

Estimation of a Definite Integral

Monte Carlo inference, as for statistical inference generally, can be formulated as estimation of either a definite integral

$$\theta = \int_D f(x)dx$$

or, given the integral θ , of a domain D , or of a function f that satisfies certain optimality conditions.

If the integral can be evaluated in closed form, there is no need for Monte Carlo methods.

Function Decomposition

If the function f is decomposed to have a factor that is a probability density function, say

$$f(x) = g(x)p(x),$$

where

$$\int_D p(x)dx = 1$$

and $p(x) \geq 0$, then the integral θ is the expectation of the function g of the random variable with probability density p ; that is,

$$\theta = E(g(X)) = \int_D g(x)p(x)dx.$$

With a random sample x_1, \dots, x_m from the distribution with probability density p , an estimate of θ is

$$\hat{\theta} = \frac{\sum g(x_i)}{m}.$$

Monte Carlo Estimation

We use this technique in many settings in statistics. There are three steps:

1. decompose the function of interest to include a probability density function as a factor;
2. identify an expected value;
3. use a sample (simulated or otherwise) to estimate the expected value.

Random Numbers

Where can you get “random numbers”?

Start with any integer x_0 (a “random” one) and then let

$$x_i \equiv \alpha x_{i-1} \pmod{m},$$

with

$$0 \leq x_i \leq m - 1.$$

Then take

$$u_i = x_i/m.$$

If α and m are properly chosen, the u_i 's will “look like” they are randomly and uniformly distributed between 0 and 1.

Congruential generator.

Note that the congruential method is

1. deterministic (hence, at best “pseudo-random”) and
2. cyclic (will repeat after at most $m - 1$ iterations).

The length of the cycle is called the period.

There are many ways of testing generators for good distributional properties, but, unfortunately, bad generators often pass tests.

For any deterministic generator, a test could be constructed that it would fail.

Other Examples of Uses of Random Numbers

- Provide a random deal of 4 bridge hands from a standard deck.
- Provide a simple random sample of size 100 from a file of size 10,000.

The sample sizes are $\sim 10^{28}$ and $\sim 10^{243}$.

Commonly used generators will yield about $\sim 2 \times 10^9$ different 'random numbers'.

Feedback Shift Register Methods

Feedback shift register methods and generalized feedback shift register (GFSR) methods are based on a sequence

$$a_k \equiv (c_p a_{k-p} + c_{p-1} a_{k-p+1} + \cdots + c_1 a_{k-1}) \pmod{2}.$$

Successive p -tuples (or in a 'generalized' form, l -tuples formed from subsequences) are taken as the binary representation of the random numbers.

In a particular case of interest, $p = 521$, with $c_{521} = c_{32} = 1$ and all other c s equal to zero. The period is 2^{521} .

Other Uniform Generators

There are several other types of uniform generators.

- Lagged Fibonacci:

$$x_i \equiv (x_{i-j} + x_{i-k}) \pmod{m}.$$

Requires an initial sequence, rather than just a single seed. If j , k , and m are chosen properly, the lagged Fibonacci generator can perform well. If m is a prime and $k > j$, the period can be as large as $m^k - 1$.

$$\begin{aligned}j &= 24 \\k &= 55 \\m &= 2^{32}\end{aligned}$$

Period of about 10^{25}

Combining Generators

The fractional part of the sum of independent random variables one of which is $U(0, 1)$ is also a $U(0, 1)$ random variable.

For any independent Y_1 and Y_2 over $[0, 1]$, and $W = Y_1 + Y_2$, $p(w) = \int_{-\infty}^{\infty} f_2(w - y_1) f_1(y_1) dy_1$.

So we can show $X = (Y_1 + Y_2) \bmod 1$ has density

$$p(x) = \int_0^x f_2(x - y_1) f_1(y_1) dy_1 + \int_x^1 f_2(1 + x - y_1) f_1(y_1) dy_1$$

More generally, if X_i are independently distributed with PDF's p_i , and $|p_i(x) - 1| \leq \epsilon_i$ over $[0, 1]$, then for the PDF of the sum, p ,

$$|p(x) - 1| \leq \prod \epsilon_i$$

over $[0, 1]$.

This allows a simple method of combining generators.

Another Approach: Quasirandom Numbers

A Halton sequence is formed by reversing the digits in the representation of some sequence of integers in a given base.

Although this can be done somewhat arbitrarily, a straightforward way of forming a d -dimensional Halton sequence x_1, x_2, \dots , where $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ is first to choose d bases, b_1, b_2, \dots, b_d , perhaps the first d primes. The j^{th} base will be used to form the j^{th} component of each vector in the sequence.

