Chapter 17

Markov chain Monte Carlo

A very significant application of Markov chains in statistical modelling is based around Monte Carlo methods. The subject is too extensive to be covered adequately here. Rather, we explain the underlying theory and algorithms, and give a few examples.

The Markov chain Monte Carlo method is a technique to search a high-dimensional space or evaluate 'impossible' sums or integrals over a high-dimensional space. *Monte Carlo* means that we do this by random sampling, and *Markov chain* provides the recipe/algorithm for carrying out the sampling. This has the disadvantage of introducing sampling error to a problem that did not have it. However, one can live with these errors if it can repay the investment with giving approximate answers to, otherwise, intractable problems. In my view, Monte Carlo methods are the last resort of a good scientist, but sometimes one has no choice.

The method originated in physics where the interest was in simulating the dynamics of large complex physical systems with very limited computing power. it continues to be used to this day, but in the meantime the method has found widespread applications in other areas, particularly in statistical modelling. However, it can be widely applied to any problems in complex systems or those with many variables.

A very important application occurs in statistical inference - that is using data to infer the underlying statistical model. However, the mathematical foundations can be easily explained. One proposes a statistical model which may have parameters (or may be non-parametric), and one wishes to determine the best values for the parameter consistent with the data. So one then defines what one means by 'best'. In mathematical terms this often means defining a (object) function that one wishes to maximize or minimize. For example, one could define a likelihood function (or posterior conditional probability) that must be minimized or a residual (error) function that must be minimized, perhaps subject to some constraints. The function will then be dependent on the parameters (as variables) which of course may be dependent variables.

17.1 Optimization and simulated annealing

This defines the well-known *constrained optimization* problem of operational research, for which there are many well-developed techniques. This is not surprising as constrained optimization problems arise in many fields of science and technology. One method employs Markov chains to solve the problem, and is particularly good when many variables are involved - a high-dimensional parameter space. By good I

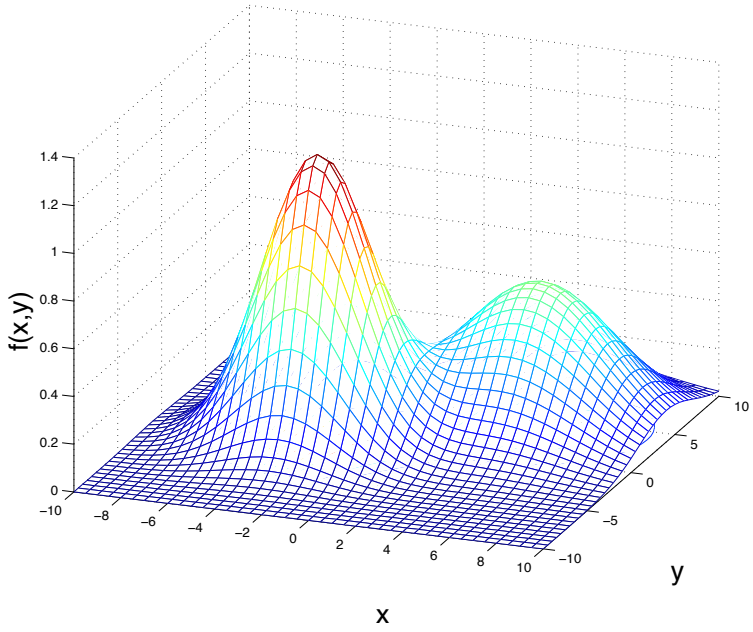

Figure 17.1: Sketch of the function (17.1). The function has two maxima, near $(-3,-1)$ and $(4.5,4)$

mean accurate and computationally efficient. it is better known by the name *simulated annealing* in this context.

Let's state a simple problem, in two variables, and see how the method applies. We wish to find the value of the coordinates (x,y), within a given domain, such that a function, $f(x,y)$ has its maximum value. Specifically let us demonstrate the process for the function presented in figure (17.1) which is simply the superposition of two Gaussian functions:

$$f(x,y) = a_1 e^{-\alpha_1(x-x_1)^2-\alpha_1(y-y_1)^2} + a_2 e^{-\alpha_2(x-x_2)^2-\alpha_2(y-y_2)^2} \quad , \tag{17.1}$$

with $a_1 = 1.2$, $a_2 = 0.6$, $\alpha_1 = 0.1$, and $\alpha_2 = 0.05$. The centres of the peaks are $x_1 = -3, y_1 = -1$ and $x_2 = 4.5, y_2 = 4$.

One can view the problem of optimization (finding the maximum of the function) as a *search problem*. It turns out that searches guided by Markov chain Monte Carlo are quite useful.

The implementation of the method is very simple. For a Markov chain we require the 4 following elements.

(a) Define the states.

(b) Define the equilibrium distribution

(c) Define the transition probabilities

(d) Decide when to stop.

This sounds very easy. Well, it is! However, because it is rather vague it has disadvantages as well. The advantage of the formulation above, is that one has a great deal of flexibility in how to set up the method and it is simple to implement. At the same time, this is a a big disadvantage in that there is no clear prescription how to do this.

