Monte Carlo Methods PART I: Random Variables simulation

Law of Large Numbers

Theorem

Let $X1, X2, \ldots$ *be a sequence of independent and identically distributed random variables with mean* $\mu := \mathbb{E}(X) \in \mathbb{R}$. *As* $n \to \infty$, $n^{-1} \sum_{i=1}^{n} X_i$ *becomes less and less random, and converges to* μ.

- A fundamental result in probability theory and statistics.
- Makes statistical learning possible.

Law of Large Numbers

Let's write some code to check this.

- Let $X1, X2, \ldots$ be iid $U(0, 1)$.
- For each value of n in $\{50, 100, 150, 200, ..., 10000\}$, we calculate the averages $n^{-1} \sum_{k=1}^{n} X_k$.
- We expect these averages to converge towards $\mathbb{E}(X) = \int_0^1 xf(x)dx = \int_0^1 xdx = 0.5$.

Law of Large Numbers

```
N = 1e5;
n = seq(from=50, to =N, by = 50);
K = length(n);
Res=double(K)
Sple=runif(N,0,1)
for (i in 1:K){
    Res[i]=mean(Sple[1:n[i]])
}
plot(Res,type='l',xlab='n',ylab='prob')
abline(h=0.5)
```

Law of Large Numbers

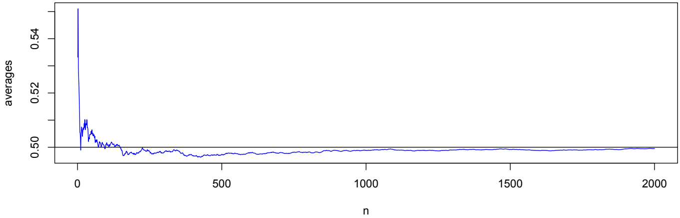

Figure 1: Illustration of the LLN

Law of Large Numbers

- More generally, let a function $h : \mathbb{R} \to \mathbb{R}$ be such that

$$\mathbb{E}(h(X)) = \sum h(x)f(x) \text{ (or } \int h(x)f(x)dx)$$

is well-defined. Then as $n \to \infty$, we have

$$\frac{1}{n}\sum_{k=1}^{n} h(X_k), \to \mathbb{E}(h(X)),$$

- In the above formula we assume that the density of X is f.

The central limit theorem

- The central limit theorem attempts to give more detail on the rate of convergence in the law of large numbers.

Theorem

$X_1, X_2, \ldots$ are i.i.d. random variables with mean $\mu = \mathbb{E}(X)$ and variance $0 < \sigma^2 < \infty$. Then for any $u \in \mathbb{R}$, as $n \to \infty$,

$$\mathbb{P}\left(\frac{1}{\sigma\sqrt{n}}\sum_{k=1}^{n}(X_k - \mu) \leq u\right) \to \mathbb{P}(Z \leq u),$$

where $Z \sim \mathcal{N}(0,1)$.

- Key point: this limiting behavior does not depend on the common distribution of the random variables $X_1, X_2, \ldots$ (but note that we need the variance to be finite).

The central limit theorem

- Basically the result says that when n is large $\frac{1}{\sigma\sqrt{n}}\sum_{k=1}^{n}(X_k - \mu)$ behaves like a standard normal random variable Z.
- This implies that

$$
\begin{aligned}
\frac{1}{n}\sum_{k=1}^{n} X_k &= \mu + \frac{1}{n}\sum_{k=1}^{n}(X_k - \mu) \\
&= \mu + \frac{\sigma}{\sqrt{n}} \underbrace{\frac{1}{\sqrt{n}}\sum_{k=1}^{n}\left(\frac{X_k - \mu}{\sigma}\right)}_{Z}, \\
&\approx \mu + \frac{\sigma Z}{\sqrt{n}}.
\end{aligned}
$$

The central limit theorem

- The CLT implies an interval estimation of μ called confidence interval.
- Because $Z \in (-1.96, 1.96)$ 95% of the time, we see that 95% of the time we have

$$-\frac{1.96\sigma}{\sqrt{n}} \leq \frac{1}{n}\sum_{i=1}^{n} X_i - \mu \leq \frac{1.96\sigma}{\sqrt{n}}.$$

  Hence, 95% of the time

$$\mu \in \left(\frac{1}{n}\sum_{i=1}^{n} X_i - \frac{1.96\sigma}{\sqrt{n}}, \frac{1}{n}\sum_{i=1}^{n} X_i + \frac{1.96\sigma}{\sqrt{n}}\right).$$

- Using the confidence interval is a better way of estimating μ than just reporting $(1/n)\sum_{i=1}^{n} X_i$.

