5 Bayesian Sampling

As emphasised before, the output of Bayesian inference is a posterior probability distribution that encodes our state of knowledge about the parameters of the model based on the observed data and prior information. In certain contexts, for example when using conjugate models, the posterior can be written down in a closed analytic form and used directly for subsequent computation of derived quantities of interest. However, most often the posterior is not known in closed form. There are three approaches to inference in such situations. One is to use a Normal approximation to the posterior, the second is to use brute force integration methods and the third is to draw a set of representative samples from the posterior for us in Monte Carlo integration over the posterior.

5.1 Posterior computation: Bayesian Central Limit Theorem

The **Bayesian Central Limit Theorem** can be used to approximate posteriors, in the limit that the number of observations, $n \to \infty$. Suppose that we have samples $X_1, \ldots, X_n \overset{\text{iid}}{\sim} p(x \mid \boldsymbol{\theta})$ and that the prior, $p(\boldsymbol{\theta})$, and likelihood, $p(x|\boldsymbol{\theta})$, are both twice differentiable near $\widehat{\boldsymbol{\theta}}_{\text{post}}$, the location of the peak of the posterior distribution. Then, for $n \to \infty$, we can approximate

$$p(\boldsymbol{\theta} \mid \mathbf{x}) \sim \mathrm{N}\left(\widehat{\boldsymbol{\theta}}_{\text{post}}, [I^{\text{post}}(\boldsymbol{\theta}, \mathbf{x})]^{-1}\right)$$

where

$$I^{\text{post}}(\boldsymbol{\theta}, \mathbf{x}) = -\left[\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \log p(\boldsymbol{\theta} \mid \mathbf{x})\right]_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}_{\text{post}}}.$$

The Bayesian central limit theorem follows from the usual central limit theorem. It used to be widely used due to the computational cost of generating posterior samples. However, it relies on the number of observations being large, which is often difficult to ensure in practice. Therefore, its use is no longer so widespread since computers are now sufficiently powerful to enable the generation of large numbers of posterior samples relatively cheaply.

5.2 Posterior computation: numerical integration

In low numbers of dimensions, posterior integrals can be computed using standard numerical integration techniques. There is a large literature on approximating integrals in various ways. The simplest is a grid approach, where the posterior is evaluated at a set of regularly spaced points in the space of waveform parameters. This can be thought of as a type of sampling approximation, where the samples are on a uniform grid. Direct integration rapidly becomes prohibitively expensive as the dimensionality of the model parameter space increases. In addition, it can be inefficient, if the posterior has relatively compact support within the space of allowed values, since many of the grid points will be in regions with low posterior weight.

5.3 Posterior computation: direct sampling methods

As discussed before, sampling methods attempt to generate a set $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$, from the posterior, which can be used to approximate integrals over the posterior

$$\int f(\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\mathbf{x})\, \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{M} \sum_{i=1}^{M} f(\boldsymbol{\theta}_i).$$

Sampling methods can be **direct** or **stochastic**. Direct methods draw samples directly (or nearly directly) from the target probability distribution. Stochastic methods use **Markov chain Monte Carlo** methods to generate a sequence of samples that are drawn form the target distribution.

5.3.1 Method of inversion

The method of inversion is a simple application of the probability integral transformation. If we denote by F the cumulative distribution function of some random variable X, then the random variable $F(X)$ follows a $U[0,1]$ distribution. Therefore, if we can analytically compute the inverse of the cumulative distribution function, we can generate samples form X by generating samples from a uniform distribution. If

$$F(x) = \mathbb{P}(X \leq x)$$

and it has inverse F^{-1} then the algorithm is simply

1. Generate $u \sim U[0,1]$.
2. Compute $x = F^{-1}(u)$.

Example: exponential distribution with parameter r Suppose we want to draw $X \sim \mathrm{Exp}(r)$. The pdf of the exponential distribution is

$$p(x|r) = r \exp(-rx)$$

which has cumulative density function

$$F(X) = \int_0^X r \exp(-rx)\mathrm{d}x = 1 - \exp(-rX).$$

The inverse can be found as

$$u = F(x) \quad \Rightarrow \quad x = F^{-1}(u) = -\frac{1}{r} \ln(1-u).$$

Samples generated by applying this inverse to $U[0,1]$ samples are shown in Figure 11.

5.3.2 Rejection sampling

Rejection sampling draws samples from a distribution that can be directly sampled and then discards a subset of them that do not match the desired distribution. The simplest rejection sampling algorithm draws uniform samples from a box that encloses the distribution. Suppose that we want to draw samples $\theta_1, \ldots, \theta_n$ from a probability distribution with pdf $p(\theta)$ and that the pdf has compact support, so $p(\theta) = 0$ if $\theta \notin [a,b]$. Suppose additionally that the pdf at the mode of the probability distribution is $M = \max[p(\theta)]$. Rejection sampling proceeds as follows

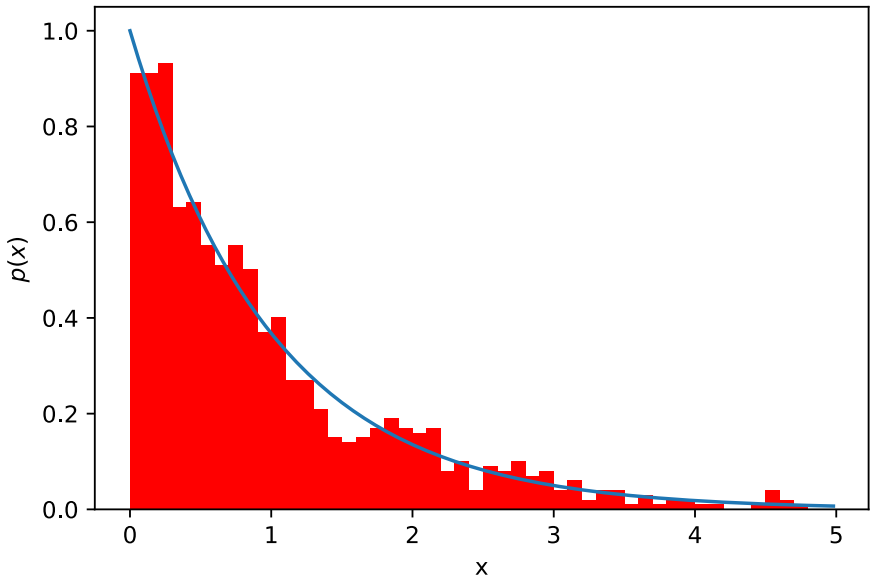


Figure 11: Histogram of samples drawn from the Exp(1) distribution using the method of inversion. The pdf of the exponential distribution is shown as a line for comparison.

1. Draw $\theta \sim U[a, b]$.
2. Draw $y \sim U[0, M]$.
3. If $y \leq p(\theta)$, accept θ as a sample from $p(\theta)$. Otherwise return to step 1.

Example: beta distribution We want to draw samples from a Beta$(3, 2)$ distribution. This has compact support on the interval $[0, 1]$ and the maximum value of the pdf is $M = 16/9$ (EXERCISE). In Figure 12 we illustrate this procedure by indicating which of the first 50 samples drawn in this way are rejected or accepted. In Figure 15 we show a histogram of the accepted samples in the first 1000 draws, which illustrates that the distribution of samples does follow the Beta$(3, 2)$ distribution as desired.

The box rejection sampling procedure does not work at all when the support of the target distribution is unbounded. In addition, it can be very inefficient for compact distributions with long tails. An alternative approach is to draw samples from an easy-to-sample distribution, $g(\theta)$, that is similar to the target distribution $p(\theta)$. First we find a number M such that $Mg(\theta) \geq p(\theta)\ \forall\theta$, i.e., we require $Mg(\theta)$ to contain the target distribution. The algorithm is then

1. Draw $\theta \sim g(\theta)$.
2. Draw $y \sim U[0, 1]$.
3. If $y \leq p(\theta)/(Mg(\theta))$, accept θ as a sample from $p(\theta)$. Otherwise return to step 1.