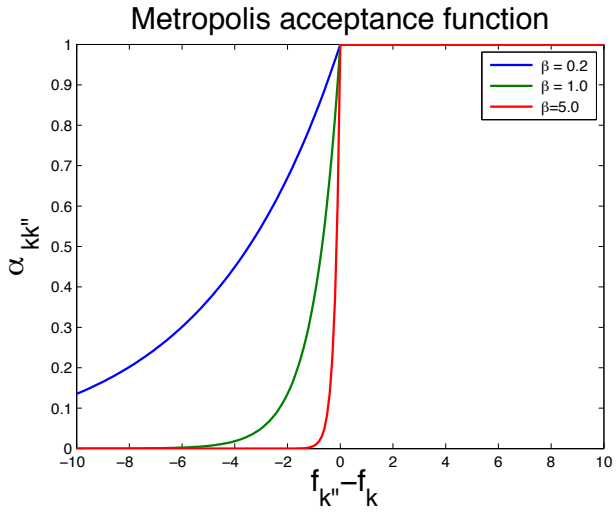

Figure 17.2: The Metropolis acceptance function: $\alpha_{kk'} = \min\{1, e^{\beta(f_{k'}-f_k)}\}$ for various β values, as a function of the function difference: $f_{k'} - f_k$. For $\beta \ll 1$ (blue line), the acceptance is high for all function values. For $\beta \gg 1$ (red line), the acceptance tends towards the *uphill* algorithm: $f_{k'} < f_k$ is highly unlikely, while for $f_{k'} > f_k$, the transition is almost certain.

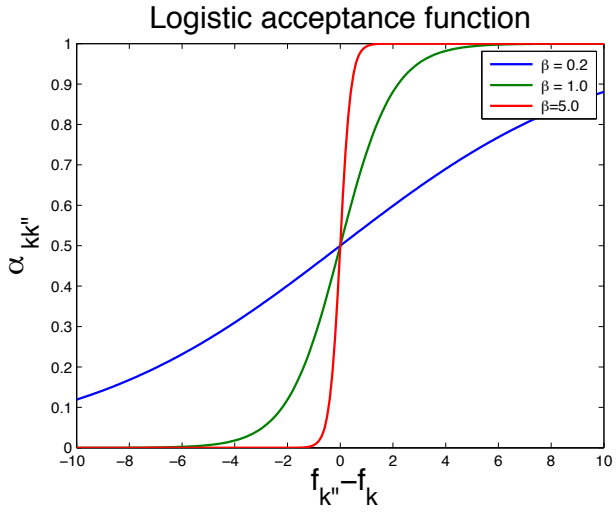

Figure 17.3: The Gibbs (logistic) acceptance function: $\alpha_{kk'} = (1 + e^{-\beta(f_{k'}-f_k)})^{-1}$ for various β values, as a function of the function difference: $f_{k'} - f_k$. For $\beta \ll 1$ (blue line), the probability function is fairly flat, which means that the transition $k \to k'$ has roughly 50% probability whether the function values differ by a large or small amount. So for this value of β the Markov chain wanders fairly aimlessly. For $\beta \gg 1$ (red line), the probability function has a sharp variation. So for $f_{k'} < f_k$ the transition $k \to k'$ is highly unlikely, whereas for $f_{k'} > f_k$, the transition is almost certain. In this case, the walk is strongly favoured towards higher values of f at every step, and strongly biased against any steps that lead to a reduction in f, no matter how small.

(a) Define the states as before, points with cartesian coordinates (x_i, y_j). Since the steps between these points always has the same length h, these form a square grid (lattice) for our walk. Since we will be using a sequence of values we can use the index k to define the state of the system (our walker's $x - y$ coordinates) after k steps. So f_k would be the value of our function at this point in the chain/walk/sequence.

(b) The equilibrium distribution. We want the walker to spend as much time as possible near the maximum of the function. So define the probability distribution as:

$$\pi_k = \frac{e^{\beta f_k}}{Z} \quad , \tag{17.2}$$

where $\beta > 0$ is some constant that we are free to choose. The main point to note is that high values of f_k mean high values of π_k. So the most probable places to find the walker will be at the function maxima.

The use of the symbol β is traditional since it is inherited from Physics ($\beta = (k_B T)^{-1}$, where k_B is the Boltzmann constant and T is the temperature in degrees Kelvin). The constant Z is a normalisation factor, which depends on β and is defined as:

$$Z(\beta) \equiv \sum_i e^{\beta f_k} \quad . \tag{17.3}$$

Again, this symbol is now conventional. Historically, it corresponds to the *partition function* of thermodynamics (the theory of heat) and the capital Z comes from the original German term: *Zustandssumme* ('state sum').

This ensures that the equilibrium distribution satisfies:

$$\sum_k \pi_k = 1 \quad . \tag{17.4}$$

Our aim is *never* to calculate Z, as this would mean calculating *all* the π_k. As we have already stated, our intention is not to calculate the function at all the possible points, just those points where the probability is significantly large. This is the whole point of the method. If we did calculate f at all possible points then we would straightaway know which was the largest value without having to search.

