30

Efficient Monte Carlo Methods

This chapter discusses several methods for reducing random walk behaviour in Metropolis methods. The aim is to reduce the time required to obtain effectively independent samples. For brevity, we will say 'independent samples' when we mean 'effectively independent samples'.

30.1 Hamiltonian Monte Carlo

The Hamiltonian Monte Carlo method is a Metropolis method, applicable to continuous state spaces, that makes use of gradient information to reduce random walk behaviour. [The Hamiltonian Monte Carlo method was originally called hybrid Monte Carlo, for historical reasons.]

For many systems whose probability $P(\mathbf{x})$ can be written in the form

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z}, \tag{30.1}$$

not only $E(\mathbf{x})$ but also its gradient with respect to $\mathbf{x}$ can be readily evaluated. It seems wasteful to use a simple random-walk Metropolis method when this gradient is available – the gradient indicates which direction one should go in to find states that have higher probability!

Overview of Hamiltonian Monte Carlo

In the Hamiltonian Monte Carlo method, the state space $\mathbf{x}$ is augmented by *momentum variables* $\mathbf{p}$, and there is an alternation of two types of proposal. The first proposal randomizes the momentum variable, leaving the state $\mathbf{x}$ unchanged. The second proposal changes both $\mathbf{x}$ and $\mathbf{p}$ using simulated Hamiltonian dynamics as defined by the Hamiltonian

$$H(\mathbf{x}, \mathbf{p}) = E(\mathbf{x}) + K(\mathbf{p}), \tag{30.2}$$

where $K(\mathbf{p})$ is a 'kinetic energy' such as $K(\mathbf{p}) = \mathbf{p}^{\mathsf{T}}\mathbf{p}/2$. These two proposals are used to create (asymptotically) samples from the joint density

$$P_H(\mathbf{x}, \mathbf{p}) = \frac{1}{Z_H} \exp[-H(\mathbf{x}, \mathbf{p})] = \frac{1}{Z_H} \exp[-E(\mathbf{x})] \exp[-K(\mathbf{p})]. \tag{30.3}$$

This density is separable, so the marginal distribution of $\mathbf{x}$ is the desired distribution $\exp[-E(\mathbf{x})]/Z$. So, simply discarding the momentum variables, we obtain a sequence of samples $\{\mathbf{x}^{(t)}\}$ that asymptotically come from $P(\mathbf{x})$.

```
g = gradE ( x ) ;                  # set gradient using initial x
E = findE ( x ) ;                  # set objective function too

for l = 1:L                        # loop L times
   p = randn ( size(x) ) ;         # initial momentum is Normal(0,1)
   H = p' * p / 2 + E ;            # evaluate H(x,p)

   xnew = x ;  gnew = g ;
   for tau = 1:Tau                 # make Tau 'leapfrog' steps

      p = p - epsilon * gnew / 2 ; # make half-step in p
      xnew = xnew + epsilon * p ;  # make step in x
      gnew = gradE ( xnew ) ;      # find new gradient
      p = p - epsilon * gnew / 2 ; # make half-step in p

   endfor

   Enew = findE ( xnew ) ;         # find new value of H
   Hnew = p' * p / 2 + Enew ;
   dH = Hnew - H ;                 # Decide whether to accept

   if ( dH < 0 )                accept = 1 ;
   elseif ( rand() < exp(-dH) ) accept = 1 ;
   else                         accept = 0 ;
   endif

   if ( accept )
      g = gnew ;   x = xnew ;    E = Enew ;
   endif
endfor
```

Algorithm 30.1. `Octave` source code for the Hamiltonian Monte Carlo method.

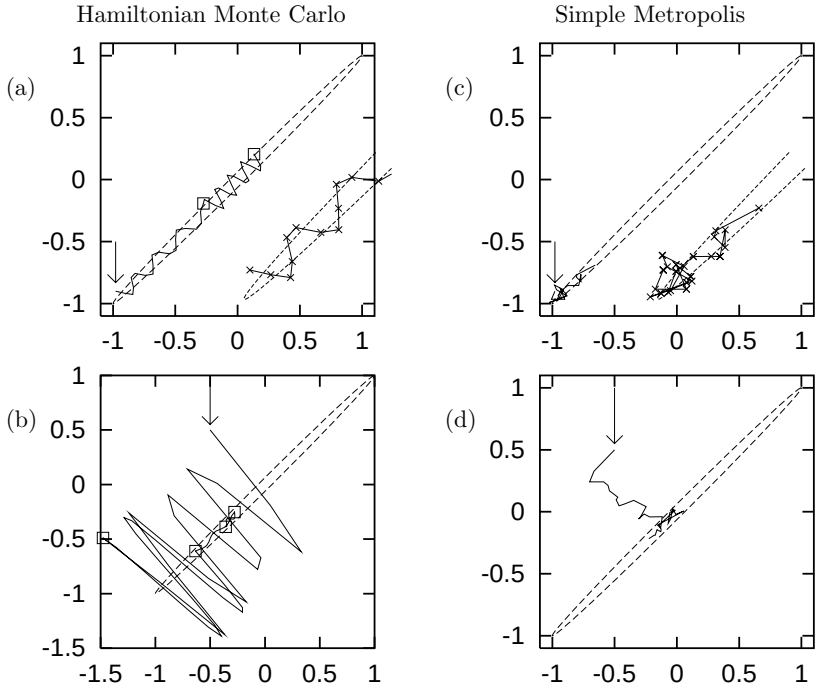


Figure 30.2. (a,b) Hamiltonian Monte Carlo used to generate samples from a bivariate Gaussian with correlation $\rho = 0.998$. (c,d) For comparison, a simple random-walk Metropolis method, given equal computer time.

