CHAPTER

A Tutorial Introduction to Monte Carlo Methods, Markov Chain Monte Carlo and Particle Filtering

19

A. Taylan Cemgil

Department of Computer Engineering, Boğaziçi University 34342 Bebek, Istanbul, Turkey

1.19.1 Introduction

Monte Carlo (method) (MC) is an umbrella name for an arsenal of numerical techniques for computing approximate estimates via random sampling. The estimates are very often given as the result of an intractable integral and the technique could have been named "numerical integration in high dimensions via random sampling." The term Monte Carlo was initially coined during 1940's by von Neumann, Ulam and Metropolis [1,2] as a "cuteness" [3] to refer to the famous casino of 1940s. However, due to the central importance of the subject and wide use of the concept, it soon became an established technical term.

Monte Carlo techniques have been further popularized in applied sciences with the wider availability of computing power, starting from the 90s, most notably in statistics, computer science, operational research and signal processing. In signal processing or operational research, MC experiments are extensively used for assessing the performance of a system or a computational method by generating a random sample of typical inputs. While the term Monte Carlo experiment is also used extensively here, the goal in such applications is "system simulation," i.e., to understand the properties of a system under uncertainty. The Monte Carlo methods in this paper are slightly different from system simulation; we have a well defined "deterministic" computational problem with a single answer and randomization is used as a tool for approximate computation. As such, we will refer to MC in this context and the applications will be parameter estimation, prediction and model selection.

In this tutorial, we will sketch the aims, the basic techniques and the principles of Monte Carlo computation. After the illustration of the law of large numbers and central limit theorem [4], we cover basic Monte Carlo methods for sampling from elementary distributions: inversion, transform and rejection techniques [5]. We cover also Markov Chain Monte Carlo (MCMC) methods [6]; but rather than giving only the basic algorithms, we sketch the implications of some of the key results in the theory of finite Markov chains [7]. We also cover importance sampling and sequential Monte Carlo methods [8–10]. Finally we give an overview of a more advanced technique, the reversible jump method [11]. Our goal is to provide a self contained introduction with a unified notation, build up a basic appreciation of modern Monte Carlo techniques and to sharpen the readers intuition using several toy examples. Parts of this material have been used in advanced undergraduate and introductory graduate courses on Monte Carlo computation, taken primarily by students interested in techniques for data analysis with signal processing, machine learning or data mining background with some working knowledge of probability theory and statistics. Experience has shown that even tutorial texts on statistical computation were not immediately accessible due to the notation barrier and students benefited mostly from toy examples. Inevitably, in an elementary tutorial we had to omit a lot of techniques but still tried to retain the "gist" of the subject.

1.19.2 The Monte Carlo principle

In an abstract setting, the main principle of a Monte Carlo technique is to generate a set of samples $x^{(1)}, \ldots, x^{(N)}$ from a target distribution $\pi(x)$ to estimate some features of this target density π. Features are simply expectations of "well behaving" functions that can be approximated as averages:

$$\mathsf{E}_\pi(\varphi(x)) \approx \frac{\varphi(x^{(1)}) + \cdots + \varphi(x^{(N)})}{N} \equiv \bar{E}_{\varphi,N}.$$

Provided that N is large enough, we hope that our estimate $\bar{E}_{\varphi,N}$ converges to the true value of the expectation $\mathsf{E}_\pi(\varphi(x))$. More concrete examples of test functions $\varphi(x)$ will be provided in the next section. For independent and identically distributed samples, this is guaranteed by two key mathematical results: the strong Law of Large Numbers (LLN) and the Central Limit Theorem (CLT) [4,12]. Assuming that $\mathsf{E}_\pi(\varphi(x)) = \mu$ and $\mathsf{V}_\pi(\varphi(x)) = \sigma^2$ (we have finite mean and variance), the LLN states that

$$\bar{E}_{\varphi,N} \to \mu \quad \text{a.s.}$$

Here, a.s. denotes convergence *almost surely*, meaning that $\Pr\{\lim_{N\to\infty} |\mu - E_{\varphi,N}| = 0\} = 1$. Whilst fluctuations are inevitable (since our approximation will be based on a random and finite sample set $x^{(1)} \ldots x^{(N)}$) we wish these fluctuations to be small. This is guaranteed by the CLT: for sufficiently large N, the fluctuations are approximately Gaussian distributed

$$\bar{E}_{\varphi,N} \sim \mathcal{N}\left(\bar{E}_{\varphi,N} | \mu \sigma^2/N\right)$$

and the variance of the estimate drops with increasing N, while its mean is the desired value. This result has important practical consequences. If we can generate i.i.d. samples from the distribution of x, denoted as $\pi(x)$, we can estimate expectations of $\varphi(x)$:

1. Monte Carlo provides a "noisy" but unbiased estimate of the true value $\mu = \mathsf{E}_\pi \varphi(x)$.
2. The error behaves as $O(N^{-1/2})$.
3. The convergence rate is independent of the dimensionality of x.

The third point is particularly important. It suggests that, at least in principle, with a quite small number of samples one can compute approximate solutions for arbitrary large parameter estimation problems. All one needs is obtaining independent samples. The difficulty with this approach, however, is in generating independent samples from a target distribution $\pi(x)$ and various Monte Carlo methods aim to provide techniques for this.

1.19.2.1 Illustration of the MC principle

In this section, we will illustrate the MC principle, that averages obtained from several random experiments tends to stabilize. As the running example we will focus on a problem mentioned in the famous letters between Pascal and Fermat, first writings that are considered to mark the start of the modern probability theory [13]. A French nobleman and gambler Chevalier de Méré contacted Pascal. Méré was betting that in four rolls of a single die, at least one six would turn up. Later, Méré "extended" his schema where he was betting that in 24 rolls of two dice, a pair of sixes would turn up. He was not happy with this new schema and was calling Pascal for an explanation.

Indeed, the analytical solution for this problem is elementary; we merely calculate the probability that in 4 consecutive independent throws at least a 6 comes up. A quick calculation shows indeed there is a slight advantage for Méré:

$$\begin{aligned}\Pr\{\text{Méré wins in 4 throws}\} &= 1 - \Pr\{\text{no 6 is thrown}\}^4\\ &= 1 - (5/6)^4 = 0.5177.\end{aligned}$$

In a sense, this experiment is equivalent to playing head and tail with a carefully forged coin. The bias is small and may not be easily detected by a naive opponent. Like all Casinos, Méré exploited this subtle bias to his advantage to earn money on average in repeated trials.

However, the analytical solution for the two dice game reveals that throwing the dice only 24 times is not a good bet for Méré:

$$\Pr\{\text{ Méré wins in 24 throws}\} = 1 - (35/36)^{24} = 0.4914,$$

but with 25 throws, the odds turn into his favor as

$$\Pr\{\text{Méré wins in 25 throws}\} = 1 - (35/36)^{25} = 0.5055.$$

The natural question here is if one could detect with certainty that there is indeed a systematic deviation from 0.5 by just looking at the outcomes of a sequence of games. Indeed for many problems of interest, such probability calculations will be significantly harder, if at all possible, and it is natural to ask how reliable it is to estimate such quantities using data obtained from MC experiments.

Playing the game repetitively is a MC experiment. By just recording the outcome of each game (and the total number of games played), we could in principle estimate the winning probability. To be more precise, consider the single die, 4 times throw game. Let $x_k^{(n)}$ denote the outcome of the die at kth throw in the nth game. Formally, a game is represented by the tuple $x \equiv (x_1, x_2, x_3, x_4)$. We assume a fair die, as such for $k = 1, \ldots, 4$

$$x_k \sim \mathcal{U}(1, 6),$$

where $\mathcal{U}(a, b)$ denotes a uniform distribution on integers x such that $a \leq x \leq b$. Since each throw in the game is independent, the target distribution $\pi(x)$ is the uniform distribution on $\mathcal{X} = \{1, \ldots, 6\}^4$ and each game is simply a random point in $\mathcal{X}$.

To estimate the probability of winning, we define formally the indicator function for winning the nth game

$$\varphi(x^{(n)}) = \mathbb{I}\left\{0 < \sum_{k=1}^{4} \mathbb{I}\left\{x_k^{(n)} = 6\right\}\right\}.$$

Here, $\mathbb{I}\{x\}$ is an indicator function that is 1 if x is true and 0 if false. The test function φ looks awkward but it just checks if in 4 throws at least a 6 is obtained. Finally, the probability θ that Méré wins in 4 throws can be estimated as

$$\theta = \mathsf{E}_{\pi}(\varphi(x)) \approx \frac{1}{N}\sum_{n=1}^{N}\varphi(x^{(n)}) = \bar{E}_{\varphi,N}.$$

Similarly, for the two dice, 24 throws game, formally we let $x_{i,k}^{(n)}$ denote the outcome of the die i at kth throw in the nth game.

$$\varphi(x^{(n)}) = \mathbb{I}\left\{0 < \sum_{k=1}^{24}\mathbb{I}\left\{x_{1,k}^{(n)} = 6\right\}\mathbb{I}\left\{x_{2,k}^{(n)} = 6\right\}\right\},$$
$$\bar{E}_{\varphi,N} = \frac{1}{N}\sum_{n=1}^{N}\varphi(x^{(n)}). \tag{19.1}$$

The mean and variance of $\phi(x^{(n)})$ are

$$\mathsf{E}\left(\varphi(x^{(n)})\right) = \theta$$
$$\mathrm{Var}\{\varphi(x^{(n)})\} = \theta(1-\theta)$$

respectively.[1] As these are finite, the strong law of large numbers applies. The law guarantees the very intuitive fact that with increasing number of games N, the estimates become increasingly more and more accurate. Technically, when $N \to \infty$ we have convergence

$$\bar{E}_{\varphi,N} \to \theta.$$

The "speed" is given by the central limit theorem, that says that for large N the estimate is distributed

$$\bar{E}_{\varphi,N} \sim \mathcal{N}(\,\bar{E}_{\varphi,N}\,|\,\theta, \theta(1-\theta)/N).$$

Provided that we can generate independent samples, the error will be $O(N^{-1/2})$. The most important aspect of this convergence result, that can not be overemphasized is that the error does not depend on the dimensionality of the parameter to be estimated!

To illustrate these concepts with an example consider Figure 19.1 where we simulate the estimate for a two dice, 24 throw game case for several N. The path shows the estimate $\bar{E}_{\varphi,N}$ in Eq. (19.1) for each N. We can see that the path converges to the true value. For each N, the error bars correspond to the standard deviations given by the central limit theorem.

[1] The mean and variance of a Bernoulli random variable.

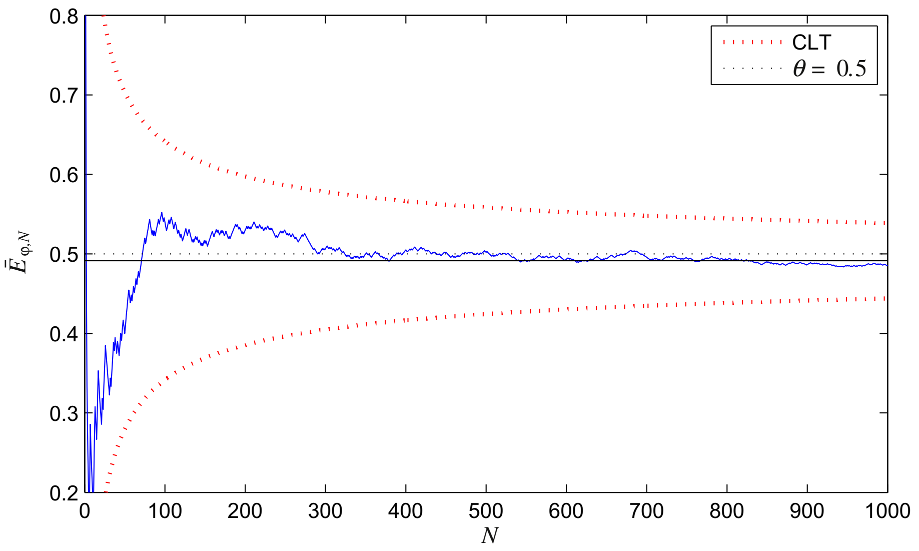

FIGURE 19.1

The estimate of winning probability for a pair of 6 in 24 throws game. The horizontal solid line shows the true odds θ_{true} that is smaller than 0.5. The path shows the estimate $\bar{E}_{\varphi,N}$ as a function of N. Law of large numbers says that all these curves eventually will converge to the true θ_{true}, shown as a solid line.

To illustrate the implication of the central limit theorem, consider now a collection of estimates, obtained from results of N games per night with a total of K different nights. Formally, the outcome of the nth game at the kth night is denoted as $\varphi(x^{(n,i)})$ for $i = 1, \ldots, K$ and $n = 1, \ldots, N$. The estimate at the ith night after observing the Nth game as

$$\bar{E}^{(i)}_{\varphi,N} = \frac{1}{N}\sum_{n=1}^{N}\varphi(x^{(n,i)}).$$

Now, we visualise all the paths together, where each path is depicted corresponds to the sequence of estimates for each independent night i (Figure 19.2). Each path corresponds to the estimate as a function of the number of games played N. We can clearly see that all the independent paths eventually stay in the "narrow tube," as predicted by the CLT.

In the above examples, instead of experimenting with real dice, we have simulated the experiment on a computer. It turns out that it is surprisingly delicate to simulate "truly" random numbers on deterministic hardware. Instead, *pseudo-random numbers* are generated using deterministic algorithms and are briefly reviewed in the next section.

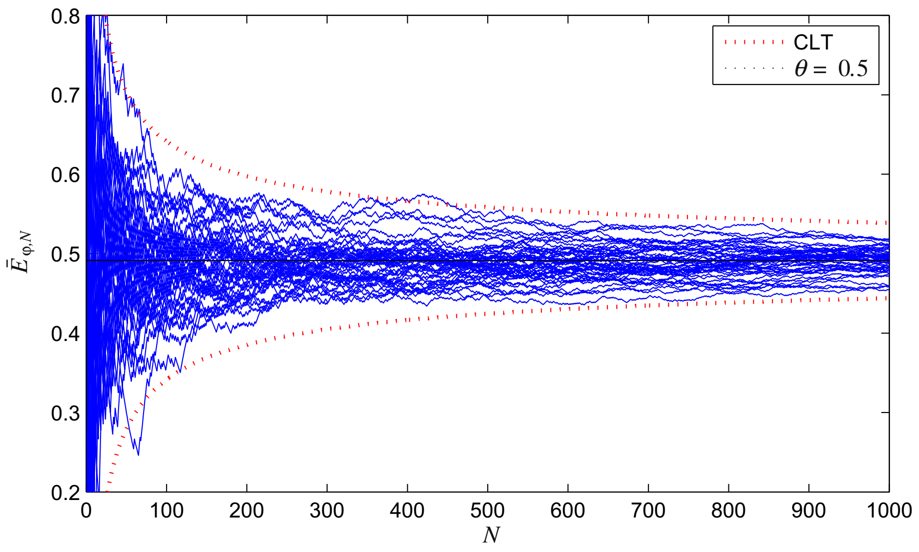

FIGURE 19.2

Verification of the central limit theorem. We estimate the winning probability for the 2 dice, 24 throws game. Each path corresponds to the estimate $\bar{E}^{(i)}_{\varphi,N}$ as a function of N, where $i = 1, \dots, K$ denotes the independent sequences of games. CLT says that at any fixed N, $\bar{E}^{(i)}_{\varphi,N}$ the values of these estimates is approximately normal distributed with standard deviation $\sigma/\sqrt{N}$. For each N, the red dotted curves designate the $[\theta_{\text{true}} - 3\sigma/\sqrt{N}, \theta_{\text{true}} + 3\sigma/\sqrt{N}]$. It is a common folklore in statistics to plot the $\pm 3\sigma$ interval for a univariate Gaussian as this interval contains more than 99% of the probability mass. As given by the CLT, the paths exit the $\pm 3\sigma/\sqrt{N}$ tube only occasionally.

1.19.2.2 Random number generation

"The generation of random numbers is too important to be left to chance"[2] and truly random numbers are impossible to generate on a deterministic computer. Published tables or other mechanical methods such as throwing dice, flipping coins, shuffling cards or turning the roulette wheels are clearly not very practical for generating the random numbers that are needed for computer simulations. Other techniques, more suited to computation exist; these rely on chaotic behavior, such as the thermal noise in Zener diodes or other analog circuits as well as the atmospheric noise (see, e.g., www.Random.org) or running a hash function against a frame of a video stream. Still, the vast amount of random numbers are obtained from *pseudo-random number* generators. Apart from being very efficient, one additional advantage of these techniques is that the sequences are reproducible by setting a seed, this property is key for debugging MC code.

The most well known method for generating random numbers is based on a Linear Congruential Generator (LCG). The theory is well understood, the method is easy to implement and it runs very fast.

[2]Wikipedia entry on Pseudo Random Number generator.

A LCG is defined by the recurrence relation:

$$x_{n+1} \equiv (ax_n + c) \pmod{M}.$$

If the coefficients a and c are chosen carefully (e.g., relatively prime to M), x will be roughly uniformly distributed between 0 and $M - 1$. Roughly uniformly means that, the sequence of numbers $x_n$will pass many reasonable tests for randomness. A such test suite is the so called DIEHARD tests, developed by George Marsaglia [14] that are a battery of statistical tests for measuring the quality of a random number generator.

A more recently proposed generator is the Mersenne Twister algorithm, by Matsumoto and Nishimura [15]. It has several desirable features such as a long period and being very fast. Many public domain implementations exist and it is the preferred random number generator for statistical simulations and Monte Carlo computations.

Good random number generators provide uniformly distributed numbers on an interval. Hence, provided we have a good random number generator that can generate uniformly random integers (say between 0 and $2^{64} - 1$), the resulting integers can be used to generate double precision numbers almost uniformly distributed in [0, 1). We will assume in the sequel that we have access to a good uniform random number generator that can generate real numbers on [0, 1); knowing that in practice such a generator generates only rational numbers (say double precision floating point numbers). We will denote this generator by $\mathcal{U}(0, 1)$ and denote a realization, i.e., a sample drawn as $u \sim \mathcal{U}(0, 1)$.

1.19.3 Basic techniques for simulating random variables

The techniques described in this section sketch the basic techniques to generate (mostly univariate and bivariate random variables) with a given density. This is a surprisingly deep and interesting subject touching many facets of computer science. For further information, the reader is invited to look at the fantastic book by Luc Devroye, on non uniform random variate generation [16].