(c) Define the transition probabilities.

We decide to choose these to ensure an irreducible chain, so that (according to the ergodic theorem) the maximum time will be spent at the regions of highest π_k, that is highest f_k. Thus we aim to ensure that our random walker will (in the long run) linger around the highest value of f and hence we can identify the corresponding (x_i, y_j). That is, we would not waste time looking in the wrong place.

Now the question is how to choose the transition matrix. Since we are after a reversible irreducible chain, we can ensure this by enforcing detailed balance.

We decide to form the transition matrix in the following manner:

$$p_{kk'} = q_{kk'} \alpha_{kk'} \quad , \quad k \neq k' \quad . \tag{17.5}$$

and

$$p_{kk} = 1 - \sum_{k' \neq k} p_{kk'} \quad . \tag{17.6}$$

In this expression, $q_{kk'}$ is a transition matrix (called the *proposal function*) which selects the target state k', and α (called the acceptance function) determines whether a transition to the target state is made or not. In our case we perform a random walk in two dimensions. Thus the next step $k \to k'$, will be a small step, of length h, either north/south/east or west. In mathematical notation:

$$q_{kk'} = \begin{cases} \frac{1}{4} & , \quad x_{k'} = x_k \pm h \quad \text{or} \quad y_{k'} = y_k \pm h \\ 0 & , \quad \text{otherwise} \end{cases} \quad . \tag{17.7}$$

Therefore, the transition matrix $q_{kk'} = q_{k'k}$ is symmetric. We can consider a few possible forms for the acceptance function.

(a) Uphill - we *always* accept a step that increases f:

$$\alpha_{kk'} = \begin{cases} 0 & \pi_{k'} \leq \pi_k \\ 1 & \pi_{k'} > \pi_k \end{cases} \quad . \tag{17.8}$$

This can also be written as:

$$\alpha_{kk'} = \frac{1}{(\pi_{k'} - \pi_k)} \max\{0, \pi_{k'} - \pi_k\} \quad . \tag{17.9}$$

In this Markov chain, all states are transient, apart from the (local) maxima which are absorbing. This is *not* a reversible recurrent process, and thus does not satisfy detailed balance, as can be verified. That's not to say that this is a bad search algorithm. It's just one we would not recommend as a search method. However, it is quite useful in polishing off a search method once we know where the maximum lies.

(b) The *Metropolis algorithm* [1] for the acceptance function is:

$$\alpha_{kk'} = \min\{1, \pi_{k'}/\pi_k\} \quad . \tag{17.10}$$

Since:

$$\pi_k p_{kk'} = \pi_k q_{kk'} \min\{1, \pi_{k'}/\pi_k\} = q_{kk'} \min\{\pi_k, \pi_{k'}\} \quad , \tag{17.11}$$

and

$$\pi_{k'} p_{k'k} = \pi_{k'} q_{k'k} \min\{1, \pi_k/\pi_{k'}\} = q_{k'k} \min\{\pi_{k'}, \pi_k\} \quad , \tag{17.12}$$

and since q is symmetric, and $\min\{a, b\} = \min\{b, a\}$, then the expressions (17.11,17.12) are identical and detailed balance holds.

(c) The Gibbs algorithm for the acceptance function is:

$$\alpha_{kk'} = \frac{\pi_{k'}}{\pi_k + \pi_{k'}} = \frac{1}{1 + \pi_{k'}/\pi_k} \quad . \tag{17.13}$$

Since:

$$\pi_k p_{kk'} = \frac{q_{kk'} \pi_k \pi_{k'}}{\pi_k + \pi_{k'}} \quad , \tag{17.14}$$

[1] N. Metropolis and others (1953). Journal of Chemical Physics 21 (6): 10871092.

and

$$\pi_{k'}p_{k'k} = \frac{q_{k'k}\pi_{k'}\pi_k}{\pi_{k'}+\pi_k} \quad , \tag{17.15}$$

the detailed balance relation, $\pi_{k'}p_{k'k} = \pi_k p_{kk'}$, is satisfied.

Thus the *Metropolis algorithm* (17.20) corresponds to choosing the acceptance function:

$$\alpha_{kk'} = \min\{1, e^{-\beta(f_k - f_{k'})}\} \quad , \tag{17.16}$$

That is:

$$\alpha_{kk'} = \begin{cases} 1 & , \quad f_k \leq f_{k'} \\ e^{-\beta(f_k - f_{k'})} & , \quad f_k > f_{k'} \end{cases} \tag{17.17}$$

The function is sketched in figure (17.2).

The Gibbs algorithm is then (17.13):

$$\alpha_{kk'} = \frac{1}{1+e^{-\beta(f_{k'}-f_k)}} \quad . \tag{17.18}$$

and shown in figure (17.3). This is our favourite, but mainly for aesthetic reasons. You may recognise this as the logistic function, and it is drawn in figure (17.3)

Note that, according to (17.13), the probability of $k \to k'$ is favoured when $f_{k'} > f_k$, but not with certainty - and the degree of certainty can be controlled by a judicious choice of β. If we choose $\beta \gg 1$, then this becomes a simple *uphill* search in which one *always* (with certainty) moves to a neighbouring point when the function is higher there. The danger with such a deterministic algorithm is that one could easily converge towards a *false summit*. That is, one could find a local maximum of the function, and never leave that location, overlooking a much higher maximum. Choosing an intermediate value of β allows the random walk to explore more. However, this in turn has the disadvantage that it is slow to converge.