Details of Hamiltonian Monte Carlo

The first proposal, which can be viewed as a Gibbs sampling update, draws a new momentum from the Gaussian density $\exp[-K(\mathbf{p})]/Z_K$. This proposal is always accepted. During the second, dynamical proposal, the momentum variable determines where the state $\mathbf{x}$ goes, and the *gradient* of $E(\mathbf{x})$ determines how the momentum $\mathbf{p}$ changes, in accordance with the equations

$$\dot{\mathbf{x}} = \mathbf{p} \tag{30.4}$$

$$\dot{\mathbf{p}} = -\frac{\partial E(\mathbf{x})}{\partial \mathbf{x}}. \tag{30.5}$$

Because of the persistent motion of $\mathbf{x}$ in the direction of the momentum $\mathbf{p}$ during each dynamical proposal, the state of the system tends to move a distance that goes *linearly* with the computer time, rather than as the square root.

The second proposal is accepted in accordance with the Metropolis rule. If the simulation of the Hamiltonian dynamics is numerically perfect then the proposals are accepted every time, because the total energy $H(\mathbf{x}, \mathbf{p})$ is a constant of the motion and so a in equation (29.31) is equal to one. If the simulation is imperfect, because of finite step sizes for example, then some of the dynamical proposals will be rejected. The rejection rule makes use of the change in $H(\mathbf{x}, \mathbf{p})$, which is zero if the simulation is perfect. The occasional rejections ensure that, asymptotically, we obtain samples $(\mathbf{x}^{(t)}, \mathbf{p}^{(t)})$ from the required joint density $P_H(\mathbf{x}, \mathbf{p})$.

The source code in figure 30.1 describes a Hamiltonian Monte Carlo method that uses the 'leapfrog' algorithm to simulate the dynamics on the function `findE(x)`, whose gradient is found by the function `gradE(x)`. Figure 30.2 shows this algorithm generating samples from a bivariate Gaussian whose energy function is $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{A}\mathbf{x}$ with

$$\mathbf{A} = \begin{bmatrix} 250.25 & -249.75 \\ -249.75 & 250.25 \end{bmatrix}, \tag{30.6}$$

corresponding to a variance–covariance matrix of

$$\begin{bmatrix} 1 & 0.998 \\ 0.998 & 1 \end{bmatrix}. \tag{30.7}$$

In figure 30.2a, starting from the state marked by the arrow, the solid line represents two successive trajectories generated by the Hamiltonian dynamics. The squares show the endpoints of these two trajectories. Each trajectory consists of `Tau` $= 19$ 'leapfrog' steps with `epsilon` $= 0.055$. These steps are indicated by the crosses on the trajectory in the magnified inset. After each trajectory, the momentum is randomized. Here, both trajectories are accepted; the errors in the Hamiltonian were only $+0.016$ and -0.06 respectively.

Figure 30.2b shows how a sequence of four trajectories converges from an initial condition, indicated by the arrow, that is not close to the typical set of the target distribution. The trajectory parameters `Tau` and `epsilon` were randomized for each trajectory using uniform distributions with means 19 and 0.055 respectively. The first trajectory takes us to a new state, $(-1.5, -0.5)$, similar in energy to the first state. The second trajectory happens to end in a state nearer the bottom of the energy landscape. Here, since the potential energy E is smaller, the kinetic energy $K = \mathbf{p}^2/2$ is necessarily larger than it was at the start of the trajectory. When the momentum is randomized before the third trajectory, its kinetic energy becomes much smaller. After the fourth

