1. Introduction

The development of high throughput assays such as short read sequencing are driving cancer biology to become a data rich scientific field. Analysing this data and extracting biological meaning is becoming increasingly challenging as large data sets filled with biological and technical noise are created. There are many approaches computational biologists can use to meet this challenge. In these notes we will consider how probabilistic modelling can be useful. We will see that probabilistic modelling provides a coherent framework to analyse noisy data and extract interpretable results. We adopt the Bayesian view of statistics in these notes, where model parameters are treated as random variables. The Bayesian view provides a systematic framework to construct and fit probabilistic models.

1.1. Why do we need probabilistic models

Real world data is not perfect, there is always "noise" whether it be from measurement error or some underlying variation. This is especially true in biology with the development of high throughput assays such as short read sequencing. As we develop new methods such as single cell sequencing there is a constant tension between throughput and accuracy. For example, when we sequence single cells we often have a choice. Do we sequence more cells with low coverage or fewer cells with high coverage. Lower coverage data will have more technical noise but allow us to survey a broader population and account for biological variability. How can we make use of such data in the presence of this noise?

In the simplest sense probabilistic modelling is about looking at the world when there is noise. More accurately it is about constructing models which account for some form of randomness. A common example across many scientific fields is to assume the observed values are affected by "random" noise. This assumption stems from issues such as measurement error. While we believe there is some true value we wish to measure, the instruments available to make measurements intrinsically perturb this value. We can't learn a great deal from a single measurement, but if we make a lot measurements then we can average them to get a more accurate estimate. Of course, the confidence we have in this measurement will be dependent on the variation we observe in the data. Classically, we report such uncertainty with quantities like confidence intervals. If we adopt such a process we are actually constructing a simple probabilistic model. We are assuming the data is sampled from a Normal distribution with a mean and variance parameter that we don't know. Roughly speaking our average is the estimate of the mean (the true parameter) and the confidence interval an estimate of the variance (noise).

Probabilistic modelling formalises the intuition of the last paragraph. We construct models where we encode quantities of interest as parameters and observations as random samples from a distribution that depends on these parameters. Models can be complex and have 1000s of parameters and 1,000,000s of observations. The challenge for computational biologists is how to construct and fit these models. In these notes we will look at few key problems.

1. How do we construct an appropriate model for the data?
2. How can we estimate the model parameters?
3. How do we report these model parameters i.e. how certain are we?

In many ways, the first point is the most challenging. Constructing useful probabilistic models requires an understanding of the problem domain. The most critical thing that needs to be identified is: **What is the question?** While this may seem simple, precisely defining what you are trying to ask is the single most important step in data analysis and is often poorly done. Once you have a question identified you then need to leverage knowledge of the domain to begin constructing models. What are biologically realistic assumptions? What is the data and what are the types of noise it exhibits? Building models which encapsulate this information is as much an art as science. Models that are too simple risk fitting the data poorly, whereas models that are too complex risk becoming impossible to interpret. In the next section we will discuss some basic ingredients to building models.

The second and third point are somewhat more straightforward to address. We will adopt a Bayesian viewpoint in these notes. The clear advantage of the Bayesian method is that it provides a simple prescription for how to estimate model parameters. *Everything you want to know is encapsulated in the posterior distribution*. The hard part then is not how to estimate the model parameters or report uncertainty, but how to compute the posterior. For reasons to be discussed later this is computationally challenging.

One factor that is often overlooked is the interplay between model construction and inference. If we build a model that we cannot fit, then it is useless. Conversely, if we do not know how to fit a model we will not construct it. Thus, having an extensive toolbox of methods for fitting models opens up broader modelling possibilities.

1.2. Probability distributions

The basic ingredient of probabilistic modelling is probability theory. As we will see later, we will make assumptions about the distributions that govern the data and model parameters. To be able to do this we need to have a reasonable knowledge of the distributions out there. In this section we will review a few common distributions, though this is by no means an exhaustive. Before we do that we discuss a basic rule of thumb that can help guide the selection of distribution.

1.2.1. Picking a distribution

The most fundamental question that needs to be addressed when choosing a distribution for modelling is: *What possible values can the quantity have?* For example, if your observed data can only take values in the positive integers (such as read depth), then a Normal distribution will be a poor choice to use in the model. Recall the set of possible values (formally the support of the distribution) of a Normal random variable is $\mathbb{R}$, all possible real numbers. A much better choice in this case is the Poisson distribution which can take values in $\mathbb{Z}^+$, the set of all non-negative integers.

