Computational Statistics

Lecture 8: Accept-Reject Methods

Raymond Bisdorff

University of Luxembourg

December 13, 2019

Content of Lecture

Principles of Monte Carlo integration

The generic problem of Monte Carlo Integration (MCI) consists in evaluating the following integral:

$$E_f[h(X)] = \int_{S_x} h(x)f(x)dx, \qquad (*)$$

where S_x denotes the set where the random variable X takes its value, which is usually equal to the support of the density f. The principle of MCI method for approximating Integral (*) is to generate a sample $(X_1.X_2, ..., X_n)$ from the density f and propose as an approximation for $E_f[h(X)]$ the empirical average $\overline{h_n}$ as follows:

$$\overline{h_n} = \frac{1}{n}\sum_{j=1}^{n} h(x_j). \qquad (**)$$

By the Strong Law of Large Numbers, $\overline{h_n}$ converges indeed to $E_f[h(X)]$.

Monte Carlo Integration – continue

When $h(X)$ has a finite expectation under f, the convergence takes place at a speed $O(\sqrt{n})$ and the asymptotic variance of the approximation (**) is

$$var(\overline{h_n}) = \frac{1}{n}\int_{\mathcal{X}} (h(x) - E_f[h(X)])^2 f(x)dx,$$

which can be estimated from the sample $(X_1.X_2, ..., X_n)$ through

$$v_n = \frac{1}{n^2}\sum_{j=1}^{n}[h(x) - \overline{h_n}]^2.$$

Due to the CLT, for large n,

$$\frac{\overline{h_n} - E_f[h(X)]}{\sqrt{v_n}} \rightsquigarrow \mathcal{N}(0,1).$$

MCI application

MCI of $h(x) = \left(\cos(50x) + \sin(20x)\right)^2$ over the interval $[0,1]$ may be achieved with a sample $(U_1, ..., U_n)$ of 10^4 i.i.d $\mathcal{U}(0,1)$ random variables. We approximate $\int h(x)dx$ with $\sum h(U_i)/n$.

Example R session:

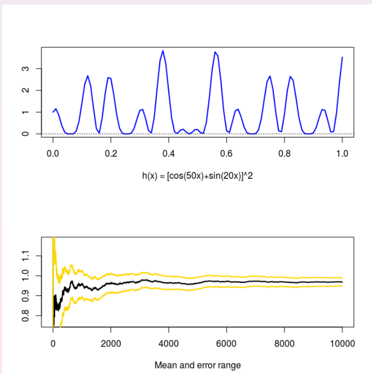
```
> h = function(x){
+         ( cos(50*x) + sin(20*x) )^2 }
> integrate(h,0,1)
0.9652009 with |error| < 1.9e-10
> n = 10^4
> x = runif(n)
> hx = h(x)
> estint=cumsum(hx)/(1:n)
> estint[n]
[1] 0.9681744
> esterr=sqrt(cumsum(
+          (hx-estint)^2) / (1:n)^2 )
> esterr[n]
[1] 0.01044141
```


MCI application – continue

The upper panel in the figure below shows the function $h(x)$ over the domain $[0,1]$. The lower panel shows the running means with bounds of $2\times$ the estimated standard error depending on the sample size $n = 10^4$.

Example R session:

```
> par(mfrow=c(2,1))
> curve(h,0,1,xlab="h(x) =
+ [cos(50x)+sin(20x)]^2",ylab="",
+ lwd=2,col="blue")
> abline(h=0,lty=3)
> plot(estint,
+  xlab="Mean and error range",
+  type="l",lwd=2,,ylab="",
+  ylim=mean(hx)+
+        20*c(-esterr[n],esterr[n]))
> lines(estint-2*esterr,col="gold",
+                            lwd=2)
> lines(estint+2*esterr,col="gold",
+                            lwd=2)
```



Simple MC integration in action

Examples

1. To approximate the integral $\int_0^1 x^4 dx$ in the interval $[0,1]$ one may use the following R code:

```
> U = runif(10^5)
> mean(U^4)
[1] 0.2008846
```

