16

Monte Carlo methods

16.1 The Monte Carlo method

The Monte Carlo method is simple, robust, and useful. It has many applications. It is used, for instance, in numerical integration, data analysis, statistical mechanics, lattice gauge theory, chemical physics, biophysics, and finance.

16.2 Numerical Integration

Suppose one wants to numerically integrate a function $f(x)$ of a vector $x = (x_1, \ldots, x_n)$ over a region $\mathcal{R}$. One generates a large number N of pseudorandom values for the n coordinates x within a hyperrectangle of length L that contains the region $\mathcal{R}$, keeps the $N_\mathcal{R}$ points $x_k = (x_{1k}, \ldots, x_{nk})$ that fall within the region $\mathcal{R}$, computes the average $\langle f(x_k) \rangle$, and multiplies by the hypervolume $V_\mathcal{R}$ of the region

$$\int_\mathcal{R} f(x)\, d^n x \approx \frac{V_\mathcal{R}}{N_\mathcal{R}} \sum_{k=1}^{N_\mathcal{R}} f(x_k). \tag{16.1}$$

If the hypervolume $V_\mathcal{R}$ is hard to compute, you can have the Monte Carlo code compute it for you. The hypervolume $V_\mathcal{R}$ is the volume L^n of the enclosing hypercube multiplied by the number $N_\mathcal{R}$ of times the N points fall within the region $\mathcal{R}$

$$V_\mathcal{R} = \frac{N_\mathcal{R}}{N}\, L^n. \tag{16.2}$$

The integral formula (16.1) then becomes

$$\int_\mathcal{R} f(x)\, d^n x \approx \frac{L^n}{N} \sum_{k=1}^{N_\mathcal{R}} f(x_k). \tag{16.3}$$

The error falls like $1/\sqrt{N_{\mathcal{R}}}$. So to get better accuracy, one simply runs the program again with more points; one does not have to write a new code with finer n-dimensional grids of points that span the region $\mathcal{R}$.

Example 16.1 (Numerical integration) Suppose one wants to integrate the function

$$f(x,y) = \frac{e^{-2x-3y}}{\sqrt{x^2+y^2+1}} \tag{16.4}$$

over the quarter of the unit disk in which x and y are positive. In this case, $V_{\mathcal{R}}$ is the area $\pi/4$ of the quarter disk.

One may compute this integral by using the Fortran program integrate.f95 or Sean Cahill's C++ program integrate.cc both of which are in the repository Monte_Carlo_methods at `github.com/kevinecahill`.

To generate fresh random numbers, one must set the seed for the code that computes them. The program integrate.f95 sets the seed by calling its subroutine init_random_seed(). With some compilers, one can just write "call random_seed()." □

16.3 Quasirandom numbers

The method of the previous section is easy to use, but one can improve its accuracy by using **quasirandom numbers**, which occur with equal density in every region. Examples on the interval $(0,1)$ are the **Halton** sequences of bases 2 and 3

$$\begin{aligned} &1/2,\ 1/4,\ 3/4,\ 1/8,\ 5/8,\ 7/8,\ 1/16,\ 9/16,\ldots \\ &1/3,\ 2/3,\ 1/9,\ 4/9,\ 7/9,\ 2/9,\ 5/9,\ 8/9,\ 1/27,\ldots \end{aligned} \tag{16.5}$$

and those invented by Sobol'. If one uses quasirandom numbers instead of pseudorandom numbers to estimate integrals, then the error falls like $1/N^{2/3}$ or $1/N$, both of which are faster than the $1/\sqrt{N}$ decrease one gets with pseudorandom numbers. Codes that generate Halton and Sobol' sequences are at `people.sc.fsu.edu/~jburkardt/f_src/rnglib/rnglib.html`.

16.4 Applications to Experiments

Physicists accumulate vast quantities of data and sometimes must decide whether a particular signal is due to a defect in the detector, to a random fluctuation in the real events that they are measuring, or to a new and

unexpected phenomenon. For simplicity, let us assume that the background can be ignored and that the real events arrive randomly in time apart from extraordinary phenomena. One reliable way to evaluate an ambiguous signal is to run a Monte Carlo program that generates the kinds of real random events to which one's detector is sensitive and to use these events to compute the probability that the unusual signal occurred randomly.