This simple rule of thumb is surprisingly powerful, it at least guarantees the model makes some sense. Of course it is just a rule of thumb. There are many distributions which have support in $\mathbb{Z}^+$, the Negative-Binomial (NB) being another popular choice. Why you would prefer the Poisson or NB then comes down to several other factors. The most important is how variable is the data? The NB is a two parameter distribution, whereas the Poisson has only a single parameter. The extra parameter in the NB provides more flexibility in modelling the variance of the data. In fact, the NB is what is called over-dispersed relative to the Poisson. There are many pairs of distributions which have this relationship. Perhaps the most famous pair is the Normal and Student-t (over-dispersed) distributions.

There are other factors such as distribution skew i.e. is the distribution symmetric and tail behaviour i.e. do we have more outliers than we expect. Knowing which distribution to use is often an empirical question, and rigorously should be performed as part of model selection. In practice it is best to start with the simplest (fewest parameter) distribution, that is easiest to interpret. If you observe a poor fit to the data, for example your model predictions deviate wildly from expectation, then begin to consider more complex choices.

1.3. Bayesian inference

Once you have constructed a model you need to *fit* the model to the data. What fit means depends on your chosen paradigm. In statistics there are two major paradigms, the frequentist and Bayesian paradigms. There are philosophical and practical differences between then two. We adopt the Bayesian paradigm in this work. The main reasons for doing do are:

- Prior information - There is a natural mechanism to incorporate existing knowledge into the Bayesian framework through the use of a prior distribution.
- Hierarchical models - Model parameters are random variables in the Bayesian paradigm. This means that they can have distributions which govern their behaviour. The parameters in these distributions can in turn have their own distributions and so on. This allows for the construction of powerful hierarchies of variables that can be exploited to *share statistical* strength.

- Coherent inference - As we will see, fitting model parameters in the Bayesian paradigm amounts to computing the posterior distribution.

1.3.1. The posterior distribution

Abstractly, we assume that we have observed data X and a model with a collection of parameters θ. The observed data X is modelled by a distribution which depends on θ, often called the *likelihood*. We denote the density of this distribution $p(X|\theta)$. In the Bayesian paradigm we also have a distribution for the parameters, referred to as the *prior distribution* $p(\theta)$. The prior distribution reflects our belief about the parameters before observing the data. Our goal is then to compute the *posterior distribution* $p(\theta|X)$. The posterior distribution describes our belief about the parameter values after observing the data. Computing the posterior is a trivial exercise in applying Bayes' rule.

$$p(\theta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} \tag{1}$$

$$p(X) = \int p(X|\theta)\,p(\theta)\,\mathrm{d}\theta \tag{2}$$

While it is conceptually simple to apply equation 1 to obtain the posterior, it is computationally challenging. The main issue is the need to compute the integral in 2 which is typically high-dimensional and has no closed form. We will see later how we can address this problem using maximum a posterior (MAP) or Monte Carlo Markov Chain (MCMC) methods to approximate the posterior.

The quantity $p(X|\theta)\,p(\theta) = p(X,\theta)$ which appears in the numerator of the posterior is commonly referred to as the *joint distribution*. Note that the joint distribution is proportional to the posterior distribution up to a constant which does not depend on the parameters θ. While the posterior distribution is typically hard to evaluate, the joint distribution is often easy. We make use of this fact frequently when fitting models.

Example 1.

We consider the simple problem of inferring the probability a coin will show a head when flipped. Assume we have coin with probability ρ of showing heads and we perform n coin flips, x of which show heads. A reasonable likelihood for this problem is a Binomial with parameters n and ρ.

If we assume no existing knowledge then a reasonable prior distribution is a continuous distribution. More generally we can assume that we ρ follows a Beta distribution with parameters a and b. If we set $a = b = 1$ we recover the Uniform distribution. Alternatively a value $a = b = 2$ would reflect a weak prior belief that ρ is near 0.5 and the coin is fair.