The central limit theorem

- Example: $X_1, \ldots, X_n$ i.i.d. $\mathcal{U}(0,1)$. therefore $\mathbb{E}(X) = 0.5$ and $\mathrm{Var}(X) = 1/12$. We compare the distribution of $\sum_{k=1}^{n} (X_k - 0.5)/(\sigma\sqrt{n})$ for different values of n.
- One way to examine the distribution of the random variable $\sum_{k=1}^{n} (X_k - 0.5)/(\sigma\sqrt{n})$ is to generate it a large number of times, say $N = 1,000$ times and check the histogram.

The central limit theorem

```
genCLTDist=function(n){
    N=1000
    replicate( N, sqrt(12/n)*sum(runif(n,0,1)-0.5) )
}

nVec=c(1,10,30)
Res=sapply(nVec,genCLTDist)
```

The central limit theorem

```
par(mfrow = c(1,3))
hist(Res[,1],col='blue',breaks=30,prob=T,
          main='n=1',xlim=c(-4,4),ylim=c(0,0.5))
par(new=T)
curve(dnorm,xlim=c(-4,4),col='red',ylim=c(0,0.5))
hist(Res[,2],col='blue',breaks=30,prob=T,
          main='n=10',xlim=c(-4,4),ylim=c(0,0.5))
par(new=T)
curve(dnorm,xlim=c(-4,4),col='red',ylim=c(0,0.5))
hist(Res[,3],col='blue',breaks=30,prob=T,
          main='n=30',xlim=c(-4,4),ylim=c(0,0.5))
par(new=T)
curve(dnorm,xlim=c(-4,4),col='red',ylim=c(0,0.5))
```

The central limit theorem

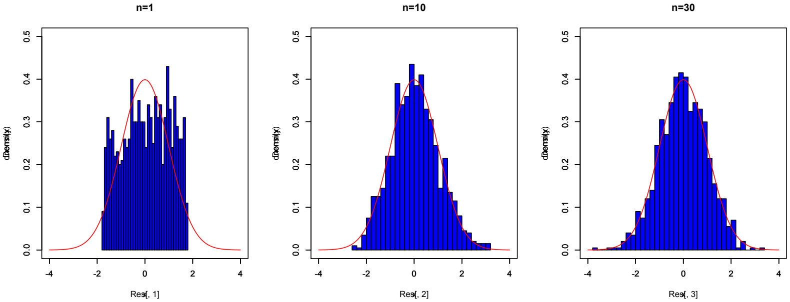

Figure 1: Illustration of the CLT

Random numbers generation

- Most commonly used random variables can be generated in *R* using appropriate functions. There is a naming convention in *R* best explained using the Gaussian distribution:
  - To call the pdf of the normal distribution, use dnorm,
  - To call the cdf, use pnorm and to compute quantiles, use qnorm,
  - To generate a normal distribution, use rnorm.
- For example, for the geometric distribution the names become respectively dgeom, pgeom, qgeom and rgeom etc...

Random numbers generation

Distribution	Generator
beta	rbeta
binomial	rbinom
chi-squared	rchisq
exponential	rexp
F	rf
gamma	rgamma
geometric	rgeom
lognormal	rlnorm
negative binomial	rnbinom
normal	rnorm
Poisson	rpois
Student's t	rt
uniform	runif

Table: Functions to generate usual distributions in *R*.

Random numbers generation

- If F is a cdf with pdf f (or pmf f), the notation $X \sim F$ (resp. $X \sim f$) means that X is a random variable with distribution F and pdf (resp. pmf) f.
- Question: what does it mean to "simulate a random variable from a given distribution"?