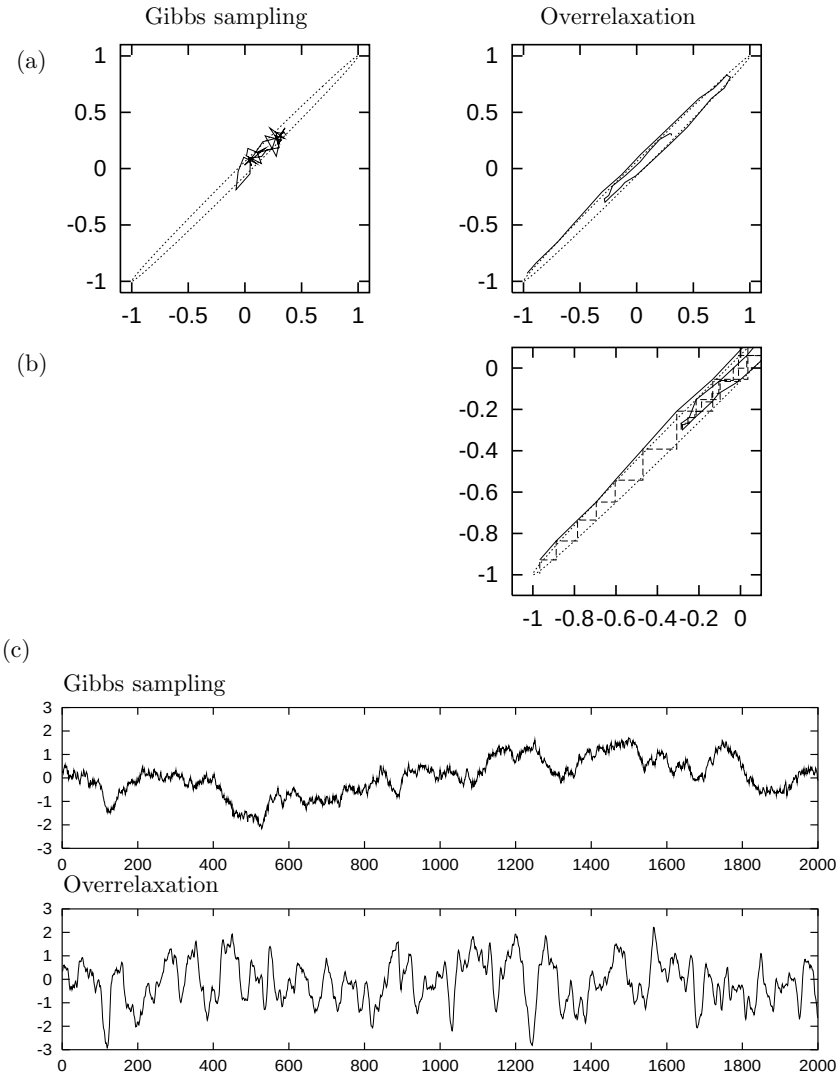


Figure 30.3. Overrelaxation contrasted with Gibbs sampling for a bivariate Gaussian with correlation $\rho = 0.998$. (a) The state sequence for 40 iterations, each iteration involving one update of both variables. The overrelaxation method had $\alpha = -0.98$. (This excessively large value is chosen to make it easy to see how the overrelaxation method reduces random walk behaviour.) The dotted line shows the contour $\mathbf{x}^{\mathrm{T}}\mathbf{\Sigma}^{-1}\mathbf{x} = 1$. (b) Detail of (a), showing the two steps making up each iteration. (c) Time-course of the variable x_1 during 2000 iterations of the two methods. The overrelaxation method had $\alpha = -0.89$. (After Neal (1995).)

trajectory has been simulated, the state appears to have become typical of the target density.

Figures 30.2(c) and (d) show a random-walk Metropolis method using a Gaussian proposal density to sample from the same Gaussian distribution, starting from the initial conditions of (a) and (b) respectively. In (c) the step size was adjusted such that the acceptance rate was 58%. The number of proposals was 38 so the total amount of computer time used was similar to that in (a). The distance moved is small because of random walk behaviour. In (d) the random-walk Metropolis method was used and started from the same initial condition as (b) and given a similar amount of computer time.

▶ 30.2 Overrelaxation

The method of *overrelaxation* is a method for reducing random walk behaviour in Gibbs sampling. Overrelaxation was originally introduced for systems in which all the conditional distributions are Gaussian.

> An example of a joint distribution that is *not* Gaussian but whose conditional distributions *are* all Gaussian is $P(x, y) = \exp(-x^2y^2 - x^2 - y^2)/Z$.

Overrelaxation for Gaussian conditional distributions

In ordinary Gibbs sampling, one draws the new value $x_i^{(t+1)}$ of the current variable x_i from its conditional distribution, ignoring the old value $x_i^{(t)}$. The state makes lengthy random walks in cases where the variables are strongly correlated, as illustrated in the left-hand panel of figure 30.3. This figure uses a correlated Gaussian distribution as the target density.

In Adler's (1981) overrelaxation method, one instead samples $x_i^{(t+1)}$ from a Gaussian that is biased to the *opposite* side of the conditional distribution. If the conditional distribution of x_i is Normal(μ, σ^2) and the current value of x_i is $x_i^{(t)}$, then Adler's method sets x_i to

$$x_i^{(t+1)} = \mu + \alpha(x_i^{(t)} - \mu) + (1 - \alpha^2)^{1/2}\sigma\nu, \tag{30.8}$$

where $\nu \sim$ Normal$(0, 1)$ and α is a parameter between -1 and 1, usually set to a negative value. (If α is positive, then the method is called under-relaxation.)

Exercise 30.1.[2] Show that this individual transition leaves invariant the conditional distribution $x_i \sim$ Normal(μ, σ^2).