Thus the model we have is

$$\begin{aligned} \rho &\sim \text{Beta}(\cdot|a,b) \\ x|n,\rho &\sim \text{Binomial}(\cdot|n,\rho) \end{aligned}$$

if this notation is unfamiliar see the section on hierarchical modelling. We can write down the distributions

$$\begin{aligned} p(\rho) &= \frac{1}{\mathcal{B}(a,b)}\,\rho^{a-1}\,(1-\rho)^{b-1}\,\mathbb{I}(\rho \in [0,1]) \\ p(x|n,\rho) &= \binom{n}{x}\,\rho^{x}\,(1-\rho)^{n-x} \end{aligned}$$

and we have the joint distribution

$$\begin{aligned} p(x,n,\rho) &= p(x|n,\rho)\,p(\rho) \\ &= \binom{n}{x}\,\rho^{x}\,(1-\rho)^{n-x}\,\frac{1}{\mathcal{B}(a,b)}\,\rho^{a-1}\,(1-\rho)^{b-1} \\ &= \frac{\binom{n}{x}}{\mathcal{B}(a,b)}\rho^{x+a-1}\,(1-\rho)^{n-x+b-1} \end{aligned}$$

To compute the posterior we need to evaluate

$$\begin{aligned} p(\rho|x,n) &= \frac{p(x,n,\rho)}{\int p(x,n,\rho)\,\mathrm{d}\rho} \\ &= \frac{\frac{\binom{n}{x}}{\mathcal{B}(a,b)}\rho^{x+a-1}(1-\rho)^{n-x+b-1}}{\int \frac{\binom{n}{x}}{\mathcal{B}(a,b)}\rho^{x+a-1}(1-\rho)^{n-x+b-1}\,\mathrm{d}\rho} \\ &= \frac{\rho^{x+a-1}(1-\rho)^{n-x+b-1}}{\int \rho^{x+a-1}(1-\rho)^{n-x+b-1}\,\mathrm{d}\rho} \end{aligned}$$

Now the challenge is to evaluate the integral in the denominator. We note a trick here, the term $\rho^{x+a-1}(1-\rho)^{n-x+b-1}$ is exactly the same term that appears in a Beta$(x+a, n-x+b)$ distribution neglecting the normalisation constant. So it follows the integral equals the normalisation constant $\mathcal{B}(x+a, n-x+b)$. So we have

$$p(\rho|x,n) = \frac{1}{\mathcal{B}(x+a,n-x+b)}\,\rho^{x+a-1}(1-\rho)^{n-x+b-1}$$

or alternatively

$$\rho|x,n \sim \text{Beta}(\cdot|x+a,n-x+a)$$

In the previous example we could explicitly compute the normalisation constant. This is possible because the Beta and Binomial distributions are a *conjugate* pair. Conjugacy plays an important role in many Bayesian models, making exact computation of the posterior possible. However, there are many models which are non-conjugate and for which the normalisation cannot be compute explicitly. In the previous example this is still not a major problem since we could simply numerically compute the one dimensional integral over ρ. This can be easily done with functions in your favourite scientific computing package. However, if the dimensionality of the integral is even moderately high this becomes computationally prohibitive and alternative approaches will be required.

1.3.2. Summarising posteriors

Once we have a posterior distribution (or approximation) the question becomes how to report the results. While the posterior distribution provides all the information we could want, it is difficult for humans to make sense of complex and multi-dimensional distributions. As a result we often resort to reporting some form of point or interval estimate to summarise the posterior. The simplest quantities to report are the posterior mean and variance. These are often reasonable values, but care must be taken if the posterior is multi-modal. In that case the mean may not be a typical value for the distribution. An alternative approach is to report an α credible region. This is a region which contains α of the mass of the posterior. There are many ways to construct such intervals, one sensible way is to find the smallest region R where $\int_R p(\theta|X)\mathrm{d}\theta = \alpha$. This is referred to as the high probability density (HPD) region, since by definition it captures the region of highest density.

Example 2.

We return to the coin flipping example. From the previous section we know the posterior distribution is

$$p(\rho|x) = \frac{1}{\mathcal{B}(x+a,n-x+b)}\,\rho^{x+a-1}(1-\rho)^{n-x+b-1}$$

it is easy to check the mean of this distribution is $\frac{x+a}{n+a+b}$ and the variance is $\frac{x+a}{(n+a+b)^2}$.