Trial samples are taken uniformly from within the region between the curve $Mg(\theta)$ and the θ axis. Samples that fall in the region between $p(\theta)$ and $Mg(\theta)$ are rejected. Therefore we

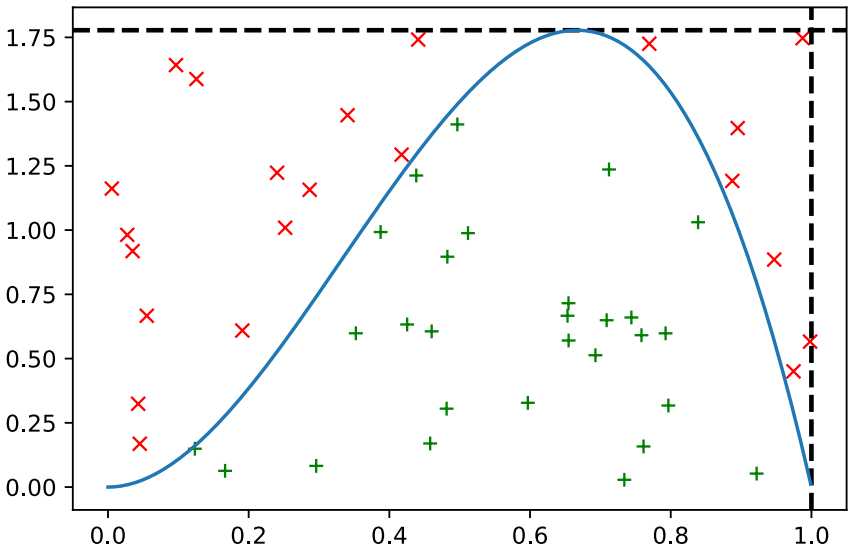


Figure 12: Accepted (green plusses) and rejected (red crosses) samples in the first 50 draws of the rejection sampling algorithm used to simulate the Beta$(3, 2)$ distribution. Only samples that lie within the target pdf are accepted.

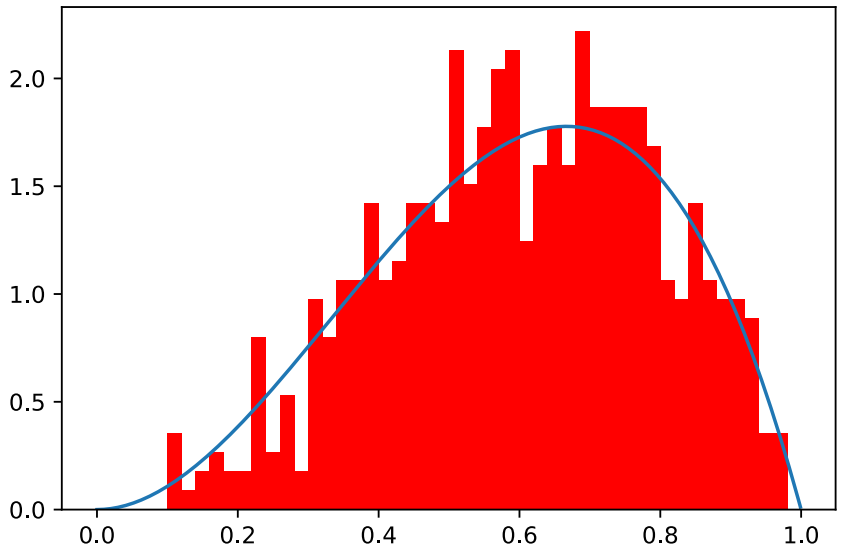


Figure 13: Histogram of the accepted samples in 1000 iterations of the rejection sampling algorithm. We compare the distribution to Beta$(3, 2)$, which is the target distribution.

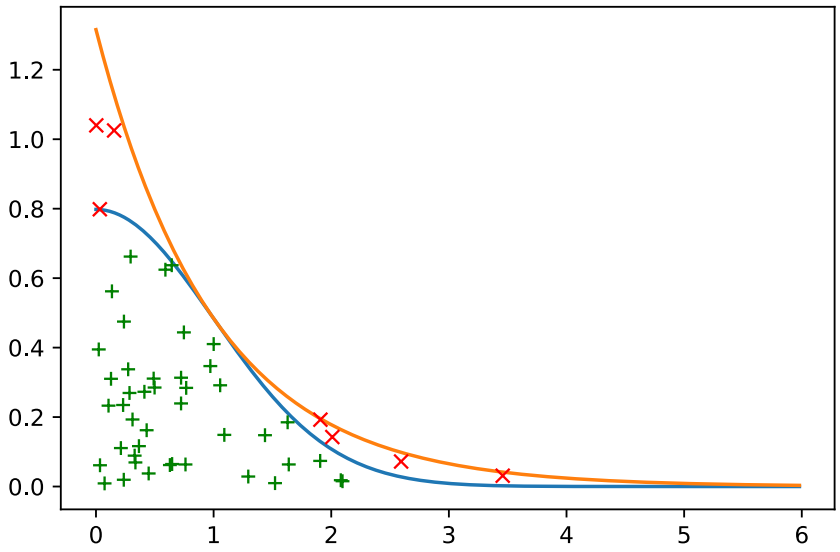


Figure 14: Accepted (green plusses) and rejected (red crosses) samples in the first 50 draws of the rejection sampling algorithm to simulate draws from a half-Normal distribution. Only samples that lie within the target pdf are accepted.

make the efficiency (i.e., the fraction of samples that are accepted) as large as possible by making the choice

$$M = \sup_{\theta} \left(\frac{p(\theta)}{g(\theta)} \right).$$

Example: half-Normal distribution We want to draw samples from the half-Normal distribution with pdf

$$p(\theta) = \begin{cases} \sqrt{\frac{2}{\pi}} \mathrm{e}^{-\frac{\theta^2}{2}} & \text{for } x \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

We will take $g(\theta) = \exp(-\theta)$, i.e., the exponential distribution with rate 1. We find M from

$$M = \sup_{\theta} \left(\frac{p(\theta)}{g(\theta)} \right) = \sup_{\theta > 0} \left(\sqrt{\frac{2}{\pi}} \exp\left[-\frac{1}{2}(\theta - 1)^2 + \frac{1}{2} \right] \right) = \sqrt{\frac{2}{\pi}} \mathrm{e}^{\frac{1}{2}}.$$

In Figure 14 we show the samples accepted and rejected during the first 50 iterations of the algorithm, and in Figure **??** we show a histogram of the accepted samples during 1000 iterations of the algorithm. We see that the histogram is correctly approximating the desired distribution.

5.3.3 Importance sampling

Rejection sampling can be effective and easy to implement, but it is not always possible to find an easy-to-sample target distribution that closely matches the target distribution. Additionally effort is wasted drawing samples and evaluating the posterior at points which

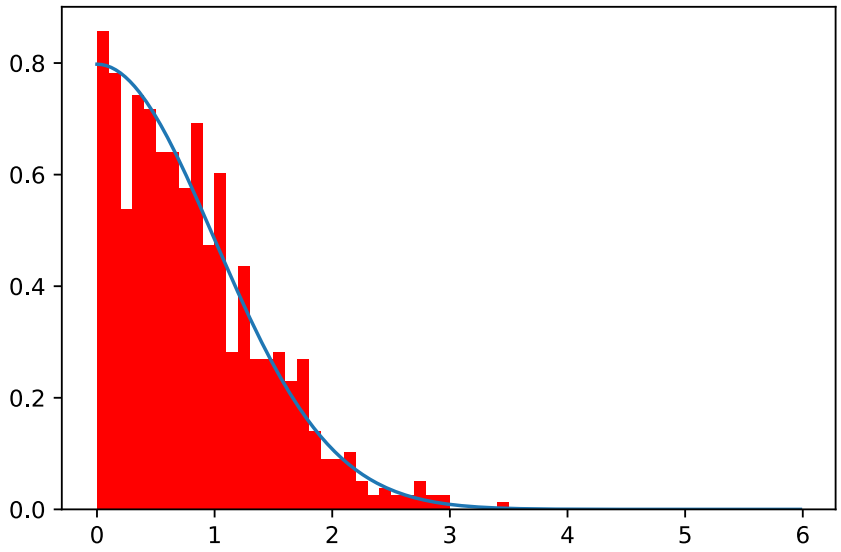


Figure 15: Histogram of the accepted samples in 1000 iterations of the rejection sampling algorithm. We compare the distribution to the target distribution, which in this case is a half-Normal distribution with mean 0 and variance 1.

are subsequently discarded as rejected samples. **Importance sampling** attempts to address the latter problem by using all samples.

