayesian inference. This might be because we cannot do some of the calculations inside them exactly, there are too many variables that we want to infer at once, or simply because the necessary calculations take too long even for a powerful computer. In this thesis we consider several of these sorts of problems, and we show that by making use of recent developments in machine learning and statistics, we can create workable solutions for some of these inference problems.

As an example problem, astronomers are often interested in counting the number and energies of photons arriving at a telescope in order to characterise stars, galaxies and other sources that emitted them. Unfortunately, a commonly used sensor in many telescopes does not let them do this. Instead, it only reports the total amount of energy received from the photons over a given amount of time. Given a model of this process, it is hard to infer the parameters of it. Even if there are only a few parameters describing the star or galaxy, there are a very large number of parameters describing all of the different possible ways the photons could have produced the total energy we observed.

We show that by explicitly counting out all the possible numbers of photons, we can then use a method from statistics that can work with large numbers of parameters. We can use it to infer the possible energy values of each possible photon. This information can then be used to infer the parameters describing the star. We also show that it is possible to train a common machine learning method called a *neural network* to learn the relationship between the parameters of the star or galaxy and simulated data produced by the model. When given real data, these neural networks can then tell us what they think the parameter values most likely to have produced the data are.

Another problem we consider is how to fit a particular type of model to data produced by the Gaia satellite. The satellite has made imprecise measurements of the locations and speeds of over a billion stars in our galaxy, but the current computer code to fit models to this type of imprecise data cannot cope with the extremely large number of measurements. Fortunately, fitting models to very large amounts of data is a common task in machine learning. We show that by adapting some of the methods used to do this, it is possible to fit this particular type of model in much less time than the previous method, and with much larger datasets.

We then demonstrate that it is possible to fit a different type of model called a *normalising flow* to the imprecise Gaia data. These *normalising flows* are commonly used in machine learning, and make use of neural networks internally. They do a better job of making predictions about the data than the previous model, and we show that they can be used to refine the imprecise measurement of the distance to a particular cluster of stars in our galaxy. We also show that they can be used to predict the distance to a star based on its colour.

Acknowledgements

Even under the best possible circumstances, a PhD requires the support of many people to complete successfully. This has never been been more true than when a large proportion of the work involved took place during a global pandemic.

Firstly, I need to thank Iain Murray for advising me throughout my journey to becoming a researcher. His insightful questions and eye for detail were critical in helping clarify my thinking. Whenever the research seemed to have reached a dead-end, he always had a suggestion for what to try next, and he was almost always right.

Michael Gutmann and Joe Zuntz kindly agreed to form my review panel. They offered their time to answer my questions, read my proposals and helped keep me on track by ensuring I turned my vague ideas into a concrete plan that would lead to completion. Daniel Mortlock and Amos Storkey served as my examiners, and I am grateful to them for their questions and suggestions for improvements.

Daniela Huppenkothen provided the initial idea to work on inference for models with pileup. David Hogg confirmed that applying normalising flows to deconvolution was a worthwhile idea, and helped us get started with the Gaia catalogue. Tim Dockhorn was crucial in helping get an earlier version of the variational inference work done as a workshop paper, and the results of his experiments demonstrated that the idea was worth persevering with. Without them this document would be a literature review and not a thesis.

The production of this thesis and all of the experiments within relied almost exclusively on open-source software. The efforts of the countless contributors to the many packages I used made this research feasible.

The staff of the Centre for Doctoral Training in Data Science and the School of Informatics worked hard to ensure we could all focus on our research, and I will be forever thankful. I am particularly grateful to the CDT administrators Sally Galloway, Sio Carroll and Sandra Nicol for making the programme run smoothly.

Colleagues, acquaintances and friends from my office, my research group, the CDT, Informatics and beyond helped make the days fly by. Our conversations during coffee breaks, over lunch, and elsewhere never failed to cheer me up. My only regret during this PhD is that so many of them could not be

in-person.

I am grateful to my parents Brian and Nicola, and my sister Anna. They supported my decision to leave stable paid employment and return to academia. More importantly, they raised me as part of a family that taught me to appreciate the value of learning for its own sake.

Finally, the most important person throughout this entire experience has been Lucy. I would not have finished this thesis without her continued support. She started this journey with me by moving up to Edinburgh, and she finished it with me as my wife. Thank you.

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

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.

To Lucy.

Table of Contents

Chapter 1

Introduction

Bayesian inference has been used effectively to estimate the parameters of scientific models in a wide variety of fields, including astronomy (Bailer-Jones et al. 2021), virology (Obermeyer et al. 2022), economics (Meager 2019) and public health (Burnett et al. 2018). Despite this widespread usage, there are still many models which present issues for the application of Bayesian inference as a result of computationally intensive likelihoods, high-dimensional parameter spaces or large dataset sizes. Fortunately, recent advances in statistics and probabilistic machine learning offer potential solutions to the problems these models present. In this thesis we consider some of the issues around applying these newer techniques to several models which present challenges for Bayesian inference.

Many Bayesian inference techniques are not trivial to implement. A huge driver in the adoption of Bayesian inference has therefore been the availability of high-quality software packages. In particular, probabilistic programming languages such as BUGS, Stan, PyMC3 and Pyro aim to allow practitioners to concentrate on modelling by providing an interface for specifying models, whilst automating as much of the inference procedure as possible (Lunn et al. 2000; Carpenter et al. 2017; Salvatier, Wiecki, and Fonnesbeck 2016; Bingham et al. 2019). A key enabler of these probabilistic programming languages and other inference approaches is the availability of algorithms that require minimal interaction from the user in order to run (Hoffman and Gelman 2014; Kucukelbir et al. 2017; Foreman-Mackey et al. 2013). They also typically integrate diagnostics that can alert the user to issues with the inference procedure. Any newer approaches to inference should aim to replicate these goals of abstracting away the mechanics of inference and providing self-tuning methods and diagnostics.

A frequent issue in many modelling problems is that there are often multiple techniques available to do inference, but a lack of comparative work to guide practitioners as to which method to choose for a particular problem. As an example, there are several different ways one could approach the problem of inferring the parameters of an ordinary differential equation (ODE) in a Bayesian manner (Girolami 2008; Macdonald and Husmeier 2015; Lueckmann, Goncalves, et al. 2017). We propose a workflow for inferring the parameters of a model of the cardiac action potential by combining existing methods (Simitev and Biktashev 2011). The effectiveness of our approach was validated when it achieved first place in an inference competition which attempted to resolve how best to handle this problem (MacDonald 2018).

As another example of an area where which technique to use is not obvious, we consider the problem of inferring spectral models for astronomy in the presence of a phenomenon known as pileup (Ballet 1999). Pileup complicates the fitting of even simple spectral models by introducing a large variable number of discrete and continuous latent variables that scales with the size of the data. By marginalising out the discrete variables, we show that is possible to sample from the posterior of a small synthetic problem with pileup using Hamiltonian Monte Carlo (HMC, Duane et al. 1987; Neal 2011).

As a comparison, we also consider some newer simulation-based inference (SBI) techniques that make use of neural networks. We consider some of the issues around configuring and checking them, and demonstrate that some of the techniques can produce comparable inferences to the HMC method, whilst being more scalable. We then extend both the HMC and SBI techniques to more realistic spectral models.

As many of the neural SBI techniques lack theoretical guarantees and self-diagnostics, checking their inferences are correct is critical. As an example demonstrating that the flow of ideas in this thesis is not only in the direction from machine learning to statistics, we show that commonly used convergence diagnostics used with Markov chain Monte Carlo (MCMC) methods can also be applied to neural SBI methods. Whilst not a sufficient condition for correctness, these diagnostics can detect issues that would otherwise require much more expensive calibration checks.

Sometimes we only wish to fit relatively simple models, but the explosion in the availability of data renders existing implementations inadequate. New methods are required to allow our inference techniques to scale up to these datasets. As an example, the Hipparcos mission was a satellite that produced noisy astrometric measurements of 118,218 stars (Perryman et al. 1997). Extreme deconvolution is a method for fitting Gaussian mixture models (GMMs) as density estimators to noisy data, originally intended for use with the Hipparcos dataset (Bovy, Hogg, and Roweis 2011).

The successor misson to Hipparcos, Gaia, currently provides around 1.46×10^9 noisy measurements (Babusiaux, Fabricius, et al. 2022). Whilst the existing reference implementation of extreme deconvolution could theoretically handle this dataset using a specialised computer with an extremely large quantity of memory, it would be impractically slow to run, and we would ideally like to be able to fit these sorts of models using more standard workstations. One way of doing this is to borrow ideas from the large-scale optimisation methods commonly used in machine learning (Bottou, Curtis, and Nocedal 2018). By making use of minibatch methods, automatic differentiation and GPU-based computation, we show that we can fit extreme deconvolution models to data from the Gaia catalogue with comparable density estimates, but with much faster convergence and far lower memory requirements. Using simple synthetic examples we also highlight some issues that a practitioner may encounter when fitting these types of model.

With the availability of more data, we can also start to think about adopting more flexible models. Normalising flows are an alternative class of density estimator to GMMs, and have demonstrated the ability to accurately model high-dimensional datasets, including large collections of image data (Papamakarios, Nalisnick, et al. 2021; Kingma and Dhariwal 2018). Unfortunately we cannot use them as a direct replacement for GMMs in the context of density estimation with noisy data, as the standard objective function used by extreme deconvolution only has an analytical solution for GMMs with Gaussian noise. We show instead that we can fit them as density estimators for deconvolution by leveraging advances in amortized variational inference (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014). By fitting normalising flows using this method to an extended sample of the Gaia catalogue, we show that they can provide superior density estimates for certain tasks. We use the fitted models to construct a colour-magnitude diagram (CMD) from the Gaia catalogue, and refine noisy distance measurements to the M67 cluster.

1.1 List of Contributions

1. In Chapter 3, we provide a workflow for inferring the parameter of an ordinary differential equation modelling the cardiac action potential.

2. In Chapter 4, we derive an MCMC-based method that can allow us to do inference of spectral models in the presence of pileup. This allows the use of Bayesian analyses for this problem, a specific example of the more general class of problems involving compound Poisson distributions. We evaluate and check its calibration on a simple synthetic problem. We also use this problem to evaluate newer neural SBI methods, which we find to be comparable to the MCMC-based approach whilst being more scalable. We evaluate the methods on more realistic models. We also show that standard convergence diagnostics used with MCMC methods can also be used with neural SBI methods.

3. In Chapter 5, we propose two minibatch variants of the widely used extreme deconvolution method by using an online version of the expectation-maximisation algorithm and stochastic gradient descent. These improvements allow the method to scale to much larger datasets than was previously possible. Using a selection of astrometric data from the Gaia catalogue, we show that both methods provide comparable density estimates to the existing reference implementation, whilst being much faster to train. An early version of this chapter was published as a workshop paper (Ritchie and Murray 2019).

4. In Chapter 6, we show that the Extreme Deconvolution method can be extended to work with non-Gaussian noise and alternative density estimators by using variational inference. This allows the use of much more flexible density estimators such as normalising flows. We fit a normalising flow to an extended dataset from the Gaia catalogue using this method, which results in better density estimation than a Gaussian mixture model. As an example application of our fitted normalising flow, we use it to produce a denoised Colour-Magnitude Diagram and to refine distance es-

timates to the M67 cluster. An early version of this chapter was published as a workshop paper with Tim Dockhorn as a joint first co-author (Dockhorn et al. 2020). All of the work, writing and experiments included in the chapter has subsequently been redone by us.

Chapter 2

Background

In this chapter we present a review of the statistical and machine learning techniques we will be using in later chapters to enable Bayesian inference for the problems we are considering. We assume some basic familiarity with statistics and machine learning. We start with a brief introduction to Bayesian inference in general, then cover Markov chain Monte Carlo methods as well as some of their diagnostics. Finally we provide a review of density estimation, and cover some recent approaches to Simulation Based Inference that make use of density estimators.

We use the notation $p(\mathbf{x})$ to denote a probability distribution over $\mathbf{x}$ both as an abstract entity in itself, and more specifically to denote the probability density function (PDF) of that distribution. $p(y \mid \mathbf{x})$ denotes a probability distribution over y conditioned on the value of $\mathbf{x}$, and $p_\phi(\mathbf{x})$ denotes a distribution with parameters ϕ that we wish to optimise for a specific task.

2.1 Bayesian Inference

Bayesian inference is a method of inferring the parameters of a probabilistic model given some observed data that we believe to have been generated by the model. We formalise this belief through the *posterior distribution* $p(\theta \mid \mathcal{D})$, a probability distribution over parameters θ given the data $\mathcal{D}$. This posterior distribution can be derived from Bayes' Rule, itself derived from the product and sum rules,

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}. \tag{2.1}$$

It relies on the more general interpretation of probability as a measure of belief rather than the frequency of outcomes in repeated experiments (Jaynes 2003, Chapter 1). The prior distribution $p(\theta)$ represents our beliefs about the parameters before we have seen any data. Bayes rule then represents the logical updating of our beliefs about the parameters after we have seen the data via the likelihood $p(\mathcal{D} \mid \theta)$. The resultant posterior distribution $p(\theta \mid \mathcal{D})$ represents our beliefs about the parameters after we have seen the data.

This use of Bayes' rule gives rise to the name of *Bayesian inference*. Simply using Bayes' rule does not necessarily mean a statistical model is considered Bayesian. As an example, the naive Bayes classifier makes use of Bayes Rule internally to make predictions with uncertainty on unknown labels. It does not use Bayes rule to infer a distribution describing any beliefs about the parameters of the model, and is thus not typically considered "Bayesian" as such (Murphy 2012, Chapter 3). Generally an approach is considered "Bayesian" if it uses the posterior distribution to quantify the uncertainty in our beliefs about some unknown parameters.

Throughout this section we will use a synthetic classification problem as a working example. Figure 2.1 plots 2D data separated into two classes, which we wish to use to infer the parameters of the logistic regression model.

2.1.1 Likelihoods

The likelihood $p(\mathcal{D} \mid \theta)$ is a function of the model parameters θ. Given θ, it describes the probability of observing the data $\mathcal{D}$. It is often expressed as the probability of observing some aspect of $\mathcal{D}$ under some tractable distribution such as a Gaussian or Categorical distribution. The parameters of this tractable distribution are arbitrary functions of the model parameters θ, and possibly also some other aspect of the data $\mathcal{D}$.

For the logistic regression example, the parameters would be a vector of weights $\mathbf{w}$ and a bias term b. Given a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ consisting of N pairs of predictor vectors $\mathbf{x}_i$ and a binary variable y_i indicating a class, the likelihood of the parameters given one pair is

$$p(y_i \mid \mathbf{x}_i, \mathbf{w}, b) = \text{Bernoulli}(y_i \mid \text{logit}^{-1}(\mathbf{w}^T\mathbf{x}_i + b)) \tag{2.2}$$

The Bernoulli distribution is the discrete probability distribution which produces $y = 1$ when sampled with probability q and $y = 0$ with probability

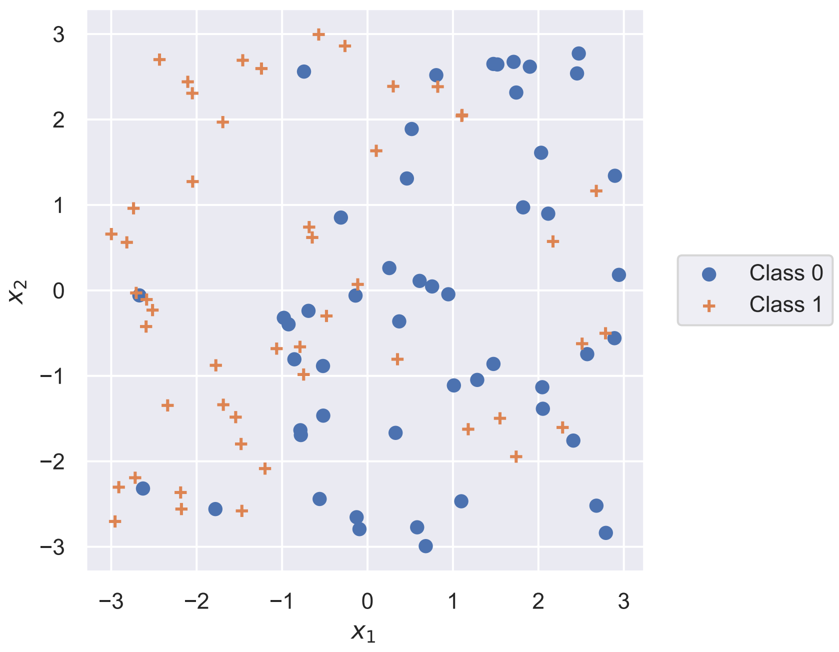


Figure 2.1: Scatter plot of labelled 2-class data which we wish to use to infer the parameters of a logistic regression model.

$1 - q$ (Murphy 2012, Chapter 2). The function logit^{-1} is the inverse logit function (sometimes called the sigmoid function) and is defined as (Murphy 2012, Chapter 1)

$$\text{logit}^{-1}(a) = \frac{1}{1 + e^{-a}}. \tag{2.3}$$

It maps unconstrained values a into the range $[0, 1]$, and is necessary to ensure that the parameter q of the Bernoulli is appropriately constrained.[1] The total likelihood given the entire dataset is

$$p(\mathcal{D} \mid \theta) = \prod_{i=1}^{N} \text{Bernoulli}(y_i \mid \text{logit}^{-1}(\mathbf{w}^T \mathbf{x}_i + b)). \tag{2.4}$$

We consider $\mathbf{x}_i$ to be a known deterministic variable, hence why it appears as a conditioner in $p(y_i \mid \mathbf{x}_i, \mathbf{w}, b)$ despite being part of the data $\mathcal{D}$.

[1]Other functions to do this mapping can be used. The inverse logit function is the so-called *canonical link function* for the Bernoulli distribution (Murphy 2012, Chapter 9).

2.1.2 Priors

Having specified the parameters we are interested in, we need to define our beliefs over their possible values before we have seen any data. We do this by defining a distribution of the parameters $p(\theta)$, referred to as the *prior distribution* or just the *prior*. It is common to use a tractable distribution for which we can easily evaluate the PDF and sample from.

As an example if we knew that a parameter had to be bounded between two limits, but had no other information, we might choose a uniform distribution as a prior.

$$p(\theta) = \text{Uniform}(a, b). \tag{2.5}$$

Another option is to use a normal distribution as a prior with mean μ and standard deviation σ,

$$p(\theta) = \mathcal{N}(\mu, \sigma^2). \tag{2.6}$$

A common choice with a normal prior if we have some idea about the scale of the parameter is to set $\mu = 0$ and σ to the order of magnitude we expect the parameter to be on.

A frequent philosophical objection to Bayesian inference is the need to specify priors, with this being seen as a subjective process which somehow prevents the data from "speaking for themselves" (Gelman, Carlin, et al. 2014, Chapter 2). This objection is correct that specifying priors *is* a subjective process, but fails to recognise that so are all of the other modelling choices we make (MacKay 2003, Chapter 3)! For the logistic regression example, we make the assumption that the likelihood was best described by a Bernoulli distribution parametrised by a linear function of $\mathbf{x}_i$, and that $\mathbf{x}_i$ was not a random variable.

Given that we cannot avoid making subjective assumptions, it therefore makes sense to be as explicit as possible about those assumptions. Specifying our prior beliefs as a distribution forces us to make our assumptions clear. It requires us to justify our choices, and allows others to criticise them.

Having specified a prior and likelihood, a common way of describing the resultant model is to write it down as series of sampling statements. These sampling statements describe the model process that we think generated the data. For our logistic regression model with normal priors, the sampling statements would be

$$b \sim \mathcal{N}(0, \sigma_b^2), \tag{2.7}$$
$$\mathbf{w} \sim \mathcal{N}(0, \sigma_w^2), \tag{2.8}$$
$$y_i \sim \text{Bernoulli}(\text{logit}^{-1}(\mathbf{w}^T\mathbf{x}_i + b)), \quad i \in [1..N]. \tag{2.9}$$

We can use this generative process to reason about our prior choices by running a prior predictive check (Gabry et al. 2019). The procedure works by drawing a number of samples from the prior, then generating simulated data or other quantities from those samples. By inspecting the distribution of these quantities, we can reason about what data our model considers plausible.

Consider trying to select a standard deviation $\sigma_{\mathbf{w}}$ for the prior $p(\mathbf{w})$ on the weights with the logistic regression example. In the absence of other information, we might think that setting $\sigma_{\mathbf{w}} = 100$ would prevent the parameters from being overly constrained. We can run a prior predictive check by sampling from the prior, then computing the Bernoulli probabilities for each datapoint. The left plot in Figure 2.2 shows a histogram over probabilities with 1000 samples from this prior, with $\sigma_b = 1$ The histogram shows us that our model expects the data to be almost always perfectly separable, and there would be no point in trying to fit a logistic regression model to it. The right hand plot shows the same histogram, but now with $\sigma_{\mathbf{w}} = 1$. The simulated data now looks much more reasonable, with a range of probabilities possible.

2.1.3 Marginal Likelihood

The denominator in Equation 2.1 is sometimes referred to as the *evidence* or *marginal likelihood*. The latter term arises as it is can be found by marginalising the likelihood under the prior,

$$p(\mathcal{D}) = \int p(\mathcal{D} \mid \theta)p(\theta)\,\mathrm{d}\theta. \tag{2.10}$$

This integral is what makes Bayesian inference challenging for many problems. For limited combinations of likelihood and prior, this integral will have a closed-form. The priors in this case are referred to as *conjugate priors*, as the posterior distribution will be in the same family of distributions as the prior (Gelman, Carlin, et al. 2014, Chapter 2).

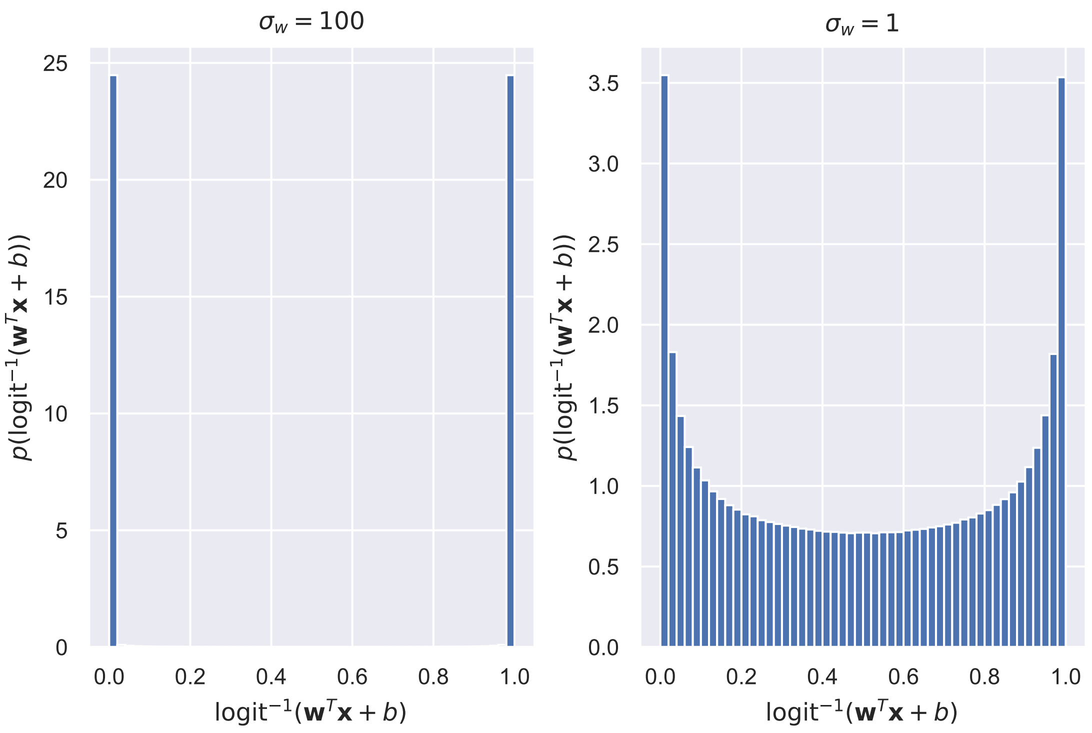


Figure 2.2: An example of doing a prior predictive check for the logistic regression example. With the prior standard deviation $\sigma_w = 100$, our model expects data that is almost always entirely perfectly separable. With $\sigma_w = 1$ the range of possibilities is more reasonable.

For most models (including the logistic regression model) there is no closed-form. If we want to evaluate the integral directly, we will need to use numerical methods. The computational cost of doing this scales exponentially with the dimensionality of θ, and is only feasible for very small dimensionalities (MacKay 2003, Chapter 21). The inability to evaluate $p(\mathcal{D})$ exactly prevents us from evaluating the PDF of the posterior straightforwardly. [2]

2.1.4 Expectations

Bayesian inference is often stated in such a way as to make it seem like the ultimate goal is to find or approximate the posterior distribution. In reality *expectations* of functions under the posterior are more generally useful than the posterior itself, especially if we wish to make decisions based on our inferences. The expectation of a function $f(\theta)$ under the posterior is defined as

$$\mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)] = \int f(\theta)p(\theta \mid \mathcal{D})\,\mathrm{d}\theta. \tag{2.11}$$

A common expectation of interest is the posterior mean of the parameters, $\mathbb{E}_{p(\theta|\mathcal{D})}[\theta]$. Other expectations of interest may include certain quantiles of the parameters under the posterior, or the difference between two particular quantiles. Unfortunately, in general we cannot compute Equation 2.11 as we cannot in general compute $p(\theta \mid \mathcal{D})$. Even if we could compute $p(\theta \mid \mathcal{D})$ or an approximation to it, the integral would still be intractable for similar reasons as for Equation 2.10.

Most of the value of a posterior expectation in high dimensions is dominated by a volume of the posterior distribution support referred to as the *typical set*, a concept borrowed from information theory (MacKay 2003, Chapter 4). Loosely defined, the typical set is the volume between the tails of the posterior (where the volume is large but the density is low) and the mode of the posterior (where the density is high but the volume is small). The fraction of the posterior support occupied by the typical set decreases as the dimensionality of the support increases.

Fortunately, if we could draw samples from $p(\theta \mid \mathcal{D})$ (or its approximation), we can compute an approximation solution to these expectations using Monte

[2]It is possible to approximate the marginal likelihood directly, and methods which do this are often used for model comparison (Llorente et al. 2023).

Carlo integration (MacKay 2003, Chapter 29),

$$\mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)] \approx \frac{1}{S}\sum_{s=1}^{S} f(\theta_s), \quad \theta_s \sim p(\theta \mid \mathcal{D}). \tag{2.12}$$

One important class of posterior expectations that are frequently of interest are those that predict the values of new unseen variables $\tilde{\mathcal{D}}$ given observed data $\mathcal{D}$. This can be done by computing the expectation of the likelihood under the posterior,

$$p(\tilde{\mathcal{D}} \mid \mathcal{D}) = \int p(\tilde{\mathcal{D}} \mid \theta) p(\theta \mid \mathcal{D})\,\mathrm{d}\theta, \tag{2.13}$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} p(\tilde{\mathcal{D}} \mid \theta_s), \quad \theta_s \sim p(\theta \mid \mathcal{D}). \tag{2.14}$$

This distribution is often referred to as the posterior predictive distribution (Gelman, Carlin, et al. 2014, Chapter 5). If we expect to use this distribution frequently, it is common to store a set of samples $\{\theta_i\}_{i=1}^{N}$ from the posterior rather than resampling whenever we need to make a prediction.

In the logistic regression case, the posterior predictive distribution for new samples $\tilde{\mathbf{x}}$ can be approximated as

$$\mathbf{w}_s, b_s \sim p(\mathbf{w}, b \mid \{\mathbf{x}_i, y_i\}_{i=1}^{N}), \tag{2.15}$$

$$p(\tilde{y} \mid \tilde{\mathbf{x}}, \{\mathbf{x}_i, y_i\}_{i=1}^{N}) = \frac{1}{S}\sum_{s=1}^{S} \text{Bernoulli}(\tilde{y} \mid \text{logit}^{-1}(\mathbf{w}_s^T\tilde{\mathbf{x}} + b_s)), \tag{2.16}$$

which for the specific case $\tilde{y} = 1$ is

$$p(\tilde{y} = 1 \mid \tilde{\mathbf{x}}, \{bx_i, y_i\}_{i=1}^{N}) = \frac{1}{S}\sum_{s=1}^{S} \text{logit}^{-1}(\mathbf{w}_s^T\tilde{\mathbf{x}} + b_s). \tag{2.17}$$

Deferring the question of how we obtained the samples to Section 2.2, Figure 2.3 plots $p(\tilde{y} = 1 \mid \tilde{\mathbf{x}}, \{bx_i, y_i\}_{i=1}^{N})$ as a function of $\tilde{\mathbf{x}}$ for the logistic model.

2.2 Markov Chain Monte Carlo

Despite not being able to evaluate the PDF of a posterior in general, is possible to produces samples from the posterior. One such way of doing this is to use Markov chain Monte Carlo (MCMC) methods (Chapter 29 MacKay 2003). We only need to be able to evaluate the unnormalised posterior density

$$p^{\star}(\theta \mid \mathbf{x}) = p(\mathcal{D} \mid \theta)p(\theta). \tag{2.18}$$

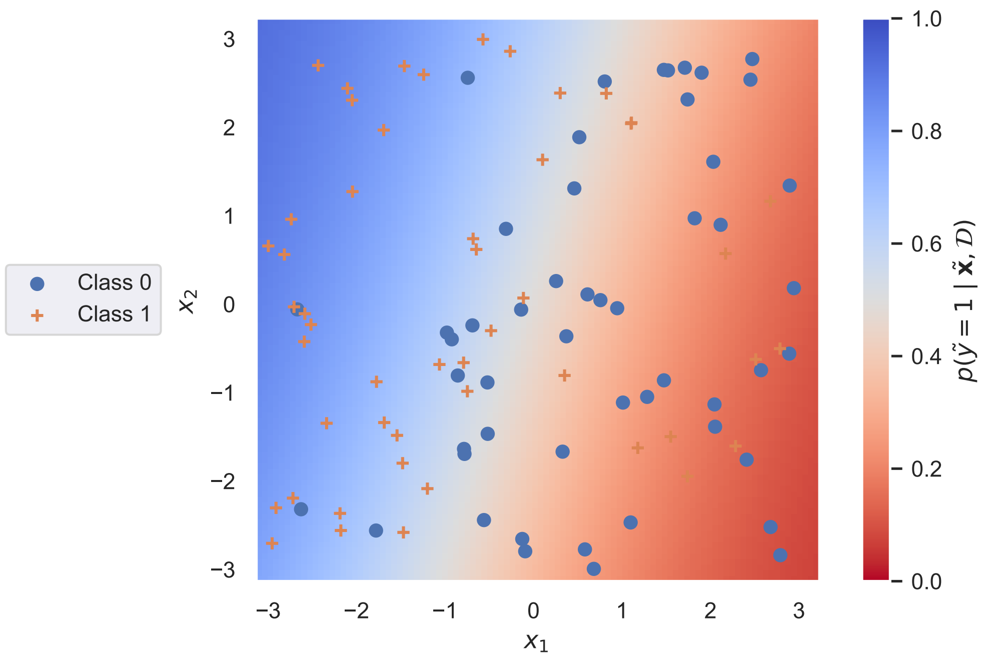


Figure 2.3: The posterior predictive distribution for the logistic regression model as a function of x. Also plotted is the data used to infer the parameters and predictive distribution. Even with a fine grid computing all of the Monte Carlo approximations to the expectation took less than a second.

This sampling is done by iteratively constructing a Markov chain of samples $\{\theta_s\}_{s=1}^{S}$, where the sample θ_{s+1} depends only on θ_s. There are many algorithms than can construct such a chain where the samples correspond to draws from $p(\theta \mid \mathcal{D})$ (MacKay 2003, Chapters 28-29). Here we only consider the Metropolis-Hastings method, and an extension of it, Hamiltonian Monte Carlo.

2.2.1 Metropolis-Hastings

The Metropolis-Hastings method is a standard MCMC method which is straightforward to implement (Hastings 1970). Given the current state of a chain θ_s, we can sample from a tractable distribution $q(\theta' \mid \theta_s)$ to get a new proposed state θ'.

$$\theta' \sim q(\theta' \mid \theta_s) \tag{2.19}$$

We then evaluate the unnormalised posterior density with the new proposed state and compute the quantity

$$a = \frac{p^\star(\theta' \mid \mathbf{x}) q(\theta_s \mid \theta')}{p^\star(\theta_s \mid \mathbf{x}) q(\theta' \mid \theta_s)} \tag{2.20}$$

a is referred to as the acceptance ratio. If $a \geq 1$, we set $\theta_{s+1} = \theta'$. If $a < 1$, we set $\theta_{s+1} = \theta'$ with probability a, and otherwise set $\theta_{s+1} = \theta_s$,

$$u \sim \text{Uniform}[0, 1], \tag{2.21}$$

$$\theta_{s+1} = \begin{cases} \theta' & \text{if } u \leq a \\ \theta_s & \text{if } u > a \end{cases}. \tag{2.22}$$

Intuitively this works because samples in higher density regions of $p(\theta \mid \mathcal{D})$ are more likely to be accepted compared to samples from lower density regions. As we usually start each chain at a point sampled from a distribution other than the posterior, it may take some time for our chain to reach the typical set. It is therefore necessary to discard some fraction of the start of the chain as *burn-in* or *warm-up*, sometimes as much as half of the chain (Gelman, Carlin, et al. 2014, Chapter 11). As each sample depends on the previous sample, unless the proposal distribution is very efficient, successive samples will be correlated over a given timescale. This means that the effective sample size is typically smaller than the actual number of samples.

A typical choice for $q(\theta' \mid \theta_s)$ is a diagonal Gaussian centred around θ_s with standard deviation σ_q,

$$\theta' \sim \mathcal{N}(\theta' \mid \theta_i, \sigma_q^2). \tag{2.23}$$

In this case, the proposal distribution is symmetric, $q(\theta' \mid \theta_s) = q(\theta_s \mid \theta')$, and the acceptance ratio simplifies to

$$a = \frac{p^\star(\theta' \mid \mathcal{D})}{p^\star(\theta_s \mid \mathcal{D})}. \tag{2.24}$$

This simplified version is the Metropolis algorithm (Metropolis et al. 1953).[3] Careful selection of the value of σ_q is necessary to get the method to perform

[3]There is some controversy around the naming of the algorithm. Edward Teller stated in his memoirs that all of the authors worked for "days (and nights)" on the publication (Teller and Schoolery 2009). Marshall Rosenbluth contradicts this, stating that whilst Edward Teller provided an initial crucial suggestion, it was Marshall who did the theoretical work, acknowledging several helpful conversations with John von Neumann (Gubernatis 2005). Marshall also recalled that Augusta Teller started writing the computer code, then Ariana Rosenbluth took it over and rewrote it from scratch, whilst Nicholas Metropolis' only contribution was providing computer time. So perhaps it should be called the Rosenbluth-Teller method.

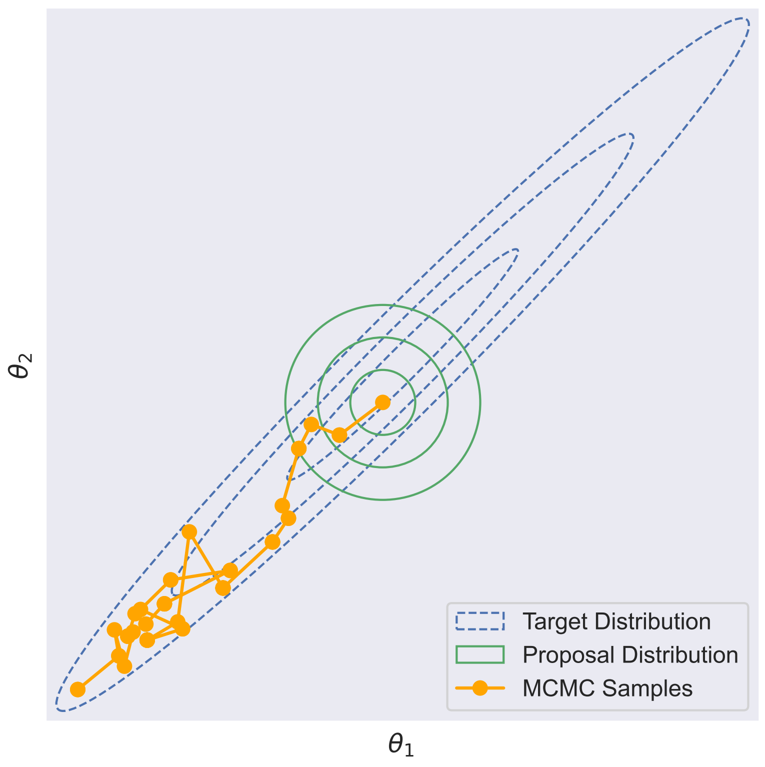


Figure 2.4: Sampling from an example posterior using the Metropolis method after 50 steps. The target distribution features strong correlations, so the diagonal Gaussian proposal distribution must have a relatively small standard deviation in order to keep the acceptance rate around about 0.5. The chain can only take small highly correlated steps, and is not close to the target distribution even after 50 steps.

well. If it is too small, the chain will need to take a large number of steps to fully explore the posterior. Too large, and most proposals will land outside of the typical set and be rejected. If the posterior has strong correlations and/or large lengthscales in some directions and small lengthscales in others, these two requirements may be in conflict. Figure 2.4 shows an example of sampling from a posterior with such strong correlations. This problem gets worse as the dimensionality of the posterior increases, and for very high dimensions most proposals will be outside the effective support of the posterior unless σ is very small, requiring the chain to run for many steps (MacKay 2003, Chapter 29).

Figure 2.5 shows samples drawn using Metropolis-Hastings from the posterior of the logistic regression model, with both prior σ values set to 1. We visualise the samples by using a corner plot, which shows 2D histograms of each pairwise combination of dimensions in θ along with 1D histograms showing the marginal distribution of each dimension. Samples from the tails of the distribution are plotted directly on the 2D histograms. The contours in the 2D histograms contain an estimated 39%, 86% and 99% of the density respectively, corresponding to the percentage of the density of a standard 2D Gaussian contained by circles centred at the mean of radius 1, 2 and 3 (Foreman-Mackey 2016). Where available we also mark ground truth values, which give a rough indication of whether the posterior approximation is reasonably behaved. We use this format throughout this thesis.

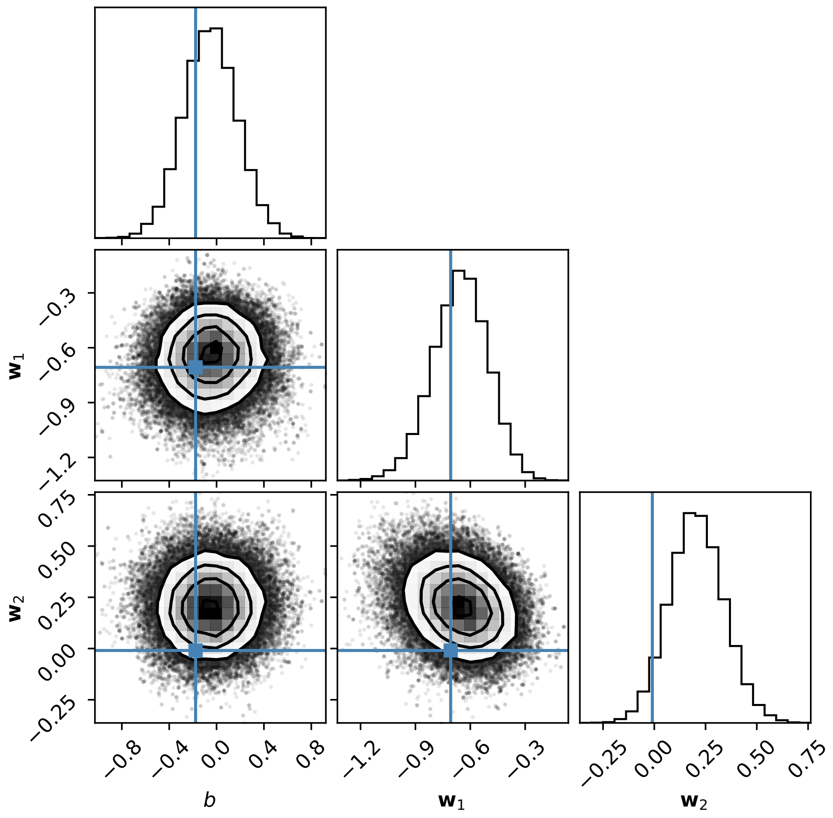


Figure 2.5: Samples from the logistic regression posterior, visualised as a corner plot using *corner.py* (Foreman-Mackey 2016). The vertical and horizontal lines indicate the ground truth values used to generate the dataset. The posterior has identified where the ground-truth parameters are, but is not totally certain as to their exact location.

2.2.2 Hamiltonian Monte Carlo

As discussed, Metropolis-Hastings with a standard Gaussian proposal distribution will struggle with high-dimensional posteriors, as most proposals will land outside the typical set and will be rejected. If we could differentiate the posterior PDF with respect to the parameters θ, we could use the resulting gradients to explore the posterior. Hamiltonian Monte Carlo is an MCMC method that uses these gradients to make efficient proposals which are more likely to be accepted (Duane et al. 1987; Neal 2011). Here we follow the derivation of MacKay (2003, Chapter 30) to provide a brief overview of the method. For a comprehensive review of HMC and the adaptive variants see Betancourt (2018).

We assume the posterior can be written in the form

$$E(\theta) = \log p(\mathcal{D} \mid \theta) + \log p(\theta), \tag{2.25}$$

$$p(\theta \mid \mathcal{D}) = \frac{e^{-E(\theta)}}{Z}, \tag{2.26}$$

where Z is the normalising constant. This is not a particularly restrictive assumption as we generally express $p^\star(\theta_i \mid \mathcal{D})$ on a log-scale to avoid numerical errors. We also assume that we can differentiate $E(\theta)$ with respect to θ, which restricts HMC to working with posteriors over continuous parameters but is otherwise easy to do using automatic differentiation (Baydin, Pearlmutter, et al. 2018).

HMC proposes new samples by treating the current sample as a particle on a surface defined by $E(\theta)$ at position θ_s. By augmenting θ_s with a momentum vector $\mathbf{v}_s$, we can then simulate the trajectory of this particle over the surface defined by $E(\theta)$ using Hamiltonian mechanics. Formally, we define the Hamiltonian as the sum of a potential energy given by $E(\theta)$ and a kinetic energy $K(\mathbf{v})$,

$$H(\theta, \mathbf{v}) = E(\theta) + K(\mathbf{v}). \tag{2.27}$$

This defines a joint distribution over θ and $\mathbf{v}$

$$p_H(\theta, \mathbf{v}) = \frac{1}{Z_H} e^{-H(\theta, \mathbf{v})}, \tag{2.28}$$

$$= \frac{1}{Z_H} e^{-E(\theta)} e^{-K(\mathbf{v})}. \tag{2.29}$$

As this distribution is separable, the marginal distribution of θ is the posterior distribution. A typical choice for the kinetic energy (without units) is

$$K(\mathbf{v}) = \frac{\mathbf{v}^T \mathbf{v}}{2} \tag{2.30}$$

which defines a standard normal distribution over momentum

$$p(\mathbf{v}) = \mathcal{N}(\mathbf{v} \mid \underline{0}, I). \tag{2.31}$$

The trajectory of the particle over the surface $E(\theta)$ can be simulated by formulating it as differential equation with derivatives

$$\dot{\theta} = \mathbf{v}, \tag{2.32}$$

$$\dot{\mathbf{v}} = -\frac{\partial E(\theta)}{\partial \theta} \tag{2.33}$$

and computing a solution using an Leapfrog numerical integrator.[4] The initial condition for the integrator is the current sample θ_s, with $\mathbf{v}_s$ drawn from the distribution defined by $K(\mathbf{v})$. The integrator runs for L steps with stepsize ϵ, producing a new proposal $\theta', \mathbf{v}'$ at the end. If the numerical integrator was perfect, the Hamiltonian would be conserved exactly and we would always accept the final position of the particle as a new sample. In practice there will be numerical errors in the trajectory, so we use a Metropolis accept-reject step with

$$a = \frac{p_H(\theta', -\mathbf{v}')}{p_H(\theta_s, \mathbf{v}_s)} \tag{2.34}$$

If accepted, we can drop the proposed $\mathbf{v}'$ as we are only interested in θ. Repeated iterations of this process will asymptotically result in samples θ_s being drawn from $p(\theta \mid \mathcal{D})$.

In practice the effectiveness of HMC is sensitive to the choice of number of integrator steps L and the stepsize ϵ (Neal 2011). If L is too small, the trajectory may not get very far when we apply the Metropolis accept-reject step when it could have gone further. If it is too large, the trajectory may turn back on itself and propose a new sample that is closer than the furthest away point it reached. If ϵ is too large, the error in the trajectory will grow too large and the proposal will be more likely to be rejected. If it is too small, the trajectory will waste computation time taking unnecessarily small steps.

The no-U-turn sampler (NUTS) alleviates this issue by permitting the number of steps to be selected dynamically whilst the integration is running, and by automatically tuning the step size during the warmup phase (Hoffman and Gelman 2014). Subsequent improvements to the acceptance criteria and

[4]The choice of integrator matters (Betancourt 2018). It needs to be time-reversible (to satisfy detailed balance) and symplectic (so the error does not grow with L) (Leimkuhler and Reich 2004). The leapfrog integrator satisfies both of these conditions.

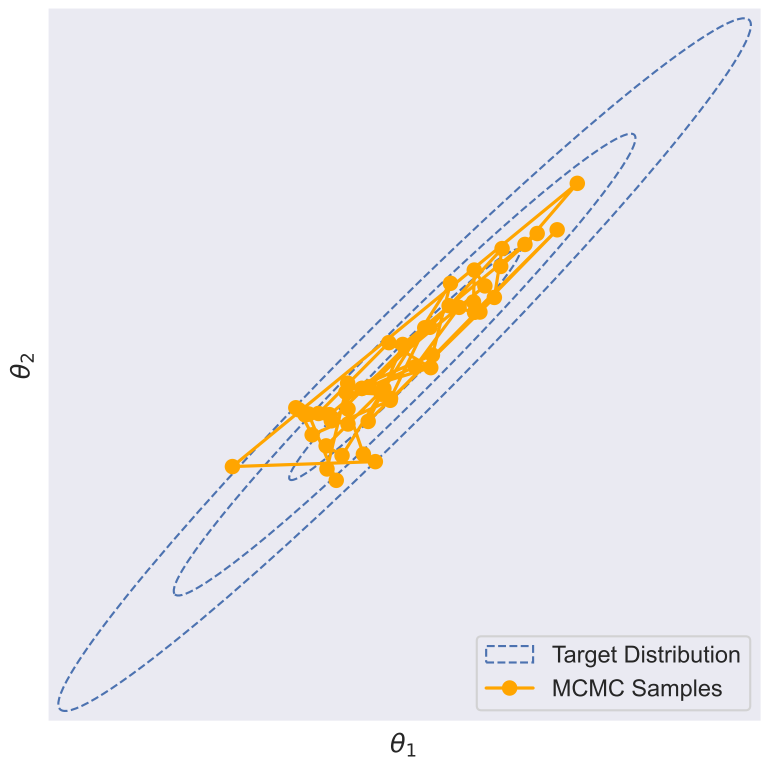


Figure 2.6: Sampling from the same posterior as in Figure 2.4 but this time using adaptive HMC. Successive samples in the chain are much less correlated, whilst the acceptance rate was 0.95.

warmup have lead to the term *Adaptive HMC* being used for the self-tuning variant of HMC which is almost always used in practice (Betancourt 2016b; Stan Development Team 2022). Figure 2.6 shows samples drawn from the same distribution as Figure 2.4 using an implementation of adaptive HMC (Bingham et al. 2019). Successive samples are less correlated than in Figure 2.4 as the sampler is able to propose samples which are much further apart whilst maintaining an acceptance rate of around 0.95.

2.3 Diagnostics

With MCMC methods it is typical to run multiple chains independently of each other. By comparing the chains we can check for problems. Here we describe two diagnostics we can use to check how well the chains are sampling from the posterior. Note that whilst we define these diagnostics in terms of MCMC methods, they could be applied to any method that can produce sequences of samples from the posterior independently, an idea we revisit in Chapter 4.

2.3.1 Potential Scale Reduction Factor

The potential scale reduction factor $\hat{R}$ provides an indication of whether a series of chains have converged to the same stationary distribution (Gelman and Rubin 1992). Here we follow the derivation given in Gelman, Carlin, et al. (2014, Chapter 11), which contains improvements to the original formulation. It is typically applied to scalar quantities of interest. To keep the notation simple we assume that our parameter θ is a scalar but we could extend the calculations to each element of θ if it were a vector, or to scalar functions $f(\theta)$.

The method works by comparing the variances within each chain to the variances between each chain. To detect issues where a single chain has not reached a stationary distribution, it is common practice to split each chain in half after discarding the warmup samples and treat the halves as separate chains (sometimes specifically referred to as split-$\hat{R}$). After splitting we have M chains each with N samples θ_{nm}. We calculate the between-chain variance B as

$$\bar{\theta}_{\cdot m} = \frac{1}{N}\sum_{n=1}^{N}\theta_{nm}, \tag{2.35}$$

$$\bar{\theta}_{\cdot\cdot} = \frac{1}{M}\sum_{m=1}^{M}\bar{\theta}_{\cdot m}, \tag{2.36}$$

$$B = \frac{N}{M-1}\sum_{m=1}^{M}(\bar{\theta}_{\cdot m} - \bar{\theta}_{\cdot\cdot})^2. \tag{2.37}$$

The within-chain variance W is calculated as

$$s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}(\theta_{nm} - \bar{\theta}_{\cdot m})^2, \tag{2.38}$$

$$W = \frac{1}{M}\sum_{m=1}^{M}s_m^2. \tag{2.39}$$

We can then estimate the variance of θ under the posterior $\mathrm{var}(\theta \mid \mathcal{D})$ as a weighted combination of B and W,

$$\widehat{\mathrm{var}}^{+}(\theta \mid \mathcal{D}) = \frac{N-1}{N}W + \frac{1}{N}B. \tag{2.40}$$

If the chains start with samples which are overdispersed relative to the posterior (for example, because we started them with a sample from the prior and after warmup the chain has not yet converged to a stationary distribution) then $\widehat{\mathrm{var}}^+(\theta \mid \mathcal{D})$ will overestimate $\mathrm{var}(\theta \mid \mathcal{D})$. If the chain starts with samples from the posterior, or in the limit $N \to \infty$, it will be an unbiased estimator.