A more general theory of Bayesian point estimation exists and makes use of the concept of a loss function. A loss function, $L(x, y)$, is bi-variate function which takes on positive real values. The interpretation is that the loss function represents our perceived loss if we were to estimate x but the true value is y. In a Bayesian setting we seek to minimise the expected loss under the posterior as our best estimate.

$$\hat{\theta} \;=\; \underset{\theta'}{\operatorname{argmin}} \int L(\theta', \theta)\, p(\theta|X)\, \mathrm{d}\theta$$

Common examples of loss functions for continuous variables include the L^1 and L^2 norm

$$\begin{aligned} L(x, y) &= |x - y| \\ L(x, y) &= \|x - y\|^2 \end{aligned}$$

Other loss functions can be used to estimate summary regions or even distributions. The same basic framework also applies to discrete variables, though the minimisation of the loss can be more difficult if the space is large as we cannot use gradient methods.

1.3.3. Hierarchical models

The theory of Bayesian inference is coherent and elegantly contained in a few basic principles. However, the real utility of the paradigm is the ability to construct complex hierarchical models. This is possible because we view the model parameters as random variables, in the same way we view the data. There are several useful notational constructs for describing hierarchical models. The most explicit is to list out the distributional assumptions. For example, we have many statements of the form $a|b \sim \mathrm{G}(\cdot|b)$. This reads as follows: the distribution of variable a given variable b follows distribution G with parameter b. You may also see this written as $a|b \sim \mathrm{G}(a|b)$ or $a|b \sim \mathrm{G}(b)$. In general the distribution G can depend on b in more complex ways, possibly through some transformation as the next example illustrates.

Example 3.

Here we will describe the basic Bayesian linear regression model. Recall that linear regression assumes we observe covariate data $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ and outcome variables $\boldsymbol{y} = (y_1, ..., y_N)$ for N data points. Here $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{0, 1\}$. Informally we observe some real valued covariates for a data point $\boldsymbol{x}_n$, and we believe these influence the outcome variables $y_n \in \{0, 1\}$. We will also assume there is regression coefficient vector $\boldsymbol{\beta} = (\beta_1, ..., \beta_D) \in \mathbb{R}^D$ that controls the influence of the covariates on the outcome. A common use for this model is to learn predictors to separate data into two classes, such as patients that will respond to treatment ($y_n = 1$) and patients that will not ($y_n = 0$).

Since the variables y_n are $0/1$ valued, a natural distribution to model them is the Bernoulli. We will assume the regression covariates are independent and model β_d as Normally distributed. To link these two items together we need make the assumption the probability $y_n = 1$ is of the form $\sigma(\sum_d \beta_d\, x_{nd})$ where $\sigma(x) = \frac{1}{1 + e^{-x}}$ is the logistic function.

Our model then has the form

$$\begin{aligned} \beta_d|\mu, \sigma^2 &\sim \mathrm{Normal}(\cdot|\mu, \sigma^2) \\ y_n|\boldsymbol{x}_n, \boldsymbol{\beta} &\sim \mathrm{Bernoulli}\Big(\cdot|\sigma\Big(\sum_d \beta_d\, x_{nd}\Big)\Big) \end{aligned}$$

where μ and σ^2 are hyper-parameters we assume known.

By listing out the distributional assumptions we fully specify the model. This notation also lays bare any conditional independence assumptions. Recall variables are conditionally independent if the value of one variable is independent from another given the value of a third. Conditional independence is useful as it allows us to construct modular models, where we need only focus on the distributional assumptions of a small number of variables. It can also have computational implications, for example allowing for the design of Gibbs samplers. Once we are given the conditional distributions, it is straightforward to write down the joint distribution for the model.

Example 4.