The challenge is to have one's cake and eat it [2]. It is important to explore all interesting regions, what is called *good mixing* in MCMC parlance. Conversely, one does not want to waste time exploring uninteresting regions.

We allow the stochastic process to proceed until, in the long run, it converges towards an equilibrium, in which it spends the majority of its time in a single state. We identify this state as the value of x for which f is maximum. In fact we often take the time (sample) average as the best estimate. We must be wary since we have created an artificial irreducible Markov chain. Thus, in the long run the process will start to wander away (for a finite time) from the most likely location. Therefore, we should not take the last point, but rather, we should take the sample average over an extended time towards the end of our sample. In this way we avoid the transient burn-in period, and average out the brief excursions from the maximum.

We also should consider the need to restart. The choice of the initial (starting) point, is arbitrary. In order to eliminate this (possible) bias from the system, one should repeat the process with random initial points in order to be confident that the answer is not sensitive to this choice. The time taken for the Markov process to reach a repeated equilibrium is called the *burn in* time. Unfortunately, there is no guide as too how long burn in takes, this all depends on the problem in question.

As already mentioned, the choice of β is also pragmatic. A common trick is to adjust β *during* the process,

[2] Italian version: Vuoi la botte piena e la moglie ubriaca!

starting with a small β to explore the states quite widely and then gradually increasing β to home in on the regions of high f. This trick explains the origin of the term *simulated annealing*. In physics, as mentioned before, β corresponds to the inverse of temperature. So increasing β corresponds to cooling the system and allowing the quantity, and $-f$ is the energy of the system. So increasing β corresponds to *cooling* the Markov chain, until it reaches the lowest-energy state (highest f). In fact a good check that your system has converged to the best value of x, is to 'reheat' the system (that is reduce the value β) and restart the random walk. In this way, we allow the system to recommence the random excursion. Then 'recooling' the system (in a mathematical way this means increasing β) will take us towards a maximum of f, which may well be the same region that we found previously. At *zero temperature* $\beta \to +\infty$, the system freezes at a single point, the function maximum.

It should be clear from the above discussion that, this method is not as automatic as one would like. It requires some experimentation and trial and error. That's the price one pays for the flexibility!

For the example problem, finding the maximum of (17.1) these difficulties will become obvious. Let us address the first task (a) the choice of states. We choose to perform a random walk with step size $h = 0.2$, and choose as the starting point $(-2, 2)$. These choices are arbitrary - the choice of step size will dictate the size of the region covered (as well as the number of steps chosen). If the step size is chosen too large, it might miss the important regions where sharp peaks occur. On the other hand if the step size is too small, it will take an extremely long time to cover the region of interest. We choose the number of steps $N_S = 50000$ and decide on $\beta = 12$ as a first guess, for the annealing process. Again, we are being pragmatic and would need to experiment with these values to be convinced that the answer makes sense.

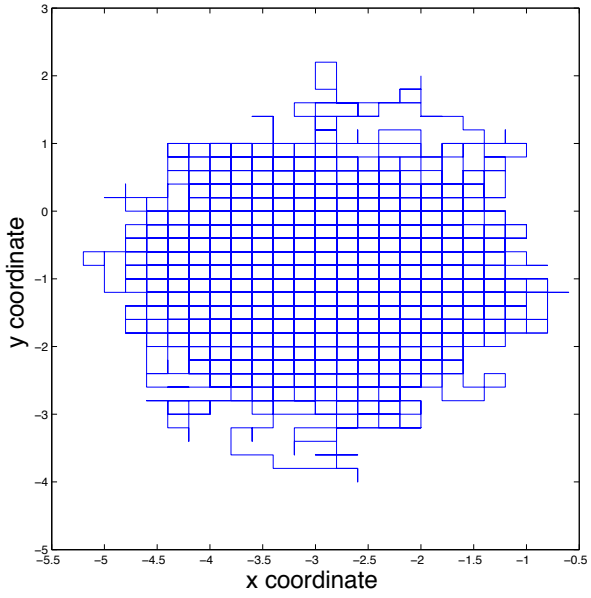

Figure 17.4: A map of the 2D random walk, beginning at $(-2, 2)$, with $\beta = 12$ and step-size $h = 0.2$. The figure shows the result of $N = 50000$ steps. We know, in advance, that the function $f(x, y)$ given by (17.1) has a maximum near $(-3, -1)$. The choice of large β inhibits long excursions away from the maximum.