Importance sampling uses an easy-to-sample reference distribution $g(\theta)$ as before, but now this is not rescaled, the only stipulation is that the support is common to that of the target distribution, i.e., if $p(\theta) > 0$ then $g(\theta) > 0$. No samples are discarded. Instead the samples are defined **importance weights** via

$$w_i = \frac{p(\theta)}{g(\theta)}$$

and integrals over the target distribution are approximated by weighted averages over the samples

$$\int f(\theta)p(\theta)\mathrm{d}\theta \approx \frac{1}{M}\sum_{i=1}^{M} w_i f(\theta_i).$$

It is straightforward to see that

$$\mathbb{E}_g(w_i f(\theta_i)) = \int w(\theta)f(\theta)g(\theta)\mathrm{d}\theta = \int \frac{p(\theta)}{g(\theta)}f(\theta)g(\theta)\mathrm{d}\theta = \int f(\theta)p(\theta)\mathrm{d}\theta = \mathbb{E}_p(f(\theta))$$

so the importance sampling estimate is unbiased. However

$$\begin{aligned}\mathrm{var}_g(w_i f(\theta_i)) &= \int w^2(\theta)f^2(\theta)g(\theta)\mathrm{d}\theta - [\mathbb{E}_p(f(\theta))]^2 = \int \frac{p(\theta)}{g(\theta)}f^2(\theta)p(\theta)\mathrm{d}\theta - [\mathbb{E}_p(f(\theta))]^2 \\ &= \mathbb{E}_p\left(\frac{p(\theta)}{g(\theta)}f^2(\theta)\right) - [\mathbb{E}_p(f(\theta))]^2. \end{aligned} \tag{85}$$

We see that the importance sampling estimate can suffer from high variance if $g(\theta)$ is much smaller than $p(\theta)$ in regions where the function of interest has significant support.

Note that the above assumes that the normalisation of the target distribution is known, but this is not always the case when sampling from posterior distributions due to the difficulty of computing the Bayesian evidence. If the posterior is not normalised the weights can be renormalised as

$$\tilde{w_i} = \frac{w_i}{\sum_{j=1}^{M} w_j}.$$

The results on the mean and variance are now only approximate, but are valid asymptotically.

Example: Cauchy distribution Suppose we have a standard Cauchy distribution with pdf

$$p(\theta) = \frac{1}{\pi(1+\theta^2)}$$

and want to compute $\mathbb{P}(\theta > 2)$. We can sample from the distribution $g(\theta) = 2/\theta^2 \mathbb{I}(\theta > 2)$ using the method of inversion. This has the same support as the portion of $p(\theta)$ of interest. We define the importance weights

$$w_i = \frac{\theta_i^2}{2\pi(1+\theta_i^2)}$$

and then compute

$$\hat{p}_{>2} = \frac{1}{M}\sum_{i=1}^{M} w_i$$

since we are interested in $\mathbb{P}(\theta > 2)$ which is the integral of $\mathbb{I}(\theta > 2)$, but this equal to 1 throughout the region where $g(\theta)$ has support. Note that in this case it would be wrong to renormalise the weights since then we would compute the probability as 1. As an exercise, verify that using the above weights in the usual sampling estimate gives the expected result.

In Figure 16 we show the convergence of the importance sampling estimate of $\mathbb{P}(\theta > 2)$ as a function of the number of importance samples. We see that it converges much faster than if we used Monte Carlo draws from the Cauchy distribution itself. The correct probability is $\pi/2 - \tan^{-1}(2)/\pi = 0.14758$.

5.3.4 Sampling importance resampling

Sampling importance resampling is a simple extension of importance sampling that uses the importance samples to generate samples approximately from the target distribution. Given M importance samples, $\{\theta_1, \ldots, \theta_M\}$, the importance weights are computed and normalised as described above. Then M samples, $\{\phi_1, \ldots, \phi_M\}$ are drawn, with replacement, from the original set using the normalised weights as probabilities. Integrals over the target distribution can then be approximated by

$$\int f(\theta)p(\theta)\mathrm{d}\theta \approx \frac{1}{M}\sum_{i=1}^{M} f(\phi_i).$$

Sampling importance resampling is a form of **particle filtering**. One problem that it can suffer form is **particle depletion**, where a small number of samples carry the majority of

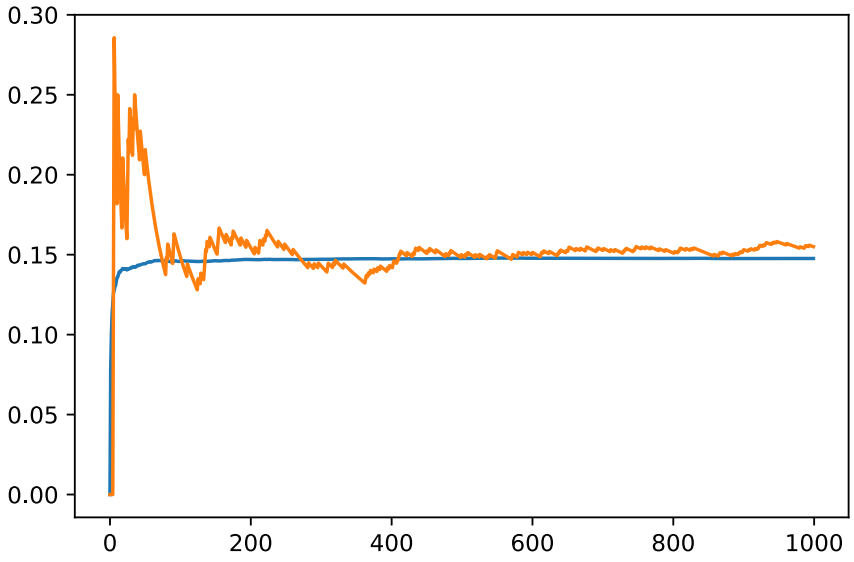


Figure 16: Importance sampling estimate of $\mathbb{P}(\theta > 2)$ for the standard Cauchy distribution as a function of number of samples (blue line), compared to Monte Carlo estimate using direct samples from the Cauchy distribution (yellow line).

the weight and therefore only a small number of points are represented repeatedly in the final data set. Particle depletion leads to poor estimates of derived quantities.

Example: Cauchy distribution We use sampling importance resampling to generate samples from the Cauchy distribution with $\theta > 2$ using the samples generated for the example in the previous section. A histogram of these values is shown in Figure 17, where they are compared to the target distribution, which is a truncated Cauchy distribution.

5.4 Posterior computation: Markov chain Monte Carlo

Direct sampling methods suffer form the problem of dimensionality. They are typically easy to implement in one dimension, but become increasingly challenging, inefficient or impossible to implement as the number of dimensions increases. In higher dimensions it is more common to use stochastic methods, in which a sequence of samples is constructed that has a distribution that follows the target distribution. Typically this is done using Markov chain Monte Carlo algorithms.

A **Markov Chain** is a sequence of random numbers, $\theta^1, \theta^2 \ldots$, such that the value of θ^{n+1} depends only on the previous values, θ^n, and not on earlier numbers in the sequence. A Markov chain can be simulated using a **transition kernel**, $\mathcal{K}(\theta^{n+1}|\theta^n)$, which is a conditional probability distribution for θ^{n+1} given the value of θ^n. The transition kernel uniquely defines the Markov chain. If we assume the Markov chain is **aperiodic** and **irreducible** then the distribution of samples in the Markov chain will converge to a **stationary distribution**, which is independent of the initial starting state of the chain. In Bayesian inference, the goal is to construct a Markov chain such that the stationary distribution is the posterior distribution, $p(\boldsymbol{\theta}|\mathbf{x})$.