A single iteration of Adler's overrelaxation, like one of Gibbs sampling, updates each variable in turn as indicated in equation (30.8). The transition matrix $T(\mathbf{x}'; \mathbf{x})$ defined by a complete update of all variables in some fixed order does not satisfy detailed balance. Each individual transition for one coordinate just described *does* satisfy detailed balance – so the overall chain gives a valid sampling strategy which converges to the target density $P(\mathbf{x})$ – but when we form a chain by applying the individual transitions in a fixed sequence, the overall chain is not reversible. This temporal asymmetry is the key to why overrelaxation can be beneficial. If, say, two variables are positively correlated, then they will (on a short timescale) evolve in a directed manner instead of by random walk, as shown in figure 30.3. This may significantly reduce the time required to obtain independent samples.

Exercise 30.2.[3] The transition matrix $T(\mathbf{x}'; \mathbf{x})$ defined by a complete update of all variables in some fixed order does not satisfy detailed balance. If the updates were in a *random order*, then T would be symmetric. Investigate, for the toy two-dimensional Gaussian distribution, the assertion that the advantages of overrelaxation are lost if the overrelaxed updates are made in a random order.

Ordered Overrelaxation

The overrelaxation method has been generalized by Neal (1995) whose *ordered overrelaxation* method is applicable to *any* system where Gibbs sampling is used. In ordered overrelaxation, instead of taking one sample from the conditional distribution $P(x_i \mid \{x_j\}_{j \neq i})$, we create K such samples $x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(K)}$, where K might be set to twenty or so. Often, generating $K - 1$ extra samples adds a negligible computational cost to the initial computations required for making the first sample. The points $\{x_i^{(k)}\}$ are then sorted numerically, and the current value of x_i is inserted into the sorted list, giving a list of $K + 1$ points. We give them ranks $0, 1, 2, \ldots, K$. Let κ be the rank of the current value of x_i in the list. We set x_i' to the value that is an equal distance from the other end of the list, that is, the value with rank $K - \kappa$. The role played by Adler's α parameter is here played by the parameter K. When $K = 1$, we obtain ordinary Gibbs sampling. For practical purposes Neal estimates that ordered overrelaxation may speed up a simulation by a factor of ten or twenty.

30.3 Simulated annealing

A third technique for speeding convergence is *simulated annealing*. In simulated annealing, a 'temperature' parameter is introduced which, when large, allows the system to make transitions that would be improbable at temperature 1. The temperature is set to a large value and gradually reduced to 1. This procedure is supposed to reduce the chance that the simulation gets stuck in an unrepresentative probability island.

We asssume that we wish to sample from a distribution of the form

$$P(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{Z} \tag{30.9}$$

where $E(\mathbf{x})$ can be evaluated. In the simplest simulated annealing method, we instead sample from the distribution

$$P_T(\mathbf{x}) = \frac{1}{Z(T)}\, e^{-\frac{E(\mathbf{x})}{T}} \tag{30.10}$$

and decrease T gradually to 1.

Often the energy function can be separated into two terms,

$$E(\mathbf{x}) = E_0(\mathbf{x}) + E_1(\mathbf{x}), \tag{30.11}$$

of which the first term is 'nice' (for example, a separable function of $\mathbf{x}$) and the second is 'nasty'. In these cases, a better simulated annealing method might make use of the distribution

$$P'_T(\mathbf{x}) = \frac{1}{Z'(T)}\, e^{-E_0(\mathbf{x}) - E_1(\mathbf{x})/T} \tag{30.12}$$

with T gradually decreasing to 1. In this way, the distribution at high temperatures reverts to a well-behaved distribution defined by E_0.

Simulated annealing is often used as an optimization method, where the aim is to find an $\mathbf{x}$ that minimizes $E(\mathbf{x})$, in which case the temperature is decreased to zero rather than to 1.

As a Monte Carlo method, simulated annealing as described above doesn't sample exactly from the right distribution, because there is no guarantee that the probability of falling into one basin of the energy is equal to the total probability of all the states in that basin. The closely related 'simulated tempering' method (Marinari and Parisi, 1992) corrects the biases introduced by the annealing process by making the temperature itself a random variable that is updated in Metropolis fashion during the simulation. Neal's (1998) 'annealed importance sampling' method removes the biases introduced by annealing by computing importance weights for each generated point.

30.4 Skilling's multi-state leapfrog method

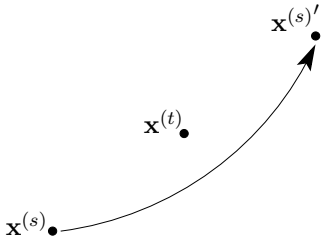
A fourth method for speeding up Monte Carlo simulations, due to John Skilling, has a similar spirit to overrelaxation, but works in more dimensions. This method is applicable to sampling from a distribution over a continuous state space, and the sole requirement is that the energy $E(\mathbf{x})$ should be easy to evaluate. The gradient is not used. This leapfrog method is not intended to be used on its own but rather in sequence with other Monte Carlo operators.