The exact answer naturally is $[x^5/5]_0^1 = 1/5 - 0 = 0.2$

2. To approximate the integral $\int_2^5 \sin(x)dx$ one may use the following R code:

```
> U = runif(10^5, min=2, max = 5)
> mean(sin(U)) * (5-2)
[1] -0.6984924
```

The exact answer is $[-cos(x)]_2^5$, with

```
> cos(2) - cos(5)
[1] -0.699809
```

multiple Monte Carlo integration

Let U_1, U_2, ...,U_n and V_1, V_2, ...,V_n be two sets of independent uniform distributed random variables on the interval $[0,1]$, and suppose $g(x,y)$ is now an integrable function of two variables x and y, then the CLT states that

$$\left(\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\left(g(U_i,V_i)\right)\right)(b-a)(d-c) \;=\; \int_a^b\int_c^d g(x,y)dxdy$$

with probability 1.
So we can approximate the integral $\int_a^b\int_c^d g(x,y)dxdy$ by generating two sets of independent uniform numbers, computing $g(U_i,V_i)$ for each one, and taking the sampled average multiplied by the respective integration intervals.

Example of MMC integration

Example

To approximate the integral $\int_3^{10}\int_1^7 \sin(x-y)dxdy$ one may use the following R code:

```
> U = runif(10^5,min=1,max=7)
> V = runif(10^5,min=3,max=10)
> mean(sin(U-V)) * (7-1) * (10-3)
[1] 0.07989664
```


Importance Sampling Principle

If the density of a random variable is $f(x)$ then

$$E\left[\frac{f(x)}{g(x)}\right] \;=\; \int_{-\infty}^{+\infty}\left(\frac{f(x)}{g(x)}\right)g(x)dx \;=\; \int_{-\infty}^{+\infty} f(x)dx$$

Hence we can approximate the last integral by taking the average of a sample X_i of ratios $f(X_i)/g(X_i)$.

Example

If we are interested in tail probabilities like $P(Z>4.5)=\int_{4.5}^{\infty} f(z)dz$ if $Z\sim\mathcal{N}(0,1)$, which is very small (3.4e-06), we may enhance the MCI approach by using a smart instrumental density $g(x)$ like the exponential distribution truncated at 4.5:

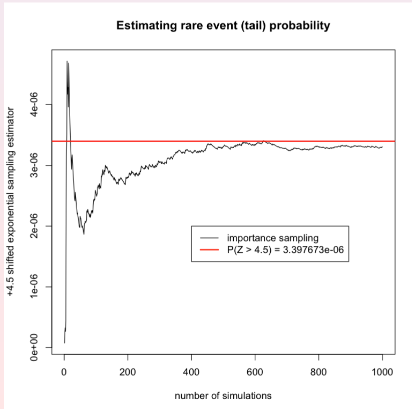
$$g(x)=\frac{e^{-x}}{\int_{4.5}^{\infty} e^{-(x-4.5)}},$$

Example of importance sampling

In the example above, the importance sampling estimator of the tail probability becomes:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{f(X^{(i)})}{g(X^{(i)})} \;=\; \frac{1}{n}\sum_{i=1}^{n}\frac{e^{-X_i^2/2+X_i-4.5}}{\sqrt{2\pi}}$$

```
> pnorm(-4.5)
[1] 3.397673-06
> Nsim=10^3
> x = rexp(Nsim) + 4.5
> isest = cumsum(dnorm(x)/
+           dexp(x-4.5))/1:Nsim
> plot(isest,type="l")
> abline(a=pnorm(-4.5),b=0,
+           col="red")
```



Content of Lecture

Accept-reject principle

Accept-reject Monte Carlo methods are the most powerful and may simulate virtually any integral or density distribution.
We only need to know the target density function f up to a multiplicative constant. We use a simpler instrumental density g verifying the following two conditions:

(i) f and g have a compatible support $[low, high]$, i.e. $g(x) > 0$ when $f(x) > 0$ and $x \in [low, high]$;

(ii) There is a constant M with $f(x)/g(x) \leqslant M$ for all $x \in [low, high]$.