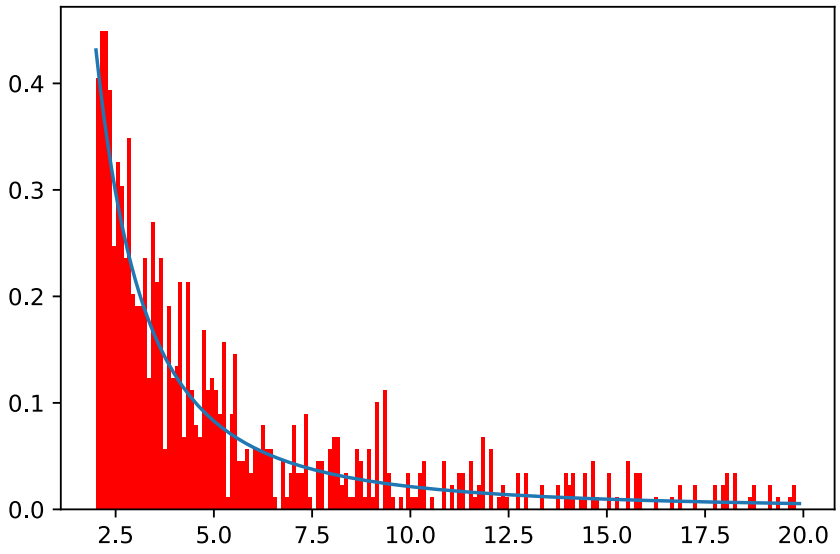


Figure 17: Histogram of 1000 sampling importance resampling samples for the Cauchy distribution in the region $\theta > 2$. These were generated from the importance samples constructed in the example in the last section. The line shows the expected distribution, which is the truncated standard Cauchy distribution.

A Markov chain with transition kernel $\mathcal{K}(\theta^{n+1}|\theta^n)$ is said to satisfy **detailed balance** for a distribution $\pi(\theta)$ if

$$\pi(\theta)\mathcal{K}(\phi|\theta) = \pi(\phi)\mathcal{K}(\theta|\phi) \; \forall \, \phi, \theta,$$

in which case $\pi(\theta)$ is the stationary distribution of the Markov chain. Enforcing detailed balance in the Markov chain, for $\pi(\theta) = p(\theta|\mathbf{x})$, will ensure we generate samples from the posterior distribution.

There are two widely used approaches to construct Markov chains satisfying detailed balance with a particular stationary distribution — Gibbs sampling and the Metropolis-Hastings algorithm.

5.4.1 Gibbs Sampling

Gibbs sampling for multi-variate probability distributions works by sampling sequentially from full conditional distributions on each parameter given the current state of the other parameters. Algorithmically it works as follows. We suppose that the distribution of interest, $p(\boldsymbol{\theta}|\mathbf{x})$, depends on a multi-dimensional parameter vector, $(\theta) = (\theta_1, \theta_2, \ldots, \theta_p)$. We use $(\theta)^k$, θ_i^k to denote the value of the full parameter vector and its i'th component at iteration k of the algorithm. We denote by $\boldsymbol{\theta}_{(i)}$ the vector of all parameter values except the i'th and use $p(\theta_i|\boldsymbol{\theta}_{(i)}, \mathbf{x})$ to denote the full conditional distribution of θ_i, given the values of all the other components and the data. If the value of the Markov chain at step t is $\boldsymbol{\theta}^t$, then the value at step $t+1$ is obtained via

- Sample θ_1^{t+1} from $p(\theta_1|\theta_2^t, \theta_3^t, \ldots, \theta_p^t, \mathbf{x})$.
- Sample θ_2^{t+1} from $p(\theta_2|\theta_1^{t+1}, \theta_3^t, \ldots, \theta_p^t, \mathbf{x})$.

- .....
- Sample θ_i^{t+1} from $p(\theta_i|\theta_j^{t+1}$ for $j < i$ and θ_j^t for $j > i, \mathbf{x})$.
- .....
- Sample θ_p^{t+1} from $p(\theta_p|\theta_1^{t+1}, \ldots, \theta_{p-1}^{t+1}, \mathbf{x})$.

This set of sequential updates is repeated at each iteration of the algorithm to generate a set of samples from the target distribution.

The transition kernel in Gibbs sampling is

$$\mathcal{K}_G(\boldsymbol{\theta}^{t+1}|\boldsymbol{\theta}^t) = \prod_{i=1}^{k} p(\theta_i|\theta_j^{t+1} \text{ for } j < i \text{ and } \theta_j^t \text{ for } j > i, \mathbf{x})$$

which satisfies detailed balance with target distribution $p(\boldsymbol{\theta}|\mathbf{x})$.

5.4.2 Metropolis-Hastings algorithm

In the Metropolis-Hastings algorithm all the parameters of the model are typically updated simultaneously. This is achieved using a **proposal distribution**, $q(\boldsymbol{\phi}|\boldsymbol{\theta})$, to propose a new point $\boldsymbol{\phi}$, given the current parameter values $\boldsymbol{\theta}$. The algorithm is as follows

1. Initialise $\boldsymbol{\theta}^0$ by drawing from a distribution of starting vlaues (often the prior can be used for this).
2. At step t:
   (a) Propose a new point $\boldsymbol{\phi} \sim q(\boldsymbol{\phi}|\boldsymbol{\theta}^{t-1})$.
   (b) Compute the **acceptance probability**

$$\alpha = \min\left(1, \frac{p(\boldsymbol{\phi}|\mathbf{x})q(\boldsymbol{\theta}^{t-1}|\boldsymbol{\phi})}{p(\boldsymbol{\theta^{t-1}}|\mathbf{x})q(\boldsymbol{\phi}|\boldsymbol{\theta}^{t-1})}\right).$$

   (c) Draw $u \sim U[0,1]$. If $u < \alpha$, set $\boldsymbol{\theta}^t = \boldsymbol{\phi}$, otherwise set $\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1}$.
3. Repeat until the desired number of iterations, T, have been completed.

the initial version of this algorithm, due to Metropolis, used symmetric proposal distributions and so that factor cancels out of the acceptance probability. A subsequent paper by Metropolis and Hastings generalised the result to non-symmetric proposals.

It can be readily verified in this case as well that the Markov chain constructed in this way satisfies detailed balance with target distribution equal to the posterior $p(\boldsymbol{\theta}|\mathbf{x})$.

There are a few special cases of the Metropolis-Hastings algorithm

- **The Metropolis Algorithm** This is the case described above where the proposal is symmetric, $q(\boldsymbol{\phi}|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\phi})$, and the acceptance probability reduces to

$$\alpha = \min\left(1, \frac{p(\boldsymbol{\phi}|\mathbf{x})}{p(\boldsymbol{\theta^{t-1}}|\mathbf{x})}\right).$$

- **Random Walk Metropolis** If we use $q(\boldsymbol{\theta}|\boldsymbol{\phi}) = f(\boldsymbol{\phi} - \boldsymbol{\theta})$, with f some function satisfying $f(\mathbf{y} = f(-\mathbf{y})$, then the kernel driving the chain is a random walk. This is a symmetric proposal and so the accpetance probability is as in the Metropolis Algorithm above.

- **The Independence Sampler** If we take $q(\boldsymbol{\theta}|\boldsymbol{\phi}) = f(\boldsymbol{\theta})$, the candidate value is independent of the current value. The acceptance probability is

$$\alpha = \min\left(1, \frac{w(\boldsymbol{\phi})}{w(\boldsymbol{\theta})}\right)$$

  where $w(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathbf{x})/f(\boldsymbol{\theta})$.