The time-average value of the position, calculated for one 50,000 step run was found to be:

$$\langle x \rangle = -2.941 \quad , \quad \langle y \rangle = -0.969$$

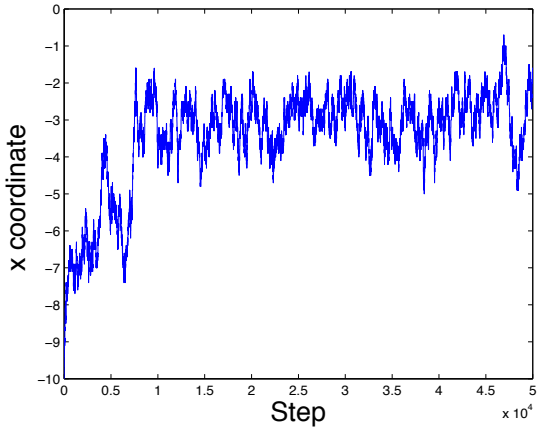

Figure 17.5: Variation in time (step) of the x-coordinate for the MCMC search for the function maximum. The Gibbs (heat bath) algorithm was used, with step-size 0.2. $N = 5000$ steps were used with starting value: $x = -10$. The *burn in* time is the number of steps it takes to reach the equilibrium distribution; in this case around 1×10^4 steps.

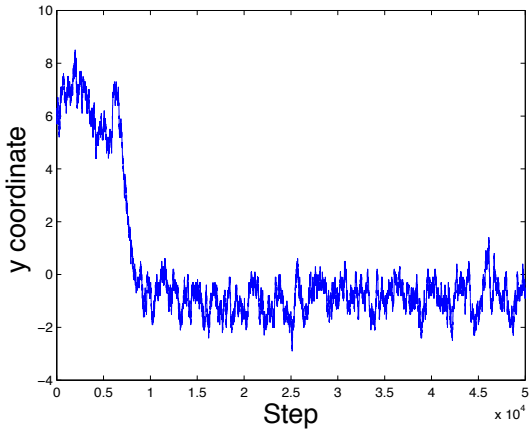

Figure 17.6: Variation in time (step) of the y-coordinate for the MCMC search for the function maximum. The Gibbs (heat bath) algorithm was used, with step-size 0.2. $N = 5000$ steps were used with starting value: $y = 6$. The two-dimensional version of this picture is shown in figure (17.4) above.

and these are acceptable approximations to the true answers $x = -2.967$ and $y = -0.978$, though not very accurate. We can always improve our results by restarting from this point, with a much smaller h and much larger β. In this case, the walk will spend nearly all its time at the highest value of f, where it is frozen in place.

17.1.1 High dimensions

This method seems unnecessarily inefficient and not that accurate, and this is certainly true in 2D where the search space is limited. However, given a function of 4 variables, each with 200 possible points, in the state space that we wish to search, an exhaustive search, in which we calculate the function at every point, requires $200^4 = 1.6 \times 10^9$ function evaluations. Such a calculation would take several days on a standard laptop. It is clear that if the function depended on 10 variables with, say, 500 points for each variable, a search by exhaustively evaluating all the function values is frankly impossible, with even hundreds of computers at one's disposal.

It is just in such circumstances that the MCMC search method becomes an attractive alternative. In this case, one can often get good approximate answers with a few million function evaluations, which only takes a few minutes on a laptop. We have explained the bare essentials of the MCMC method. Naturally, there have been many suggestions for accelerating (improving) the efficiency of the MCMC method, and the accuracy of the method, based on better sampling approaches. However, the classic work horses of the Gibbs and Metropolis sampling methods still retain an appeal because they are so straightforward to implement.

17.2 Expectations

Suppose we have a pair of discrete random variables, $X \in \{x_1, \ldots, x_M\}$ and $Y \in \{y_1, \ldots, y_N\}$, with a joint probability mass:$f_{XY}(x_i, y_j)$. The object is to calculate the expected value of a function $g(x, y)$.

We'll explain later why this might be important.

The expected value of any function $g(x, y)$ is given by:

$$\mathbb{E}\left(g(X,Y)\right) = \sum_{i=1}^{M}\sum_{j=1}^{N} g(x_i, y_j) f_{XY}(x_i, y_j) \qquad . \tag{17.19}$$

The frustrating aspect of doing the calculation directly by evaluating the sum is that, many of the terms in the sum are small and thus these function evaluations are a waste of time. This is much in the same way that in optimisation, one is only interested in a small region of importance. Just as in optimisation, for a function of two variables, such as this, the cost of the function evaluations is not very punitive. However, consider a probability function of m variables each of these variables having $n_1, n_2, \ldots, n_m$ values, respectively. Then this would require $n_1 \times n_2 \times \cdots \times n_{m-1} \times n_m$ function evaluations.

Such a task would be impossible. However, we know that each term is proportional to our (known) probability mass function. Thus knowing where the probability mass is largest gives us an indication where the largest terms in the sum would be found.

17.2.1 Markov chain averaging

Let's illustrate the principle by calculating the double sum (17.19). This will be approximated by a sample sum. As before, our aim is to construct a reversible irreducible Markov chain to perform this sampling.

The *states* of the Markov chain will be the values of the random variables. So in general, the point (x_i, y_j) is one of the $M \times N$ states. Next we define the transition probabilities.