Definition

We will say that an algorithm generates a random variable with density f, if by repeating the algorithm, it can produce a sequence $x_1, x_2, \ldots,$ such that for any numbers $a < b$,

$$\frac{\#\{1 \leq i \leq n : \; x_i \in (a, b)\}}{n} \to \int_a^b f(x)dx, \quad \text{as } n \to \infty,$$

for all practical purposes.

Random numbers generation

- Here, for all practical purposes means that we have no way of proving that this convergence does not hold (even if it does not).
- Note that we do not assume that the sequence $x_1, x_2, \ldots$ is random. Hence computer programs can generate sequences that satisfies our definition. We call these pseudo-random numbers.
- This definition can be easily extended with obvious modifications to non-continuous distributions.

Two general methods

- The number of distribution is infinite and it is not possible to have a software that can generate from any given distribution.
- It is therefore important to know few general principles that you can use to design your own generators if needed.

The inversion method

Let F be a cdf with positive density f. Therefore F is continuous nondecreasing and possesses an inverse F^{-1}. We have the following result.

Proposition

Let $U \sim \mathcal{U}(0,1)$ and define $X = F^{-1}(U)$. Then $X \sim F$.

In other words, we can always generate from any density f if we can compute F^{-1}, the inverse of its cdf. Here is an example.

The inversion method: example

- Suppose we want to generate a rv with density $f(x) = 3x^2$, $0 < x < 1$.
- The cdf is $F(x) = \int_{-\infty}^{x} f(t)dt = x^3$, $0 < x < 1$ and $F^{-1}(u) = u^{1/3}$.
- Therefore, we can generate from f by doing the following: generate $U \sim \mathcal{U}(0,1)$, set $X = U^{1/3}$.

The inversion method: example

```
n=10000
u=runif(n)
x=u^{1/3}
hist(x,prob=TRUE) # produces an hist of the sample
y=seq(0,1,length=200)
lines(y,3*y^2)
```

The inversion method: example

- Consider the exponential distribution. $f(x) = \lambda e^{-\lambda x}$, $x \geq 0$.
- Then the cdf is $F(x) = 1 - e^{-\lambda x}$, and $F^{-1}(u) = -\frac{1}{\lambda}\log(1-u)$.
- This gives a very easy way to generate exponential random variables: Generate $U \sim \mathcal{U}(0,1)$ and take $X = -\frac{1}{\lambda}\log(U)$.

The inversion method: example

```
n=10000
lambda=1
x=-(1/lambda)*log(runif(n))
hist(x,prob=TRUE) # produces an hist of the sample
y=seq(0,10,length=1000)
lines(y,dexp(y,rate = 1))
```

The inversion method: example

Practice problem: Design an inversion algorithm to simulate from the logistic distribution with cdf $F(x) = \frac{e^x}{1+e^x}$, $x \in \mathbb{R}$.

The inversion method

For discrete distributions, the inverse methods takes a slightly different form and is based on the following result.

Proposition

Consider the discrete distribution that takes value x_i with probability p_i, ($\sum_{i=1}^{\infty} p_i = 1$). Let $U \sim \mathcal{U}(0,1)$, and define

$$X := \inf \left\{ i \geq 1 : \sum_{k=1}^{i} p_k \geq U \right\}.$$

Then $X \sim p$.

The formula above means that X is the first integer $i \geq 1$ for which $\sum_{k=1}^{i} p_k \geq U$.

The inversion method

To obtain X we do the following.

- We first generate a uniform random variable $U \sim \mathcal{U}(0,1)$.
- If $p_1 \geq U$, then $X = 1$. If not we check to see if $p_1 + p_2 \geq U$. If so, $X = 2$. If not we check to see if $p_1 + p_2 + p_3 \geq U$, etc...

The inversion method: a three-points distribution example

- Suppose we want to generate from a distribution on $\{1, 2, 3\}$, where $\mathbb{P}(X = i) = p_i$, $i = 1, 2, 3$, $p_1 + p_2 + p_3 = 1$.
- We do the following according to Proposition 3.2:
  1. We generate $U \sim \mathcal{U}(0, 1)$.
  2. If $p_1 \geq U$, we return 1. Otherwise if $p_1 + p_2 \geq U$, we return 2. Otherwise we return 3.