Instead of moving just one state vector $\mathbf{x}$ around the state space, as was the case for all the Monte Carlo methods discussed thus far, Skilling's leapfrog method simultaneously maintains a *set* of S state vectors $\{\mathbf{x}^{(s)}\}$, where S

might be six or twelve. The aim is that all S of these vectors will represent independent samples from the same distribution $P(\mathbf{x})$.

Skilling's leapfrog makes a proposal for the new state $\mathbf{x}^{(s)\prime}$, which is accepted or rejected in accordance with the Metropolis method, by leapfrogging the current state $\mathbf{x}^{(s)}$ over another state vector $\mathbf{x}^{(t)}$:

$$\mathbf{x}^{(s)\prime} = \mathbf{x}^{(t)} + (\mathbf{x}^{(t)} - \mathbf{x}^{(s)}) = 2\mathbf{x}^{(t)} - \mathbf{x}^{(s)}. \tag{30.13}$$



All the other state vectors are left where they are, so the acceptance probability depends only on the change in energy of $\mathbf{x}^{(s)}$.

Which vector, t, is the partner for the leapfrog event can be chosen in various ways. The simplest method is to select the partner at random from the other vectors. It might be better to choose t by selecting one of the nearest neighbours $\mathbf{x}^{(s)}$ – nearest by any chosen distance function – as long as one then uses an acceptance rule that ensures detailed balance by checking whether point t is still among the nearest neighbours of the new point, $\mathbf{x}^{(s)\prime}$.

Why the leapfrog is a good idea

Imagine that the target density $P(\mathbf{x})$ has strong correlations – for example, the density might be a needle-like Gaussian with width ϵ and length $L\epsilon$, where $L \gg 1$. As we have emphasized, motion around such a density by standard methods proceeds by a slow random walk.

Imagine now that our set of S points is lurking initially in a location that is probable under the density, but in an inappropriately small ball of size ϵ. Now, under Skilling's leapfrog method, a typical first move will take the point a little outside the current ball, perhaps doubling its distance from the centre of the ball. After all the points have had a chance to move, the ball will have increased in size; if all the moves are accepted, the ball will be bigger by a factor of two or so in all dimensions. The rejection of some moves will mean that the ball containing the points will probably have elongated in the needle's long direction by a factor of, say, two. After another cycle through the points, the ball will have grown in the long direction by another factor of two. So the typical distance travelled in the long dimension grows *exponentially* with the number of iterations.

Now, maybe a factor of two growth per iteration is on the optimistic side; but even if the ball only grows by a factor of, let's say, 1.1 per iteration, the growth is nevertheless exponential. It will only take a number of iterations proportional to $\log L / \log(1.1)$ for the long dimension to be explored.

▷ Exercise 30.3.[2, p.398] Discuss how the effectiveness of Skilling's method scales with dimensionality, using a correlated N-dimensional Gaussian distribution as an example. Find an expression for the rejection probability, assuming the Markov chain is at equilibrium. Also discuss how it scales with the strength of correlation among the Gaussian variables. [Hint: Skilling's method is invariant under affine transformations, so the rejection probability at equilibrium can be found by looking at the case of a *separable* Gaussian.]

This method has some similarity to the 'adaptive direction sampling' method of Gilks *et al.* (1994) but the leapfrog method is simpler and can be applied to a greater variety of distributions.

▶ 30.5 Monte Carlo algorithms as communication channels

It may be a helpful perspective, when thinking about speeding up Monte Carlo methods, to think about the information that is being communicated. Two communications take place when a sample from $P(\mathbf{x})$ is being generated.

First, the selection of a particular $\mathbf{x}$ from $P(\mathbf{x})$ necessarily requires that at least $\log 1/P(\mathbf{x})$ random bits be consumed. [Recall the use of inverse arithmetic coding as a method for generating samples from given distributions (section 6.3).]

Second, the generation of a sample conveys information about $P(\mathbf{x})$ from the subroutine that is able to evaluate $P^*(\mathbf{x})$ (and from any other subroutines that have access to properties of $P^*(\mathbf{x})$).

Consider a dumb Metropolis method, for example. In a dumb Metropolis method, the proposals $Q(\mathbf{x}'; \mathbf{x})$ have nothing to do with $P(\mathbf{x})$. Properties of $P(\mathbf{x})$ are only involved in the algorithm at the acceptance step, when the ratio $P^*(\mathbf{x}')/P^*(\mathbf{x})$ is computed. The channel from the true distribution $P(\mathbf{x})$ to the user who is interested in computing properties of $P(\mathbf{x})$ thus passes through a bottleneck: all the information about P is conveyed by the string of acceptances and rejections. If $P(\mathbf{x})$ were replaced by a different distribution $P_2(\mathbf{x})$, the only way in which this change would have an influence is that the string of acceptances and rejections would be changed. I am not aware of much use being made of this information-theoretic view of Monte Carlo algorithms, but I think it is an instructive viewpoint: if the aim is to obtain information about properties of $P(\mathbf{x})$ then presumably it is helpful to identify the channel through which this information flows, and maximize the rate of information transfer.