Meanwhile W will underestimate $\mathrm{var}(\theta \mid \mathcal{D})$ for finite N as each chain has not converged to the target distribution. In the limit $N \to \infty$ it will approach $\mathrm{var}(\theta \mid \mathcal{D})$. We can use these under-and-over estimates to compute the potential scale reduction factor,

$$\hat{R} = \sqrt{\frac{\widehat{\mathrm{var}}^+(\theta \mid \mathcal{D})}{W}}, \tag{2.41}$$

which estimates the scale by which the current distribution of θ might be reduced if we continued the chains in the limit $N \to \infty$. If $\hat{R}$ is at or close to 1, it suggests that each chain has converged to the same distribution. What constitutes a good value of $\hat{R}$ has been subject to a process of "deflation" over the years as better MCMC methods have been made available. Brooks and Gelman (1998) suggest 1.2 as a threshold, Gelman, Carlin, et al. (2014, Chapter 11) state 1.1 whilst Vehtari, Gelman, Simpson, et al. (2021) treat anything greater than 1.01 as suspicious. If after running a series of chains we find we have a bad $\hat{R}$, we can try running them again with an increased number of samples and/or an increased number of warm-up samples. If this does not improve the value of $\hat{R}$, something is wrong and we should not trust expectations computed with these samples.

2.3.2 Effective Sample Size

A related statistic is the effective sample size N_{ESS}. If the NM samples we use to compute a Monte Carlo posterior estimate of the mean θ were independent, then the Monte Carlo standard error (MCSE) on the mean is

$$\mathrm{MCSE}(\theta) = \sqrt{\frac{\mathrm{var}(\theta \mid \mathcal{D})}{NM}} \tag{2.42}$$

where $\mathrm{var}(\theta \mid \mathcal{D})$ is the posterior variance of θ. If the samples are not independent then the MCSE will be different. Again we follow Gelman, Carlin, et al. (2014, Chapter 11) to describe an estimator for N_{ESS}.

The MCSE for the average of M correlated sequences of length N is

$$MCSE(\theta) = \sqrt{\frac{(\sum_{t=-\infty}^{\infty} \rho_t) \operatorname{var}(\theta \mid \mathcal{D})}{NM}}, \tag{2.43}$$

$$= \sqrt{\frac{(1 + \sum_{t=1}^{\infty} \rho_t) \operatorname{var}(\theta \mid \mathcal{D})}{NM}} \tag{2.44}$$

where ρ_t is the autocorrelation of θ at lag t,

$$\rho_t = \frac{1}{\operatorname{var}(\theta \mid \mathcal{D})} \mathbb{E}[\theta_n \theta_{n+t}]. \tag{2.45}$$

By equating Equation 2.44 to Equation 2.42, we can derive the equivalent number of samples that would have given us the same MCSE if we had used independent samples,

$$N_{\text{ESS}} = \frac{NM}{1 + \sum_{t=1}^{\infty} \rho_t}, \tag{2.46}$$

We can estimate this quantity by first computing the variance estimator $\widehat{\operatorname{var}}^{+}(\theta \mid \mathcal{D})$ from Equation 2.40, then computing the variogram V_t as

$$V_t = \frac{1}{M(N-t)} \sum_{m=1}^{M} \sum_{n=t+1}^{N} (\theta_{nm} - \theta_{(n-t)m})^2, \tag{2.47}$$

$$\approx \mathbb{E}[(\theta_n - \theta_{(n-t)})^2] \tag{2.48}$$

By noting that $\mathbb{E}[(\theta_n - \theta_{(n-t)})^2] = 2(1 - \rho_t) \operatorname{var}(\theta \mid \mathcal{D})$ we can produce an estimate of the autocorrelation ρ_t,

$$\hat{\rho}_t = 1 - \frac{V_t}{2\widehat{\operatorname{var}}^{+}(\theta \mid \mathcal{D})} \tag{2.49}$$

which in turn can be used to estimate N_{ESS},

$$\hat{N}_{\text{ESS}} = \frac{MN}{1 + \sum_{t=1}^{\infty} \hat{\rho}_t}. \tag{2.50}$$

For larger values of t the estimate $\hat{\rho}_t$ will be very noisy. This can be avoided by truncating the summation over t in Equation 2.50 when the sum of two successive values $\hat{\rho}_{t'} + \hat{\rho}_{t'+1}$ is negative. For convenience we denote the estimator $\hat{N}_{\text{ESS}}$ as N_{ESS}.

As well as using N_{ESS} to calculate the MCSE on estimates of the mean of θ, we can use it as a diagnostic. If N_{ESS} does not increase as N increases, or the ratio $\frac{N_{\text{ESS}}}{NM}$ is very small, it suggests that the chains are not doing a good job of exploring the posterior and we should not trust expectations computed using the samples.

In practice there are some pathological cases where both $\hat{R}$ and N_{ESS} as formulated here could indicate that a series of chains were behaving correctly even though they had not converged. Vehtari, Gelman, Simpson, et al. (2021) demonstrate these cases and provide alternative formulations based on rank statistics and quantiles that are robust to these failures, but the principal is the same. We use this improved formulation throughout this thesis, making use of the implementation available in the *ArviZ* package (Kumar et al. 2019).[5]

The potential scale reduction factor $\hat{R}$ only indicates if our chains have converged to the same distribution, it does not tell us if that distribution is actually the posterior $p(\theta \mid \mathcal{D})$. Simulation-based calibration (SBC) is a method to help check whether software we have written is actually producing samples from the posterior (Talts et al. 2020). We describe it in more detail in Chapter 4 and use it to check inference method implementations we have written.

2.4 Density Estimation

Fitting a tractable distribution to some target distribution using samples of x from that target distribution is generically referred to as *density estimation* and the distribution we fit to it is referred to as a *density estimator*. A typical density estimator would be a distribution with parameters we can optimise to match the target density. A simple distribution such as a Gaussian could be used as a density estimator, but their limited flexibility makes them inappropriate when we have a complex target distribution we need to approximate as accurately as possible. In this section we describe two more flexible families of distributions more suited for performing density estimation.

One application of density estimation is to use an estimator to approximate a posterior distribution. We defer the discussion of how to actually do this approximation in a specific case to Section 2.5.1, and for a more general case to Chapter 6.

[5]There are actually two versions of the ESS presented by Vehtari, Gelman, Simpson, et al. (2021). We use the version called *bulk-ESS* as it works as a convergence diagnostic although it cannot be used to calculate the MCSE directly.

2.4.1 Gaussian Mixture Models

Gaussian mixture models (GMMs) take the form

$$q_\phi(\mathbf{x}) = \prod_{j=1}^{K} \alpha_j \mathcal{N}(\mathbf{x} \mid \mu_j, \Sigma_j), \quad \phi = \{\alpha_j, \mu_j, \Sigma_j\}_{j=1}^{K} \tag{2.51}$$

where μ_j and Σ_j are the means and covariances of multivariate Gaussians which make up the components of the mixture (Murphy 2012, Chapters 2, 11). The mixture weights α_j have the constraints $\alpha_j \geq 0, \sum_{j=1}^{K} \alpha_j = 1$ in order to ensure the GMM is a valid distribution.

GMMs are commonly thought of as an unsupervised learning technique with the aim of identifying clusters that make up the observed data (Murphy 2012, Chapter 11). However, they can also be interpreted more generally as a density estimator, using multiple components to approximate a more complex distribution. Figure 2.7 shows an example of approximating a complex target distribution using a two component mixture model. We discuss methods of fitting GMMs in Chapter 5.

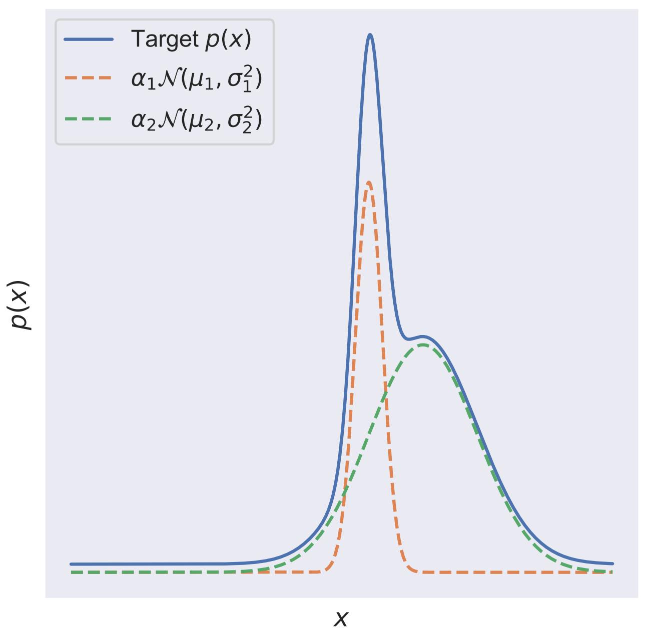


Figure 2.7: Approximating a target density with a two component mixture of Gaussians.

GMMs can perform well at density estimation in lower dimensions if the target density is relatively smooth, and are well-suited to approximating multi-modal distributions. As the dimensionality of the target distribution increases, an exponentially larger number of components is needed in order to adequately cover the support, limiting their ability to model high-dimensional densities. They also perform poorly when the target density has sharp cut-offs.

2.4.2 Normalising Flows

Normalising flows are an alternative class of density estimator that can potentially outperform GMMs, especially for higher-dimensional densities. They leverage the flexibility of neural networks to approximate complex target densities, and have successfully been used for problems such as the generative modelling of images (Kingma and Dhariwal 2018). Here we provide a brief introduction to them. For a more comprehensive review see Papamakarios, Nalisnick, et al. (2021).

The core idea behind normalising flows is that we can represent $\mathbf{x}$ as an invertible transformation $T(\mathbf{z})$ of a sample $\mathbf{z}$ from some tractable base distribution $\pi(\mathbf{z})$.

$$\mathbf{z} \sim \pi(\mathbf{z}), \tag{2.52}$$

$$\mathbf{x} = T(\mathbf{z}). \tag{2.53}$$

By the multivariate change-of-variables formula (Murphy 2012, Chapter 2), the density $q(\mathbf{x})$ is

$$q(\mathbf{x}) = \pi(\mathbf{z})|\det J_T(\mathbf{z})|^{-1}, \tag{2.54}$$

$$= \pi(T^{-1}(\mathbf{x}))|\det J_{T^{-1}}(\mathbf{x})| \tag{2.55}$$

where $J_T(\mathbf{z})$ is the Jacobian matrix of $T(\mathbf{z})$. In order for $q(\mathbf{x})$ to be well-defined, $T(\mathbf{z})$ must be differentiable and must have an inverse $T^{-1}(\mathbf{x})$. Ideally $|\det J_T(\mathbf{z})|$ will also be straightforward to compute. The name "normalising flow" is derived from the fact that the transformation $T(\mathbf{z})$ allows density to *flow* from the base distribution towards $q(\mathbf{x})$, whilst the correction $|\det J_T(\mathbf{z})|$ ensures that $q(\mathbf{x})$ still *normalises* to one.

If $T_\phi(\mathbf{z})$ has parameters ϕ, we can fit $q_\phi(\mathbf{x})$ to a dataset of observed samples

$\{\mathbf{x}_i\}_{i=1}^N$ from the target distribution $p(\mathbf{x})$ by maximising the log-likelihood,

$$\mathcal{L}(\phi, \{\mathbf{x}_i\}_{i=1}^N) = \sum_{i=1}^N \log q_\phi(\mathbf{x}_i), \tag{2.56}$$

$$= \sum_{i=1}^N \left[\log \pi(T_\phi^{-1}(\mathbf{x}_i)) + \log|\det J_{T_\phi^{-1}}(\mathbf{x}_i)|\right] \tag{2.57}$$

How well $q_\phi(\mathbf{x})$ can model $p(\mathbf{x})$ will depend on how flexible $T_\phi(\mathbf{z})$ is.

The detail in normalising flows is in finding good parametrisable families of invertible transforms. Standard neural networks are flexible but cannot be used directly as a transform because they are not generally invertible. One way around this problem is to use a neural network $f_\phi(\mathbf{z})$ with parameters ϕ that takes $\mathbf{z}$ as an input and outputs the parameters of a simpler invertible transform that can be applied to each element of $\mathbf{z}$ such as an affine shift-and-scale transform,

$$H \leftarrow f_\phi(\mathbf{z}), \text{ where } H = \{\mathbf{a}, \mathbf{b}\}, \tag{2.58}$$

$$\mathbf{x} \leftarrow \exp(\mathbf{a}) \odot \mathbf{z} + \mathbf{b}. \tag{2.59}$$

The exponentiation (or other appropriate function) keeps the scale non-zero, ensuring the transform is always invertible. More complicated transforms such as that used in the neural spline flow allow for more expressive normalising flows (Durkan, Bekasov, et al. 2019).

Care needs to be taken with how the neural network uses the input $\mathbf{z}$ in order to keep the transformation invertible. One way of doing this is by using a *coupling* transformation (Dinh, Sohl-Dickstein, and Bengio 2017). In a coupling transformation, the first half of the vector $\mathbf{z}$ is copied straight to $\mathbf{x}$. The neural network then takes the first half of the vector $\mathbf{z}$ as input, and uses it to compute the parameters of the transformations to be applied to the second half of $\mathbf{z}$,

$$\mathbf{x}_{1:d} \leftarrow \mathbf{z}_{1:d} \text{ where } d = D/2 \tag{2.60}$$

$$H \leftarrow f(\mathbf{z}_{1:d}) \text{ where } H = \{\mathbf{a}_{1:d}, \mathbf{b}_{1:d}\} \tag{2.61}$$

$$\mathbf{x}_{d:D} \leftarrow \exp(\mathbf{a}_{1:d}) \odot \mathbf{z}_{d:D} + \mathbf{b}_{d:D}. \tag{2.62}$$

When we need to invert the transform, the affine parameters $\mathbf{a}$ and $\mathbf{b}$ needed to invert $\mathbf{x}_{d:D}$ can be computed using $\mathbf{x}_{1:d}$.

Another way of ensuring an invertible transform is by enforcing an *autoregressive* structure on the neural net (Kingma, Salimans, et al. 2016; Papamakarios,

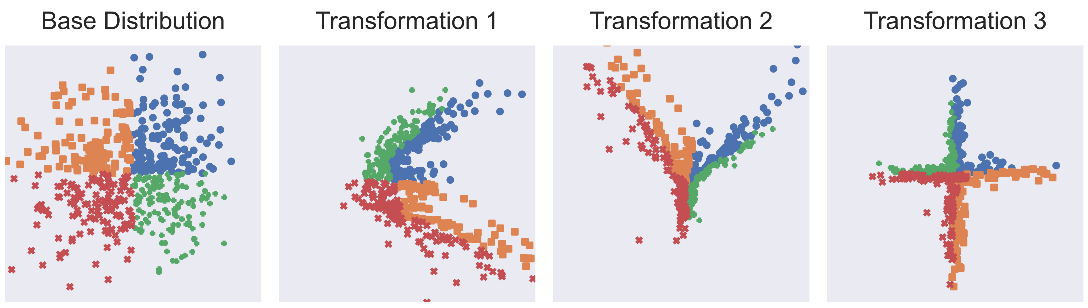


Figure 2.8: Successive affine autoregressive transformations in a normalising flow from a standard Gaussian base distribution towards a cross-shaped distribution.

Pavlakou, and Murray 2017). With an autoregressive structure, the parameters of the transformation applied to $\mathbf{z}_d$ will only depend on the elements of $\mathbf{z}$ preceding it, $\mathbf{z}_{1:(d-1)}$. Autoregressive transforms are generally more flexible than coupling transforms but require the neural network to be evaluated D times when inverting the transformation in order to sequentially reconstruct the input $\mathbf{z}$ (Papamakarios, Nalisnick, et al. 2021). This means that one of either sampling from the flow with Equation 2.54 or evaluating the log-likelihood with Equation 2.55 will be D times more expensive than the other operation.

Both transformations have the advantage of a lower-diagonal structured Jacobian, which makes evaluating the determinant in Equation 2.54 have complexity $\mathcal{O}(D)$ rather than $\mathcal{O}(D^3)$ for a general Jacobian matrix. On their own both coupling and autoregressive transforms are limited in their flexibility as they strongly depend on the ordering of the elements in $\mathbf{z}$. More expressive transforms can be achieved by stacking multiple transformations together and permuting the order of their elements in between transformations. Figure 2.8 shows an example of a flow with 3 stacked affine autoregressive transformations warping a standard Gaussian base distribution. The neural networks inside each affine transformation step were trained jointly by maximising the likelihood of samples drawn from a cross-shaped target distribution.

2.4.3 Conditional Density Estimation

We often wish to have a density estimator be conditional on some value. For example, we may wish for a density estimator approximating a posterior $p(\theta \mid$

$\mathcal{D}$) to work with any value of $\mathcal{D}$ rather than one specific value. This is referred to as *conditional density estimation*.

One way of accomplishing this is to train some function to output the parameters of the density estimator when given the data as an input rather than optimising the parameters directly. Given their ubiquity and flexibility, a common choice is to use a neural network for this function. Appropriate transformations can be used to ensure that the output parameters of the density estimator are correctly constrained.

As a concrete example, to turn a GMM into a conditional density estimator, we can fit a neural network that takes the data $\mathcal{D}$ as input, and outputs the means μ_j and covariances C_j for each component along with the mixture weights α_j. This is sometimes referred to as a mixture density network (MDN, Bishop 1994). For a normalising flow, the data $\mathcal{D}$ can be provided as an additional input to the neural network at each step of the flow.

2.5 Simulation-Based Inference

For many scientific modelling problems, we can write high-fidelity simulations of the phenomenon we are interested in investigating. Formally, we have a simulator $f(\theta)$ which takes parameters θ as inputs, and produces simulated data $\hat{\mathrm{x}}$ as an output. We then wish to use this simulator to make inferences about the parameters which produced some real observed data x. In the Bayesian paradigm, this would involved placing a prior $p(\theta)$ on θ, then using the posterior $p(\theta \mid \mathrm{x})$ to compute expectations of interest.

The simulator often (but not necessarily) produces stochastic outputs using internal random numbers ϵ. If we are able to access and control the generation of these random numbers, one possible approach would be to infer the full posterior $p(\theta, \epsilon \mid \mathrm{x})$ using any standard approximate inference method, using the full likelihood $p(\mathrm{x} \mid \theta, \epsilon)$ (Graham and Storkey 2017; Baydin, Shao, et al. 2019). In many cases this is not actually practical, typically because we do not have access to the internals of the simulator, the full posterior has pathological aspects for our chosen approximate inference method, or simply because the full likelihood is too computationally expensive to evaluate inside an inference loop.

In such a scenario we must resort to treating the simulator as a black box and assume we do not have access to the likelihood $p(\mathbf{x} \mid \theta)$ and that we cannot evaluate the posterior up to a constant. This leads to the common name of likelihood-free inference (LFI), with the first method in this class referred to as approximate Bayesian computation (ABC). There is a long history of these methods being using for scientific inference, with comprehensive reviews presented by Lintusaari et al. (2017) and Sisson, Fan, and Beaumont (2018). In recent years there has been a trend towards referring to it instead as simulation-based inference (SBI), especially in the work from the machine learning-adjacent community (Cranmer, Brehmer, and Louppe 2020). The likelihood, whilst not directly available, is still defined implicitly, and we are often directly or indirectly attempting to estimate it.

In this section we present three different approaches to SBI that take advantage of recent advances in neural network-based density estimation and classification techniques. For each approach, unless samples of provided parameters and simulated data are pre-provided, there is a question of how to select parameters to simulate in order to generate training data. The distribution used to do this is often referred to as the *proposal* distribution, denoted by $\hat{p}(\theta)$.

If we wish to use our approximate posterior with a wide range of observed datasets, then an obvious choice for the proposal distribution is the prior $p(\theta)$. If however we only have one dataset we wish to do inference on, then it makes sense to concentrate on learning the behaviour of the simulator around plausible parameters that could have generated our observed dataset, and avoid wasting computation on simulations and inference with parameters that were highly unlikely to have generated the dataset.

A natural solution is to perform SBI over several sequential rounds. On the first round we start by running the simulator with samples drawn from the prior to generate training data. After fitting the approximate posterior to the training data on each round, on the subsequent round we draw samples from it to select parameters, adding them to the set of all simulations ran so far. In this way our simulations are concentrated around plausible parameters as we refine our approximate posterior. Algorithm 1 describes this generic approach, which we denote by adding the prefix *Sequential* to each SBI variant name.

Algorithm 1 Generic Sequential SBI

```
Input: Data x, Prior $p(\theta)$, Simulator $f(\theta)$, Approximate Posterior $q_\phi(\theta \mid \mathbf{x})$,
Simulations per Round $N$, Rounds $T$
$\hat{p}_1(\theta|\mathbf{x}) \leftarrow p(\theta)$
for $t \leftarrow 1, T$ do
    for $i \leftarrow 1, N$ do
        $\theta_{t,i} \sim \hat{p}_t(\theta|\mathbf{x})$
        $\hat{\mathbf{x}}_{t,i} \leftarrow f(\theta_{t,i})$
    end for
    Update $\phi$ using $[\{\theta_{t',i}, \hat{\mathbf{x}}_{t',i}\}_{n=1}^{N}]_{t'=1}^{t}$   ▷ Use samples from all rounds so far.
    $\hat{p}_{t+1}(\theta \mid \mathbf{x}) \leftarrow q_\phi(\theta \mid \mathbf{x})$   ▷ Use the current posterior to propose new
samples
end for
return $q_\phi(\theta \mid \mathbf{x})$
```

2.5.1 Posterior Estimation

With posterior estimation, we aim to fit a conditional density estimator $q_\phi(\theta \mid \mathbf{x})$ to directly estimate the posterior $p(\theta \mid \mathbf{x})$. If our training samples $\{\theta_i, \hat{\mathbf{x}}_i\}_{i=1}^{N}$ were produced by using the prior $p(\theta)$ as a proposal distribution, then fitting the conditional density estimator is simply a matter of finding the parameters ϕ which maximises the log-likelihood of $q_\phi(\theta \mid \mathbf{x})$, with loss function

$$\mathcal{L}(\phi, \{\theta_i, \hat{\mathbf{x}}_i\}_{i=1}^{N}) = -\sum_{i=1}^{N} \log q_\phi(\theta_i \mid \hat{\mathbf{x}}_i). \tag{2.63}$$

If $q_\phi(\theta \mid \mathbf{x})$ takes the form of a neural network, then this process is typically referred to as neural posterior estimation (NPE) (Papamakarios and Murray 2016).

A complication arises if we wish to use some *proposal* distribution $\hat{p}(\theta)$ that is not the prior when doing the sequential neural posterior estimation (SNPE) variant. Directly maximising the log-likelihood over all of the training samples will result in an estimator which does not target $p(\theta \mid \mathbf{x})$ but instead targets

$$\hat{p}(\theta \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \theta)\hat{p}(\theta)}{\int p(\mathbf{x} \mid \theta)\hat{p}(\theta) d\theta}. \tag{2.64}$$

This *proposal posterior* can be rewritten in terms of the original posterior and prior by substituting for $p(\mathbf{x} \mid \theta)$,

$$\hat{p}(\theta \mid \mathbf{x}) = \frac{p(\theta \mid \mathbf{x})p(\mathcal{D})}{p(\theta)} \cdot \frac{\hat{p}(\theta)}{\int p(\mathbf{x} \mid \theta)\hat{p}(\theta)d\theta}, \tag{2.65}$$

$$= p(\theta \mid \mathbf{x}) \cdot \frac{\hat{p}(\theta)}{p(\theta)} \cdot \frac{p(\mathcal{D})}{\int p(\mathbf{x} \mid \theta)\hat{p}(\theta)d\theta}, \tag{2.66}$$

$$= p(\theta \mid \mathbf{x}) \cdot \frac{\hat{p}(\theta)}{p(\theta)} \cdot \frac{p(\mathcal{D})}{\hat{p}(\mathbf{x})}. \tag{2.67}$$

Any density estimator trained on these proposals must be adjusted in order to target the correct posterior, and there are multiple versions of SNPE which can do this.

SNPE-A (Papamakarios and Murray 2016) trains the density estimator to target the proposal posterior $\hat{p}(\theta \mid \mathbf{x})$ using the loss from Equation 2.63, then applies a correction step after fitting to target the posterior $p(\theta \mid \mathbf{x})$. So that the correction step has a closed form solution, the prior $p(\theta)$ must be uniform or Gaussian, density estimators used as proposal distributions at intermediate steps must be (neural-network conditioned) Gaussians, and the final posterior approximation can only be a Gaussian or mixture of Gaussians.

SNPE-B (Lueckmann, Goncalves, et al. 2017) instead targets $p(\theta \mid \mathbf{x})$ directly when training, using an importance weighted loss,

$$\mathcal{L}_{\text{SNPE-B}}(\phi) = -\sum_{i=1}^{N} \frac{p(\theta_i)}{\hat{p}(\theta_i)} \log q_\phi(\theta_i \mid \hat{\mathbf{x}}_i). \tag{2.68}$$

Compared to SNPE-A, this has the advantage of having no restrictions on the choice of prior or density estimator. However, the importance weights $p(\theta_i)/\hat{p}(\theta_i)$ can be high variance, leading to slow training.

SNPE-C (Greenberg, Nonnenmacher, and Macke 2019) works by noting that if $q_\phi(\theta \mid \mathbf{x})$ estimates $p(\theta \mid \mathbf{x})$, then from Equation 2.67,

$$\hat{p}(\theta \mid \mathbf{x}) \propto p(\theta \mid \mathbf{x})\frac{\hat{p}(\theta)}{p(\theta)}, \tag{2.69}$$

and therefore

$$\hat{q}_\phi(\theta \mid \mathbf{x}) \propto q_\phi(\theta \mid \mathbf{x})\frac{\hat{p}(\theta)}{p(\theta)}, \tag{2.70}$$

$$= q_\phi(\theta \mid \mathbf{x})\frac{\hat{p}(\theta)}{p(\theta)}\frac{1}{Z(\mathbf{x}, \phi)}. \tag{2.71}$$

We can fit $q_\phi(\theta \mid \mathbf{x})$ using the loss function

$$\mathcal{L}_{SNPE-C}(\phi) = -\sum_{i=1}^{N} \log \hat{q}(\theta_i \mid \hat{\mathbf{x}}_i), \tag{2.72}$$

but to make $\hat{q}_\phi(\theta \mid \mathbf{x})$ tractable, we need the normalisation constant

$$Z(\mathbf{x}, \phi) = \int q_\phi(\theta \mid \mathbf{x}) \frac{\hat{p}(\theta)}{p(\theta)} d\theta. \tag{2.73}$$

At first glance this integral seem infeasible, unless we restrict the forms of $q_\phi(\theta \mid \mathbf{x})$, $\hat{p}(\theta)$ and $p(\theta)$ as with SNPE-A. However, we do not actually need to use the exact form of $\hat{p}(\theta)$ used to generate the samples, and can instead consider a uniform categorical distribution $P(\theta_m)$ over a set of samples $\{\theta'_m\}_{m=1}^M$. The integral in Equation 2.73 then reduces to a summation

$$Z(\mathbf{x}, \phi) = \sum_{m=1}^{M} q_\phi(\theta'_m \mid \mathbf{x}) \frac{1}{p(\theta'_m)}, \tag{2.74}$$

and hence Equation 2.71 becomes

$$\hat{q}_\phi(\theta \mid \mathbf{x}) = \frac{q_\phi(\theta \mid \mathbf{x})/p(\theta)}{\sum_{m=1}^{M} q(\theta'_m \mid \mathbf{x})/p(\theta'_m)} \tag{2.75}$$

Algorithm 2 describes the general process of fitting the SNPE variants using the selected loss.

Algorithm 2 Sequential Neural Posterior Estimation

```
Input: Data x, Prior $p(\theta)$, Simulator $f(\theta)$, Approximate Posterior $q_\phi(\theta \mid \mathbf{x})$,
Simulations per Round $N$, Rounds $T$
$\hat{p}_1(\theta|\mathbf{x}) \leftarrow p(\theta)$
for $t \leftarrow 1, T$ do
    for $i \leftarrow 1, N$ do
        $\theta_{t,i} \sim \hat{p}_t(\theta|\mathbf{x})$
        $\hat{\mathbf{x}}_{t,i} \leftarrow f(\theta_{t,i})$
    end for
    $\phi \leftarrow \text{argmin}_\phi \mathcal{L}_{\text{SNPE}}(\phi, [\{\theta_{t',i}, \hat{\mathbf{x}}_{t',i}\}, \hat{p}_{t'}(\theta \mid \mathbf{x})]_{t'=1}^{t})$
    $\hat{p}_{t+1}(\theta \mid \mathbf{x}) \leftarrow q_\phi(\theta \mid \mathbf{x})$
end for
return $q_\phi(\theta \mid \mathbf{x})$
```

2.5.2 Likelihood Estimation

Sequential neural likelihood estimation (Papamakarios, Sterratt, and Murray 2019, SNLE) builds on the idea that a density estimator can be used to approximate the likelihood $p(\mathbf{x} \mid \theta)$ instead of the posterior (Wood 2010). At the end of every round, we use all of the parameter samples and simulated data $\{\theta_i, \hat{\mathbf{x}}_i\}_{i=1}^N$ to train a conditional density estimator $\hat{q}_\phi(\mathbf{x} \mid \theta)$ by maximum likelihood. Compared to SNPE it does not require a correction step to account for the proposal distribution, but as it does not target the posterior directly an MCMC method is required to sample from $p(\theta \mid \mathbf{x}) \propto \hat{q}_\phi(\mathbf{x} \mid \theta)p(\theta)$. Depending on the simulator, it may be more or less difficult to get the conditional density estimator to approximate the likelihood compared to the posterior. Algorithm 3 describes SNLE in detail.

Algorithm 3 Sequential Neural Likelihood Estimation

```
Input: Data x, Prior $p(\theta)$, Simulator $f(\theta)$, Approximate Likelihood $\hat{q}_\phi(\mathbf{x} \mid \theta)$,
Simulations per Round $N$, Rounds $T$
$\hat{p}_1(\theta|\mathbf{x}) \leftarrow p(\theta)$
for $t \leftarrow 1, T$ do
    for $i \leftarrow 1, N$ do
        $\theta_{t,i} \sim \hat{p}_t(\theta|\mathbf{x})$        ▷ Use MCMC to draw samples from $\hat{q}_\phi(\mathbf{x} \mid \theta)p(\theta)$
        $\hat{\mathbf{x}}_{t,i} \leftarrow f(\theta_{t,i})$
    end for
    $\phi \leftarrow \text{argmax}_\phi \sum_{t'=1}^{T} \sum_{i=1}^{N} \log \hat{q}_\phi(\hat{\mathbf{x}}_{t,i} \mid \theta_{t,i})$
    $\hat{p}_{t+1}(\theta \mid \mathbf{x}) \leftarrow \hat{q}_\phi(\mathbf{x} \mid \theta)p(\theta)$        ▷ Only proportional to posterior
end for
return $\hat{q}_\phi(\mathbf{x} \mid \theta)$
```

2.5.3 Ratio Estimation

Unlike SNPE or SNLE, ratio estimation methods for SBI do not require a density estimator. They only require a probabilistic binary classifier with parameters ϕ such as a logistic regression model or a neural network with a sigmoid function on the output. Binary classification can be used to estimate density ratios (Sugiyama, Suzuki, and Kanamori 2012). Using Bayes' rule and assuming an even distribution of positive and negative samples a-priori, we can express a classifier that takes input x and outputs probability of belonging to the class

$y = 1$ as

$$p(y=1 \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y=1)}{p(\mathbf{x} \mid y=1) + p(\mathbf{x} \mid y=0)}, \tag{2.76}$$

$$= \frac{r(\mathbf{x})}{r(\mathbf{x})+1} \quad \text{where } r(\mathbf{x}) = \frac{p(\mathbf{x} \mid y=1)}{p(\mathbf{x} \mid y=0)}. \tag{2.77}$$

This means if we train a classifier to distinguish between samples of x drawn from two distributions, we can estimate the ratio between the densities of the two distributions.

Here we describe the ratio estimation method of Hermans, Begy, and Louppe (2020) for use in SBI, referred to as sequential neural ratio estimation (SNRE)-A. We can consider positive examples $y = 1$ as pairs of samples $(\hat{\mathbf{x}}, \theta)$ drawn from the joint distribution $p(\mathbf{x}, \theta)$ and negative examples $y = 0$ as pairs of samples $(\mathbf{x}, \theta)$ drawn from the marginal distributions $p(\hat{\mathbf{x}})$ and $p(\theta)$. Samples from the joint distribution can be created by sampling from the prior $p(\theta)$ then running the simulator $f(\theta)$. Marginal samples θ are just samples from the prior, whilst marginal samples x̂ can be generated by sampling the prior, running the simulator then discarding the prior sample.

By training a binary classifier to distinguish between these jointly and marginally drawn pairs, we can estimate the ratio

$$r(\mathbf{x}, \theta) = \frac{p(\mathbf{x}, \theta)}{p(\mathbf{x})p(\theta)}, \tag{2.78}$$

$$= \frac{p(\mathbf{x} \mid \theta)}{p(\mathbf{x})}. \tag{2.79}$$

We can then evaluate an estimate of the posterior PDF

$$p(\theta \mid \mathbf{x}) = r(\mathbf{x}, \theta)p(\theta) \tag{2.80}$$

and draw samples from it using MCMC methods.

To make this into a sequential variant using a previous posterior as a proposal distribution $\hat{p}(\theta)$, the only thing that changes is that the joint distribution is now $p(\mathbf{x} \mid \theta)\hat{p}(\theta)$ and the ratio estimated is

$$r(\mathbf{x}, \theta) = \frac{p(\mathbf{x}, \theta)}{\hat{p}(\theta)}. \tag{2.81}$$

This means that we can now only estimate the posterior PDF up to a constant

$$p(\theta \mid \mathbf{x}) \propto r(\mathbf{x}, \theta)p(\theta), \tag{2.82}$$

Algorithm 4 Sequential Neural Ratio Estimation

```
**Input**: Data x, Prior $p(\theta)$, Simulator $f(\theta)$, Classifier $p_\phi(y \mid \mathbf{x}, \theta)$, Simulations per Round $N$, Rounds $T$
$\hat{p}_1(\theta|\mathbf{x}) \leftarrow p(\theta)$
for $t \leftarrow 1, T$ do
    for $i \leftarrow 1, N$ do
        $\theta_{t,i} \sim \hat{p}_t(\theta|\mathbf{x})$        ▷ Use MCMC to draw samples from $r(\mathbf{x}, \theta)p(\theta)$
        $\hat{\mathbf{x}}_{t,i} \leftarrow f(\theta_{t,i})$
        $\theta'_{t,i} \sim \hat{p}_t(\theta|\mathbf{x})$        ▷ No $\hat{\mathbf{x}}$ simulated
        $\mathcal{D} \leftarrow \{\hat{\mathbf{x}}_{t,i}, \theta_{t,i}, y_{t,i} = 1\}_{i=1}^N \cup \{\hat{\mathbf{x}}_{t,i}, \theta'_{t,i}, y_{t,i} = 0\}_{i=1}^N$
    end for
    $\phi \leftarrow \text{argmax}_\phi \sum_{i=1}^{2N} \log p(y_{t,i} \mid \hat{\mathbf{x}}_{t,i}, \theta_{t,i}), \quad (\mathbf{x}_{t,i}, \theta_{t,i}, y_{t,i}) \in \mathcal{D}$
    $\hat{p}_{t+1}(\theta \mid \mathbf{x}) \leftarrow r(\mathbf{x}, \theta)p(\theta)$        ▷ Only proportional to posterior
end for
return $p_\phi(y \mid \mathbf{x}, \theta)$
```

but this still allows us to use MCMC methods to sample from it. Algorithm 4 describes this variant of SNRE in detail.

Likelihood-free inference by ratio estimation (LFIRE) is an earlier approach that estimates the ratio in Equation 2.79 using a logistic regression classifier that only takes x as input, requiring a new classifier to be trained for every value of θ. (Thomas et al. 2021). Durkan, Murray, and Papamakarios (2020) extended the approach of SNRE-A by changing the binary classifier into a multi-class classifier and posing the ratio estimation problem as a problem of determining which of K samples θ_k generated a given value x. By doing this they note similarities with the SNPE-C loss given in Equation 2.75. This variant is referred to as SNRE-B (Tejero-Cantero et al. 2020).

Chapter 3

Bayesian Inference for Ordinary Differential Equations

3.1 Introduction

Differential equations are a useful way of modelling many processes, for example in systems biology (Jones, Plank, and Sleeman 2009). Despite their ubiquity, how best to do statistical inference with them is an open question. *Competitive statistical inference for differential equations 2018* (Cside) was a competition that aimed to start resolving this problem (MacDonald 2018). In this chapter we present our entry to the competition, building on the basic example presented in Chapter 2 by providing a worked example of how to do Bayesian inference for a non-trivial scientific modelling problem.

As a demonstration problem for ordinary differential equations (ODEs), the competition organisers provided as data a noisy output from a system of coupled ODEs modelling the cardiac action potential, along with an implementation of the ODE solver used to generate the data (Simitev and Biktashev 2011). Competitors were asked to submit estimates of the parameters that generated the data, along with an estimate of the underlying noise-free ODE output. By retaining the ground truth parameters until after the competition had finished, the organisers were able to simulate a real inference problem without access to these parameters whilst being able to quantitatively judge the parameter estimates.

Here we show how we constructed our solution to the competition as a Bayesian inference problem by specifying a likelihood and prior. We describe how we used the *emcee* MCMC sampler to draw samples from the posterior, and discuss some of the practical details required to do so. Whilst none of the individual components of our procedure are novel, the best way to do inference for ODEs is not certain (as evidenced by the creation of the competition). This chapter is intended as a extended worked example rather than advocating for our method as the best way of approaching this problem (which would require the presentation of a more rigorous head-to-head comparison between methods), but our solution was effective enough to enable us to obtain first place in the ODE track of the competition, beating several alternative methods.

3.2 Methods

There are a large variety of methods available to do Bayesian inference with models using ODEs. Perhaps the most straightforward is to treat an ODE inference problem as a generic Bayesian inference problem. We specify priors over the parameters and compute the likelihood using a numerical ODE solver, then use samples from a standard MCMC method to approximate the posterior, as in Section 2.2.

We might wish to extend this approach further by using an adaptive HMC method, as in Section 2.2.2. This requires an implementation of an ODE solver which is differentiable. Many libraries implementing adaptive HMC provide such solvers as part of their framework for specifying models (Stan Development Team 2022; Salvatier, Wiecki, and Fonnesbeck 2016).

These direct methods require running a numerical ODE solver, which is potentially computationally expensive. Gradient matching is a method which avoids this requirement (Macdonald and Husmeier 2015). A surrogate model is fitted directly to the observed data, then the gradients of the observed data are estimated from the surrogate model. These estimated gradients are then used to fit the differential equations directly. This avoids the problem of running the solver, but replaces it with an additional statistical problem of inferring the gradients from noisy data.

Finally, we could make use of a simulation-based inference (SBI) approach as described in Section 2.5. Using SBI approaches to infer ODE parameters (in the context of epidemiological models) has been described as a "paradigmatic use case" (Lueckmann, Boelts, et al. 2021). In this context it would involve treating the ODE solver as a simulator with an implicit or explicit surrogate model fitted to it. This would still require running the ODE solver to produce training data for the surrogate model, but it might be possible to approximate the posterior with fewer evaluations of the solver.

3.3 Likelihood

Before deciding which approach we will use, we need to consider the model we are using. The output of the cardiac model is a time series with three channels consisting of a trans-membrane voltage E and two gating variables h and n collectively denoted as vector $\mathbf{y}$ with dimension $D = 3$. The derivatives that define the model are simplified from the original paper (Simitev and Biktashev 2011), and are given as

$$\frac{\partial E}{\partial t} = G_{Na}(E_{Na} - E)\mathbf{1}(E - E_{\star})h + \tilde{g}_2(E)n^4 + \tilde{G}(E), \tag{3.1}$$

$$\frac{\partial h}{\partial t} = F_h(\mathbf{1}(E_{\dagger} - E) - h), \tag{3.2}$$

$$\frac{\partial n}{\partial t} = F_n(\mathbf{1}(E - E_{\dagger}) - n), \tag{3.3}$$

where $\mathbf{1}(\cdot)$ is the Heaviside step function and

$$\tilde{g}_2(E) = g_{21}\mathbf{1}(E_{\dagger} - E) + g_{22}\mathbf{1}(E - E_{\dagger}), \tag{3.4}$$

$$\tilde{G}(E) = \begin{cases} k_1(E_1 - E), & E \in (-\infty, E_{\dagger}), \\ k_2(E - E_2), & E \in [E_{\dagger}, E_{\star}), \\ k_3(E_3 - E), & E \in (E_{\star}, +\infty), \end{cases} \tag{3.5}$$

$$E_2 = (\frac{k_1}{k_2} + 1)E_{\dagger} - \frac{k_1}{k_2}E_1, \tag{3.6}$$

$$E_3 = (\frac{k_2}{k_3} + 1)E_{\star} - \frac{k_2}{k_3}E_1. \tag{3.7}$$

The scalar parameters of these equations we wish to infer are

$$\theta = \{k_1, k_2, k_3, E_1, E_{Na}, E_{\dagger}, E_{\star}, F_h, F_n, G_{Na}, g_{21}, g_{22}\}. \tag{3.8}$$

Given parameters θ we can run the ODE model by passing θ and implementations of Equations 3.1–3.3 to a numerical integrator or ODE solver denoted by $f(\theta)$ to produce an output $\{\mathbf{y}^t\}_{t=1}^{T}$ at a set of pre-specified times. We also need to provide a set of initial conditions $\mathbf{y}^0$ for the solver to start from. In the context

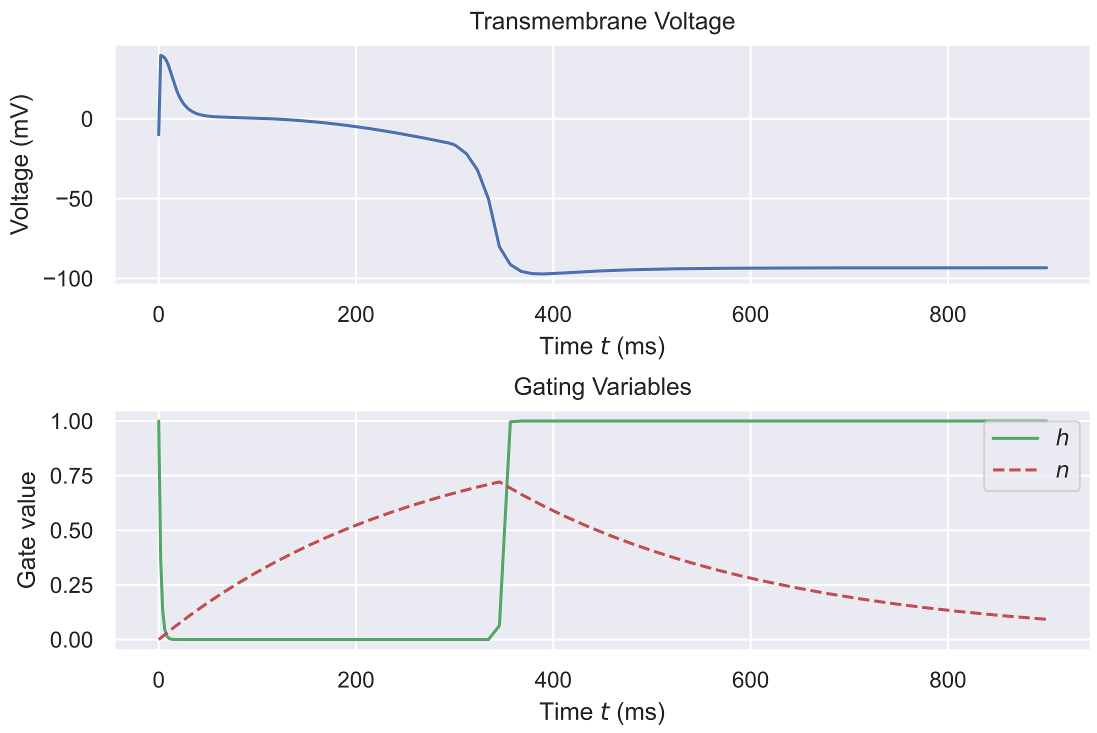


Figure 3.1: Sample output from the ODE solver for a given set of example parameters. Each channel has large changes occurring over a range of timescales.

of the Cside competition these were known and fixed at $\mathbf{y}^0 = [-10, 1, 0]^T$. If we did not know them, we could treat them as additional values to be inferred alongside θ.

Figure 3.1 shows an example output for a given set of sample parameters. We can see that in some parts the output changes rapidly over short timescales, and in others the output changes slowly over a longer timescale. The combination of varying timescales and stepwise Heaviside functions makes this a so-called stiff ODE. If we were to run this ODE with a standard solver, the numerical integrator would be forced to take very many small timesteps internally to produce an accurate solution. Instead a dedicated stiff ODE solver can run the model accurately with fewer timesteps. For the competition we used the reference implementation provided, which makes use of Matlab's *ode15s* solver (Shampine and Reichelt 1997). In doing so we eliminated the potential for model misspecification, which is useful in the context of the competition, but unrealistic for real applications.

3.3.1 Noise

The data available to estimate the parameters θ consists of a noisy sample $\mathcal{D} = \{\hat{\mathbf{y}}^t\}_{t=1}^T$ of the output of the ODE rather than the noise-free version $\{\mathbf{y}^t\}_{t=1}^T$. The noisy series is generated by adding zero-mean Gaussian noise ϵ_d^t to the noise-free output

$$\epsilon_d^t \sim \mathcal{N}(0, \sigma_d), \tag{3.9}$$

$$\hat{y}_d^t = y_d^t + \epsilon_d^t. \tag{3.10}$$

The standard deviations needed to be inferred along with the parameters, but were known to be set such that

$$\frac{\text{Var}[y_d^t]}{\sigma_d^2} \approx 10, \tag{3.11}$$

reflecting the fact that for a real experiment we would generally have a rough idea of the signal-to-noise ratio.

Figure 3.2 shows a plot of the provided data. A visual inspection shows that we can roughly identify where each signal is and where the significant changepoints are, but identifying the exact values of the signal is challenging.

Given the parameters θ and the noise scales σ_d, the total likelihood is given by

$$p(\mathcal{D} \mid \theta, \boldsymbol{\sigma}) = \prod_{t=1}^{T} \prod_{d=1}^{3} \mathcal{N}(\hat{y}_d^t \mid f(\theta)_d^t, \sigma_d^2) \tag{3.12}$$

where $f(\theta)_d^t$ is dimension d of the output of the ODE integrator at time t. The requirement to run an ODE solver makes evaluating the likelihood a comparatively computationally expensive operation. In addition, many parameter settings will result in the ODE solver failing to converge within a specified tolerance. The practical solution to this problem is to return a likelihood of zero if the solver fails.

3.4 Prior

Having considered the likelihood, we now need to choose appropriate priors $p(\theta)$. Even without being experts in mathematical physiology, we can still extract useful information from the original paper describing the model (Simitev and Biktashev 2011). The first thing to note is that many of the parameters

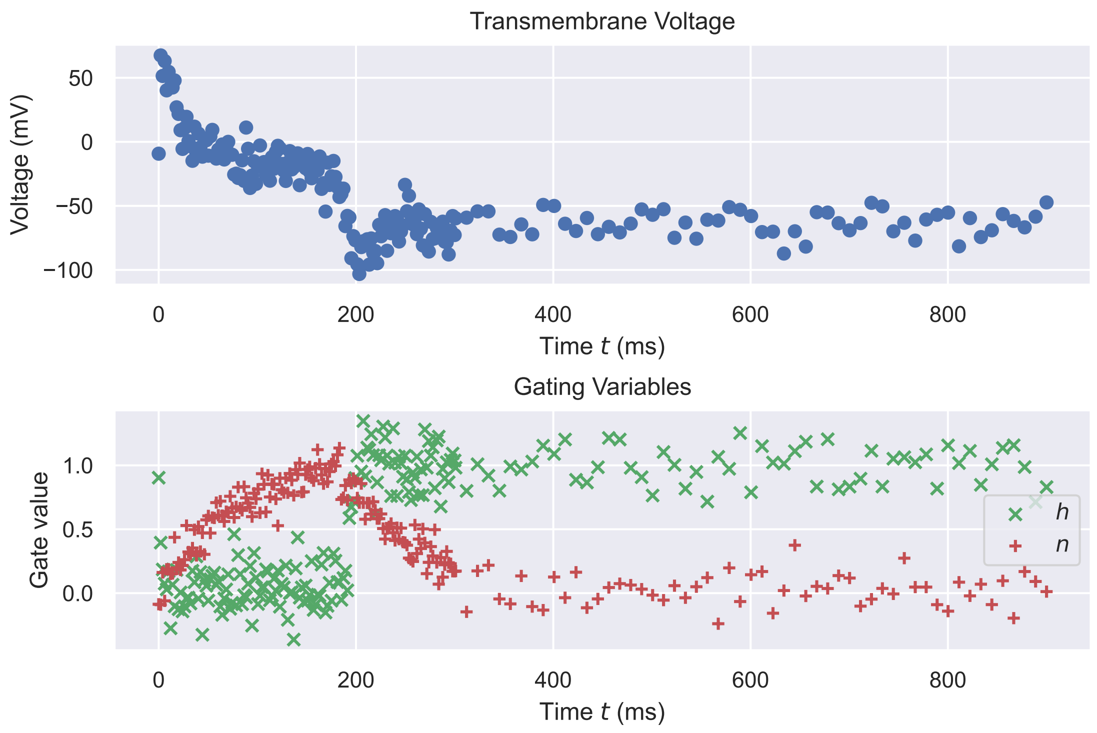


Figure 3.2: Scatter plot of the supplied data. The approximate behaviour of the ODE can be determined visually, but identifying exactly where each signal lies is challenging.

are constrained. Parameters k_1, k_2, k_3, F_h and F_n are constrained to be positive, g_{21} and g_{22} are negative, and there is an ordering constraint such that $E_1 < E_\dagger < E_\star$. In general, most optimisers and samplers work best with unconstrained variables, so we apply appropriate transforms to make the parameters unconstrained. We denote the unconstrained version of the parameters θ as $\tilde{\theta}$. We could choose put priors on the constrained parameters θ, then apply the appropriate change-of-variables correction to get $p(\tilde{\theta})$. Instead we placed priors directly on the unconstrained transformed parameters, which helps keep the implementation simple.