1.19.3.1 Inversion

The basic technique is based on transformation; we generate a uniform random number $u \sim \mathcal{U}(0, 1)$ and transform it to obtain $x = g(u)$ where g is a function. This method requires calculating the cumulative density function and inverting it. Recall the definition of a Cummulative Density function (CDF)

$$F_X(x) = \int_{-\infty}^{x} f_X(\tau) d\top,$$

where f_X is the probability density of the random variable X. The key observation behind the inversion method is the fact that when $X \sim f_X$, the quantity $U = F_X(X)$, when viewed as a random variable, is uniformly distributed on [0, 1). To see this, assume that the inverse of F_X exists, and we can find $X = F_X^{-1}(U)$,

$$\begin{aligned} F_X(x) &= \Pr\{X < x\}, \\ F_X(F_X^{-1}(u)) &= \Pr\{F_X^{-1}(U) < F_X^{-1}(u)\}, \\ u &= \Pr\{U < u\} \quad 0 \le u \le 1. \end{aligned}$$

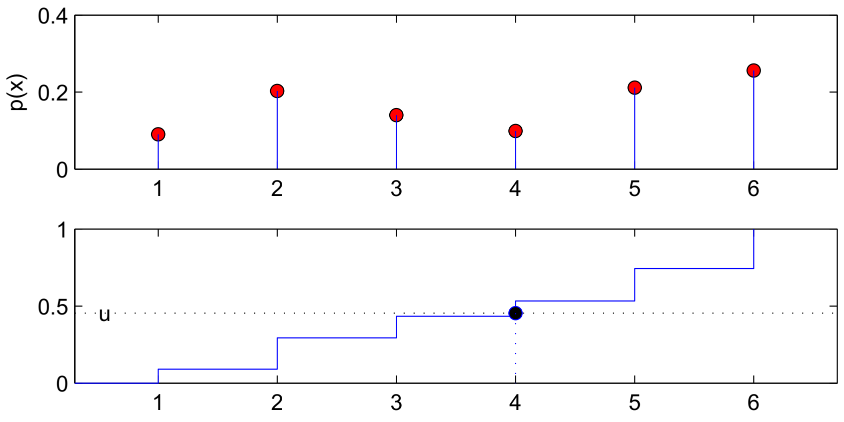

FIGURE 19.3

Generation of samples from a discrete distribution. We generate u uniformly on $[0, 1)$ and find the corresponding x via the generalised inverse $F^{-}(u)$. The discrete case is particularly intuitive: think of dividing a stick of length 1 into pieces of length $\pi_i = f_X(x_i)$ and label each region with x_i. Select a point u uniformly random and return the label of the region that u falls in.

By taking the derivative, we see that the density of u is uniform. The argument works also in the opposite direction: when we first choose uniform U and pass it through the function $X = F_X^{-1}(U)$, the density of X will be f_X. To allow for discrete distributions or continuous densities with atoms (where single points have positive probability mass), we define the generalised inverse CDF

$$F^{-}(u) \equiv \inf\{x : F(x) \geq u\}.$$

The x generated by this method are guaranteed to be distributed by f_X. Figure 19.3 illustrates this construction. The generalized inverse allows for jumps in the cumulative density when atoms have nonzero probability (see Figure 19.4).

Example 1. Exponential random variables can be derived by the inversion method from uniform samples. The exponential density with *rate* parameter λ is

$$\mathcal{E}(x; \lambda) = \lambda \exp(-\lambda x).$$

The CDF of the exponential distribution is found as

$$F(x) = \int_0^x \lambda \exp(-\lambda \tau) = -\exp(-\lambda x) + 1.$$

The generalised inverse is obtained via

$$u = F(x) = 1 - \exp(-\lambda x),$$
$$x = -\log(1 - u)/\lambda = F^{-}(u).$$

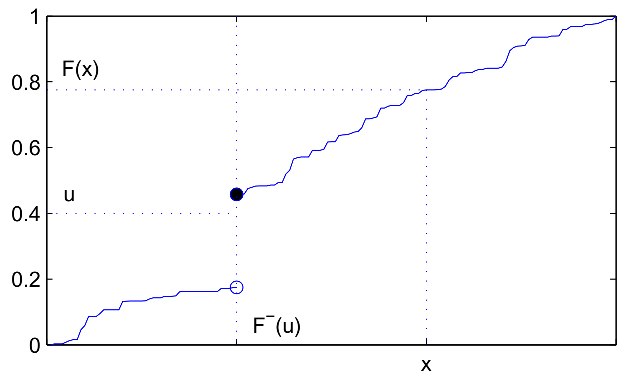

FIGURE 19.4

Visualisation of the generalised inverse. The blue curve shows a CDF of a probability distribution with an atom, as can be seen from the discontinuity. For a given probability u, $F^{-}(u)$ is the "minimum" x such that the probability of the interval $(\infty, x]$ is equal or exceeds u.

We can even avoid calculating $1-u$ by noting that both u and $w = 1-u$ are distributed uniformly on $[0, 1]$, so

$$x = F^{-}(w) = -\log(w)/\lambda.$$

1.19.3.2 Transformation

The generalised inverse method works by applying a suitable function (the generalized inverse $F^{-}(u)$) on a random input with a known density (u with a uniform density). This method can be made more general by a technique known as the change of variables [4].

To illustrate the transformation method, we will first give an example where the mapping function will be affine (that is linear plus a constant term). Suppose we are given a random variable X with density $f_X(x)$ and wish to find the density of Y where

$$Y = aX + b.$$

It is clear that the density of Y, denote by f_Y should be somewhat related to f_X. For example if X is uniform ($f_X = \mathcal{U}$) on $[0, 1)$ and $a = 2$ and $b = -1$, one could see that Y is uniform on $[-1, 1)$. The general case, when f_X is any density, is perhaps slightly harder to see. To find the density of Y, we first will find the CDF $F_Y(y)$ of Y and obtain the desired density $f_Y(y)$ by taking the derivative w.r.t. y. In particular,

$$F_Y(y) = \Pr\{Y \le y\} = \Pr\{aX + b \le y\} = \begin{cases} \Pr\{X \le (y-b)/a\}, & a > 0, \\ \Pr\{X \ge (y-b)/a\}, & a < 0. \end{cases}$$

By taking derivative we see that the density is given by

$$f_Y(y) = \frac{dF_Y}{dy} = \frac{1}{|a|} f_X((y-b)/a).$$

For the more general case, we have a more general mapping g where

$$Y = g(X),$$
$$X = g^{-1}(Y).$$

The density can be found as

$$F_Y(y) = \Pr\{Y \leq y\} = \Pr\{g(X) \leq y\} = \Pr\{X \leq g^{-1}(y)\}$$
$$f_Y(y) = \frac{dF_Y}{dy} = f_X(g^{-1}(y))\left|\frac{dg^{-1}(y)}{dy}\right| \equiv f_X(x(y))|J(y)| \tag{19.2}$$

here, the derivative of g^{-1} is denoted as the *Jacobian* and the absolute value ensures the positivity of the densities. In many textbooks it is common to drop the explicit reference to the function g; authors often use $y(x)$ for $y = g(x)$ and $x(y)$ for the inverse $g^{-1}(y)$. In the sequel, we will also adopt this notation. The multivariate case is analogous; here we illustrate the bivariate case, the most encountered case in derivations. The function g, that is also called a *transform*, is defined as

$$(y_1, y_2) = g(x_1, x_2).$$

If we can invert g, which is not always easy to do analytically, we find

$$x_1 = x_1(y_1, y_2),$$
$$x_2 = x_2(y_1, y_2).$$

The derivative term is the *Jacobian* determinant defined as

$$J = \begin{vmatrix} \partial x_1/\partial y_1 & \partial x_2/\partial y_1 \\ \partial x_1/\partial y_2 & \partial x_2/\partial y_2 \end{vmatrix} = J(y_1, y_2).$$

Consequently, the density of the transformed variable, similar to Eq. (19.2), is given by

$$f_Y(y_1, y_2) = f_X(x_1(y_1, y_2), x_2(y_1, y_2))|J(y_1, y_2)|.$$

Example 2. To illustrate the method, we describe the Box-Müller method for generating normal random variables. The Box-Müller method generates the pair (x_1, x_2)

$$x_1 \sim \mathcal{E}(x_1; 1/2) = \frac{1}{2}\exp(-x_1/2),$$
$$x_2 \sim \mathcal{U}(x_2; 0, 2\pi) = \frac{1}{2\pi}\mathbb{I}\{0 \leq x_2 \leq 2\pi\},$$

where x_1 is the square of the magnitude and x_2 is the angle of a point in polar coordinates. The method transforms this point into cartesian coordinates

$$y_1 = \sqrt{x_1}\cos(x_2),$$
$$y_2 = \sqrt{x_1}\sin(x_2).$$

It turns out, that y_1 and y_2 obtained via this method are independent and unit Gaussian ($\mathcal{N}(0,1)$) distributed. To show this, we find the inverse mapping

$$\begin{aligned} x_1 &= y_1^2 + y_2^2, \\ x_2 &= \arctan(y_2/y_1). \end{aligned}$$

The Jacobian determinant is found as

$$J = \begin{vmatrix} \partial x_1/\partial y_1 & \partial x_2/\partial y_1 \\ \partial x_1/\partial y_2 & \partial x_2/\partial y_2 \end{vmatrix} = \begin{vmatrix} 2y_1 & \frac{1}{1+(y_2/y_1)^2}\frac{-y_2}{y_1^2} \\ 2y_2 & \frac{1}{1+(y_2/y_1)^2}\frac{1}{y_1} \end{vmatrix} = 2.$$

The resulting density is

$$\begin{aligned} f_Y(y_1, y_2) &= \mathcal{E}(x_1(y_1, y_2); 1/2)\mathcal{U}(x_2(y_1, y_2); 0, 2\pi)|J(y_1, y_2)| \\ &= \frac{1}{2}\exp(-(y_1^2 + y_2^2)/2)\frac{1}{2\pi}2 = \frac{1}{\sqrt{2\pi}}\exp(-y_1^2/2)\frac{1}{\sqrt{2\pi}}\exp(-y_2^2/2) \\ &= \mathcal{N}(y_1; 0, 1)\mathcal{N}(y_2; 0, 1). \end{aligned}$$

1.19.3.3 Rejection sampling

Rejection sampling is a technique for indirectly sampling from a target distribution π by sampling from a *proposal* distribution q. We reject some of the generated samples to compensate for the fact that $q \neq \pi$. The algorithm is as follows: We sample $x^{(i)}$ for $i = 1, \ldots, N$ independently from q. We accept the sample $x^{(i)}$ with acceptance probability $a(x^{(i)})$ where the acceptance probability is

$$a(x) \equiv \frac{\pi(x)}{Mq(x)}.$$

Here, M is a positive number that guarantees $\pi(x) \leq Mq(x)$ for all x, i.e., that $Mq(x)$ completely covers the density $\pi(x)$. The approach also works when the normalization constant Z of π is unknown, that is we can only evaluate ϕ pointwise where

$$\pi(x) = \frac{1}{Z}\phi(x).$$

In this case, we let the acceptance probability be

$$\tilde{a}(x) \equiv \frac{\phi(x)}{\tilde{M}q(x)},$$

where $\phi(x) \leq \tilde{M}q(x)$ for all x. If a suitable M (or $\tilde{M}$) can be found, the algorithm is simple to implement and requires only sampling from q and pointwise evaluation of ϕ. To understand the idea behind rejection sampling, we observe that for any density function $f(x)$, (including obviously our target π and the proposal q) we have the following identity:

$$f(x) = \int_0^{f(x)} 1\, d\tau = \int \mathbb{I}\{0 \leq \tau \leq f(x)\} d\tau. \tag{19.3}$$

In other words, $f(x)$ can be written as a *marginal density* of a distribution with density $\mathbb{I}\{0 \le \tau \le f(x)\}$ on the extended space (x, τ). This density is simply the uniform density over the volume under the curve $f(x)$. The equation Eq. (19.3) says that generating samples from $f(x)$ is equivalent to uniformly generating samples on the set $A = \{(x, \tau) : 0 \le \tau \le f(x)\}$ and then simply ignoring the τ coordinate. When $f(x) = \phi(x)/Z$ and we don't know Z

$$f(x) = \frac{1}{Z}\phi(x) = \frac{1}{Z}\int_0^{\phi(x)} 1 d\tau = \frac{1}{Z}\int \mathbb{I}\{0 \le \tau \le \phi(x)\}d\tau. \tag{19.4}$$

By integrating over x on both sides we get the normalization constant Z as the area of the region $A = \{(x, \tau) : 0 \le \tau \le \phi(x)\}$:

$$Z = \int\int \mathbb{I}\{0 \le \tau \le \phi(x)\}d\tau\, dx.$$

Hence

$$f(x) = \frac{\int \mathbb{I}\{0 \le \tau \le \phi(x)\}d\tau}{\int\int \mathbb{I}\{0 \le \tau \le \phi(x)\}d\tau\, dx}.$$

This equation suggests that $f(x)$ is the marginal density of uniform density over the set A. A useful metaphor for understanding the concept is of thinking fishes in a lake [5], where x is a coordinate on the water surface and τ is the depth, which is illustrated in Figure 19.5. If all fishes in the lake are distributed uniformly in the water, when viewed from above they will be marginally distributed with density $f(x)$ (see Figure 19.6).

Note that the reasoning of Eq. (19.3) works also in the reverse direction: when we have samples $x^{(i)}$ for $i = 1, \ldots, N$ generated from a proposal $q(x)$, we can select the τ coordinate for each $x^{(i)}$ simply by choosing $\tau^{(i)}$ uniformly on $[0, Mq(x^{(i)})]$. The resulting pairs $(x^{(i)}, \tau^{(i)})$ will be uniformly distributed on the set $A_{Mq} = \{(x, \tau) : 0 \le \tau \le Mq(x)\}$. Provided that $\phi(x) \le Mq(x)$, if we now select only the pairs such that $\tau^{(i)} < \phi(x^{(i)})$, these accepted tuples will be distributed uniformly on the set $A_\phi = \{(x, \tau) : 0 \le \tau \le \phi(x)\}$. But, by Eq. (19.4), the marginal of this uniform density will be p.

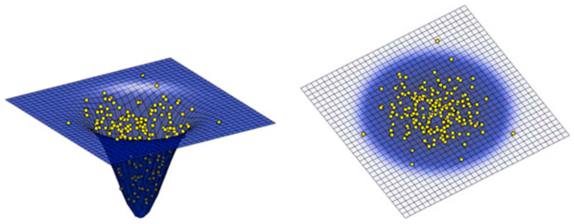

FIGURE 19.5

Fishes in a lake. We view the bivariate density function $f(x)$ as the bottom surface of a lake. Take uniformly distributed points in this volume. When viewed from above, i.e., when we ignore their depth, the points will be distributed more densely where the lake is deeper.

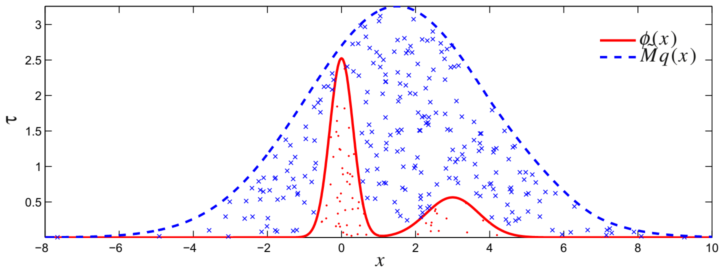

FIGURE 19.6

Rejection sampling. Sampling x from the proposal $q(x)$ and then sampling the associated τ uniformly on $[0, \tilde{M}q]$ results in uniformly distributed tuples (x, τ) (all the points under $\tilde{M}q(x)$). Restricting those to only points such that $\tau \leq \phi(x)$ provides the accepted points (red dots), the other points are rejected (blue crosses). Note that the probability $\Pr\{\tau \leq \phi(x)\} = \phi(x)/\tilde{M}q(x)$ is the acceptance probability $\tilde{a}(x)$.

Algorithm 1. Rejection Sampling

```
1: Construct an easy to sample density $q(x)$ and positive number $\tilde{M}$ such that $\phi(x) < \tilde{M}q(x)$
2: for $i = 1, 2, 3, \ldots$ do
3:     Draw $x^{(i)} \sim q(x)$
4:     Accept $x^{(i)}$ with probability $\tilde{\alpha}(x^{(i)}) = \phi(x^{(i)})/\tilde{M}q(x^{(i)})$
5: end for
```

1.19.4 Markov Chain Monte Carlo

In Markov Chain Monte Carlo methods [6,17–20], instead of directly attempting to sample from $\pi(x)$, we sample from a Markov chain with transition density ("kernel") $K(x^{(n)}|x^{(n-1)})$ where n indexes the sample number.[3] To see how this connects to our ultimate desire to sample from a target $\pi(x)$ consider first drawing samples from the Markov chain. Given an initial state $x^{(1)}$, we draw $x^{(2)}$, then conditioned on $x^{(2)}$ we draw $x^{(3)}$ and so on by iteratively drawing from $K(x^{(n)}|x^{(n-1)})$. The distribution of state occupancy at time n, $\pi_n(x)$, is found from

$$\pi_n(x) = \int K(x|x')\pi_{n-1}(x')dx'$$

[3]Note that, the sample number is conceptually different than the "time" index in a time series model; for example, in a time series model x_t corresponds to the state at time t, whereas here x^n can correspond to a vector of all state variables—a full state trajectory. In this context, it is convenient to think of n simply as the iteration number of the sampling algorithm and $x^{(n)}$ as a particular realisation of all random variables being sampled.

which we write more compactly as $\pi_n = K\pi_{n-1}$. This generates a sequence of marginal distributions $\pi_1, \pi_2, \ldots$ Does this sequence of distributions converge and, if so, what does it converge to ? A stationary distribution, π satisfies $\pi = K\pi$. A key result here is that for an ergodic chain (i.e., irreducible (chain) and aperiodic (chain)) there exists a unique distribution π for which $\pi = K\pi$ and therefore a unique stationary distribution to which all the occupancy distributions converge, irrespective of their initial states [7]. For finite state Markov chains, irreducibility means that each state can be visited starting from each one (related to connectedness of the state transition diagram) and aperiodicity means that each state can be visited at any time n larger than some fixed number.