Then begin with some integer m and

1. choosing t_{mj} suitably large, represent m in each base:

$$m = \sum_{k=0}^{t_{mj}} a_{mk} b_j^k, \quad j = 1, \dots, d,$$

2. form

$$x_{ij} = \sum_{k=0}^{t_{mj}} a_{mk} b_j^{k-t_{mj}-1}, \quad j = 1, \dots, d,$$

3. set $m = m + 1$ and repeat.

More on Halton Sequences

Suppose, for example, $d = 3$, $m = 15$, and we use the bases 2, 3, and 5. We form $15 = 1111_2$, $15 = 120_3$, and $15 = 30_5$, and deliver the first x as $(0.1111_2, 0.021_3, 0.03_5)$, or $(0.937500, 0.259259, 0.120000)$.

The Halton sequences are acceptably uniform for lower dimensions, up to about 10. For higher dimensions, however, the quality of the Halton sequences degrades rapidly because the two-dimensional planes occur in cycles with decreasing periods. Generalized Halton sequences have been proposed and studied. The basic idea is to permute the a_{mk} 's in step 2.

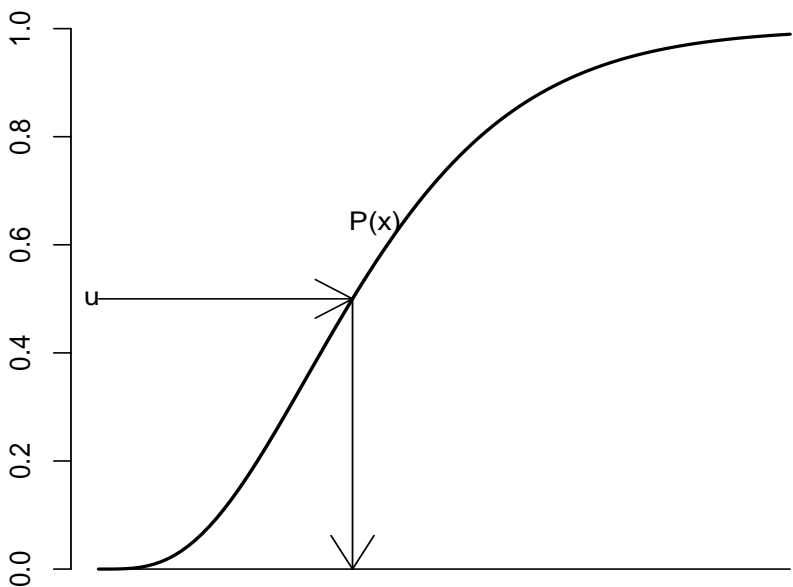
In a “leaped Halton sequence” the cycles of the Halton sequence are destroyed by using only every l^{th} Halton number, where l is a prime different from all of the bases b_1, b_2, \dots, b_d .

Generation of Non-Uniform Random Numbers

The inverse CDF method.

$$U = P_X(X)$$

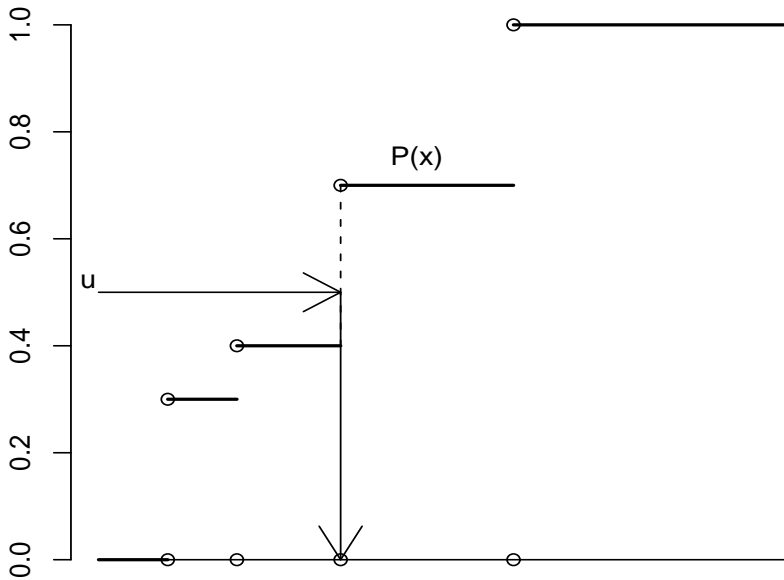
$$X = P_X^{-1}(U)$$



For discrete distributions,

$$U = P_X(X)$$

$$X = \max_x \{x | U \leq P_X(x)\}$$



Generation of Non-Uniform Random Numbers

The acceptance/rejection method.