The inversion method

Proposition 3.2 is particularly useful to sample from arbitrary discrete random variable as we have in the following algorithm which extends the example above.

Algorithm (Generate from $\Pr(X = x_i) = p_i,\ i = 1 \ldots, n$)

- $U \sim \mathcal{U}(0,1)$; $i = 1$; $S = p_i$.
- *While* $S < U$:
  1. $i = i + 1$;
  2. $S = S + p_i$;
- *Return* x_i.

Note: This method will probably not be very efficient for large n. More efficient methods will exploit the structure of $\{p_i\}$

The inversion method for Poisson rv

Generate $X \sim \mathcal{P}(\lambda)$. Its pmf is given by $p_\lambda(x) = e^{-\lambda}\lambda^k/k!$.

```
InvrPois = function(lambda, n){
    X=vector('numeric',n)
    for (i in 1:n){
        U=runif(1);k=0;S=exp(-lambda)
        while(U>S){
            k=k+1;S=S+exp(-lambda +k*log(lambda)
                                      -lfactorial(k))
        }
        X[i]=k
    }
    return(X)
}
```

Notice how we implement the ratio $\frac{\lambda^k}{k!}$ on a logarithm scale.

The inversion method: exercise

- The Pareto distribution has cdf $F(x) = 1 - \left(\frac{b}{x}\right)^a$, $x \in [b, \infty)$, $a > 0, b > 0$. Derive an inversion method to generate random variables according to the Pareto distribution. Implement your method in R.
- Derive an inversion method to generate random variables from the following discrete distribution. Implement your method in R.

x	0	1	2	3	4
$p(x)$	0.1	0.2	0.2	0.2	0.3

The inversion method

The obvious limitation of the inversion method is that it requires the calculation of F^{-1}. Another limitation is that it does not carry over easily to spaces with dimensional greater than 1.

The Rejection method

- Also known as the Accept-Reject method.
- One of the first truly general simulation method we will see in this class. Also one of the most beautiful simulation method.
- Based on what one could call the fundamental principle of simulation.

Theorem (Fundamental principle of simulation)

Generating points uniformly under the curve of a density f is equivalent to generating random variables distributed according to f.

The Rejection method

Fundamental Principle Part I: If $(X_i, Y_i)_{1\leq i\leq n}$ are independent and uniformly distributed under the curve of a density f, then $(X_i)_{1\leq i\leq n}$ are i.i.d. with density f.

Question: How do we generate points under the curve of f? If the support of f is bounded, we could proceed as follows.

- We find a box (or a hyper-cube) that envelops f.
- We draw random points uniformly in that box. This is easy!
- Of those random points, we retain only those that fall under the curve f. These points are necessarily also uniformly distributed under the curve f.

The Rejection method

Example

Consider the density $f(x) = \cos(x)$, $x \in (0, , \pi/2)$. We can box this density using the rectangle $[0, \frac{\pi}{2}] \times [0, 1]$. So we can simulate from f by rejection method as follows.

Algorithm (Rejection Sampling)

1. *Generate* $U_1 \sim \mathcal{U}(0, \frac{\pi}{2})$ *and* $U_2 \sim \mathcal{U}(0, 1)$.
2. *If* $U_2 \leq \cos(U_1)$, *ACCEPT* U_1. *Otherwise, go back to 1., and start over.*

The Rejection method

```
n=1000
U1=runif(n,0,pi/2)
U2=runif(n,0,1)
layout(matrix(c(1,2,3),ncol=3))
plot(U1,U2,pch='x',col='red')
U1_Accept=U1[which(U2<=cos(U1))]
U2_Accept=U2[which(U2<=cos(U1))]
plot(U1_Accept,U2_Accept,pch='x',col='red')
hist(U1_Accept,nclass=50,prob=T,col='blue',
         xlim=c(0,pi/2),ylim=c(0,1.2))
par(new=T)
curve(cos,xlim=c(0,pi/2),
          col='red',ylim=c(0,1.2))
```

The Rejection method

Fundamental Principle Part II: If we can simulate directly from a density, say g, then we can produce points uniformly under the curve of $c \times g$, for any constant c, using the following algorithm.