The first question is, given a Markov chain with transition kernel K, how to find the marginal distribution $\pi_n(x)$ for large n. The second question is how to construct a transition kernel that is easy to sample from and converges to the desired target distribution. In the finite state case, a solution to the first question is provided by the eigenvalue-eigenvector decomposition of the transition matrix that will be introduced in the next section. The answer to the second question turns out to be very elegant and simple and underlies the celebrated Metropolis-Hastings Markov Chain Monte Carlo method [2,21] that will be covered afterwards.

1.19.4.1 Stationary distribution of a Markov Chain with finite number of states

We first focus our attention on the convergence of a Markov chain and illustrate results in the finite state case. The finite state case may seem to be too restrictive, and in a sense it indeed is. From a mathematical perspective, the theory of Markov chains for more general spaces is more delicate and it does not rely on linear algebra. Nevertheless, it turns out that the results for more general state spaces have similar flavor and understanding what is happening in the finite case provides the basic intuition about the issues involved. For technicalities on countable state spaces, see [7] or uncountable state spaces, see [22], Chapter 13.

For the finite state case, we proceed as follows. First, we note that the marginals of a Markov chain satisfy the following recursive relation

$$\pi_n(x_n) = \sum_{x_{n-1}} K(x_n|x_{n-1})\pi(x_{n-1}).$$

Now, a probability mass function on a finite state space is simply a vector of positive entries which sum up to one so in index notation we write

$$\pi_n(i) = \sum_j K_{i,j}\pi_{n-1}(j),$$

where the marginal distribution of the chain is denoted by $\pi_n(i) \equiv \pi_n(x_n)$ and the transition model is denoted by the transition matrix K with the entries $K_{i,j} = K(x_n = i|x_{n-1} = j)$. Adopting the matrix notation, we write $\pi_n = K\pi_{n-1}$. This gives the expression for the marginal π_n as a function of the initial occupancy density π_0

$$\pi_n = K^n\pi_0.$$

This is simply a (linear) fixed point iteration and the question reduces to finding a stationary point π that satisfies

$$\pi = \lim_{n\to\infty} K^n\pi_0 \equiv K^\infty\pi_0,$$

where $K^{\infty} = \lim_{n\to\infty} K^n$. An eigen-decomposition of the transition matrix K

$$K = D\Lambda D^{-1}$$

provides insight about the limit behavior of K^n where D is the matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues. It turns out that for transition matrices K, the maximum eigenvalue is always 1. If the magnitude of the second largest eigenvalue λ_2 is strictly less, $|\lambda_2| < 1$, we have a unique stationary distribution. To see this consider

$$K^n = D\Lambda D^{-1} D\Lambda D^{-1} \cdots D\Lambda D^{-1} = D\Lambda^n D^{-1}.$$

But as Λ is diagonal, Λ^n for large n is a matrix very close to all zeros matrix with only a single 1 on the diagonal. Say without loss of generality that the first eigenvalue $\lambda_1 = 1$ is at position $\Lambda(1, 1)$ (otherwise construct a permutation matrix P and consider $T = (DP^{\top})(P\Lambda P^{\top})(PD^{-1}) = \tilde{D}\tilde{\Lambda}\tilde{D}^{-1}$). Then K^{∞} is the rank-one matrix given by the outer product

$$K^{\infty} = D(\,:, 1)D^{-1}(1, :\,).$$

Here, $D(\,:, i)$ denotes the ith column of the matrix D and $D^{-1}(i, :\,)$ is the ith row of D^{-1}. These are respectively the right and left eigenvectors. But what is $D^{-1}(1, :\,)$? By definition we have

$$\begin{aligned} D^{-1}K &= \Lambda D^{-1}, \\ D^{-1}(1, :\,)K &= \Lambda(1, 1)D^{-1}(1, :\,) = D^{-1}(1, :\,). \end{aligned}$$

But as K is a transition matrix, all its columns sum up to one. In other words, if we would left multiply K by an all-ones vector v, we get $vK = v$ so it is easy to see that the left eigenvector corresponding to the largest eigenvalue is proportional to the all ones vector

$$D^{-1}(1, :\,) = \begin{pmatrix} 1 & 1 & \ldots & 1 \end{pmatrix}.$$

But this implies that the columns of K^{∞} are all the same as

$$K^{\infty} = [D(\,:, 1), D(\,:, 1), \ldots, D(\,:, 1)].$$

We can conclude that, regardless of the initial state, the probabilities are all the same and the stationary density is proportional to the (right) eigenvector corresponding to the eigenvalue $\lambda_1 = 1$. This is pictorially illustrated using a random transition matrix K in Figure 19.7. In the MC literature, we use the term transition kernel instead of transition matrix as the latter is restricted only to countable state spaces.

A natural question to ask is how fast a chain converges to stationarity. For Markov Chains, the rate of convergence is governed by the second eigenvalue. The convergence to the stationary distribution is geometric and is given by

$$\|K^n\pi_0 - \pi\|_{\text{var}} \leq C|\lambda_2|^n,$$

where C is a positive constant and the function $\|\cdot\|_{\text{var}}$ measures the distance between two densities. It is known as the total variation norm and is given by

$$\|P - Q\|_{\text{var}} \equiv \frac{1}{2}\sum_{s\in\mathcal{X}} |P(s) - Q(s)|.$$

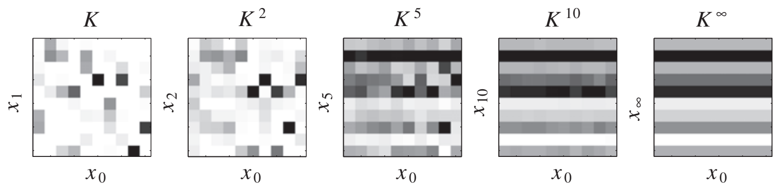

FIGURE 19.7

The powers K^τ of a transition matrix $K(x_n = i|x_{n-1} = j)$ with $\tau = 1, 2, 5, 10, \infty$. Darker colors correspond to higher transition probabilities. The identical columns of K^∞ depicts the stationary distribution, which is reached independent of any starting density. We see that for this transition matrix K, even after a few iterations τ the columns of K^τ are close to the stationary density, suggesting a rapid convergence. The magnitude of the second eigenvalue is $|\lambda_2| \approx 0.64$ which verifies that our observation is indeed correct.

However, for a large transition matrix, it may be hard to show algebraically that $|\lambda_2| < 1$. Fortunately, there is a convergence result that states that if K is irreducible and aperiodic, then there exist $0 < r < 1$ and $C > 0$ s.t.

$$\|K^n \pi_0 - \pi\|_{\text{var}} \leq Cr^n,$$

where π is the invariant distribution [20]. This result also generalizes to more general Markov chains, such as on countable infinite or uncountable state spaces where explicit calculation of the second largest eigenvalue of the transition kernel is intractable.

To see an example how convergence speed is affected consider a very simple chain x_n with two states, labeled as 1 and 2 with the transition matrix

$$K = \begin{pmatrix} \epsilon & 1-\epsilon \\ 1-\epsilon & \epsilon \end{pmatrix}, \tag{19.5}$$

where $0 \leq \epsilon \leq 1$. In other words, if at time t the chain is in state i, with probability ϵ, the chain stays put, or jumps to the other state with probability $1-\epsilon$. One natural question to ask is: where would we find the chain if we stop it at a large n ? Say we start initially from state i but in the previous section we saw that this initial choice won't matter (most of the time). It turns out that for $0 < \epsilon < 1$ the answer is 0.5: it is equally likely to find the chain in any of the two states, regardless of the initial state but for each ϵ the rate of convergence is different. We illustrate these concepts on Figure 19.8.

1.19.4.2 Designing a Markov Chain for a given target distribution: The Metropolis-Hastings method

The discussion of the previous section suggests that, if we can design a transition kernel K such that the target density $\pi(x)$ is its stationary distribution, at least in principle one can generate samples from the Markov chain that eventually will tend to be drawn from the target distribution. After ignoring samples obtained from an initial "burn in" period, as the chain moves towards the stationary distribution, the generated samples can be subsequently used to estimate expectations under the target distribution, as if they are independent. There are a couple of caveats: formally we require the chain to be

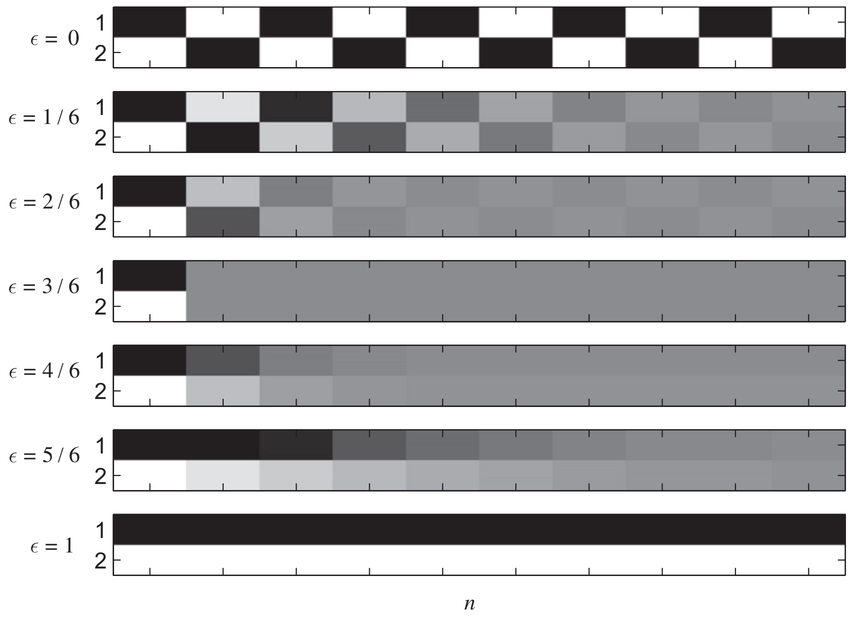

FIGURE 19.8

Convergence to the stationary distribution for the transition matrix K defined in Eq. (19.5) with $\epsilon = 0, 1/6, 2/6, \ldots, 1$ from top to bottom. Horizontal axis denotes the time n and the vertical axis is the state x. The probability $\pi_n(x)$ is high for darker regions. For $\epsilon = 1$ (bottom), the state transition matrix is $K = I$ and the states are no longer connected. The chain fails to be irreducible and the chain fails to have a unique stationary distribution. For $\epsilon = 0$ (top), the states alternate periodically. The chain is periodic and similarly fails to have a unique stationary distribution. For all other cases with $0 < \epsilon < 1$ we have convergence to the uniform distribution, whilst with different speed. The second eigenvalues are $\lambda_2 = 2\epsilon - 1$; as expected the chains converge slower when $|\lambda_2|$ is closer to one.

ergodic—otherwise the chain might never reach the desired stationary distribution. Even if the chain is ergodic, typically we would not know in practice how close the chain is to the stationary distribution.

Here we set aside the issue of ergodicity and concentrate on designing a transition kernel K for a given target π. This is surprisingly simple via the approach proposed by Metropolis [1], later generalised by Hastings [21], for a rigorous treatment see [17,23]. Note that this approach does not provide explicit formula for the transition Kernel K (which is often intractable to compute even in finite state spaces); it merely states a procedure to draw a new sample given the previous one. Suppose we are given a target density $\pi = \phi/Z$, where Z is a (possibly unknown) normalisation constant. The Metropolis-Hastings (MH) algorithm uses a proposal density $q(x'|x)$, which generates a candidate sample x' possibly dependent[4] on the current sample x. The algorithm is very simple, we define a chain by the

[4]Note that whilst one can consider proposals $q(x')$ which are independent of x, the actual MH transition $K(x'|x)$ nevertheless depends on x.

following rule: Accept the proposed sample with probability

$$\alpha(x \to x') = \min\left\{1, \frac{q(x|x')\pi(x')}{q(x'|x)\pi(x)}\right\},$$

where α is the *acceptance probability* satisfying clearly the condition $0 \le \alpha(x|x') \le 1$. If the sample is not accepted, the chain stays put at the old location x. This procedure implicitly defines the following MH transition kernel K:

$$K(x|x') = q(x|x')\alpha(x|x') + \delta(x - x')\rho(x'), \tag{19.6}$$

here, $\delta(x)$ is the Dirac delta ρ is the *rejection probability* satisfying $0 \le \rho(x') \le 1$ is given by

$$\rho(x') \equiv \int (1 - \alpha(x|x'))q(x|x')dx.$$

This simply sums up all the probability of rejection. Note that we don't directly sample from K, we merely sample from the proposal q but reject some of the samples. In practice we typically don't need to evaluate K – indeed very often this is intractable. Recall that the stationary distribution should satisfy the following condition

$$\pi(x) = \int K(x|x')\pi(x')dx'.$$

Algorithm 2. Metropolis-Hastings

```
1: Initialise $x^{(1)}$ arbitrarily
2: for $n = 2, 3, \ldots$ do
3:     Propose a candidate: $x^{new} \sim q(x^{new}|x^{(n-1)})$
4:     Compute acceptance probability:
           $\alpha(x^{new}\,|\,x^{(n-1)}) = \min\left\{1, \frac{q(x^{(n-1)}|x^{new})\pi(x^{new})}{q(x^{new}|x^{(n-1)})\pi(x^{(n-1)})}\right\}$
5:     Sample from uniform distribution: $a \sim \mathcal{U}(0, 1)$
6:     if $a < \alpha$ then
7:         Accept candidate: $x^{(n)} \leftarrow x^{new}$
8:     else
9:         Reject candidate: $x^{(n)} \leftarrow x^{(n-1)}$
10:    end if
11: end for
```

For an arbitrary kernel K, it is hard to verify stationarity. However, if K happens to satisfy a stronger condition,[5] known as the *detailed balance condition*

$$K(x'|x)\pi(x) = K(x|x')\pi(x'),$$

[5]A sufficient condition; a kernel K may have π as the stationary distribution while not satisfying detailed balance.

one sees directly that π must be the stationary distribution by integrating both sides over x'. Verification of detailed balance for the Metropolis-Hastings kernel is straightforward:

$$\begin{aligned}
K(x|x')\pi(x') &= (q(x|x')\alpha(x|x') + \delta(x-x')\rho(x'))\pi(x') \\
&= q(x|x')\min\left\{1, \frac{q(x'|x)\pi(x)}{q(x|x')\pi(x')}\right\}\pi(x') + \delta(x-x')\rho(x')\pi(x') \\
&= \min\{q(x|x')\pi(x'), q(x'|x)\pi(x)\} + \delta(x-x')\rho(x')\pi(x') \\
&= q(x'|x)\min\left\{\frac{q(x|x')\pi(x')}{q(x'|x)\pi(x)}, 1\right\}\pi(x) + \delta(x'-x)\rho(x)\pi(x) \\
&= K(x'|x)\pi(x).
\end{aligned}$$

Note that to compute the acceptance probability α, we only need to evaluate π up to a normalisation, since the normalisation constant cancels out. For a given target π and assumed proposal $q(x'|x)$ we now have a procedure for sampling from the Markov chain $K(x'|x)$ with stationary distribution π. Sampling from the transition (19.6) can be achieved using the procedure detailed in Algorithm 2.

1.19.4.3 Illustration

In this section, we will derive a MH algorithm for sampling from a discrete distribution $\pi(x) = \phi(x)/Z$ where $x \in \{1, 2, \ldots, 20\}$. The target, which has two bumps is shown on Figure 19.9. Our proposal $q(x'|x)$ is a mixture of a random walk with a uniform density: with probability ν, we choose a random walk. If we choose the random walk, with equal probability we let $x' \leftarrow \max\{0, x-1\}$ or $x' \leftarrow \min\{20, x+1\}$. Alternatively, with probability $1-\nu$, we choose x' uniformly on $\{1, 2, \ldots, 20\}$. The random walk component corresponds to a "local exploration" whereas the uniform jump corresponds to a "restart." Here, the parameter ν is a algorithm parameter that will effect the behavior of our overall proposal and we will show how this may effect the convergence rate of the resulting MH algorithm. In Figure 19.10, we show the proposals and corresponding MH transition matrices.

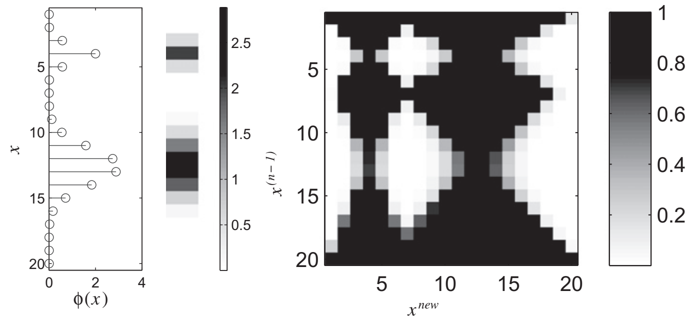

FIGURE 19.9

Target density and the acceptance probability $\alpha(x^{new}|x^{(n-1)})$ under a symmetric proposal $q(x^{new}|x^{(n-1)}) = q(x^{(n-1)}|x^{new})$.

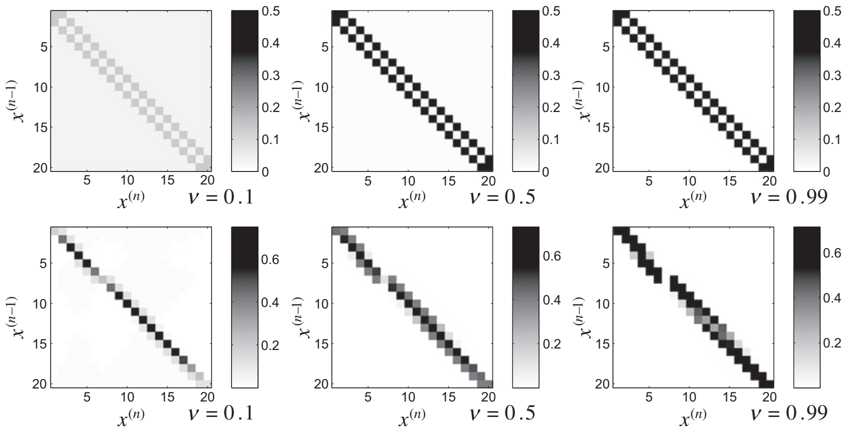

FIGURE 19.10

(Top, left to right) The proposal density with $\nu = 0.1, 0.5, 0.99$. (Bottom, left to right) The corresponding transition matrices K, computed by Eq. (19.6).