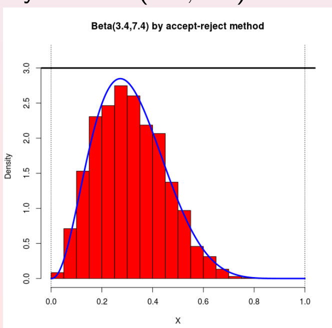
In this case, we proceed like this:

1. Generate independently $Y \sim g$ and $U \sim \mathcal{U}(low, high)$.
2. If $MY \leqslant f(U)$, we set $X = Y$.

Generating a Beta random variable

The support of the beta density is the interval $[0, 1]$. We suppose that $\alpha > 1$ and $\beta > 1$. The upper bound M of the acceptance domain is the highest density observed for $\mathcal{B}eta(a, b)$. For $a = 3.4$ and $b = 7.4$ we notice that $dbeta(3.4, 7.4) < M = 3$. With $U \sim \mathcal{U}(low = 0, high = 1)$, and a uniform intrumental density $Y \sim \mathcal{U}(0, 1)$, we may generate the beta random variable $X \sim \mathcal{B}eta(a = 3.4, b = 7.4)$, by accepting all pairs (U, Y) where MY is strictly below the density of $\mathcal{B}eta(3.4, 7.4)$:

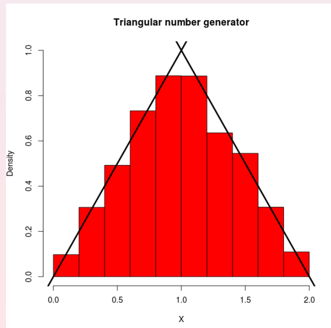
```
> Nsim = 10^4
> low = 0; high = 1
> U = runif(Nsim,low,high)
> Y = runif(Nsim)
> a = 3.4; b = 7.4,M = 3
> X = U[M*Y < dbeta(U,a,b)]
> hist(X,freq=F,xlim=c(0,1),ylim=c(0,3),
+ main="Beta(3.4,7.4)
+       by accept-reject method")
```



Simulating a triangular density function

We may use as well this accept-reject method for simulating a random number generator with a triangular density function$f(x) = 1 - |1 - x|$ for x taking values in the interval $[0, 2]$. The intrumental density may be uniform again. The triangular density being bounded by 1.0, we can set M equal to 1:

```
> Nsim = 10^4
> low = 0 ; high = 2
> U = runif(Nsim,low,high)
> Y = runif(Nsim)
> M = 1
> X = U[M*Y < 1-abs(1-U)]
> hist(X,freq=F,xlim=c(0,2),ylim=c(0,1),
+ main="Triangular number generator")
> abline(0,1);abline(2,-1)
```



Accept-reject based generators - Exercises

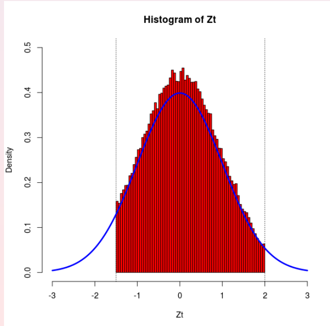
Exercise

1. *Accept-reject methods based generators do not deliver a fixed number of random numbers. Update the method in order to deliver a given number* `Nsim` *of instances.*
2. *Generalize the previous approach to implement a parametric generator for triangular random numbers defined on the real interval* $[m = 0, M = 10]$ *with mode* $x_{mo} = 4$ *and a probability* $r = 0.6$ *to observe a value before or equal* x_{mo} *and* $1 - r = 0.4$ *after it.*

Application: Simulate a truncated Gaussian

We want to simulate the standard normal $Z \sim \mathcal{N}(0,1)$ random variable restricted to the domain $[-1.5, +2]$.
As instrumental distribution we take the standard Z variable and we accept only the observations z that are in the required range. We thus obtain the following truncated Gaussian random variable Zt:

```
> Nsim = 10^5
> low = -1.5; high = 2
> Z = rnorm(Nsim)
> Zt = Z[(Z > low) & (Z < high)]
> hist(Zt,freq=F,breaks=51,
      xlim=c(-3,3),col="red")
> z = seq(-3,3,length=500)
> lines(z,dnorm(z),col="blue")
```