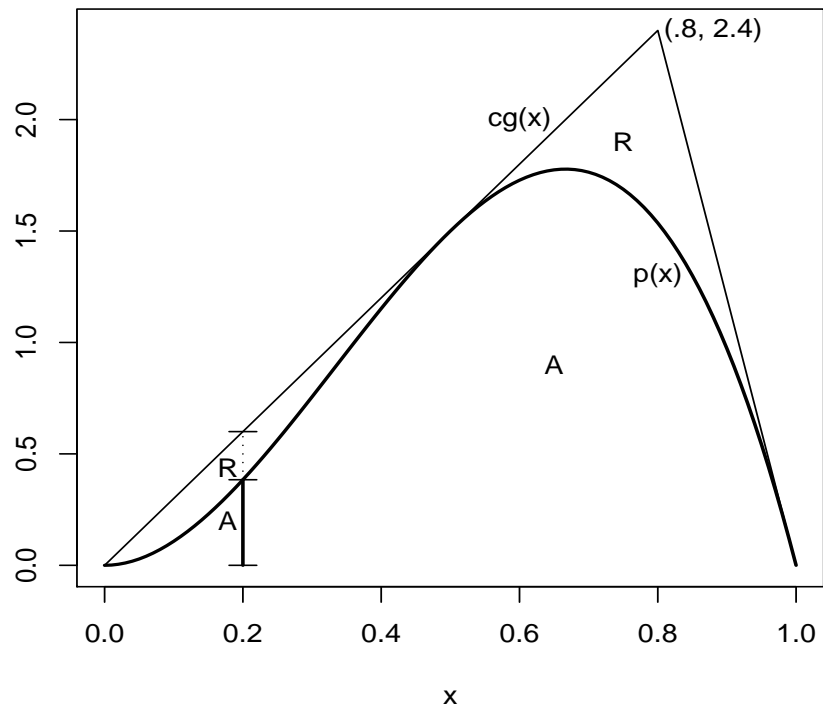
Suppose we wish to generate a random variate from a distribution that has a probability density $p(\cdot)$ and suppose that we already can generate random variates from a distribution that has a probability density $g(\cdot)$ (it might be the uniform distribution, for example).

Further, suppose that $h(x) = cg(x)$ is such that $h(x) \geq p(x)$.

1. Generate a variate y from the distribution having pdf g .
2. Generate independently a variate u from the uniform $(0,1)$ distribution.
3. If $u \leq p(y)/h(y)$, then accept y as the variate, otherwise, reject y and return to step 1.

See figure.

How It Works



Why It Works

Let Z be the random variable delivered. For any x , because Y (from the density g) and U are independent, we have

$$\begin{aligned}\Pr(Z \leq x) &= \Pr\left(Y \leq x \mid U \leq \frac{p(Y)}{cg(Y)}\right) \\ &= \frac{\int_{-\infty}^x \int_0^{p(t)/cg(t)} g(t) \, ds \, dt}{\int_{-\infty}^{\infty} \int_0^{p(t)/cg(t)} g(t) \, ds \, dt} \\ &= \int_{-\infty}^x p(t) \, dt,\end{aligned}$$

the distribution function corresponding to p . Differentiating this quantity with respect to x yields $p(x)$.

The acceptance/rejection method can be visualized as choosing a subsequence from a sequence of independently and identically distributed (i.i.d.) realizations from the distribution with density g_Y in such a way the subsequence has density p_X .

i.i.d. from g_Y	y_i	y_{i+1}	y_{i+2}	y_{i+3}	\cdots	y_{i+k}	\cdots
accept?	no	yes	no	yes	\cdots	yes	\cdots
i.i.d. from p_X	x_j		x_{j+1}		\cdots	x_{j+l}	\cdots

Obviously, the closer $cg(x)$ is to $p(x)$, the faster the acceptance/rejection algorithm will be, if we ignore the time required to generate y from the dominating density g . A good majorizing function would be such that the l is almost as large as k .

Often, g is chosen to be a very simple density, such as a uniform or a triangular density. When the dominating density is uniform, the acceptance/rejection method is similar to the “hit-or-miss” method.

Variations of Acceptance/Rejection

There are many variations of the basic acceptance/rejection.

One is called *transformed rejection*. In the transformed acceptance/rejection method, the steps of the algorithm are combined and rearranged slightly. Let G be the CDF corresponding to the dominating density g . Let $H(x) = G^{-1}(x)$, and let $h(x) = dH(x)/dx$. If v is a uniform (0,1) deviate, step 1 in the original method is equivalent to $y = H(v)$, so we have the following.