The second thing to note from reading the original paper is that many of the parameters have units of measurement attached to them. As an example, all of the E parameters have units of millivolts. Given this and the context (a model of a single cell in the heart), it seems unlikely that we would see values on the order of gigavolts. Therefore a reasonably informative default prior is to set Gaussian priors on each unconstrained parameter with means and scales such that the constrained parameters are restricted to plus or minus one order

Table 3.1: Unconstrained parameters and Gaussian prior values for the model.

Unconstrained Parameter	Mean	Scale
$\log k_1$, $\log k_2$, $\log k_3$	-2	1
E_1, E_{Na}	0	100
$\log(E_\dagger - E_1)$	2	1
$\log(E_\star - E_\dagger)$	3	1
$\log F_h$	0	1
$\log F_n$	-2	1
G_{Na}	0	100
$\log(-g_{21})$	0	1
$\log(-g_{22})$	1	1
$\log \sigma_1$	1	1
$\log \sigma_2$, $\log \sigma_3$	-1	1

of magnitude of the sample parameters given in the paper. Table 3.1 reports the unconstrained parametrisation and prior values used for each parameter.

Given that we do not know the exact value of the signal-to-noise ratio (SNR) defined in Equation 3.11, we will also need to infer the values of σ_d. We could exploit the fact that we know the SNR is around about 10 and set an informative prior on it, but for simplicity we instead choose to set priors directly on the values of σ_d on the log-scale.

3.5 Posterior

From Bayes' rule, the posterior over the parameters is proportional to the prior multiplied by the likelihood,

$$p(\tilde{\theta}, \boldsymbol{\sigma} \mid \mathcal{D}) \propto p(\{\hat{\mathbf{y}}^t\}_{t=1}^T \mid \tilde{\theta}, \boldsymbol{\sigma})\, p(\tilde{\theta}, \boldsymbol{\sigma}) \tag{3.13}$$

To prevent numerical underflow, it is typical to work on a log-scale,

$$\log p(\tilde{\theta}, \boldsymbol{\sigma} \mid \mathcal{D}) = \log p(\{\hat{\mathbf{y}}^t\}_{t=1}^T \mid \tilde{\theta}, \boldsymbol{\sigma}) + \log p(\tilde{\theta}, \boldsymbol{\sigma}) + C \tag{3.14}$$

where C is a constant with respect to $\tilde{\theta}$ and $\boldsymbol{\sigma}$ corresponding to the intractable log marginal-likelihood $\log p(\mathcal{D})$.

Given an implementation of this target function equal to the log-posterior up to a constant, we need to choose which method we will use to approximate the posterior. We have no reason to believe the posterior can be reasonably approximated with a Gaussian distribution, which rules out methods such as the Laplace approximation (Laplace 1774; Stigler 1986). Instead, we wish to use an MCMC-based approach to draw samples from the posterior, which we can use to infer the distribution of parameters and make predictions about the output. The likelihood is relatively expensive to evaluate, so we want any Markov Chain to propose successive samples which are as uncorrelated as possible to avoid wasting computation. We suspect the posterior may have strong correlations, so any sampler would ideally account for this when making proposals. The provided implementation of the ODE solver which we use to evaluate the likelihood is not differentiable, which prevents us from using Hamiltonian Monte Carlo (HMC, Duane et al. 1987; Neal 2011).

These criteria lead us to choose *emcee* (Foreman-Mackey et al. 2013), an implementation in Python of the Affine Invariant MCMC Ensemble sampler (Goodman and Weare 2010). *emcee* works by running a large number of chains in parallel, making proposals for each chain using the current positions of all the remaining chains, and does not require the target function to be differentiable. Provided the dimensionality of the posterior is not too high (on the order of 10), using spread out chains in this manner allows *emcee* to roughly estimate the shape of the posterior and make reasonably spaced proposals (Huijser, Goodman, and Brewer 2022). It does not in general require much tuning of parameters to work well. In our case the posterior only spans 15 dimensions, so we would expect *emcee* to work reasonably well.

To use any MCMC sampler, we need to select starting points for each chain. Random samples from an arbitrary distribution are unlikely to work well for this problem, as the parameters result in degenerate solutions from the ODE solver. Samples from the prior are better, but most still result in solver output that looks nothing like the data. Instead, we find the maximum-a-posteriori (MAP) estimate by finding the value of the parameters that maximises the target density function. As our model is not differentiable we use Powell's method, a derivative-free optimiser, to find the optimum (Powell 1964). We then add a small amount of Gaussian noise to the estimate to produce slightly different starting points for each chain.

Table 3.2: Estimates of the parameters θ, along with the effective sample size N_{ESS} and the potential scale reduction factor $\hat{R}$. The highest density posterior interval (HDPI) limits are for a 94% interval.

	Mean	SD	HDPI-L	HDPI-U	N_{ESS}	$\hat{R}$
k_1	0.14	0.04	0.07	0.21	876148.00	1.01
k_2	0.10	0.04	0.04	0.17	941552.00	1.00
k_3	0.14	0.05	0.07	0.22	331.00	1.03
E_1	-64.11	1.15	-66.24	-61.91	784711.00	1.01
E_{Na}	60.46	5.63	49.94	71.11	989449.00	1.00
$E_\dagger$	-58.52	2.82	-63.46	-53.19	618448.00	1.03
$E_\star$	-23.57	3.01	-29.21	-17.83	934756.00	1.00
F_h	0.37	0.07	0.26	0.50	759410.00	1.01
F_n	0.01	0.00	0.01	0.02	960131.00	1.00
G_{Na}	30.04	17.63	1.43	61.22	887429.00	1.00
g_{21}	-7.38	1.99	-11.23	-3.95	751634.00	1.01
g_{22}	-3.76	1.50	-6.45	-1.39	596981.00	1.03

With the starting points, we ran *emcee* with 50 parallel chains for 10,000 steps. We repeated each run a total of 4 times. After sampling, we discarded the first half of each chain as burn-in. To check the posterior estimates have converged, we computed the potential scale reduction factor $\hat{R}$ for each parameter, as well as the effective sample size N_{ESS} (Vehtari, Gelman, Simpson, et al. 2021). It is important to note that $\hat{R}$ cannot be computed directly from the parallel chains from one run of *emcee*. The calculations for $\hat{R}$ require the chains to be independent, which is not the case for the parallel chains. Instead we flatten the chains from one run and treat them as a single chain, applying the $\hat{R}$ calculations over the 4 repeated runs. We also ran our procedure with (our own) simulated data using multiple sets of parameters to check we could recover them.

3.5.1 Results

Table 3.2 reports our estimates of the parameters using summary statistics computed from our posterior samples, along with the diagnostic values. In general our posterior distribution has shrunk the estimates relative to the prior, and most of the diagnostic statistics look good. The only exception is k_3, where the effective sample size N_{ESS} is very small relative to the number of samples drawn N, and the potential scale reduction value $\hat{R}$ is away from 1.0. Increasing the number of samples does not substantially improve the diagnostic values for k_3, suggesting the sampler is having difficulties with moving the value of it. In an idealised situation we would put a more informative prior on k_3, which often helps solve sampling difficulties (Gelman, Vehtari, et al. 2020).

Figure 3.3 shows a corner plot of the posterior samples, along with the ground-truth values of the parameters used to generate the data. We did not have access to the ground-truth parameters at the time of sampling. Our posterior samples have identified all of the parameters, with the possibly exception of k_3, although the ground-truth value is still within the tails of the posterior distribution. Whilst this plot alone is not sufficient to show the correctness of our procedure (which would require a more exhaustive calibration check as done in Chapter 4), it does indicate that our posterior samples are not unreasonable.

We can see there are some strong correlations, and the posterior looks very non-Gaussian in places, justifying our decision to use an MCMC method.

The competition format required us to submit estimates of the parameters in two forms:

1. A single point estimate for each parameter, judged by the weighted root mean square error relative to the ground-truth parameters.

2. A mean and full covariance for all of the parameters, in order to judge our method's ability to quantify uncertainty. This estimate was judged by calculating the likelihood of the ground-truth parameters under a multivariate Gaussian distribution with the estimated mean and covariance.[1]

For the first estimate form, we submitted the posterior mean of our parameters, as this would minimise the expected square error. For the second estimate form, we took advantage of Bayesian inference's natural quantification of uncertainty and submitted the empirical mean and covariance of our MCMC samples. As we noted the posterior is very non-Gaussian, in doing so we are throwing away

[1]One flaw with this measurement as a competition criteria is that a competitor could deliberately submit underestimates of the uncertainty. There is an increased risk that the ground-truth parameters would be outside of your uncertainty estimate, but the payoff would be a higher score if the ground-truth parameters were still within your estimate.

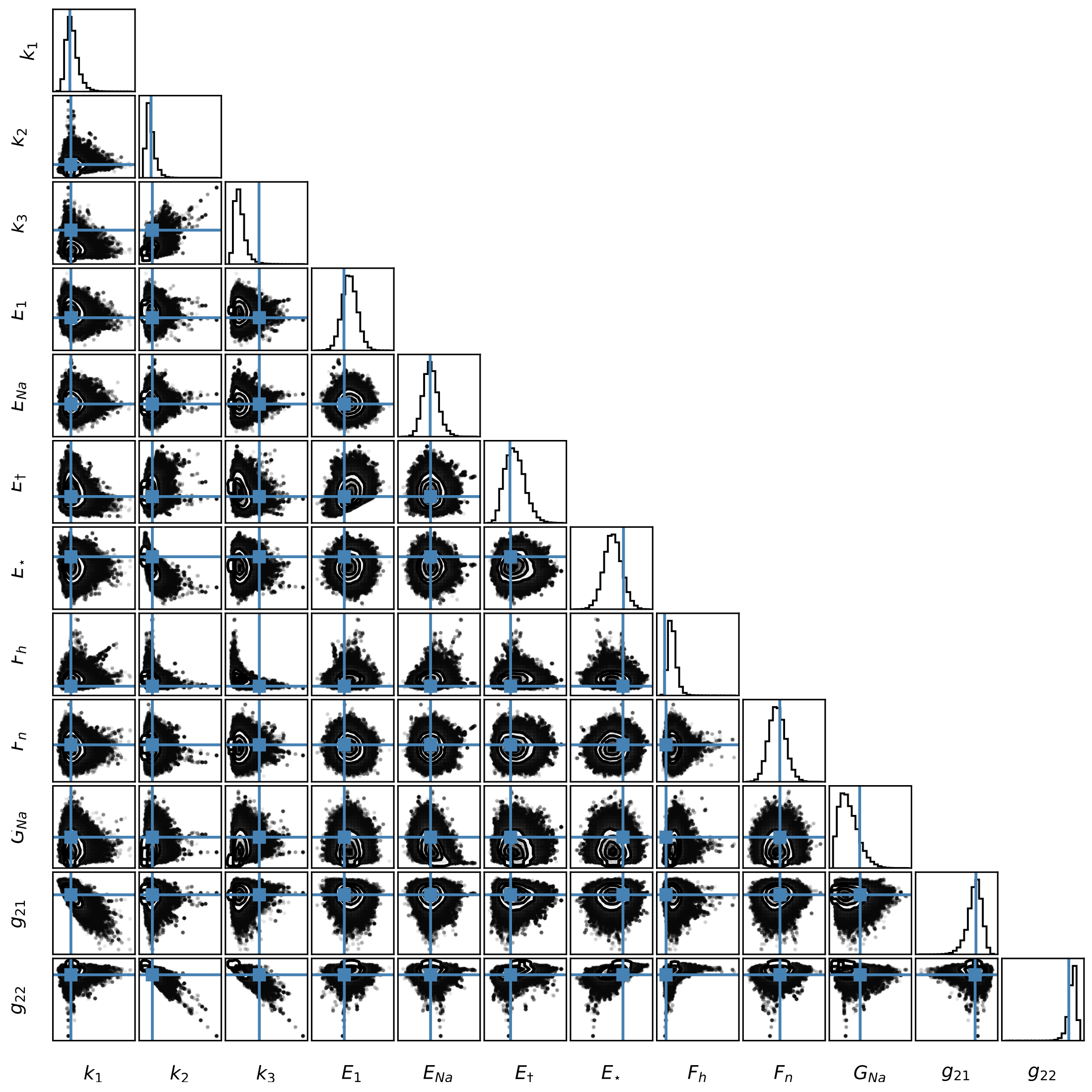


Figure 3.3: Corner plot of samples from the posterior, with markers indicating the ground-truth parameter values. All parameters are within reasonable credible intervals, except for k_3 which is still within the tails of the posterior.

information about the uncertainty in our estimates, but this a problem with the competition criteria rather than our method.

3.5.2 Posterior Predictions

A common task in Bayesian inference is use our posterior over parameters to make predictions about unseen data. For the competition, as well as submitting estimates of the parameters, we also needed to submit point estimates of the underlying values $\mathbf{y}$ used to generate the noisy observed data $\hat{\mathbf{y}}$, to be judged by the root mean squared error relative to the ground-truth signal. As with the parameter estimates a sensible choice for a point estimate of $\mathbf{y}$ is just the expectation of $\mathbf{y}$ under the posterior,

$$\begin{aligned}\mathbb{E}_{p(\theta|\mathcal{D})}[\mathbf{y}] &= \mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)] \qquad (3.15)\\ &\approx \frac{1}{N}\sum_{i=1}^{N} f(\theta^i), \quad \theta^i \sim p(\theta \mid \mathcal{D}). \qquad (3.16)\end{aligned}$$

As $f(\theta)$ is a non-linear function, in general $\mathbb{E}_{p(\theta|\mathcal{D})}[f(\theta)] \neq f(\mathbb{E}_{p(\theta|\mathcal{D})}[\theta])$. We could rerun the ODE solver $f(\theta)$ for every value θ after sampling to compute this expectation. Instead, as we needed to run the solver to compute the likelihood, it makes sense to return the output inside the MCMC sampling routine and save it for later use. Figure 3.4 shows a plot of the 94% highest-density interval over $\mathbf{y}$ using our posterior samples, along with the actual values of $\mathbf{y}$ generated from the unknown ground-truth parameters.

As our posterior did a good job of estimating the ground-truth parameters, the interval almost entirely contains the ground-truth values of $\mathbf{y}$. We were not required to submit estimates of the uncertainty in $\mathbf{y}$, but we could have done so if needed at no extra cost.

3.6 Discussion

The effectiveness of our method was externally validated by our estimates winning first place in the competition. Nevertheless there are several aspects of our procedure that could be improved on, and alternative solutions are possible.

Our priors are only very weakly informative, as they only restrict the parameters away from implausibly large values. We ran prior predictive checks, where

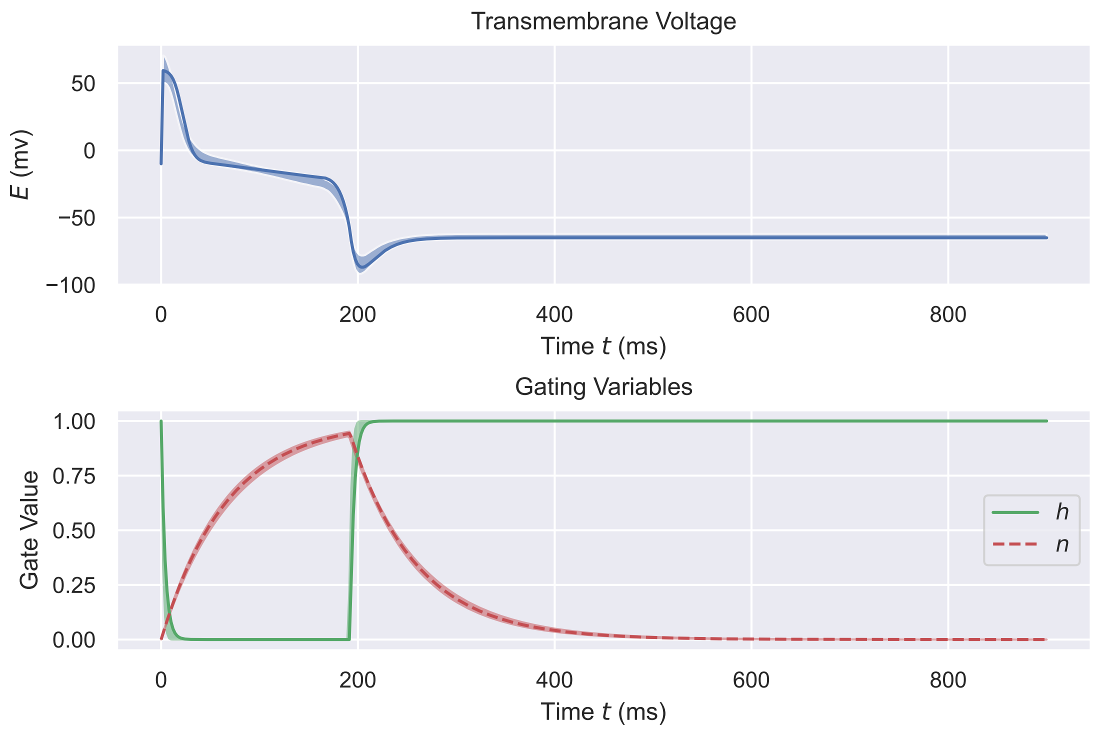


Figure 3.4: ODE solution using the ground-truth parameters use to generate the noisy data. The shaded regions indicate the 94% highest density interval computed from the posterior samples. The ground-truth solution almost entirely lies within our interval.

we draw samples from the prior and use them to simulate data (Gabry et al. 2019). These resulted in ODE outputs that whilst having reasonable values corresponded to degenerate solutions where the model collapses almost instantly to a steady state and where many of the parameters are unidentifiable. This issue also causes problems for inference checking methods such as simulation-based calibration (SBC) as they rely on generating data via simulation using prior draws (Talts et al. 2020). If we had better expert knowledge we could have more informative priors, although given the number of parameters and complexity of the model it would still be hard to specify them as distributions.

Model checking is not absolutely critical in the case of the competition, because we happen to know that the ODE solver exactly matches the data generating process. For real applications this is almost never the case, and we would like to know how well our model matches the data. One standard way to do this would be with a cross-validation procedure, with variants available that are designed to work for Bayesian inference (Vehtari, Gelman, and Gabry

2017). The inference problem as described here presents a challenge for cross-validation, as we effectively only have one independent datapoint as a result of only performing one run of the ODE solver. In a situation with real observed data, it would be advisable to obtain multiple observations where possible.

The fact that our posterior samples contain the ground-truth parameters as show in Figure 3.3 is a necessary but not sufficient condition for them to be correct. Running simulation-based calibration (SBC) checks as done in Chapter 4 would give us more confidence that our posterior samples are correct (Talts et al. 2020).

emcee would struggle if the parameter space was much larger. HMC can allow MCMC to scale to larger dimensional spaces, but requires us to take gradients with respect to the parameters (Duane et al. 1987; Neal 2011). This was not possible with the provided ODE solver by default, but other solvers are differentiable. We tried running this problem in the *Stan* probabilistic programming language, making use of its implementations of differentiable ODE solvers and adaptive HMC. We were unable to get it to work, as the warm-up phase required to tune the HMC parameters would fail due to the large number of degenerate and failed ODE solutions encountered. More informative priors could help here by restricting infeasible parameters.

Chapter 4

Inference for Pileup Processes

4.1 Introduction

In this chapter we consider another class of model where there are multiple techniques available to do inference, and no obvious solution. Pile-up is a phenomenon that occurs in astronomy when using charge-coupled device (CCD) detectors to observe bright sources (Ballet 1999). We would like to measure both the arrival times and energies of photons emitted by astronomical sources in order to characterise their spectrum and intensity. CCDs allow us to do this, by measuring the energy of each photon when they interact with a pixel of the CCD sensor. However, they can only measure the total amount of energy absorbed across a given time interval, rather than the energy of each individual photon. If multiple photons arrive during a single time interval, the CCD cannot disambiguate the energy of each, or even determine the number of photons. Figure 4.1 shows an illustration of this process.

There are a variety of changes to experimental setups that can be made to minimise the probability of pile-up occurring, but these typically involve trade-offs that reduce the quantity or quality of the data that can be collected (Chandra X-ray Science Center 2010). Consequently, it is desirable to be able to analyse data that has been affected by pile-up. The standard approach to fitting spectrums using data is to bin the data into a histogram over observed energies, compute a histogram over expected energies for a given spectrum model, and then find model parameters that minimise a goodness-of-fit statistic such as the χ^2 statistic (Arnaud, Smith, and Siemiginowska 2011, Chapter 5). If pileup is known or suspected to be present, then the spectrum model can incorporate a

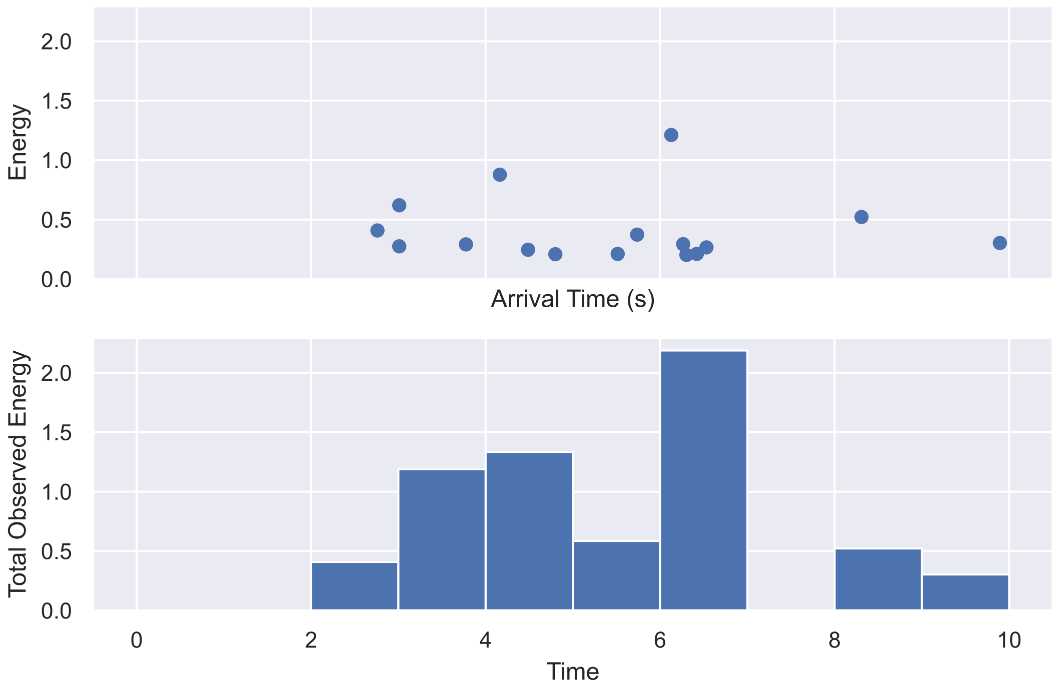


Figure 4.1: Illustration of the CCD pile-up process. The top plot shows energies and arrival times of photons from a hypothetical astronomical source that are observed by a single CCD pixel. The bottom plot shows the binned data actually read out from the CCD sensor as a result of summing the photon energies in a single timestep together.

pileup component (Davis 2001).

If we assume that the photons counts can be modelled by a Poisson distribution, then the pile-up generating process is a specific example of the general class of compound Poisson distributions. Compound poisson distributions have been used for a wide variety of modelling problems. For example, in hydrology, if the number of rainfall events in a day is assumed to be Poisson distributed, and the amount of rain per rainfall event is exponentially distributed, then the total amount of rainfall per day has a compound Poisson distribution (Revfeim 1984). Similarly, the total claim value for insurance applications can be modelled by assuming the number of claims is Poisson distributed and the value per claim is gamma distributed (Smyth and Jørgensen 2002). In this chapter we propose Bayesian inference methods for problems involving compound Poisson distributions, using (very simplified) models of pile-up as working examples.

Closed-form expressions for the probability density function of a compound

Poisson distribution exist only for certain distributions over the summand terms, such as the gamma distribution (Jorgensen 1987). This is particularly problematic for doing Bayesian inference with general compound Poisson distributions, as it will require the count variables and summand terms to be included in the posterior distribution. Sampling from such a posterior using MCMC methods in order to compute expectations under it is not trivial. The support of the posterior covers a variable high-dimensional parameter space, of which some are discrete, which causes issues for many MCMC methods.

As this is a problem for which we can easily develop a simulator, an alternative option is to make use of recent advances in the field of simulation-based inference (SBI), which only require the ability to simulate data for a given model and a set of parameters for it (Cranmer, Brehmer, and Louppe 2020). In the SBI literature the default assumption is that the likelihood function is completely intractable and unavailable, leading to the alternative name of likelihood-free inference (LFI). For many problems this is not actually the case, but the likelihood function has certain aspects which makes it difficult to use with inference methods such as MCMC which require access to it. Probabilistic programming tools such as Stan (Carpenter et al. 2017), PyMC3 (Salvatier, Wiecki, and Fonnesbeck 2016) and Pyro (Bingham et al. 2019) can make it much easier to implement these difficult likelihood functions and use them with scalable MCMC methods. An example of this from a different scientific field is the problem of inferring the parameters of susceptible-infected-recovered (SIR) models used in epidemiology, which has been described as a "paradigmatic use case" for SBI (Lueckmann, Boelts, et al. 2021). For many SIR models MCMC-based approaches are in fact viable and have been used effectively for models which inform public health policy. (Scott et al. 2020; Moore, Rosato, and Maskell 2021; UK Health Security Agency 2022).

The pileup observation model is another of this class of problems for which the likelihood function is not quite intractable enough to rule out MCMC-based approaches, but sufficiently complicated that SBI-based approaches might still be worth doing. In this chapter we develop an MCMC-based solution to problems involving pileup that bypasses the problematic aspects, and evaluate it using a range of experiments with simulated data. We also evaluate several standard SBI methods using the same experiments. We compare both approaches in terms of accuracy, flexibility, computational cost and diagnostics.

We find that SBI approaches (particularly those using ratio estimation) can be comparable in terms of accuracy whilst being easier to implement and having favourable scaling properties when used with larger datasets. However, these SBI approaches require the use of potentially expensive calibration checks in order to be reasonably certain of their correctness, due to the lack of intrinsic diagnostics available. MCMC approaches could therefore still be favourable under certain circumstances.

4.2 Model

For demonstration purposes we will make use of an extremely simplified model without units attached to the parameters. We will assume we have some photon-emitting source we wish to characterise which generates photons via a marked Poisson process, where the energy of each photon is drawn from a power-law or Pareto Type 1 distribution. We observe this source with a single CCD cell or pixel, for which the total photon energy absorbed is read out at every timestep t with unit length. The main parameters of the model are α, which controls the shape of the Pareto distribution, and λ, which denotes the rate of the Poisson process. We set priors on their values,

$$\alpha \sim \mathrm{LogNormal}(1, 0.25), \tag{4.1}$$

$$\lambda \sim \mathrm{Gamma}(2, 2). \tag{4.2}$$

Our observed data is then generated by drawing a count N_t from a Poisson distribution with rate λ for each timestep t, then drawing N_t values from the Pareto distribution parametrised by α and a minimum energy e_{min}. The observed energy E_t is then drawn by sampling from a Gaussian with mean equal to the sum of drawn energies e_{it} within each timestep t and standard deviation σ,

$$N_t \sim \mathrm{Poisson}(\lambda) \tag{4.3}$$

$$e_{it} \sim \mathrm{Pareto}(\alpha, e_{\mathrm{min}}) \tag{4.4}$$

$$E_t \sim \mathcal{N}(\sum_{i=1}^{N_t} e_{it}, \sigma) \tag{4.5}$$

We will assume that both $e_{\min}$ and σ are known, but this is not a strict requirement and the inference approaches we will describe here can be easily generalised to the case where they also need to be inferred. A compound Poisson distribution with a Pareto component is not one of the forms for which a closed-form expression for the probability density function exists, so we will need to use approximate inference methods to do inference with this model. [1].

4.3 Posterior

The parameters we are primarily interested in are α and λ. Unfortunately, the nuisance parameters N_t and e_{it} are also latent, so we will also need to infer them as well. To simplify the notation, we will consider the posterior only for a single observation E, allowing us drop the indexing over T observations.

$$p(\alpha, \lambda, N, \{e_i\}_{i=1}^{N} \mid E) \propto \mathcal{N}(E \mid \sum_{i=1}^{N} e_i, \sigma) \prod_{i=1}^{N} p(e_i \mid \alpha, e_{\min}) p(N \mid \lambda) p(\alpha) p(\lambda) \tag{4.6}$$

If we have multiple observations, the posterior will factorise easily as the values of e_{it} are independent given α and λ. This posterior is problematic when doing inference for several reasons:

1. The support can potentially have a very high dimensionality, proportional to the number of observations T. Many Bayesian inference methods do not scale well to such high-dimensional posteriors.
2. The support includes discrete parameters.
3. The exact dimensionality of the support changes depending on the value of the unknown variables N_t.

4.4 Prior Choice

Before discussing how we can do inference using this posterior, we need to discuss our choice of priors over α and λ given in Equations 4.4 and 4.5. Careful prior choice is necessary to avoid unreasonable posteriors which can cause

[1]In fact, there is no easily tractable expression for the sum of Pareto random variables even when the number of terms in the sum is known (Nadarajah, Zhang, and Pogány 2018)

issues for inference (Gelman, Simpson, and Betancourt 2017). For this simple demonstration model, we will assume both α and λ are somewhere on the order of 1. Uniform priors are often not a good idea unless we have very good reason to believe that the system we are modelling imposes hard constraints on the parameter values, which is not the case here. Even using a reparametrisation to remove the constraints, a uniform prior can induce unreasonable posterior behaviour, causing problems for many sampling methods (Gelman and Yao 2021).

Given our assumptions about the parameter scales and a bias against uniform priors, it might therefore seem reasonable to use something like a $\mathrm{HalfNormal}(0, 1)$ prior to constrain the order of magnitude of the parameters. To validate our choice of prior, we should then run a prior predictive check by simulating data using draws from the prior. The top row of Figure 4.2 shows a boxplot of sampled energy values of e_i when using such a prior on α. Whilst the interquartile range covers a single order of magnitude, the 99% range spans about 23 orders of magnitude (wider than all of the named parts of the electromagnetic spectrum!), and the typical maximum value of the observed energy was around 10^{37}.

Clearly, what initially seems to be a reasonably weakly informative prior is in fact not quite so reasonable when we run a prior predictive check. This unreasonable behaviour happens because the Pareto distribution becomes extremely pathological when the value of α approaches zero. To prevent this from happening, we need to use a boundary-avoiding prior to keep the value of α away from zero, hence the $\mathrm{LogNormal}(1.0, 0.25)$ prior. The bottom row of Figure 4.2 shows a boxplot when we rerun our predictive check with this prior. The values of e_i are now much more reasonable.

The prior on λ is not quite so critical for avoiding extreme behaviour. We assume that we will observe some photons even if zero counts are possible, so we wish to keep the value of λ away from zero, hence the $\mathrm{Gamma}(2, 2)$ distribution. One further interesting aspect of the Pareto distribution is that it has an indefinite mean for $\alpha \leq 1$, and indefinite variance for $\alpha \leq 2$. Whilst this does not necessarily cause problems for inference, if we were to believe that the value of α was not likely to cover these limits, a prior keeping the value away from them might be a reasonable choice.

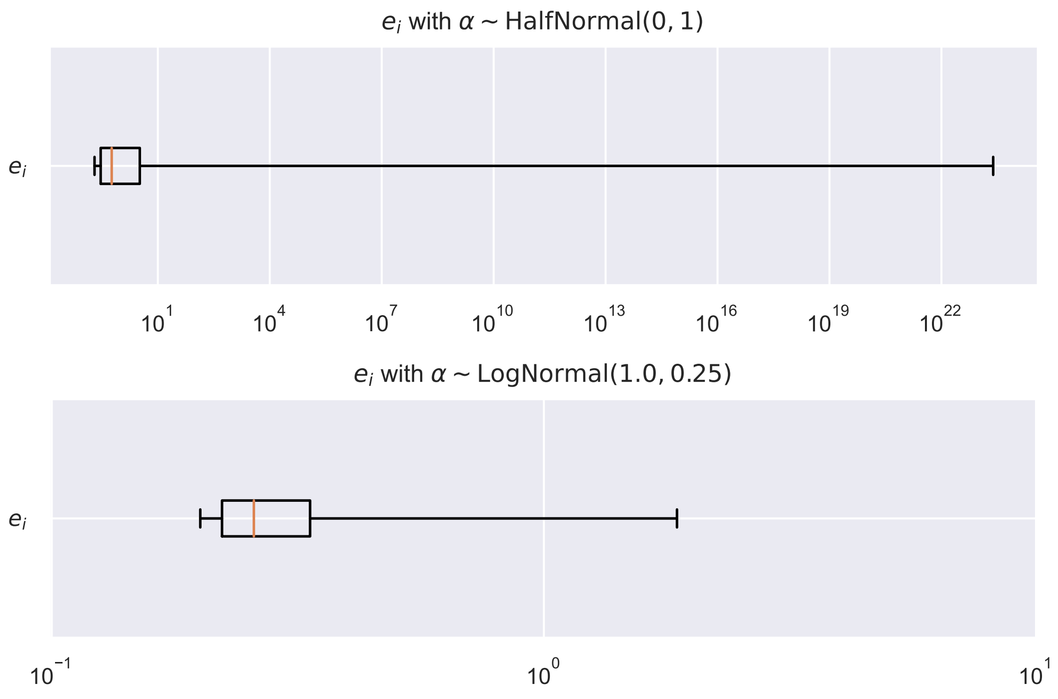


Figure 4.2: Boxplot of sample values of e_i with different priors on α. The box shows the median and interquartile range, and the whiskers show the 0.5% and 99.5% quantiles. Note the different scales between rows.

4.5 Markov Chain Monte Carlo

A standard method for estimating expectations under the posterior is to use Markov chain Monte Carlo (MCMC) methods to draw samples from the posterior, and to use the samples to compute Monte Carlo estimates of expectations of interest. On their own, each of the problematic aspects of our posterior can be dealt with using an appropriate MCMC scheme. Standard Metropolis-Hastings and Gibbs samplers can work with discrete variables. Reversible Jump MCMC permits inference with posteriors of varying dimensionality (Green 1995). General MCMC methods exist that can scale to some high dimensional posteriors.

However, none of the above solutions can work with all three problematic aspects combined. General discrete samplers do not scale well to high dimensions, and neither does Reversible Jump MCMC. Samplers that can work with high-dimensional posteriors require the parameter size to be fixed and the parameters to be continuous in order to take gradients with respect to the potential. Here we present a solution that allows us to overcome these combined issues and use MCMC methods with this posterior.

4.5.1 Marginalisation

The issues of the posterior containing both discrete parameters and variable numbers of parameters can both be solved by marginalising out N_T directly. To understand how this process works, it is helpful to consider a different but equivalent representation of the generative model outlined in Section 4.2. We will again consider only a single observation to clarify the notation by dropping the index t. The generative process still starts by drawing values of α and λ from the prior,

$$\alpha \sim \mathrm{LogNormal}(1, 0.25), \tag{4.7}$$

$$\lambda \sim \mathrm{Gamma}(2, 2). \tag{4.8}$$

However, rather than proceeding to then sample a value of N, we first sample sets $\{e_m^M\}_{m=1}^M$ with values of M corresponding to all possible values of N,

$$e_m^M \sim \mathrm{Pareto}(\alpha, e_{\mathrm{min}}) \text{ for } m \text{ in } [1, \ldots, M],\ M \text{ in } [0, \ldots, \infty]. \tag{4.9}$$

We then sample a value of N, and use it to select the set $\{e_m^M\}_{m=1}^M$ where $M = N$ to produce the observed energy E. This generative process produces identical results to the previous process, but has a different posterior as we now need to consider all of the possible sets of $\{e_m^M\}_{m=1}^M$.

$$p(\alpha, \lambda, N, \left\{\{e_m^M\}_{m=1}^M\right\}_{M=0}^{\infty} \mid E) \propto \\ \mathcal{N}(E \mid \sum_{i=1}^{N} e_i^N, \sigma) \prod_{M=0}^{\infty} \prod_{m=1}^{M} \left[p(e_m^M \mid \alpha, e_{\mathrm{min}})\right] p(N \mid \lambda) p(\alpha) p(\lambda) \tag{4.10}$$

The value of N can then be marginalised out of the posterior by summing over it,

$$p(\alpha, \lambda, \left\{ \{e_m^M\}_{m=1}^M \right\}_{M=0}^\infty \mid E) = \sum_{N=0}^{\infty} p(\alpha, \lambda, N, \left\{ \{e_m^M\}_{m=1}^M \right\}_{M=0}^\infty \mid E) \tag{4.11}$$

$$\propto \sum_{N=0}^{\infty} \mathcal{N}(E \mid \sum_{i=1}^{N} e_i^N, \sigma) \prod_{M=0}^{\infty} \prod_{m=1}^{M} \left[p(e_m^M \mid \alpha, e_{\min}) \right] p(N \mid \lambda) p(\alpha) p(\lambda) \tag{4.12}$$

$$\propto p(\alpha) p(\lambda) \prod_{M=0}^{\infty} \prod_{m=1}^{M} \left[p(e_m^M \mid \alpha, e_{\min}) \right] \sum_{N=0}^{\infty} \left[\mathcal{N}(E \mid \sum_{i=1}^{N} e_i^N, \sigma) p(N \mid \lambda) \right] \tag{4.13}$$

This result could have been arrived at by directly marginalising the posterior presented in Equation 4.6, but this presentation makes it easier to see how the values e_m^M are actually independent of N and can therefore be taken outside of the summation. If we extend the posterior to multiple observations, the marginalisation is still easy to calculate as the posterior still factorises, avoiding an exponential T term in the complexity of the computation.

In practice, because the prior limits the plausible values of N, we can truncate the infinite summation and product at a cut-off $N_{\max} = M_{\max}$. In doing this marginalisation, we have removed the discrete parameters and fixed the dimensionality of the support.

4.5.2 Hamiltonian Monte Carlo

Having solved the issue of discrete variables and a varying dimensionality support, we still have the issue of the high dimensionality of the posterior. The dimensionality D is proportional to $TN_{\max}^2$. Even with only $T = 100$ observations and truncating the marginalisation at $N_{\max} = 10$, our posterior will have dimensionality $D = 552$.

Fortunately, as the marginalisation has removed the discrete variables, we can take advantage of the ability to differentiate the unnormalised log-density function with respect to the parameters and make use of Hamiltonian Monte Carlo (HMC), an MCMC method which can scale to very high-dimensional posteriors (Duane et al. 1987; Neal 2011). Section 2.2 provides a brief review of HMC. HMC is very sensitive to the choice of several tuning parameters, so in practice an adaptive variant is almost always used (Hoffman and Gelman 2014; Betancourt 2018; Stan Development Team 2022).

4.5.3 Reparametrisation

HMC is sensitive to constraints on the values of the parameters. It is therefore standard practice to reparameterise the posterior so that the support of each sampled variable is unconstrained. For example, as both α and λ are constrained to be positive, we instead sample $\log \alpha$ and $\log \lambda$ directly, which have unconstrained support. As the density is still evaluated on α and λ, we need to perform a change-of-variables adjustment using the log-Jacobian-determinant when computing the potential function. Here we consider some additional reparametrisations which could improve the performance of our HMC sampler.

4.5.3.1 Rescaling Transformation

Under the prior, there will be a strong correlation between the values of α and e^M_{mt}. If the likelihood is not sufficiently informative, then this will also lead to strong correlations in the posterior. Whilst adaptive HMC can work even in the presence of strong correlations between parameters, it will still work better if those correlations can be removed via an appropriate reparametrisation (Stan Development Team 2022).

Fortunately, such a reparametrisation exists for the Pareto distribution. Rather than sampling e_{it} directly from the Pareto distribution, it can be sampled as follows,

$$z^M_{mt} \sim \text{Exponential}(1), \tag{4.14}$$

$$e^M_{mt} = \exp(\frac{z^M_{mt}}{\alpha}) \cdot e_{\min}. \tag{4.15}$$

This reparametrisation works as whenever a new value of α is proposed, the values of e^M_{mt} will be automatically consistent with it, making it easier for the chain to mix well. As we are only evaluating the density on z^M_{mt}, we do not need to make a change-of-variables adjustment. We will refer to this reparametrisation as the rescaling parametrisation.

4.5.3.2 Fractions Transformation

For a given value of t and M the corresponding latent energies e^M_{mt} will be highly correlated with each other, as they must add up to some value close to the observed energy E_t. In the limit of zero noise, they will be hard-constrained to lie on a hyperplane and the final energy can be completely determined by the previous $M-1$ energies. With non-zero noise, they will be soft-constrained to lie close to a hyperplane. If we can eliminate this soft-constraint, HMC might be able to propose more independent samples and mix more easily.

One possible method that would allow this soft-constraint removal is to reparameterise it in terms of a latent total energy $\hat{E}_t$ and fractions f_{it}^M that describe the proportion of the total energy to be allocated to each latent photon energy,

$$e_{mt}^M = \hat{E}_t \cdot f_{mt}^M, \tag{4.16}$$

$$e_{mt}^M \sim \text{Pareto}(\alpha, e_{\min}), \tag{4.17}$$

$$E_t \sim \mathcal{N}(\hat{E}_t, \sigma). \tag{4.18}$$

The fractions f_{mt}^M can be themselves be parametrised from a unconstrained space of M-1 variables using a stick-breaking transformation (Stan Development Team 2022). We take $\{u_{mt}^M\}_{m=1}^{M-1}$ unconstrained variables, map them into the range $[0,1]$ with a sigmoid transformation, then use each mapped variable in turn to describe the remainder of a stick to be broken off to form f_{mt}^M, with the stick starting at unit length. The final fraction is whatever is left over from the unit stick.

As the Pareto distribution has a minimum cut-off $e_{\min}$, there is an additional constraint that the latent total energy must be no less than $M \cdot e_{\min}$ for each case of M in the marginalisation. Thus we will need a latent energy $\hat{E}_t^M$ for each count in order to be entirely unconstrained, rather than sharing one value of $\hat{E}_t$ over all counts. As we evaluate the density on e_{mt}^M rather than f_{mt}^M, we will need to make a change-of-variables adjustment. We will refer to this reparametrisation as the fractions parametrisation.

4.5.4 Experiments

4.5.4.1 Parameter Recovery

As an initial check that our MCMC approach is feasible, we attempt to recover the parameters used to generate some synthetic data. We also use these experiments to compare the efficiency of the different possible reparametrisations we have identified. We set the values of α and λ as shown in Table 4.1 then created a synthetic dataset with $T = 100$ observations using our generative model and

Parameter	Value
α	2.5
λ	1.0
σ	0.01
$e_{\min}$	0.2

Table 4.1: Parameter values used for the parameter recovery experiments.

Table 4.2: Estimates of the parameters α and λ for each MCMC parametrisation, along with diagnostic statistics.

		Mean	SD	HDPI-L	HDPI-U	N_{ESS}	$\hat{R}$
	Parameter						
Standard	α	2.334	0.273	1.853	2.853	298.0	1.02
	λ	1.114	0.122	0.884	1.339	1378.0	1.00
Rescaled	α	2.343	0.277	1.836	2.860	713.0	1.00
	λ	1.111	0.121	0.881	1.335	1854.0	1.00
Fractions	α	2.345	0.272	1.854	2.861	337.0	1.01
	λ	1.103	0.119	0.871	1.319	1378.0	1.01

a fixed random seed. Our MCMC experiments were written with JAX using NumPyro's implementation of adaptive HMC (Bradbury et al. 2018; Bingham et al. 2019; Phan, Pradhan, and Jankowiak 2019). For each parametrisation, we ran 4 independent chains for 5000 steps each, with 5000 warm-up steps.

Table 4.2 summarises the posterior mean, standard deviation and 94% highest density posterior interval (HDPI) for both α and λ across each parametrisation, as well as the effective sample size (N_{ESS}) and the potential scale reduction factor ($\hat{R}$) (Vehtari, Gelman, Simpson, et al. 2021). We can see that the estimates of the parameter values are more or less identical down to the second or third decimal place. Figure 4.3 provides an alternative visualisation of the parameter estimates, clearly indicating that they are self-consistent and reasonably consistent across parametrisations. Figures 4.4, 4.5 and 4.6 show corner plots of the samples of α and λ for each parametrisation. Consistent with the results from the table, the posteriors look to be identically shaped, and contain the true parameters used to generate the data.

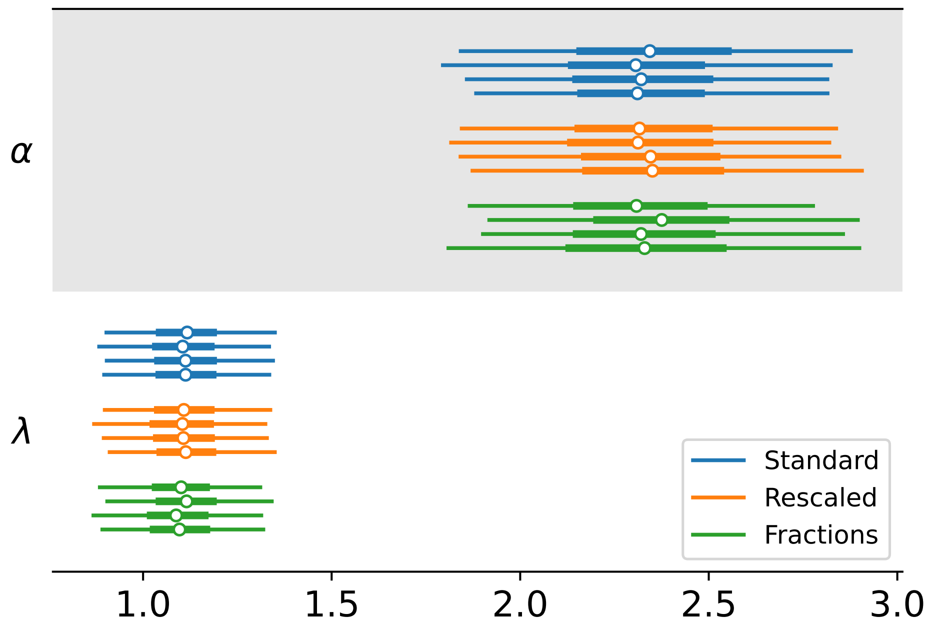


Figure 4.3: Forest plot showing the 94% HDPI, interquartile range and median for all MCMC chains using each parametrisation.

Where the parametrisations differ is in the values of the diagnostic statistics. The value of N_{ESS} for α and the rescaled parametrisation is more than twice that of the other parametrisations. This shows that the chains with the rescaled parametrisation have better mixing with more independent samples.

The $\hat{R}$ value shows how consistent the estimates of the parameters are both within the chains and across them. A value close to 1.0 for all parameters indicates that the chains have converged. The values in the table indicate that we can be much more certain that the rescaled parametrisation has converged compared to the other parametrisations. Whilst not presented in the table, $\hat{R}$ values were also computed for the remaining parameters as well as α and λ. For the rescaled parametrisation, no parameters had $\hat{R}$ greater than 1.01, whilst the standard parametrisation had 4 parameters greater than 1.01. The fractions parametrisation had 9 parameters with $\hat{R}$ over 1.01, with a value of 1.29 for one parameter, suggesting that the chains have failed to converge properly.

There are also useful diagnostics to consider which are specific to HMC. If the posterior contains pathological regions of high curvature, the numerical integrator used to simulate the Hamiltonian trajectory will show increasing error if the integrator step-size is not sufficiently small, referred to as "divergences" (Betancourt 2018). No divergences were found in any of the chains for all of the parametrisations. None of the chains exceeded the maximum number of allowable steps when simulating each Hamiltonian trajectory, another good indication that the chains are mixing well.

The Bayesian fraction of missing information (BFMI) is a diagnostic value which indicates whether the warmup procedure has done a good job of tuning the mass matrix needed to compute the momentum in the Hamiltonian (Betancourt 2016a). Failing to tune this properly can result in chains which struggle to reach the tails of the posterior. All four chains for the standard parametrisation are below the empirically recommended threshold of 0.3, whilst two out of four chains are below the threshold for the fractions parametrisation and the remaining two are only marginally above it. In contrast, every chain for the rescaled parametrisation is well above the threshold. In general, these results suggest that parameter estimates using the rescaled parametrisation are the most trustworthy, as the chains are able to mix well more easily, explore the tails of the posterior, and converge to the same estimates.