To illustrate the use of random-event generators, we will consider the work of a graduate student who spent 100 days counting muons produced in an underground detector by atmospheric GeV neutrinos. Each of the very large number N of primary cosmic rays that hit the Earth every day can collide with a nucleus and make a shower of pions which in turn produce atmospheric neutrinos that can make muons in the detector. The probability p that a given cosmic ray will make a muon in the detector is very small, but the number N of primary cosmic rays is very large. In this experiment, their product pN was $\langle n \rangle = 0.1$ muons per day. Since N is huge and p tiny, the probability distribution is Poisson, and so by (15.66) the probability that n muons would be detected on any particular day is

$$P(n, \langle n \rangle) = \frac{\langle n \rangle^n}{n!} e^{-\langle n \rangle} \tag{16.6}$$

in the absence of a failure of the anti-coincidence shield or some other problem with the detector—or some hard-to-imagine astrophysical event.

The graduate student might have used a program like muons.f90 or like Sean Cahill's muons.cc to generate 1,000,000 random histories of 100 days of events distributed according to the Poisson distribution (16.6) with $\langle n \rangle = 0.1$. Both codes are in Monte_Carlo_methods at `github.com/kevinecahill`.

Figure 16.1 plots the results from this simple Monte Carlo of 1,000,000 histories of 100 days each. The boxes show that the maximum number of muons detected on a single day was $n = 1$, 2, and 3 on 62.6%, 35.9%, and 1.5% of the runs—and was $n = 0$, 4, 5, and 6 on only 36, 410, 9, and 1 runs. Thus if the actual run detected no muons at all, that would be by (15.92) about a 4σ event, while a run with more than 4 muons on a single day would be an event of more than 4σ. Either would be a reason to examine the apparatus or the heavens; the Monte Carlo can't tell us which. The curve shows how many runs had a total of n muons in 100 days; 125,142 histories had 10 muons.

Of course, one could compute the data of Fig. 16.1 by hand without running a Monte Carlo. But suppose one's aging phototubes reduced the mean number of muons detected per day to $\langle n \rangle = 0.1(1 - \alpha d/100)$ on day d? Or suppose one needed the probability of detecting more than one muon

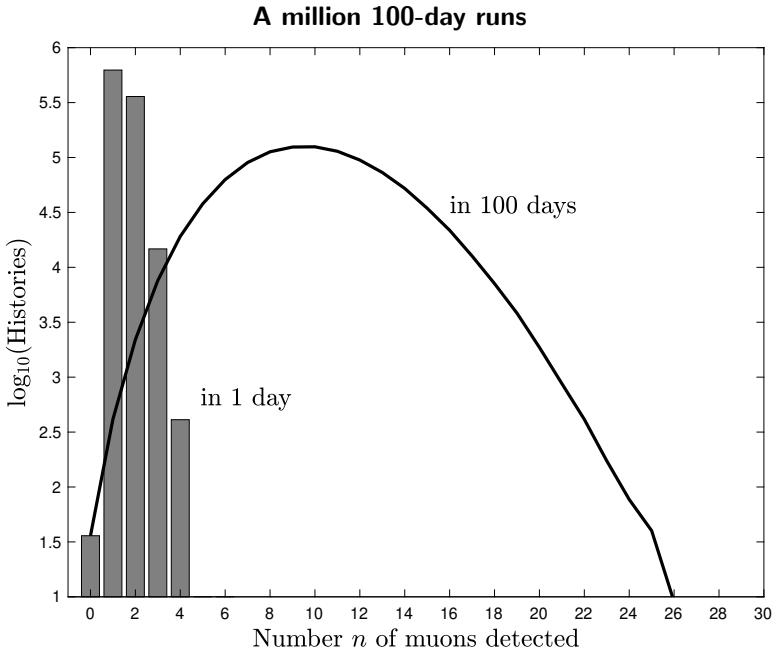

Figure 16.1 The number of histories of 100 days (out of 1,000,000 histories) in which a maximum of n muons is detected on a single day (boxes) and in 100 days (curve). A Matlab script for this figure is in Monte_Carlo_methods at `github.com/kevinecahill`.

on two days separated by one day of zero muons? In such cases, the analytic computation would be difficult and error prone, but the student would need to change only a few lines in the Monte Carlo program.