Example 30.4. The information-theoretic viewpoint offers a simple justification for the widely-adopted rule of thumb, which states that the parameters of a dumb Metropolis method should be adjusted such that the acceptance rate is about one half. Let's call the acceptance history, that is, the binary string of accept or reject decisions, $\mathbf{a}$. The information learned about $P(\mathbf{x})$ after the algorithm has run for T steps is less than or equal to the information content of $\mathbf{a}$, since all information about P is mediated by $\mathbf{a}$. And the information content of $\mathbf{a}$ is upper-bounded by $TH_2(f)$, where f is the acceptance rate. This bound on information acquired about P is maximized by setting $f = 1/2$.

Another helpful analogy for a dumb Metropolis method is an evolutionary one. Each proposal generates a progeny $\mathbf{x}'$ from the current state $\mathbf{x}$. These two individuals then compete with each other, and the Metropolis method uses a noisy survival-of-the-fittest rule. If the progeny $\mathbf{x}'$ is fitter than the parent (i.e., $P^*(\mathbf{x}') > P^*(\mathbf{x})$, assuming the Q/Q factor is unity) then the progeny replaces the parent. The survival rule also allows less-fit progeny to replace the parent, sometimes. Insights about the rate of evolution can thus be applied to Monte Carlo methods.

Exercise 30.5.[3] Let $\mathbf{x} \in \{0, 1\}^G$ and let $P(\mathbf{x})$ be a separable distribution,

$$P(\mathbf{x}) = \prod_g p(x_g), \tag{30.14}$$

with $p(0) = p_0$ and $p(1) = p_1$, for example $p_1 = 0.1$. Let the proposal density of a dumb Metropolis algorithm Q involve flipping a fraction m of the G bits in the state $\mathbf{x}$. Analyze how long it takes for the chain to

converge to the target density as a function of m. Find the optimal m and deduce how long the Metropolis method must run for.

Compare the result with the results for an evolving population under natural selection found in Chapter 19.

The insight that the fastest progress that a standard Metropolis method can make, in information terms, is about one bit per iteration, gives a strong motivation for speeding up the algorithm. This chapter has already reviewed several methods for reducing random-walk behaviour. Do these methods also speed up the rate at which information is acquired?

Exercise 30.6.[4] Does Gibbs sampling, which is a smart Metropolis method whose proposal distributions do depend on $P(\mathbf{x})$, allow information about $P(\mathbf{x})$ to leak out at a rate faster than one bit per iteration? Find toy examples in which this question can be precisely investigated.

Exercise 30.7.[4] Hamiltonian Monte Carlo is another smart Metropolis method in which the proposal distributions depend on $P(\mathbf{x})$. Can Hamiltonian Monte Carlo extract information about $P(\mathbf{x})$ at a rate faster than one bit per iteration?

Exercise 30.8.[5] In importance sampling, the weight $w_r = P^*(\mathbf{x}^{(r)})/Q^*(\mathbf{x}^{(r)})$, a floating-point number, is computed and retained until the end of the computation. In contrast, in the dumb Metropolis method, the ratio $a = P^*(\mathbf{x}')/P^*(\mathbf{x})$ is reduced to a single bit ('is a bigger than or smaller than the random number u?'). Thus in principle importance sampling preserves more information about P^* than does dumb Metropolis. Can you find a toy example in which this extra information does indeed lead to faster convergence of importance sampling than Metropolis? Can you design a Markov chain Monte Carlo algorithm that moves around adaptively, like a Metropolis method, and that retains more useful information about the value of P^*, like importance sampling?

In Chapter 19 we noticed that an evolving population of N individuals can make faster evolutionary progress if the individuals engage in sexual reproduction. This observation motivates looking at Monte Carlo algorithms in which multiple parameter vectors $\mathbf{x}$ are evolved and interact.

▶ 30.6 Multi-state methods

In a multi-state method, multiple parameter vectors $\mathbf{x}$ are maintained; they evolve individually under moves such as Metropolis and Gibbs; there are also interactions among the vectors. The intention is either that eventually all the vectors $\mathbf{x}$ should be samples from $P(\mathbf{x})$ (as illustrated by Skilling's leapfrog method), or that information associated with the final vectors $\mathbf{x}$ should allow us to approximate expectations under $P(\mathbf{x})$, as in importance sampling.

Genetic methods

Genetic algorithms are not often described by their proponents as Monte Carlo algorithms, but I think this is the correct categorization, and an ideal genetic algorithm would be one that can be proved to be a valid Monte Carlo algorithm that converges to a specified density.

I'll use R to denote the number of vectors in the population. We aim to have $P^*(\{\mathbf{x}^{(r)}\}_1^R) = \prod P^*(\mathbf{x}^{(r)})$. A genetic algorithm involves moves of two or three types.