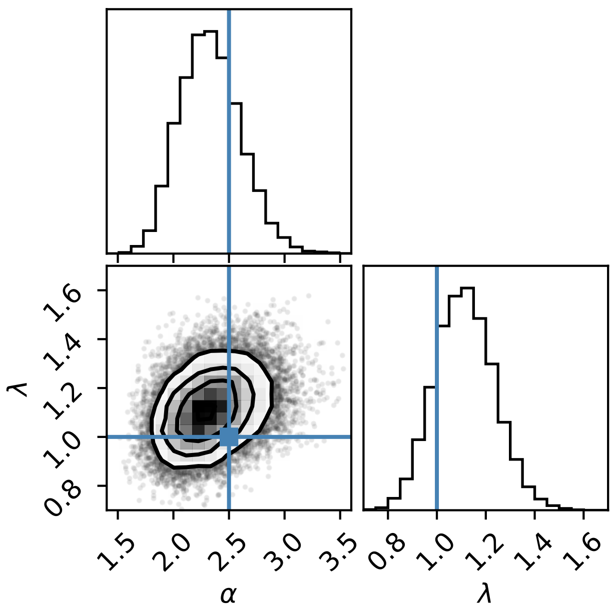


Figure 4.4: Corner plot with 1D marginals and a 2D joint plot of MCMC posterior samples of α and λ using the standard parametrisation. The blue lines indicate the ground-truth parameter values. The posterior contains the ground-truth values.

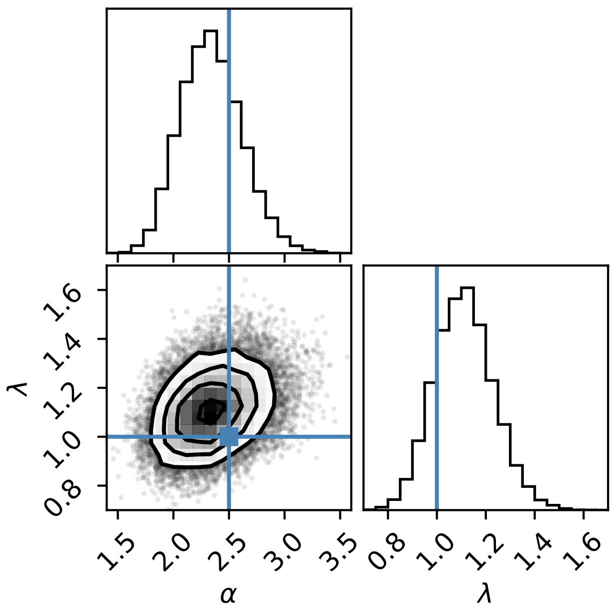


Figure 4.5: Corner plot of MCMC posterior samples using the rescaled parametrisation with the same format as Figure 4.4.

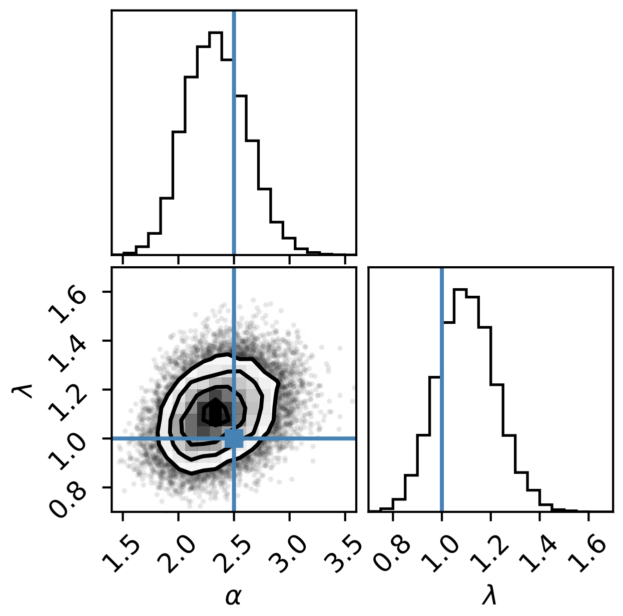


Figure 4.6: Corner plot of MCMC posterior samples using the fractions parametrisation with the same format as Figure 4.4.

4.5.4.2 Simulation-Based Calibration

The fact that our MCMC procedure can recover some example parameters is not sufficient evidence to indicate that it works properly. The approximate posterior could be too narrow or too broad compared to the true posterior, or biased in one particular direction, whilst still containing the true parameters. The $\hat{R}$ diagnostics indicate that the independent chains have converged to the same distribution, not that they have converged to the correct distribution.

Simulation-based calibration (SBC) is a method which can help diagnose a procedure which is producing incorrect posterior samples (Talts et al. 2020). The key idea of SBC is that if we have a prior $p(\theta)$ and a likelihood $p(y \mid \theta)$ that we can simulate from, we can generate simulated data $\hat{y}$ using samples $\hat{\theta}$ from the prior. If we then use our procedure under investigation to generate samples from the posterior $p(\theta \mid \hat{y})$ for each set of simulated data, the posterior averaged over the data should be equal to the prior,

$$p(\theta) = \int p(\theta \mid \hat{y}) p(\hat{y} \mid \hat{\theta}) p(\hat{\theta}) d\hat{y} d\hat{\theta}. \tag{4.19}$$

In order to evaluate the equality of the prior and data-averaged posterior, the rank statistic of each parameter in the prior sample can be computed relevant to the posterior samples for each simulated dataset. If the posterior is correct, the distribution of rank statistics should be uniform. This can be tested by plotting the rank statistics as a histogram. A too-broad posterior will result in the extreme ranks being under-represented, whilst conversely a too-narrow posterior will have the extreme ranks over-represented. A biased posterior will have one side of the extreme ranks over-represented and the other under-represented.

To evaluate our MCMC implementation using SBC, we simulated 640 datasets of $T = 100$ observations using draws from the prior. As our previous experiments indicated that the rescaled parametrisation was the most reliable, we restricted our experiments to that parametrisation. For each simulated dataset, we ran a single chain for 5000 steps with 5000 warmup steps. The 5000 samples were uniformly thinned to produce 31 independent samples.

Figure 4.7 shows the histogram of ranks obtained for both α and λ. The ranks have been rebinned by a factor of 2. The horizontal dashed lines indicate the expected 99% coverage interval assuming uniformly distributed ranks. In general the histograms look reasonably uniform, suggesting that the posterior is not too broad or narrow or biased in one direction.

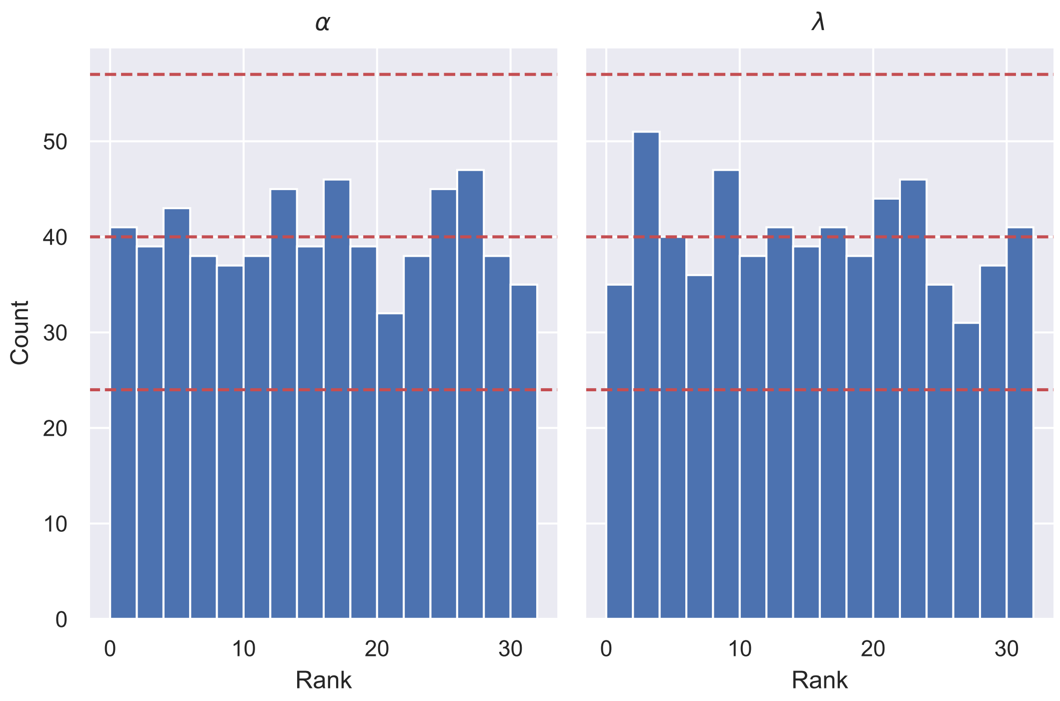


Figure 4.7: Histograms of the prior sample ranks with respected to the posterior samples when running SBC using our MCMC implementation. Dashed lines indicate the median and 99% interval assuming a uniform distribution of ranks.

Another visual method of checking the rank statistics for a uniform distribution is to plot the empirical CDF (ECDF) against the 99% expected interval and the median ECDF assuming uniformity. Figure 4.8 shows the ECDF along with the expected 99% interval and median on the top row for both α and λ. The bottom row shows the ECDFs with the expected median subtracted, allowing us to see the deviations more clearly. We can see that the ECDF stays within the expected interval, suggesting the ranks are reasonably close to uniformly distributed and that the posterior is not obviously deficient.

These experiments have shown that MCMC is a potentially viable option when doing inference using CCD data with an observation model involving pileup. However, the implementation of this approach was not straightforward. Unlike the probabilistic programming paradigm where we would just write down a generative model and have the language compute the posterior and sample from it automatically, we had to derive an appropriate posterior, then

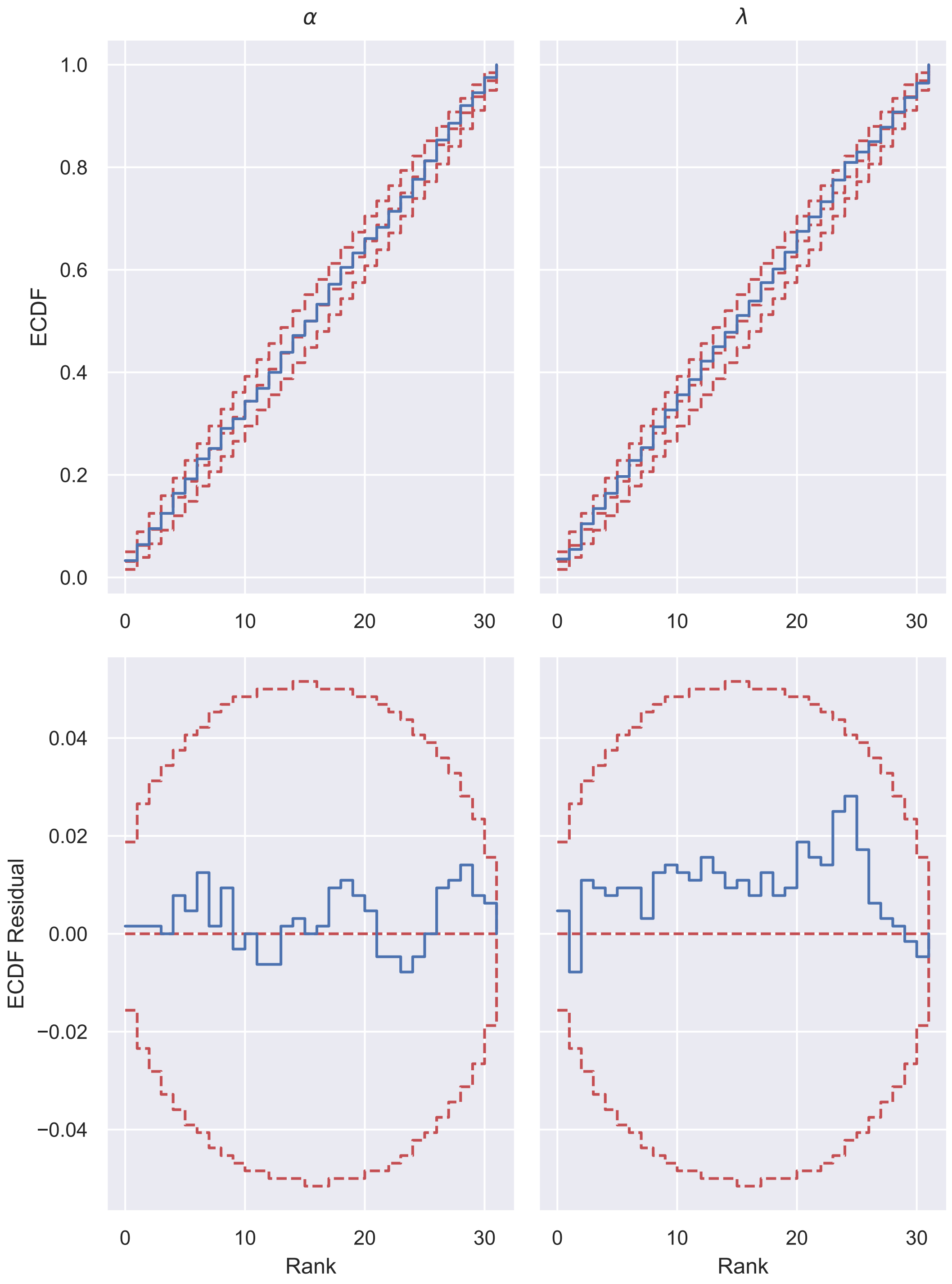


Figure 4.8: The top row shows plots of the Empirical CDF of the SBC prior sample ranks, along with dashed lines indicting the 99% expected interval and the median. The bottom row shows the same data but with the median subtracted from each line, making the differences clearer.

write a corresponding marginalised log-density function by hand. If we were to use a more realistic pileup model we would need to re-derive a new posterior.

This could be a steep barrier against widespread adoption of this method. Some work has been done to enable probabilistic programming languages to assist the user by performing marginalisation of discrete variables automatically, but this is typically still experimental and restricted to bounded distributions (Bingham et al. 2019; Gorinova et al. 2021). There are also issues around choosing a truncation value $N_{\max}$. If we select too large a value, our MCMC chains will waste time exploring parameter values that are incredibly unlikely to have generated the observed data, but if we select a value that is too small our posterior estimates will be biased.

4.6 Simulation Based Inference

4.6.1 Background

Simulation-based inference (SBI) methods are a family of techniques for doing Bayesian inference when the likelihood is intractable or otherwise presents difficulties for standard inference methods. SBI methods only require access to a prior $p(\theta)$, some observed data $\tilde{\mathbf{x}}$, and a function which simulates our model by taking parameters θ as inputs (α and λ in our case), and produces stochastic outputs $\mathbf{x}$ (the energies E_t in our case). In doing so we can ignore the nuisance random variables within the stochastic simulator and only produce posterior estimates for expectations involving the parameters of interest. We provide a more in-depth summary of SBI methods in Section 2.5.1

SBI methods work by selecting samples θ (often drawn from the prior), running the simulator for each sample to produce simulated data $\mathbf{x}$, then using pairs of $\{\theta, \mathbf{x}\}$ to learn the behaviour of the simulator and fit the posterior $p(\theta \mid \mathbf{x})$. Some newer methods make use of a neural network-based conditional density estimator such as a mixture density network (Bishop 1994) or normalising flow (Papamakarios, Nalisnick, et al. 2021) to learn a particular conditional distribution. After training, we can then use our observed data $\tilde{\mathbf{x}}$ as an input to draw samples from the posterior $p(\theta|\tilde{\mathbf{x}})$.

A more efficient process for some SBI methods is to proceed over several rounds. On the first round we use θ samples from the prior for initial simulations as before. On subsequent rounds we use θ samples from the previous round's posterior estimate $p(\theta \mid \tilde{x})$ to run simulations, making use of our observed data x̃. In doing so we can concentrate on parameters that were more likely to have generated our data, and avoid wasting computation time learning the behaviour of the simulator with parameters that were very unlikely to have generated our data. These variants of SBI are typical referred to as sequential methods.

Below we discuss the setups for the three main variants of sequential SBI methods we will be considering for this problem. Section 2.5 contains a more comprehensive review of neural SBI methods.

4.6.1.1 Posterior Estimation

Sequential neural posterior estimation (SNPE) uses a neural-network based conditional density estimator to learn the posterior $p(\theta \mid \mathbf{x})$ directly. For our experiments we used the SNPE-C variant, using a masked autoregressive flow (MAF) as the conditional density estimator (Papamakarios, Pavlakou, and Murray 2017). Sampling from the conditional density estimator can be generally be done quickly without the need for an MCMC scheme.

4.6.1.2 Likelihood Estimation

Sequential neural likelihood estimation (SNLE) also uses neural network-based conditional density estimator, but tackles the problem from a different direction, learning the likelihood $p(\mathbf{x} \mid \theta)$ instead (Papamakarios, Sterratt, and Murray 2019). In conjunction with the prior $p(\theta)$, we can then use the learned likelihood with an MCMC scheme to draw samples from the posterior. When published SNLE originally had the advantage over SNPE-A of being able to use arbitrary density estimators, but the subsequent improvements to SNPE have removed this difference. Fitting a conditional density estimator to x can sometimes be challenging if the data contains atoms of singular density. The density estimator can easily maximise both the training and validation log-likelihoods by placing all of the mass on the atoms and ignoring the rest of the data. As with SNPE we use a MAF as the conditional density estimator for our experiments.

4.6.1.3 Ratio Estimation

Sequential neural ratio estimation (SNRE, Hermans, Begy, and Louppe 2020) builds on the likelihood-free inference by ratio estimation method (LFIRE, Thomas et al. 2021). This procedure works by using a neural network to perform classification to estimate the ratio $\frac{p(\theta,\mathbf{x})}{p(\mathbf{x})p(\theta)}$ and to account for some training pairs being drawn using the estimated posterior from previous iterations instead of the prior. The advantage of SNRE compared to SNLE or SNPE is that in general it is much easier to fit a classifier than a conditional density estimator. However, like SNLE, it only outputs a function proportional to the posterior up to a constant, so an MCMC scheme is needed to produce samples from the posterior. For the experiments we use a version of SNRE described by Durkan, Murray, and Papamakarios (2020), sometimes referred to as SNRE-B.

4.6.2 Summary Statistics

As the data outputs from the simulators we would like to use are often high-dimensional and reasonably high variance, it is typically necessary to use a set of summary statistics as inputs for the conditional density estimator or classifier. Designing an appropriate set of summary statistics can be difficult, balancing the need to capture all of the relevant information contained in the data against the need to reduce variance and dimensionality. For this application, as we assume the observed energies E_t are independent and identically distributed (IID), one possible option would be to summarise them with a normalised histogram. To do this we need to fix the width and locations of the histogram, but this causes problems when the data can span different ranges.

Instead of using a histogram, we could alternatively use the quantiles of the series to summarise it. This is effectively equivalent to using a histogram, but with the bin heights fixed in advance and the bin widths fitted to the data. The advantage of this approach is that the summary statistic is now no longer suited only to a particular scale. This is particular critical for making the calibration experiments work, as under even an informative prior observed energies can span several orders of magnitude, as shown in Section 4.4. For our experiments we used 20 linearly spaced quantiles from 5% to 95% as summary statistics, fitted to the observed energies from the simulated model on a log-scale.

4.6.3 Experiments

4.6.3.1 Parameter Recovery

To evaluate each SBI method for this task, we repeated the parameter recovery experiment from Section 4.5.4.1. Following standard practice in the SBI literature, we run the experiments for each method with a varying number of simulations, using 10 rounds of sequential sampling and fitting for each simulation budget. We used the `SBI` Python package to implement each method (Tejero-Cantero et al. 2020). For the SNPE method, we added a log transformation on top of the flow to ensure that our posterior approximation matches the support of the parameters. For SNLE and the SNRE methods, we used adaptive HMC to perform the MCMC sampling of the parameters (Bingham et al. 2019), using a reparametrisation to put both α and λ on a log-scale.

We deviate slightly from usual practice in the SBI literature by performing 4 independent runs of each inference procedure, rather than performing a single run. Each run uses the same dataset but different initial random seeds, a process analogous to the standard MCMC practice of running multiple independent chains. The posterior samples are then combined at the end when computing estimates. This lets us check the convergence of each method using the potential scale reduction factor $\hat{R}$ commonly used with MCMC methods. If the $\hat{R}$ value is not close to 1 for each parameter, then each run has not converged to produce samples from the same distribution. Therefore at least one of the runs must be wrong, and the correctness of any posterior expectations computed using these samples will be suspect. $\hat{R}$ values close to 1 for all parameters do not mean that the method is producing samples from the posterior distribution, just that each run of the method converged to the same (but not necessarily correct) distribution.

The results are presented in Table 4.3. For each method, the $\hat{R}$ values are far from 1 when using only 10^3 simulations, indicating that this simulation budget is too low to correctly identify the posterior regardless of the method. When using 10^4 simulations, the $\hat{R}$ values become more reasonable, although for SNLE the values are still relatively high. With 10^5 simulations, $\hat{R}$ values for both SNPE and SNRE are less than 1.01, indicating good convergence. However, for SNLE, values actually get considerably worse with 10^5 simulations. Whilst counter-intuitive, this has been observed before with SNLE on other

Table 4.3: Estimates of the parameters α and λ for each SBI method and simulation budget.

			Mean	SD	HDPI-L	HDPI-U	$\hat{R}$
Method	Simulations	Param					
SNPE	10^3	α	2.254	1.017	1.006	3.293	1.51
		λ	0.878	0.794	0.041	1.331	1.59
	10^4	α	2.449	0.912	1.804	3.112	1.01
		λ	1.137	0.138	0.892	1.401	1.03
	10^5	α	2.431	0.322	1.863	3.055	1.00
		λ	1.138	0.127	0.895	1.371	1.00
SNLE	10^3	α	2.527	0.377	1.899	3.243	1.19
		λ	1.225	0.185	0.864	1.568	1.37
	10^4	α	2.519	0.381	1.831	3.227	1.08
		λ	1.090	0.149	0.836	1.385	1.15
	10^5	α	2.129	1.391	0.016	3.886	2.15
		λ	0.630	0.393	0.000	1.117	2.47
SNRE	10^3	α	2.512	0.398	1.795	3.253	1.12
		λ	1.186	0.141	0.929	1.453	1.06
	10^4	α	2.555	0.363	1.909	3.245	1.02
		λ	1.155	0.131	0.919	1.402	1.00
	10^5	α	2.459	0.336	1.854	3.084	1.00
		λ	1.141	0.133	0.901	1.396	1.00

problems (Lueckmann, Boelts, et al. 2021). In this case the problem happens because the flow used to model the likelihood is overfitting to atoms of density in the distribution of the summary statistics caused by zero-energy observations. These atoms are more likely to occur with samples from the prior, so using more simulations on the first round makes the overfitting more likely to occur.

Figure 4.9 visualises the estimates of each parameter for each run of each method when using a budget of 10^4 simulations. This provides a visual confirmation that SNLE produces less self-consistent estimates as indicated by the poor $\hat{R}$ values. SNPE and SNRE appear to be roughly consistent with each other, and it is obvious that SNRE is very self-consistent when estimating λ.

Figures 4.10, 4.11 and 4.12 show corners plots of the combined samples from

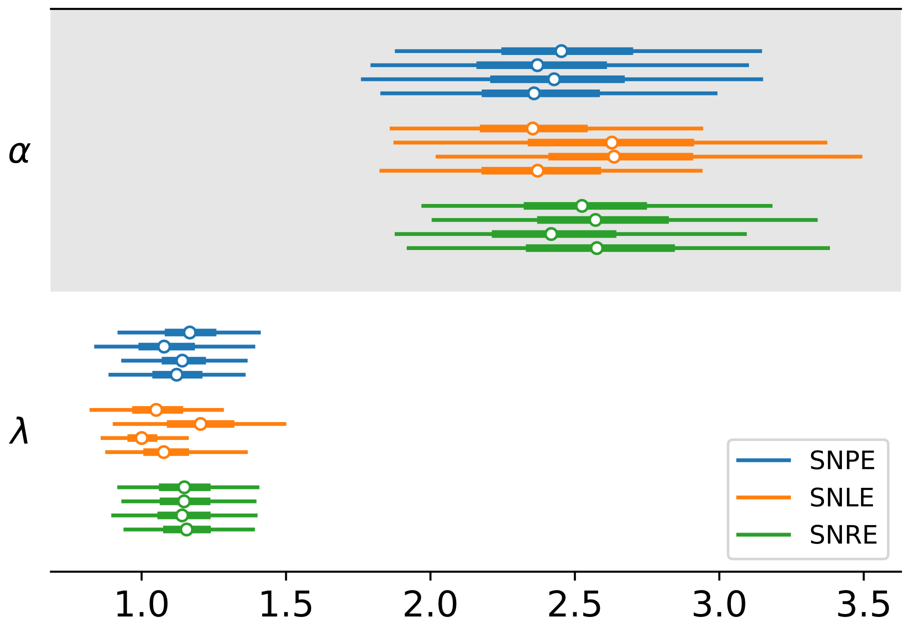


Figure 4.9: Forest plot showing the 94% HDPI, interquartile range and median for both parameters and each run of each method using 10^4 simulations.

all runs when using a budget of 10^4 simulations. Each posterior appears to be roughly the same shape, but there are small differences, and the plots are not as consistent as those for the different MCMC parametrisations.

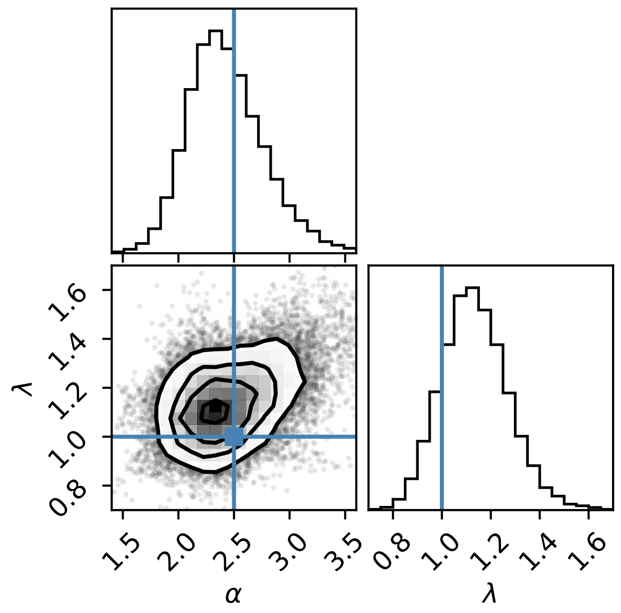


Figure 4.10: Corner plot with 1D marginals and a 2D joint plot of SNPE samples of α and λ. The blue lines indicate the ground-truth parameter values.

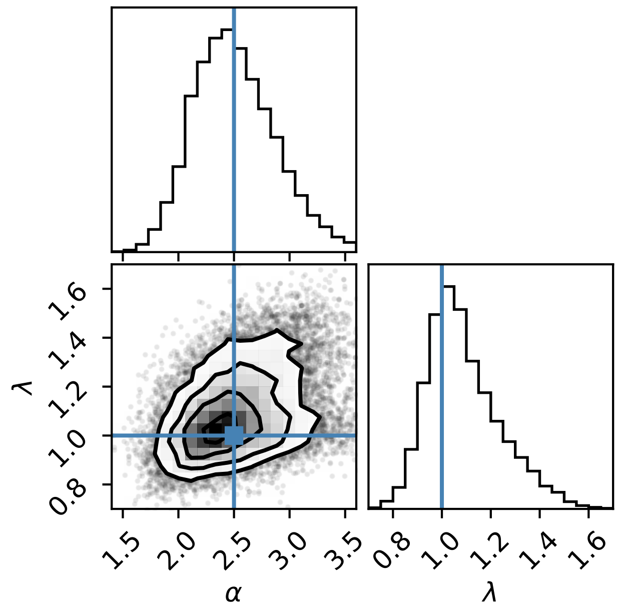


Figure 4.11: Corner plot of SNLE posterior samples.

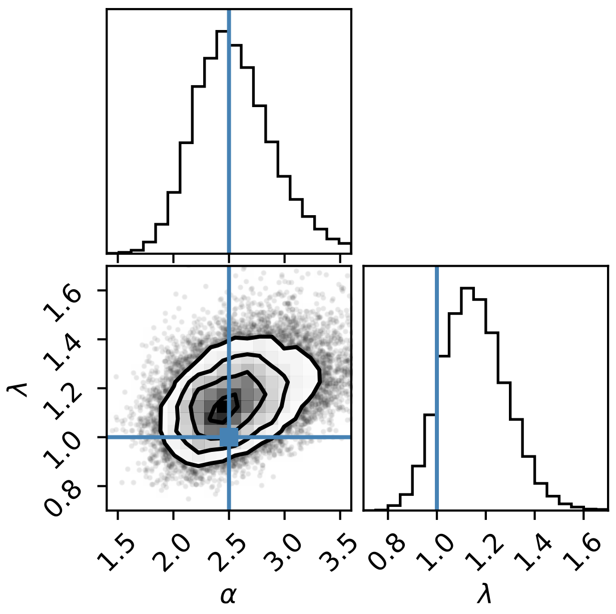


Figure 4.12: Corner plot of SNRE samples.

4.6.3.2 Simulation Based Calibration

As with the MCMC parametrisations, we also ran the SBC experiments as a check on the correctness of each SBI method. We drew 640 samples from the prior and simulated a dataset for each, before running each SBI method on the dataset once, drawing 31 samples from the posterior and computing the rank of the prior sample relative to the posterior samples. For computational reasons we restricted the SBI runs to a budget of 10^4 simulations. For SNLE and SNRE we thinned the samples from a single MCMC chain by a factor of 4 to produce the final posterior samples. As SNPE produces independent samples directly no thinning was applied.

Figures 4.13 and 4.14 show the histogram and ECDF plots respectively. Inspecting the histograms first, we can see that SNPE and SNRE look reasonable for both parameters, as none of the bin heights appear to be too extreme. The histogram for SNLE shows that the minimum and maximum ranks for α are overrepresented, as well as the maximum rank for λ. This suggests that the posterior that SNLE is fitting is too narrow for α, and biased for λ. The normalised ECDF plot for SNLE in Figure 4.14 shows this at a more granular level without rebinning the ranks. Both α and λ are outside the expected 99% interval at the extreme ranks. This shows that the SNLE's posterior estimates for λ are not just biased but also slightly too narrow.

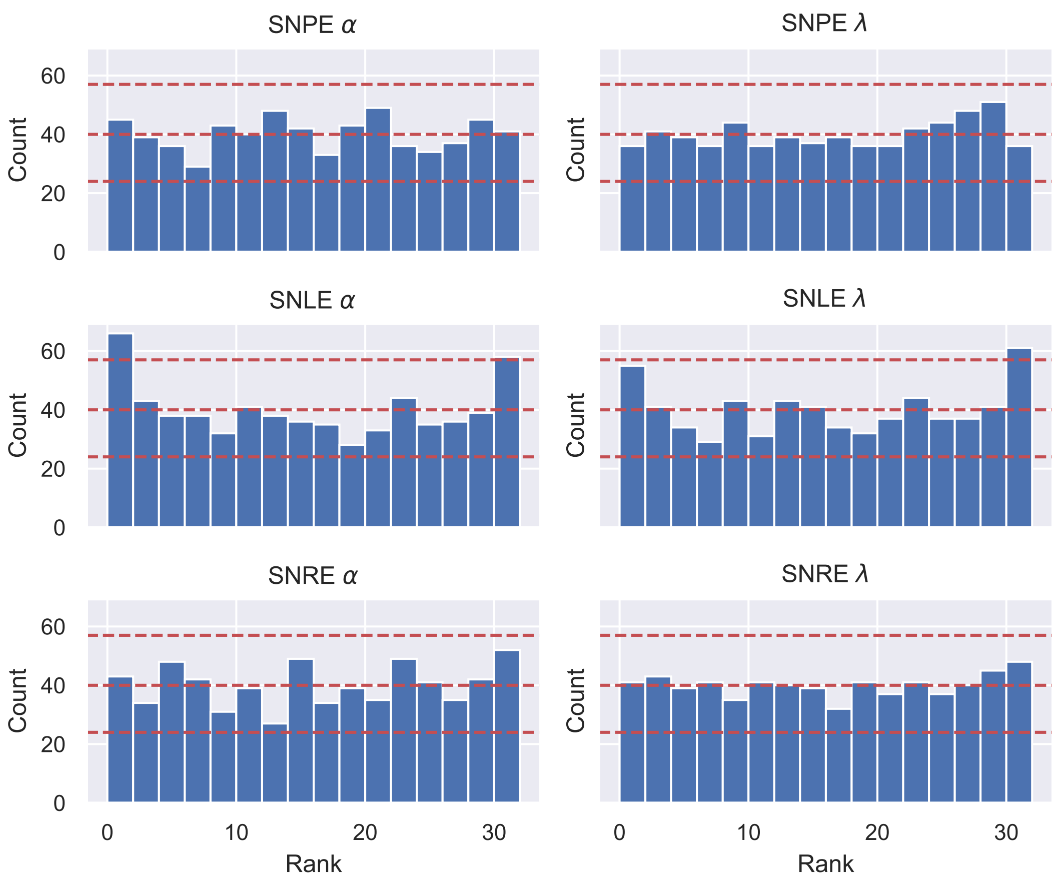


Figure 4.13: Histograms of the prior sample ranks with respect to the posterior samples for each SBI method and parameter. Dashed lines indicate the median and 99% interval assuming a uniform distribution of ranks. The extreme ranks are overrepresented for both parameters with SNLE, indicating a too-narrow posterior.

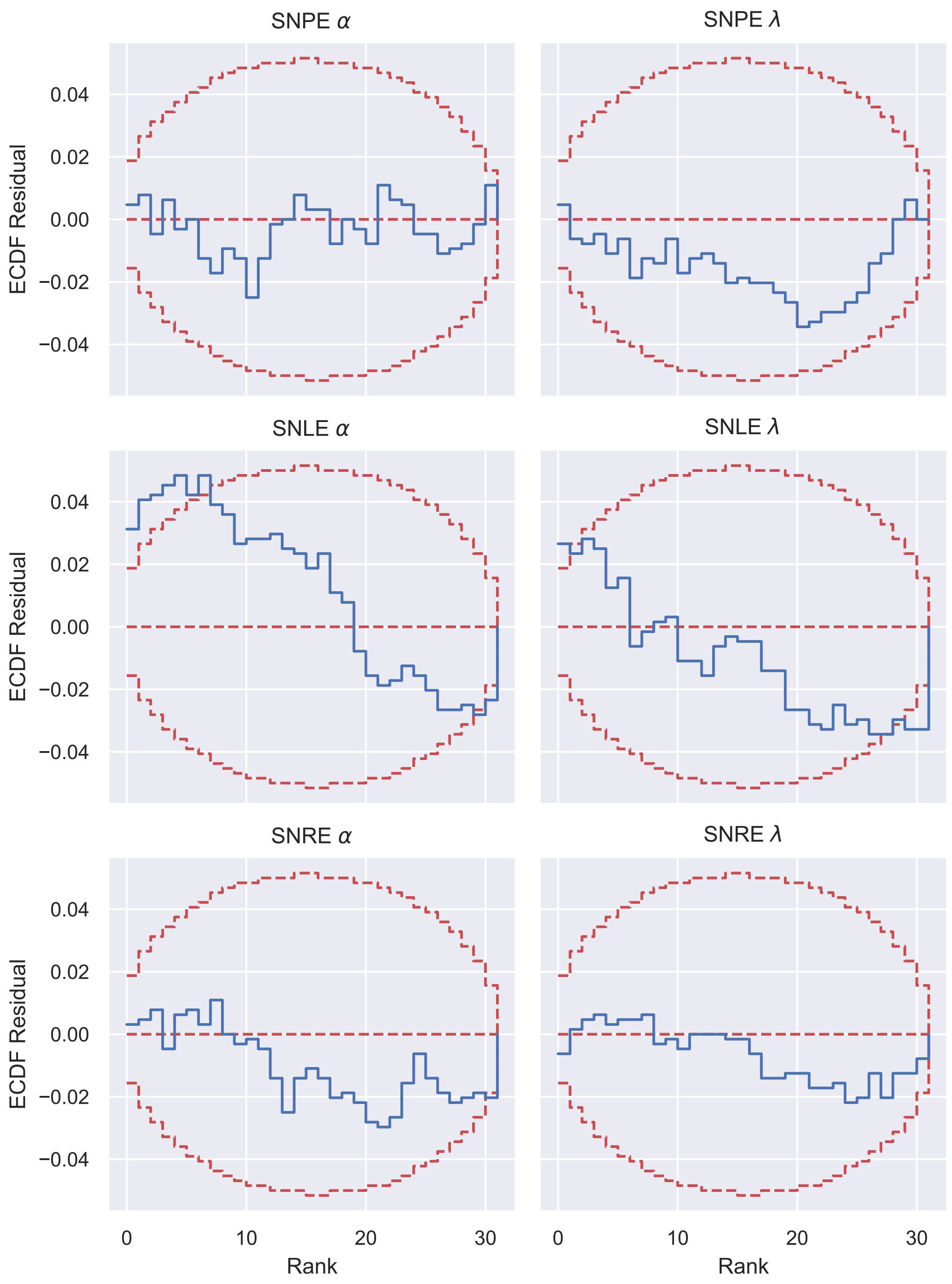


Figure 4.14: Normalised Empirical CDF plot of prior sample ranks for each SBI method and parameter. Plots follow the same format as the bottom row of Figure 4.8. The ECDF for SNLE is outside the 99% interval for both parameters at both rank extremes, indicating a miscalibrated posterior.

4.7 Computational Efficiency

It is difficult to compare the computational cost of the MCMC and SBI approaches directly, as an evaluation of the likelihood is not the same as running a simulation, and both approaches are implemented in different numerical frameworks. In general, the computational cost of the MCMC approach is dominated by the need to make many evaluations of the likelihood and gradients, whilst the SBI methods are dominated by the cost of training the neural network and sampling from the posterior approximation on each round, as the simulation is relatively quick to run.

We can do a theoretical analysis of the computational complexity of each part. The time complexity of standard HMC scales as $\mathcal{O}(D^{5/4})$ with dimensionality D (Neal 2011), and the dimensionality D scales proportional to $TN_{\max}^2$, giving a total complexity of $\mathcal{O}([TN_{\max}^2]^{5/4})$. For the SBI methods, the actual fitting process complexity will be approximately the same assuming the summary statistic size, simulation budget, and the number of parameters in the neural networks remain constant. The only variable cost is the run time of the simulator, which has time complexity $\mathcal{O}(TN_{\max})$. As already noted, for this setup the simulation cost is small relatively to the time taken to train the neural networks and draw samples from them, so we would expect total training time to be approximately constant with respect to T.

To verify this analysis, we reran the parameter recovery experiments for different values of T. We used the rescaled parametrisation for the MCMC approach, and the SNPE and SNRE methods for the SBI approach, as the previous experimental results indicated these were the most reliable. Table 4.4 reports the mean and standard deviation of each experiment run time across 5 repetitions. It is difficult to make meaningful comparisons in terms of absolute differences in run times, but Figure 4.15 shows a plot of the times for each method allowing us to see the relative scaling. This confirms our theoretical analysis that run time will be approximately constant with respect to T for the SBI approaches, and will scale proportional to T for the MCMC approach. For some pileup analyses T can be on the order of 10^4 (Davis 2001), so the SBI approaches could have a substantial advantage in terms of run time.

Table 4.4: Mean and standard deviation of experiment run times in minutes for each method and different numbers of datapoints.

T	20	50	100	200
Method				
MCMC	23.44 ± 5.03	48.07 ± 9.20	175.31 ± 4.41	402.34 ± 65.53
SNPE	14.17 ± 2.24	14.03 ± 1.02	14.43 ± 1.03	15.07 ± 0.47
SNRE	19.06 ± 2.66	20.50 ± 3.30	19.52 ± 1.32	18.71 ± 0.54

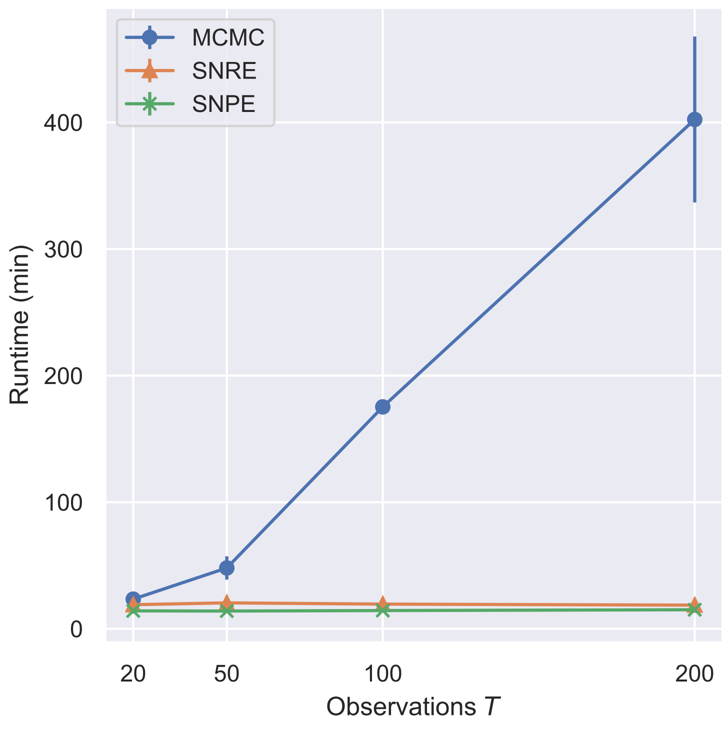


Figure 4.15: Plot of run times against number of observations for different methods. Errorbars show ± 1 standard deviation.

4.8 Censored Data

In a more realistic setting, a CCD sensor is generally only well-calibrated over a certain range of photon energies. This is particularly important for power law distributed energies, where at smaller values of α a substantial number of observed energies will be beyond the right cut-off of the calibrated range due to the heavy-tailed nature of the distribution. For the Chandra ACIS sensor, Davis (2001) considers energy observations beyond 10 keV to be poorly calibrated. Here we consider modifications to allow both the MCMC and SBI approaches to work with censored data.

For modelling purposes, we will denote variables associated with censored observations using the superscript $*$. We will assume that we know the number of censored observations, denoting it as T^*, and the cut-off value E_c beyond which observations are censored.

4.8.1 MCMC

We can theoretically extend the MCMC approach to censored data by treating the censored observations $E^*_{t^*}$ as latent variables to be inferred. Modifying the posterior presented in Equation 4.6 to include this latent variable, and again dropping the indexing over T and T^* to simplify the notation gives us a posterior

$$
\begin{aligned}
p(\alpha, \lambda, N, \{e_i\}_{i=1}^{N}, N^*, \{e_j^*\}_{j=1}^{N^*}, E^* \mid E) \propto \\
\mathcal{N}(E \mid \sum_{i=1}^{N} e_i, \sigma) \prod_{i=1}^{N} p(e_i \mid \alpha, e_{\min}) p(N \mid \lambda) \\
\times \mathcal{N}(E^* \mid \sum_{j=1}^{N*} e_j, \sigma) \prod_{j=1}^{N*} p(e_j \mid \alpha, e_{\min}) p(N^* \mid \lambda) \\
\times p(\alpha) p(\lambda). \quad (4.20)
\end{aligned}
$$

We can apply the same marginalisation process as before to make this posterior useable with HMC, with the addition of E^* as another latent variable. An appropriate reparametrisation enables us to constrain $E^* > E_c$.

In practice when implemented this approach does not actually work. The censored energy E^* is highly correlated with the latent sum of individual energies $\sum_{j=1}^{N*} e_j$. Worse still, when applying the marginalisation approach the marginalised sets of individual energies $\{e_m^{M^*}\}_{m=1}^{M^*}$ are now highly correlated through E^* between the sets corresponding to different values of M^* as well as within sets. These strong correlations result in a posterior with regions of extremely high curvature, which the numerical integrator used to simulate the Hamiltonian trajectory within HMC struggles to deal with, resulting in a large number of divergences and poorly mixing chains.

We can improve on this approach by noting that we do not actually care about the value of E^*. Instead, we can condition on the posterior on $E^* > 10$. Denoting the cumulative distribution function (CDF)

$$F_x(X \mid \mu, \sigma) = p(x \leq X), \quad x \sim \mathcal{N}(\mu, \sigma), \tag{4.21}$$

and the complementary cumulative distribution function (CCDF) as

$$\bar{F}_x(X \mid \mu, \sigma) = 1 - F_x(X \mid \mu, \sigma), \tag{4.22}$$

the posterior becomes

$$\begin{aligned} p(\alpha, \lambda, N, \{e_i\}_{i=1}^{N}, N^*, \{e_j^*\}_{j=1}^{N^*} \mid E, E^* > E_c) \propto \\ \mathcal{N}(E \mid \sum_{i=1}^{N} e_i, \sigma) \prod_{i=1}^{N} p(e_i \mid \alpha, e_{\min}) p(N \mid \lambda) \\ \times \bar{F}_{E^*}(E_c \mid \sum_{j=1}^{N*} e_j, \sigma) \prod_{j=1}^{N*} p(e_j \mid \alpha, e_{\min}) p(N^* \mid \lambda) \\ \times p(\alpha) p(\lambda). \end{aligned} \tag{4.23}$$

We can apply the same marginalisation approach as before to eliminate N and N^*.

To validate the approach, we repeated the parameter recovery experiment from Section 4.5.4.1. We use the same parameters as before, except for the value of α which we change to 0.38 to match the power law index Davis 2001 inferred for the spectrum of the Quasar S5 0836+7104 (Fang et al. 2001). We also changed the prior $p(\alpha)$ to a $\text{Gamma}(3, 3)$ distribution to allow the value of α to be closer to zero. We set the cut-off for censoring simulated observations at 10 keV. In order to have the chains mix well, we found we needed to use the rescaled parametrisation for the fully observed measurements and the standard parametrisation for the censored observations.

Table 4.5 summarises the parameter estimates along with diagnostic statistics. Figure 4.16 shows a corner plots of the samples of α and λ. We can see that

Table 4.5: Estimates of the parameters α and λ when using MCMC with censoring.

	Mean	SD	3% HDI	97% HDI	N_{ESS}	$\hat{R}$
α	0.351	0.049	0.260	0.442	271.0	1.01
λ	1.145	0.143	0.876	1.411	1655.0	1.00

this approach allows us to correctly recover the parameters. We find that the $\hat{R}$ values are reasonably close to one for all parameters, and our chains do not have any divergent samples. However, the BFMI values are below 0.3 for all chains, indicating that they are struggling to explore the tails of the distribution. This is a known problem with HMC for heavy-tailed distributions (Betancourt 2018), which the Pareto distribution is for small values of α.

The inability to reach the tails of the posterior distribution is exacerbated by the need to select a very small step size for the numerical integrator, required in order to adequately explore the sharp transition in the posterior induced by the CCDF when the sums of photon energies are around the cut-off value of E_c. Consequently, the Hamiltonian trajectory must be simulated for many steps in order to be able to reach the tails, and in practice we found it would always use the maximum number of steps permitted. Each sample from the censored posterior used $2^{15} - 1$ steps, whilst samples from the non-censored posterior from the experiments in Section 4.5.4.1 required no more than $2^{11} - 1$ steps even when the maximum was set to $2^{15} - 1$, resulting in a runtime approximately 16 times longer for the censored experiments.

4.8.2 SBI

Modelling the censoring process with the SBI methods is relatively straightforward. After running the simulator, we remove any simulated observations that exceed the cut-off before computing the quantiles, then report the fraction of censored observations as part of the summary statistics. Again we repeated the parameter recovery experiments with the same setup as for MCMC, but for this round we only used the SNPE and SNRE methods as the previous results indicate SNLE is less reliable on this type of problem. For both methods we used 10^5 simulations.

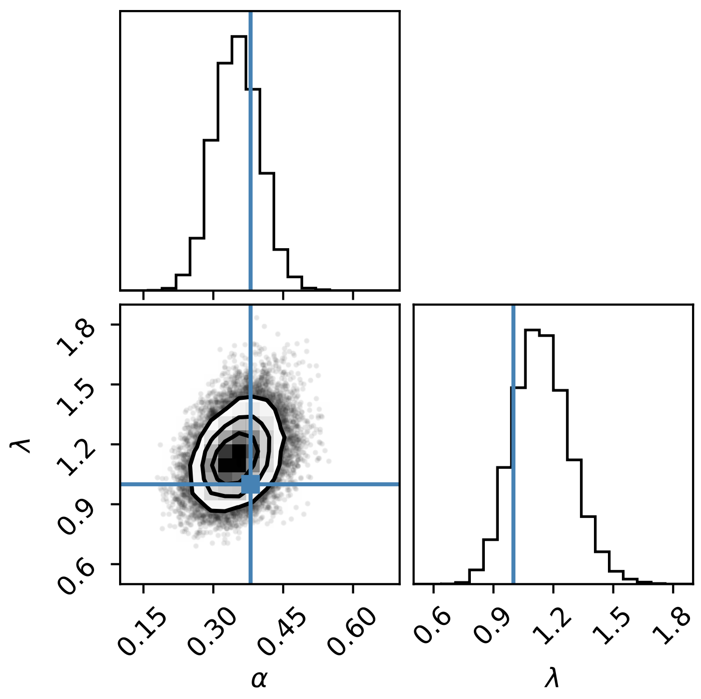


Figure 4.16: Corner plot of MCMC posterior samples accounting for censoring

Table 4.6 summarises the parameter estimates for both methods along with diagnostic statistics. Figures 4.17 and 4.18 show corner plots of the posterior samples from each method. We can see that both methods have managed to recover the ground truth parameters, although the credible intervals for the estimates produced by SNPE are wider, especially for λ. The $\hat{R}$ values for SNPE are away from 1, indicating that each run has not converged to the same estimates.

4.8.3 Comparison

Figure 4.19 visualises the estimates of each parameter for the MCMC approach and both SBI methods across each run when using censored data. We can see that the estimates produced by SNPE are neither self-consistent nor comparable to the other methods. By contrast, the estimates for SNRE and MCMC are consistent with each other as well as with themselves. This suggests that SNRE and MCMC produce more reliable estimates than SNPE. Comparing SNRE and MCMC, the SNRE approach has the advantage of being straightforward to implement, as well as having no appreciable difference in run time. The MCMC

Method	Parameter	Mean	SD	3% HDI	97% HDI	$\hat{R}$
SNPE	α	0.358	0.075	0.223	0.501	1.05
	λ	1.023	0.256	0.561	1.485	1.06
SNRE	α	0.346	0.050	0.255	0.439	1.01
	λ	1.134	0.143	0.876	1.405	1.00

Table 4.6: Estimates of the parameters α and λ when using SBI methods on censored data.