We deploy a version of the Metropolis algorithm. This should be familiar from the discussion on optimisation, but it will be repeated here for completeness. We suppose we are currently in a state (i, j), which we will denote by a single label k. Then we choose, at random, a single possible target state from the $(M \times N - 1)$ possible states. Let's call this state (i', j'), and label is as k'. For example, we could select k' by generating a pair of random numbers (u_1, u_2) from uniform distributions. Then we choose:

$$i' = \lfloor M u_1 + 1 \rfloor \qquad , \qquad j' = \lfloor N u_2 + 1 \rfloor \qquad .$$

where $\lfloor z \rfloor$ means the largest integer less than z.

The *Metropolis algorithm* can be expressed as follows:

$$p_{kk'} = \min\left\{1, \frac{\pi_{k'}}{\pi_k}\right\} \tag{17.20}$$

where:

$$\pi_k = f_{XY}(x_i, y_j) \qquad , \qquad \pi_{k'} = f_{XY}(x_{i'}, y_{j'}) \qquad . \tag{17.21}$$

This develops the rules for the Markov chain. Then a simulation creating a sample average would be given by:

$$\mathbb{E}\left(g(X,Y)\right) \approx \frac{1}{N_S} \sum_{k=N_B+1}^{N_B+N_S} g(x_k, y_k) \qquad . \tag{17.22}$$

In this sample average, sampling after a burn in time, chosen beforehand, as N_B. Then N_S sample steps are taken. Explicitly, if we have a sample of N_S steps then this will involve a number of visits $N_{i,j}$ to

each state (x_i, y_j), with $\sum_{i,j} N_{i,j} = N_S$, and we can count these repeat visits as:

$$S_{N_S} \equiv \frac{1}{N_S} \sum_{k=N_B+1}^{N_B+N_S} g(x_k, y_k) = \sum_{i,j} g(x_i, y_i) \times \frac{N_{i,j}}{N_S} \quad , \tag{17.23}$$

The larger the sample, longer the Markov chain, the better the convergence to the true expectation, by the law of large numbers (ergodic theorem). Then the fraction of visits $(N_{i,j}/N_S)$ will tend to the equilibrium distribution.

$$\boxed{\lim_{N_S \to \infty} \frac{N_{i,j}}{N_S} = \pi_{i,j} = f_{XY}(x_i, y_j)} \quad . \tag{17.24}$$

Then we have that,

$$\boxed{\mathbb{E}\left(g(X,Y)\right) = \sum_{i,j} g(x_i, y_j) f_{XY}(x_i, y_j) = \lim_{N_S \to \infty} \frac{1}{N_S} \sum_{k=N_B+1}^{N_B+N_S} g(x_k, y_k)} \quad . \tag{17.25}$$

This expression summarizes the slightly odd approach of the MCMC method. We have converted a spatial average (the sums over i and j) which we can do exactly (without approximation) into a time (sample) average (the sum over k). The advantage of this is that the sum involved in the time averaging is a great deal shorter that the sums of i and j. The disadvantage is that, because we are using a sample, we only have an approximate answer. The law of large numbers encourages us to take a bigger and bigger sample. The convergence of the approximation to the correct result is given by the central-limit theorem for Markov chains, which is just beyond the scope of this course.

17.2.2 Marginal distributions

As previously discussed, In general, one is not interested in the probability mass or density per se, but rather expectations based on the mass/density. If one is interested in the probability mass then one can simply use the ergodic theorem (17.24). However, the marginal mass is often of interest when one is calculating expectations of a single variable. These are defined, by the partial sums, in the case of two variables:

$$f_X(x_i) = \sum_j f_{XY}(x_i, y_j) \quad , \quad f_Y(y_j) = \sum_i f_{XY}(x_i, y_j) \quad . \tag{17.26}$$

In higher dimensions, we sum over all non-marginal variables. The equivalent MCMC estimates follow from the ergodic theorem (17.24) and approximated by:

$$f_X(x_i) = \lim_{N_S \to \infty} \frac{1}{N_S} \sum_j N_{i,j} \quad , \quad f_Y(y_j) = \lim_{N_S \to \infty} \frac{1}{N_S} \sum_i N_{i,j} \quad . \tag{17.27}$$

17.2.3 Markov chain integration

Suppose the variables are continuous. Then the expected value of any function $g(x, y)$ is given by:

$$\mathbb{E}\left(g(X,Y)\right) = \int_a^b \int_c^d g(x,y) f_{XY}(x,y) dy \; dx \quad . \tag{17.28}$$

In practice, these integrals are done numerically, that is using some kind of quadrature rule. That is, one discretises the variable, and thus, this is equivalent to

$$\mathbb{E}\left(g(X,Y)\right) = \sum_{i=1}^{M}\sum_{j=1}^{N} w_{ij}\, g(x_i, y_j) f_{XY}(x_i, y_j) \qquad . \tag{17.29}$$

where w_{ij} are the (two-dimensional) quadrature weights, and x_i and y_j the pivots for the appropriate quadrature rules. Thus we can apply exactly the same arguments as above, though in this case the quadrature weights are combined with $g(x_i, y_j$, and we arrive at the same result, just replacing $g(x_k, y_k) \to w_k g(x_k, y_k)$:

$$\mathbb{E}\left(g(X,Y)\right) = \int_a^b \int_c^d g(x,y) f_{XY}(x,y) dy\; dx \approx \lim_{N_S\to\infty} \frac{1}{N_S} \sum_{k=N_B+1}^{N_B+N_S} w_k\; g(x_k, y_k) \qquad . \tag{17.30}$$

17.3 Statistical Modelling

Statistical inference describes how we make inferences (deductions) from a set of observations (data) from a representative sample.