First, individual moves in which one state vector is perturbed, $\mathbf{x}^{(r)} \rightarrow \mathbf{x}^{(r)\prime}$, which could be performed using any of the Monte Carlo methods we have mentioned so far.

Second, we allow crossover moves of the form $\mathbf{x}, \mathbf{y} \rightarrow \mathbf{x}', \mathbf{y}'$; in a typical crossover move, the progeny $\mathbf{x}'$ receives half his state vector from one parent, $\mathbf{x}$, and half from the other, $\mathbf{y}$; the secret of success in a genetic algorithm is that the parameter $\mathbf{x}$ must be encoded in such a way that the crossover of two independent states $\mathbf{x}$ and $\mathbf{y}$, both of which have good fitness P^*, should have a reasonably good chance of producing progeny who are equally fit. This constraint is a hard one to satisfy in many problems, which is why genetic algorithms are mainly talked about and hyped up, and rarely used by serious experts. Having introduced a crossover move $\mathbf{x}, \mathbf{y} \rightarrow \mathbf{x}', \mathbf{y}'$, we need to choose an acceptance rule. One easy way to obtain a valid algorithm is to accept or reject the crossover proposal using the Metropolis rule with $P^*(\{\mathbf{x}^{(r)}\}_1^R)$ as the target density – this involves comparing the fitnesses before and after the crossover using the ratio

$$\frac{P^*(\mathbf{x}')P^*(\mathbf{y}')}{P^*(\mathbf{x})P^*(\mathbf{y})}. \tag{30.15}$$

If the crossover operator is reversible then we have an easy proof that this procedure satisfies detailed balance and so is a valid component in a chain converging to $P^*(\{\mathbf{x}^{(r)}\}_1^R)$.

▷ Exercise 30.9.[3] Discuss whether the above two operators, individual variation and crossover with the Metropolis acceptance rule, will give a more efficient Monte Carlo method than a standard method with only one state vector and no crossover.

The reason why the sexual community could acquire information faster than the asexual community in Chapter 19 was because the crossover operation produced diversity with standard deviation $\sqrt{G}$, then the Blind Watchmaker was able to convey lots of information about the fitness function by *killing off* the less fit offspring. The above two operators do *not* offer a speed-up of $\sqrt{G}$ compared with standard Monte Carlo methods because there is no killing. What's required, in order to obtain a speed-up, is two things: multiplication and death; and at least one of these must operate *selectively*. Either we must kill off the less-fit state vectors, or we must allow the more-fit state vectors to give rise to more offspring. While it's easy to sketch these ideas, it is hard to define a valid method for doing it.

Exercise 30.10.[5] Design a birth rule and a death rule such that the chain converges to $P^*(\{\mathbf{x}^{(r)}\}_1^R)$.

I believe this is still an open research problem.

Particle filters

Particle filters, which are particularly popular in inference problems involving temporal tracking, are multistate methods that mix the ideas of importance sampling and Markov chain Monte Carlo. See Isard and Blake (1996), Isard and Blake (1998), Berzuini *et al.* (1997), Berzuini and Gilks (2001), Doucet *et al.* (2001).

▶ 30.7 Methods that do not necessarily help

It is common practice to use *many* initial conditions for a particular Markov chain (figure 29.19). If you are worried about sampling well from a complicated density $P(\mathbf{x})$, *can* you ensure the states produced by the simulations are well distributed about the typical set of $P(\mathbf{x})$ by ensuring that the initial points are 'well distributed about the whole state space'?

The answer is, unfortunately, no. In hierarchical Bayesian models, for example, a large number of parameters $\{x_n\}$ may be coupled together via another parameter β (known as a hyperparameter). For example, the quantities $\{x_n\}$ might be independent noise signals, and β might be the inverse-variance of the noise source. The joint distribution of β and $\{x_n\}$ might be

$$\begin{aligned} P(\beta, \{x_n\}) &= P(\beta) \prod_{n=1}^{N} P(x_n \,|\, \beta) \\ &= P(\beta) \prod_{n=1}^{N} \tfrac{1}{Z(\beta)} \, e^{-\beta x_n^2/2}, \end{aligned}$$

where $Z(\beta) = \sqrt{2\pi/\beta}$ and $P(\beta)$ is a broad distribution describing our ignorance about the noise level. For simplicity, let's leave out all the other variables – data and such – that might be involved in a realistic problem. Let's imagine that we want to sample effectively from $P(\beta, \{x_n\})$ by Gibbs sampling – alternately sampling β from the conditional distribution $P(\beta \,|\, x_n)$ then sampling all the x_n from their conditional distributions $P(x_n \,|\, \beta)$. [The resulting marginal distribution of β should asymptotically be the broad distribution $P(\beta)$.]

If N is large then the conditional distribution of β given any particular setting of $\{x_n\}$ will be tightly concentrated on a particular most-probable value of β, with width proportional to $1/\sqrt{N}$. Progress up and down the β-axis will therefore take place by a slow random walk with steps of size $\propto 1/\sqrt{N}$.

