Monte Carlo Lecture Notes I, Random Number Generators

Jonathan Goodman *
Courant Institute of Mathematical Sciences, NYU

January 29, 1997

1 Introduction to Monte Carlo Methods

A Monte Carlo method is a numerical procedure that deliberately uses random numbers (or numbers that are supposed to mimic random numbers) to compute something. Usually, the quantity being computed is itself not random, although it may be defined in terms of random numbers, for example, the expectation of a random variable. Equally common is the introduction of random numbers to solve a problem that did not have randomness in its original definition, such as the ground state energy of an atom described by the Schrödinger equation. I would not use the term Monte Carlo if you don't want a specific number but just want to simulate a random process (e.g. to make a random map for a computer game).

Usually, Monte Carlo methods are used to overcome the "curse of dimensionality" that prohibits the use of "deterministic" methods. For example, suppose you want to compute

$$\int_{R^d} f(x)dx$$

*goodman@cims.nyu.edu, or http://www.math.nyu.edu/faculty/goodman,

where $x = \left(x^1, \ldots, x^d\right)$, and the dimension of integration, d, is large. Such integrals arise in many fields with d in the hundreds or millions or more. A quadrature formula would approximate the integral by a sum

$$\int_{R^d} f(x)dx \approx \Delta x^d \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_d=1}^{n} f(x^1_{i_1}, \ldots, x^d_{i_d}) \ .$$

If we use n quadrature points in each coordinate direction, the total number of points is n^d. For $n = 10$ and $d = 30$, this is infeasible on present or likely future computers. All grid based methods have the property that the work scales exponentially with the dimension.

Suppose, on the other hand that you can write $f(x) = g(x)\rho(x)$ where $\rho(x)$ is a probability density function for a d dimensional random variable. Suppose also that you can make many independent samples, X_k, from this probability density. Then you can make the Monte Carlo approximation

$$\int f(x)dx = \int g(x)\rho(x)dx = E_\rho\left[g(X)\right] \approx \frac{1}{n}\sum_{k=1}^{n} g(X_k) \ .$$

The error in this approximation usually is on the order of $n^{-1/2}$, which is not great; but it is much better, for large d, than the quadrature formula. In many cases, there are Monte Carlo methods with the property that the work needed to reach a specified accuracy, ϵ, grows polynomially in $1/\epsilon$ and d.

2 Pseudo Random Number Generators

Almost all random quantities in Monte Carlo algorithms are in the end made by manipulating hypothetical independent uniformly distributed random variables, ξ. That is, we assume that we have random variables, ξ_1, ξ_2, ..., that are independent and all have probability density $\rho_u(x) = 1$ if $0 \leq x \leq 1$, and $\rho_u(x) = 0$ otherwise. For most of the course, we will just assume that such ξ can be made and not worry about how it's done. In fact, it is impossible to do it exactly. Any deterministic computer algorithm will produce numbers that are correlated with each other to some extent. There are stories about Monte Carlo computations that got the wrong answer because of correlations in the pseudo random numbers. The best random number generators have correlations that are almost impossible to detect and make almost any Monte Carlo method work correctly, as if the numbers were truly random.

Almost all pseudo random number generators in use today work in the following way. There is a "seed", $s = (s^1, \ldots, s^m)$ that consists of m 32 bit computer integers. There is a mapping, $s' = F(s)$, that produces a new seed from a given one and a mapping, $\xi = G(s)$, that produces a number in the interval $[0, 1]$ from a seed. The mappings, F, and G, are usually given by some modular arithmetic. To use such a random number generator, you specify an initial seed, s_1, and then produce the sequence ξ_k by repeatedly using F and G:

$$\xi_k = G(s_k) \ , \qquad s_{k+1} = F(s_k) \ .$$

In C, this would be done[1] (with $m = 2$), by `RanSet(int s1, int s2)` to set the seed, and `xi = Rand()` to get a new uniform random variable independent of the previous ones. The mappings F and G are carried out by the procedure `Rand()`. There probably will be a routine `RanGet(int *s1, int *s2)` that sets `s1` and `s2` to be the current seed. This allows programmers to do a long Monte Carlo run in several batches.

To use a pseudo random number generator, it is important to remember to set the seed in the beginning and not to set it again. If you do two runs with the same initial seed and the same code, you will get exactly the same result. (Pseudo random numbers are not really "random".) This can be very helpful in debugging computer programs. A pseudo random number generator with $m = 1$ (a 32 bit generator) should not be used because it must cycle (produce the same numbers in the same order again) in no more than $k = 2^{32} = 4$ billion steps. Monte Carlo computations often use more than this.

[1] The actual procedure and variable names will depend on the random number generator you use.