- **Single-updates** Individual parameters of the parameter vector can be updated sequentially in the Metropolis-Hastings algorithm in the same way they are during the Gibbs sampling algorithm. At step t we sequentially propose updates, ϕ_j, to each component, θ_j, of the parameter vector in turn. After updating parameter j, the new parameter vector is $(\theta_1^{t+1}, \ldots \theta_{j-1}^{t+1}, \theta_j, ^{t+1}\, \theta_{j+1}^t, \ldots, \theta_p^t)$. The new value, θ_j^{t+1} is chosen by the algorithm

  1. Propose a new candidate value $\phi_j \sim q(\phi_j|\boldsymbol{\theta}_j^t)$ and set $\boldsymbol{\phi}_j = (\theta_1^{t+1}, \ldots, \theta_{j-1}^{t+1}, \phi_j, \theta_{j+1}^t, \ldots, \theta_p^t)$.
  2. Evaluate the acceptance probability

$$\alpha = \min(1, A), \quad \text{where} \quad A = \frac{p(\boldsymbol{\phi}_j|\mathbf{x})q(\theta_j^t|\boldsymbol{\phi}_j)}{p(\boldsymbol{\theta}_j^t|\mathbf{x})q(\phi_j|\boldsymbol{\theta}_j^t)} = \frac{p(\phi_j|\boldsymbol{\theta}_{(j)}^t, \mathbf{x})q(\theta_j^t|\boldsymbol{\phi}_j)}{p(\theta_j^t|\boldsymbol{\theta}_{(j)}^t, \mathbf{x})q(\phi_j|\boldsymbol{\theta}_j^t)}$$

  3. Draw $u \sim U[0, 1]$. If $u < \alpha$, set $\theta_j^{t+1} = \phi_j$, otherwise set $\theta_j^{t+1} = \theta_j^t$.

5.4.3 MCMC diagnostics

The Markov chain is only guaranteed to converge to the stationary distribution asymptotically so it is natural to ask how many samples are needed before the sample is representative of the posterior. The first issue to address is **burn-in**. A Markov chain retains some memory of its initial state for a number of iterations. If the initial sample is in a region of low probability in the stationary distribution, then the first samples will typically not be very characteristic of the stationary distribution. These initial samples should be discarded and samples only retained after the initial burin-in period used for inference. Typically between a few hundred and a few thousand burn-in samples are required and it can be diagnosed using a **trace plot**, which is a plot of the parameter value in the chain as a function of iteration number. Initially the trace plot will show a trend as the chain moves toward parameter values with high posterior support. Once the chain is sampling properly, the values will oscillate back and forth. This is illustrated in Figure 18. The trace plot allows the burn-in period to be identified and removed, and is also a useful diagnostic of the performance of the algorithm. Chains that are moving back and forth rapidly are sampling well from the posterior.

MCMC samples are used to produce Monte Carlo estimates of parameters of interest. If the samples were independent draws from the posterior then these estimates are unbiased and would have a variance that scales like σ^2/M, where σ^2 is the variance of a single sample and M is the number of samples. This could in principle be used to estimate how many

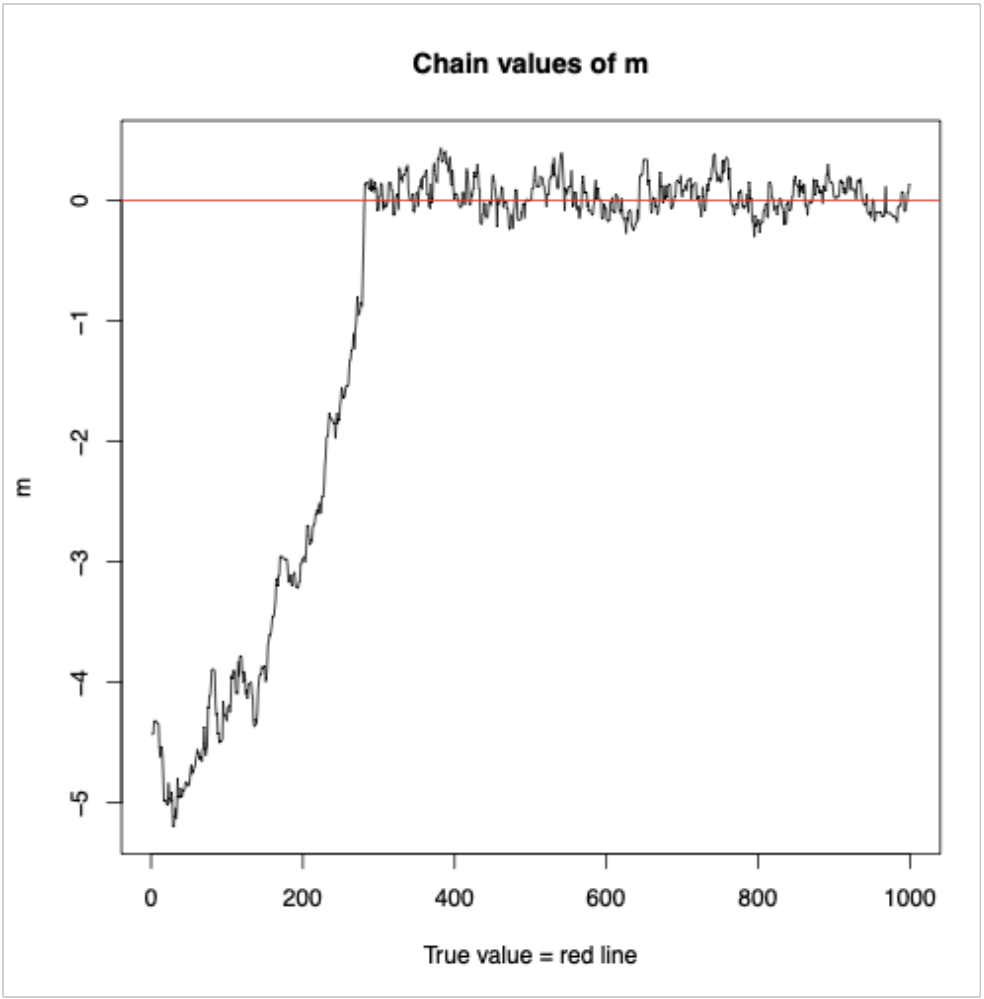


Figure 18: Trace plot for burn-in period of a chain. Initially the chain moves form the starting point to the region of high probability density, so there is a tendency to move in a particular direction. Once the chain reaches the correct region it oscillates back and forth in the region of high posterior support.

samples are needed to achieve a certain target precision on a quantity of interest. However, MCMC samples are not independent. This modifies the variance estimate to

$$\sigma^2 = \mathrm{var}_p(\theta) + 2\sum_{k=2}^{\infty} \mathrm{cov}(\theta^1, \theta^k).$$

This is difficult to compute in practice, so what is usually done is to generate m different chains of length M, estimate the value of the quantity of interest in each one, $\bar{f}_1, \ldots, \bar{f}_m$, compute the value using the pooled samples from all chains, $\bar{f}$, and then construct the *batch means estimate*

$$\hat{\sigma}^2 = \frac{T}{m-1}\sum_{i=1}^{m}(\bar{f}_i - \bar{f})^2.$$

The estimated Monte Carlo error in f is then $\hat{\sigma}^2/n$.

Correlation in MCMC samples can also be estimated using the **autocorrelation function** (ACF). The *lag-k autocorrelation coefficient* or *autocorrelation at lag-k* is $\mathrm{cov}(\theta^i, \theta^{i+k})$ and computed via

$$\rho_k = \frac{\sum_{i=1}^{N-k}(\theta^i - \bar{\theta})(\theta^{i+k} - \bar{\theta})}{\sum_{i=1}^{M}(\theta^i - \bar{\theta})^2}$$

where θ now denotes one parameter of the target distribution, and $\bar{\theta}$ is the mean of that parameter in the chain. Looking at ACF plots is another useful diagnostic of MCMC performance. Examples of good, bad and normal ACF plots are given in Figure 19.