So, to the initialization strategy. Can we finesse our slow convergence problem by using initial conditions located 'all over the state space'? Sadly, no. If we distribute the points $\{x_n\}$ widely, what we are actually doing is favouring an initial value of the noise level $1/\beta$ that is *large*. The random walk of the parameter β will thus tend, after the first drawing of β from $P(\beta \,|\, x_n)$, always to start off from one end of the β-axis.

Further reading

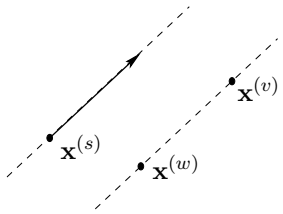
The Hamiltonian Monte Carlo method (Duane *et al.*, 1987) is reviewed in Neal (1993b). This excellent tome also reviews a huge range of other Monte Carlo methods, including the related topics of simulated annealing and free energy estimation.

▶ 30.8 Further exercises

Exercise 30.11.[4] An important detail of the Hamiltonian Monte Carlo method is that the simulation of the Hamiltonian dynamics, while it may be inaccurate, must be perfectly reversible, in the sense that if the initial condition $(\mathbf{x}, \mathbf{p})$ goes to $(\mathbf{x}', \mathbf{p}')$, then the same simulator must take $(\mathbf{x}', -\mathbf{p}')$ to $(\mathbf{x}, -\mathbf{p})$, and the inaccurate dynamics must conserve state-space volume. [The leapfrog method in algorithm 30.1 satisfies these rules.]

Explain why these rules must be satisfied and create an example illustrating the problems that arise if they are not.

Exercise 30.12.[4] A multi-state idea for slice sampling. Investigate the following multi-state method for slice sampling. As in Skilling's multi-state leapfrog method (section 30.4), maintain a set of S state vectors $\{\mathbf{x}^{(s)}\}$. Update one state vector $\mathbf{x}^{(s)}$ by one-dimensional slice sampling in a direction $\mathbf{y}$ determined by picking two other state vectors $\mathbf{x}^{(v)}$ and $\mathbf{x}^{(w)}$ at random and setting $\mathbf{y} = \mathbf{x}^{(v)} - \mathbf{x}^{(w)}$. Investigate this method on toy problems such as a highly-correlated multivariate Gaussian distribution. Bear in mind that if $S - 1$ is smaller than the number of dimensions N then this method will not be ergodic by itself, so it may need to be mixed with other methods. Are there classes of problems that are better solved by this slice-sampling method than by the standard methods for picking $\mathbf{y}$ such as cycling through the coordinate axes or picking $\mathbf{u}$ at random from a Gaussian distribution?



▶ 30.9 Solutions

Solution to exercise 30.3 (p.393). Consider the spherical Gaussian distribution where all components have mean zero and variance 1. In one dimension, the nth, if $x_n^{(1)}$ leapfrogs over $x_n^{(2)}$, we obtain the proposed coordinate

$$(x_n^{(1)})' = 2x_n^{(2)} - x_n^{(1)}. \tag{30.16}$$

Assuming that $x_n^{(1)}$ and $x_n^{(2)}$ are Gaussian random variables from Normal$(0, 1)$, $(x_n^{(1)})'$ is Gaussian from Normal$(0, \sigma^2)$, where $\sigma^2 = 2^2 + (-1)^2 = 5$. The change in energy contributed by this one dimension will be

$$\frac{1}{2}\left[(2x_n^{(2)} - x_n^{(1)})^2 - (x_n^{(1)})^2\right] = 2(x_n^{(2)})^2 - 2x_n^{(2)}x_n^{(1)} \tag{30.17}$$

so the typical change in energy is $2\langle(x_n^{(2)})^2\rangle = 2$. This positive change is bad news. In N dimensions, the typical change in energy when a leapfrog move is made, at equilibrium, is thus $+2N$. The probability of acceptance of the move scales as

$$e^{-2N}. \tag{30.18}$$

This implies that Skilling's method, as described, is not effective in very high-dimensional problems – at least, not once convergence has occurred. Nevertheless it has the impressive advantage that its convergence properties are independent of the strength of correlations between the variables – a property that not even the Hamiltonian Monte Carlo and overrelaxation methods offer.

About Chapter 31

Some of the neural network models that we will encounter are related to Ising models, which are idealized magnetic systems. It is not essential to understand the statistical physics of Ising models to understand these neural networks, but I hope you'll find them helpful.

Ising models are also related to several other topics in this book. We will use exact tree-based computation methods like those introduced in Chapter 25 to evaluate properties of interest in Ising models. Ising models offer crude models for binary images. And Ising models relate to two-dimensional constrained channels (cf. Chapter 17): a two-dimensional bar-code in which a black dot may not be completely surrounded by black dots, and a white dot may not be completely surrounded by white dots, is similar to an antiferromagnetic Ising model at low temperature. Evaluating the entropy of this Ising model is equivalent to evaluating the capacity of the constrained channel for conveying bits.

If you would like to jog your memory on statistical physics and thermodynamics, you might find Appendix B helpful. I also recommend the book by Reif (1965).