A moments thought would reveal that the local moves are not sufficient to cross the low probability region between the two bumps of the target, so setting ν close to 1 might be a poor choice in terms of the convergence, also known as the *mixing time*. In this case, having the uniform proposal is certainly useful as it may help the chain to investigate the state space, hence reducing the mixing time. However, longer jumps have the risk of hitting low probability regions, increasing the rejections. The optimal choice for ν seems to be somewhere in between those two extremes and we will numerically illustrate this.

In this simple example, we can numerically compute the effective MH transition matrix and compute the *spectral gap*, given as $1 - |\lambda_2|$; the distance between magnitudes of two largest eigenvalues. For all ν, the spectral gap is plotted in Figure 19.11. The bigger this gap, the faster is the convergence: an illustration of this fact is shown in Figure 19.12.

For real problems, the above computations are typically intractable but fortunately they are also not needed at all for the application of the MH algorithm. All we need is to sample from the proposal and evaluate the acceptance probability. In Figure 19.13, we show the results that would reflect a typical MH run. For different parameter settings with $\nu = 0.1, 0.5, 0.99$, we generate sample paths using the algorithm in 2. Then, following a burn in period of 100 iterations, for each state x we simply count the number of times the chain visits x to estimate $\pi(x)$. Formally, we construct the estimator

$$\bar{\pi}_n(x) = \frac{1}{n-100} \sum_{\tau=101}^{n} [x^{(\tau)} = x]. \tag{19.7}$$

Note that, while the samples $x^{(\tau)}$ are correlated, we calculate our estimate as if they are independent. This is still valid as after convergence all the $x^{(\tau)}$ are effectively drawn from the same stationary distribution $\pi(x)$. The results are shown in Figure 19.13.

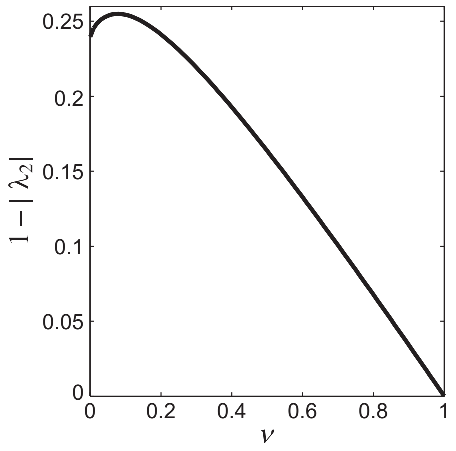

FIGURE 19.11

The spectral gap $(1-|\lambda_2|)$ computed numerically for the transition matrix K obtained using the proposals shown in Figure 19.10. The picture suggests that for fastest convergence, we need to set $\nu \approx 0.1$. Only using the local moves ($\nu = 1$ case) results in poor mixing and uniform proposal seems to be a better choice for this problem. Yet, using local moves occasionally improves mixing as the gap at $\nu = 0.1$ is slightly bigger than $\nu = 0$.

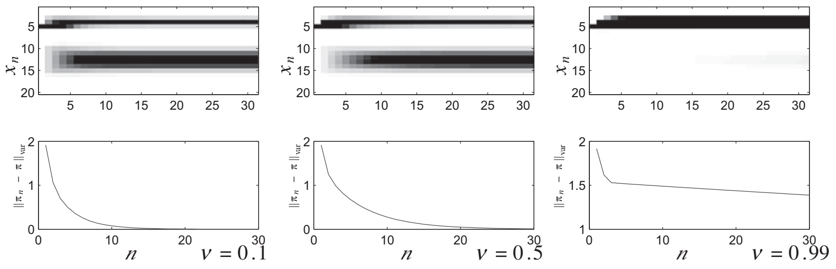

FIGURE 19.12

Convergence to the stationarity when using the proposal with parameters $\nu = 0.1, 0.5, 0.99$. (Top) The marginals π_n, as computed via $K^n\pi_0$ where darker color indicates higher probability. (Bottom) The total variation distance between stationary distribution $\pi = \pi_\infty$ and the marginal π_n. As predicted, the distance drops very slowly for $\nu = 0.99$ and faster for the other cases. This example suggests that the algorithm is not very sensitive to ν and a broad range of values are useful in practice.

Example 3 (Object position estimation from range measurements). In this section, we will illustrate the MH algorithm on a synthetic example of object localization. In this example, we have 3 noisy sensors at known positions s_j, $j = 1, \ldots, 3$, each measuring with noise the distance to an object at an unknown

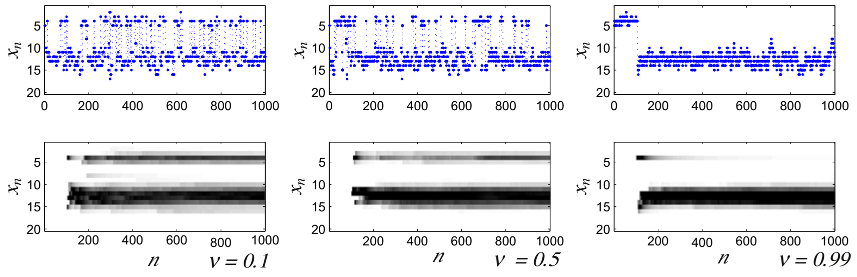

FIGURE 19.13

Sample paths and estimates of the marginal when using the proposal with parameters $\nu = 0.1, 0.5, 0.99$. (Top) An example of a sample paths generated by the algorithm. (Bottom) Estimate of the marginal $\bar{\pi}_\tau(x)$ (see Eq. (19.7)) using averages over a single realization after an initial burn-in period of 100 iterations.

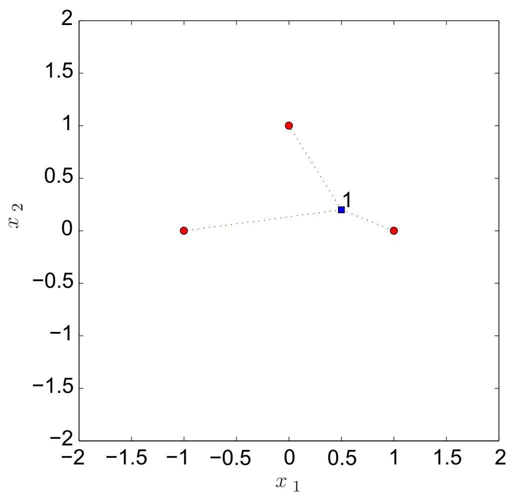

FIGURE 19.14

Sensor locations s_1, s_2 and s_3 (round red points) and the true object position (blue square). The observations are noisy versions of the true distances.

location x. The setup is shown in Figure 19.14. Each sensor observes a noisy distances with the following model

$$y_j | x, s_j \sim \mathcal{N}(y_j; \|x - s_j\|, R),$$

where $\|x\| \equiv \left(\sum_k x_k^2\right)^{1/2}$ denotes the Euclidian distance and R is the noise variance of each sensors. We assume that we don't have any information about the position of the object so we choose a flat prior

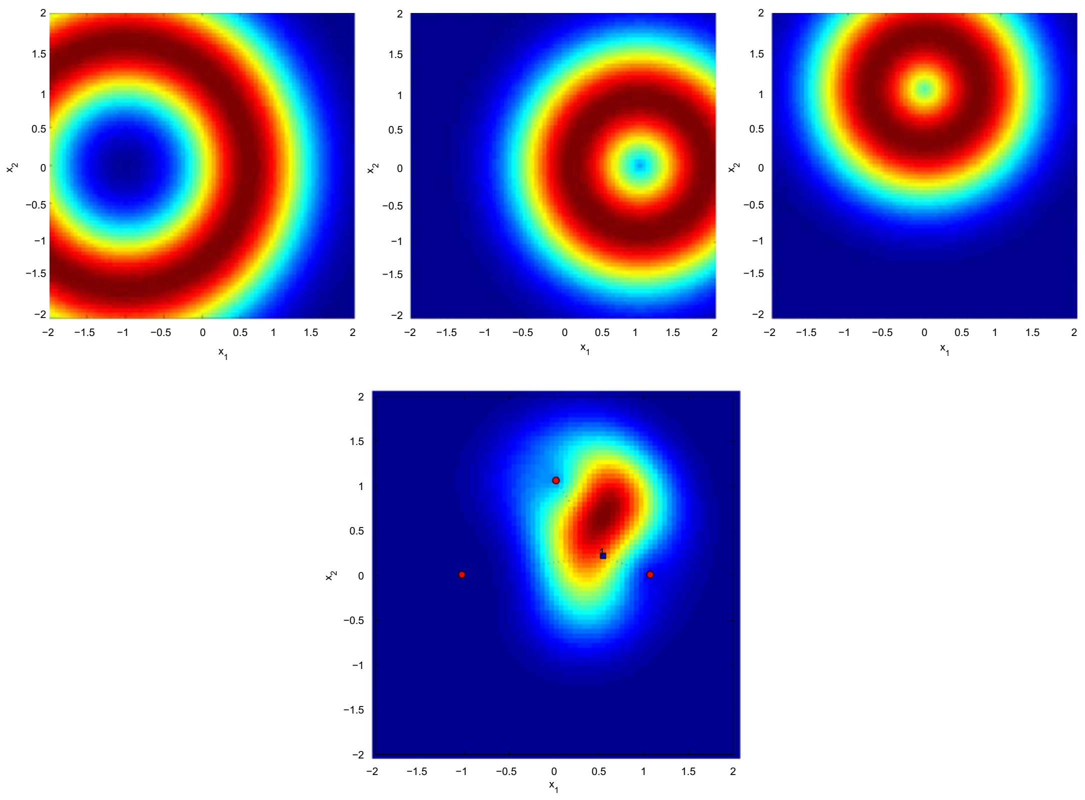

FIGURE 19.15

(Top) The likelihood terms $p(y_i|x)$, (Bottom) The target posterior, proportional to $\phi(x) = \prod_i p(y_i|x)$.

$p(x) \propto 1$. The solution for this problem is given by the posterior

$$\pi(x) = p(x|y_1, y_2, y_3) = \frac{1}{Z} p(y_1|x)p(y_2|x)p(y_3|x)p(x) \equiv \frac{1}{Z}\phi(x).$$

The true target posterior density is depicted in Figure 19.15. As the target posterior $\pi(x)$ is nonstandard, we will sample from it using a MH algorithm. We design a MH algorithm with a symmetric random walk proposal where $q(x'|x) = \mathcal{N}(x, \sigma^2 I)$. The acceptance probability is

$$\alpha(x \to x') = \min\left\{1, \frac{\phi(x')}{\phi(x)}\right\}.$$

Note that the symmetric proposal has been canceled out. By construction, the resulting kernel will admit π as the unique stationary density. In Figure 19.16, we show a sequence of samples obtained from this MH algorithm. We see that the chain visits states around the high probability region. The consecutive

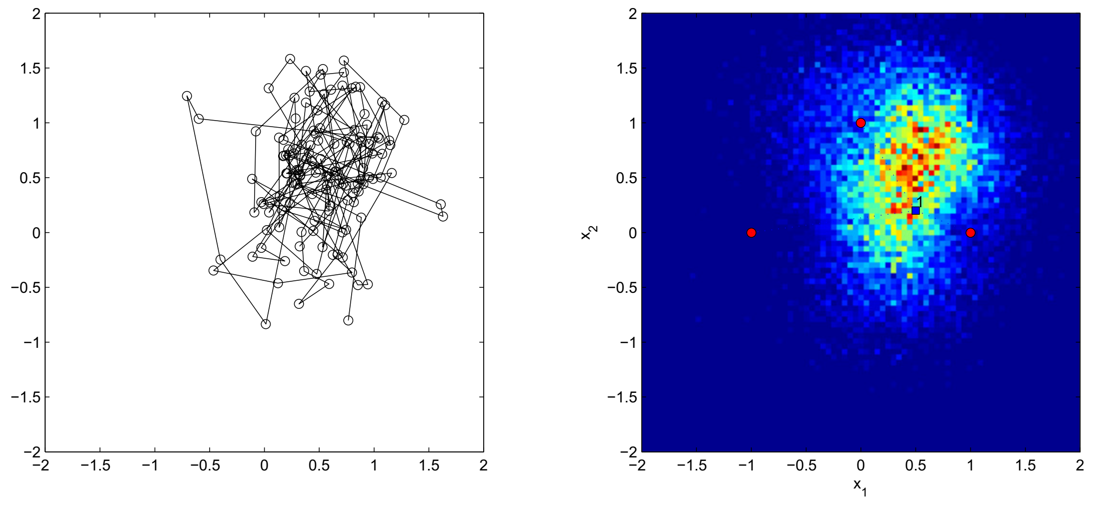

FIGURE 19.16

(Left) A state trajectory obtained from the MH algorithm. (Right) A 2-D histogram estimate of the target density (from a longer state trajectory).

samples are dependent (as can be seen from the connecting lines) but as expected, a 2-D histogram obtained from a single chain shows a fairly accurate approximation.

1.19.4.4 The Gibbs sampler

The MH schema is a very general technique for deriving MCMC algorithms. This generality stems partially from the arbitrariness of the proposal q; in complex models the design of a reasonable proposal such that the chain mixes well can require a lot of work. It would be desirable if somehow the proposal design phase could be automated. The Gibbs sampler [20,24] is an attempt in this direction.

The Gibbs sampler is suitable for sampling a multivariate random variable $x = (x_1, \ldots, x_D)$ with intractable joint target density $\pi(x)$. Gibbs sampling proceeds by partitioning the set of variables x into a chosen variables x_d and the rest, $x = (x_d, x_{-d})$. The assumption is that the *full conditional densities* $\pi(x_d|x_{-d})$ are tractable. One then proceeds coordinatewise by sampling from full conditionals as in Algorithm 3. The Gibbs sampler is actually a MH schema with a sequence of proposals for $d = 1, \ldots, D$

$$q_d(x|x') = p(x_d|x'_{-d})\delta\left(x_{-d} - x'_{-d}\right)$$

each corresponding to a Gibbs transition kernels K_d. Moreover, it is easy to check that using this proposal results in a MH acceptance probability of 1, so that every move is accepted. This guarantees that each kernel K_d admits the same stationary distribution π as a stationary distribution, so we have $\pi = K_d\pi$ for all d. Now, we see that $\pi = K_1\pi$ and $\pi = K_2\pi$ implies

$$\pi = K_1K_2\pi$$

Algorithm 3. Gibbs sampler

```
1: Initialize $x^{(1)} = (x_1, \ldots, x_D)^{(1)}$ arbitrarily
2: for $n = 2, 3, \ldots$ do
3:     $x_1^{(n)} \sim p(x_1 | x_2^{(n-1)}, x_3^{(n-1)}, \ldots, x_D^{(n-1)})$
4:     $x_2^{(n)} \sim p(x_2 | x_1^{(n)}, x_3^{(n-1)}, \ldots, x_D^{(n-1)})$
       ⋮
5:     $x_D^{(n)} \sim p(x_D | x_1^{(n)}, x_2^{(n)}, \ldots, x_{D-1}^{(n)})$
6: end for
```

and by induction

$$\pi = (K_1 K_2 \ldots K_D)\pi \equiv K\pi.$$

Here, the transition kernel K is the effective Gibbs kernel after we sweep over all the variables x_d once. Unless the full conditionals are degenerate, the effective kernel K will be aperiodic and irreducable while none of the individual kernels K_d are. Note that we could visit the variables in any deterministic or order, provided each x_d is visited infinitely often in the limit. If we choose a deterministic order, the resulting sampler is known as a *deterministic sca Gibbs sampler*; if we choose a random order, not surprisingly, the sampler is called a *random scan Gibbs sampler*.

The ease of use of the Gibbs sampler has resulted the method being employed in many applications. There are very effective programs that generate a Gibbs sampler automatically from a model specification, such as BUGS. In the sequel, we will illustrate the Gibbs sampler on a change-point model for count data [25].

Example 4 (Gibbs sampling for a change-point model). In this model, at each time t we observe the count of an event y_t. All the counts up to an unknown time τ come from the same distribution after which the distribution changes. We assume that the change-point τ is uniformly distributed over $1, \ldots, T$. The two different distributional regimes up to and after τ are indicated by the random variables λ_i, $i = 1, 2$, which are assumed to follow a Gamma distribution.

$$\mathcal{G}(\lambda_i; a, b) = \frac{1}{\Gamma(a)} b^a \lambda_i^{a-1} e^{-b\lambda_i} = e^{(a-1)\log\lambda - b\lambda - \log\Gamma(a) + a\log b}.$$

Under regime λ_i, the counts are assumed to be identically Poisson distributed

$$\mathcal{PO}(y_t; \lambda_i) = \frac{\lambda_i^{y_t}}{y_t!} e^{-\lambda_i} = e^{y_t \log\lambda_i - \lambda_i - \log(y_t!)}.$$

This leads to the following generative model:

$$\begin{aligned} p(\tau) &= 1/T \quad \text{uniform}, \\ \lambda_i &\sim \mathcal{G}(\lambda_i; a, b), \quad i = 1, 2, \\ y_t &\sim \begin{cases} \mathcal{PO}(y_t; \lambda_1) & 1 \le t \le \tau, \\ \mathcal{PO}(y_t; \lambda_2) & \tau < t \le T. \end{cases} \end{aligned}$$

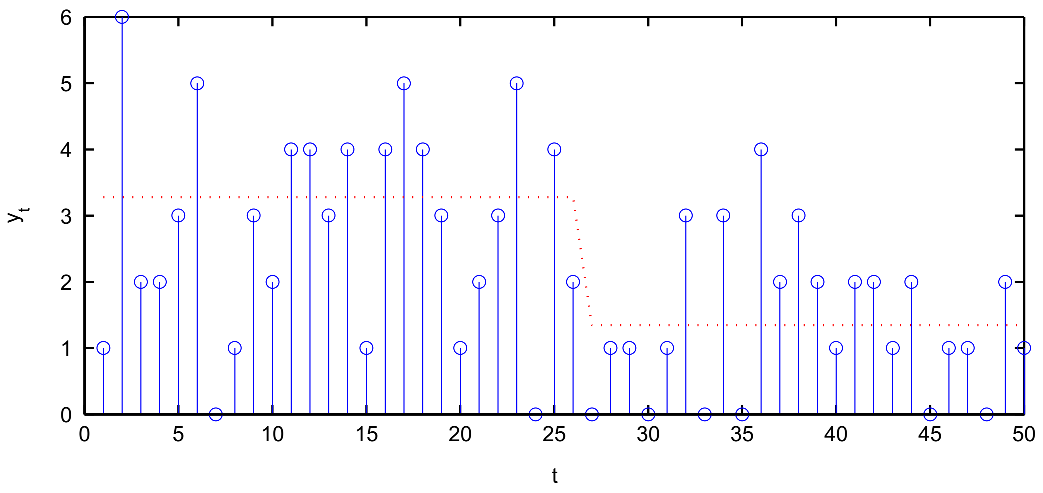

FIGURE 19.17

A typical realisation from the change-point model. True intensities are shown with a dotted line. The intensity has dropped from $\lambda_1 = 3.2$ to $\lambda_2 = 1.2$ at $\tau = 26$. The time index is indicated by t and the number of counts by y_t. Generating such synthetic datasets provide intuition about the underlying models qualitative behaviour.