If MCMC chains have very high lags, most likely they are not taking big enough jumps in parameter space and so the size of proposed jumps should be increased. It is typical to monitor **acceptance rates** when using the Metropolis-Hastings algorithm and a target acceptance rate is used to adjust proposed jump sizes. If proposed jumps are too small, the acceptance rate will be high but there will also be high autocorrelation between samples. If the proposed jumps are too large, the acceptance rate will be low, but those samples that are accepted will show very low autocorrelation.Ultimately we care about maximising the rate at which we obtain new independent samples. This can be estimated by tracking the **effective sample size**

$$\mathrm{ESS} = \frac{M}{1 + 2\sum_{k=1}^{\infty}\rho_k}$$

where M is the number of samples in the chain. It has been shown that, under certain assumptions, the optimal rate of obtaining new effective samples is achieved by aiming to have an acceptance rate around 23.4%.

The final diagnostic we will mention here is the use of multiple chains. For complex probability distributions that have many modes it is possible for Markov chains to get stuck sampling from only one of them. Chains starting from different points in parameter space may end up exploring different modes. As a diagnostic of this kind of behaviour, it is good practice to run a handful of runs, starting at different points in parameter space. We can be confident in the final results once the different chains are producing samples that are consistent with one another. This consistency can be quantified using the **Gelman-Rubin statistic**.

Suppose we have m independent chains and have discarded the initial burn-in samples

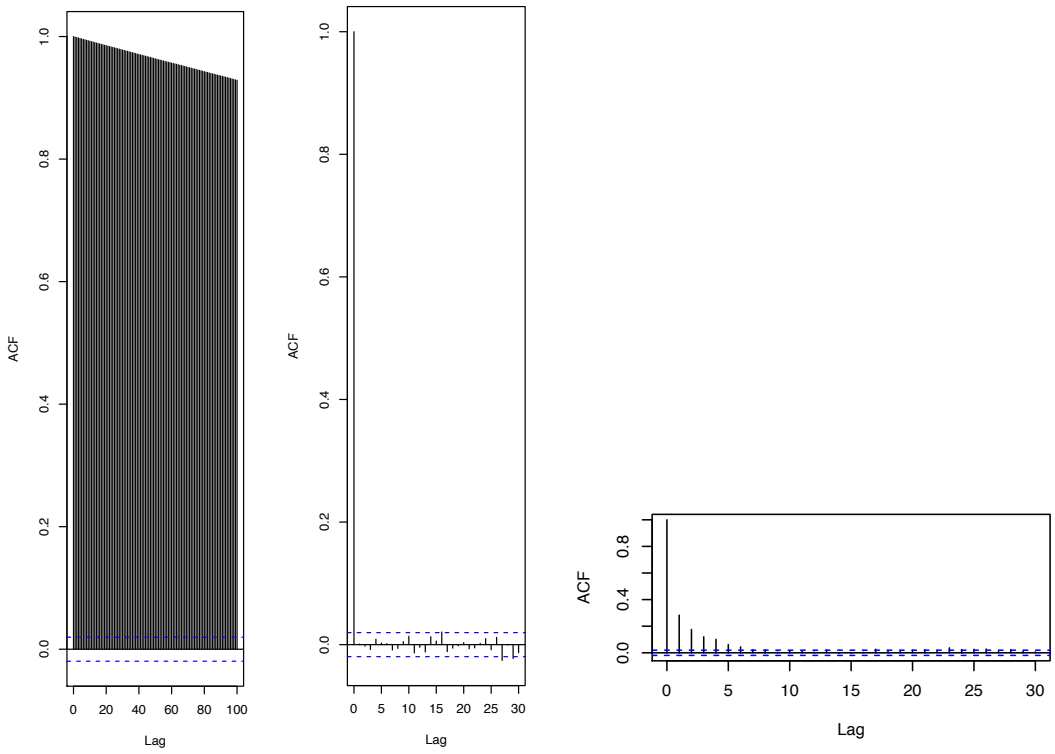


Figure 19: Examples of plots of the autocorrelation function. This should decline to numbers close to 0 for short lags. In the left hand plot, the ACF is still above 0.8 at a lag of 100, indicating highly correlated samples, which is not desirable. In the middle plot we show an ideal example where the ACF is already close to zero at lag of 1, indicating a high level of independence in the samples. The right hand plot is a typical example of MCMC chains that are sampling well. The ACF falls to low values for lags of a few.

to leave chains of length N. We calculate the *within chain variance*

$$W = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{N-1}\sum_{i=1}^{N}(\theta_{ij}-\bar{\theta}_j)^2$$

where θ_{ij} is the i'th sample in the j'th chain. We similarly define the *between chain variance*

$$B = \frac{N}{m-1}\sum_{j=1}^{m}(\bar{\theta}_j-\bar{\bar{\theta}})^2, \quad \text{where } \bar{\bar{\theta}} = \frac{1}{m}\sum_{j=1}^{m}\bar{\theta}_j.$$

Note that we are assuming that θ is a one-dimensional parameter, which could be one component of a multi-dimensional parameter vector. The variance in this parameter can be computed as

$$\text{var}(\theta) = \left(1-\frac{1}{N}\right)W + \frac{1}{N}B$$

from which the **potential scale-reduction factor** can be computed

$$\hat{R} = \sqrt{\frac{\text{var}(\theta)}{W}}.$$

Values of R greater than about 1.1 or 1.2 indicate that the chains are not yet converged.

5.4.4 Speeding up MCMC

MCMC can be made faster by a good choice of the proposal distribution. Proposal distributions that are well approximated to the form of the target distribution are to be preferred. As well as tuning the proposal distribution, accelerated convergence can be achieved using **annealing**. The idea of annealing is to transform the posterior surface as

$$p(\boldsymbol{\theta}|\mathbf{x}) \to [p(\boldsymbol{\theta}|\mathbf{x})]^{\beta}, \quad \text{where } \beta = \frac{1}{kT}.$$

As $T \to \infty$ the new distribution becomes flatter and flatter, so the contrast in probabilities between different points is reduced. This means that moves proposed in a Metropolis-Hastings algorithm are more likely to be accepted. Figure 20 shows the effect of the annealing transformation on the probability distribution being sampled as the temperature increases.

There are two common applications of annealing. In **simulated annealing** the temperature is gradually changed as the initial phase of the run progresses, according to some scheme, for example, a linear decrease with iteration number. The idea is that in the early phase the chain explores the parameter space widely and rapidly, identifying areas of higher posterior density. As the temperature decreases the chain gets trapped in a region of high posterior probability, hopefully the primary mode of the distribution. The simulated annealing phase does not produce useful samples, since detailed balance is satisfied, but after the simulated annealing phase, the chain will evolve as normal and return valid samples from the posterior.

The other use of annealing is **parallel tempering**. In parallel tempering, a number of chains are evolved simultaneously at different temperatures. At each iteration, a given chain will update its parameters as normal, but with a certain probability an interchange is proposed, in which the states of two chains (usually neighbouring in temperature) will

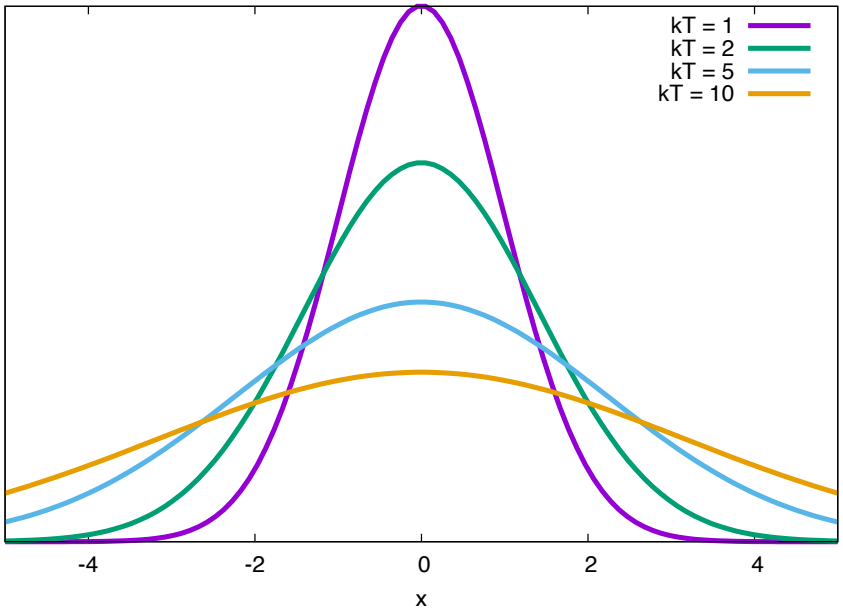


Figure 20: Effect of annealing on the target probability distribution.