16.5 Statistical mechanics

The Metropolis algorithm can generate a sequence of states or configurations of a system distributed according to the Boltzmann probability distribution (1.392). Suppose the state of the system is described by a vector x of many components. For instance, if the system is a protein, the vector x might be the $3N$ spatial coordinates of the N atoms of the protein. A protein composed of 200 amino acids has about 4000 atoms, and so the vector x would have some 12,000 components. Suppose $E(x)$ is the energy of configuration x of the protein in its cellular environment of salty water crowded with

macromolecules. How do we generate a sequence of "native states" of the protein at temperature T?

We start with some random or artificial initial configuration x^0 and then make random changes δx in successive configurations x. One way to do this is to make a small, random change δx_i in coordinate x_i and then to test whether to accept this change by comparing the energies $E(x)$ and $E(x')$ of the two configurations x and x', which differ by δx_i in coordinate x_i. (Estimating these energies is not trivial; Gromacs and TINKER can help.)

It is important that the random changes be symmetric, that is, the probability of choosing to test whether to go from x to x' when one is at x should be equal to the probability of choosing to test whether to go from x' to x when one is at x'. A simple way to ensure this symmetry is to define x'_i in terms of x_i, a suitable step size δx, and a random number r (uniformly distributed between 0 and 1) as

$$x'_i = x_i + \left(r - \tfrac{1}{2}\right) \delta x. \tag{16.7}$$

Also, the sequences of configurations should be **ergodic**; that is, from any configuration x, one should be able to get to any other configuration x' by a suitable sequence of changes $\delta x_i = x'_i - x_i$.

How do we decide whether to accept or reject δx_i? We use the following **Metropolis step**: If the energy $E' = E(x')$ of the new configuration x' is less than the energy $E(x)$ of the current configuration x, then we accept the new configuration x'. But if $E' > E$, then we accept x' with probability

$$P(x \to x') = e^{-(E'-E)/kT} \tag{16.8}$$

by generating a random number $r \in [0, 1]$ and accepting x' if

$$r < e^{-(E'-E)/kT}. \tag{16.9}$$

If one does not accept x', then the system remains in configuration x.

In FORTRAN90, the Metropolis step might be

```
if ( newE <= oldE ) then ! accept
   x(i) = x(i) + dx
else ! accept conditionally
   call random_number(r)
   if ( r <= exp(- (newE - oldE)/(k*T)) ) then ! accept
      x(i) = x(i) + dx
   end if
end if
```

The next step is to vary another coordinate, such as x_{i+1}. Once one has varied all of the coordinates, one has finished a **sweep** through the system. After thousands or millions of such sweeps, the protein is said to be **thermalized**. Once the protein has thermalized, one can start measuring its properties, such as its shape. One computes a physical quantity every hundred or thousand sweeps and takes the average of these measurements. That average is the mean value of the physical quantity at temperature T.

Why does this work? Consider two configurations x and x' which respectively have energies $E = E(x)$ and $E' = E(x')$ and are occupied with probabilities $P_t(x)$ and $P_t(x')$ as the system is thermalizing. If $E > E'$, then the rate $R(x \to x')$ of going from x to x' is the rate v of choosing to test x' when one is at x times the probability $P_t(x)$ of being at x, that is, $R(x \to x') = v\, P_t(x)$. The reverse rate $R(x' \to x)$ is $R(x' \to x) = v\, P_t(x')\, e^{-(E-E')/kT}$ with the same v since the random walk is symmetric. The net rate from $x \to x'$ then is

$$R(x \to x') - R(x' \to x) = v \left(P_t(x) - P_t(x')\, e^{-(E-E')/kT} \right). \tag{16.10}$$