1. Generate u and v independently from a uniform (0,1) distribution.
2. If $u \leq p(H(v))h(v)/c$, then
 - 2.a. take $H(v)$ as the desired realization;otherwise
 - 2.b. return to step 1.

■

Acceptance/Rejection for Discrete Distributions

There are various ways that acceptance/rejection can be used for discrete distributions. One advantage of these methods is that they can be easily adapted to changes in the distribution.

Consider the discrete random variable X_s such that

$$\begin{aligned}\Pr(X_s = x_i) &= p_{si} \\ &= \frac{a_{si}}{a_{s1} + a_{s2} + \cdots + a_{sk}}, \quad i = 1, \dots, k.\end{aligned}$$

(If $\sum_{i=1}^k a_{si} = 1$, the numerator a_{si} is the ordinary probability p_{si} at the mass point i .) Suppose that there exists an a_i^* such that $a_i^* \leq a_{si}$ for $s = 1, 2, \dots$, and $b > 0$ such that $\sum_{i=1}^k a_{si} \geq b$ for $s = 1, 2, \dots$. Let

$$a^* = \max\{a_i^*\},$$

and let

$$P_{si} = a_{si}/a^* \quad \text{for } i = 1, \dots, k.$$

Acceptance/Rejection for Discrete Distributions

1. Generate u from a uniform $(0,1)$ distribution and let $i = \lceil ku \rceil$.
2. Let $r = i - ku$.
3. If $r \leq P_{si}$, then
 - 3.a. take i as the desired realization;otherwise
 - 3.b. return to step 1. ■

Suppose for the random variable X_{s+1} , $p_{s+1,i} \neq p_{si}$ for some i . (Of course, if this is the case for mass point i , it is also necessarily the case for some other mass point.) For each mass point whose probability changes, reset $P_{s+1,i}$ to $a_{s+1,i}/a^*$ and continue.

Acceptance/Rejection for Multivariate Distributions

The acceptance/rejection method is one of the most widely applicable methods for random number generation. It is used in many different forms, often in combination with other methods.

It is clear from the description of the algorithm that the acceptance/rejection method applies equally to multivariate distributions. (The uniform random number is still univariate, of course.)

Dependent Random Variables

The methods described before use a sequence of i.i.d. variates from the majorizing density. It is also possible to use a sequence from a conditional majorizing density. A method using a non-independent sequence is called a Metropolis method, and there are variations of these, with their own names.

Suppose for a collection of random variables X_1, X_2, \dots, X_k , we know the conditional distributions of $X_i|X_j (i \neq j)$. Can we simulate the collection?

Alternatively, suppose we know the functional form (up to the normalizing constant) of the joint density of X_1, X_2, \dots, X_k , and that we know the distribution of at least one $X_i|X_j (i \neq j)$. Can we simulate the collection?

Markov Chain Monte Carlo

If the density of interest, p , is the density of the stationary distribution of a Markov chain, correlated samples from the distribution can be generated by simulating the Markov chain.

This appears harder than it is.

An aperiodic, irreducible, positive recurrent Markov chain is associated with a *stationary distribution* or *invariant distribution*, which is the limiting distribution of the chain.

Convergence?

An algorithm based on a stationary distribution of a Markov chain is an *iterative method* because a sequence of operations must be performed until they *converge*.

Several schemes for assessing convergence have been proposed. Gelman and Rubin (1992), for example, suggest multiple starting points and then an ANOVA-type test comparing variances within and across the multiple streams.

A Markov chain is the basis for several schemes for generating random numbers. The interest is not in the sequence of the Markov chain itself. The elements of the chain are accepted or rejected in such a way as to form a different chain whose stationary distribution is the distribution of interest.

The Metropolis Algorithm

For a distribution with density p , the Metropolis algorithm, introduced by Metropolis et al. (1953) generates a random walk and performs an acceptance/rejection based on p evaluated at successive steps in the walk.

In the simplest version, the walk moves from the point y_i to a candidate point $y_{i+1} = y_i + s$, where s is a realization from $U(-a, a)$, and accepts y_{i+1} if

$$\frac{p(y_{i+1})}{p(y_i)} \geq u,$$

where u is an independent realization from $U(0, 1)$.

This method is also called the “heat bath” method because of the context in which it was introduced. The random walk of Metropolis et al. is the basic algorithm of *simulated annealing*, which is currently widely used in optimization problems.

A histogram is not affected by the sequence of the output in a large sample.

The von Mises distribution is an easy one to simulate by the Metropolis algorithm. This distribution is often used by physicists in simulations of lattice gauge and spin models, and the Metropolis method is widely used in these simulations.