be exchanged. If the two chains are labelled i and j, have temperatures T_i and T_j, and current parameter values $\boldsymbol{\theta}^i$ and $\boldsymbol{\theta}^j$, then the appropriate acceptance probability for the swap $\boldsymbol{\theta}^i \leftrightarrow \boldsymbol{\theta}^j$ is

$$\alpha = \min\left(1, \left[\frac{p(\boldsymbol{\theta}^j|\mathbf{x})}{p(\boldsymbol{\theta}^i|\mathbf{x})}\right]^{\frac{1}{T_i}} \left[\frac{p(\boldsymbol{\theta}^i|\mathbf{x})}{p(\boldsymbol{\theta}^j|\mathbf{x})}\right]^{\frac{1}{T_j}}\right).$$

The idea of parallel tempering is that higher posterior density regions of the parameter space that the widely-exploring high temperature chains identify, propagate down to lower temperature chains, which explore them thoroughly. Efficiency is dependent on the difference in the temperatures of neighbouring chains, so the number of chains and their spacing must be tuned for each given problem.

5.5 Posterior computation: variable model dimension

In some circumstances we might be interested in fitting multiple different models to the data simultaneously. the most common situation is when the total number of parameters needed to describe the data is unknown. In a gravitational wave context this arises when the total number of sources present in the data set is unknown, e.g., for the LISA gravitational wave detector. In these circumstances one can still construct Markov chains, but now these chains can move between different models. The fraction of samples that the chain spends in each model is proportional to the evidence for that model and, in the case of models that differ only in the total number of sources, the evidences give the relative probabilities for the unknown number of sources in the data.

The most widely used algorithm for fitting multiple models is **reversible jump Markov**

chain Monte Carlo (RJMCMC). RJMCMC generates a Markov chain such that at each step either an update within the model is proposed, or, with a certain probability, a jump to an alternative model is proposed. Usually the jumps are between models that differ by only one source if that is the type of model hierarchy being considered. When proposing a jump to a new model, with parameters $\boldsymbol{\theta}'$, the values of the parameters of that model must also be proposed. This is achieved by generating a set of random numbers $\mathbf{u}$ from some distribution $q(\mathbf{u})$. In order to ensure reversibility we imagine that these random numbers are part of the parameters of the model, but because they are random we only need to generate them when they are used in a between-model jump. Similarly we may need some random variables $\mathbf{u}'$ to propose jumps back form the new model space to the original model parameters $\boldsymbol{\theta}$. The dimensionality of the joint space $(\boldsymbol{\theta}, \mathbf{u})$ must equal that of $(\boldsymbol{\theta}', \mathbf{u}')$ and there will be a deterministic, invertible mapping between the two. In the case of nested model, the reverse jump might just delete a set of parameters and so the dimensionality of $\mathbf{u}'$ is 0. However, if the particular source is deleted at random rather than, say, the lowest SNR source always being deleted, a random variable that selects which source to delete is required. The generalisation of the acceptance probability for RJMCMC is

$$\alpha = \min\left(1, \frac{p(\boldsymbol{\theta}'|\mathbf{x})q(\boldsymbol{\theta}')}{p(\boldsymbol{\theta}|\mathbf{x})q(\boldsymbol{\theta})}\left|\frac{\partial(\boldsymbol{\theta}', \mathbf{u}')}{\partial(\boldsymbol{\theta}, \mathbf{u})}\right|\right)$$

where the last term is the Jacobian for the transformation between the two sets of variables.

Example: mixture of Gaussians Suppose that model M_1 is a single Gaussian with mean θ_1 and unit variance and model M_2 is a mixture of two Gaussians with means θ_1' and θ_2' and both of unit variance. We have random variables $\mathbf{u} = (u_1, u_2)$ with $u_1 \sim N(0, \sigma_0^2)$ and $u_2 \sim U[0,1]$ in the M_1 model space and $\mathbf{u}' = u_1' \sim U[0,1]$ in the M_2 model space. The random variable u_1 gives the value of the mean of the new Gaussian to be added, while u_1' selects which Gaussian to delete in the reverse step. The second random variable u_2 ensures the dimensionality is consistent. We can define the mapping between the parameter spaces via

$$\begin{aligned}
\theta_1' &= \begin{cases} \theta_1 & \text{if } u_2 < 0.5 \\ u_1 \text{if } u_2 \geq 0.5 \end{cases} \\
\theta_2' &= \begin{cases} u_1 & \text{if } u_2 < 0.5 \\ \theta_1 \text{if } u_2 \geq 0.5 \end{cases} \\
u_1' &= u_2.
\end{aligned}$$

and the reverse mapping

$$\begin{aligned}
\theta_1 &= \begin{cases} \theta_1' & \text{if } u_2 < 0.5 \\ \theta_2' \text{if } u_1' \geq 0.5 \end{cases} \\
u_1 &= \begin{cases} \theta_2' & \text{if } u_2 < 0.5 \\ \theta_1' \text{if } u_1' \geq 0.5 \end{cases} \\
u_2 &= u_1'.
\end{aligned}$$

The Jacobian for this transformation is 1 and so the acceptance probability is just

$$\alpha = \min\left(1, \frac{p(\boldsymbol{\theta}'|\mathbf{x})q(\boldsymbol{\theta}')}{p(\boldsymbol{\theta}|\mathbf{x})q(\boldsymbol{\theta})}\right).$$

5.6 Evidence computation

As described earlier, the Bayesian evidence is required for model comparison and Bayesian model selection, but it is difficult to compute accurately using standard MCMC methods. **Nested sampling** (Skilling 2004) was developed as an alternative approach, specifically tuned for evidence computation. It calculates the evidence by transforming the multi–dimensional evidence integral into a one–dimensional integral that is easy to evaluate numerically. This is accomplished by defining the prior volume X as $dX = \pi(\mathbf{\Theta})d^D\mathbf{\Theta}$, so that

$$X(\lambda) = \int_{\mathcal{L}(\mathbf{\Theta})>\lambda} \pi(\mathbf{\Theta})d^N\mathbf{\Theta}, \tag{86}$$

where the integral extends over the region(s) of parameter space contained within the iso-likelihood contour $\mathcal{L}(\mathbf{\Theta}) = \lambda$. The evidence integral, Eq. (**??**), can then be written as

$$\mathcal{Z} = \int_0^1 \mathcal{L}(X)dX, \tag{87}$$

where $\mathcal{L}(X)$, the inverse of Eq. (86), is a monotonically decreasing function of X. Thus, if one can evaluate the likelihoods $\mathcal{L}_i = \mathcal{L}(X_i)$, where X_i is a sequence of decreasing values,

$$0 < X_M < \cdots < X_2 < X_1 < X_0 = 1, \tag{88}$$

as shown schematically in Fig. 21, the evidence can be approximated numerically using standard quadrature methods as a weighted sum

$$\mathcal{Z} = \sum_{i=1}^{M} \mathcal{L}_i w_i, \tag{89}$$

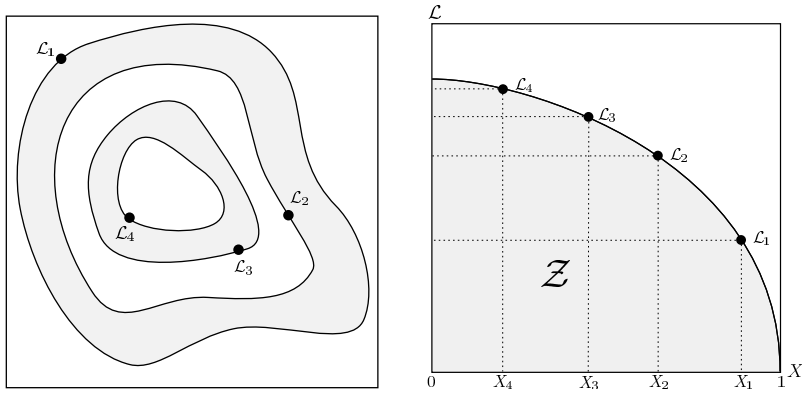
where the weights w_i for the simple trapezium rule are given by $w_i = \frac{1}{2}(X_{i-1} - X_{i+1})$. An example of a posterior in two dimensions and its associated function $\mathcal{L}(X)$ is shown in Fig. 21.