A typical draw from this model is shown in Figure 19.17. The inferential goal is to compute the posterior distribution of the change-point location and the intensities given the count data, $p(\lambda_1, \lambda_2, \tau | y_{1:T})$. In this problem the posterior is actually tractable [26] and could serve to assess the quality of the Gibbs sampling approximation.

To implement Gibbs sampling we need to compute the distribution of each variable, conditioned on the rest. These conditionals can be derived by writing the log of the full joint distribution and collecting terms that depend only on the free variable.[6] To derive the full conditional densities $p(\lambda_1 | \cdot)$, $p(\lambda_2 | \cdot)$ and $p(\tau | \cdot)$, we proceed by writing the full joint density. The log of the full joint density is given by:[7]

$$p(y_{1:T}, \lambda_1, \lambda_2, \tau) = \left(\prod_{t=1}^{\tau} p(y_t|\lambda_1)\right)\left(\prod_{t=\tau+1}^{T} p(y_t|\lambda_2)\right) p(\lambda_1)p(\lambda_2)p(\tau),$$

$$\begin{aligned}\log p(y_{1:T}, \lambda_1, \lambda_2, \tau) &= \sum_{t=1}^{\tau}(+\, y_t \log \lambda_1 - \lambda_1 - \log(y_t!)) \\ &+ \sum_{t=\tau+1}^{T}(+\, y_t \log \lambda_2 - \lambda_2 - \log(y_t!))\end{aligned}$$

[6] since $p(X_d | x_{-d} = x_{-d}) = p(X_d, x_{-d} = x_{-d}) / p(x_{-d} = x_{-d}) \propto p(X_d, x_{-d} = x_{-d})$.
[7] Here, $f(x) =^{+} g(x)$ means $f(x) = g(x) + C$ where C is an irrelevant constant. This implies $\exp(f(x)) \propto \exp(g(x))$.

$$\begin{aligned}
&+(a-1)\log\lambda_1 - b\lambda_1 - \log\Gamma(a) + a\log b \\
&+(a-1)\log\lambda_2 - b\lambda_2 - \log\Gamma(a) + a\log b - \log T,
\end{aligned}$$

and collect terms that depend only on the free variable.

$$\log p(\lambda_1|\tau,\lambda_2,y_{1:T}) =^+ \left(a + \sum_{t=1}^{\tau} y_t - 1\right)\log\lambda_1 - (\tau+b)\lambda_1 =^+ \log\mathcal{G}(a + \sum_{t=1}^{\tau} y_t, \tau + b).$$

Calculation of $p(\lambda_2|\cdot)$ is similar.

$$\begin{aligned}
\log p(\tau|\lambda_1,\lambda_2,y_{1:T}) &= \sum_{t=1}^{\tau}\big(+y_t\log\lambda_1 - \lambda_1 - \log(y_t!)\big) \\
&\quad + \sum_{t=\tau+1}^{M}\big(+y_t\log\lambda_2 - \lambda_2 - \log(y_t!)\big) \\
&=+ \left(\sum_{t=1}^{\tau} y_t\right)\log\lambda_1 - \tau\lambda_1 + \left(\sum_{t=\tau+1}^{T} y_t\right)\log\lambda_2 - (T-\tau)\lambda_2.
\end{aligned}$$

We have to evaluate the above expression for $\tau = 1, \ldots, T$. This Gibbs procedure is summarized in Algorithm 4.

Algorithm 4. A Gibbs sampler for the change-point model

```
1: Initialize $\lambda_2^1, \tau^1$
2: for $n = 2, 3, \ldots$ do
3:     $\lambda_1^{(n)} \sim p(\lambda_1|\lambda_2^{(n-1)}, \tau^{(n-1)}, y_{1:T}) = \mathcal{G}(a + \sum_{t=1}^{\tau} y_t, \tau + b)$
4:     $\lambda_2^{(n)} \sim p(\lambda_2|\lambda_1^{(n)}, \tau^{(n-1)}, y_{1:T}) = \mathcal{G}(a + \sum_{t=\tau+1}^{T} y_t, T - \tau + b)$
5:     $\tau^{(n)} \sim p(\tau|\lambda_1^{(n)}, \lambda_2^{(n)}, y_{1:T})$
6: end for
```

We illustrate the algorithm on a coal mining disaster dataset [27], see Figure 19.18. The data consists of the number of deadly coal-mining disasters in England per year over a time span of 112 years from 1851 to 1962. It is widely agreed in the statistical literature that a change in the intensity (the expected value of the number of disasters) occurs around the year 1890, after new health and safety regulations were introduced. In Figure 19.18 we show the samples obtained from the posterior of the changepoint location. The posterior of the intensities λ_1, λ_2 are shown in Figure 19.19.

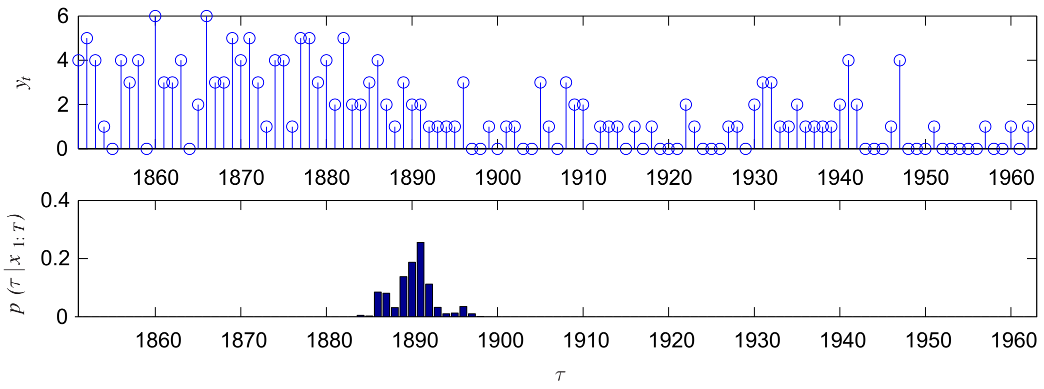

FIGURE 19.18

Estimation of change points on the coal mining disaster dataset. (Top) The number of deadly disasters y_t each year t. (Bottom) The posterior distribution of the changepoint location $p(\tau|y_{1:T})$.

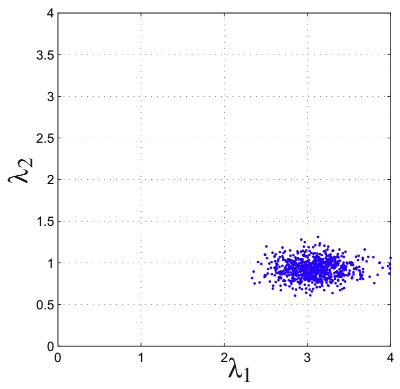

FIGURE 19.19

Samples drawn from the posterior intensities $p(\lambda_1, \lambda_2|y_{1:T})$ on the coal mining disaster dataset. This results clearly indicate that after the health and safety regulations, the intensity has dropped from about $\lambda_1 = 3$ to $\lambda_2 = 1$.

1.19.5 Sequential Monte Carlo

The MCMC techniques described in the previous section are widely used in many areas of applied sciences and engineering. However, they may have some potential drawbacks in signal processing: MCMC methods are by construction batch algorithms that require availability of all data records. This can be prohibitive in some signal processing application such as tracking when the data arrives sequentially and decisions have to be taken in real time. One could in principle design online MCMC algorithms that operate on the full data record observed so far. However, such batch algorithms take increasingly more time. In such cases, it is desirable to have alternative methods which process data sequentially and take a constant time per observation.

Sequential Monte Carlo (SMC) is an umbrella term for Monte Carlo techniques that process data in an online fashion [9]. Popular techniques such as particle filtering or sequential importance sampling (SIS) are special SMC algorithms. SMC techniques are particularly natural for probabilistic signal models where data has a natural order.[8] A very large class of signal models can be described as *state space models*, that have the following general form:

$$x_0 \sim p(x_0),$$
$$x_t|x_{t-1} \sim p(x_t|x_{t-1}), \tag{19.8}$$
$$y_t|x_t \sim p(y_t|x_t). \tag{19.9}$$

Here, x_t is the state and y_t are the observations and $t = 0, \dots, T$ is a discrete time index. Note that t does not have to correspond to the wall clock, it only specifies some natural ordering of the underlying the data. The Eq. (19.8) and Eq. (19.9) are known as *transition* and *observation* models. State space models, also named as hidden Markov models, are ubiquitous in signal processing and many popular models, such as linear dynamical systems or autoregressive (AR) models, can be formulated as special cases.

There are several inference problems of interest for state space models, where inference in this context is the task of using a distribution to answer questions of interest. A common inference problem is the *prediction* of an unseen future variable y_{T+1}, which requires computing $p(_{T+1}|y_{1:T})$ from a joint distribution. Another one is the so called *filtering* problem, estimation of the state x_t given observations so far $y_{1:t}$. This is achieved via calculation of the posterior quantity $p(x_t|y_{1:t})$. One of the challenges in statistical signal processing is to develop efficient inference routines for answering such questions.

In this context, SMC techniques [10,28] have proved very useful. SMC methods are based on importance sampling/resampling which we review in the following sections. We first review importance sampling and resampling techniques and subsequently introduce the SMC algorithm as a sequential importance sampler.

1.19.5.1 Importance sampling (IS)

Consider a target distribution $\pi(x) = \phi(x)/Z$ where the non-negative function $\phi(x)$ is known, but the overall normalisation constant Z is assumed to be computationally intractable. The main idea of (IS) is to estimate expectations $\mathsf{E}_\pi(\varphi(x))$ by using weighted samples from a tractable IS distribution $q(x)$. Note that unlike other MC methods, our goal is not directly generating samples but estimating expectations via weighted samples. However, we can also generate unweighted independent samples, if we desire to so by a mechanism called *resampling*.

More specifically, we can write the normalization constant as

$$Z = \int \phi(x)dx = \int \frac{\phi(x)}{q(x)}q(x)dx = \int W(x)q(x)dx = E_q(W(x)),$$

[8]SMC techniques are not necessarily limited to such systems, though, see, e.g., [20].

where $W(x) \equiv \phi(x)/q(x)$ is called *weight function*. Therefore we have

$$\mathsf{E}_\pi(\varphi(x)) = \frac{1}{Z}\int \varphi(x)\frac{\phi(x)}{q(x)}q(x)dx = \frac{\mathsf{E}_q(\varphi(x)W(x))}{\mathsf{E}_q(W(x))}.$$

A Monte Carlo estimate of $\mathsf{E}_\pi\varphi(x)$ is then given by

$$\hat{\mu}_N = \frac{\sum_{i=1}^N W^{(i)}\varphi(x^{(i)})/N}{\sum_{i=1}^N W^{(i)}/N},$$

where $W^{(i)} \equiv W(x^{(i)})$ and the "particles" $x^{(1)}, \ldots, x^{(N)}$ are sampled from $q(x)$. Using normalised weights $w^{(i)} \equiv W^{(i)}/\sum_{i'=1}^N W^{(i')}$ we can write the approximation as

$$\hat{\mu}_N = \sum_{i=1}^N w^{(i)}\varphi(x^{(i)}).$$

In other words, instead of sampling from the target density $p(x)$, we sample from a different tractable distribution $q(x)$ and correct by reweighing the samples accordingly. An example is given in Figure 19.20.

1.19.5.1.1 Resampling

A practical issue is that, unless the IS sampling density $q(x)$ is close to the target density $\pi(x)$, the normalised weights will typically have mass in only a single component. This can be partially addressed using re-sampling as we now describe. Given a weighted particle system $\sum_{i=1}^N w^{(i)}\delta(x - x^{(i)})$, resampling is the term for a set of methods for generating *randomly* an unweighted particle system of the form $\frac{1}{M}\sum_{j=1}^M \delta(x - \tilde{x}^{(i)})$ such that

$$\mathsf{E}\left(\frac{1}{M}\sum_{j=1}^M \delta(x - \tilde{x}^{(j)})\right) = \sum_{i=1}^N w^{(i)}\delta(x - x^{(i)}),$$

where the expectation is over the draws from the resampling algorithm. In resampling, typically the total number of particles are unchanged, so we have $N = M$. Alternatively, one can view resampling as a randomised pruning algorithm where we discard particles with low weight and increase the number of particles with high weight. Unlike a deterministic pruning algorithm, the random nature of resampling does not introduce a systematic bias and ensures an asymptotically consistent algorithm when the number of samples N goes to infinity. In Figure 19.21, we illustrate the result of resampling from a weighted particle set. It is instructive to compare histograms obtained from weighted particle set (Figure 19.20); the resampling stage introduces additional Monte Carlo variation but the histogram is still a reasonable approximation to the underlying density. Two popular methods for resampling are multinomial resampling and systematic resampling, which we will review in the sequel. For a discussion and comparison of various resampling schemata, see [10,29].

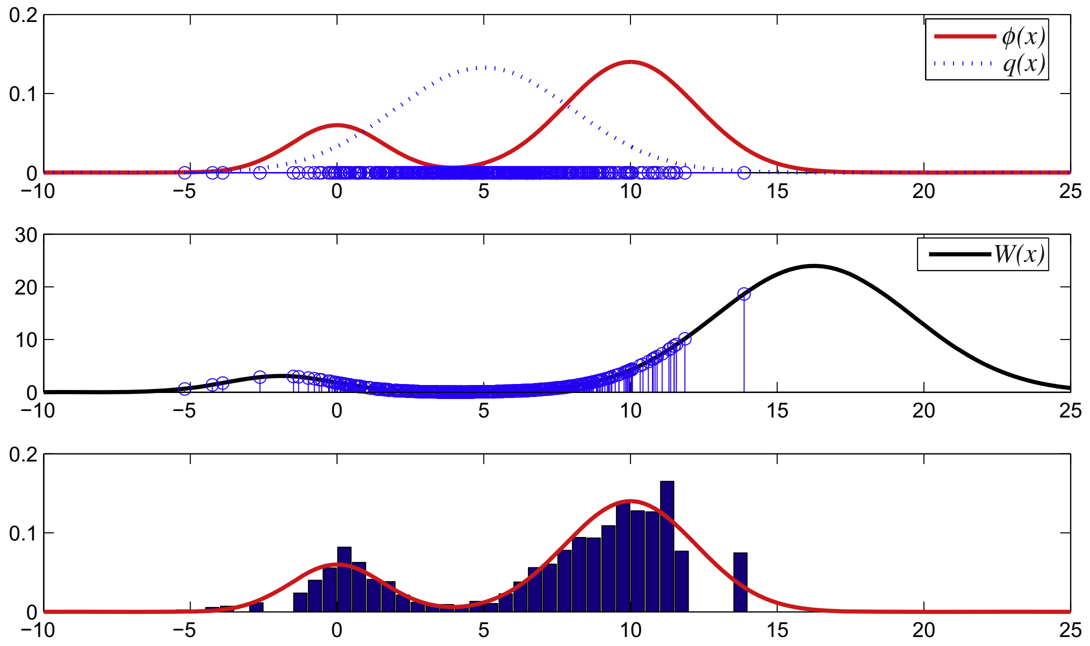

FIGURE 19.20

Illustration of Importance Sampling. (Top): The solid curve denotes the unnormalised target density $\phi(x)$ and the dashed curve the tractable IS density $q(x)$. We assume that it is straightforward to generate samples $x^{(i)}$ from $q(x)$; these are plotted on the x axis. Middle: To account for the fact that the samples are from q and not from the target $\pi = \phi/Z$, we need to reweight the samples. In this example, the IS density q generated far too many samples where π has low mass, and too few where π has high mass. The samples in these regions are reweighted according to the weight function $W(x)$. Bottom: Binning the weighted samples from q, we obtain an approximation to π such that averages with respect to this approximation will be close to averages with respect to π.

Multinomial resampling equivalent to the inversion method when the target is a discrete distribution, as explained in Figure 19.3. Here, we view the weighted sample set $x^{(i)}$ as a discrete distribution with categories $i = 1, \ldots, N$ where the probability of the ith category is given by the normalized weight $w^{(i)}$. To sample M times independently from this target, we generate $u^{(j)} \sim \mathcal{U}(0,1)$ for $j = 1, \ldots, M$ and we obtain an unweighted set of particles by evaluating the generalized inverse at each $u^{(j)}$, as explained in Figure 19.3.

Systematic resampling is quite similar to multinomial resampling, only the generation of $u^{(j)}$ is not entirely random but "systematic." To generate M samples we only select $u^{(1)}$ uniformly random from the interval $[0, 1/M]$ and set $u^{(j)} = u^{(1)} + (j-1)/M$. The u are located on a uniform grid with a random initial shift. In Figure 19.22, we illustrate both methods.