Following the previous example we wish to write down the joint distribution

$$\begin{aligned} p(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\beta}, \mu, \sigma^2) &= \prod_{n=1}^{N} p(y_n, \boldsymbol{x}_n, \boldsymbol{\beta}, \mu, \sigma^2) \\ &= \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\beta})\, p(\beta_d|\mu, \sigma^2) \\ &= p(\beta_d|\mu, \sigma^2) \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\beta}) \\ &= \mathcal{N}(\beta_d|\mu, \sigma^2) \prod_{n=1}^{N} \left[\sigma\Big(\sum_d \beta_d x_{nd}\Big)\right]^{y_n} \left[1 - \sigma\Big(\sum_d \beta_d x_{nd}\Big)\right]^{1-y_n} \end{aligned}$$

where $\mathcal{N}$ indicates the Normal distribution density. Now to compute the posterior for $\boldsymbol{\beta}$ we have

$$p(\boldsymbol{\beta}|\boldsymbol{X}, \boldsymbol{y}, \mu, \sigma^2) = \frac{p(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\beta}, \mu, \sigma^2)}{\int p(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\beta}, \mu, \sigma^2)\, \mathrm{d}\boldsymbol{\beta}}$$

However, the integral in the denominator is no longer simple. If D is small then we could attempt to compute it numerically, but for realistic problems we will need to turn to alternative methods.

One other benefit of listing out the distributional assumptions is that it makes it possible to simulate data from the model. This can be particularly useful for performing a quick visual check that the data simulated from the model looks similar to real data. It can also provide a means to explore the behaviour of the model in different parameter regimes, such as high variance or varying numbers of data points. Finally, it can also be a powerful debugging tool to test inference algorithms such as MCMC methods. If the simulated data comes from a distribution that is not very noisy we should expect our inference algorithm to infer values that are close to the known truth.

1.4. Posterior inference

The central quantity in Bayesian inference is the posterior distribution. Typically computing the posterior is hard due to the need to compute the normalisation constant $\int p(X, \theta)\mathrm{d}\theta$. In realistic models the dimensionality of the parameters θ can be high making numerical approximation of the integral using quadrature or similar methods impractical.

As we cannot usually compute the posterior exactly, we will attempt to approximate it accurately. There are many strategies for approximating the posterior distribution. We will use two in these notes. The first is to simply report the parameter values that maximise the posterior, often referred to as maximum a posteriori (MAP) inference. This is equivalent to using a delta function (spike) centred at the MAP parameters as our approximating distribution. The second approach is to use Markov Chain Monte Carlo methods. Here we approximate the posterior distribution using a collection of samples drawn from the posterior distribution.

1.4.1. MAP estimation

MAP estimation is the simplest way to approximate a posterior distribution. It can be viewed as point estimation of parameters. However, a better way to view it is as approximating the posterior with a distribution that is a single spike at the MAP value. This value has probability one while all other values have probability zero. Formally the MAP estimate is defined as follows.

$$\hat{\theta} = \operatorname*{argmax}_{\theta} p(\theta|X)$$

Of course this definition is not very helpful since we cannot compute the posterior $p(\theta|X)$ by assumption. The key insight is that $p(\theta|X) = \frac{p(\theta, X)}{p(X)}$ and that $p(X)$ does not depend on θ. Thus maximising $p(\theta|X)$ is equivalent to maximising the joint distribution $p(\theta, X)$. In many cases we can evaluate the joint since we have that $p(\theta, X) = p(X|\theta)\, p(\theta)$, that is the joint is the product of the prior and likelihood. So in practice we have that

$$\hat{\theta} = \operatorname*{argmax}_{\theta} p(\theta, X)$$

Once we have our MAP estimate, the posterior distribution is given by

$$q(\theta) \;=\; \delta_{\hat{\theta}}(\cdot)$$

The main problem with using MAP estimation is that we cannot quantify the uncertainty in our parameter estimates. In some situations this is not an issue, particularly when we are only interested in predictive performance.

Computing the map estimate is typically straightforward if the model parameters are continuous. In this case we can compute the gradient of the joint distribution and use methods such as gradient descent. If you use tools like Tenserflow which provide automatic differentiation this requires no additional code to be written. If the model parameters are discrete and the state space large then computing the MAP estimate is hard. Unless the problem has a special structure, the only solution is to enumerate all possible values and compute the joint distribution for each value. At that point computing the exact posterior is no harder, as we simply need to sum these values to obtain the normalisation constant.