This net flow of probability from $x' \to x$ is positive if and only if

$$P_t(x)/P_t(x') > e^{-(E-E')/kT}. \tag{16.11}$$

The probability distribution $P_t(x)$ therefore flows with each sweep toward the Boltzmann distribution $\exp(-E(x)/kT)$. The flow slows and stops when the two rates are equal $R(x' \to x) = R(x \to x')$ a condition called **detailed balance**. At this equilibrium, the distribution $P_t(x)$ satisfies

$$P_t(x) = P_t(x')\, e^{-(E-E')/kT} \tag{16.12}$$

in which $P_t(x')\, e^{E'/kT}$ is independent of x. So the thermalizing distribution $P_t(x)$ approaches the distribution $P(x) = c\, e^{-E/kT}$ in which c is independent of x. Since the sum of these probabilities must be unity, we have

$$\sum_x P(x) = c \sum_x e^{-E(x)/kT} = 1 \tag{16.13}$$

which means that the constant c is the inverse of the **partition function**

$$Z(T) = \sum_x e^{-E(x)/kT}. \tag{16.14}$$

The thermalizing distribution approaches Boltzmann's distribution (1.392)

$$P_t(x) \to P_B(x) = e^{-E(x)/kT}/Z(T). \tag{16.15}$$

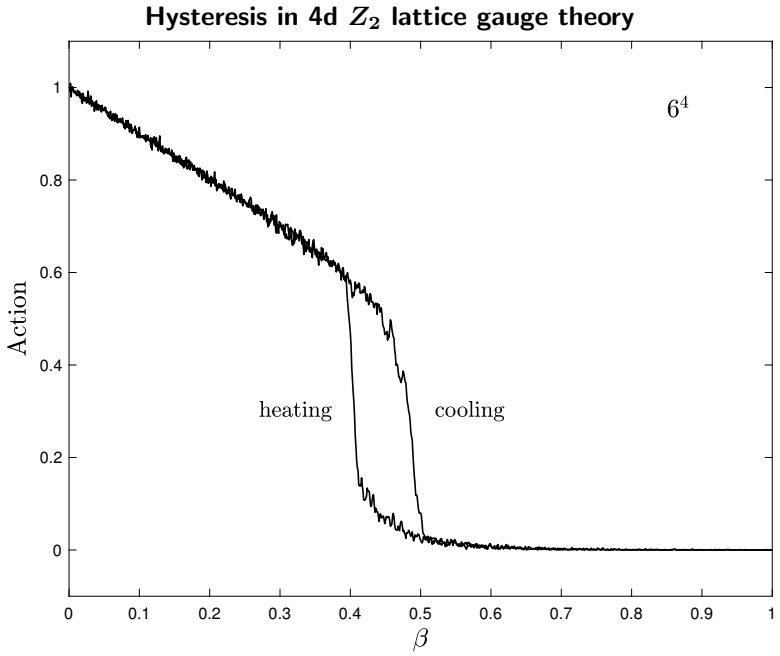

Figure 16.2 The hysteresis loop is a sign of a first-order phase transition.

Example 16.2 (Z_2 lattice gauge theory) To simulate Z_2 gauge theory on a lattice, one represents spacetime as a lattice of points in d dimensions. Two nearest-neighbor points are separated by the lattice spacing a and joined by a link. One puts an element $U = \pm 1$ of the group Z_2 on each link. One then assigns an action $S_\square$ to each elementary square or *plaquette* of the lattice. For the Z_2 gauge group (example 11.6), the action $S_\square$ of a square with vertices 1, 2, 3, and 4 is

$$S_\square = 1 - U_{1,2}\, U_{2,3}\, U_{3,4}\, U_{4,1} \tag{16.16}$$

where each $U = \pm 1$. Then, one replaces $E(x)/kT$ with βS in which the action S is a sum of all the plaquette actions S_p.

You can study Z_2 lattice gauge theory by using the program puregauge.cc available at Michael Creutz's website (`latticeguy.net/lattice.html`). By running it on a 6^4 lattice from low temperature $\beta = 1$ to high temperature $\beta = 0$ and back again, you can exhibit hysteresis as in Fig. 16.2. □

Example 16.3 ($SU(3)$ lattice gauge theory) For each elementary square of the lattice, the plaquette variable U_p is the product of elements U of the gauge group $SU(3)$ around the square, $U_p = U_{1,2}\, U_{2,3}\, U_{3,4}\, U_{4,1}$. The euclidian action of the theory is then the sum over all the plaquettes of the lattice of

the traces

$$S = \beta \sum_p \left[1 - \frac{1}{6} \mathrm{Tr} \left(U_p + U_p^\dagger \right) \right] \tag{16.17}$$

in which $\beta = 6/g^2$ is inversely proportional to the coupling constant g. □

Although the generation of configurations distributed according to the Boltzmann probability distribution (1.392) is one of its most useful applications, the Monte Carlo method is much more general. It can generate configurations x distributed according to any probability distribution $P(x)$.