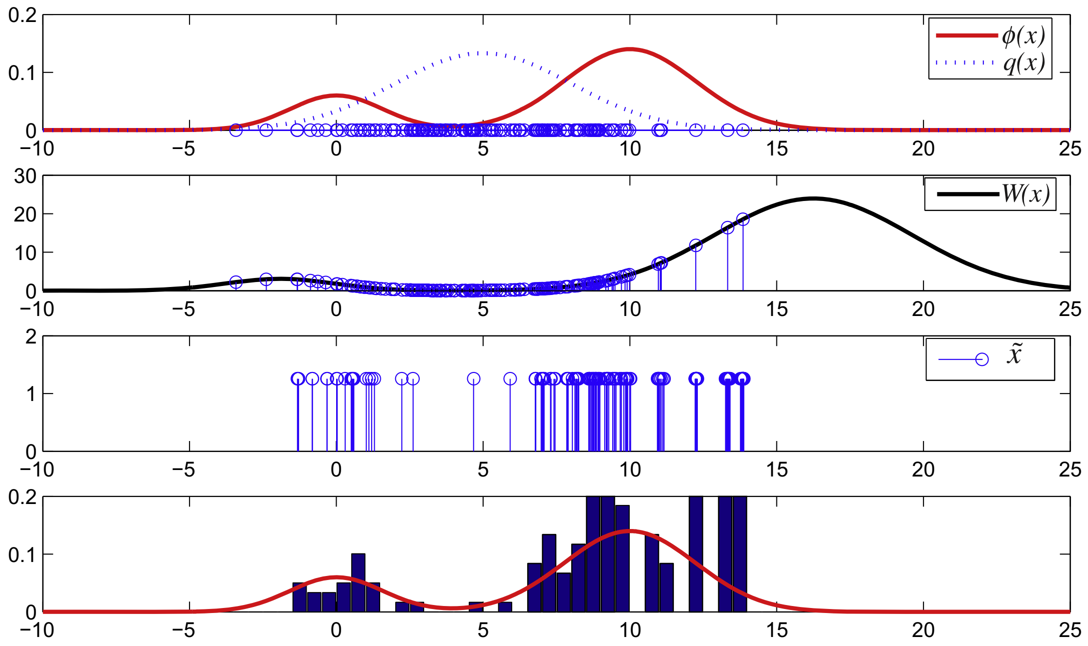

FIGURE 19.21

Importance Sampling with Resampling. We can generate from the weighted samples x generated by importance sampling unweighted samples $\tilde{x}$ by resampling. Some of the particles in the original set will be replicated while others will be simply discarded.

From only a theoretical perspective, the actual details of a resampling schema are not important and the estimates converge to their true values in the infinite particle limit. One has to only ensure that the expectation under the resampled particle set is *on average* equal to the expectation under the original weighted particle set. However, with a finite number of particles, there may be notable differences.

It can be shown that systematic resampling achieves a lower variance while both methods have the same expectation and no bias. Coupled with its ease of implementation, this makes systematic resampling a popular choice in practice.

1.19.5.2 Sequential importance sampling

We now apply importance sampling to the state space model introduced earlier in Eq. (19.8) and Eq. (19.9). "Plain" importance sampling is actually not very effective in high dimensional problems but when executed sequentially using resampling techniques, it has become a very efficient and effective tool for inference. The resulting sequential IS methods are also known as particle filters [10]. The goal

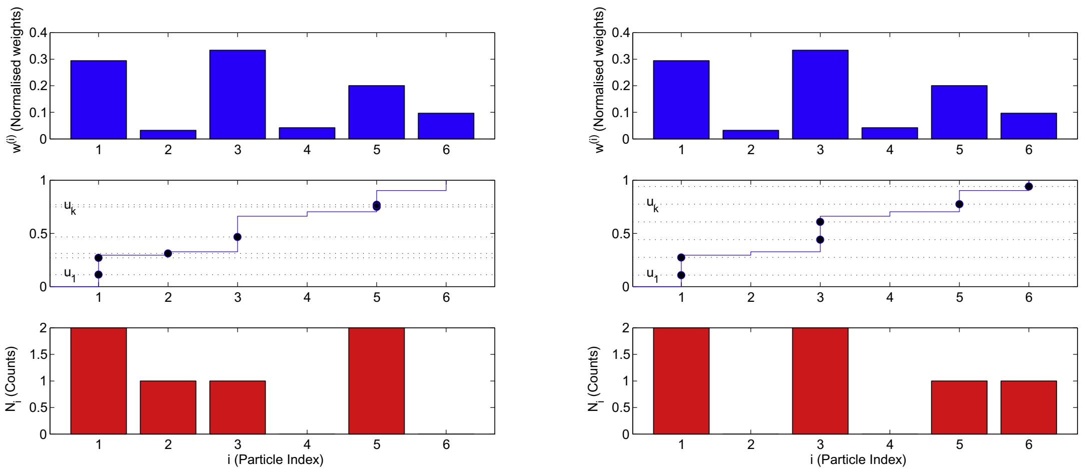

FIGURE 19.22

Multinomial Resampling (left column) versus Systematic Resampling (right column). (Top) True weights, (Middle) Cummulative weight function, (Bottom) Bin counts after resampling. Note that the resulting histograms can be thought of as approximations of the original discrete distribution.

is to draw samples from the posterior

$$p(x_{1:t}|y_{1:t}) = \underbrace{p(y_{1:t}|x_{1:t})p(x_{1:t})}_{\phi(x_{1:t})} \,/\, \underbrace{p(y_{1:t})}_{Z_t},$$

where we assume that the normalisation term Z_t is intractable. An importance sampling approach uses at each time t an importance distribution $q_t(x_{1:t})$, from which we draw samples $x_{1:t}^{(i)}$ with corresponding importance weights

$$W_t^{(i)} = \phi(x_{1:t}^{(i)})/q_t(x_{1:t}^{(i)}).$$

The key idea in SMC is the sequential construction of the IS distribution q and the recursive calculation of the importance weights. Without loss of generality, we may write

$$q_t(x_{1:t}) = q_t(x_t|x_{1:t-1})q_t(x_{1:t-1}).$$

In particle filtering, one chooses a IS proposal q that only updates the current x_t and leaves previous samples unaffected. This is achieved using

$$q_t(x_{1:t}) = q_t(x_t|x_{1:t-1})q_{t-1}(x_{1:t-1}).$$

As we are free to chose the IS proposal q fairly arbitrarily, we can also "construct" the proposal on the fly conditioned on the observations seen so far:

$$q_t(x_{1:t}|y_{1:t}) = q_t(x_t|x_{1:t-1}, y_{1:t})q_{t-1}(x_{1:t-1}|y_{1:t-1}).$$

In the sequel, we won't include $y_{1:t}$ in our notation but it should be understood that the proposal q can be constructed at the observations so far.

Due to the sequential nature of the state space model and q, the weight function $W_t(x_{1:t})$ admits a *recursive* computation

$$W_t(x_{1:t}) = \frac{\phi(x_{1:t})}{q_t(x_{1:t})} = \frac{p(y_t|x_t)p(x_t|x_{t-1})\prod_{\tau=1}^{t-1} p(y_\tau|x_\tau)p(x_\tau|x_{\tau-1})}{q_t(x_t|x_{1:t-1})\prod_{\tau=1}^{t-1} q_\tau(x_\tau|x_{1:\tau-1})}$$
$$= \underbrace{\frac{p(y_t|x_t)p(x_t|x_{t-1})}{q_t(x_t|x_{1:t-1})}}_{v_t} W_{t-1}(x_{1:t-1}),$$

where v_t is called the incremental weight. Particle filtering algorithms differ in their choices for $q_t(x_t|x_{1:t-1})$. The optimal choice (in terms of reducing the variance of weights) is the one step filtering distribution [30]

$$q_t(x_t|x_{1:t-1}) = p(x_t|x_{t-1}, y_t).$$

However, sampling from this filtering density is often difficult in practice, and simpler methods are required. The popular *bootstrap* filter uses the model transition density as the proposal

$$q_t(x_t|x_{1:t-1}) = p(x_t|x_{t-1}),$$

for which the incremental weight is $v_t = p(y_t|x_t)$. For the bootstrap filter, the IS distribution does not make any use of the recent observation and therefore has the tendency to lose track of the high-mass regions of the posterior. Indeed, it can be shown that the variance of the importance weights for the bootstrap filter increases in an unbounded fashion [20,30] so that the state estimates are not reliable. In practice, therefore, after a few time steps the particle set typically loses track of the exact posterior mode. A crucial extra step to make the algorithm work is resampling which prunes branches with low weights and keeps the particle set located in high probability regions. It can be shown that although the particles become dependent due to resampling, the estimations are still consistent and converge to the true values as the number of particles increases to infinity. A generic particle filter algorithm is shown in Algorithm 5.

Example 5 (Particle Filtering). We will consider the following model of a sequence of Poisson random variables y_t with intensities $\exp(l_t)$:

$$y_t|l_t \sim p(y_t|l_t) = \mathcal{PO}(y_t; \exp(l_t)) = \exp(y_t l_t - \exp(l_t) - \log(y_t!)). \tag{19.10}$$

The latent log intensities l_t are assumed to be changing slowly with t; this can be modeled by a random drift

$$l_t|l_{t-1} \sim p(l_t|l_{t-1}) = \mathcal{N}(l_t; l_{t-1}, R) = \exp\left(-\frac{1}{2}\frac{(l_t - l_{t-1})^2}{R} - \frac{1}{2}\log 2\pi R\right). \tag{19.11}$$

This model can be considered as an alternative to the single change point model introduced in Section 4. Rather than allowing for a single abrupt change, this model proposes slowly varying intensities where the drift is controlled by the transition variance parameter R. If R is small, consecutive log intensities l_t

Algorithm 5. Particle Filter

```
for $i = 1, \ldots, N$ do
    Compute the IS distribution: $q_t(x_t|x_{1:t-1}^{(i)})$ possibly using the latest observation $y_t$
    Generate offsprings: $\hat{x}_t^{(i)} \sim q_t(x_t|x_{1:t-1}^{(i)})$
    Evaluate importance weights:
        $v_t^{(i)} = \frac{p(y_t|\hat{x}_t^{(i)})p(\hat{x}_t^{(i)}|x_{t-1}^{(i)})}{q_t(\hat{x}_t^{(i)}|x_{1:t-1}^{(i)})}, \quad W_t^{(i)} = v_t^{(i)} W_{t-1}^{(i)}$
end for
$\text{ESS} = 1/\sum_i \left(\tilde{w}_{1:t}^{(i)}\right)^2$
if ESS > Threshold then
    No need to Resample
    Extend particles: $x_{1:t}^{(i)} = (x_{1:t-1}^{(i)}, \hat{x}_t^{(i)}),\ i = 1, \ldots, N$
else
    Resample
    Normalise importance weights:
        $\tilde{Z}_t \leftarrow \sum_j W_t^{(j)}, \quad \tilde{\mathbf{w}}_t \leftarrow (W_t^{(1)}, \ldots, W_t{}^{(N)})/\tilde{Z}_t$
    Generate Associations via a Resampling method
        $(a(1), \ldots, a(N)) \leftarrow \texttt{Resample}(\tilde{\mathbf{w}}_t)$
    Discard or Keep particles and Reset weights:
        $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{a(i)}, \hat{x}_t^{a(i)}),\ W_t^{(i)} \leftarrow \tilde{Z}_t/N, i = 1, \ldots, N$
end if
```

and l_{t+1} will be close to each other. In the limit of R going to zero, the model degenerates into a constant intensity model. In this example, we will assume that we know R.

To design a SMC algorithm, we need to decide upon a proposal mechanism and derive the expression for the incremental importance weight

$$v_t(l_t|l_{1:t-1}) = \frac{p(y_t|l_t)p(l_t|l_{t-1})}{q_t(l_t|l_{1:t-1})}.$$

For illustration purposes, we will consider here two different proposal mechanisms:

1. *Transition density as the proposal* (the Bootstrap filter):
   Here, we choose as our proposal the models transition density

   $$q_t(l_t|l_{1:t-1}) = p(l_t|l_{t-1}).$$

   This choice is the most widely used one for constructing proposals because of its simplicity, ease of implementation and the intuitive interpretation as a "the survival of the fittest" algorithm. The resulting algorithm is known in the SMC literature as the *bootstrap filter* and in computer vision as the *condensation algorithm*. For the bootstrap filter, the incremental weight expression becomes

$$\begin{aligned} v_t(l_t|l_{1:t-1}) &= \frac{p(y_t|l_t)p(l_t|l_{t-1})}{p(l_t|l_{t-1})} = p(y_t|l_t) \\ &= \exp(y_t l_t - \exp(l_t) - \log(y_t!)), \end{aligned}$$

2. *The observation likelihood as the proposal*
Here, we choose a density that is proportional to the likelihood term

$$q_t(l_t|l_{1:t-1}) = q_t(l_t) = p(y_t|l_t)/c_t,$$

where $c_t = \int p(y_t|l_t)dl_t$. Note that this proposal can only be used when the normalizer c_t is not infinite hence a density exists.[9] This option is perhaps less intuitive to understand but could be a lot more efficient than the bootstrap in problems where the transition model is very diffuse but the observations are very informative. Using a bootstrap filter may be inefficient in this case, as most of the proposed particles will fall into low probability regions.

$$v_t(l_t|l_{1:t-1}) = \frac{p(y_t|l_t)p(l_t|l_{t-1})}{p(y_t|l_t)/c_t} = c_t p(l_t|l_{t-1}).$$

In practice, unless we need to explicitly evaluate the marginal likelihood $p(y_{1:t})$, we do not need to evaluate normalizer c_t as it will cancel out when calculating the normalized weights.
To find this proposal density, we make the following observation: the Poisson density, when viewed as a function of the intensity is proportional to a Gamma density[10]

$$\begin{aligned} \mathcal{PO}(y;\lambda) &\propto \exp(y\log\lambda - \lambda) \\ &\propto \exp((y+1-1)\log\lambda - \lambda - \log\Gamma(y+1)) = \mathcal{G}(\lambda; y+1, 1). \end{aligned}$$

So for a given Poisson observation, we can easily generate λ by a unit gamma random number generator with shape parameter $y+1$. To generate the log intensities l, we sample λ and set $l = \log(\lambda)$. The density of $l = \log\lambda$ when λ is gamma distributed is found via the transformation method

$$\begin{aligned} l(\lambda) &= \log(\lambda), \\ \lambda(l) &= \exp(l), \\ |J| &= \partial\lambda(l)/\partial l = \exp(l), \\ p(l) &= \mathcal{G}(\lambda(l); a, b)\exp(l). \end{aligned}$$

So the density of our proposal is

$$q_t(l_t) \propto \exp((y_t+1)l_t - \exp(l_t) - \log\Gamma(y_t+1)).$$

[9]Here, c_t is not to be confused with the actual normalizing constant of the model which is the marginal likelihood $Z_t = p(y_t|y_{1:t-1})$.
[10]This is due to the so called conjugacy property.

The weight expression is

$$v_t(l_t|l_{1:t-1}) = \frac{\exp(y_t l_t - \exp(l_t) - \log\Gamma(y_t))\exp\left(-\frac{1}{2}\frac{(l_t-l_{t-1})^2}{R}\right)}{\exp((y_t+1)l_t - \exp(l_t) - \log\Gamma(y_t+1))}$$
$$\propto \exp\left(-\frac{1}{2}\frac{(l_t-l_{t-1})^2}{R} - l_t\right).$$

Here, we have simply ignored the terms that do not depend on l_t as these do not contribute to the normalized importance weight.

As mentioned earlier, in practice the transition model is taken often as the proposal density. It should be stressed that, the choice of an efficient proposal has a profound effect on the performance and the transition model need not to be always the best choice. As the above example illustrates, there are typically always alternative proposals and in the context of an application, at least a few alternatives should be considered before opting to the transition density. As a guideline, one should look for a proposal that is close to the filtering density but that does not require extensive computation to sample. If a proposal is too costly to sample from, one may consider using a simpler but poorer proposal with more samples for a given computational cost.

1.19.6 Advanced Monte Carlo methods

We have reviewed some of the basic Monte Carlo techniques in previous sections. However, there are many problems that need techniques beyond standard strategies. In this section, we will review such a technique: reversible jump MCMC (RJ-MCMC) [11,31]. This technique is an extension of the Metropolis-Hastings method to more general state spaces [23]. It is mostly used in settings where we wish to perform model selection, such as inferring a model where the order is unknown.

1.19.6.1 Reversible jump MCMC

The reversible jump MCMC is a Metropolis-Hastings method for sampling from Markov chains in state spaces of varying dimensionality. Such state spaces arise naturally in model selection problems where we have several candidate models $\{\mathcal{M}_k\}$ with model index k, formally denoted as $k \in \mathcal{K}$ for some finite set $\mathcal{K}$. With each model $\mathcal{M}_k$ there is an associated parameter vector θ_k of dimensionality N_k, i.e., $\theta_k \in \mathbb{R}^{N_k}$. The inferential goal is to sample from the joint density $\pi(k, \theta_k)$ of model index k and the parameters θ_k where

$$\pi(\theta_k, k) = \frac{1}{Z_k}\phi_k(\theta_k)\tilde{p}_k, \tag{19.12}$$

$$\tilde{p}_k \equiv \frac{p_k}{\sum_{K'\in\mathcal{K}} pk'}. \tag{19.13}$$

Here, for all models $k \in \mathcal{K}$, $Z_k = \int d\theta_k \phi_k(\theta_k)$ are unknown normalizing constants and p_k are prior weights, usually taken as flat with $p_k = 1$. These quantities have direct counterparts in a Bayesian model selection setting: we have a probability model that couples observations x with parameters as $p(x, \theta_k, k) = p(x, \theta_k | k) p(k)$. In this context, the target density is the posterior

$$\pi(\theta_k, k) = p(\theta_k, k | x) = p(\theta_k | k, x) p(k | x)$$

and the joint distribution of parameters and data for each model $\mathcal{M}_k$ is $\phi_k(\theta_k) = p(x, \theta_k | k)$ and $Z_k = \int d\theta_k \phi_k(\theta_k) = p(x|k)$. To find the model order after observing the data x we would need the posterior marginal

$$p(k|x) = \frac{p(k, x)}{\sum_{k' \in \mathcal{K}} p(k, x)} = \frac{\tilde{p}_k Z_k}{\sum_{\kappa \in \mathcal{K}} \tilde{p}_\kappa Z_\kappa} = \frac{p_k Z_k}{\sum_{\kappa \in \mathcal{K}} p_\kappa Z_\kappa},$$

where the equalities follow by noting that $p(k, x) = \int p(\theta_k, k, x) d\theta_k = \tilde{p}_k Z_k$ and $p(x) = \sum_k \tilde{p}_k Z_k$. With these results, the posterior can be written as

$$\pi(\theta_k, k) = \frac{1}{Z_k} \phi_k(\theta_k) \frac{p_k Z_k}{\sum_{\kappa \in \mathcal{K}} p_\kappa Z_\kappa} = \phi_k(\theta_k) \frac{p_k}{\sum_{\kappa \in \mathcal{K}} p_\kappa Z_\kappa}.$$

One possible approach for calculating the posterior of models order $p(k|x)$ would require estimation of individual normalizing constants Z_k. The RJ-MCMC approach circumvents this need by setting up a chain to sample from (k, θ_k) pairs. The difficulty is that now the target density π is defined on a nonstandard state space $\mathcal{E}$ where

$$\mathcal{E} = \bigcup\nolimits_{k \in \mathcal{K}} \mathcal{E}_k,$$
$$\mathcal{E}_k \equiv \{\{k\} \times \mathbb{R}^{N_k}\},$$

and we need to set up a so called *transdimensional* Markov chain that can jump across spaces of different dimensions $\mathcal{E}_k$ and $\mathcal{E}_{k'}$ as $(k, \theta_k) \to (k', \theta_{k'})$. This chain is "nonstandard" in the sense that its dimension varies over time but is otherwise completely natural for the class of model selection problems.