1.4.2. Monte Carlo simulation

MAP estimation provides a poor approximation to the posterior. A much better approximation can be obtained using Markov Chain Monte Carlo (MCMC) methods. To motivate MCMC methods remember that we want to compute the posterior. Once we have the posterior we will be interested in computing summaries which typically means taking expectations. Recall the the expected value of a function h under the distribution p is given by the following formula.

$$\mathbb{E}_p[h] \;=\; \int h(x)\, p(x)\, \mathrm{d}x$$

If we can sample from the distribution p then we can use Monte Carlo (MC) methods to approximate this expectation. Assume we have drawn S samples from p denoted $\{x^{(s)}\}_{s=1}^{S}$. Then we can make the following approximation.

$$\mathbb{E}_q[h] \;\approx\; \frac{1}{S}\sum_{s=1}^{S} h(x^{(s)})$$

This approximation is unbiased and the law of large numbers guarantees it will converge to the true value.

The point of this discussion is that if we can sample values from the posterior, we can use these values to compute any expectation we want. In practice sampling directly from the posterior is usually as hard as evaluating its density. To solve this issue we typically have to use MCMC methods. The basic idea of MCMC methods is to construct a Markov chain which has the target (posterior) distribution as its invariant distribution. The samples we obtain from this chain can then be used as our draws from the target distribution.

Because the target distribution of the Markov chain is the invariant distribution we will not typically be sampling from this distribution immediately. If the chain is initialised to a random point it will take several iterations before we are close to sampling from the target. In practice this means we typically discard some number of samples from the start of the chain as *burnin*.

Another issue that MCMC methods face is that the samples will not be independent. Thus drawing S samples from an MCMC method is not equivalent to drawing S samples independently from the target. This in turn means we need more samples from an MCMC method than we would if we could make independent draws from the target distribution to achieve the same level of accuracy. A common way to measure the efficiency of an MCMC algorithm is to look at the autocorrelation between samples. If this decays rapidly then we say the sampler is *mixing* well and we are close to the ideal situation where we can sample directly from the target. If the decay of the autocorrelation is slow then the chain is mixing poorly and we will need many samples to get an accurate approximation. A common procedure to reduce autocorrelation is to only collect samples after multiple steps of the Markov chain. This is referred to as *thinning*. There is a misconception this leads to better estimates of the expectations. This is incorrect, it would be more efficient to use all samples from the chain. The only reason to perform thinning is to reduce the storage cost of samples.

1.4.3. Metropolis-Hastings algorithm

Algorithm 1

Input $\theta^{(0)}$, X and T
for $t \in \{1, ..., T\}$ do
  $\theta' \sim q(\cdot|\theta^{(t-1)})$
  $u \sim \text{Uniform}(\cdot|[0,1])$
  if $u < \alpha(\theta', \theta^{(t-1)})$ do:
    $\theta^{(t)} \leftarrow \theta'$
  else do
    $\theta^{(t)} \leftarrow \theta^{(t-1)}$
return $\{\theta^{(t)}\}_{t=1}^{T}$

The Metropolis-Hastings (MH) algorithm is probably the most famous MCMC algorithm. It is widely applicable and often works well in practice with a little bit of tuning. To implement the MH algorithm we only need to be able to evaluate the joint distribution $p(\theta, X)$, not the posterior distribution.

We also require a proposal distribution $q(\theta'|\theta)$. The proposal distribution acts to produce new values θ' which we will then decide to accept or reject. The proposal distribution can depend on the current parameter value θ. We need to able to sample from q and evaluate its density. A common proposal is the random walk (RW) which proposes a new value $\theta' = \theta + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. That is the RW proposal adds a normally distributed perturbation to the current value. The variance of the normal distribution can be tuned to improved the efficiency of the algorithm.

Remark 5. Here we define the RW proposal for a scalar parameter using a univariate Normal distribution. If the parameter θ is multi-variate we can substitute a multivariate Normal distribution. We then have to tune the covariance matrix instead of the variance.

Given a proposed value θ' we then decide whether to accept or reject the values. If we accept the value we set our current value of θ to θ' and add it to the collected samples, otherwise we keep the old value and add that to our collected samples. The probability of accepting a proposal, $\alpha(\theta', \theta)$ is given by the following formula.

$$\begin{aligned}
\alpha(\theta', \theta) &= \min\left\{1, \frac{p(\theta'|X)\, q(\theta)}{p(\theta|X)\, q(\theta')}\right\} \\
&= \min\left\{1, \frac{\frac{p(\theta', X)}{p(X)}\, q(\theta)}{\frac{p(\theta, X)}{p(X)}\, q(\theta')}\right\} \\
&= \min\left\{1, \frac{p(\theta', X)\, q(\theta)}{p(\theta, X)\, q(\theta')}\right\}
\end{aligned}$$

The trick of the algorithm is that the normalisation constant of the posterior cancels in the acceptance ratio. Thus we never need to explicitly calculate it. Algorithm 1 provides pseudo-code for the MH procedure.