Algorithm

1. *Generate* $Y \sim g$.
2. *Generate* $U \sim \mathcal{U}(0, cg(Y))$.

Then (Y, U) *is uniformly distributed under the curve* cg

The Rejection method

Example

Consider $g(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, $x \in \mathbb{R}$, the density of $N(0,1)$. Set $c = 2.5$.

```
n=1000; c=2.5
Y=rnorm(n,0,1)
U=runif(n,0,c*dnorm(Y,0,1))
plot(Y,U,pch='x',col='red')
```

The Rejection method

- We can combine these two principles to simulate from any given density f.
- We first need to find another density g that we already know how to simulate from another,and a constant M such that the curve $M \times g$ envelops the density f. That is:

$$f(x) \leq Mg(x) < \infty, \quad x \in \mathbb{R}.$$

- Then According to Part II, we can simulate $Y \sim g$, and $U \sim \mathcal{U}(0, Mg(Y))$ to obtain points (Y, U) uniformly under the curve of Mg.
- Of those points (Y, U), we collect those that fall under the curve of f: $U \leq f(Y)$, and we return their Y as sampled from f.

The Rejection method

The algorithm is the following:

Algorithm (Rejection Sampling)

1. *Generate* $Y \sim g$ *and generate* $U \sim \mathcal{U}(0, Mg(Y))$.
2. *If* $U \leq f(Y)$, *Stop and return* Y.
3. *Otherwise, reject* Y *and go back to Step 1.*

Here is an equivalent version:

Algorithm (Rejection Sampling)

1. *Generate* $Y \sim g$ *and generate* $U \sim \mathcal{U}(0, 1)$.
2. *If* $UMg(Y) \leq f(Y)$, *Stop and return* Y.
3. *Otherwise, reject* Y *and go back to Step 1.*

The Rejection method: an example

A good rejection algorithm is one in which we do not reject many Y. What is the probability that we accept the proposed Y?

$$
\begin{aligned}
\Pr(\text{Accept}|Y=y) &= \Pr\left(UMg(Y) \leq f(Y)|Y=y\right) \\
&= \Pr\left(U \leq \frac{f(Y)}{Mg(Y)}|Y=y\right) = \frac{f(y)}{Mg(y)}.
\end{aligned}
$$

Thus

$$
\begin{aligned}
\Pr(\text{Accept}) &= \sum_y \Pr(\text{Accept}|Y=y)\Pr(Y=y) \\
&= \sum_y \frac{f(y)}{Mg(y)} g(y) = \frac{1}{M}.
\end{aligned}
$$

The Rejection method: an example

- Thus the number of times we try before having one success in Algorithm 3.5 is a geometric random variable with parameter $1/M$.
- Hence for a good rejection algorithm we need to choose g such that M is small.
- To achieve this goal, we will often need to use calculus.
- However in high-dimensional space this goal is typically impossible to achieve because of the curse of dimensionality.

The Rejection method: an example

- The Beta distribution has density $f(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$, $0 < x < 1$, for some parameters $\alpha > 0, \beta > 0$, where $B(\alpha, \beta) = \Gamma(\alpha)\Gamma(\beta)\Gamma(\alpha+\beta)^{-1}$, $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx$. $B(\alpha, \beta)$ is called the normalizing constant of the density.

```
densitybeta=function(x){
    alpha=2;beta=5
    dbeta(x,alpha,beta)
}
curve(densitybeta,from=0,to=1,n=200,col='blue')
```

You can play with α, β to see all the different form that this density can take.

The Rejection method: an example

- For $\alpha \geq 1$, $\beta \geq 1$, we can use the rejection method to generate from f.
- A natural candidate is the uniform distribution with density $g(x) = 1$, $0 < x < 1$.
- But we need to check the boundedness condition. $\frac{f(x)}{g(x)} = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \leq \frac{1}{B(\alpha,\beta)}$. This is our M.
- In the algorihm, the condition $UMg(Y) \leq f(Y)$ becomes:

$$\frac{U}{B(\alpha,\beta)} \leq \frac{1}{B(\alpha,\beta)} Y^{\alpha-1}(1-Y)^{\beta-1},$$

which is equiv to $U \leq Y^{\alpha-1}(1-Y)^{\beta-1}$.