The statistical viewpoint considers the data as a random sample, so the best we can do is obtain the best probability distribution that represents the data. This distribution will be some kind of measure that depends on parameters as well as the data variables. Often we explore the data (looking at descriptive statistics and marginal variables) to see if it has a familiar shape. For example, we might see a linear relation (albeit with scatter) and decide that the best model is *linear regression*, then the job is to find the best fit, often in terms of minimising least squares.

For the perspective of *information theory*, we have data that represents incomplete knowledge of the system, given limited information (data). We seek the *best* guess of what the information represents. That is the model that encapsulates the data most accurately and efficiently. The choice of *measure* to represent this data is, in a sense, random. We are adding parameter variables to the existing observed variables in the hope of making the problem simpler to understand. The parameters can be used to model/compress the sample data into a form suitable for generalisation to a larger unobserved data.

In this sense, the choice of model, and the choice of data, is also random. The values of the parameters will also be uncertain, although one would hope that the uncertainty would be quantifiable. In the *frequentist interpretation*, the best values for the parameters can be obtained by maximum-likelihood estimation. That is we propose a distribution function and find the values of the parameters, *given* the data we have, that are most probable. The method of Bayesian inference is not as prescriptive. That is, we consider both the parameter variables and the data variables to be random. Setting aside the philosophical arguments of these two approaches, the practical problems are rather similar and both exploit MCMC and both are applications of optimisation.

Let us simplify the discussion, without loss of generality, to considering two parameter variables (x, y) and S (scalar) observations: $\{d_1, d_2, \ldots, d_S\}$. In principal these 'parameters' could be discrete or continuous, categorical or quantitative. To simplify further we suppose they have discrete values, $\{x_1, x_2, \ldots, x_M\}$ and $\{y_1, y_2, \ldots, y_N\}$, and similarly for the data variable. There is no loss of generality since even continuous distributions are (in practice) discretised, and it is equally possible to create a continuous distribution by a dense discrete mass. In either case, one is simply looking to extract probability (a measure) from sets whether they are discrete or continuous. So we will say probability mass when we really mean measure.

17.3.1 Maximum Likelihood

In the maximum-likelihood approach, one proposes a probability mass which depends on the parameters (x, y) and the data variable, d:

$$P(x, y; d) \quad . \tag{17.31}$$

From this one derives a likelihood function based on the observation data $\{d_i\}$:

$$\mathcal{L}(x, y) \equiv \prod_{i=1}^{S} P(x, y; d_i) \tag{17.32}$$

More precisely, since the observation data form part of this function, but are given (fixed) variables, one can write:

$$\mathcal{L}(x, y; \{d_i\}) \equiv \prod_{i=1}^{S} P(x, y; d_i) \tag{17.33}$$

In some treatments, one uses | rather than ; to separate the variable parameters x, y from the fixed variables $\{d_i\}$, and expresses this as a conditional likelihood. The important thing to note is the separation of variables.

On can then use MCMC to explore the (x, y) space to find where this is optimal (maximum):

$$(x^*, y^*) = \max_{(x,y)} \mathcal{L}(x, y) \qquad . \tag{17.34}$$

Thus, one arrives at the optimal value of the parameters (x^*, y^*) for the probability function (17.31). One can then use an information criterion, such as BIC or AIC to estimate the goodness of fit.

17.3.2 Bayesian inference

The information theory approach, exemplified by the Bayesian method, treats the parameters as random variables. One does not speak of optimal values of the parameters, but rather a distribution of values. The logical way to define such a distribution is by conditioning on the observations (the known data/information). So one can speak of the most probable values of x and y, and the mean value of x and y as being the best estimators for these parameters, but not definite values. This uncertainty is appealing in that it reflects the arbitrariness of the choice of the model.

The postulate is that one can lump data and parameters together such that there exists a joint probability mass:

$$P(x, y, \{d_i\}) \quad , \tag{17.35}$$

with corresponding multiplication rules:

$$P(x, y, \{d_i\}) = P(x, y|\{d_i\})P(\{d_i\}) = P(\{d_i\}|x, y)P(x, y) \qquad . \tag{17.36}$$

Then by Bayes' theorem, we can define the posterior distribution:

$$P(x, y|\{d_i\}) = \frac{P(\{d_i\}|x, y) \times P(x, y)}{P(\{d_i\})} = \frac{P(\{d_i\}|x, y) \times P(x, y)}{\sum_{x,y} P(\{d_i\}|x, y)P(x, y)} \tag{17.37}$$