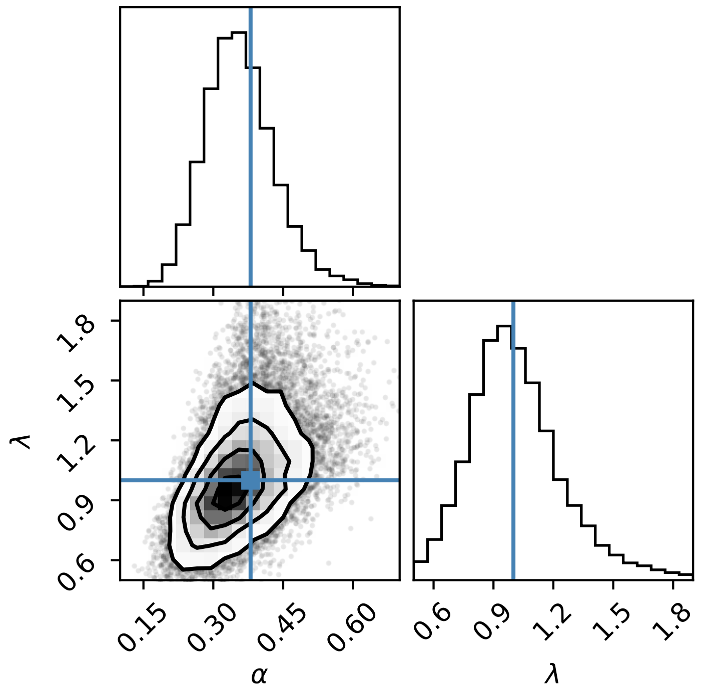


Figure 4.17: Corner plot of SNPE posterior samples using censored data.

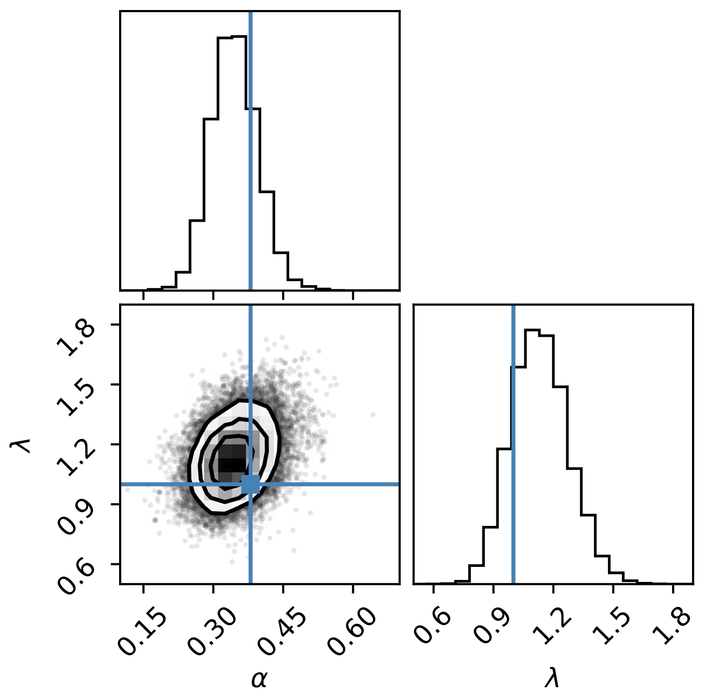


Figure 4.18: Corner plot of SNRE posterior samples using censored data.

approach required us to carefully modify the posterior over all variables to account for the censoring process, and took approximately 16 times longer to run compared to the previous experiments as a consequence of the change in posterior geometry at smaller values of α.

4.9 Line Spectrum

To show that our SBI approaches can handle more complicated and realistic models, we consider the case where our energy spectrum has a line emission component mixed in with the power law spectrum. We model the line spectrum as a narrow Gaussian, and put appropriate priors on the mean m and scale s.

$$m \sim \mathcal{N}(4, 1) \tag{4.24}$$

$$s \sim \text{HalfNormal}(0, 0.1) \tag{4.25}$$

We then model the individual photon energy distribution as a mixture of the line spectrum component and the Pareto component, with a fraction of the

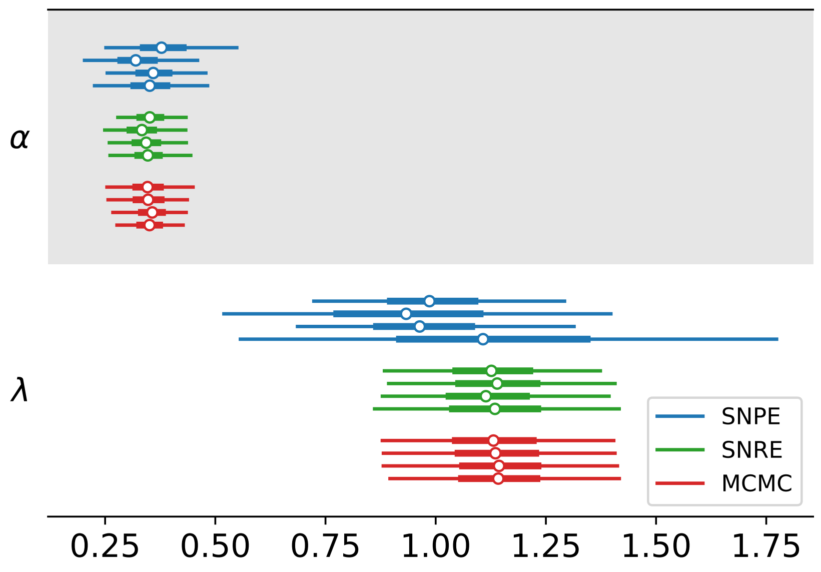


Figure 4.19: Forest plot of posterior estimates with censored data showing the 94% HDPI, interquartile range and median for each method across each run. The SNPE results are clearly less consistent.

photons drawn from the line component and the remainder from the power-law distribution. We put an appropriate prior on the mixing fraction w,

$$w \sim \text{Beta}(5, 20). \tag{4.26}$$

The individual photon energies are then drawn from the mixture distribution,

$$l_{it} \sim \text{Bernoulli}(w), \tag{4.27}$$

$$e_{it} \sim \begin{cases} \text{Pareto}(\alpha, e_{\text{min}}) & l_{it} = 0 \\ \mathcal{N}(m, s) & l_{it} = 1 \end{cases}. \tag{4.28}$$

The remainder of the generative model including the pileup and censoring process is the same as before.

We repeated the parameter recovery experiments using this model. Table 4.7 shows the settings of the parameters we used to generate the synthetic data. Figure 4.20 shows the distribution of photon energies with and without pileup when using these parameter settings. The effect of pileup is clearly visible, resulting in a shifted power law and a clearly visible phantom peak at 6 keV.

Table 4.7: Parameter values used for the parameter recovery experiments with the line spectrum model.

Parameter	Value
α	0.38
λ	2.0 photon s^{-1}
m	3.0 keV
s	0.1 keV
w	0.2
σ	0.01 keV
$e_{\min}$	0.2 keV

Censoring was done with the same 10 keV cut-off as before. We increased the dataset size to $T = 1000$ to allow for better identification of the parameters. At this dataset size in conjunction with the censoring process, using the MCMC approach would result in impractical runtimes, so we restrict our experiments to the SNPE and SNRE methods. We used a budget of 10^5 simulations for each run, using 4 runs for each method with different random seeds.

Table 4.8 shows the parameter estimates for both methods. Figures 4.21 and 4.22 show corner plots of the posterior samples along with ground truths. As with the censored data experiments, both methods seem to have recovered the parameters. However, the credible intervals are generally much wider for SNPE compared to SNRE. As the $\hat{R}$ values are far from 1.00 for the SNPE method we believe that this is the method with the incorrect estimates. Looking at the samples of s in Figure 4.21 we can also see that the posterior produced with SNPE has not substantially shrank from the prior distribution given in Equation 4.25.

Figure 4.23 shows visual comparisons of the parameter estimates across each run with a forest plot. We can see that the SNPE estimates are much less self-consistent, and often one of the runs has markedly different estimates from the rest. Together with the examples from the previous experiments, these results suggest that SNRE is much better suited to these sorts of problems than the other SBI methods.

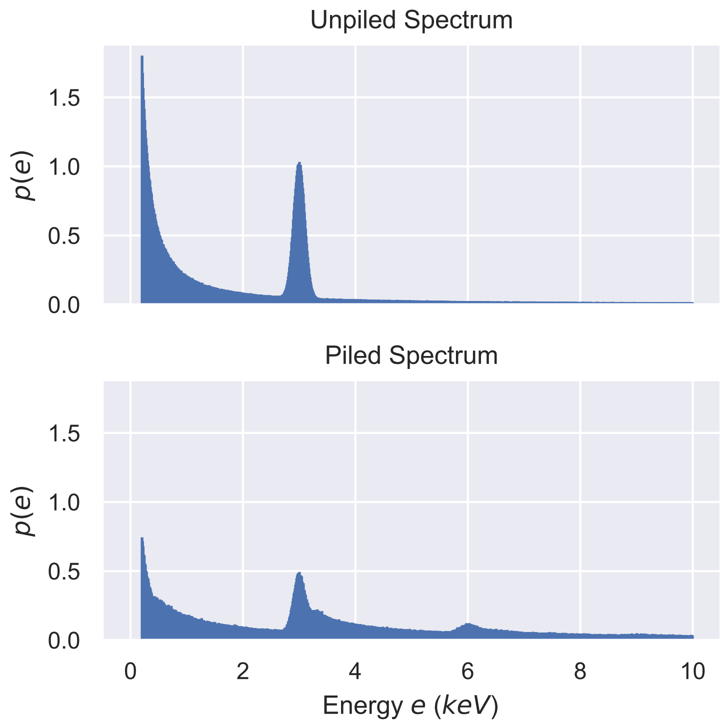


Figure 4.20: Histograms showing the distribution of photon energies with and without the presence of pileup when using the parameters in Table 4.7.

Table 4.8: Estimates of the parameters when using SBI with the line spectrum model.

Method	Parameter	Mean	SD	3% HDI	97% HDI	$\hat{R}$
SNPE	α	0.385	0.052	0.287	0.480	1.14
	λ	1.967	0.247	1.559	2.460	1.30
	m	3.363	0.358	2.642	4.043	1.12
	s	0.071	0.060	0.000	0.179	1.02
	w	0.199	0.066	0.083	0.321	1.13
SNRE	α	0.365	0.019	0.327	0.400	1.01
	λ	1.937	0.075	1.795	2.079	1.01
	m	2.984	0.031	2.928	3.043	1.02
	s	0.112	0.030	0.059	0.173	1.02
	w	0.200	0.021	0.161	0.239	1.01

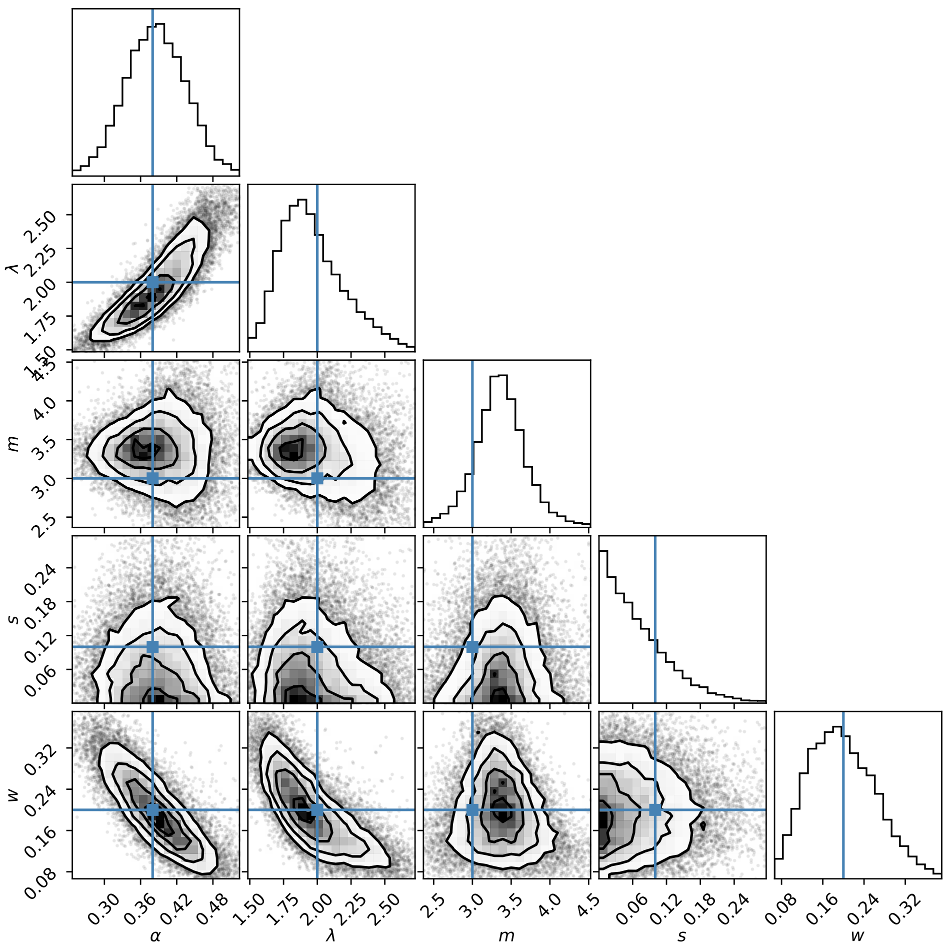


Figure 4.21: Corner plot of SNPE posterior samples with the line spectrum mixture model.

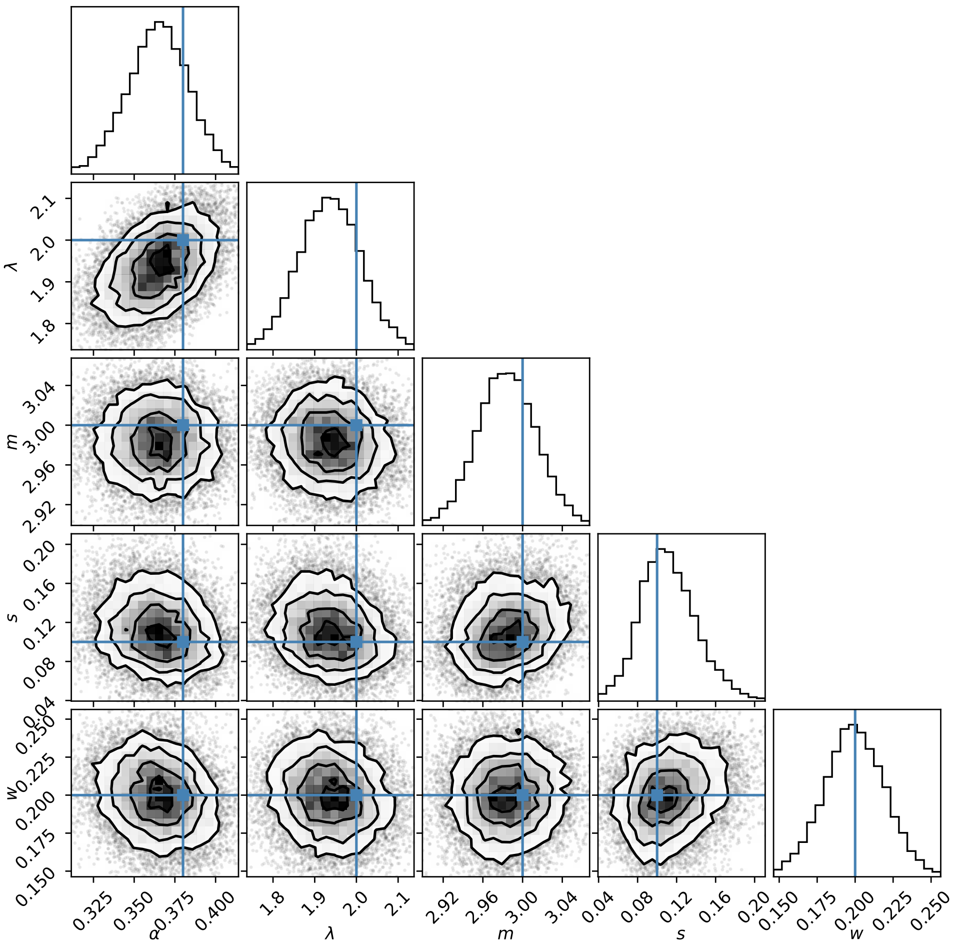


Figure 4.22: Corner plot of SNRE posterior samples with the line spectrum mixture model. Note the different scales from Figure 4.21.

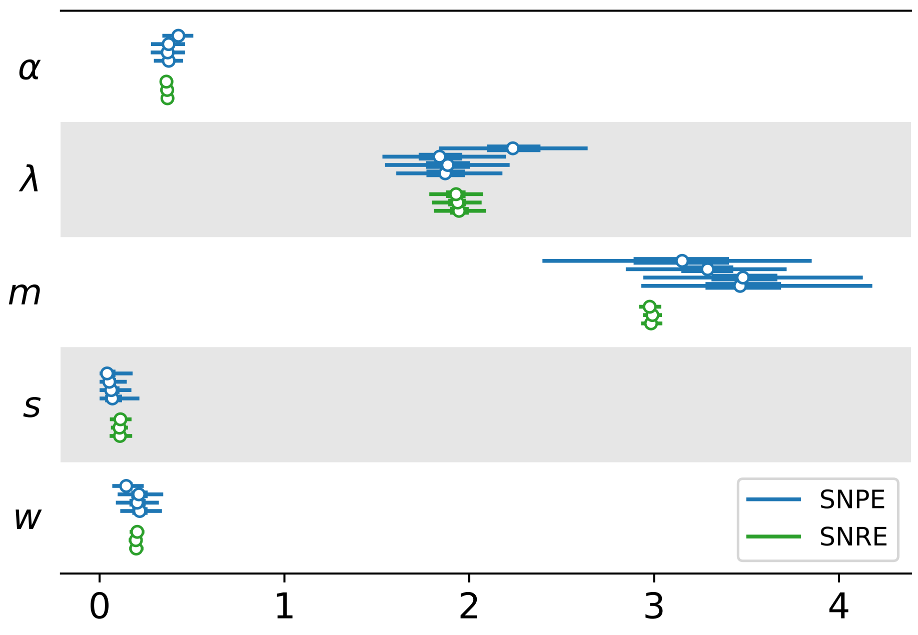


Figure 4.23: Forest plot of posterior estimates for the parameters of the model with a line spectrum component. The lines show the 94% HDPI, interquartile range and median. Again the SNPE method is clearly less consistent.

4.10 Discussion

4.10.1 Accuracy

Considering the first set of experiments with the simplified model, Tables 4.2 and 4.3 differ slightly in their estimates of α and λ, yet the SBC results indicated that the MCMC, SNPE and SNRE methods were well-calibrated. How can we resolve this apparent contradiction? Figure 4.24 shows a combined Forest plot for each run of each method. We can see that the MCMC estimates for each chain are slightly more consistent with each other compared to the SBI methods. One possible explanation is that the SNPE and SNRE posterior estimates in each SBC run are skewed compared to the correct posterior, but not systematically in one direction over multiple runs. We would expect this given that each SBC run sees a different finite set of simulated data over successive rounds. The SBC method is slightly flawed as it cannot detect this failure using the rank histograms or ECDF plots. Samples which are skewed in on direction and result in an over-representation of one of the extreme ranks will be cancelled out by samples skewed in the other direction which results in an under-representation of that rank. This contradiction could be resolved at the cost of more computation time by rerunning the SBC simulations for each prior sample with a simulation budget of 10^5 samples instead of 10^4, as the $\hat{R}$ values from Table 4.3 indicated the more expensive runs were more self-consistent.

The SNLE method did not do well on the first set of experiments, as a result of overfitting to atoms in the summary statistics. In certain situations having a fast emulator of the simulator would be useful beyond the inference process, so we should try to avoid writing it off entirely. A possible solution to this issue is to modify the density estimator, using a hurdle-type model with the flow restricted to modelling non-zero values (Cragg 1971).

A more general question is whether it actually matters that the posterior estimates are completely correct? The answer will obviously depend on the application we are producing estimates for. For an application like estimating the shape parameter of a power-spectrum for a star, it will not matter too much if the credible interval is slightly off. At the other extreme, if we were estimating the basic reproduction number R_0 for an SIR model being used to inform public health policy, it would be a good to be confident that the behaviour of the

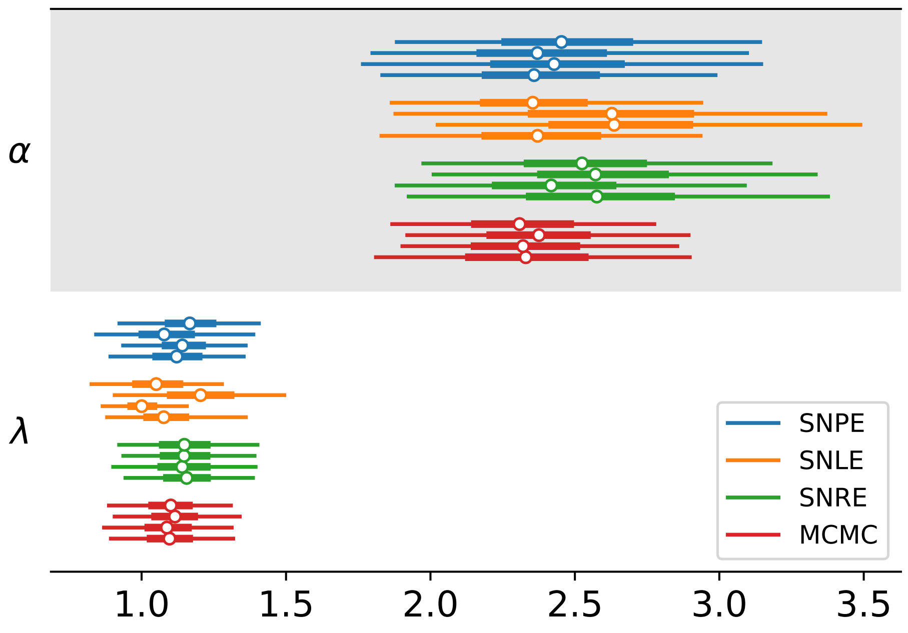


Figure 4.24: Forest plot comparing the posterior estimates of α and λ for each run/chain of each method using the first model. The lines show the 94% HDPI, interquartile range and median. The MCMC chains are generally more self-consistent that the SBI runs.

posterior tails was correct. It has been observed that many SBI methods have a tendency to produce overconfident posterior estimates for certain problems, which is arguably the worst possible failure mode if we are interested in a Popperian philosophy of empirical falsification (Hermans, Delaunoy, et al. 2021). Having correct posterior estimates can also help with resolving between limitations caused by a misspecified model, and the inability of the inference method to correctly estimate the model parameters.

4.10.2 Flexibility

The models presented here are extremely simplified versions of CCD and source behaviour for demonstration purposes. A natural question therefore is how well can each approach handle more realistic models? Typical CCDs have multiple pixels arranged in either a 1D strip or 2D grid. Photons from a point source will be distributed across multiple adjacent pixels, with the degree of spread characterised by the point spread function (PSF). We may also have a more complicated underlying model, where the source has transient or periodic behaviour.

For the MCMC approach, every change to the model requires a corresponding change to compute the likelihood, with careful consideration needed as to how each change will affect the inference process. In general, as long as the total observed energies for each pixel and timestep remain independent given the other parameters, then the outlined marginalisation approach will still be feasible. With multiple pixels, an effective rate can be calculated for each pixel as a product of the source rate λ and the total mass of the PSF covering that pixel. Similarly, a periodic signal where the source rate is only a function of time t does not break the independence between observed total energies given the other parameters, and so the marginalisation approach holds. If the independence is broken, for example if the photon arrivals follow an autoregressive process, then the marginalisation will no longer be a simple 1D summation, and may be computationally infeasible.

For the SBI approaches, as long as we can simulate a model we can generally run the inference process without having to alter it as we treat the simulator as a black box. This makes the SBI approaches more flexible in general, but different models will likely require different summary statistics. The modification of the original model to account for censoring is an example of this. As another example, if we change a model to include periodic behaviour, then our summary statistics will need to include time information.

However, just because the method runs there is no guarantee that the resultant posterior is correct, as our parameter recovery experiments show, and SBC runs will be required for each modification to simulator and summary statistics. The issues with the SNLE method in the first set of experiments also point to the limitations of treating SBI as a black box inference process. We proposed a possible solution in Section 4.10.1, but this will require us to make a bespoke density estimator for this simulator, limiting the flexibility.

The SBI approaches also introduce a large number of free hyperparameters that the user needs to select. In general we opted to use the default settings provided by the *sbi* package (Tejero-Cantero et al. 2020). It is possible that better hyperparameters could have resolved some of the issues with SNPE producing broad estimates in the final set of experiments. If we were drawing all proposals from the prior, we could do hyperparameter optimisation through cross-validation by evaluating the chosen SNPE loss on a held-out dataset. With the sequential variant, it is not immediately obvious how we should do this, as fitting the conditional density estimator is done over several rounds, with previous fits affecting the generation of data used to train subsequent estimators.

4.10.3 Computational Performance

It is hard to make comparisons between the SBI and MCMC methods in terms of absolute running time due to the differences in approach and numerical frameworks. However, our timing experiments have shown that the SBI approaches have favourable scaling properties with respect to the dataset size, and for larger dataset sizes they have a substantial advantage.

Previous work has shown that SBI methods can outperform MCMC methods in terms of run time by several orders of magnitude (Green and Gair 2021; Hahn and Melchior 2022). These examples involved posteriors with computationally expensive likelihoods but no nuisance parameters. They also used the amortized SBI variants trained solely on simulations using parameters drawn from the prior. Consequently they make the assumption of ignoring training time when comparing against MCMC methods, which is reasonable considering that they were intended to be used repeatedly with different datasets. By contrast, our results show that SBI can also be competitive with MCMC in terms of run time even in the non-amortized case when the full posterior involves a large number of nuisance variables, including training time as well as sampling from the final posterior.

4.10.4 Diagnostics

One of the most useful aspects of the MCMC approach are the diagnostics we get when using adaptive HMC. Whilst it took several rounds of running the parameter recovery experiment and debugging to eliminate all of the diagnostic errors, when we had finally done so we were confident that the inference process was correct. Subsequently we only required one round of SBC experiments to show that it worked.

By contrast, the standard approaches in the literature to SBI have no useful diagnostics beyond the benchmarks used to compare performance, which often require access to a ground-truth posterior (Lueckmann, Boelts, et al. 2021). Consequently, even though the parameter recovery experiments were producing reasonable estimates, we had to rely solely on the SBC experiments in order to be confident that each method was correct. This required several rounds of modifying summary statistics and prior choices followed by SBC experiments, which could potentially erode any computational advantages of the SBI methods. As mentioned in Section 4.6.3.1, we adapted the practice of running multiple independent MCMC chains by running multiple independent SBI runs and computing the potential scale reduction factor $\hat{R}$ across all runs. Whilst not a sufficient condition for correctness, a reasonable $\hat{R}$ value is a necessary one, and the poor values observed for SNLE did correspond to that method failing the SBC checks whilst being much quicker to run.

4.10.5 Conclusion

Our experiments have indicated that the SBI approaches should generally be considered the preferable approach for the pileup problems considered here, especially SNRE. The credible intervals of the posterior parameter estimates of have comparable widths to those produced with the MCMC approaches, and appear to be correct. Unlike MCMC, it is much easier to implement without careful derivation of target density functions suitable for use with HMC, and has the advantage of the run time not scaling significantly with dataset size.

However, it is difficult to be certain that our posterior estimates are correct without checking the calibration using SBC, with the requirement for a large number of repetitions undermining some of the computational advantages of SBI. Better diagnostics for SBI methods could help mitigate this, and the adaption of computing the $\hat{R}$ values across multiple runs is a step towards this.

Under certain circumstances the MCMC-based approaches could still be useful. Careful consideration of the problem allows the likelihood to be made tractable for use with HMC via marginalisation and appropriate parametrisations, and at smaller dataset sizes the run times are comparable. Certain posteriors can be more amenable to HMC than others, as indicated by the relative performances of HMC with light-tailed uncensored data and heavy-tailed censored data. Whilst difficult to implement, the availability of informative diagnostics meant we could be confident in the resultant posterior estimates without having to resort to expensive repeated runs of the method to perform the SBC checks.

Chapter 5

Scalable Extreme Deconvolution

5.1 Introduction

Extreme deconvolution is a method that fits Gaussian mixture models (GMMs) to noisy data where we know the covariance of the Gaussian noise added to each sample (Bovy, Hogg, and Roweis 2011). The method was originally developed to perform probabilistic density estimation on the dataset of stellar velocities produced by the Hipparcos satellite (Perryman et al. 1997). The Hipparcos catalogue consists of astrometric solutions (positions and velocities on the sky) and photometry (light intensity) for 118,218 stars, with associated noise covariances provided for each entry. The method has subsequently been widely used in variety of astronomical applications, including the analysis of quasars (White et al. 2012) and the estimation of the stellar halo mass of the Milky Way (Deason, Belokurov, and Sanders 2019). It has also been used in fields beyond astronomy including genetics (Urbut et al. 2019; Griesemer et al. 2021).

The successor to the Hipparcos mission, Gaia, aims to produce an even larger catalogue, with entries for an estimated 1 billion astronomical objects (The Gaia Collaboration 2016). Previous work using extreme deconvolution on the Gaia catalogue worked with a subset of the data and restricted the number of mixture components, but the intention is to fit models with the full dataset (Anderson et al. 2018). The existing extreme deconvolution method makes a full pass over the dataset before it can update parameters, and the reference implementation requires all the data to fit in memory. To fit such large datasets in reasonable time, we would normally use stochastic or online methods, with updates based on minibatches of data to make the methods practical on GPUs (Bottou, Curtis, and Nocedal 2018).

In this chapter, we develop two minibatch methods that extend the original extreme deconvolution method and enable the use of these stochastic methods, allowing us to fit models to larger datasets more easily. The first is based on an online variation of the expectation-maximisation (EM) algorithm and the second makes use of a gradient optimizer. Our implementations can run on GPUs, and provide comparable density estimates to the existing method, whilst being much faster to train.

5.2 Background

The aim of extreme deconvolution is to perform density estimation on a noisy d-dimensional observed dataset $\{\mathbf{x}_i\}_{i=0}^N$, where $\mathbf{x}_i$ was generated by adding zero-mean Gaussian noise ϵ_i with known per-datapoint covariance S_i to a known projection R_i of a latent noise-free value $\mathbf{z}_i$,

$$\mathbf{x}_i = R_i \mathbf{z}_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(\mathbf{0}, S_i). \tag{5.1}$$

We are interested in the distribution of noise-free values rather than noisy values, so we wish to fit a density estimator to $\mathbf{z}$, $p(\mathbf{z} \mid \theta)$ with parameters θ. This density estimator can be fitted even if we do not have access to the noise-free values by finding the parameters that maximise the marginal log-likelihood,

$$\mathcal{L}(\theta) = \sum_i^N \log \int \mathcal{N}(\mathbf{x}_i \mid R_i \mathbf{z}_i, S_i) p(\mathbf{z}_i \mid \theta) \, \mathrm{d}\mathbf{z}_i. \tag{5.2}$$

We assume that $\mathbf{z}_i$ can be modelled by a mixture of Gaussians with K components,

$$p(\mathbf{z}_i \mid \theta) = \sum_j^K \alpha_j \mathcal{N}(\mathbf{z}_i \mid \mathbf{m}_j, V_j), \quad \theta = \{\alpha_j, \mathbf{m}_j, V_j\}_{j=1}^K, \tag{5.3}$$

parametrised by mixture weight α_j, mean $\mathbf{m}_j$ and covariance V_j collectively indicated by θ. As the noise model is Gaussian and the model of the underlying density is a mixture of Gaussians, the probability of $\mathbf{x}_i$ is also a Gaussian mixture. The marginal log-likelihood of the model therefore becomes

$$\mathcal{L}(\theta) = \sum_i^N \log \sum_j^K \alpha_j \mathcal{N}(\mathbf{x}_i \mid R_i \mathbf{m}_j, T_{ij}), \quad T_{ij} = R_i V_j R_i^T + S_i. \tag{5.4}$$

The extreme deconvolution approach can be thought of as belonging to a class of inference techniques known as Empirical Bayes (Carlin and Louis 2000). The density estimator $p(\mathbf{z} \mid \theta)$ is the prior distribution, whilst the noise model is the likelihood. Rather than fixing the prior based on our pre-existing beliefs as in a conventional Bayesian analysis, we fit it to the data. This can be effective if we have large quantities of data so that we can avoid overfitting the prior.

5.2.1 Expectation-Maximisation

The expectation-maximisation (EM) algorithm is a common method for fitting mixture models by maximising the marginal log-likelihood $\mathcal{L}(\theta)$ (Dempster, Laird, and Rubin 1977). Here we follow the derivation of Murphy (2012) for a general latent variable model with observed variables $\mathbf{x}_i$, latent variables $\mathbf{z}_i$ and parameters θ.

We start by defining the complete data log-likelihood

$$\mathcal{L}_C(\theta) = \sum_i^N \log p(\mathbf{x}_i, \mathbf{z}_i \mid \theta). \tag{5.5}$$

This cannot be evaluated because we do not know the values of $\mathbf{z}_i$. Instead, we can evaluate the expected complete data log-likelihood,

$$\mathcal{L}_E(\theta) = \sum_i^N \mathbb{E}_{p(\mathbf{z}_i \mid \mathbf{x}_i, \theta)}[\log p(\mathbf{x}_i, \mathbf{z}_i \mid \theta)] \tag{5.6}$$

where the expectation is with respect to the posterior over latent variables $p(\mathbf{z}_i \mid \mathbf{x}_i, \theta)$. For the models typically used with EM including GMMs, the posterior has a closed-form and many expectations of interest under it can be calculated using its sufficient statistics. We can modify the expected complete data log-likelihood to produce the auxiliary function,

$$Q(\theta^t, \theta^{t-1}) = \sum_i^N \mathbb{E}_{p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^{t-1})}[p(\mathbf{z}_i, \mathbf{x}_i \mid \theta^t)], \tag{5.7}$$

computed as part of an iterative procedure, using at step t the previous estimates of the parameters θ^{t-1} to compute the expectations, whilst using the new estimate of the parameters θ^t to evaluate the complete-data log-likelihood.

Starting with an initial set of parameters, the EM algorithm proceeds iteratively by finding the sufficient statistics of the posterior at iteration t in the expectation step (E-step) using θ^{t-1}, then finding the value of θ^t that maximises $Q(\theta^t, \theta^{t-1})$ in the maximisation step (M-step). Again, for the models typically used with EM, the M-step has a closed-form solution where the parameters are computed by normalising sums of the expected sufficient statistics.

To show how this procedure optimises the marginal likelihood, consider writing the marginal log-likelihood as an expectation under an arbitrary distribution $q(\mathbf{z})$,

$$\mathcal{L}(\theta) = \sum_i^N \log p(\mathbf{x}_i \mid \theta), \tag{5.8}$$

$$= \sum_i^N \log \left[\int p(\mathbf{x}_i, \mathbf{z}_i \mid \theta) \, \mathrm{d}\mathbf{z}_i \right], \tag{5.9}$$

$$= \sum_i^N \log \mathbb{E}_{q(\mathbf{z}_i)} \left[\frac{p(\mathbf{x}_i, \mathbf{z}_i \mid \theta)}{q(\mathbf{z}_i)} \right]. \tag{5.10}$$

Jensen's Inequality states that for any concave function $f(u)$ and probability distribution $q(u)$ expectations of the function form a lower bound on functions of the expectation,

$$f(\mathbb{E}_{q(u)}[u]) \geq \mathbb{E}_{q(u)}[f(u)]. \tag{5.11}$$

As $\log(u)$ is a concave function, we can form a lower-bound on Equation 5.10,

$$\mathcal{L}(\theta) \geq \sum_i^N \mathbb{E}_{q(\mathbf{z}_i)} \left[\log \frac{p(\mathbf{x}_i, \mathbf{z}_i \mid \theta)}{q(\mathbf{z}_i)} \right], \tag{5.12}$$

$$\geq \sum_i^N \mathbb{E}_{q(\mathbf{z}_i)} \left[\log p(\mathbf{x}_i, \mathbf{z}_i \mid \theta) \right] - \mathbb{E}_{q(\mathbf{z}_i)} \left[\log q(\mathbf{z}_i) \right]. \tag{5.13}$$

If we evaluate $\mathcal{L}(\theta)$ at θ^t and set $q(\mathbf{z}_i)$ to $p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^{t-1})$ then the first term in Equation 5.13 becomes the auxiliary function,

$$\mathcal{L}(\theta^t) \geq Q(\theta^t, \theta^{t-1}) - \mathbb{E}_{p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^{t-1})} \left[\log p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^{t-1}) \right]. \tag{5.14}$$

As the second term in Equation 5.14 is not a function of θ^t, maximising $Q(\theta^t, \theta^{t-1})$ with respect to θ^t will also maximise $\mathcal{L}(\theta^t)$. If we rewrite Equation 5.13 in terms of the posterior $p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^t)$, and set $q(\mathbf{z}_i)$ to it, we get

$$\begin{aligned} \mathcal{L}(\theta^t) &\geq \sum_i^N \Big\{ \mathbb{E}_{p(\mathbf{z}_i|\mathbf{x}_i,\theta^t)} \left[\log p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^t) + \log p(\mathbf{x}_i \mid \theta^t) \right] \\ &\qquad - \mathbb{E}_{p(\mathbf{z}_i|\mathbf{x}_i,\theta^t)} \left[\log p(\mathbf{z}_i \mid \mathbf{x}_i, \theta^t) \right] \Big\}, \qquad (5.15) \\ &\geq \sum_i^N \mathbb{E}_{p(\mathbf{z}_i|\mathbf{x}_i,\theta^t)} \left[\log p(\mathbf{x}_i \mid \theta^t) \right], \qquad (5.16) \\ &= \sum_i^N \log p(\mathbf{x}_i \mid \theta^t) \qquad (5.17) \end{aligned}$$

so the lower bound becomes tight and $\mathcal{L}(\theta^t) = Q(\theta^t, \theta^t)$.

We have $Q(\theta^{t-1}, \theta^{t-1}) = \mathcal{L}(\theta^{t-1})$ from Equation 5.17. By definition, we have picked a value θ^t such that $Q(\theta^t, \theta^{t-1}) \geq Q(\theta^{t-1}, \theta^{t-1})$. As the bound was tight at $Q(\theta^{t-1}, \theta^{t-1})$ and because maximising $Q(\theta^t, \theta^{t-1})$ will also maximise $\mathcal{L}(\theta^t)$ according to Equation 5.14, $\mathcal{L}(\theta^t) \geq Q(\theta^t, \theta^{t-1})$. Therefore,

$$\mathcal{L}(\theta^t) \geq Q(\theta^t, \theta^{t-1}) \geq Q(\theta^{t-1}, \theta^{t-1}) = \mathcal{L}(\theta^{t-1}), \qquad (5.18)$$

$$\mathcal{L}(\theta^t) \geq \mathcal{L}(\theta^{t-1}). \qquad (5.19)$$

so every successive step of the EM algorithm will monotonically increase the marginal log-likelihood and move closer towards a local maximum.

5.2.1.1 Expectation-Maximisation for the Extreme Deconvolution Model

To make this derivation more concrete, we consider the steps for the specific case of the extreme deconvolution method. Bovy, Hogg, and Roweis (2011) make use of the fact that with a GMM prior and Gaussian noise, the distribution over a latent datapoint $\mathbf{z}_i$ conditioned on an observed datapoint $\mathbf{x}_i$ is also a GMM,

$$p(\mathbf{z}_i \mid \mathbf{x}_i, S_i, R_i) = \sum_j^K r_{ij} \mathcal{N}(\mathbf{z}_i \mid \mathbf{b}_{ij}, B_{ij}). \qquad (5.20)$$

The E-step consists of computing the parameters of this GMM,

$$r_{ij}^t = \frac{\alpha_j^{t-1} \mathcal{N}(\mathbf{x}_i \mid R_i \mathbf{m}_j^{t-1}, T_{ij}^{t-1})}{\sum_k \alpha_k^{t-1} \mathcal{N}(\mathbf{x}_i \mid R_i \mathbf{m}_k^{t-1}, T_{ik}^{t-1})}, \qquad (5.21)$$

$$\mathbf{b}_{ij}^t = m_j^{t-1} + V_j^{t-1} R_i^T (T_{ij}^{t-1})^{-1} (\mathbf{x}_i - R_i \mathbf{m}_j^{t-1}), \qquad (5.22)$$

$$B_{ij}^t = V_j^{t-1} - V_j^{t-1} R_i^T (T_{ij}^{t-1})^{-1} R_i V_j^{t-1}. \qquad (5.23)$$

The r_{ij} term is the probability of datapoint $\mathbf{x}_i$ coming from component j. The $\mathbf{b}_{ij}$ and B_{ij} terms result from the fact that $\mathbf{x}_i$ and $\mathbf{z}_i$ are jointly Gaussian given component j, so the distribution of $\mathbf{z}_i$ conditioned on $\mathbf{x}_i$ given component j is also Gaussian with mean $\mathbf{b}_{ij}$ and covariance B_{ij}.

The M-step consists of finding the parameters which maximise the expected complete data log-likelihood by summing together the expected sufficient statistics,

$$\alpha_j^t = \frac{1}{N}\sum_i^N r_{ij}^t, \tag{5.24}$$

$$\mathbf{m}_j^t = \frac{1}{\sum_i^N r_{ij}^t}\sum_i^N r_{ij}^t \mathbf{b}_{ij}^t, \tag{5.25}$$

$$V_j^t = \frac{1}{\sum_i^N r_{ij}^t}\sum_i^N r_{ij}^t \left[(\mathbf{m}_j^t - \mathbf{b}_{ij}^t)(\mathbf{m}_j^t - \mathbf{b}_{ij}^t)^T + B_{ij}^t\right]. \tag{5.26}$$

Alternating between these two steps will result in finding parameters θ corresponding to a local-maximum of the marginal log-likelihood given by Equation 5.4. Careful initialisation can avoid degenerate local maxima (for example, a model where one component has a weight $\alpha_j = 1$ and the rest zero) but finding the global optimum is NP-hard (Drineas et al. 2004; Aloise et al. 2009).

For many problems, some of the values in a single sample may be missing. This can be handled by making the projection matrix rank-deficient, so that the observed sample $\mathbf{x}_i$ is a subset of the latent sample $\mathbf{z}_i$. Alternatively, values of the noise covariance S_i can be set to very large values. For the case where samples are completely missing or under-sampled as a result of truncation effects, Melchior and Goulding (2018) propose an extension to the EM algorithm that handles this by imputing the missing values.

5.3 Methods

The method for fitting the GMM presented in Section 5.2.1 is a batch algorithm, requiring us to compute the E-step for every datapoint in the dataset before we can perform the M-step. The reference implementation of Bovy, Hogg, and Roweis (2011) requires all of the data to fit in memory, which is unfeasible for large datasets such as the Gaia catalogue without resorting to specialised servers. Even if we could make use of a machine with sufficient memory or wrote an implementation that streamed the data from disk, we might be able to converge faster if we only looked at small minibatches of data at each iteration before updating the parameters.

5.3.1 Minibatch Expectation-Maximisation

Here we describe a minibatch version of the EM algorithm based on Cappé and Moulines (2009)'s online EM algorithm for latent data models. At each iteration t, we compute the sufficient statistics of the latent data $\mathbf{v}_i$ for each component j using our current estimate of the parameters in the E-step as before, but this time only over a minibatch of data of size M rather than the whole dataset of size N.

The expected sufficient statistics are then summed together over the minibatch,

$$q_j^t = \sum_i^M r_{ij}^t, \tag{5.27}$$

$$\mathbf{s}_j^t = \sum_i^M r_{ij}^t \mathbf{b}_{ij}^t, \tag{5.28}$$

$$S_j^t = \sum_i^M r_{ij}^t [\mathbf{b}_{ij}^t \mathbf{b}_{ij}^{tT} + B_{ij}^t]. \tag{5.29}$$

Stochastic estimates $\hat{\phi}_j^t$ of the sums of sufficient statistics over the whole dataset are then updated with a sufficiently small step size λ,

$$\phi_j^t = \{q_j^t, \mathbf{s}_j^t, S_j^t\}, \tag{5.30}$$

$$\hat{\phi}_j^{t-1} = \{\hat{q}_j^{t-1}, \hat{\mathbf{s}}_j^{t-1}, \hat{S}_j^{t-1}\}, \tag{5.31}$$

$$\hat{\phi}_j^t = (1-\lambda)\hat{\phi}_j^{t-1} + \lambda\phi_j^t. \tag{5.32}$$

Finally, we normalise the updated sums of expected sufficient statistics to get updated estimates of the parameters,

$$\alpha_j^t = \frac{\hat{q}_j^t}{M}, \tag{5.33}$$

$$\mathbf{m}_{j^t} = \frac{\hat{\mathbf{s}}_j^t}{\hat{q}_j^t}, \tag{5.34}$$

$$V_j^t = \frac{\hat{S}_j^t}{\hat{q}_j^t} - \mathbf{m}_j^t \mathbf{m}_j^{tT}. \tag{5.35}$$

This procedure is repeated with new randomly-ordered minibatches until convergence. If we set $\lambda = 1$ and replace each minibatch with the entire dataset, then the update corresponds to the original batch fitting method. Numerically however, the update for V_j, as written in Equation 5.35, is inadvisable compared to the batch update given in Bovy, Hogg, and Roweis (2011). Catastrophic cancellation can occur if the variances of the components are small relative to the means, especially if single precision floats are used, as is standard with GPU computation (Schubert and Gertz 2018).

5.3.1.1 Stable Covariance Update

Here we present an alternative update for V_j that is less prone to numerical instability, and show that it is equivalent to Equation 5.35. For clarity we drop the component indicator j from the parameter.

First, we define an adjustment operation,

$$\text{adjust}(V, s, \mathbf{c}, \mathbf{d}) = sV + \frac{1}{2}(\sqrt{s}\mathbf{c} - \mathbf{d})(\sqrt{s}\mathbf{c} + \mathbf{d})^T + \frac{1}{2}(\sqrt{s}\mathbf{c} + \mathbf{d})(\sqrt{s}\mathbf{c} - \mathbf{d})^T \tag{5.36}$$

$$= s(V + \mathbf{c}\mathbf{c}^T) - \mathbf{d}\mathbf{d}^T, \tag{5.37}$$

which can be thought of as recentring a scaled variance around a new mean. Equation 5.36 is how we actually compute the adjustment, to minimise taking small differences between large values, whilst Equation 5.37 shows the identity we are interested in.

In the M-step at iteration t of our minibatch EM approach, we compute estimates of $\hat{q}_t$, α_t and $\mathbf{m}_t$ as before using Equations 5.32 and 5.35. We also compute minibatch-specific parameters using exact sums over the minibatch:

$$q_b = \sum_i^M r_i, \quad \mathbf{m}_b = \frac{\sum_i^M r_i \mathbf{x}_i}{q_b}, \quad V_b = \frac{\sum_i^M r_i[(\mathbf{x}_i - \mathbf{b}_i)(\mathbf{x}_i - \mathbf{b}_i)^T + B_i]}{q_b} \tag{5.38}$$

We then compute our new estimate of the variance V_t as a function of the previous estimates $\{\hat{q}_{t-1}, \mathbf{m}_{t-1}, V_{t-1}\}$, the minibatch values $\{q_b, \mathbf{m}_b, V_b\}$, and the current estimates $\{\hat{q}_t, \mathbf{m}_t\}$:

$$V_t = (1-\lambda)\,\text{adjust}(V_{t-1}, \frac{\hat{q}_{t-1}}{\hat{q}_t}, \mathbf{m}_{t-1}, \mathbf{m}_t) + \lambda\,\text{adjust}(V_b, \frac{q_b}{\hat{q}_t}, \mathbf{m}_b, \mathbf{m}_t) \tag{5.39}$$

$$\begin{aligned} &= (1-\lambda)\left[\frac{\hat{q}_{t-1}}{\hat{q}_t}\left(V_{t-1} + \mathbf{m}_{t-1}\mathbf{m}_{t-1}^T\right) - \mathbf{m}_t\mathbf{m}_t^T\right] \\ &\quad + \lambda\left[\frac{q_b}{\hat{q}_t}\left(V_b + \mathbf{m}_b\mathbf{m}_b^T\right) - \mathbf{m}_t\mathbf{m}_t^T\right] \end{aligned} \tag{5.40}$$

$$= (1-\lambda)\left[\frac{\hat{S}_{t-1}}{\hat{q}_t} - \mathbf{m}_t\mathbf{m}_t^T\right] + \lambda\left[\frac{S_t}{\hat{q}_t} - \mathbf{m}_t\mathbf{m}_t^T\right] \tag{5.41}$$

$$= \frac{(1-\lambda)\hat{S}_{t-1} + \lambda S_t}{\hat{q}_t} - \mathbf{m}_t\mathbf{m}_t^T \tag{5.42}$$

$$= \frac{\hat{S}_t}{\hat{q}_t} - \mathbf{m}_t\mathbf{m}_t^T \tag{5.43}$$

Again, Equation 5.39 is how we actually compute the update to minimise numerical errors, whilst Equation 5.43 shows that the update is equivalent to the covariance update defined in Equation 5.35. Whilst we found this update worked better in practice than a direct implementation, numerical instability is still possible. In particular, it is possibly for the prior distribution $p(\mathbf{z})$ and marginal distribution $p(\mathbf{x})$ to have well-behaved covariances for each component, whilst the posterior distribution $p(\mathbf{z} \mid \mathbf{x}, S)$ can have singular covariances for particular values of $\mathbf{z}_i$ in the training dataset.