The only parameter that needs to be tuned in the MH algorithm is the proposal distribution. If we use the RW proposal, then we can tune σ^2 to improve the performance of the sampler. If σ^2 is large then we will tend to propose values far from the current one. Typically these will have low probability if the chain is already in a region of high posterior probability. Thus many samples will be rejected. If σ^2 is small we will tend to propose values very near the current one. While these will frequently be accepted, the chain will not move very quickly to explore the space. In practice people typically tune σ^2 to achieve an acceptance rate of between 20-60%. The fact this is not 100% reflects the need to propose values which are far enough away to effectively explore the space. Automated approaches for tuning σ^2 can be used. There are many other choices for proposal distribution beyond the RW. Choosing an appropriate proposal is an important and challenging part of designing an efficient mixing MH sampler.

1.4.4. Gibbs sampling

The Gibbs sampler is another widely used MCMC algorithm. If it is possible to implement it often works better than MH, and requires no tuning. The Gibbs sampler works by iteratively updating a subset of the model parameters at each iteration. After we have updated all the subsets, we have obtained a new sample from the posterior.

Formally assume our parameter vector $\theta = (\theta_1, ..., \theta_B)$ where B is the number of blocks. Each θ_b can be multi-dimensional. The key constraint is that we need to be able to sample from the conditional distribution $p(\theta_b|\{\theta_i\}_{i\neq b}, X)$. That is the distribution of θ_b conditioned on all the other model parameters and the data. In some models this can be much easier than sampling from the full posterior. The two common cases are when the model has *conditionally conjugate* distributions and when the parameter θ_b is discrete. In the first case we can obtain a closed form solution for the distribution. In the second case we just need to evaluate the joint distribution at all possible values and normalise. We then sample from the categorical distribution with probabilities given by the normalised values.

1.4.5. Metropolised-Gibbs algorithm

While the MH algorithm is fairly straightforward to implement, it does not usually work well if the parameter θ is high dimensional. A simple solution is to update the parameters in blocks. Assume we have a high dimensional parameter vector $\theta = (\theta_1, ..., \theta_B)$ where B is the number of blocks. The blocks could be each dimension of the parameter or more generally some set of dimensions that is sensible to update together. The key point is we want to keep the dimensionality of the parameters in each block, θ_b, relatively low. Then we can update each parameter θ_b using an MH update targeting the conditional distribution $p(\theta_b|\{\theta_i\}_{i\neq b}, X)$. This looks like the Gibbs sampler, but we can no longer directly sample from $p(\theta_b|\{\theta_i\}_{i\neq b}, X)$, so we use an MH step. The acceptance probability is then

$$\begin{aligned}
\alpha(\theta_b', \theta_b) &= \min\left\{1, \frac{p(\theta_b'|\{\theta_i\}_{i\neq b}, X)\, q(\theta_b)}{p(\theta_b|\{\theta_i\}_{i\neq b}, X)\, q(\theta_b')}\right\} \\
&= \min\left\{1, \frac{\frac{p(\theta_b', \{\theta_i\}_{i\neq b}, X)}{p(\{\theta_i\}_{i\neq b}, X)}\, q(\theta_b)}{\frac{p(\theta_b, \{\theta_i\}_{i\neq b}, X)}{p(\{\theta_i\}_{i\neq b}, X)}\, q(\theta_b')}\right\} \\
&= \min\left\{1, \frac{p(\theta', X)\, q(\theta)}{p(\theta, X)\, q(\theta')}\right\}
\end{aligned}$$

which is exactly the same as the MH algorithm. Note that our proposal distribution q now depends on the block as we propose values of $\dim(\theta_b)$. Blocking is simple to implement but is often critical to designing an efficient MH sampler.

This Metropolised-Gibbs sampler can be used in conjunction with the standard Gibbs sampler. This is particularly useful when we can only Gibbs sample a subset of the parameter blocks θ_b.