All Bayesian inference is based on this type of distribution. As before, the practical limitations in calculating $P(x, y|\{d_i\})$ are related to the high-dimensional parameter space, and the associated normalisation

factor, the denominator of (17.37).

$$\sum_{x,y} P(\{d_i\}|x,y)P(x,y) \tag{17.38}$$

One is primarily interested in expectations of distributions (and their functions) rather than the distributions per se. For example, the mean of x as the best estimator, that is:

$$\mathbb{E}\,(x|\{d_i\}) \qquad . \tag{17.39}$$

In general, one would be interested in expectations of the general form:

$$\mathbb{E}\,(g(x,y)|\{d_i\}) \qquad . \tag{17.40}$$

Notwithstanding the fact that, with two variables, it is entirely feasible to calculate the 'unattainable' posterior distribution (17.37), iwe will show how to do this for two parameters using the Markov chain Monte Carlo approach. The procedure for 20 or 30 variables is only slightly more complicated, but the idea is exactly the same.

17.3.3 MCMC for inference

To create the (reversible) Markov chain requires:
• the probability distribution (not necessarily normalised)
• a transition matrix that satisfies detailed balance.

Recall that we are taking a random walk in the parameter space which has discrete values: $x \in \{x_1,\ldots,x_M\}$ and $y \in \{y_1,\ldots,y_N\}$. The state in the sequence, as usual, is indexed by k, the step-number (or time value) of the process, with an associated point in parameter space $(x_i, y_j$ and the next step $k' = k+1$ will correspond to another point $(x_{i'}, y_{j'})$.

Since normalisation is not an issue, we can assume a prior distribution $P(x,y)$ and then calculate the likelihood function $P(\{d_i\}|x,y)$ for any points $(x_i, y_j$ (state at k) and $(x_{i'}, y_{j'})$ (state at k'). This is straightforward.

$$\pi_k = \frac{P(\{d_i\}|x_i,y_j)P(x_i,y_j)}{Z} \quad , \quad \pi_{k'} = \frac{P(\{d_i\}|x_{i'},y_{j'})P(x_{i'},y_{j'})}{Z} \qquad . \tag{17.41}$$

We then need a transition probability that complies with detailed balance and requires little calculation. Just thinking step by step, a simple solution is to change either the x or the y value, but not both. That is, if we fix $y_{j'} = y_j$ then, essentially, then the π_k is a conditional probability on y (as well as the data). Now the big advantage of doing things one at a time is that we don't have to juggle so many balls in the air. So now, for this step, and this step only, we treat y_j as a fixed (given) variable:

$$\pi_k = \pi(x_i|y_j) \equiv \frac{P(\{d_i\}|x_i,y_j)P(x_i,y_j)}{Z} \quad , \quad \pi_{k'} = \pi(x_{i'}|y_j) \equiv \frac{P(\{d_i\}|x_{i'},y_{j'})P(x_{i'},y_j)}{Z} \qquad . \tag{17.42}$$

Here Z stands for the awkward normalisation constant. Then one can arrange detailed balance by choosing:

$$p_{kk'} = \pi(x_{i'}|y_j) \quad \text{and} \quad p_{k'k} = \pi(x_i|y_j) \quad . \tag{17.43}$$

Then clearly:

$$\pi_k p_{kk'} = \pi(x_i|y_j)\pi(x_{i'}|y_j) = p_{k'k}\pi_{k'} \tag{17.44}$$

and detailed balance is satisfied. The only remaining difficulty is that this transition probability is not correctly normalised. Remember, that we disregarded the normalisation of π_k. Since we must have a stochastic matrix this must be normalised. But this is the easy part since we are only changing one variable, then this (so-called) full conditional probability, $\pi(x_i|y_j)$, requires only a one-dimensional sum to get the normalisation correct. Thus, summing over all the possible x-values only, gives us:

$$p_{kk'} = \frac{\pi_{k'}}{\sum_{k''} \pi_{k''}} = \frac{P(\{d_i\}|x_{i'}, y_j)P(x_{i'}, y_j)}{\sum_{i''=1}^{M} P(\{d_i\}|x_{i''}, y_j)P(x_{i''}, y_j)} \quad . \tag{17.45}$$

Note that this jump allows us to go to any of the available $x_{j'}$ values. It is not the simple random walk between neighbouring points with which we started this section.

The next step requires another decision whether to go with another x step or y, and this should be done at random. This seems awfully longwinded, but the computer whizzes along in no time, so that the calculations are done in a simple and efficient manner.

The final question about this one-at-a-time sampling is to whether it will accurately sample the (x, y) fairly. The answer to this is not simple, but relies on the fact that the distribution: $P(x, y|\{d_i\})$ is uniquely determined by the full conditional distribution of its parameters x and y. Thus sampling from these is ensures that we are sampling $P(x, y|\{d_i\})$ and not another distribution.

Thus we have a recipe for exploring the (x, y) space in a systematic manner. Of course, this problem has the additional complication that we need to choose a prior distribution to get the whole process started, and indeed, in high-dimensional spaces, some thought needs to be given to the starting point in the parameter space. However, that's a story for another time.