5.3.2 Stochastic Gradient Descent

An alternative to EM-based methods is to optimise the marginal log-likelihood (Equation 5.4) directly using stochastic gradient descent (SGD, Bottou, Curtis, and Nocedal 2018). The optimization is constrained, because the mixture weights a_j are positive and sum to 1, and the covariances V_j are positive definite. Directly fitting the log-likelihood with unconstrained gradient-based optimisers requires a transformation of the parameters to remove the constraints (Williams 1996). The mixture weights α_j can be parameterised by taking the softmax of an unconstrained vector $\mathbf{u}$,

$$\alpha_j = \frac{e^{u_j}}{\sum_{k=1}^K e^{u_k}}. \tag{5.44}$$

The covariances V_j can be represented by their lower triangular Cholesky decomposition L_j, where the diagonal elements qq of L_j are constrained positive by taking the exponential of unconstrained elements $\tilde{L}_q$,

$$(L_j)_{qq} = \exp(\tilde{L}_q), \tag{5.45}$$

$$V_j = L_j {L_j}^T. \tag{5.46}$$

Having removed the constraints, we can optimise the likelihood using any standard minibatch gradient-based optimiser. In our implementation we use the Adam optimiser (Kingma and Ba 2014). We can avoid having to manually work out the gradients by making use of the automatic differentiation available in many numerical computing packages (Baydin, Pearlmutter, et al. 2018). Implementation is straightforward, with code for the log-probability calculations available in many frameworks, and even the code for performing the necessary reparameterisations (Bingham et al. 2019).

For a standard Gaussian mixture model, gradient based optimization has a scaling advantage over EM. There is no need to form the $D \times D$ covariance matrix V_j, since the Gaussian density can be evaluated directly from the Cholesky factor L_j in $O(D^2)$, whereas an EM update is $O(D^3)$. Unfortunately SGD updates are also $O(D^3)$ for the extreme deconvolution model, as we need to form the covariance T_{ij} for each datapoint.

5.4 Experiments

We implemented both minibatch approaches in PyTorch (Paszke et al. 2019), and compared against the reference implementation from Bovy, Hogg, and Roweis (2011). To evaluate each method, we first run them on a toy synthetic problem to check their correctness. We then ran them using data from the Gaia catalogue to evaluate and compare their performance.

5.4.1 Synthetic Data

First, we use a simple 2D 2-component Gaussian mixture model as our latent distribution with mixture weights, means and covariances

$$\alpha = 0.5, \tag{5.47}$$

$$\mu_0 = \mu_1 = [0, 0], \tag{5.48}$$

$$C_0 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}, \tag{5.49}$$

$$C_1 = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \tag{5.50}$$

$$k_i \sim \text{Bernoulli}(\alpha), \tag{5.51}$$

$$z_i \sim \mathcal{N}(m_{k_i}, C_{k_i}). \tag{5.52}$$

Figure 5.1a shows a corner plot of samples drawn from this mixture model.

We set the projection matrix R_i to the identity matrix for every sample. Noise was added by generating a diagonal covariance S_i for each datapoint z_i, sampling a value s_{id} for each entry on the diagonal from a unit scale Log-normal distribution, then multiplying it by a noise-scale factor η before squaring to get the variance. A noise vector ϵ_i was then sampled from a multivariate normal with zero mean and S_i covariance,

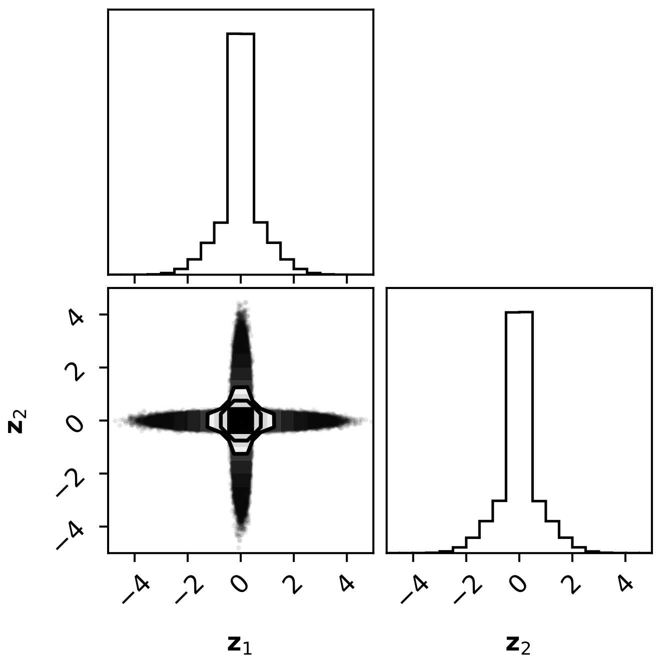
$$s_{id} \sim \text{LogNormal}(0, 1), \quad d \in \{0, 1\} \tag{5.53}$$

$$S_i = \begin{bmatrix} (\eta s_{i0})^2 & 0 \\ 0 & (\eta s_{i1})^2 \end{bmatrix}, \tag{5.54}$$

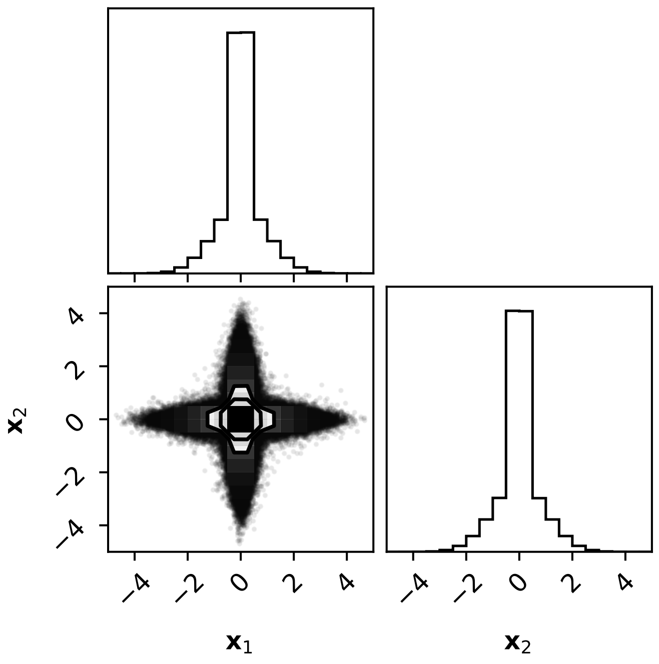
$$\epsilon_i \sim \mathcal{N}(0, S_i). \tag{5.55}$$

The use of the noise-scale η allows us to control the relative size of the noise compared to the latent data z. Figure 5.1b shows a corner plot of the observed data x drawn using this scheme and a noise-scale factor η of 0.1

We fitted both our minibatch-EM and SGD methods to this data generated using a noise-scale factor η of 0.1 and setting K to 2. We generated 2 million samples for the training set, reserving 10% for a held-out validation set. Appendix A.1 contains additional details for reproducibility. We also fitted the reference implementation of Bovy, Hogg, and Roweis (2011) as a baseline.



(a) Corner plot of latent $\mathbf{z}$ samples from the two component mixture data problem.



(b) Corner plot of noisy observed $\mathbf{x}$ samples from the mixture data problem.

Figure 5.1: Latent noise-free and observed noisy training samples from the mixture data problem described in Section 5.4.1. The Gaussian noise blurs out the underlying latent distribution.

Table 5.1: Mean per-datapoint validation log-likelihoods averaged over five runs for a two component model fitted to the synthetic data using each method as described in Section 5.4.1. All methods have comparable results, with the SGD method doing slightly better.

	$\log p(\mathbf{x})$	$\log p(\mathbf{z})$
Existing EM	-1.463 ± 0.002	-1.121 ± 0.008
Minibatch EM	-1.464 ± 0.000	-1.120 ± 0.000
SGD	-1.460 ± 0.000	-1.107 ± 0.000

Table 5.1 reports mean log-likelihoods for both $p(\mathbf{x})$ and $p(\mathbf{z})$ using the validation dataset, averaged over 5 runs. The values are all roughly comparable. Figure 5.2 shows corner plots of example samples from $p(\mathbf{z})$ fitted with each method, along with samples from the ground-truth distribution. We can see that each method has managed to recover the underlying noise-free distribution. This is a straightforward model to fit, so we cannot make any claims of superior performance for a particular method, but it demonstrates that they are behaving correctly.

5.4.2 Gaia Data

We used a random sample of rows from the Gaia DR3 source table (The Gaia Collaboration 2022). Each sample consist of five measurements of a particular star's position and motions[1], referred to as astrometric measurements, and has a full-covariance Gaussian noise level associated with it. In total there were 2 million rows divided into training, validation and test sets. We set the projection R_i to the identity matrix for every sample. This experiment uses only a small fraction of the full dataset size (approximately 0.1%), but this allows us to fit the training data into memory, a requirement for use with the original implementation of extreme deconvolution. We used a range of mixture component sizes K. In practice we would want to select a value of K by cross-validation.

The existing EM method ran on CPU, whilst the minibatch EM and SGD

[1]Right ascension, declination, parallax and proper motions in the right ascension and declination directions.

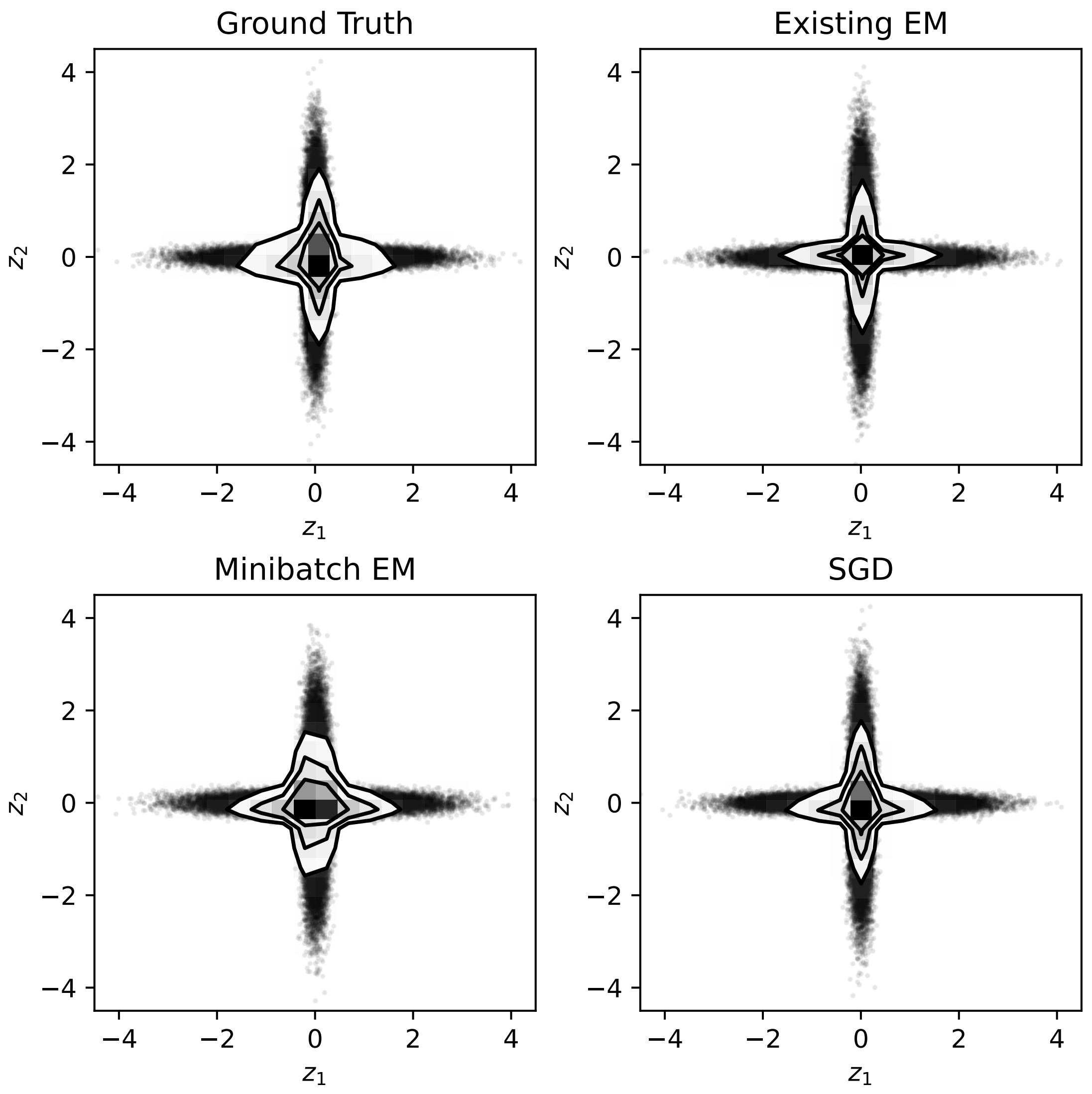


Figure 5.2: Examples of models fitted with each method to the two-component synthetic dataset along with the ground-truth, show as histograms of samples from $p(\mathbf{z})$ using the format as described in Section 2.2.1. Each method has visually matched the shape of the underlying noise-free distribution.

methods ran on GPU. While the absolute times depend strongly on hardware and fine implementation details, they give a sense of the sort of times possible on current workstations, and the relative times across model sizes illustrate how the methods scale. We used a validation set comprising 10% of the rows when developing our experiments. Final model performance was evaluated on a different held-out test set comprising an additional 10% of the rows at the last stage, with no parameter selection or development done based on this set. Additional details required for reproducibility are provided in Appendix A.2.

Table 5.2 reports the training and validation log-likelihoods for each method. In this experiment we are using real data, so we only have access to the noisy values $\mathbf{x}$ and cannot report log-likelihoods for $p(\mathbf{z})$. The minibatch-EM method required a small value (10^{-3}) to be added to the diagonal of the component covariances to ensure numerical stability, so the results are not exactly comparable. In general at the larger values of K the SGD method appears to outperform both EM-based methods. After comparing on the validation results, we evaluated $\log p(\mathbf{x})$ on the test set for $K = 512$ to confirm there was no overfitting.

Figure 5.3 plots the training log-likelihood against time-rescaled epoch for $K = 256$. Both minibatch methods converge faster in terms of wall-clock time. We also found them to converge faster by number of training epochs. Figure 5.4 shows total training time as function of mixture components K. Both minibatch methods are faster than the baseline as they utilise a GPU for computation, with the SGD method being slightly faster than the Minibatch EM method as it does not have to compute the posterior parameters.

Table 5.2: Average per-datapoint validation log-likelihoods for the Gaia data subset for different numbers of mixture components K, with average test log-likelihood for the best value of K by validation. Averaged over 5 runs with standard deviation.

Method	K	Train $\log p(\mathbf{x})$	Val $\log p(\mathbf{x})$	Test $\log p(\mathbf{x})$
Existing EM	64	-16.017 ± 0.007	-16.016 ± 0.006	–
	128	-15.978 ± 0.006	-15.980 ± 0.005	–
	256	-15.953 ± 0.002	-15.958 ± 0.002	–
	512	-15.936 ± 0.000	-15.944 ± 0.000	-15.933 ± 0.001
Minibatch EM	64	-15.986 ± 0.007	-15.989 ± 0.008	–
	128	-15.959 ± 0.001	-15.964 ± 0.001	–
	256	-15.948 ± 0.001	-15.954 ± 0.001	–
	512	-15.944 ± 0.000	-15.950 ± 0.000	-15.941 ± 0.000
SGD	64	-16.003 ± 0.007	-16.007 ± 0.008	–
	128	-15.961 ± 0.002	-15.966 ± 0.002	–
	256	-15.936 ± 0.001	-15.942 ± 0.001	–
	512	-15.921 ± 0.001	-15.929 ± 0.001	-15.920 ± 0.001

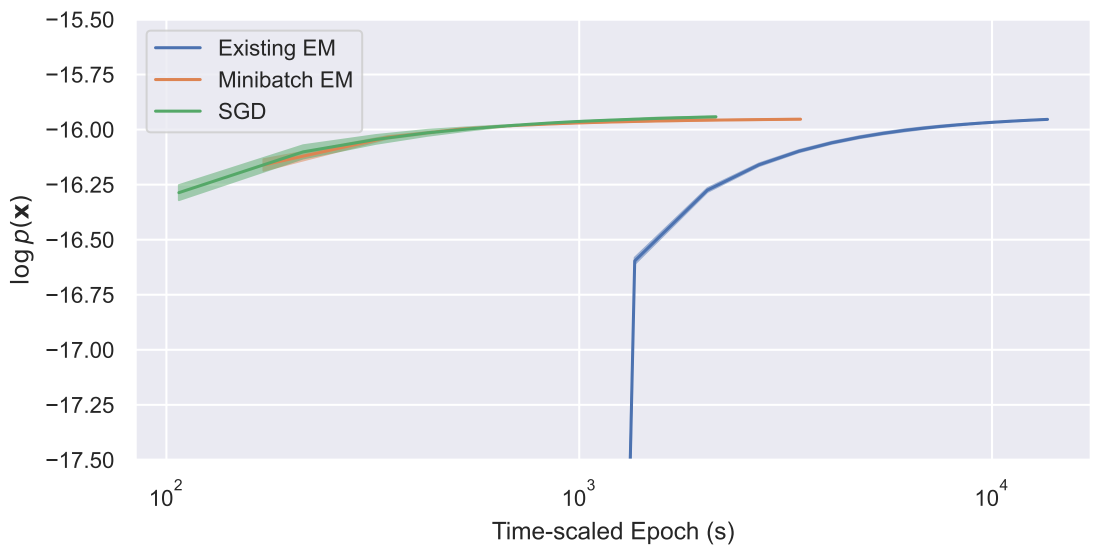


Figure 5.3: Average training log-likelihood as a function of training on the Gaia subset for models with $K = 256$. Epochs rescaled by the average training time for each method. Error bars show $\pm$ 2 standard deviations. Note the log-scale on the time axis, and that the start of the curve for the existing EM method has been clipped.

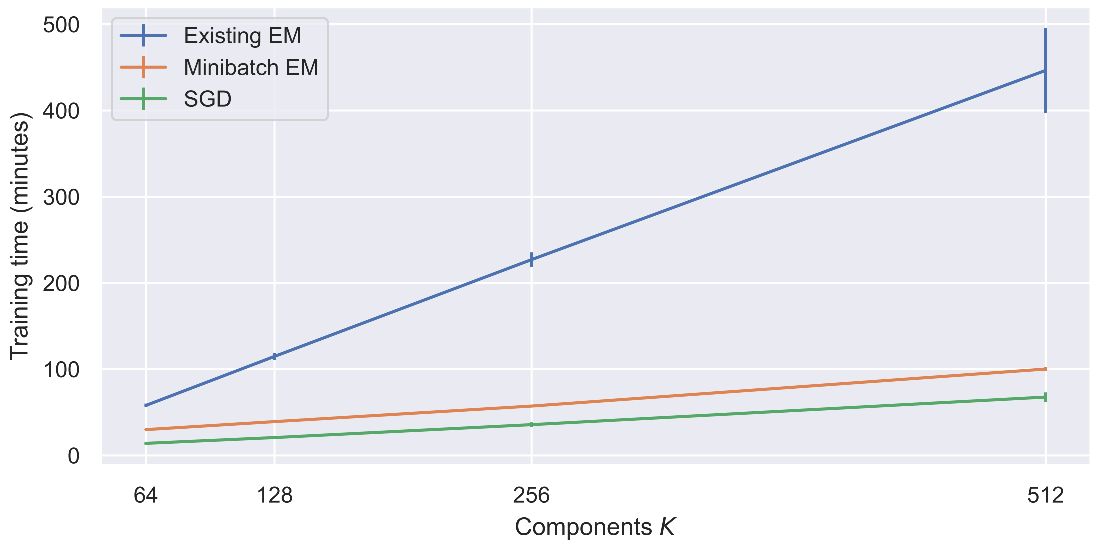


Figure 5.4: Training time as a function of mixture components K. Error bars indicate $\pm$ 2 standard deviations. Training time appears to be linear as expected. The minibatch methods running on GPU are considerably faster than the existing EM implementation running on CPU.

5.4.3 Pitfalls

Here we describe two problematic issues a practitioner using this method for deconvolution could encounter with real-world problems, using synthetic datasets to demonstrate them. Although these demonstrations were done with the SGD method, the issues highlighted are not specific to the fitting method and affect both the minibatch-EM and batch-EM methods as well.

5.4.3.1 Overparameterised Model

Here we show how the validation log-likelihood alone cannot be used to assess model performance, with a simple synthetic example. We first generate data in one dimension by drawing both values of z_i and noise values ϵ_i from standard univariate normals,

$$z_i \sim \mathcal{N}(0,1), \tag{5.56}$$

$$\epsilon_i \sim \mathcal{N}(0,1), \tag{5.57}$$

$$x_i = z_i + \epsilon_i. \tag{5.58}$$

Fitting a single Gaussian to this in the deconvolution setting is straightforward, and in practice with our implementations this can be done by fitting a single-component "GMM". We instead fit an overparametrised GMM with 32 components using the SGD method, a training set of 9000 samples and a validation set of 1000 samples. The overparametrised GMM was initialised to have equal weights for each component, with component means initialised to be linearly spaced between -3 and $+3$ and component standard deviations set to 0.001.

Table 5.3 reports validation log-likelihoods for both $p(z)$ and $p(x)$ for the fitted model along with ground-truth values. Unsurprisingly the overparameterised model has overfitted the noise-free data and this is reflected in validation log-likelihood for $p(z)$ being several orders of magnitude smaller. However, the validation log-likelihoods for $p(x)$ are much closer. In a real modelling situation we would only have access to $\log p(\mathbf{x})$ and hence would not be able to detect the overfit based solely on these values.

To see why this happens, in Figure 5.5 we plot the PDFs for both the ground-truth model and the fitted model over both z and x.[2] The fitted density estimator

[2]As we use the same standard deviation for every noise-value ϵ_i there is only a single PDF

Table 5.3: Per-datapoint validation log-likelihoods for the overparametrised model, along with values from the ground-truth model. There are extreme differences in the values for $\mathbf{z}$ compared to the differences for $\mathbf{z}$.

Model	$\log p(z)$	$\log p(x)$
Ground Truth	-1.4168	-1.7569
Fitted	-4314.0239	-1.7581

for $\log p(z)$ is far from the ground truth, consisting of a series of narrow spikes. However, when convolved with the noise distribution the fitted density is almost identical to the convolved ground truth, to the extent that in some pilot runs with only a small finite set of validation samples we observed the fitted density having a (slightly) better validation score.

This is a contrived example requiring a massively overparametrised density estimator and large noise values of equal scale to the data, but it demonstrates that we should not rely solely on the validation log marginal-likelihood when trying to assess fitted density estimators on noise-free data. Instead we should aim to have additional methods of validating that the fitted density estimator is behaving reasonably, especially if we have noise with scale comparable to the noise-free data that will substantially blur the underlying distribution when convolved with it. The choice of additional validation method will depend strongly on the downstream task the deconvolved density estimator is intended for. We could also choose to use a model selection criterion which penalises the number of model parameters such as the Bayesian information criterion (BIC) to select the number of components K (Schwarz 1978).

5.4.3.2 Conditional Noise

In our description of the problem in Section 5.4.1, we assumed that the noise ϵ_i was drawn independently of the latent noise-free data $\mathbf{z}_i$, i.e. the noise covariance S_i does not depend on the value of $\mathbf{z}_i$. In this section we show what happens if this assumption is incorrect. We reuse the same two component GMM ground-truth model from Section 5.4.1, but change the noise model. We

over x in this case, which is not true for the more general extreme deconvolution case where the noise statistics can be different for each datapoint.

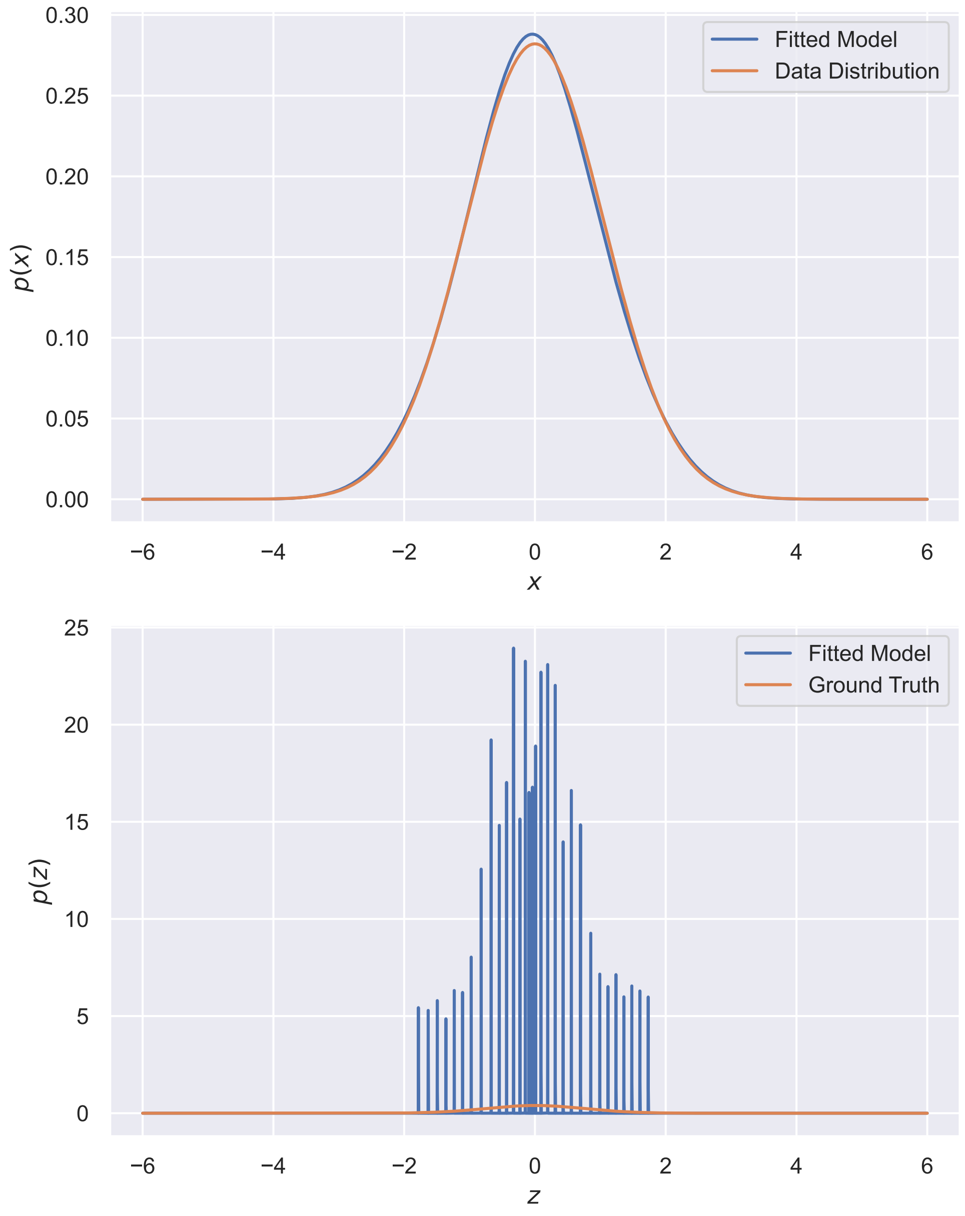


Figure 5.5: Illustration of how a model $p(\mathbf{z})$ overfitted to the latent data can still result in a reasonable marginal distribution when convolved with the noise distribution to produce $p(\mathbf{x})$. This makes it hard to assess the quality of a fitted model by relying solely on validation log-likelihoods on $p(\mathbf{x})$.

define two covariance matrices,

$$S_0 = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \tag{5.59}$$

$$S_1 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}. \tag{5.60}$$

When generating the data, we draw ϵ_i from a multivariate Gaussian with covariance S_0 if z_i was drawn from the component $j = 0$ with covariance C_0, and with covariance S_1 if C_1 was used, so S_i depends on z_i. We then fit a two-component GMM to this data using the SGD implementation. Unlike in Section 5.4.1, we can no longer recover the ground-truth model exactly, as shown in Figure 5.6. The fitted model has shrunk the distribution relative to the observed data, but it is isotropic and has failed to recover the structure of the latent data. If we resimulate the data generation by adding noise to samples from the fitted model using the same scheme, the distribution of resimulated data is wider than the original.

This example demonstrates that if we have reason to believe that the noise in an observed dataset depends on the latent noise-free values (not an unreasonable assumption for many real-world datasets including the Gaia catalogue (Anderson et al. 2018)), we should be careful to note that the framework we have presented will result in a misspecified model.

5.5 Discussion

Our results have shown that both of our proposed methods perform comparably to the existing method of fitting extreme deconvolution model in terms of final marginal log-likelihoods. For larger numbers of components K we found the SGD method slightly outperformed the other methods. Both minibatch methods converged considerably faster than the existing batch EM method, which is consistent with similar results comparing online to batch EM in discrete language tasks (Liang and Klein 2009).

The timing results show that using GPU-based computation speeds up fitting considerably. Computational cost is dominated by the need to perform K-component by N-datapoint Cholesky decompositions of the covariances T_{ij} each with $O(D^3)$ cost over every training epoch. PyTorch's GPU-backed routine

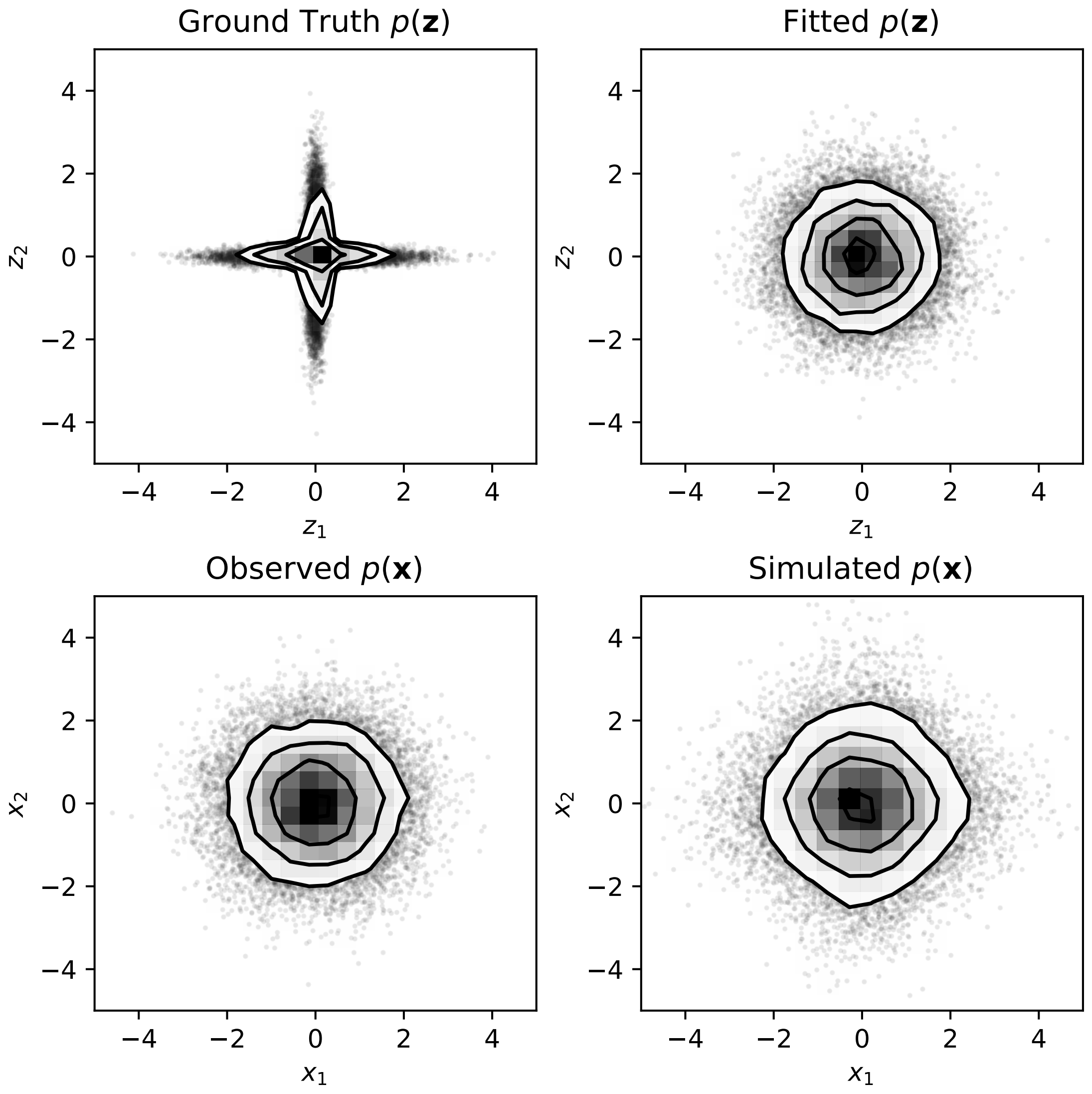


Figure 5.6: Histograms of ground-truth latent noise-free and observed noisy samples where the noise covariance depends on the component z was sampled from. The fitted model is unable to recover the underlying structure from the observed data as it is misspecified.

for the Cholesky decomposition allows large numbers of these decompositions to be done in parallel, leading to the large observed speed-up. For smaller values of K and/or smaller minibatch sizes we even observed sub-linear scaling on the GPU with the number of Cholesky decompositions performed. Our astrometric experiments used only $D = 5$ features, so for models with larger dimensionality we would expect the absolute difference in training times to be even larger. Training time could also be saved by taking advantage of the minibatch method's faster convergence, requiring fewer epochs to reach the same result. The SGD method is slightly faster than the minibatch-EM method, as it avoids having to compute the posterior parameters on every step.

Both minibatch methods could be affected by numerical issues when run on a GPU, as GPUs work primarily with single-precision floats. In Section 5.3.1.1 we noted that the GMM for the posterior $p(\mathbf{z} \mid \mathbf{x}, S)$ could have components with near-singular covariances even when the GMMS for the marginal distributions $p(\mathbf{x})$ and prior $p(\mathbf{z})$ were well-behaved. This is not an issue for fitting with the SGD method, as it does not require evaluating the posterior during training. If we want to do post-hoc sampling from the posterior, the singular covariances usually have small component weights associated with them, so we can modify the sampling routine to drop those components. It is an issue for the minibatch EM method, which requires the posterior parameters to be evaluated for the E-step. The minibatch-EM also suffers from the posterior weights underflowing for some components, resulting in divide-by-zero errors in the M-step and requiring a small correction term to be added.

In our experiments we did not regularise our parameters beyond small numerical adjustments required to ensure the stability of the minibatch-EM method. Bovy, Hogg, and Roweis (2011) show that proper regularisation can be done for the EM approach using conjugate priors on the parameters of the GMM, turning the solution into a maximum-a-posterior (MAP) estimate. This regularized objective function can also be optimized by our Minibatch-EM method. The need for a conjugate prior is required in order to ensure the M-step has a closed-form solution. This is particularly problematic for the prior over covariances, as the conjugate distribution is the Inverse-Wishart distribution, the parametrisation of which makes it hard to disentangle prior beliefs over correlations and scales (Tokuda et al. 2011). By contrast we can do MAP estimation with the SGD method using any prior we wish. This would allow us to decompose the prior over covariances into a prior over correlations using the Lewandowski-Kurowicka-Joe (LKJ) distribution (Lewandowski, Kurowicka, and Joe 2009), and separate priors over the scale or variances in each dimension (Barnard, McCulloch, and Meng 2000).

In general, going forward we would recommend the SGD method as the preferred method when fitting GMMs in the extreme deconvolution case, based on the greater flexibility in regularisation, faster convergence, ease of implementation, better numerical stability and better validation performance. We would also recommend the need for additional validation checks beyond just the validation marginal log-likelihood for any fitting method, as well as the need to be aware that the model will be misspecified if noise depends on latent-data.

Chapter 6

Extreme Deconvolution with Variational Inference

6.1 Introduction

In the previous chapter, we showed that fitting Gaussian Mixture Models (GMM) to noisy data can be done in a scalable manner by making use of minibatch optimisation methods and GPU computation, and that optimising the marginal-likelihood directly with stochastic gradient-based methods can perform better than expectation-maximisation (EM)-based methods. These developments allow deconvolution to be done on much larger datasets such as that produced by the Gaia astronomy mission, which will eventually contain noisy measurements of approximately 1 billion astronomical objects (The Gaia Collaboration 2016).

However, the approaches we outlined still have limitations. They require the marginal likelihood to be tractable, generally restricting us to the case where the density estimator is a GMM and the noise is Gaussian. Some datasets may benefit from using more flexible density estimators such as normalising flows (Papamakarios, Nalisnick, et al. 2021), and many datasets are corrupted with non-Gaussian noise as well as Gaussian noise. Prior work has attempted to fit normalising flows directly to noisy Gaia data, and has indicated potential improvements over GMMs, but relied on a heuristic weighting scheme rather than formally setting up the model as a deconvolution problem (Cranmer, Galvez, et al. 2019).

In this chapter we show how these limitations can be removed by using amortised variational inference to optimise a lower-bound on the log marginal-likelihood. This approach allows us to fit a much wider class of more-flexible density estimators with better performance to the underlying noise-free data, and to use datasets with non-Gaussian noise as well as Gaussian noise. We demonstrate this application of variational inference works in practice, using experiments on both synthetic and real-world data with both GMMs and normalising flows, as well as Gaussian and non-Gaussian noise. The results show that normalising flows enabled by the use of variational inference can outperform GMMs in terms of the density estimation.

6.2 Background

The aim of density deconvolution is to perform density estimation on some noisy d-dimensional dataset $\{\mathbf{x}_i\}_{i=1}^N$. We assume that each datapoint in the dataset was generated by the addition of a noise vector ϵ_i to a latent noise-free datapoint $\mathbf{z}_i$,

$$\mathbf{x}_i = \mathbf{z}_i + \epsilon_i. \tag{6.1}$$

The noise vector ϵ_i is drawn from some noise distribution

$$\epsilon_i \sim p(S_i) \tag{6.2}$$

where S_i is a known set of parameters. Often $p(S_i)$ will be a zero-mean multivariate Gaussian with S_i as a covariance matrix, but this is not a strict requirement for the approaches outlined in this chapter.

We wish to estimate the noise-free data probability density $p(\mathbf{z})$ rather than the noisy distribution $p(\mathbf{x})$ by fitting a density estimator $p_\theta(\mathbf{z})$ with parameters θ. In the extreme deconvolution setting, we assume that the noise parameters are unique to each datapoint (Bovy, Hogg, and Roweis 2011).

6.3 Theory

We can fit the density estimator by finding the parameters θ which maximise the marginal likelihood

$$\operatorname*{argmax}_\theta \prod_{i=1}^N p(\mathbf{x}_i) = \prod_{i=1}^N \int p(\mathbf{x}_i \mid \mathbf{z}_i, S_i)\, p_\theta(\mathbf{z}_i)\, \mathrm{d}\mathbf{z}_i, \tag{6.3}$$

or equivalently the log-marginal likelihood.

$$\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(\mathbf{x}_i) = \sum_{i=1}^{N} \log \int p(\mathbf{x}_i \mid \mathbf{z}_i, S_i) p_\theta(\mathbf{z}_i) \, \mathrm{d}\mathbf{z}_i. \tag{6.4}$$

In Chapter 5, we restricted the noise distribution $p(\mathbf{x}_i \mid \mathbf{z}_i, S_i)$ to be Gaussian and the density estimator to be a mixture of Gaussians. This makes the integral in Equation 6.4 tractable and means the marginal distribution $p(\mathbf{x})$ and posterior $p(\mathbf{z} \mid \mathbf{x}, S)$ are also mixtures of Gaussians, allowing the marginal distribution to be evaluated and optimised directly.

If we wish to use a different family of density estimators $p_\theta(\mathbf{z})$ instead of a mixture of Gaussians, or use more general noise models then the log-marginal likelihood is no longer tractable in general. Fitting these more general models will require us resort to approximate inference methods to estimate the marginal likelihood.

6.3.1 Variational Inference

In this chapter we use variational inference as our approximate inference method (e.g. Jordan et al. 1999). We follow the review of Kingma and Welling (2019) to provide an overview of variational inference in our context.

We start by noting that in the general case described above, as well as being unable to evaluate the marginal likelihood directly, we can no longer evaluate the posterior density $p(\mathbf{z} \mid \mathbf{x}, S)$ or easily draw samples from it. As an alternative, we will try to fit an approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ to match the true posterior as closely as possible by optimising its parameters ϕ. Importantly, we need to be able to both sample from and evaluate the PDF of this approximate posterior.

Assuming we have an appropriate approximate posterior family, we need an objective function in order to fit it. A common choice in variational inference is the reverse Kullback–Leibler (KL) divergence, which gives us a quantitative measure of how different one probability distribution is from another. The reverse KL divergence $D_{\mathrm{KL}}(q \,||\, p)$ between our approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$

and the true posterior $p(\mathbf{z} \mid \mathbf{x}, S)$ is formally defined as

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}, S) \,||\, p(\mathbf{z} \mid \mathbf{x}, S)) = \int q_\phi(\mathbf{z} \mid \mathbf{x}, S) \frac{\log q_\phi(\mathbf{z} \mid \mathbf{x}, S)}{\log p(\mathbf{z} \mid \mathbf{x}, S)} \, \mathrm{d}\mathbf{z}, \tag{6.5}$$

$$= \int q_\phi(\mathbf{z} \mid \mathbf{x}, S) \log q_\phi(\mathbf{z} \mid \mathbf{x}, S) \, \mathrm{d}\mathbf{z} - \int q_\phi(\mathbf{z} \mid \mathbf{x}, S) \log p(\mathbf{z} \mid \mathbf{x}, S) \, \mathrm{d}\mathbf{z}, \tag{6.6}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log q_\phi(\mathbf{z} \mid \mathbf{x}, S)] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{z} \mid \mathbf{x}, S)] \tag{6.7}$$

This divergence is non-negative ($D_{\mathrm{KL}} \geq 0$), and it is exactly zero if and only if the approximate posterior exactly equals the true posterior. We can also similarly define the forward KL-divergence $D_{\mathrm{KL}}(p \,||\, q)$ where the two distributions are swapped. In general the KL divergence is not symmetric, i.e. $D_{\mathrm{KL}}(q \,||\, p) \neq D_{\mathrm{KL}}(p || \, q)$. The log terms allow us to interpret the KL divergence in terms of information theory, and it is sometimes referred to as the *relative entropy* (MacKay 2003, Chapter 2).

Even if our approximate posterior is tractable, we still cannot use the reverse KL-divergence directly as a loss function, as Equation 6.7 requires us to be able to evaluate $\log p(\mathbf{z} \mid \mathbf{x}, S)$. We can however derive a bound from it. Starting from the definition of Bayes rule,

$$p(\mathbf{z} \mid \mathbf{x}, S) = \frac{p(\mathbf{x}|\mathbf{z}, S) p_\theta(\mathbf{z})}{p(\mathbf{x})}, \tag{6.8}$$

and therefore

$$\log p(\mathbf{z} \mid \mathbf{x}, S) = \log p(\mathbf{x}|\mathbf{z}, S) + \log p_\theta(\mathbf{z}) - \log p(\mathbf{x}). \tag{6.9}$$

Substituting this expression for $\log p(\mathbf{z} \mid \mathbf{x}, S)$ into Equation 6.7 we can rewrite the divergence as

$$\begin{aligned} D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}, S) \,||\, p(\mathbf{z} \mid \mathbf{x}, S)) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log q_\phi(\mathbf{z} \mid \mathbf{x}, S)] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[p(\mathbf{x}|\mathbf{z}, S)] \\ - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[p_\theta(\mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{x})], \end{aligned} \tag{6.10}$$

and noting that $\log p(\mathbf{x})$ is not a function of $\mathbf{z}$ we can simplify it further to

$$\begin{aligned} D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}, S) \,||\, p(\mathbf{z} \mid \mathbf{x}, S)) = & \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log q_\phi(\mathbf{z} \mid \mathbf{x}, S)] \\ & - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{x}|\mathbf{z}, S)] \\ & - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p_\theta(\mathbf{z})] \\ & + \log p(\mathbf{x}). \end{aligned} \tag{6.11}$$

As $D_{\mathrm{KL}}(q\,||\,p) \geq 0$, if we negate the expectation terms in Equation 6.11 they form a lower bound on the log marginal likelihood or evidence $\log p(\mathbf{x})$, referred to as the evidence lower bound (ELBO),

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{x}|\mathbf{z},S)] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p_\theta(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log q_\phi(\mathbf{z} \mid \mathbf{x}, S)] \leq \log p(\mathbf{x}). \tag{6.12}$$

The terms inside the expectations are tractable and by negation the lower bound can be used as a loss function $\mathcal{L}(\theta, \phi)$, allowing us to jointly fit $p_\theta(\mathbf{z})$ to the noise-free data and $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ to the corresponding posterior $p(\mathbf{z} \mid \mathbf{x}, S)$ by optimisation with respect to both θ and ϕ,

$$\mathcal{L}(\theta,\phi) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{x}|\mathbf{z},S)] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p_\theta(\mathbf{z})] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log q_\phi(\mathbf{z} \mid \mathbf{x}, S)]. \tag{6.13}$$

The last two terms in Equation 6.13 equate to the KL divergence between $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ and $p_\theta(\mathbf{z})$. For certain choices of distribution family for $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ and $p_\theta(\mathbf{z})$ (e.g. both as Gaussian distributions) this KL divergence can be evaluated analytically (Kingma and Welling 2013). In general for arbitrary choices of distribution none of the expectations can be evaluated exactly, but we can form a Monte Carlo estimate of the loss function using K samples drawn from $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ (Rezende and Mohamed 2015),

$$\mathcal{L}(\theta,\phi) \approx -\frac{1}{K}\left[\sum_{k=1}^{K} \log p(\mathbf{x} \mid \mathbf{z}_k, S) + \log p_\theta(\mathbf{z}_k) - \log q_\phi(\mathbf{z}_k \mid \mathbf{x}, S)\right]. \tag{6.14}$$

Vandegar et al. 2021 looked at a similar problem in the context of simulation-based inference, in work published subsequently to the initial version of our work (Dockhorn et al. 2020). They could not get variational inference to work well for their applications, and instead made use of approaches that computed the marginal likelihood as an expectation under the prior $p_\theta(\mathbf{z})$, using Monte Carlo estimates with samples drawn from $p_\theta(\mathbf{z})$. These approaches will not generally scale well as the dimensionality of $\mathbf{z}$ increases. The estimates will generally be high-variance as most of the samples of $\mathbf{z}$ will be far from the data $\mathbf{x}$.

6.3.1.1 Approximate Posterior Choice

Having determined an objective for fitting the prior to maximise the log-marginal likelihood by making use of an approximate posterior, we need to select an appropriate distribution $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$. A simple choice would be to use a Gaussian and optimise its mean μ and covariance Σ to maximise the ELBO. There are two drawbacks to this approach

1. We would need to optimise a different set of parameters μ_i and Σ_i for every observed datapoint $\mathbf{x}_i$ and S_i.
2. The posterior $p(\mathbf{z} \mid \mathbf{x}, S)$ may not be close to Gaussian, limiting the ability of $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ to approximate it.

The first issue can be solved by making use of *amortized variational inference* (Rezende, Mohamed, and Wierstra 2014; Kingma and Welling 2013). Instead of learning separate parameters for every datapoint, we will fit a neural network that takes the noisy observation $\mathbf{x}_i$ and the noise statistics S_i as inputs and outputs the parameters of the distribution to use as an approximate posterior. Thus the cost of finding an appropriate posterior for each datapoint is *amortized* through (spread across) the training of the neural network, and allows our approximate posterior to generalise to future datapoints without the need for refitting.

We can solve the second issue by using more flexible distributions (Rezende and Mohamed 2015; Kingma, Salimans, et al. 2016). In this application we choose to use normalising flows as a more expressive posterior approximation, as we are already using them to model the prior $p_\theta(\mathbf{z})$ and so there is not much additional complexity required to implement them. If we use a conditional normalizing flow with observed noisy data $\mathbf{x}_i$ and noise statistics S_i as the conditioning variables, then our approximate posterior will also be amortized.

The choice of which form of normalising flow to use matters. For this work we use autoregressive flows instead of coupling flows as they are generally more expressive and the dimensionality of our problems are relatively low (Papamakarios, Nalisnick, et al. 2021). For autoregressive flows the invertible transformation at each step of the flow is D times more expensive to evaluate in one direction compared to the other direction as discussed in Section 2.4.2. Therefore one of sampling from the distribution or log-probability evaluation will be D times more expensive than the other.

During training, we need to sample from our approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ and only evaluate the log-probability of those same samples. Thus it makes sense to use flows following the form of the inverse autoregressive flow (IAF), where the autoregressive transformation is constructed from the base distribution towards the target distribution (Kingma, Salimans, et al. 2016). Drawing a sample has D times less complexity than if we were to construct the autoregressive transformation in the other direction. Evaluating the log-probability of each drawn sample can be done cheaply by caching the input and output of each transformation step, to avoid explicitly inverting the transformation.

Conversely for the prior $p_\theta(\mathbf{z})$ we do not need to sample from it during training, but we do need to evaluate the log-probability of samples from the approximate posterior. Therefore we use flows following the form of the masked autoregressive flow (MAF), where the autoregressive transformation is constructed from the target distribution towards the base distribution (Papamakarios, Pavlakou, and Murray 2017). Evaluating the log-probability for any sample is therefore D times less complex than if we constructed the flow in the other direction.

For the posterior flow, one potentially useful improvement is to use a base distribution $\pi(\mathbf{z})$ proportional to the noise distribution $p(\mathbf{x} \mid \mathbf{z}, S)$ instead of a fixed distribution. This allows the approximate posterior to exploit the fact that we do have some idea of where the value of $\mathbf{z}$ lies even without knowing the prior. We have found this to improve the stability of training in cases where the scale of the noise distribution is relatively small compared to the distribution of the latent data $\mathbf{z}$.