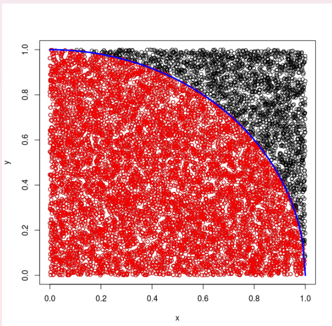


Application: Monte Carlo π estimation

The area of the circle of radius $r = 1$ is πr^2. The area of the square containing this circle is $(2r)^2 = 2^2 = 4$. The ratio of the area of the circle to the area of the square is:

$$\rho = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4} = \frac{3.141593}{4} = 0.7853982$$

```
> x = runif(Nsim)
> y = runif(Nsim)
> plot(x,y)
> rhox = x[(x^2+y^2)<1]
> rhoy = y[(x^2+y^2)<1]
> points(rhox,rhoy,col="red")
> ax = seq(0,1,0.01)
> lines(ax, sqrt(1-ax^2),
+       lwd=3,col="blue")
> 4*length(rhox)/length(x)
[1] 4 x 0.786 = 3.144
```



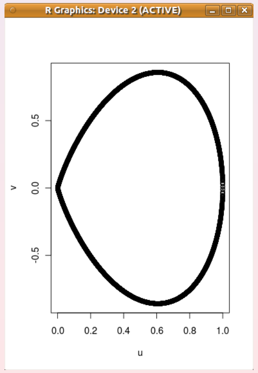
The Box-Muller accept-reject tranform

Recall the Box-Muller algorithm for the centered and reduced normal $Z \sim \mathcal{N}(0,1)$ variable. It is based on the observation that, if U_1 and U_2 are two independent and identically $\mathcal{U}(0,1)$ distributed random variables, then: $X_1 = \sqrt{-2\log(U_1)}\cos(2\pi U_2)$, $\quad X_2 = \sqrt{-2\log(U_1)}\sin(2\pi U_2)$, are two independent and identically $\mathcal{N}(0,1)$ distributed random variables.
Suppose we pick V_1 and V_2 instead as the ordinate and abcissa of a uniform random point in the unit circle around the origin. Then the sum of their squares $R^2 = V_1^2 + V_2^2$ is a uniform variable that can be used for U_1, while the angle that the point (V_1, V_2) defines with respect to the V_1 axis can serve as random angle $2\pi U_2$.
The cosine and sinus in the Box-Muller formula can now be written as $V_1/\sqrt{R^2}$ and $V_2/\sqrt{R^2}$. This implementation can in fact be seen as a kind of acept-reject method for computing trigonometric functions of a uniform random angle.
(See Box-Muller transform)

Ratio-Of-Uniforms Method

Virtually any random variable X can be simulated by the following simple prescription:

1. Construct a region A in the (u, v) plane bounded by $0 \leqslant u \leqslant [p(v/u)]^{1/2}$.
2. Choose a point $P = (u, v)$ distributed uniformly within this region A.
3. If $P(u, v) \in A$, return v/u as a required simulated random variable instance .



Fast generation of Gaussian random variable

In case of a normal $Z \sim \mathcal{N}(0, 1)$ random variable, the region A becomes:

$$A = \{(u, v) \mid v^2 < -4u^2 \ln u\}.$$

This region is entirely contained in the rectangle $R = \{0 < u < 1, -(2/e)^{1/2} < v < (2/e)^{1/2}\}$ and the accept-reject method is used to select the points $P = (u, v)$ such that $z = v/u$ delivers the variable Z.

Exercise

In 1992, Joseph Leva has published a very fast and efficient Z variable generator based on this approach (see his paper in the moodle resources).

1. *Implement this algorithm in C++ (NR), in Python and in R,*
2. *Check the quality of the generator when compared with the standard Python and R generators,*
3. *Compare the respective run times in C++, in Python and R for a sample of 100000 normal random numbers.*