To generate configurations distributed according to $P(x)$, we accept any new configuration x' if $P(x') > P(x)$ and also accept x' with probability

$$P(x \to x') = P(x')/P(x) \tag{16.18}$$

if $P(x) > P(x')$.

This works for the same reason that the Boltzmann version works. Consider two configurations x and x'. After the system has thermalized, the probabilities $P_t(x)$ and $P_t(x')$ have reached equilibrium, and so the rate $R(x \to x')$ from $x \to x'$ must equal the rate $R(x' \to x)$ from $x' \to x$. If $P(x') > P(x)$, then $R(x \to x')$ is

$$R(x \to x') = v\, P_t(x) \tag{16.19}$$

in which v is the rate of choosing $\delta x = x' - x$, while the rate $R(x' \to x)$ is

$$R(x' \to x) = v\, P_t(x')\, P(x)/P(x') \tag{16.20}$$

with the same v since the random walk is symmetric. Equating the two rates $R(x \to x') = R(x' \to x)$, we find that after thermalization

$$P_t(x) = P(x)\, P_t(x')/P(x') = c\, P(x) \tag{16.21}$$

in which c is independent of x. Thus $P_t(x)$ converges to $P(x)$ at equilibrium.

So far we have assumed that the rate of choosing $x \to x'$ is the same as the rate of choosing $x' \to x$. In **Smart Monte Carlo** schemes, physicists arrange the rates $v_{x \to x'}$ and $v_{x' \to x}$ so as to steer the flow and speed-up thermalization. To compensate for this asymmetry, they change the second part of the Metropolis step from $x \to x'$ when $P(x) > P(x')$ to accept conditionally with probability

$$P(x \to x') = P(x')\, v_{x' \to x} / \left[P(x)\, v_{x \to x'} \right]. \tag{16.22}$$

Now if $P(x) > P(x')$, then $R(x' \to x)$ is

$$R(x' \to x) = v_{x' \to x}\, P_t(x') \tag{16.23}$$

while the rate $R(x \to x')$ is

$$R(x \to x') = v_{x \to x'}\, P_t(x)\, P(x')\, v_{x' \to x} / \left[P(x)\, v_{x \to x'}\right]. \tag{16.24}$$

Equating the two rates $R(x' \to x) = R(x \to x')$, we find

$$P_t(x) = P(x)\, P_t(x')/P(x') \tag{16.25}$$

which implies that $P_t(x)$ converges to

$$P_t(x) = P(x)/Z \tag{16.26}$$

in which $Z = \int P(x)\, dx$.

Example 16.4 (Highly multiple integration) You can use the general Metropolis method (16.18–16.21) to integrate a function $f(x)$ of many variables $x = (x_1, \ldots, x_n)$ if you can find a positive function $g(x)$ similar to $f(x)$ whose integral $I[g]$ you know. You just use the probability distribution $P(x) = g(x)/I[g]$ to find the mean value of $I[g]f(x)/g(x)$:

$$\int f(x)\, d^n x = I[g] \int \frac{f(x)}{g(x)}\, \frac{g(x)}{I[g]}\, d^n x = I[g] \int \frac{f(x)}{g(x)}\, P(x)\, d^n x. \tag{16.27}$$

□

16.6 Simulated annealing

One can use the Monte Carlo method to find the absolute minimum (or maximum) of a function $E(x)$ of many variables $x = (x_1, x_2, \ldots, x_N)$. To avoid being trapped in a local minimum, one starts with a sequence of Metropolis steps (16.8–16.9) at a high value of kT and then gradually lowers the value of kT to zero.

16.7 Solving Arbitrary Problems

If you know how to generate a suitably large space of trial solutions to a problem, and you also know how to compare the quality of any two of your solutions, then you can use a Monte Carlo method to solve the problem. The hard parts of this seemingly magical method are characterizing a big enough space of solutions s and constructing a quality function or functional that

assigns a number $Q(s)$ to every solution in such a way that if s is a better solution than s', then

$$Q(s) > Q(s'). \tag{16.28}$$

But once one has characterized the space of possible solutions s and has constructed the quality function $Q(s)$, then one simply generates zillions of random solutions and selects the one that maximizes the function $Q(s)$ over the space of all solutions.