5.6.1 Evidence Evaluation

The summation in Eq. (89) is performed as follows. The iteration counter is first set to $i = 0$ and N 'active' (or 'live') samples are drawn from the full prior $\pi(\mathbf{\Theta})$, so the initial prior volume is $X_0 = 1$. The samples are then sorted in order of their likelihood and the smallest (with likelihood $\mathcal{L}_0$) is removed from the active set (hence becoming 'inactive') and replaced by a point drawn from the prior subject to the constraint that the point has a likelihood $\mathcal{L} > \mathcal{L}_0$. The corresponding prior volume contained within this iso-likelihood contour will be a random variable given by $X_1 = t_1 X_0$, where t_1 follows the distribution $\mathbb{P}(t) = Nt^{N-1}$ (i.e., the probability distribution for the largest of N samples drawn uniformly from the interval $[0, 1]$). At each subsequent iteration i, the removal of the lowest likelihood point $\mathcal{L}_i$ in the active set, the drawing of a replacement with $\mathcal{L} > \mathcal{L}_i$ and the reduction of the corresponding prior volume $X_i = t_i X_{i-1}$ are repeated, until the entire prior volume has been traversed. The algorithm thus travels through nested shells of likelihood as the prior volume is reduced. The mean and standard deviation of $\log t$, which dominates the geometrical exploration, are:

$$E[\log t] = -1/N, \quad \sigma[\log t] = 1/N. \tag{90}$$

Since each value of $\log t$ is independent, after i iterations the prior volume will shrink down such that $\log X_i \approx -(i \pm \sqrt{i})/N$. Thus, one takes $X_i = \exp(-i/N)$.



(a) (b)

Figure 21: Cartoon illustrating (a) the posterior of a two dimensional problem; and (b) the transformed $\mathcal{L}(X)$ function where the prior volumes X_i are associated with each likelihood $\mathcal{L}_i$.

5.6.2 Stopping Criterion

The nested sampling algorithm should be terminated on determining the evidence to some specified precision. One way would be to proceed until the evidence estimated at each replacement changes by less than a specified tolerance. This could, however, underestimate the evidence in (for example) cases where the posterior contains any narrow peaks close to its maximum. Skilling provides an adequate and robust condition by determining an upper limit on the evidence that can be determined from the remaining set of current active points. By selecting the maximum–likelihood $\mathcal{L}_{\max}$ in the set of active points, one can safely assume that the largest evidence contribution that can be made by the remaining portion of the posterior is $\Delta\mathcal{Z}_{\rm i} = \mathcal{L}_{\max} X_{\rm i}$, i.e. the product of the remaining prior volume and maximum likelihood value. We choose to stop when this quantity would no longer change the final evidence estimate by some user–defined value (we use 0.5 in log–evidence).

5.6.3 Posterior Inferences

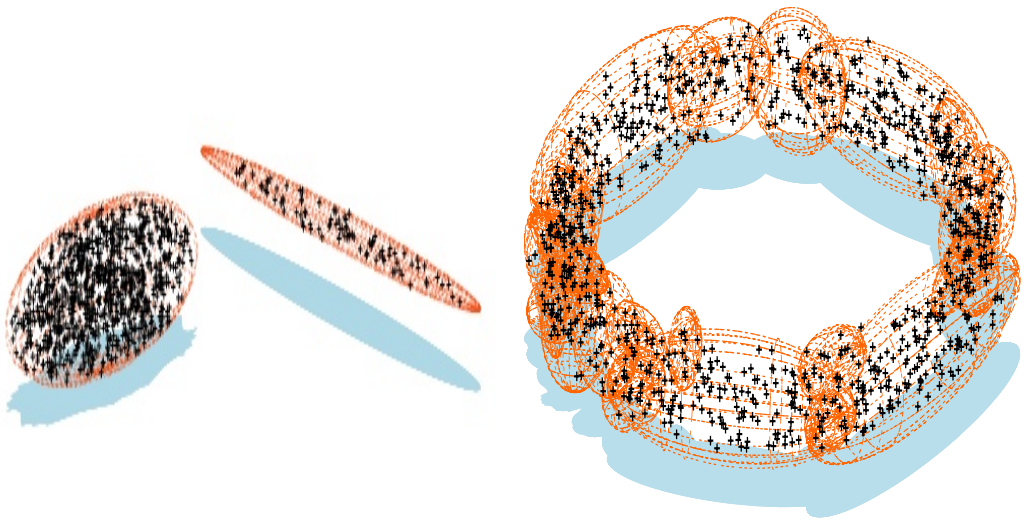
Once the evidence $\mathcal{Z}$ is found, posterior inferences can be easily generated using the final live points and the full sequence of discarded points from the nested sampling process, i.e., the points with the lowest likelihood value at each iteration i of the algorithm. Each such point is simply assigned the probability weight

$$p_i = \frac{\mathcal{L}_i w_i}{\mathcal{Z}}. \tag{91}$$

These samples can then be used to calculate inferences of posterior parameters such as means, standard deviations, covariances and so on, or to construct marginalised posterior distributions.

5.6.4 MultiNest Algorithm

The most challenging task in implementing the nested sampling algorithm is drawing samples from the prior within the hard constraint $\mathcal{L} > \mathcal{L}_i$ at each iteration i. Employing a



(a) (b)

Figure 22: Illustrations of the ellipsoidal decompositions performed by MultiNest. The points given as input are overlaid on the resulting ellipsoids. 1000 points were sampled uniformly from: (a) two non-intersecting ellipsoids; and (b) a torus.

naive approach that draws blindly from the prior would result in a steady decrease in the acceptance rate of new samples with decreasing prior volume (and increasing likelihood). The MultiNest algorithm tackles this problem through an ellipsoidal rejection sampling scheme by enclosing the live point set within a set of (possibly overlapping) ellipsoids and a new point is then drawn uniformly from the region enclosed by these ellipsoids. The number of points in an individual ellipsoid and the total number of ellipsoids is decided by an 'expectation–maximization' algorithm so that the total sampling volume, which is equal to the sum of volumes of the ellipsoids, is minimized. This allows maximum flexibility and efficiency by breaking up a mode resembling a Gaussian into a relatively small number of ellipsoids, and if the posterior mode possesses a pronounced curving degeneracy so that it more closely resembles a (multi–dimensional) 'banana' then it is broken into a relatively large number of small 'overlapping' ellipsoids (see Fig. 22).

The ellipsoidal decomposition scheme described above also provides a mechanism for mode identification. By forming chains of overlapping ellipsoids (enclosing the live points), the algorithm can identify distinct modes with distinct ellipsoidal chains, e.g., in Fig. 22 panel (a) the algorithm identifies two distinct modes while in panel (b) the algorithm identifies only one mode as all the ellipsoids are linked with each other because of the overlap between them. Once distinct modes have been identified, they are evolved independently.

Another feature of the MultiNest algorithm is the evaluation of the global as well as the 'local' evidence values associated with each mode. These evidence values can be used in calculating the probability that an identified 'local' peak in the posterior corresponds to a real object.

There are many other nested sampling algorithms around today, including Polychord, which obtains samples from within the iso-likelihood surface through slice sampling, cpnest and dynesty. The latter two samplers form part of the Bilby parameter estimation software suite for LIGO.