6.3.1.2 Gradients

We also need to be able to take gradients of the loss function with respect to the parameters. For the parameters of the prior θ this is relatively straightforward, as the expectations in the loss function are with respect to a distribution which is not a function of θ, so we can move the gradient operation inside the expectations,

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta, \phi) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{x}|\mathbf{z}, S) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}, S)], &(6.15)\\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}\left[\nabla_\theta \left(\log p(\mathbf{x}|\mathbf{z}, S) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}, S)\right)\right]. &(6.16)\end{aligned}$$

The gradients can be computed using automatic differentiation, and a Monte Carlo estimate of the expectation can be made as in Equation 6.14.

Taking gradients of the loss with respect to ϕ is not so straightforward, as the expectations are under $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ which is a function of ϕ. One option is to use the score function gradient estimator (Williams 1992), but this typically produces high-variance gradient estimates (Mohamed et al. 2020).

An alternative approach is the *reparametrisation trick*, popularised in the context of variational inference by Kingma and Welling (2013) and Rezende, Mohamed, and Wierstra (2014). This approach makes of the fact that many of the approximate posteriors we would wish to use can be written in terms of a differentiable transform $t(\phi, \epsilon)$ of samples ϵ from some fixed distribution $p(\epsilon)$ by the parameters ϕ,

$$\mathbf{z} = t(\epsilon, \phi), \quad \epsilon \sim p(\epsilon) \tag{6.17}$$

This is true in the case of normalising flows by their definition, and also for many families of distributions via a location-scale transformation, such as a Gaussian. If we can perform this reparametrisation on our approximate posterior, then we can move the gradient operation inside the expectation by rewriting them as expectations under the fixed base distribution,

$$\nabla_\phi \mathcal{L}(\theta, \phi) = \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},S)}[\log p(\mathbf{x}|\mathbf{z}, S) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}, S)], \tag{6.18}$$

$$= \nabla_\phi \mathbb{E}_{p(\epsilon)}[\log p(\mathbf{x}|t(\epsilon, \phi), S) + \log p_\theta(t(\epsilon, \phi)) - \log q_\phi(t(\epsilon, \phi) \mid \mathbf{x}, S)] \tag{6.19}$$

$$= \mathbb{E}_{p(\epsilon)}[\nabla_\phi \left(\log p(\mathbf{x}|t(\epsilon, \phi), S) + \log p_\theta(t(\epsilon, \phi)) - \log q_\phi(t(\epsilon, \phi) \mid \mathbf{x}, S)\right)], \tag{6.20}$$

$$\approx \sum_{k=1}^{K} [\nabla_\phi \left(\log p(\mathbf{x}|t(\epsilon_k, \phi), S) + \log p_\theta(t(\epsilon_k, \phi)) - \log q_\phi(t(\epsilon_k, \phi) \mid \mathbf{x}, S)\right)], \quad \epsilon_k \sim p(\epsilon). \tag{6.21}$$

These gradients are easy to implement with automatic differentiation, and often have a low enough variance that even estimates with $K = 1$ samples are useable for parameter updates (Kingma and Welling 2013). Note that we cannot apply the reparametrisation trick directly if we want to use a GMM as a density estimator, as the choice of mixture component results in a non-differentiable transformation. We can work around this by marginalising the choice of mixture component out of the density estimator.

6.3.2 Importance Weighting

Having fitted a model using the ELBO as an objective, we would like to be able to evaluate the log marginal-likelihood $\log p(\mathbf{x})$ in order to judge how well it has fitted the data and compare it to the exact GMM approach. If the approximate posterior manages to match the true posterior exactly, then the KL divergence between them will be zero and so the ELBO will equal $\log p(\mathbf{x})$. In practice this will not happen precisely, so we need a different method to estimate the log marginal-likelihood.

The importance weighted autoencoder (IWAE) proposes a bound using importance weighting on $\log p(\mathbf{x})$ which is tighter than the ELBO, and will asymptotically approach $\log p(\mathbf{x})$ as $K \to \infty$ (Burda, Grosse, and Salakhutdinov 2016). Adapting it for our case gives a lower bound on $\log p(\mathbf{x})$,

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}_1,\ldots,\mathbf{z}_K \sim q_\phi(\mathbf{z} \mid \mathbf{x}, S)} \left[\log \frac{1}{K} \sum_{k=1}^{K} \left(\frac{p(\mathbf{x} \mid \mathbf{z}_k, S) p_\theta(\mathbf{z}_k)}{q_\phi(\mathbf{z}_k \mid \mathbf{x}, S)} \right) \right], \tag{6.22}$$

which can be evaluated easily using a sufficiently large value of K to estimate the log marginal-likelihood at validation time.

6.4 Experiments

6.4.1 Synthetic Data

To validate our approach, we first try fitting models to synthetic data. This has the advantage of allowing us to evaluate model performance on the underlying latent data $\mathbf{z}$, which would not normally be available, as well as on $\mathbf{x}$.

6.4.1.1 Mixture Data

First, we reuse the simple 2D 2-component Gaussian mixture model from Section 5.4.1,

$$\alpha = 0.5, \tag{6.23}$$

$$\mu_0 = \mu_1 = [0, 0], \tag{6.24}$$

$$C_0 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 0.1 \end{bmatrix}, \tag{6.25}$$

$$C_1 = \begin{bmatrix} 0.1 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \tag{6.26}$$

$$k_i \sim \text{Bernoulli}(\alpha), \tag{6.27}$$

$$z_i \sim \mathcal{N}(m_{k_i}, C_{k_i}). \tag{6.28}$$

Figure 5.1a shows a corner plot of samples drawn from this mixture model.

We use the same noise scheme by generating a diagonal covariance S_i for each datapoint z_i, sampling a value s_{id} for each entry on the diagonal from a unit scale log-normal distribution, then multiplying it by a noise-scale factor η before squaring to get the variance. A noise vector ϵ_i was then sampled from a multivariate normal with zero mean and S_i covariance,

$$s_{id} \sim \text{LogNormal}(0, 1), \quad d \in \{0, 1\} \tag{6.29}$$

$$S_i = \begin{bmatrix} (\eta s_{i0})^2 & 0 \\ 0 & (\eta s_{i1})^2 \end{bmatrix}, \tag{6.30}$$

$$\epsilon_i \sim \mathcal{N}(0, S_i). \tag{6.31}$$

The use of the noise-scale η allows us to control the relative size of the noise compared to the latent data $\mathbf{z}$. Figure 5.1b shows a corner plot of the observed data $\mathbf{x}$ drawn using this scheme and a noise-scale factor η of 0.1

Two million samples were generated using this scheme with a noise-scale factor η of 0.1 and divided into training and validation sets. We used an unconditional normalising flow $p_\theta(\mathbf{z})$ using the autoregressive variant of the neural spline flow (NSF) to model the prior (Durkan, Bekasov, et al. 2019). For the posterior, we used a conditional normalising flow $q_\phi(\mathbf{z} \mid \mathbf{x}, S)$ using an affine masked autoregressive flow (MAF) (Papamakarios, Pavlakou, and

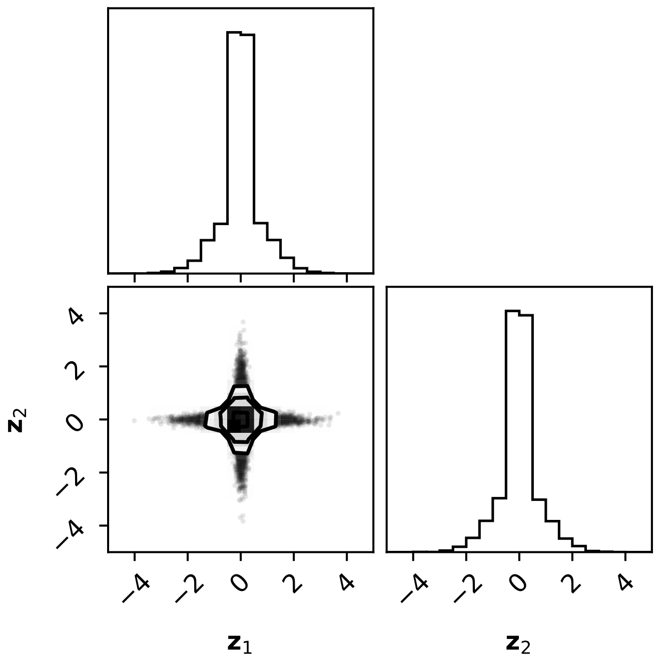
Murray 2017). We choose the direction of each flow to maximise efficiency during training as discussed in Section 6.3.1.1. We used the noise distribution itself as the base distribution for the posterior flow. Both flows were fitted jointly using the ELBO as a loss function for SGD.

As a comparison, a two component GMM was fitted to the same data using the Extreme-Deconvolution method with SGD from Chapter 5. We also fitted a two component GMM using variational inference, drawing samples from the exact posterior to compute the ELBO. The marginal log-likelihood $\log p(\mathbf{x})$ was estimated for the models fitted with variational inference using the importance weighted bound given by Equation 6.22. Full experimental details are available in Appendix B.1.

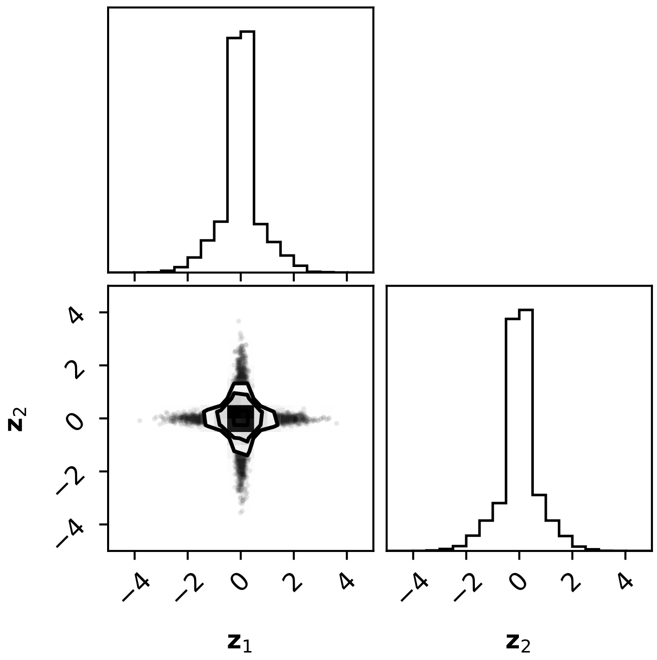
Table 6.1 shows the validation log-likelihoods averaged over datapoints for both $\mathbf{x}$ and $\mathbf{z}$, as well as the ELBO for the models fitted with variational inference. The GMM fitted exactly does best on both $\log p(\mathbf{x})$ and $\log p(\mathbf{z})$, which is unsurprising as it is the exact form of model used to generate the data. The flow-based model is close to the GMM in terms of both $\log p(\mathbf{x})$ and $\log p(\mathbf{z})$, so the extra flexibility and need to use variational inference did little harm.

Figures 6.1a and Figures 6.1b show samples from the fitted exact GMM and flow priors respectively. They appear to be identical, and match the plot of latent training samples from Figure 5.1a.

The GMM fitted using variational inference with the exact posterior has done slightly worse in terms of $\log p(\mathbf{x})$, and much worse in terms of $\log p(\mathbf{z})$. This result happens even if the GMM is initialised with the ground-truth parameters, as the higher variance of the gradients of the ELBO with respect to the parameters pushes the GMM off the ground-truth setting and into a local optima. This also demonstrates the problem where small differences in $\log p(\mathbf{x})$ can correspond to much larger differences in $\log p(\mathbf{z})$ first discussed in Chapter 5.



(a) Corner plot of latent $\mathbf{z}$ samples from the fitted GMM prior.



(b) Corner plot of latent $\mathbf{z}$ samples from the fitted flow prior.

Figure 6.1: Samples from the GMM and flow priors fitted to the mixture data problem from Section 6.4.1.1. Both produce very similar histograms.

Table 6.1: Per-datapoint average validation log-likelihoods for the synthetic data experiment using a mixture distribution. Error bars indicate the standard deviation across five independent runs. The flow gets reasonably close to the GMM in terms of modelling the noisy observed data and the latent observed data, whilst the GMM fitted with variational inference does substantially worse in terms of the latent data.

	ELBO	$\log p(\mathbf{x})$	$\log p(\mathbf{z})$
GMM (Exact)	–	-1.460 ± 0.000	-1.107 ± 0.000
GMM (VI)	-1.493 ± 0.000	-1.493 ± 0.000	-2.155 ± 0.000
Flow	-1.465 ± 0.001	-1.462 ± 0.001	-1.112 ± 0.001

6.4.1.2 Half Normal Data

The previous section has shown that variational inference with a normalising flow-based model is a viable approach for performing extreme deconvolution. However, it did not demonstrate any actual advantage over the GMM fitted exactly, as the GMM is able to match the data generation process exactly. In this section we instead use data generated from a distribution $p(\mathbf{z})$ where we would expect a flow to do better at density estimation.

To generate synthetic samples of $\mathbf{z}$, we draw each element z_{id} from a half-normal distribution with zero mean and unit scale and $D = 10$.

$$z_{id} \sim \text{HalfNormal}(0, 1). \tag{6.32}$$

Figure 6.2a shows a corner plot of the first two dimensions of $\mathbf{z}$ drawn using this scheme, showing the hard cut-off at zero. GMMs often struggle when trying to fit hard cut-offs in a distribution, requiring the number of components to increase exponentially with the number of dimensions. By contrast normalising flows do a better job of fitting to hard cut-offs. In many problems a restricted support would best be dealt with by reparametrising the underlying model. However for the exact GMM approach this is not an option, as any reparametrisation of either $\mathbf{z}$ or $\mathbf{x}$ removes the ability to evaluate the marginal log-likelihood $\log p(\mathbf{x})$ analytically by removing the requirement for the prior to be a GMM and the noise to be Gaussian.

We use the same noise-scheme as in Section 6.4.1.1, and figure 6.2b shows

Table 6.2: Per-datapoint average validation log-likelihoods for the synthetic data experiment using a $10D$ half-normal distribution. Error bars indicate the standard deviation across five independent runs. The flows are significantly better at modelling the observed noisy data, and correspondingly do a better job of modelling the latent noise-free data.

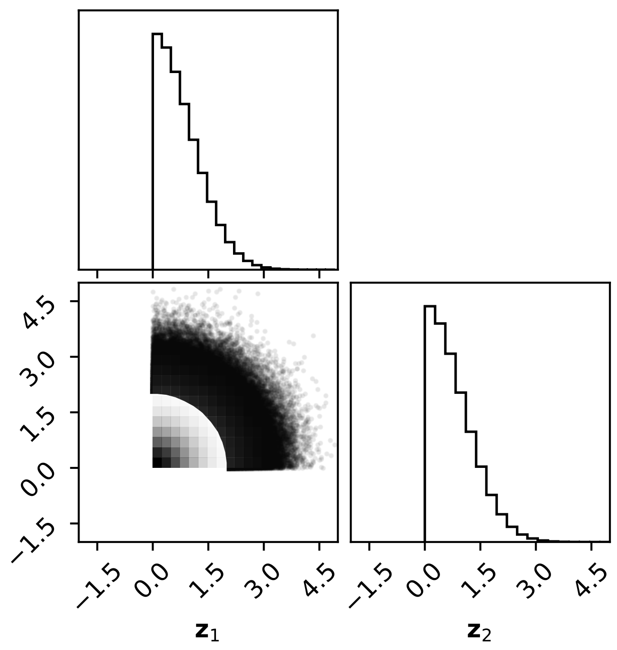
	ELBO	$\log p(\mathbf{x})$	$\log p(\mathbf{z})$
GMM (Exact)	–	-8.484 ± 0.004	-8.122 ± 0.004
GMM (VI, Exact)	-8.464 ± 0.001	-8.449 ± 0.001	-8.010 ± 0.001
GMM (VI, Flow)	-8.464 ± 0.004	-8.447 ± 0.003	-8.003 ± 0.001
Flow	-8.182 ± 0.001	-8.169 ± 0.001	-7.506 ± 0.001

the same samples with the noise added to produce $\mathbf{x}$. The noise makes it hard to determine exactly where the cut-off lies.

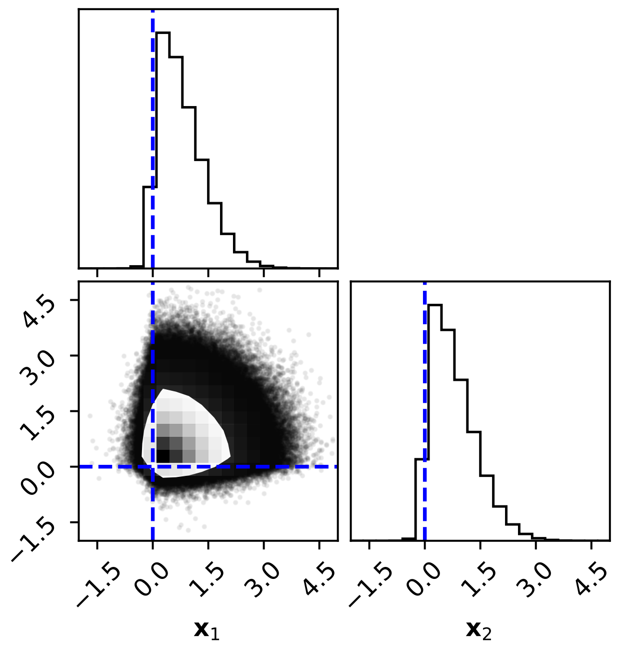
We compare the ability of a GMM and a flow-based model to fit the data using the same experimental setup as before in Section 6.4.1.1. We set the number of components for the comparison GMM to $K = 128$. As well as fitting the GMM using the exact marginal log-likelihood, we also fit it using variational inference using the exact posterior as before, and also with the flow-based posterior on its own. Appendix B.1 contains full experimental details.

Table 6.2 shows the validation log-likelihoods averaged over datapoints for both $\mathbf{x}$ and $\mathbf{z}$, as well as the ELBO for the models fitted with variational inference. The flow-based model is clearly better at fitting the underlying density, with much larger values for both the marginal log-likelihood and the latent likelihood compared to the GMM based models. In this case, fitting the GMM with variational inference actually results in slightly better values compared to fitting it exactly, so it could be that the additional noise in the gradient estimates can help avoid local maxima in some cases.

Figures 6.3a and 6.3b show samples from the fitted prior for the exact GMM and flow-based model respectively. We can see in Figure 6.3b that the flow-based model has done a better job of identifying the hard cut-off below zero. In Figure 6.3a the GMM can be seen to leak more probability mass below zero.

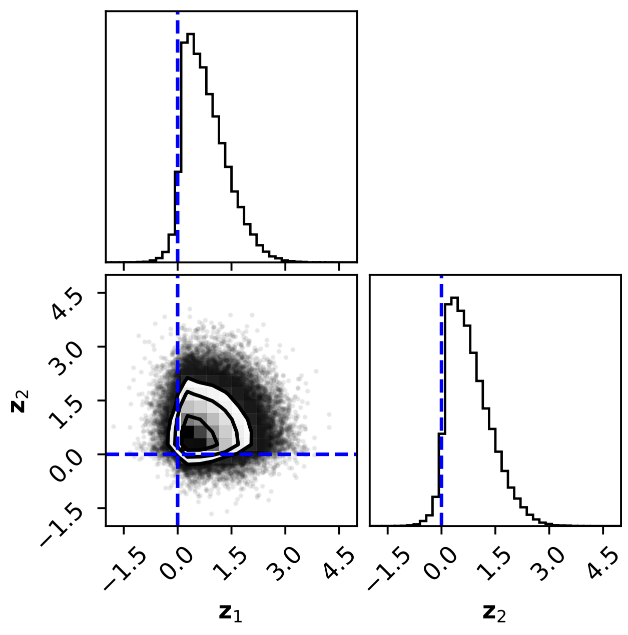


(a) Corner plot of latent $\mathbf{z}$ samples from the two component Half-Normal problem.

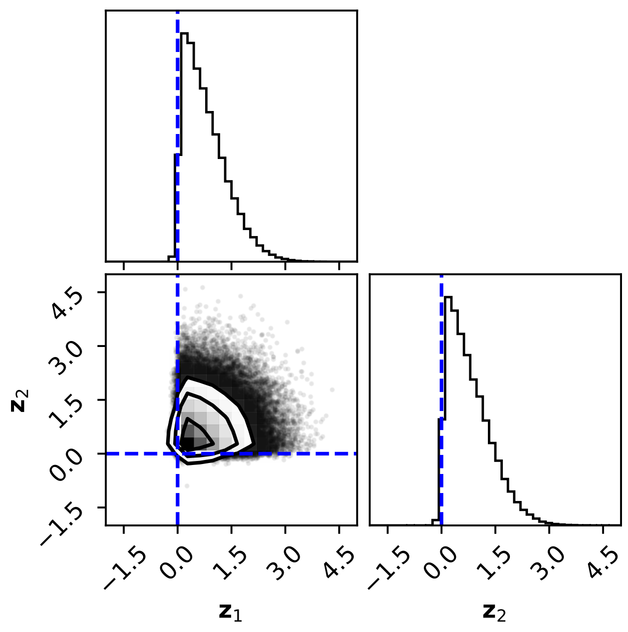


(b) Corner plot of noisy observed $\mathbf{x}$ samples from the Half-Normal problem.

Figure 6.2: The first two dimensions of latent noise-free and observed noisy training samples from the Half-Normal problem. The noise makes it harder to identify exactly where the hard cut-off lies, indicated by the blue dashed line.



(a) Corner plot of latent $\mathbf{z}$ samples from the fitted GMM prior.



(b) Corner plot of latent $\mathbf{z}$ samples from the fitted flow prior.

Figure 6.3: Samples from the GMM and flow priors fitted to the Half-Normal data problem from Section 6.4.1.2. The flow is clearly better at capturing the hard cut-offs (indicated by the blue dashed lines) compared to the GMM.

6.4.1.3 Non-Gaussian Noise

As a final synthetic experiment, we try fitting the underlying Half Normal data from Section 6.4.1.2, but with Laplacian instead of Gaussian noise, using a randomly generated scale,

$$s_{id} \sim \mathrm{LogNormal}(0, 1), \quad d \in \{0, 1\} \tag{6.33}$$

$$\epsilon_{id} \sim \mathrm{Laplace}(0, \eta s_{id}), \tag{6.34}$$

where η is again a noise-scale factor we set to 0.1 as before.

As the noise is now non-Gaussian, we cannot fit a GMM to the data by exact fitting of the marginal log-likelihood or by variational inference using the exact posterior, as the posterior is now intractable. We are therefore restricted to using the GMM fitted with variational inference and a flow-posterior, and the fully flow-based model, using the same experimental setup as in Section 6.4.1.2.

Table 6.3 shows the validation ELBO and log-likelihoods averaged over datapoints for both $\mathbf{x}$ and $\mathbf{z}$. Again the flow-based prior does a better job than the GMM in terms of both $\log p(\mathbf{x})$ and $\log p(\mathbf{z})$, and the results for $\log p(\mathbf{z})$ are fairly close to those in Table 6.2. This demonstrates that our approach can work just as well with non-Gaussian noise as Gaussian noise.

Table 6.3: Per-datapoint average validation log-likelihoods for the synthetic data experiment using non-Gaussian noise. Error bars indicate the standard deviation across five independent runs. As with the previous experiments using Gaussian noise, the flows are better at modelling the hard cut-off in the latent data.

	ELBO	$\log p(\mathbf{x})$	$\log p(\mathbf{z})$
GMM (VI, Flow)	-10.277 ± 0.066	-10.214 ± 0.064	-8.119 ± 0.023
Flow	-9.861 ± 0.009	-9.809 ± 0.009	-7.585 ± 0.001

6.4.2 Astrometric Gaia Data

Having shown with the synthetic data experiments that normalising flows fitted with variational inference can do a better job of deconvolving noisy data if the flow is able to fit the underlying noise-free density better, we now try

Table 6.4: Per-datapoint average validation results for the synthetic data experiment using the astrometric Gaia dataset. The GMM is slightly better at modelling the noisy data than the flow.

	ELBO	$\log p(\mathbf{x})$
GMM	–	-15.942 ± 0.001
Flow	-16.053 ± 0.027	-16.029 ± 0.023

our approach on real data. We reuse the Gaia dataset from Chapter 5. The dataset consists of two million samples from the Gaia Data Release 3 (DR3) catalogue (The Gaia Collaboration 2022). Each sample consist of five measurements of a particular star's position and movements, referred to as astrometric measurements, and has a full-covariance Gaussian noise level associated with it. We follow Anderson et al. (2018) in assuming that the noise covariances are independent of the latent noise free values even though this is not strictly true, noting that it may bias our results as discussed in Section 5.4.3.2.

We fit a GMM using the exact optimisation method as in the synthetic experiments, with $K = 128$ components. We use the same flow setup as for the synthetic experiments, but with modified hyperparameters. Full experimental details are provided in Appendix B.2.

Table 6.4 reports the validation log-likelihoods averaged over datapoints for $\mathbf{x}$, as well as the ELBO for the flow-based model. As the noise-free values of $\mathbf{z}$ underlying the measurements are not available in this case, we cannot report $\log p(\mathbf{z})$. The GMM has a slightly better result than the flow, suggesting that the limited dimensionality of the dataset and the relatively smoothness of the data prevents the flow from gaining much of an advantage.

Figures 6.4 and 6.5 show samples from the fitted prior from the GMM and flow respectively. Both appear fairly similar. Looking at the density plot showing the marginal distribution of the first and second dimensions, we can see that both have recovered the characteristic structure of the Milky Way.

Figure 6.6 shows histograms showing the marginal distribution of the parallax samples ω (corresponding to the third dimension) from the dataset and samples from both the GMM and flow. The parallax samples are typically the noisiest measurement in the training dataset, and should theoretically be constrained to be greater than zero. In practice the dataset contains negative

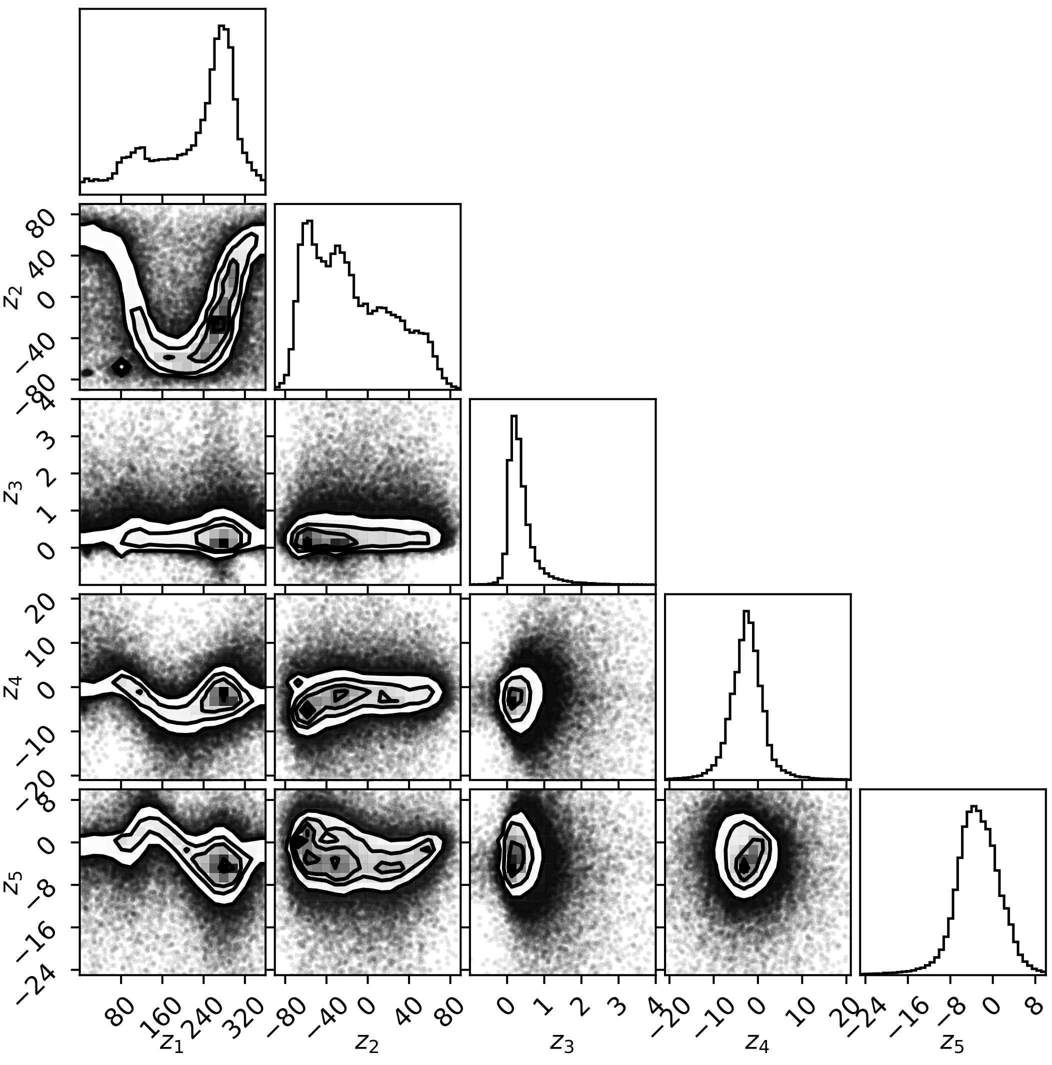


Figure 6.4: Corner plot of latent $\mathbf{z}$ samples from the GMM prior fitted to the noisy Gaia data by direct optimisation of the marginal likelihood.

parallax values as a result of systematic errors, and the noise model is not strictly Gaussian for some of them (Babusiaux, Fabricius, et al. 2022). Both models have shrunk the distribution compared to the dataset, suggesting that they are actually denoising the data. In particular, both have reduced the weight of the left tail corresponding to non-physical parallax values, with the GMM doing so more than the flow.

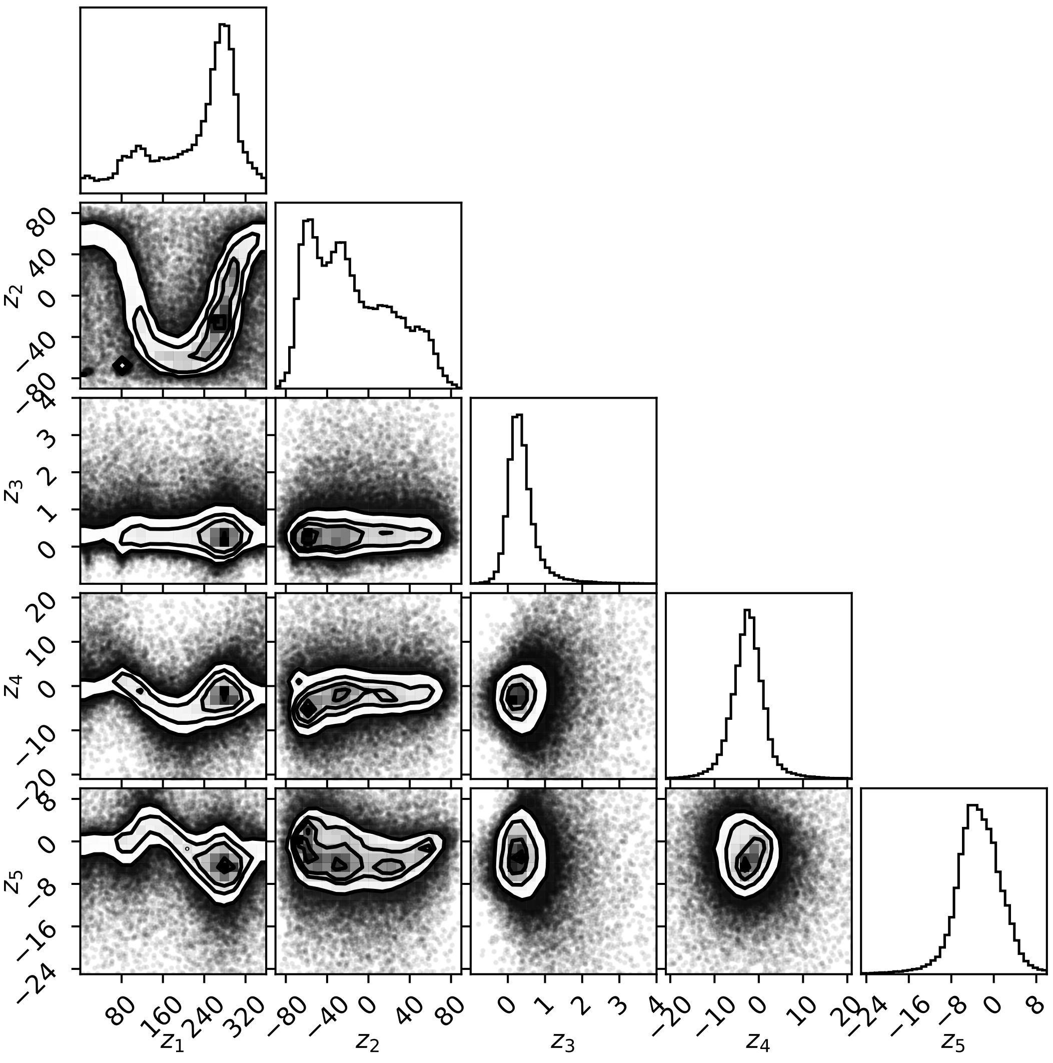


Figure 6.5: Corner plot of latent $\mathbf{z}$ samples from the flow prior fitted to the noisy Gaia data using variational inference. It appears very similar to the samples from the GMM plotted in Figure 6.4.

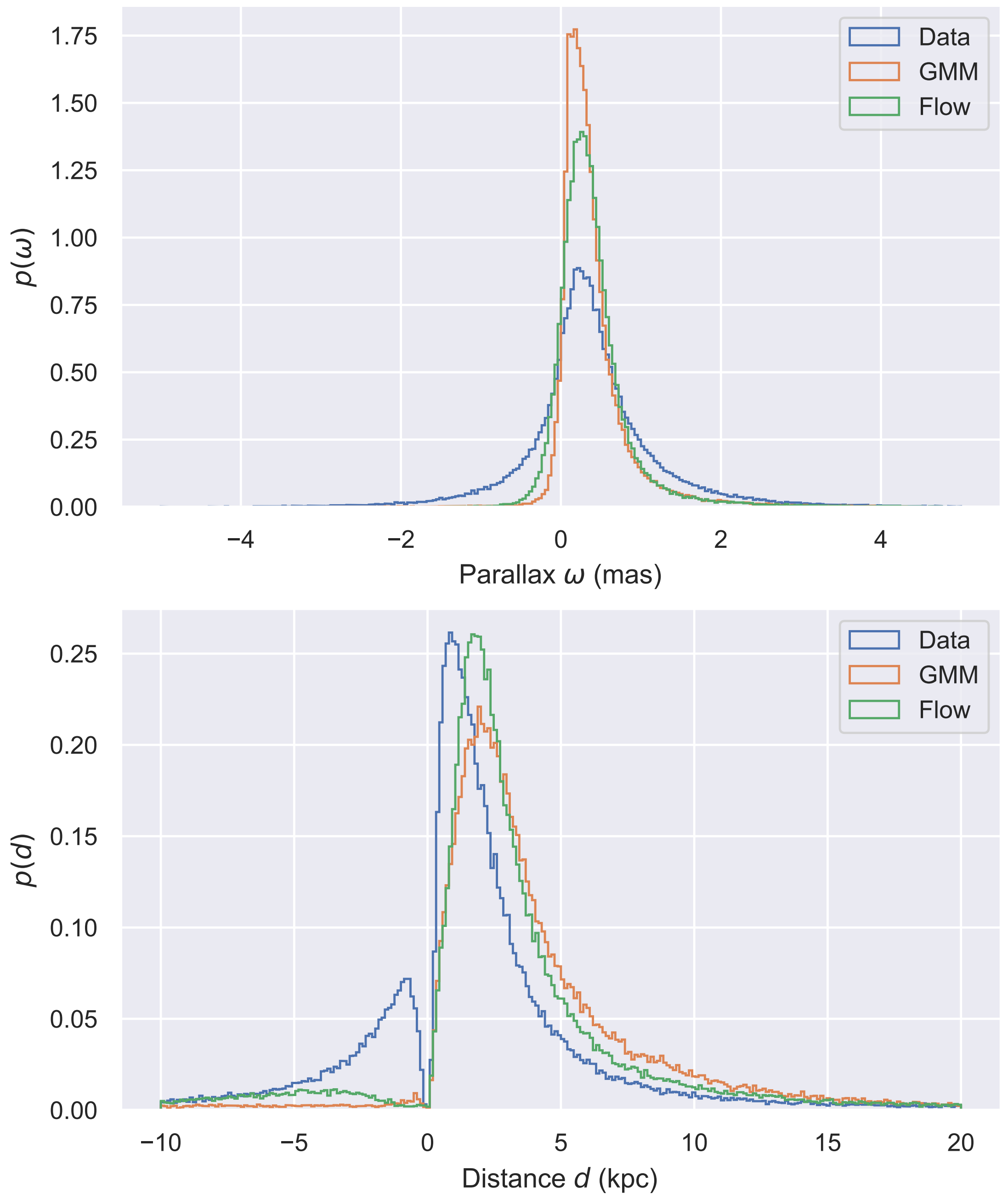


Figure 6.6: Histograms showing estimated marginal distribution of the parallax samples from the Gaia data and both priors fitted to the data. The top row shows a histogram of parallax values, whilst the bottom row shows the same data but inverted, equivalent to distance. Both models have shrunk the distribution as a result of the denoising, especially on the left tail where the parallax has non-physical negative values.

6.4.3 Astrometric and Photometric Gaia Data

We now extend our experiments on the Gaia data to include photometric measurements. We take the dataset from Section 6.4.2 and add three dimensions corresponding to photometric measurements of each source. Each photometric measurement indicates the brightness of the star over a particular range of photon wavelengths (referred to as a passband), and are originally measured in terms of photon flux at the observing sensor (Riello et al. 2021).

As the flux measurements cover a very large range, it is typically to work with them on a log-scale. In astronomy, a magnitude scale is used where the apparent magnitude of a source m is defined as

$$\begin{aligned} m &= t(f), && (6.35) \\ &= -2.5\log_{10}(f) + ZP, && (6.36) \end{aligned}$$

where f is the observed flux at the sensor and ZP is an offset such that a reference source (typically the star Vega) has zero magnitude (Chapter 17, Fraknoi, Morrison, and Wolff 2016; Riello et al. 2021). Subscripts are used to indicate the passband each magnitude and flux correspond to, e.g. m_G and f_G respectively indicate the apparent magnitude and flux in the G-band. The corresponding inverse transformation from magnitude to flux is

$$\begin{aligned} f &= t^{-1}(m), && (6.37) \\ &= f_0 10^{-\frac{2m}{5}}, && (6.38) \end{aligned}$$

where f_0 is the flux of the reference source.

A natural question is whether we should try to do density estimation in the flux or magnitude space. The noise statistics on the photometric measurements are Gaussian in the flux space, and therefore non-Gaussian in magnitude space. Therefore fitting a GMM with exact likelihood evaluation is restricted to the flux space, as it can only work with Gaussian noise. A flow or GMM fitted with variational inference can work with either space in theory. The noise distribution in magnitude space can be evaluated by performing a change-of-variables on the noise distribution in flux space,

$$p(\mathbf{x} \mid \mathbf{z}, S) = \mathcal{N}(t^{-1}(\mathbf{x}) \mid t^{-1}(\mathbf{z}), S) \cdot \left| \frac{d}{d\mathbf{x}} t^{-1}(\mathbf{x}) \right|. \tag{6.39}$$

Table 6.5: Per-datapoint average validation log-likelihoods for the experiment using the astrometric and photometric Gaia dataset. The flow does a better job of modelling the data than the GMM.

	ELBO	$\log p(\mathbf{x})$
GMM	–	-33.622 ± 0.074
Flow	-32.758 ± 0.014	-32.737 ± 0.013

Whilst in theory this can be evaluated as part of our objective function, in practice it results in an unstable fitting procedure, as even reasonable-valued samples from the approximate posterior can result in an overflow when performing the inverse transformation given by Equation 6.38 due to the exponentiation.

As a result we perform density estimation in flux space, which also allows us to directly compare a flow fitted with variational inference to a GMM fitted using the exact marginal likelihood. We use the same setups as in the experiments in Section 6.4.2, but with the number of components of the GMM increased to $K = 256$. Full experimental details are available in Appendix B.2. Table 6.5 reports the validation log-likelihoods averaged over datapoints for x, as well as the ELBO for the flow-based model. This time the flow produces better log-likelihood values than the GMM, which validates our assumption that flows could be better for some datasets. As log-likelihood values are in themselves not inherently interesting, we now consider two example applications for the flow-based model using our fitted prior and our approximate posterior respectively. We use these applications to validate that the flow-based model is behaving reasonably.

6.4.3.1 Colour Magnitude Diagram

An interesting application of the prior is to use it to plot a colour-magnitude diagram (CMD, Chapter 18, Fraknoi, Morrison, and Wolff 2016; Babusiaux, van Leeuwen, et al. 2018). There is a strong relationship between the colour of a star and its absolute magnitude. The absolute magnitude of a star is defined as being equivalent to the apparent magnitude if the star was observed at a distance of 10 parsecs, corresponding to a parallax of 100 milliarcseconds. The absolute magnitude M can be computed from an observed apparent magnitude

m if the parallax ω is known via the relationship

$$M = m + 5\log_{10}(\omega) - 10, \tag{6.40}$$

with ω having units of milliarcseconds (Babusiaux, van Leeuwen, et al. 2018). The absolute magnitude is strongly dependent on the distance d computed from the parallax.

The colour is defined as the difference between magnitudes in different passbands, or equivalently the logarithm of the ratio of fluxes in different passbands. A typical colour used with the photometric data from the Gaia catalogue is the difference between apparent magnitudes from the BP and RP passbands (Babusiaux, van Leeuwen, et al. 2018),

$$G_{BP} - G_{RP} = m_{G_{BP}} - m_{G_{RP}} \tag{6.41}$$

Colour does not strongly depend on distance. If we were to compute the colour using absolute magnitudes M instead of apparent magnitudes m, the result would be the same because the distance correction terms from Equation 6.40 would cancel out.

If we know the relationship between colour and absolute magnitude via a CMD, then we can infer the distance to stars for which we do not have parallax measurements by inferring the absolute magnitude from the observed colour. We can then compare the inferred absolute magnitude to the observed apparent magnitude to derive the distance (Chapter 19, Fraknoi, Morrison, and Wolff 2016). Construction of a CMD can be done directly from observed data such as the Gaia catalogue, but if the measurements are noisy we risk producing an inaccurate CMD.

Instead, we use samples from our fitted prior $p(\mathbf{z})$ to construct a CMD, which allows us to correct for the noisy observations, and compare it to a CMD constructed directly from the observed data. In order to construct the diagrams from both prior and data samples, we need to correct the apparent magnitudes and colours to account for interstellar dust between the star and telescope, which has the effect of diminishing the apparent magnitude and biasing the colour to the red end of the spectrum. We use the Bayestar 2017 dustmap (Green, Schlafly, Finkbeiner, et al. 2018), which takes the 3D position of each sample as an input, and use the median output as a correction factor.[1] We then use the

[1]There is a newer version of the Bayestar dustmap (Green, Schlafly, Zucker, et al. 2019), but

values derived by Cranmer, Galvez, et al. (2019) to convert this correction factor into factors for each Gaia passband.[2]

Figure 6.7 shows a normalised 2D histogram of samples from both the prior and data after dust correction, along with the marginalised histograms over colour and magnitude, making the latter a number count plot. We can see that the colour values are not significantly changed by the deconvolution process, as the colour measurements in this dataset typically have small noise values. The magnitude values have changed, as they depend strongly on the parallax measurements which can be noisy.

6.4.3.2 M67 Distance Estimates

To demonstrate an example application of our approximate posterior, we follow one of the examples from Anderson et al. 2018 and denoise the noisy measurements of stars in the M67 cluster to improve parallax measurements. Accurate distance estimates using other methods have been made, allowing it to serve as a useful test-case (Yakut et al. 2009).

We selected all sources in the DR3 catalogue within a 0.3 degree radius of the point with right-ascension of 132.8° and a declination of 11.8°. We do not filter on the absolute parallax, so our selection will include stars outside of the cluster. However, as successive data releases from Gaia have improved the noise on the parallax measurements substantially, we filter to include only stars with a signal-to-noise ratio of less than 15, in order to make the problem more challenging. This results in a dataset with 1479 samples.

We input each data sample to our approximate posterior and draw 1000 samples from it. We also draw 1000 samples from the noise distribution defined by the data sample. Figure 6.8 shows a histogram of the distance (inverse parallax) combined over all samples for both posterior and data, along with an indication of the previous distance estimate. We can see that the posterior samples have tightened the estimate towards the previous measurements compared to the noisy data. Whilst not an exact check on correctness, this validates that our approximate posterior is behaving reasonably. The posterior has also slightly

it makes use of parallaxes from the Gaia catalogue, so to avoid inadvertent double use of the data we only use the earlier version.

[2]Strictly speaking Cranmer, Galvez, et al. (2019) derived conversion values for the Gaia Data Release 2 passbands, which are slightly different from those in the Gaia DR3 catalogue which we used to fit our model, but are close enough for our purposes.

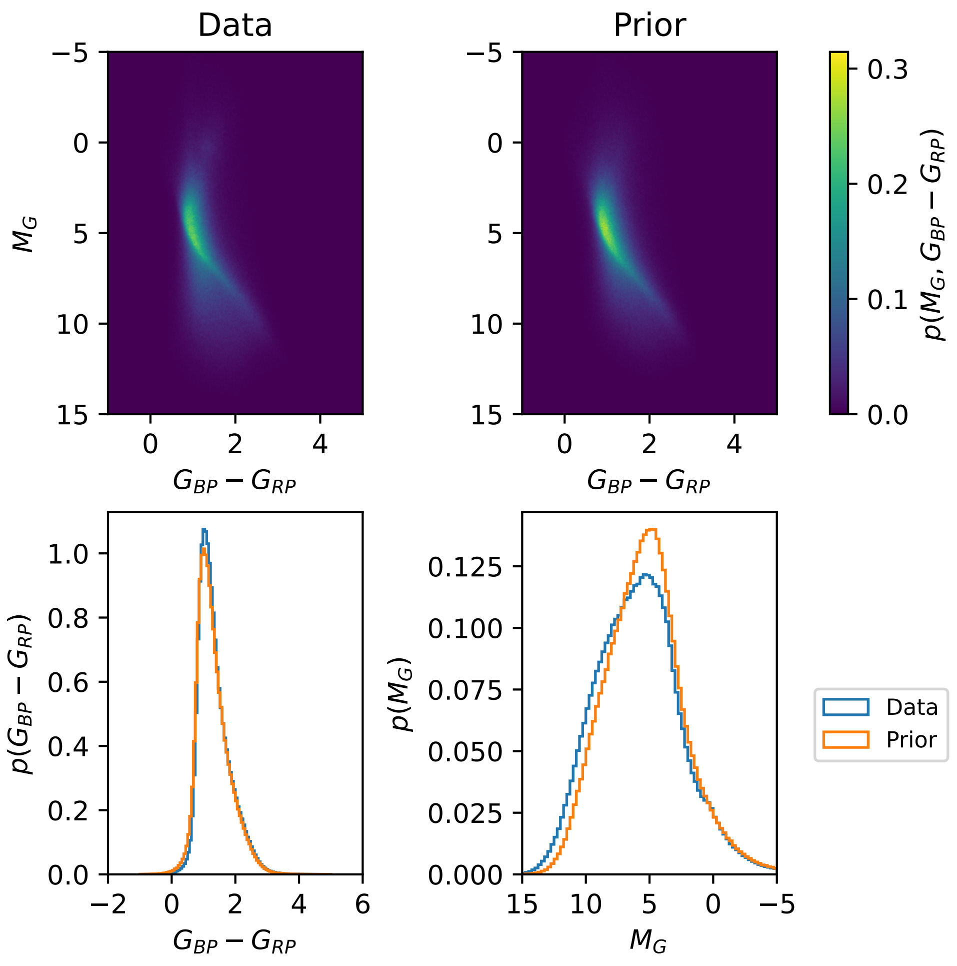


Figure 6.7: Top row: Colour Magnitude Diagrams constructed as 2D histograms directly from the noisy data and from samples from the fitted prior. Bottom row: The same samples marginalised over colour and magnitude respectively. The deconvolution process has not significantly affected the colour, as those measurements had small noise values. The magnitude values have changed, as they depend on noisy parallax measurements.

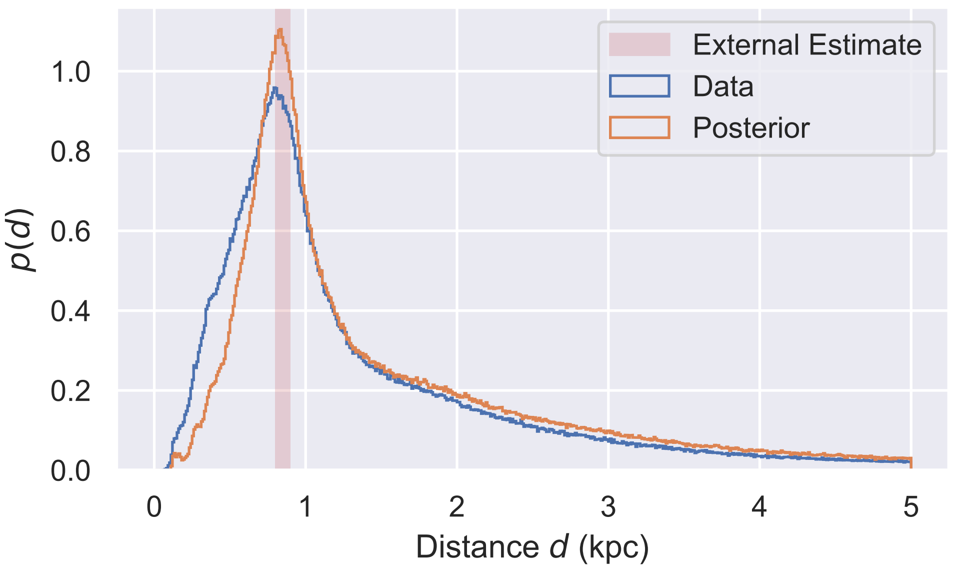


Figure 6.8: Histogram of distances to stars in the M67 cluster, using samples directly from the data, and denoised samples from the approximate flow posterior. The shaded region indicates previous estimates of the distance to M67. The posterior clearly tightens up the Gaia estimate towards the external estimate.

increased the distance estimates for background stars which are not part of the cluster, indicated in the right tail.