If one can characterize the solutions as vectors of a certain dimension, $s = (x_1, \ldots, x_n)$, then one may use the Monte Carlo method of the previous section (16.5) by setting $P(s) = Q(s)$.

16.8 Evolution

The reader may think that the use of Monte Carlo methods to solve arbitrary problems is quite a stretch. Yet nature has applied them to the problem of evolving species that survive. As a measure of the quality $Q(s)$ of a given solution s, nature used the time derivative of the logarithm of its population $\dot{P}(t)/P(t)$. The space of solutions is the set of possible genomes. Leaving aside DNA methylation, histone acetylation, and other epigenetic changes, we may idealize each solution or genome as a sequence of nucleotides $s = b_1 b_2 \ldots b_N$ some thousands or billions of bases long, each base b_k being adenine, cytosine, guanine, or thymine (A, C, G, or T). Since there are four choices for each base, the set of solutions is huge. The genome for *homo sapiens* has some 3 billion bases (or base pairs, DNA being double stranded), and so the solution space is a set with

$$\mathcal{N} = 4^{3\times 10^9} = 10^{1.8\times 10^9} \tag{16.29}$$

elements. By comparison, a googol is only 10^{100}.

In evolution, a Metropolis step begins with a random change in the sequence of bases; changes in a germ-line cell can change a new individual. Some of these changes are due to errors in the normal mechanisms by which genomes are copied and repaired; the holoenzyme DNA polymerase copies DNA with remarkable fidelity, but it makes one error per billion base pairs. Sexual reproduction makes bigger random changes in genomes. In **meiosis**, the paternal and maternal versions of each of our 23 chromosomes are duplicated, and the four versions swap segments of DNA in a process called genetic recombination or crossing-over. The cell then divides twice producing four **haploid** germ cells each with a single paternal, maternal, or mixed version of each chromosome. Two haploid cells, one from each parent, join to start

a new individual. Sexual reproduction makes evolution more ergodic, which is why most complex modern organisms use it.

The second part of the evolutionary Metropolis step is done by the newly born individual: if he or she survives and multiplies, the change is accepted; if he or she dies without progeny, it is rejected. In 4 billion years, evolution has turned simple molecules into human beings.

John Holland and others have used analogs of these Metropolis steps to write **genetic algorithms** that can solve wide classes of problems (Holland, 1975; Vose, 1999; Schmitt, 2001).

Further reading

The classic *Quarks, Gluons, and Lattices* (Creutz, 1983) is a marvelous introduction to the subject; his website (latticeguy.net/lattice.html) is an extraordinary resource, as is Rubinstein's *Simulation and the Monte Carlo Method* (Rubinstein and Kroese, 2007). *Molecular Biology of the Cell* (Alberts et al., 2014) is one of the best textbooks ever written.

Exercises

16.1 Go to Michael Creutz's website `latticeguy.net/lattice.html` and get his C-code for Z_2 lattice gauge theory. Compile and run it, and make a graph like Fig. 16.2 which exhibits hysteresis.

16.2 Modify his code and produce a graph showing the coexistence of two phases at the critical coupling $\beta_t = 0.5 \ln(1+\sqrt{2})$. Hint: Do a cold start and then 100 updates at β_t, then do a random start and do 100 updates at β_t. Plot the values of the action against the update number 1, 2, 3, ... 100.

16.3 Modify Creutz's C code for Z_2 lattice gauge theory so as to be able to vary the dimension d of spacetime. Show that for $d = 2$, there's no hysteresis loop (there's no phase transition). For $d = 3$, show that any hysteresis loop is minimal (there's a second-order phase transition).

16.4 What happens when $d = 5$?

16.5 Use example 16.4 to compute the ten-dimensional integral

$$\mathcal{I} = \int \exp\left[-\left(x^2 + (x^2)^2\right)\right] d^{10}x \tag{16.30}$$

over $\mathbb{R}^{10}$ where $x^2 = x_1^2 + \ldots + x_{10}^2$.