There is a technical caveat: The "ordinary" Metropolis-Hastings algorithm requires that we design a proposal

$$q(k', \theta_{k'} | k, \theta_k) = q(\theta_{k'} | k', k, \theta_k) q(k' | k, \theta_k)$$

and evaluate the acceptance probability as

$$\alpha((k, \theta_k) \to (k', \theta_{k'})) = \min \left\{ 1, \frac{q(k, \theta_k | k', \theta_{k'}) \pi(k', \theta_{k'})}{q(k', \theta_{k'} | k, \theta_k) \pi(k, \theta_k)} \right\}.$$

We need to construct the proposals carefully, as as a proper lower dimensional space embedded into a higher dimensional space may have measure zero with respect to the probability measure defined on the larger space.[11] The remedy comes from the transformation method introduced earlier. By introducing auxiliary variables, we design the proposal as a "reversible" transformation between both spaces.

[11]The probability of jumping to an isolated point (a proper low dimensional space) under a proposal with an absolutely continuous density is zero. As an example, think of the probability of hitting an interval $(-\epsilon, \epsilon)$ when $\epsilon \to 0$.

Suppose we wish to jump from a lower dimensional space $\mathcal{E}_k$ to a higher dimensional space $\mathcal{E}_{k'}$. We define $\bar{N}_k$ such that $N_{k'} = N_k + \bar{N}_k$ and draw the $\bar{N}_k$ dimensional vector u_k from a proposal $q_k(u_k)$ and match the dimensions of both spaces via a transformation defined as

$$\theta_{k'} = g_{k\to k'}(\theta_k, u_k).$$

The mapping $g_{k\to k'}$ is fairly general but must be invertible such that $g^{-1}_{k\to k'} = g_{k'\to k}$ exists. As an alternative notation for g or its inverse g^{-1}, we omit the letter g and write $\theta_{k'}(\theta_k, u_k)$and $(\theta_k(\theta_{k'}), u_k(\theta_{k'}))$ respectively. More naturally, it is also possible to match dimensions on a higher dimensional space where $N_{k'} + \bar{N}_{k'} = N_k + \bar{N}_k$ where $(\theta_{k'}, u_{k'}) = g(\theta_k, u_k)$ but we won't discuss this option further here.

In order to design a MH algorithm, we will construct our proposal in two stages as in

$$q(k', \theta_{k'}|k, \theta_k) = q(k'|k, \theta_k)q(\theta_{k'}|k', k, \theta_k).$$

The first factor in this proposal is the probability of choosing a jump from k to k' when the chain is in θ_k. Often, this term is taken to be independent of θ_k, so $q(k'|k) = q(k'|k, \theta_k)$. We will denote the jump probabilities $q(k'|k)$ as $q_{k\to k'}$. The second factor, which we will denote as $f_{k'}(\theta_{k'}) \equiv q(\theta_{k'}|k', k, \theta_k)$, will be defined via the variable transformation $g_{k\to k'}$. Noting that $\theta_{k'} = g_{k\to k'}(\theta_k, u_k)$, for some density $f_k(\theta_k)$, the density of $\theta_{k'}$ can be written as is

$$\begin{aligned} f_{k'}(\theta_{k'}) &= f_k(\theta_k(\theta_{k'}))q_k(u_k(\theta_{k'})) \left| \frac{\partial g^{-1}_{k\to k'}(\theta_{k'})}{\partial \theta_{k'}} \right| \\ &= f_k(\theta_k(\theta_{k'}))q_k(u_k(\theta_{k'})) \left| \frac{\partial g_{k\to k'}(\theta_k, u_k)}{\partial(\theta_k, u_k)} \right|^{-1}. \end{aligned}$$

To simplify the notation, in the sequel we will denote the Jacobian terms as

$$J_{k\to k'} \equiv \frac{\partial g_{k\to k'}(\theta_k, u_k)}{\partial(\theta_k, u_k)} \qquad J_{k'\to k} \equiv \frac{\partial g^{-1}_{k\to k'}(\theta_{k'})}{\partial \theta_{k'}} = J^{-1}_{k\to k'}.$$

Now, interpret $f_k(\theta_k)$ as a reverse jump proposal $q(\theta_k|k, k', \theta_{k'})$. The ratio of the proposals to compute the MH-acceptance probability can be written for this case as

$$\begin{aligned} \frac{q(k, \theta_k|k', \theta_{k'})}{q(k', \theta_{k'}|k, \theta_k)} = \frac{f_k(\theta_k)q_{k'\to k}}{f_{k'}(\theta_{k'})q_{k\to k'}} &= \frac{f_k(\theta_k)q_{k'\to k}}{f_k(\theta_k(\theta_{k'}))q_k(u_k(\theta_{k'}))q_{k\to k'}} \left| \frac{\partial g_{k\to k'}(\theta_k, u_k)}{\partial(\theta_k, u_k)} \right| \\ &= \frac{q_{k'\to k}}{q_k(u_k(\theta_{k'}))q_{k\to k'}} \left| J_{k\to k'} \right|. \end{aligned}$$

The ratio of the targets is

$$\frac{\pi(k', \theta_{k'})}{\pi(k, \theta_k)} = \frac{\phi_{k'}(\theta_{k'}) \frac{p_{k'}}{\sum_{\kappa\in\mathcal{K}} p_\kappa Z_\kappa}}{\phi_k(\theta_k) \frac{p_k}{\sum_{\kappa\in\mathcal{K}} p_\kappa Z_\kappa}} = \frac{\phi_{k'}(\theta_{k'}) p_{k'}}{\phi_k(\theta_k) p_k}.$$

Consequently, we arrive at the rather complicated looking formula for the RJ-MCMC acceptance probability

$$\alpha((k,\theta_k) \to (k',\theta_{k'})) = \min\left\{1, \frac{\phi_{k'}(\theta_{k'})p_{k'}}{\phi_k(\theta_k)p_k}\frac{q_{k'\to k}}{q_k(u_k)q_{k\to k'}}|J_{k\to k'}|\right\} \tag{19.14}$$

$$\equiv \min\{1, r_{k\to k'}\} \tag{19.15}$$

$$\equiv \alpha_{k\to k'}. \tag{19.16}$$

The $r_{k\to k'}$ argument min function is known as the MH-ratio. The acceptance probability of the jump in the reverse direction is

$$\alpha_{k'\to k} = \min\left\{1, \frac{\phi_k(\theta_k)p_k}{\phi_{k'}(\theta_{k'})p_{k'}}\frac{q_k(u_k)q_{k\to k'}}{q_{k'\to k}}|J_{k'\to k}|\right\}$$
$$= \min\{1, r_{k'\to k}\} = \min\left\{1, r_{k\to k'}^{-1}\right\}$$

Note that for a pair of reversible jumps $(k \to k')$ and $(k' \to k)$, the corresponding MH-ratios are simply the inverses of each other $r_{k'\to k} = r_{k\to k'}^{-1}$.

1.19.6.2 RJ-MCMC algorithm

The RJ-MCMC algorithm is actually a special Metropolis Hastings method with a special proposal mechanism for transdimensional moves. To construct the algorithm, for each pair $k, k' \in \mathcal{K}$ we need to decide on the jump probabilities $q_{k\to k'}$ and $q_{k'\to k}$. Note that these are just parameters of the sampling algorithm and not the model, but a careful choice is important for good mixing. In practice, often only jumps between "adjacent" k and k' are used as it is relatively simpler to design proposals with reasonable acceptance probability. Here, a popular choice is only allowing moves with $k' = k+1$, $k' = k-1$, $k' = k$, where these are called *birth, death* and *update* moves.

If a pair of moves have nonzero probability, $q_{k\to k'}q_{k'\to k} > 0$, we have to have to design the following:

- For jumps to a larger space $N_{k'} > N_k$ we design a suitable invertible transformation $g_{k\to k'}$ and a proposals for $q_k(u_k)$ to sample a $\bar{N}_k$ dimensional random vector to match the dimensions such that $N_{k'} = N_k + \bar{N}_k$. Note that our choice completely fixes the moves in the reverse direction as well.
- Calculate the Jacobian determinants $|J_{k\to k'}| = |J_{k'\to k}|^{-1}$.
- Calculate the acceptance probabilities. Remember that if

$$\alpha_{k'\to k} = \min\{1, r_{k'\to k}\} \Leftrightarrow \alpha_{k\to k'} = \min\{1, r_{k\to k'}\} = \min\left\{1, \frac{1}{r_{k'\to k}}\right\}$$

A summary of the RJ-MCMC is shown as Algorithm 6

Example 6 (Reversible Jump - MCMC). As a illustrative toy example, we define a target density on the union of two spaces $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$ where

$$\mathcal{E}_1 = \{\{1\} \times r\} \qquad \mathcal{E}_2 = \{\{2\} \times (x, y)\}$$

Algorithm 6. Reversible Jump Markov Chain Monte Carlo

```
1:Initialise $\left(k^{(1)}, \theta_{(1)}^{(1)}\right)$ arbitrarily
2: **for** $\tau = 2, 3, \ldots$ **do**
3:   Let $(k, \theta_k) = \left(k^{(\tau-1)}, \theta_{k^{(\tau-1)}}^{(\tau-1)}\right)$
4:   Propose a move type: $k^{\text{new}} \sim q_{k \to k'}$
5:   **if** $k = k^{\text{new}}$, a ordinary move in the same space **then**
6:       $\theta^{\text{new}} \sim q_k\left(\theta_k | \theta_k^{(\tau-1)}\right)$
7:   **end if**
8:   **if** $k < k^{\text{new}}$, a transdimensional move to a larger space **then**
9:     Propose $u \sim q_k(u_k)$
10:       $\theta^{\text{new}} \leftarrow g_{k \to k^{\text{new}}}(\theta_k, u)$
11:     Compute the acceptance probability
     $\alpha\left((k, \theta_k) \to (k^{\text{new}}, \theta^{\text{new}})\right) = \min\left\{1, \frac{\phi_{k^{\text{new}}}(\theta_{k^{\text{new}}}) p_{k^{\text{new}}}}{\phi_k(\theta_k) p_k} \frac{q_{k^{\text{new}} \to k}}{q_k(u) q_{k \to k^{\text{new}}}} |J_{k \to k^{\text{new}}}|\right\}$
12:   **end if**
13:   **if** $k > k^{\text{new}}$, a transdimensional move to a smaller space **then**
14:       $(\theta^{\text{new}}, u) \leftarrow g_{k \to k^{\text{new}}}(\theta_k)$
15:       Compute the acceptance probability
     $\alpha\left((k, \theta_k) \to (k^{\text{new}}, \theta^{\text{new}})\right) = \min\left\{1, \frac{\phi_{k^{\text{new}}}(\theta_{k^{\text{new}}}) p_{k^{\text{new}}}}{\phi_k(\theta_k) p_k} \frac{q_{k^{\text{new}}}(u) q_{k^{\text{new}} \to k}}{q_{k \to k^{\text{new}}}} |J_{k \to k^{\text{new}}}|\right\}$
16:   **end if**
17:   Sample from uniform distribution: $a \sim \mathcal{U}(0, 1)$
18:   **if** $a < \alpha\left((k, \theta_k) \to (k^{\text{new}}, \theta^{\text{new}})\right)$ **then**
19:     Accept candidate: $\left(k^{(\tau)}, \theta_{(\tau)}^{(\tau)}\right) \leftarrow (k^{\text{new}}, \theta^{\text{new}})$
20:   **else**
21:     Reject candidate and stay put: $\left(k^{((\tau)}, \theta_{k^{(\tau)}}^{(\tau)}\right) \leftarrow \left(k^{(\tau-1)}, \theta_{k^{(\tau-1)}}^{(\tau-1)}\right)$
22:   **end if**
23: **end for**
```

with parameters corresponding to each space as

$$\theta_1 \equiv r \qquad \theta_2 \equiv (x, y)$$

We choose, for $k = 1$, a target proportional to a truncated exponential distribution. For $k = 2$, our target is a two dimensional Gaussian density truncated on an elliptical region

$$\phi_1(r) = e^{-\lambda r}[0 \leq r \leq 1]$$
$$\phi_2(x, y) = \exp\left(-\frac{(x - \mu_x)^2 + (y - \mu_y)^2}{2\sigma^2}\right)\left[\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 \leq 1\right]$$

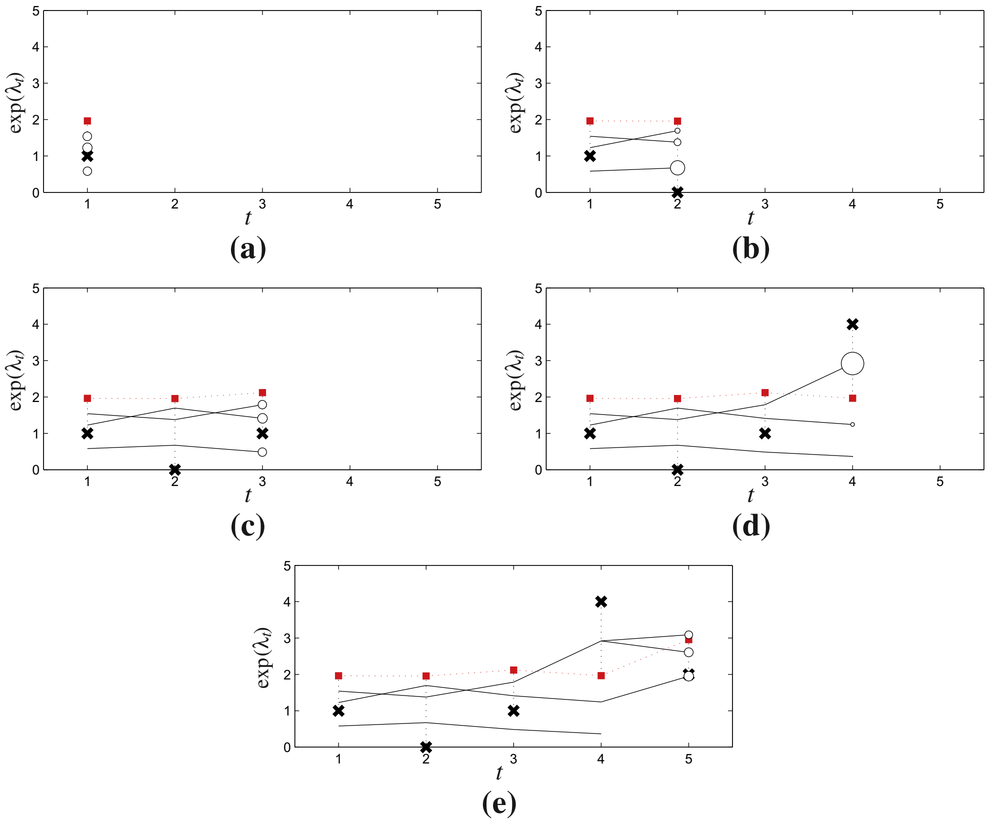

FIGURE 19.23

Illustration of the Bootstrap particle filter for the model in Eq. (19.10) and Eq. (19.11) using $N = 3$ particles. The observations y_t are shown as black crosses. The squares show the true state of the process $\exp(\lambda_t)$ used for generating the observations y_t. Remember that $\langle y_t \rangle = \exp(\lambda_t)$ due to the Poisson assumption so it is natural to plot the y_t and $\exp(\lambda_t)$ on the same plot. At each time t, the particles are shown as individual trajectories of $\lambda_{1:t}$ with the circle size proportional to the normalized weight at time t. The edges denote the ancestral links. While the term "particle" suggests that each particle is only a point in λ_t, it is actually more natural to view a particle as an entire trajectory of $\lambda_{1:t}$. At each time t, each particle selects a new position l_{t+1} and the new weight $W_{t+1}(l_{1:t+1})$ is computed recursively. If the effective sample size drops below a chosen threshold, we compute the normalized weights $\tilde{w}^{(1:N)}$ and and select each particle via resampling (See at time $t = 4$) and reset the weights to $1/N$.

the normalization constants $Z_1 = \int_0^1 e^{-\lambda r} dr$ and $Z_2 = \int \int \phi_2(x, y) dx dy$ are assumed to be unknown. As shown earlier, these cancel out in the acceptance probability. The target density is

$$\pi(k, \theta_k) = \frac{1}{p_1 Z_1 + p_2 Z_2} ([k = 1]\phi_1(r) p_1 + [k = 2]\phi_2(x, y) p_2) \tag{19.17}$$

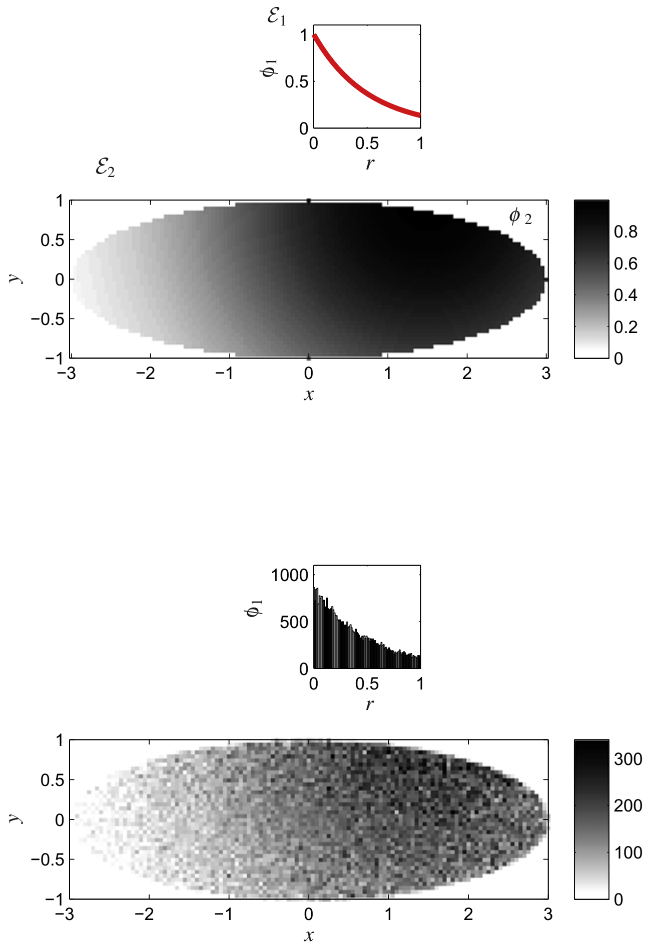

FIGURE 19.24

(Top) The target density $\pi \propto [k=1]\phi_1(r)p_1 + [k=2]\phi_2(x,y)p_2$, with parameters $\lambda = 1, a = 3, b = 1, \mu_x = 1.5, \mu_y = 1, \sigma^2 = 4, p_1 = p_2 = 1$. (Bottom) Histograms obtained from samples of the RJ-MCMC algorithm.