6.5 Discussion

Our experiments have shown that our approach to extreme deconvolution using normalizing flows with variational inference is viable. Whilst flow-based approaches do not show an across-the-board improvement compared to GMMs on the tasks we have evaluated them on, they do show better modelling performance for sufficiently complex underlying distributions. We have also demonstrated that using variational inference allows us to work with non-Gaussian noise, something a GMM fitted by exact optimisation of the marginal likelihood cannot do.

One downside of using normalising flows is that we have introduced a large number of free hyperparameters that the user needs to select. The particular parameters we chose are detailed in Appendix B. A more rigorous approach would be to select the hyperparameters through a cross-validation scheme. However, as we noted in Section 5.4.3.1, an overparameterised model can be hard to detect purely through the log marginal-likelihood, which could lead to suboptimal density estimates.

We have quantitatively compared the GMM and flow-based approaches in terms of log-marginal likelihood. Another interesting aspect to the compare them on would be computational cost. Unfortunately, it is difficult to draw strong conclusions about this from our experiments as our approaches have substantially different implementations. Computational cost of the GMM is dominated by the cost of performing $N \times K$ Cholesky decompositions at each training step where N is the size of each minibatch and K is the number of mixture components. For the flow-based models, computation time is mainly spent running the neural networks inside the flow transformations. Any comparison would be heavily dependent on how well the underlying operations have been optimised. However, a rough estimate from the photometric experiment suggests that our implementation of the flow-based approach is approximately three times faster to train compared to our GMM implementation.

6.5.1 Related Work

Our setup using variational inference and an amortized posterior looks similar to that of the variational auto-encoder (VAE), and was indeed partly inspired by it (Kingma and Welling 2013). A key difference is that a VAE is primarily intended for generative modelling, with learnable parameters as part of the likelihood, and the latent space used as an encoding to help achieve this. By contrast in our setup, we have no learnable parameters in the likelihood, and we construct our latent space to be a noise-free representation of the data. Our prior and likelihood (noise) are distributions over the same space, whereas for the VAE the latent space is typically of a much lower dimensionality than the data. The original implementation of the VAE used a fixed prior, but other work has expanded on this with more flexible learnable priors including normalizing flows (Nalisnick, Hertel, and Smyth 2016; Dilokthanakul et al. 2017; Chen et al. 2017).

Anderson et al. 2018 used the existing GMM implementation to fit a CMD using astrometric data from the Gaia DR1 catalogue and photometric data from the 2MASS catalogue (Skrutskie et al. 2006), with the intention of producing a CMD solely from data without making any physical modelling assumptions.

They fit the GMM to a transformation of the CMD in order to ensure the noise was Gaussian distributed, a requirement for the GMM implementation. This approach is not entirely physics-free, as they use current estimates of the parallax inside the training loop to look up dereddening values from a dustmap and correct absolute magnitudes and colours. This also adds potential for bias when fitting the model.

In contrast our approach does not use a dustmap when fitting our model, instead using it only at the end when producing a CMD as a transformation of the features our density estimator is fitted to. We cannot make a direct comparison between our results and theirs, as we use different datasets and modelling approaches, and they do not provide marginal-log likelihood values, but we do draw similar conclusions from the example applications. Our CMD is different to theirs as we use different photometric catalogues, but both approaches generally show a tightened CMD compared to one constructed directly from the data. Both M67 distance estimates tighten the noisy estimates towards the externally-determined value. If we were to drop a normalising flow directly into their modelling setup in place of the GMM it may not result in much improvement, as their GMM is only providing density estimates in two dimensions on a distribution which appears to be smooth.

As previously mentioned the paper by Cranmer, Galvez, et al. (2019) reports the first attempt we are aware of fitting a denoised CMD using a normalizing flow. The model is not formally set up as a deconvolution problem, instead using a heuristic weighting scheme to fit a normalising flow directly to the noisy data. The flow models the CMD directly as a prior, and like Anderson et al. 2018 made use of a dustmap inside the training loop to deredden samples during the fitting process. The paper did not explicitly state how to draw samples from the posterior, although MCMC methods could be used with minimal further assumptions. This approach is hard to compare directly with ours without log-marginal likelihood values, but it also appears to produce a plausible-looking CMD and lower variance estimates of the distance to M67, demonstrating the potential value of normalising flows for this application.

Chapter 7

Conclusion

It is clear that there is still work to be done in meeting the goal of "abstracting away the mechanics of inference and providing self-tuning methods and diagnostics" stated in the introduction. To make inference on the pileup problem possible with Markov chain Monte Carlo (MCMC) methods, we needed to manually derive an exact marginalisation scheme in order to make our chosen MCMC method work. The neural simulation-based inference (SBI) methods we considered are more able to treat the model as black box, but are still not perfect. The SBI methods have many free parameters for the user to choose, and it is not obvious how we should be guided when selecting them. They also lack self-diagnostics which make it hard to be sure they are correct even if they run without errors.

We highlighted potential issues that might be encountered when fitting models with the extreme deconvolution method. We noted that the log marginal-likelihood could be misleading when judging a fitted model's performance, with no straightforward way to detect when this was happening. We also showed that the deconvolution method will produce incorrect results if the assumption of noise statistics being independent of the latent-noise free value is violated. The use of normalising flows in-place of Gaussian mixture models (GMMs) results in a large increase in the number of free hyperparameters the user is required to tune.

Even with these issues remaining, we have still made progress in tackling the problems we set out to solve. We have shown that it is possible to do Bayesian inference with spectral models involving pileup, and that both the MCMC and SBI methods we have proposed can be well-calibrated. The SBI methods also have the advantage of being able to scale to larger problems, and by applying the potential scale reduction factor diagnostic to them we can detect some problems without having to resort to expensive calibration checks.

By leveraging techniques for fitting large-scale machine learning models, we can make it practical to use the extreme deconvolution method with the extremely large catalogues of data produced by the Gaia mission. Adapting advances in variational inference and density estimation allowed us to make use of normalising flows for density deconvolution, which resulted in better density estimates for some datasets. Together, these improvements have shown that advances in statistical methods and probabilistic machine learning can enable practical Bayesian inference for a wider class of challenging scientific models.

Bibliography

Aloise, Daniel, Amit Deshpande, Pierre Hansen, and Preyas Popat (May 1, 2009). "NP-hardness of Euclidean Sum-of-Squares Clustering". In: *Machine Learning* 75.2, pp. 245–248 (cit. on p. 105).

Anderson, Lauren, David W. Hogg, Boris Leistedt, Adrian M. Price-Whelan, and Jo Bovy (Sept. 2018). "Improving Gaia Parallax Precision with a Data-driven Model of Stars". In: *The Astronomical Journal* 156.4, p. 145 (cit. on pp. 100, 120, 141, 148, 151, 152).

Arnaud, Keith, Randall Smith, and Aneta Siemiginowska, eds. (2011). *Handbook of X-ray Astronomy*. Cambridge Observing Handbooks for Research Astronomers (cit. on p. 51).

Babusiaux, C., C. Fabricius, S. Khanna, T. Muraveva, C. Reylé, F. Spoto, A. Vallenari, X. Luri, F. Arenou, M. A. Alvarez, F. Anders, T. Antoja, E. Balbinot, C. Barache, N. Bauchet, D. Bossini, D. Busonero, T. Cantat-Gaudin, J. M. Carrasco, C. Dafonte, S. Diakite, F. Figueras, A. Garcia-Gutierrez, A. Garofalo, A. Helmi, O. Jimenez-Arranz, C. Jordi, P. Kervella, Z. Kostrzewa-Rutkowska, N. Leclerc, E. Licata, M. Manteiga, A. Masip, M. Monguio, P. Ramos, N. Robichon, A. C. Robin, M. Romero-Gomez, A. Saez, R. Santovena, L. Spina, G. Torralba Elipe, and M. Weiler (June 13, 2022). "Gaia Data Release 3: Catalogue Validation". arXiv: `2206.05989 [astro-ph]` (cit. on pp. 3, 142).

Babusiaux, C., F. van Leeuwen, et al. (Aug. 1, 2018). "Gaia Data Release 2 - Observational Hertzsprung-Russell Diagrams". In: *Astronomy & Astrophysics* 616, A10 (cit. on pp. 146, 147).

Bailer-Jones, C. A. L., J. Rybizki, M. Fouesneau, M. Demleitner, and R. Andrae (Feb. 2021). "Estimating Distances from Parallaxes. V. Geometric and Photogeometric Distances to 1.47 Billion Stars in Gaia Early Data Release 3". In: *The Astronomical Journal* 161.3, p. 147 (cit. on p. 1).

Ballet, J. (Mar. 1, 1999). “Pile-up on X-ray CCD Instruments”. In: *Astronomy and Astrophysics Supplement Series* 135.2 (2), pp. 371–381 (cit. on pp. 2, 51).

Barnard, John, Robert McCulloch, and Xiao-Li Meng (2000). “Modeling Covariance Matrices in Terms of Standard Deviations and Correlations, with Application to Shrinkage”. In: *Statistica Sinica* 10.4, pp. 1281–1311. JSTOR: `24306780` (cit. on p. 123).

Baydin, Atilim Gunes, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind (2018). “Automatic Differentiation in Machine Learning: A Survey”. In: *Journal of Machine Learning Research* 18.153, pp. 1–43 (cit. on pp. 18, 109).

Baydin, Atilim Gunes, Lei Shao, Wahid Bhimji, Lukas Heinrich, Saeid Naderiparizi, Andreas Munk, Jialin Liu, Bradley Gram-Hansen, Gilles Louppe, Lawrence Meadows, Philip Torr, Victor Lee, Kyle Cranmer, Mr. Prabhat, and Frank Wood (2019). “Efficient Probabilistic Inference in the Quest for Physics Beyond the Standard Model”. In: *Advances in Neural Information Processing Systems*. Vol. 32 (cit. on p. 29).

Betancourt, Michael (Apr. 3, 2016a). *Diagnosing Suboptimal Cotangent Disintegrations in Hamiltonian Monte Carlo*. arXiv: `1604.00695 [stat]`. URL: `http://arxiv.org/abs/1604.00695` (visited on 11/29/2022). preprint (cit. on p. 64).

— (Jan. 2, 2016b). *Identifying the Optimal Integration Time in Hamiltonian Monte Carlo*. arXiv: `1601.00225 [stat]`. URL: `http://arxiv.org/abs/1601.00225` (visited on 11/21/2022). preprint (cit. on p. 20).

— (July 15, 2018). *A Conceptual Introduction to Hamiltonian Monte Carlo*. arXiv: `1701.02434 [stat]`. URL: `http://arxiv.org/abs/1701.02434` (visited on 11/29/2022). preprint (cit. on pp. 18, 19, 59, 64, 84).

Bingham, Eli, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman (2019). “Pyro: Deep Universal Probabilistic Programming”. In: *Journal of Machine Learning Research* 20, 28:1–28:6 (cit. on pp. 1, 20, 53, 62, 70, 73, 109, 173).

Bishop, Christopher M. (1994). “Mixture Density Networks”. Technical Report, Aston University (cit. on pp. 29, 70).

Bottou, Léon, Frank E. Curtis, and Jorge Nocedal (Jan. 2018). "Optimization Methods for Large-Scale Machine Learning". In: *SIAM Review* 60.2, pp. 223–311 (cit. on pp. 3, 101, 108).

Bovy, Jo, David W. Hogg, and Sam T. Roweis (June 2011). "Extreme Deconvolution: Inferring Complete Distribution Functions from Noisy, Heterogeneous and Incomplete Observations". In: *The Annals of Applied Statistics* 5 (2B), pp. 1657–1677 (cit. on pp. 3, 100, 104, 105, 107, 109, 110, 122, 125, 171).

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018). *JAX: Composable Transformations of Python+NumPy Programs*. Version 0.2.5 (cit. on p. 62).

Brooks, Stephen P. and Andrew Gelman (Dec. 1, 1998). "General Methods for Monitoring Convergence of Iterative Simulations". In: *Journal of Computational and Graphical Statistics* 7.4, pp. 434–455 (cit. on p. 22).

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (Nov. 7, 2016). *Importance Weighted Autoencoders*. arXiv: `1509.00519 [cs, stat]`. URL: `http://arxiv.org/abs/1509.00519` (visited on 11/29/2022). preprint (cit. on p. 132).

Burnett, Richard, Hong Chen, Mieczysław Szyszkowicz, Neal Fann, Bryan Hubbell, C. Arden Pope, Joshua S. Apte, Michael Brauer, Aaron Cohen, Scott Weichenthal, Jay Coggins, Qian Di, Bert Brunekreef, Joseph Frostad, Stephen S. Lim, Haidong Kan, Katherine D. Walker, George D. Thurston, Richard B. Hayes, Chris C. Lim, Michelle C. Turner, Michael Jerrett, Daniel Krewski, Susan M. Gapstur, W. Ryan Diver, Bart Ostro, Debbie Goldberg, Daniel L. Crouse, Randall V. Martin, Paul Peters, Lauren Pinault, Michael Tjepkema, Aaron van Donkelaar, Paul J. Villeneuve, Anthony B. Miller, Peng Yin, Maigeng Zhou, Lijun Wang, Nicole A. H. Janssen, Marten Marra, Richard W. Atkinson, Hilda Tsang, Thuan Quoc Thach, John B. Cannon, Ryan T. Allen, Jaime E. Hart, Francine Laden, Giulia Cesaroni, Francesco Forastiere, Gudrun Weinmayr, Andrea Jaensch, Gabriele Nagel, Hans Concin, and Joseph V. Spadaro (Sept. 18, 2018). "Global Estimates of Mortality Associated with Long-Term Exposure to Outdoor Fine Particulate Matter". In: *Proceedings of the National Academy of Sciences* 115.38, pp. 9592–9597 (cit. on p. 1).

Cappé, Olivier and Eric Moulines (2009). “On-Line Expectation–Maximization Algorithm for Latent Data Models”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71.3, pp. 593–613 (cit. on p. 106).

Carlin, Bradley P. and Thomas A. Louis (Dec. 1, 2000). “Empirical Bayes: Past, Present and Future”. In: *Journal of the American Statistical Association* 95.452, pp. 1286–1289 (cit. on p. 102).

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell (Jan. 11, 2017). “Stan: A Probabilistic Programming Language”. In: *Journal of Statistical Software* 76, pp. 1–32 (cit. on pp. 1, 53).

Chandra X-ray Science Center (July 18, 2010). *The Chandra ABC Guide to Pileup*. URL: `https://cxc.harvard.edu/ciao/download/doc/pileup_abc.pdf` (visited on 08/22/2023) (cit. on p. 51).

Chen, Xi, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel (Mar. 4, 2017). *Variational Lossy Autoencoder*. arXiv: `1611.02731 [cs, stat]`. URL: `http://arxiv.org/abs/1611.02731` (visited on 10/06/2022). preprint (cit. on p. 151).

Cragg, John G. (1971). “Some Statistical Models for Limited Dependent Variables with Application to the Demand for Durable Goods”. In: *Econometrica* 39.5, pp. 829–844. JSTOR: `1909582` (cit. on p. 94).

Cranmer, Kyle, Johann Brehmer, and Gilles Louppe (Dec. 1, 2020). “The Frontier of Simulation-Based Inference”. In: *Proceedings of the National Academy of Sciences* 117.48, pp. 30055–30062 (cit. on pp. 30, 53).

Cranmer, Miles D., Richard Galvez, Lauren Anderson, David N. Spergel, and Shirley Ho (Aug. 21, 2019). “Modeling the Gaia Color-Magnitude Diagram with Bayesian Neural Flows to Constrain Distance Estimates”. arXiv: `1908.08045 [astro-ph, stat]` (cit. on pp. 124, 148, 152).

Davis, John E. (Nov. 2001). “Event Pileup in Charge-coupled Devices”. In: *The Astrophysical Journal* 562.1, pp. 575–582 (cit. on pp. 52, 80, 82, 83).

Deason, Alis J, Vasily Belokurov, and Jason L Sanders (Dec. 11, 2019). “The Total Stellar Halo Mass of the Milky Way”. In: *Monthly Notices of the Royal Astronomical Society* 490.3, pp. 3426–3439 (cit. on p. 100).

Dempster, A. P., N. M. Laird, and Donald B. Rubin (1977). “Maximum Likelihood from Incomplete Data via the EM Algorithm”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38. JSTOR: `2984875` (cit. on p. 102).

Dilokthanakul, Nat, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan (Jan. 13, 2017). *Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders*. Version 2. arXiv: `1611.02648 [cs, stat]`. URL: `http://arxiv.org/abs/1611.02648` (visited on 10/06/2022). preprint (cit. on p. 151).

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (Feb. 27, 2017). *Density Estimation Using Real NVP*. arXiv: `1605.08803 [cs, stat]`. URL: `http://arxiv.org/abs/1605.08803` (visited on 11/22/2022). preprint (cit. on p. 27).

Dockhorn, Tim, James A. Ritchie, Yaoliang Yu, and Iain Murray (July 13, 2020). “Density Deconvolution with Normalizing Flows”. arXiv: `2006.09396 [cs, stat]` (cit. on pp. 5, 128).

Drineas, P., A. Frieze, R. Kannan, S. Vempala, and V. Vinay (July 1, 2004). “Clustering Large Graphs via the Singular Value Decomposition”. In: *Machine Learning* 56.1, pp. 9–33 (cit. on p. 105).

Duane, Simon, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth (Sept. 3, 1987). “Hybrid Monte Carlo”. In: *Physics Letters B* 195.2, pp. 216–222 (cit. on pp. 2, 18, 44, 50, 59).

Durkan, Conor, Artur Bekasov, Iain Murray, and George Papamakarios (2019). “Neural Spline Flows”. In: *Advances in Neural Information Processing Systems*. Vol. 32 (cit. on pp. 27, 133, 173).

Durkan, Conor, Iain Murray, and George Papamakarios (Nov. 21, 2020). “On Contrastive Learning for Likelihood-free Inference”. In: *Proceedings of the 37th International Conference on Machine Learning*, pp. 2771–2781 (cit. on pp. 36, 72).

Fang, Taotao, Herman L. Marshall, Greg L. Bryan, and Claude R. Canizares (July 1, 2001). “Chandra Observations of Two High-Redshift Quasars”. In: *The Astrophysical Journal* 555.1, p. 356 (cit. on p. 83).

Foreman-Mackey, Daniel (June 8, 2016). “Corner.Py: Scatterplot Matrices in Python”. In: *Journal of Open Source Software* 1.2, p. 24 (cit. on p. 17).

Foreman-Mackey, Daniel, David W. Hogg, Dustin Lang, and Jonathan Goodman (Mar. 2013). "Emcee: The MCMC Hammer". In: *Publications of the Astronomical Society of the Pacific* 125.925, pp. 306–312. arXiv: `1202.3665` (cit. on pp. 1, 44).

Fraknoi, Andrew, David Morrison, and Sidney C. Wolff (Oct. 13, 2016). *Astronomy*. OpenStax. 1194 pp. (cit. on pp. 145–147).

Gabry, Jonah, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman (2019). "Visualization in Bayesian Workflow". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 182.2, pp. 389–402 (cit. on pp. 10, 49).

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin (2014). *Bayesian Data Analysis*. Third edition. Texts in Statistical Science Series. 667 pp. (cit. on pp. 9, 10, 13, 15, 21, 22).

Gelman, Andrew and Donald B. Rubin (Nov. 1992). "Inference from Iterative Simulation Using Multiple Sequences". In: *Statistical Science* 7.4, pp. 457–472 (cit. on p. 21).

Gelman, Andrew, Daniel Simpson, and Michael Betancourt (Oct. 2017). "The Prior Can Often Only Be Understood in the Context of the Likelihood". In: *Entropy* 19.10 (10), p. 555 (cit. on p. 56).

Gelman, Andrew, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian Bürkner, and Martin Modrák (Nov. 3, 2020). *Bayesian Workflow*. arXiv: `2011.01808 [stat]`. URL: `http://arxiv.org/abs/2011.01808` (visited on 11/29/2022). preprint (cit. on p. 46).

Gelman, Andrew and Yuling Yao (Jan. 1, 2021). "Holes in Bayesian Statistics". In: *Journal of Physics G: Nuclear and Particle Physics* 48.1, p. 014002. arXiv: `2002.06467 [math, stat]` (cit. on p. 56).

Girolami, Mark (Nov. 17, 2008). "Bayesian Inference for Differential Equations". In: *Theoretical Computer Science*. Computational Methods in Systems Biology 408.1, pp. 4–16 (cit. on p. 2).

Goodman, Jonathan and Jonathan Weare (Jan. 31, 2010). "Ensemble Samplers with Affine Invariance". In: *Communications in Applied Mathematics and Computational Science* 5.1, pp. 65–80 (cit. on p. 44).

Gorinova, Maria I., Andrew D. Gordon, Charles Sutton, and Matthijs Vákár (Dec. 9, 2021). "Conditional Independence by Typing". In: *ACM Transactions on Programming Languages and Systems* 44.1, 4:1–4:54 (cit. on p. 70).

Graham, Matthew M. and Amos J. Storkey (Jan. 2017). "Asymptotically Exact Inference in Differentiable Generative Models". In: *Electronic Journal of Statistics* 11.2, pp. 5105–5164 (cit. on p. 29).

Green, Gregory M., Edward Schlafly, Catherine Zucker, Joshua S. Speagle, and Douglas Finkbeiner (Dec. 2019). "A 3D Dust Map Based on Gaia, Pan-STARRS 1, and 2MASS". in: *The Astrophysical Journal* 887.1, p. 93 (cit. on p. 147).

Green, Gregory M., Edward F. Schlafly, Douglas Finkbeiner, Hans-Walter Rix, Nicolas Martin, William Burgett, Peter W. Draper, Heather Flewelling, Klaus Hodapp, Nicholas Kaiser, Rolf-Peter Kudritzki, Eugene A. Magnier, Nigel Metcalfe, John L. Tonry, Richard Wainscoat, and Christopher Waters (July 21, 2018). "Galactic Reddening in 3D from Stellar Photometry – an Improved Map". In: *Monthly Notices of the Royal Astronomical Society* 478.1, pp. 651–666 (cit. on p. 147).

Green, Peter J. (Dec. 1, 1995). "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination". In: *Biometrika* 82.4, pp. 711–732 (cit. on p. 57).

Green, Stephen R. and Jonathan Gair (June 2021). "Complete Parameter Inference for GW150914 Using Deep Learning". In: *Machine Learning: Science and Technology* 2.3, 03LT01 (cit. on p. 97).

Greenberg, David, Marcel Nonnenmacher, and Jakob Macke (May 24, 2019). "Automatic Posterior Transformation for Likelihood-Free Inference". In: *Proceedings of the 36th International Conference on Machine Learning*, pp. 2404–2414 (cit. on p. 32).

Griesemer, Dustin, James R. Xue, Steven K. Reilly, Jacob C. Ulirsch, Kalki Kukreja, Joe R. Davis, Masahiro Kanai, David K. Yang, John C. Butts, Mehmet H. Guney, Jeremy Luban, Stephen B. Montgomery, Hilary K. Finucane, Carl D. Novina, Ryan Tewhey, and Pardis C. Sabeti (Sept. 2021). "Genome-Wide Functional Screen of 3′UTR Variants Uncovers Causal Variants for Human Disease and Evolution". In: *Cell* 184.20, 5247–5260.e19 (cit. on p. 100).

Gubernatis, J. E. (May 2005). "Marshall Rosenbluth and the Metropolis Algorithm". In: *Physics of Plasmas* 12.5, p. 057303 (cit. on p. 15).

Hahn, ChangHoon and Peter Melchior (Mar. 14, 2022). "Accelerated Bayesian SED Modeling Using Amortized Neural Posterior Estimation". arXiv: `2203.07391 [astro-ph, stat]` (cit. on p. 97).

Hastings, W. K. (Apr. 1, 1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". In: *Biometrika* 57.1, pp. 97–109 (cit. on p. 14).

Hermans, Joeri, Volodimir Begy, and Gilles Louppe (Nov. 21, 2020). "Likelihood-Free MCMC with Amortized Approximate Ratio Estimators". In: *Proceedings of the 37th International Conference on Machine Learning*, pp. 4239–4248 (cit. on pp. 35, 72).

Hermans, Joeri, Arnaud Delaunoy, François Rozet, Antoine Wehenkel, and Gilles Louppe (Oct. 14, 2021). *Averting A Crisis In Simulation-Based Inference*. arXiv: `2110.06581 [cs, stat]`. URL: `http://arxiv.org/abs/2110.06581` (visited on 11/29/2022). preprint (cit. on p. 95).

Hoffman, Matthew D. and Andrew Gelman (2014). "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". In: *Journal of Machine Learning Research* 15.47, pp. 1593–1623 (cit. on pp. 1, 19, 59).

Huijser, David, Jesse Goodman, and Brendon J. Brewer (2022). "Properties of the Affine-Invariant Ensemble Sampler's 'Stretch Move' in High Dimensions". In: *Australian & New Zealand Journal of Statistics* 64.1, pp. 1–26 (cit. on p. 44).

Jaynes, E.T. (2003). *Probability Theory: The Logic of Science*. Ed. by G.L. Bretthorst (cit. on p. 7).

Jones, D. S., Michael Plank, and B. D. Sleeman (Nov. 12, 2009). *Differential Equations and Mathematical Biology*. 2nd ed. 462 pp. (cit. on p. 37).

Jordan, Michael I., Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul (Nov. 1, 1999). "An Introduction to Variational Methods for Graphical Models". In: *Machine Learning* 37.2, pp. 183–233 (cit. on p. 126).

Jorgensen, Bent (1987). "Exponential Dispersion Models". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 49.2, pp. 127–162. JSTOR: `2345415` (cit. on p. 53).

Kingma, Diederik P. and Jimmy Ba (Dec. 22, 2014). "Adam: A Method for Stochastic Optimization". arXiv: `1412.6980 [cs]` (cit. on pp. 109, 171–173).

Kingma, Diederik P. and Prafulla Dhariwal (2018). "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems*. Vol. 31 (cit. on pp. 3, 26).

Kingma, Diederik P., Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). "Improved Variational Inference with Inverse Autoregressive Flow". In: *Advances in Neural Information Processing Systems*. Vol. 29 (cit. on pp. 27, 129, 130).

Kingma, Diederik P. and Max Welling (Dec. 20, 2013). "Auto-Encoding Variational Bayes". arXiv: `1312.6114 [cs, stat]` (cit. on pp. 3, 128, 129, 131, 151).

— (2019). "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392. arXiv: `1906.02691` (cit. on p. 126).

Kucukelbir, Alp, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei (Jan. 1, 2017). "Automatic Differentiation Variational Inference". In: *The Journal of Machine Learning Research* 18.1, pp. 430–474 (cit. on p. 1).

Kumar, Ravin, Colin Carroll, Ari Hartikainen, and Osvaldo Martin (Jan. 15, 2019). "ArviZ a Unified Library for Exploratory Analysis of Bayesian Models in Python". In: *Journal of Open Source Software* 4.33, p. 1143 (cit. on p. 24).

Laplace, Pierre Simon (1774). "Memoir on the Probability of the Causes of Events". In: *Statistical Science* 1.3, pp. 364–378. JSTOR: `2245476` (cit. on p. 44).

Leimkuhler, Benedict and Sebastian Reich (2004). *Simulating Hamiltonian Dynamics*. 464 pp. (cit. on p. 19).

Lewandowski, Daniel, Dorota Kurowicka, and Harry Joe (Oct. 1, 2009). "Generating Random Correlation Matrices Based on Vines and Extended Onion Method". In: *Journal of Multivariate Analysis* 100.9, pp. 1989–2001 (cit. on p. 123).

Liang, Percy and Dan Klein (2009). "Online EM for Unsupervised Models". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, p. 611 (cit. on p. 120).

Lintusaari, Jarno, Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander (Jan. 1, 2017). "Fundamentals and Recent Developments in Approximate Bayesian Computation". In: *Systematic Biology* 66.1, e66–e82 (cit. on p. 30).

Llorente, F., L. Martino, D. Delgado, and J. López-Santiago (Feb. 7, 2023). "Marginal Likelihood Computation for Model Selection and Hypothesis Testing: An Extensive Review". In: *SIAM Review* 65.1, pp. 3–58 (cit. on p. 12).

Lueckmann, Jan-Matthis, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke (Mar. 18, 2021). "Benchmarking Simulation-Based Inference". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics, pp. 343–351 (cit. on pp. 38, 53, 74, 98).

Lueckmann, Jan-Matthis, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke (2017). "Flexible Statistical Inference for Mechanistic Models of Neural Dynamics". In: *Advances in Neural Information Processing Systems*. Vol. 30 (cit. on pp. 2, 32).

Lunn, David J., Andrew Thomas, Nicky Best, and David Spiegelhalter (Oct. 1, 2000). "WinBUGS - A Bayesian Modelling Framework: Concepts, Structure, and Extensibility". In: *Statistics and Computing* 10.4, pp. 325–337 (cit. on p. 1).

MacDonald, Benn (2018). *Cside 2018*. URL: `https://www.gla.ac.uk/schools/mathematicsstatistics/events/conferences/cside2018/` (visited on 12/05/2018) (cit. on pp. 2, 37).

Macdonald, Benn and Dirk Husmeier (2015). "Gradient Matching Methods for Computational Inference in Mechanistic Models for Systems Biology: A Review and Comparative Analysis". In: *Frontiers in Bioengineering and Biotechnology* 3 (cit. on pp. 2, 38).

MacKay, David JC (2003). *Information Theory, Inference and Learning Algorithms*. 628 pp. (cit. on pp. 9, 12–14, 16, 18, 127).

Meager, Rachael (Jan. 2019). "Understanding the Average Impact of Microcredit Expansions: A Bayesian Hierarchical Analysis of Seven Randomized Experiments". In: *American Economic Journal: Applied Economics* 11.1, pp. 57–91 (cit. on p. 1).

Melchior, P. and A. D. Goulding (Oct. 1, 2018). "Filling the Gaps: Gaussian Mixture Models from Noisy, Truncated or Incomplete Samples". In: *Astronomy and Computing* 25, pp. 183–194 (cit. on p. 105).

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller (June 1953). “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092 (cit. on p. 15).

Mohamed, Shakir, Mihaela Rosca, Michael Figurnov, and Andriy Mnih (2020). “Monte Carlo Gradient Estimation in Machine Learning”. In: *Journal of Machine Learning Research* 21.132, pp. 1–62 (cit. on p. 131).

Moore, R. E., C. Rosato, and S. Maskell (Nov. 8, 2021). “Assessing the Uncertainty of Epidemiological Forecasts with Normalised Estimation Error Squared”. arXiv: `2111.04498 [stat]` (cit. on p. 53).

Murphy, Kevin P. (Sept. 7, 2012). *Machine Learning: A Probabilistic Perspective*. 1102 pp. (cit. on pp. 7, 8, 25, 26, 102).

Nadarajah, Saralees, Yuanyuan Zhang, and Tibor K. Pogány (Apr. 2018). “On Sums of Independent Generalized Pareto Random Variables with Applications to Insurance and CAT Bonds”. In: *Probability in the Engineering and Informational Sciences* 32.2, pp. 296–305 (cit. on p. 55).

Nalisnick, Eric, Lars Hertel, and Padhraic Smyth (2016). “Approximate Inference for Deep Latent Gaussian Mixtures”. In: *Workshop on Bayesian Deep Learning, NeurIPS 2016*, p. 4 (cit. on p. 151).

Neal, Radford M (2011). “MCMC Using Hamiltonian Dynamics”. In: *Handbook of Markov Chain Monte Carlo* 2.11, p. 2 (cit. on pp. 2, 18, 19, 44, 50, 59, 80).

Obermeyer, Fritz, Martin Jankowiak, Nikolaos Barkas, Stephen F. Schaffner, Jesse D. Pyle, Leonid Yurkovetskiy, Matteo Bosso, Daniel J. Park, Mehrtash Babadi, Bronwyn L. MacInnis, Jeremy Luban, Pardis C. Sabeti, and Jacob E. Lemieux (June 17, 2022). “Analysis of 6.4 Million SARS-CoV-2 Genomes Identifies Mutations Associated with Fitness”. In: *Science* 376.6599, pp. 1327–1332 (cit. on p. 1).

Papamakarios, George and Iain Murray (2016). “Fast ϵ-Free Inference of Simulation Models with Bayesian Conditional Density Estimation”. In: *Advances in Neural Information Processing Systems* 29 (cit. on pp. 31, 32).

Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2021). “Normalizing Flows for Probabilistic Modeling and Inference”. In: *Journal of Machine Learning Research* 22.57, pp. 1–64 (cit. on pp. 3, 26, 28, 70, 124, 129).

Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). "Masked Autoregressive Flow for Density Estimation". In: *Advances in Neural Information Processing Systems*. Vol. 30 (cit. on pp. 27, 71, 130, 133, 173).

Papamakarios, George, David Sterratt, and Iain Murray (Apr. 11, 2019). "Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pp. 837–848 (cit. on pp. 33, 71).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32 (cit. on pp. 109, 173).

Perryman, M A C, L Lindegren, J. Kovalevsky, E Høg, U. Bastian, P L Bernacca, M. Crézé, F Donati, M Grenon, M Grewing, F Van Leeuwen, H Van Der Marel, F Mignard, C A Murray, R S Le Poole, H Schrijver, C Turon, Frédéric Arenou, M Froeschlé, and C S Petersen (1997). "The Hipparcos Catalogue". In: *Astronomy and Astrophysics - A&A* 323.1, pp. 49–52 (cit. on pp. 3, 100).

Phan, Du, Neeraj Pradhan, and Martin Jankowiak (2019). "Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro". arXiv: `1912.11554` (cit. on p. 62).

Powell, M. J. D. (Jan. 1, 1964). "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives". In: *The Computer Journal* 7.2, pp. 155–162 (cit. on p. 44).

Revfeim, K. J. A. (Dec. 5, 1984). "An Initial Model of the Relationship between Rainfall Events and Daily Rainfalls". In: *Journal of Hydrology* 75.1, pp. 357–364 (cit. on p. 52).

Rezende, Danilo Jimenez and Shakir Mohamed (June 1, 2015). "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1530–1538 (cit. on pp. 128, 129).

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (June 18, 2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286 (cit. on pp. 3, 129, 131).

Riello, M., F. De Angeli, D. W. Evans, P. Montegriffo, J. M. Carrasco, G. Busso, L. Palaversa, P. W. Burgess, C. Diener, M. Davidson, N. Rowell, C. Fabricius, C. Jordi, M. Bellazzini, E. Pancino, D. L. Harrison, C. Cacciari, F. van Leeuwen, N. C. Hambly, S. T. Hodgkin, P. J. Osborne, G. Altavilla, M. A. Barstow, A. G. A. Brown, M. Castellani, S. Cowell, F. De Luise, G. Gilmore, G. Giuffrida, S. Hidalgo, G. Holland, S. Marinoni, C. Pagani, A. M. Piersimoni, L. Pulone, S. Ragaini, M. Rainer, P. J. Richards, N. Sanna, N. A. Walton, M. Weiler, and A. Yoldas (May 1, 2021). "Gaia Early Data Release 3 - Photometric Content and Validation". In: *Astronomy & Astrophysics* 649, A3 (cit. on p. 145).

Ritchie, James A. and Iain Murray (Nov. 26, 2019). "Scalable Extreme Deconvolution". arXiv: `1911.11663 [cs, stat]` (cit. on p. 4).

Salvatier, John, Thomas V. Wiecki, and Christopher Fonnesbeck (Apr. 6, 2016). "Probabilistic Programming in Python Using PyMC3". In: *PeerJ Computer Science* 2, e55 (cit. on pp. 1, 38, 53).

Schubert, Erich and Michael Gertz (2018). "Numerically Stable Parallel Computation of (Co-)Variance". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*, 10:1–10:12 (cit. on p. 107).

Schwarz, Gideon (Mar. 1978). "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2, pp. 461–464 (cit. on p. 118).

Scott, James A., Axel Gandy, Swapnil Mishra, Juliette Unwin, Seth Flaxman, and Samir Bhatt (2020). *Epidemia: Modeling of Epidemics Using Hierarchical Bayesian Models* (cit. on p. 53).

Sculley, D. (2010). "Web-Scale k-Means Clustering". In: *Proceedings of the 19th International Conference on World Wide Web*, p. 1177 (cit. on p. 171).

Shampine, Lawrence F. and Mark W. Reichelt (Jan. 1997). "The MATLAB ODE Suite". In: *SIAM Journal on Scientific Computing* 18.1, pp. 1–22 (cit. on p. 40).

Simitev, R. D. and V. N. Biktashev (Jan. 2011). "Asymptotics of Conduction Velocity Restitution in Models of Electrical Excitation in the Heart". In: *Bulletin of Mathematical Biology* 73.1, pp. 72–115 (cit. on pp. 2, 37, 39, 41).

Sisson, Scott A., Yanan Fan, and Mark Beaumont (Sept. 3, 2018). *Handbook of Approximate Bayesian Computation*. 424 pp. (cit. on p. 30).

Skrutskie, M. F., R. M. Cutri, R. Stiening, M. D. Weinberg, S. Schneider, J. M. Carpenter, C. Beichman, R. Capps, T. Chester, J. Elias, J. Huchra, J. Liebert, C. Lonsdale, D. G. Monet, S. Price, P. Seitzer, T. Jarrett, J. D. Kirkpatrick, J. E. Gizis, E. Howard, T. Evans, J. Fowler, L. Fullmer, R. Hurt, R. Light, E. L. Kopan, K. A. Marsh, H. L. McCallon, R. Tam, S. Van Dyk, and S. Wheelock (Feb. 2006). "The Two Micron All Sky Survey (2MASS)". in: *The Astronomical Journal* 131.2, pp. 1163–1183 (cit. on p. 151).

Smyth, Gordon K. and Bent Jørgensen (May 2002). "Fitting Tweedie's Compound Poisson Model to Insurance Claims Data: Dispersion Modelling". In: *ASTIN Bulletin: The Journal of the IAA* 32.1, pp. 143–157 (cit. on p. 52).

Stan Development Team (2022). *Stan Modeling Language Users Guide and Reference Manual, Version 2.31.0* (cit. on pp. 20, 38, 59–61).

Stigler, Stephen M. (1986). "Laplace's 1774 Memoir on Inverse Probability". In: *Statistical Science* 1.3, pp. 359–363. JSTOR: `2245475` (cit. on p. 44).

Sugiyama, Masashi, Taiji Suzuki, and Takafumi Kanamori (2012). *Density Ratio Estimation in Machine Learning* (cit. on p. 34).

Talts, Sean, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman (Oct. 21, 2020). "Validating Bayesian Inference Algorithms with Simulation-Based Calibration". arXiv: `1804.06788 [stat]` (cit. on pp. 24, 49, 50, 67).

Tejero-Cantero, Alvaro, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J. Gonçalves, David S. Greenberg, and Jakob H. Macke (2020). "SBI: A Toolkit for Simulation-Based Inference". In: *Journal of Open Source Software* 5.52, p. 2505 (cit. on pp. 36, 73, 96).

Teller, Edward and Judith Schoolery (Sept. 9, 2009). *Memoirs: A Twentieth Century Journey In Science And Politics*. 678 pp. (cit. on p. 15).

The Gaia Collaboration (Nov. 2016). "The Gaia Mission". In: *Astronomy and Astrophysics* 595, A1 (cit. on pp. 100, 124).

— (July 30, 2022). *Gaia Data Release 3: Summary of the Content and Survey Properties*. arXiv: `2208.00211 [astro-ph]`. URL: `http://arxiv.org/abs/2208.00211` (visited on 08/30/2022). preprint (cit. on pp. 112, 141).

Thomas, Owen, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann (Jan. 2021). "Likelihood-Free Inference by Ratio Estimation". In: *Bayesian Analysis* -1 (-1), pp. 1–31 (cit. on pp. 36, 72).

Tokuda, Tomoki, Ben Goodrich, I. Van Mechelen, Andrew Gelman, and F. Tuerlinckx (2011). "Visualizing Distributions of Covariance Matrices". In: *Columbia Univ., New York, USA, Tech. Rep*, pp. 18–18 (cit. on p. 122).

UK Health Security Agency (2022). *Reproduction Number (R) and Growth Rate: Methodology*. GOV.UK. URL: `https://www.gov.uk/government/publications/reproduction-number-r-and-growth-rate-methodology/` (visited on 02/15/2022) (cit. on p. 53).

Urbut, Sarah M., Gao Wang, Peter Carbonetto, and Matthew Stephens (Jan. 2019). "Flexible Statistical Methods for Estimating and Testing Effects in Genomic Studies with Multiple Conditions". In: *Nature Genetics* 51.1 (1), pp. 187–195 (cit. on p. 100).

Vandegar, Maxime, Michael Kagan, Antoine Wehenkel, and Gilles Louppe (Mar. 18, 2021). "Neural Empirical Bayes: Source Distribution Estimation and Its Applications to Simulation-Based Inference". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics, pp. 2107–2115 (cit. on p. 128).

Vehtari, Aki, Andrew Gelman, and Jonah Gabry (Sept. 1, 2017). "Practical Bayesian Model Evaluation Using Leave-One-out Cross-Validation and WAIC". in: *Statistics and Computing* 27.5, pp. 1413–1432 (cit. on p. 49).

Vehtari, Aki, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner (June 2021). "Rank-Normalization, Folding, and Localization: An Improved R^ for Assessing Convergence of MCMC (with Discussion)". In: *Bayesian Analysis* 16.2, pp. 667–718 (cit. on pp. 22, 24, 45, 62).

White, Martin, Adam D. Myers, Nicholas P. Ross, David J. Schlegel, Joseph F. Hennawi, Yue Shen, Ian McGreer, Michael A. Strauss, Adam S. Bolton, Jo Bovy, X. Fan, Jordi Miralda-Escude, N. Palanque-Delabrouille, I. Paris, P. Petitjean, D. P. Schneider, M. Viel, David H. Weinberg, Ch. Yeche, I. Zehavi, K. Pan, S. Snedden, D. Bizyaev, H. Brewington, J. Brinkmann, V. Malanushenko, E. Malanushenko, D. Oravetz, A. Simmons, A. Sheldon, and Benjamin A. Weaver (Aug. 1, 2012). "The Clustering of Intermediate-Redshift Quasars as Measured by the Baryon Oscillation Spectroscopic Survey: Quasar Clustering". In: *Monthly Notices of the Royal Astronomical Society* 424.2, pp. 933–950 (cit. on p. 100).

Williams, Peter M. (May 1996). "Using Neural Networks to Model Conditional Multivariate Densities". In: *Neural Computation* 8.4, pp. 843–854 (cit. on p. 108).

Williams, Ronald J. (May 1, 1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8.3, pp. 229–256 (cit. on p. 131).

Wood, Simon N. (Aug. 2010). "Statistical Inference for Noisy Nonlinear Ecological Dynamic Systems". In: *Nature* 466.7310 (7310), pp. 1102–1104 (cit. on p. 33).

Yakut, K., W. Zima, B. Kalomeni, H. Van Winckel, C. Waelkens, P. De Cat, E. Bauwens, M. Vučković, S. Saesen, L. Le Guillou, M. Parmaksızoğlu, K. Uluç, I. Khamitov, G. Raskin, and C. Aerts (Aug. 1, 2009). "Close Binary and Other Variable Stars in the Solar-Age Galactic Open Cluster M 67". In: *Astronomy & Astrophysics* 503.1 (1), pp. 165–176 (cit. on p. 148).

Appendix A

Experimental Details for Chapter 5

A.1 Synthetic Data

For each method, parameter initialisation was done by running a minibatch variant of the K-means clustering algorithm on the training dataset (Sculley 2010). Component means were set to the cluster location. Component weights were set to the fraction of datapoints assigned to each cluster. Component covariances were set to diagonal unit variances.

Each method was run for 20 epochs. For the baseline method using the existing EM approach, we used the Python interface to the implementation of Bovy, Hogg, and Roweis 2011 with the default settings.

Both minibatch methods used minibatch sizes of 4096 samples. For the minibatch-EM method, we used a step size λ of 10^{-2} For the minibatch-SGD method, we used a learning rate of 10^{-3} and the Adam optimiser (Kingma and Ba 2014).

A.2 Astrometric Data

The following ADQL query was used to retrieve the data from the Gaia archive:

```
SELECT TOP 2000000
ra , dec , parallax , pmra, pmdec, ra_error , dec_error ,
    parallax_error , pmra_error , pmdec_error ,
    ra_dec_corr , ra_parallax_corr , ra_pmra_corr ,
    ra_pmdec_corr , dec_parallax_corr , dec_pmra_corr ,
    dec_pmdec_corr, parallax_pmra_corr,
    parallax_pmdec_corr, pmra_pmdec_corr
FROM gaiadr3.gaia_source
WHERE ruwe < 1.4
ORDER BY random_index;
```

The first 80% of the data was used for training, the next 10% for validation and the final 10% for testing.

The same minibatch K-means initialisation method was used as for the synthetic dataset experiment. For the baseline method we used the same implementation as for the synthetic data experiment with the default settings. We ran it for 100 epochs.

For both minibatch methods we used minibatch sizes of 4096 samples, with 100 epochs. For the minibatch-EM method, we used an initial step size λ of 10^{-2}, reducing it by a factor of 10 if no improvement in validation log-likelihood was seen for 10 epochs. For the minibatch-SGD method, we used an initial learning rate of 10^{-3} and the Adam optimiser (Kingma and Ba 2014), with the same reduction criteria as the minibatch-EM method.

Appendix B

Experimental Details for Chapter 6

B.1 Synthetic Datasets

For the experiments using GMMs, we reused the same setup as in Chapter 5, with mixture components K as mentioned in the main text.

For the prior flow we used multiple transforms of the autoregressive variant of the Neural Spline Flow applied to a standard normal base distribution (Durkan, Bekasov, et al. 2019). For the posterior flow we used multiple transforms of the affine Masked Autoregressive Flow with the noise distribution as a base distribution (Papamakarios, Pavlakou, and Murray 2017). After each transform we reversed the ordering of the elements. Both flows used fully-connected neural networks to parametrise the transforms. We used the flow implementations from the `pyro` probabilistic programming library written in `PyTorch` along with their framework for variational inference (Bingham et al. 2019; Paszke et al. 2019).

The posterior flow used a fully-connected neural network as an embedding network that took the noisy observation and noise covariance as inputs. It outputted a vector with the same dimensionality as the hidden units, which was prepended to the inputs of the posterior flow neural networks as a conditioning input.

We used different numbers of Monte Carlo samples to estimate the loss at each training step, validation (at the end of every epoch) and final evaluations. We use the Adam optimiser to perform stochastic gradient descent (Kingma and Ba 2014). For validation and final evaluation we also used the samples to compute the IWAE bound to estimate the marginal log-likelihood. Table B.1

Table B.1: Hyperparameter settings used for the flow experiments with synthetic datasets.

Hyperparameter	Value
Learning Rate	5e-4
Minibatch Size	1024
Max Epochs	100
Training Loss Samples	10
Validation Loss Samples	25
Final Loss Samples	100
Embedding Hidden Units	128
Embedding Hidden Layers	3
Posterior Flow Transforms	10
Posterior Flow Hidden Units per Transform	128
Posterior Flow Hidden Layers per Transform	3
Prior Flow Transforms	5
Prior Flow Hidden Units per Transform	64
Prior Flow Hidden Layers per Transform	3

reports the complete hyperparameter settings.

B.2 Gaia Datasets

The following ADQL query was used to retrieve the data from the Gaia archive:

```
SELECT TOP 20000000
ra, dec, parallax, pmra, pmdec, phot_g_mean_flux,
    phot_bp_mean_flux, phot_rp_mean_flux, ra_error,
    dec_error, parallax_error, pmra_error, pmdec_error
    , phot_g_mean_flux_error, phot_bp_mean_flux_error,
     phot_rp_mean_flux_error, ra_dec_corr,
    ra_parallax_corr, ra_pmra_corr, ra_pmdec_corr,
    dec_parallax_corr, dec_pmra_corr, dec_pmdec_corr,
    parallax_pmra_corr, parallax_pmdec_corr,
    pmra_pmdec_corr
```

```
FROM gaiadr3.gaia_source
WHERE ruwe < 1.4
AND dec > −30.0
AND phot_g_mean_flux IS NOT NULL
AND phot_g_mean_flux < 100000
AND phot_bp_mean_flux IS NOT NULL
AND phot_bp_mean_flux < 50000
AND phot_rp_mean_flux IS NOT NULL
AND phot_rp_mean_flux < 100000
ORDER BY random_index;
```

The first 80% of the data was used for training, the next 10% for validation and the final 10% for testing.

For the experiments using GMMs, we reused the same setup and hyperparameters as in Chapter 5. For the astrometric-only experiments, we set the number of mixture components to $K = 128$. For the photometric experiments, we increased the number of mixture components to $K = 512$.

For experiments using flows, we used the same setup as for the synthetic data experiments with the hyperparameters in Table B.2.

Table B.2: Hyperparameter settings used for the flow experiments with the Gaia datasets.

Hyperparameter	Value
Learning Rate	5e-4
Minibatch Size	1024
Max Epochs	100
Training Loss Samples	5
Validation Loss Samples	25
Final Loss Samples	100
Embedding Hidden Units	256
Embedding Hidden Layers	4
Posterior Flow Transforms	10
Posterior Flow Transform Hidden Units	256
Posterior Flow Transform Hidden Layers	3
Prior Flow Transforms	10
Prior Flow Transform Hidden Units	256
Prior Flow Transform Hidden Layers	3