Notice the simplicity of the algorithm: we did not need to determine a majorizing density, nor even evaluate the Bessel function that is the normalizing constant for the von Mises density.

random walk	y_i	$y_{i+1} =$	$y_{i+3} =$	$y_{i+2} =$	
		$y_i + s_{i+1}$	$y_{i+1} + s_{i+2}$	$y_{i+2} + s_{i+3}$	\dots
accept?	no	yes	no	yes	\dots
i.i.d. from p_X		x_j		x_{j+1}	\dots

The Markov chain samplers require a “burn-in” period, that is, a number of iterations before the stationary distribution is achieved. In practice, the variates generated during the burn-in period are discarded. The number of iterations needed varies with the distribution, and can be quite large, sometimes several hundred. The von Mises example is unusual; no burn-in is required. In general, convergence is much quicker for univariate distributions with finite ranges such as this one.

It is important to remember what convergence means; it does *not* mean that the sequence is independent from the point of convergence forward. The deviates are still from a Markov chain.

The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm uses a more general chain for the acceptance/rejection step.

To generate deviates from a distribution with density p_X it uses deviates from a Markov chain with density $g_{Y_{t+1}|Y_t}$. The conditional density $g_{Y_{t+1}|Y_t}$ is chosen so that it is easy to generate deviates from it.

The Metropolis-Hastings Algorithm

0. Set $k = 0$.
1. Choose $x^{(k)}$ in the range of p_X . (The choice can be arbitrary.)
2. Generate y from the density $g_{Y_{t+1}|Y_t}(y|x^{(k)})$.
3. Set r :

$$r = p_X(y) \frac{g_{Y_{t+1}|Y_t}(x^{(k)}|y)}{p_X(x^{(k)})g_{Y_{t+1}|Y_t}(y|x^{(k)})}$$

4. If $r \geq 1$, then
 - 4.a. set $x^{(k+1)} = y$;otherwise
 - 4.b. generate u from uniform(0,1) and
 - if $u < r$, then
 - 4.b.i. set $x^{(k+1)} = y$,
 - otherwise
 - 4.b.ii. set $x^{(k+1)} = x^{(k)}$.

5. If convergence has occurred, then

5.a. deliver $x = x^{(k+1)}$;

otherwise

5.b. set $k = k + 1$, and go to step 2.

The idea is similar to the basic acceptance/rejection method.

The analogue to the majorizing function in the Metropolis-Hastings algorithm is

$$\frac{g_{Y_{t+1}|Y_t}(x|y)}{p_X(x) g_{Y_{t+1}|Y_t}(y|x)}.$$

As with the acceptance/rejection methods with independent sequences, the acceptance/rejection methods based on Markov chains apply immediately to multivariate random variables.

We can see that this algorithm delivers realizations from the density p_X by determining the CDF and differentiating.

The CDF is the probability-weighted sum of the two components corresponding to whether the chain moved or not. In the case in which the chain does move, that is, in the case of acceptance, for the random variable Z whose realization is y , we have

$$\begin{aligned}
 \Pr(Z \leq x) &= \Pr\left(Y \leq x \mid U \leq p(Y) \frac{g(x_i|Y)}{p(x_i)g(Y|x_i)}\right) \\
 &= \frac{\int_{-\infty}^x \int_0^{p(t)g(x_i|t)/(p(x_i)g(t|x_i))} g(t|x_i) \, ds \, dt}{\int_{-\infty}^{\infty} \int_0^{p(t)g(x_i|t)/(p(x_i)g(t|x_i))} g(t|x_i) \, ds \, dt} \\
 &= \int_{-\infty}^x p_X(t) \, dt.
 \end{aligned}$$

Gibbs Sampling

Gibbs sampling is often useful in higher dimensions. It depends on the convergence of a Markov chain to its stationary distribution, so a burn-in period is required.

0. Set $k = 0$.
1. Choose $x^{(k)} \in S$.
2. Generate $x_1^{(k+1)}$ conditionally on $x_2^{(k)}, x_3^{(k)}, \dots, x_d^{(k)}$,
Generate $x_2^{(k+1)}$ conditionally on $x_1^{(k+1)}, x_3^{(k)}, \dots, x_d^{(k)}$,
...
Generate $x_{d-1}^{(k+1)}$ conditionally on $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_d^{(k)}$,
Generate $x_d^{(k+1)}$ conditionally on $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{d-1}^{(k+1)}$.
3. If convergence has occurred, then
 - 3.a. deliver $x = x^{(k+1)}$;otherwise
 - 3.b. set $k = k + 1$, and go to step 2.