This target density is illustrated on Figures 19.23,19.24,19.25.

We will derive now a RJ-MCMC method to sample from the target density defined in Eq. (19.17).

1.19.6.2.1 ***Acceptance probabilities***

To design a RJ-MCMC algorithm, we should define an "ordinary" MCMC algorithm to sample within each space $\mathcal{E}_k$ for all $k \in \mathcal{K}$ and transdimensional moves for each pair k, k' such that $k \neq k'$ using the general formula for the RJ-MCMC acceptance probability α as given in 19.14.

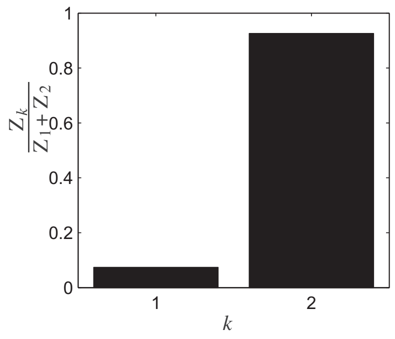

FIGURE 19.25

Histogram estimate of the marginal $\pi(k)$. By simply counting and normalizing the number of steps the chain spends in each space, we can estimate the marginal $\pi(k)$, with probabilities $p_1 Z_1/(p_1 Z_1 + p_2 Z_2)$ and $p_2 Z_2/(p_1 Z_1 + p_2 Z_2)$. Such quantities are useful for model comparison.

$\mathcal{E}_1 \rightarrow \mathcal{E}_1$.
We use a Metropolis algorithm with an independent uniform proposal such that

$$r' \sim \mathcal{U}(0, 1),$$

giving the acceptance probability as

$$\alpha_{1\rightarrow 1} = \min\left\{1, \frac{\phi_1(r')}{\phi_1(r)}\right\}, \tag{19.18}$$

$\mathcal{E}_2 \rightarrow \mathcal{E}_2$
We choose a symmetric random walk proposal

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \epsilon, \tag{19.19}$$

where $\epsilon \sim \mathcal{N}(0, vI)$. Since the proposal is symmetric, the Metropolis acceptance probability is

$$\alpha_{2\rightarrow 2} = \min\left\{1, \frac{\phi_2(x', y')}{\phi_2(x, y)}\right\} \tag{19.20}$$

$\mathcal{E}_1 \rightarrow \mathcal{E}_2$
In order to make a transition from the one dimensional space $\mathcal{E}_1$ to the two dimensional space $\mathcal{E}_2$, we define the auxiliary variable u_1 such that

$$u_1 \sim q_1(u_1) = U(0, 2\pi) = \frac{1}{2\pi} \tag{19.21}$$

and define the following one-to-one transformation $(x, y) = g(r, u)$ as

$$x = ar\cos(u) \qquad y = br\sin(u).$$

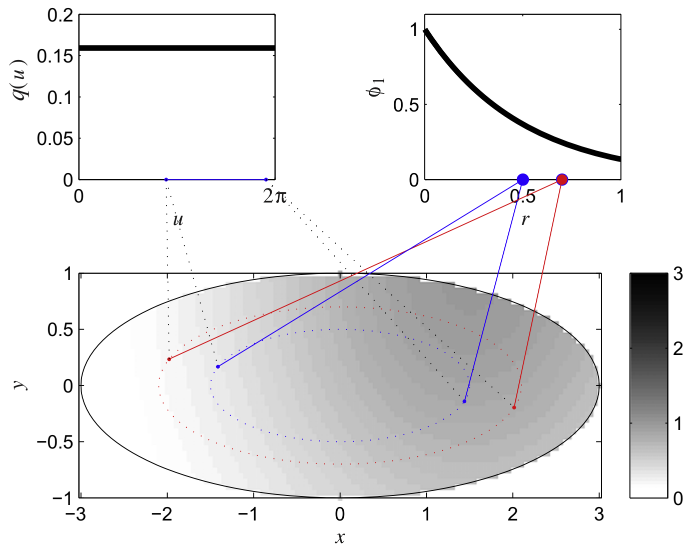

FIGURE 19.26

The mapping $(x, y) = g(r, u)$. For trans-dimensional jumps between the spaces $\mathcal{E}_1$ (top, right) and $\mathcal{E}_2$ (bottom), to match the dimensions, we draw an angle uniformly $u \sim q(u) = \mathcal{U}(0, 2\pi)$ and construct a one-to-one mapping g. A proposal f_1 on $\mathcal{E}_1$ translates to a proposal f_2 on $\mathcal{E}_2$ as $f_2(x, y) = f_1(r(x, y))q(u(x, y))|J_{2\to 1}|$. The acceptance probability does not depend on f_1 or f_2 as these cancel out; only the term $q(u(x, y))|J_{2\to 1}|$ remains.

Each r is mapped uniformly into a point on the ellipse $x^2/a + y^2/b = r^2$. This mapping is depicted in Figures 19.26,19.27. To compute the acceptance probability, we need the Jacobian. The partial derivatives of the transformation are obtained as follows

$$\frac{\partial x}{\partial r} = a\cos(u) \qquad \frac{\partial x}{\partial u} = -ar\sin(u) \tag{19.22}$$
$$\frac{\partial y}{\partial r} = b\sin(u) \qquad \frac{\partial y}{\partial u} = br\cos(u)$$

and the Jacobian term can then be determined as

$$\begin{aligned}|J_{1\to 2}| &= \left|\frac{\partial x}{\partial r}\frac{\partial y}{\partial u} - \frac{\partial x}{\partial u}\frac{\partial y}{\partial r}\right| \\ &= |a\cos(u)br\cos(u) + ar\sin(u)b\sin(u)| = \left|abr(\cos^2(u) + \sin^2(u))\right| = abr\end{aligned}$$

Finally, substituting the results the acceptance probability $\alpha_{1\to 2}$ is obtained as

$$\alpha_{1\to 2} = \min\left\{1, \frac{\phi_2(x, y)p_2}{\phi_1(r)p_1}\frac{q_{2\to 1}}{q_{1\to 2}}\frac{1}{(1/2\pi)}|J_{1\to 2}|\right\} \tag{19.23}$$

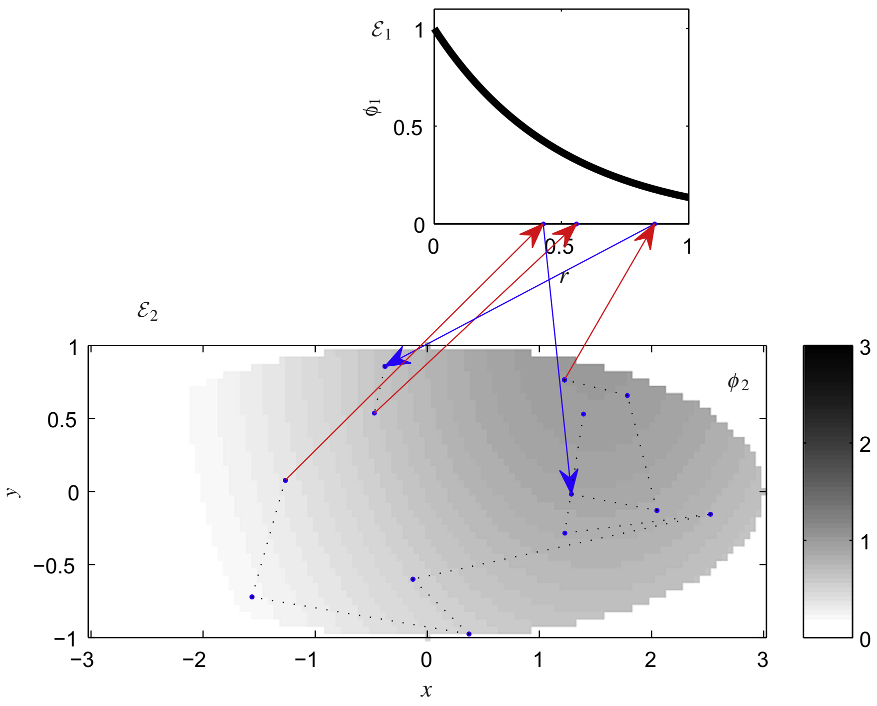

FIGURE 19.27

Illustration of the RJ-MCMC algorithm on the toy problem. The dotted lines corresponds to accepted moves within the same space, and arrows correspond to transdimensional moves. The algorithm proceeds as follows: Suppose at step τ, the chain is in space $\mathcal{E}_{k^{(\tau)}}$. We first propose the next index k' with probability $q_{k^{(\tau)}\to k'}$. If $k' = k^{(\tau)}$, we use an ordinary MH move. Otherwise, we draw a new $u \sim p(u)$ and propose a new point $\theta_{k'}(\theta^{(\tau)}_{k^{(\tau)}}, u) \in \mathcal{E}_{k'}$ and accept it with probability $\alpha_{k^{(\tau)}\to k}$. If the move is accepted we set $(k^{(\tau+1)}, \theta^{(\tau+1)}_{k^{(\tau+1)}}) \leftarrow (k', \theta_{k'})$. Otherwise, the chain stays at the same point point is equal to the old point $(k^{(\tau+1)}, \theta^{(\tau+1)}_{k^{(\tau+1)}}) \leftarrow (k^{(\tau)}, \theta^{(\tau)}_{k^{(\tau)}})$.

$\mathcal{E}_2 \to \mathcal{E}_1$
We find the inverse transformation g^{-1} ,

$$r = \sqrt{\frac{x^2}{a^2} + \frac{y^2}{b^2}} \qquad u = \arccos\left(\frac{xb}{\sqrt{x^2b^2 + y^2a^2}}\right)$$

The Jacobian $J_{2\to 1}$ is obtained by calculating the partial derivatives

$$\frac{\partial r}{\partial x} = \frac{xb}{a\sqrt{x^2b^2 + y^2a^2}} \qquad \frac{\partial r}{\partial y} = \frac{ya}{b\sqrt{x^2b^2 + y^2a^2}}$$
$$\frac{\partial u}{\partial x} = -\frac{yab}{x^2b^2 + y^2a^2}\frac{y}{|y|} \qquad \frac{\partial u}{\partial y} = \frac{xab}{x^2b^2 + y^2a^2}\frac{y}{|y|}$$

and the determinant is

$$|J_{2\to 1}| = \frac{1}{\sqrt{x^2b^2 + y^2a^2}}$$

Note that we could have avoided this calculation by directly using the Jacobian of the inverse as $|J_{2\to 1}| = |J_{1\to 2}|^{-1} = 1/(abr)$, giving the same result. The acceptance probability $\alpha_{2\to 1}$ is found as follows:

$$\alpha_{2\to 1} = \min\left\{1, \frac{\phi_1(r)p_1}{\phi_2(x,y)p_2}\frac{q_{1\to 2}(1/2\pi)}{q_{2\to 1}}|J_{2\to 1}|\right\} \tag{19.24}$$

1.19.7 Open issues and problems

In this tutorial, we have sketched in a self contained manner the basic techniques and the principles of Monte Carlo computation, along with several examples. Monte Carlo computation is a very broad topic and with the available computing power, researchers are tackling increasingly more complex models and problems. The need for extracting structure from large datasets forces the researchers to devise increasingly more complicated models that ask for more efficient inference methodologies. In some research communities, most notably machine learning, the MC methods are considered not very scalable to very large data sets due to their heavy computational requirements. It is an open issue in research for developing MC methods for such datasets. The research frontier, at the time of this writing, is swiftly moving. Researchers deal with techniques such as adaptive methods that learn suitable proposals[32], combining sequential Mote Carlo with MCMC [33], likelihood free inference [6] or exploiting parallel computation [34] for scalability.

1.19.8 Further reading

To investigate topics more deeper, we suggest a few key references. For basic probability theory, Grimmet and Stirzaker [4] is a very good reference. More rigorous material, needed for a deeper understanding of advanced subjects is covered in Rosenthal [12] and for the theory of Markov chains Norris is a key reference [7]. A very gentle introduction to basic Monte Carlo methods from a machine learning perspective is in MacKay's book [5]. A good tutorial introduction to Markov chain Monte Carlo methods can be found in [6,18]. There are also excellent tutorials on sequential Monte Carlo methods, most notably by Doucet et. al. [10,30] or Liu et. al. [8,20].

Glossary

acceptance probability	17
aperiodic (chain)	14
Box-Müller method	14
CDF Cummulative Density Function	8
central limit theorem	4

References

[1] N. Metropolis, S. Ulam, The Monte Carlo method, Journal of the Am. Statist. Assoc. 44 (247) (1949) 335–341.
[2] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equations of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1091.
[3] C. Geyer, Handbook of Markov Chain Monte Carlo, chapter Introduction to Markov Chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods) 2011.
[4] G.R. Grimmett, D.R. Stirzaker, Probability and Random Processes, Oxford University Press, 2001.
[5] D.J.C. MacKay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003.
[6] S. Brooks, A. Gelman, G. Jones, X. Meng (eds.), Handbook of Markov Chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods) 2011.
[7] J.R. Norris, Markov Chains, Cambridge University Press, 1997.
[8] J.S. Liu, R. Chen, Sequential Monte Carlo methods for dynamic systems, J. Am. Statist. Assoc. 93 (1998) 1032–1044.
[9] A. Doucet, N. de Freitas, N.J. Gordon (eds.), Sequential Monte Carlo Methods in Practice, Springer Verlag, 2001.
[10] A. Doucet, A.M. Johansen, Handbook of Nonlinear Filtering, chapter A Tutorial on Particle Filtering and Smoothing: Fifteen years Later, Oxford University Press, 2010.
[11] P.J Green, Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, Biometrika 82 (4) (1995) 711-732.
[12] Jeffrey S. Rosenthal, A first look at rigorous probability theory, World Scientific, second ed., 2006.
[13] Charles M. Grinstead, Laurie J. Snell, American Mathematical Society, fourth ed., July 2006.
[14] The marsaglia random number cd-rom with the diehard battery of tests of randomness.
[15] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator, ACM Trans. Model Comput. Simul. 8 (1) (1998) 3–30.
[16] Luc Devroye, Non-Uniform Random Variate Generation, 1986.
[17] Luke Tierney, Markov chains for exploring posterior distributions, Ann. Statist. 22 (1994) 1701–1762.
[18] W.R. Gilks, S. Richardson, D.J Spiegelhalter (Eds.), Markov Chain Monte Carlo in Practice, CRC Press, London, 1996.
[19] G.O. Roberts, J.S. Rosenthal, Markov Chain Monte Carlo: Some practical implications of theoretical results, Can. J. Statist. 26 (1998) 5–31.
[20] J.S. Liu, Monte Carlo Strategies in Scientific Computing, Springer, 2004.
[21] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1970) 97–109.
[22] O. Cappé, E. Moulines, T. Rydén, Inference in Hidden Markov Models, Springer-Verlag, New York, 2005.
[23] Luke Tierney, A note on metropolis-hastings kernels for general state spaces, Ann. Appl. Probab. 8 (1998) 1–9.
[24] S. Geman, D. Geman. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images, in: M.A. Fischler, O. Firschein (Eds.), Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, Kaufmann, Los Altos, CA, 1987, pp. 564–584.
[25] A.M. Johansen, L. Evers, N. Whiteley, Monte Carlo Methods, Online Resource, 2008 (Lecture Notes).
[26] P. Fearnhead, Exact and Efficient Bayesian inference for multiple changepoint problems, Technical report, Dept. Math. Stat., Lancaster University, 2003.
[27] Jarrett, A note on the intervals between coal mining disasters, Biometrika 66 (1979) 191–193.

[28] A. Doucet, N. de Freitas, N.J. Gordon (eds.), Sequential Monte Carlo Methods in Practice, Springer-Verlag, New York, 2001.

[29] O. Cappe, R. Douc, E. Moulines, Comparison of resampling schemes for particle filtering, in: 4th International Symposium on Image and Signal Processing and Analysis (ISPA), Zagrep, Croatia, September 2005.

[30] A. Doucet, S. Godsill, C. Andrieu, On sequential Monte Carlo sampling methods for Bayesian filtering, Statist. Comput. 10 (3) (2000) 197-208.

[31] Y. Fan, S. Sisson, Handbook of Markov Chain Monte Carlo, Chapter Reversible jump Markov chain Monte Carlo (Chapman & Hall/CRC Handbooks of Modern Statistical Methods) 2011.

[32] C. Andrieu, J. Thoms. A tutorial on adaptive mcmc, Statist. Comput. 18 (4) (2008) 343–373.

[33] Pierre Del Moral, Arnaud Doucet, Ajay Jasra. Sequential Monte Carlo samplers, J. Roy. Statist. Soc. Ser. B Stat. Methodol. 68 (3) (2006) 411–436.

[34] Anthony Lee, Christopher Yau, Michael B Giles, Arnaud Doucet, Christopher C. Holmes, On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods, May 2009, pp. 4–6.