This lead to the algorithm.

The Rejection method: an example

Algorithm (Rejection Sampling I)

1. *Generate* $Y \sim \mathcal{U}(0,1)$ *(from* g*) and generate* $U \sim \mathcal{U}(0,1)$.
2. *If* $U \leq Y^{\alpha-1}(1-Y)^{\beta-1}$, *Stop and return* Y.
3. *Otherwise, reject* Y *and go back to Step 1.*

The Rejection method: an example

```
ARBeta1=function(n,alpha,beta){
    Vy=numeric(n);Vcpt=integer(n);
    j=1;cpt=0;
    while (j<=n){
        u=runif(1); y=runif(1);cpt=cpt+1
        if (u<=y^(alpha-1)*(1-y)^(beta-1)){
            Vy[j]=y; Vcpt[j]=cpt
            j=j+1; cpt=0
        }
    }
    return(list(Vy,Vcpt))
}
```

Exercise: Write the same code using for and while loops.

The Rejection method: an example

But is not hard to find a better bound for $\frac{f(x)}{g(x)}$. Indeed suppose that $\alpha \leq \beta$. Then

$$\frac{f(x)}{g(x)} = B^{-1}(\alpha,\beta)x^{\alpha-1}(1-x)^{\beta-1} = B^{-1}(\alpha,\beta)\left(x(1-x)\right)^{\alpha-1}(1-x)^{\beta-\alpha}$$
$$\leq B^{-1}(\alpha,\beta)\left(x(1-x)\right)^{\alpha-1} \leq B^{-1}(\alpha,\beta)\left(\frac{1}{4}\right)^{\alpha-1}.$$

And we get another (almost similar) algorithm:

Algorithm (Rejection Sampling II)

1. *Generate $Y \sim \mathcal{U}(0,1)$ and generate $U \sim \mathcal{U}(0,1)$.*
2. *If $U \leq 4^{min(\alpha,\beta)-1}Y^{\alpha-1}(1-Y)^{\beta-1}$, Stop and return Y.*
3. *Otherwise, reject Y and go back to Step 1.*

The Rejection method: an example

```
ARBeta2=function(n,alpha,beta){
     a=min(alpha,beta)-1
    Vy=numeric(n);Vcpt=integer(n);
    j=1;cpt=0;
    while (j<=n){
        u=runif(1); y=runif(1);cpt=cpt+1
        if (u<=4^a*y^(alpha-1)*(1-y)^(beta-1)){
            Vy[j]=y; Vcpt[j]=cpt
            j=j+1; cpt=0
        }
    }
    return(list(Vy,Vcpt))
}
```

The Rejection method: an example

```
alpha=2;beta=5
n=2500;
system.time(Res1<-ARBeta1(n,alpha=2,beta=5));
system.time(Res2<-ARBeta2(n,alpha=2,beta=5));
c(mean(Res1[[2]]),mean(Res2[[2]]))
```

The Rejection method

To conclude on the rejection method:

- It is a very general and elegant idea to sample from any density.
- However, for it to work well we need to find a tight box, or a good density g, around the density f.
- In high-dimensional spaces, this requirement is hard to achieve because of the curse of dimensionality.

Illustration of the curse of dimensionality

- In R^n, the volume of the ball $B_n(a) = \{x \in \mathrm{R}^n : \|x\| \leq a\}$ is
$$V_n(a) = \frac{\pi^{n/2}}{\Gamma\left(\frac{n}{2}+1\right)} a^n.$$
- Hence
$$\frac{V_n(a+\epsilon)}{V_n(a)} = e^{n \log\left(1+\frac{\epsilon}{a}\right)} \approx e^{n\epsilon/a}.$$
- If $a = 10$, $\epsilon = 0.1$, $\frac{V_1(a+0.1)}{V_1(a)} = 1.01$, and $\frac{V_{1000}(a+0.1)}{V_{1000}(a)} > 2 \times 10^4$.
- Hence, as $n \to \infty$, $B_n(a)$ becomes minuscule in $B_n(a+\epsilon)$, no matter how small ϵ.
- The Monte Carlo problem is a fundamentally difficult problem.