

# Monte-Carlo method I

( generating random variables)

Woojoo Lee

## Motivation

An interesting quantity can be expressed as the expectation of a function of a random variable:  $E(h(X))$

Suppose that a random variable  $X$  has the density function  $f(x)$ .

For examples,

$$\mu = E(X) = \int x f(x) dx$$

$$\sigma^2 = E(X - \mu)^2 = \int (x - \mu)^2 f(x) dx = \int x^2 f(x) dx - \mu^2$$

$$P(X > 2) =$$



Likewise,

$$E(h(X)) = \int h(x) f(x) dx.$$

But, this is often difficult to compute !

We will solve this problem by using some sampling methods.

Consider the law of large number. When  $X_i$  are i.i.d with a finite first moment,

Average of random events   $\frac{1}{n} \sum_{i=1}^n X_i \rightarrow E(X_1)$   expectation

The strong law of large numbers

The weak law of large numbers      <-    We focus this in our class.

An illustration of the law of large numbers with a die

```
x<-c(1,2,3,4,5,6)
```

```
y<-sample(x,size=10000,replace=TRUE)
```

```
z<-rep(1,10000)
```

```
average<-cumsum(y)/cumsum(z)
```

```
plot(cumsum(z),average)
```

We can show why the left hand side converges to the right hand side.

For simplicity of the proof, we assume that the second moment is finite. Note that without the finite second moment assumption, this can be proved. But, more complicated !)

Consider

$$E\left(\frac{1}{n} \sum_{i=1}^n X_i\right)$$

and

$$Var\left(\frac{1}{n} \sum_{i=1}^n X_i\right)$$

Likewise, the (weak) law of large numbers holds for  $h(X_1), h(X_2) \dots, :$   
Assume that  $X_i$  are iid and  $E(h(X_1))$  is finite. Then,

$$\frac{1}{n} \sum_{i=1}^n h(X_i) \rightarrow E(h(X_1)).$$

Example: estimating  $\pi$

```
#### Estimating pi
set.seed(1234)
n<-100000
x<-runif(n,0,1)
y<-runif(n,0,1)

dist.r<-function(x,y){
  r<-sqrt((x-0.5)^2+(y-0.5)^2)
  return(r)
}

rr<-dist.r(x,y)
4*sum(rr<=0.5)/n
```

Example) Compute  $\int_{-2}^1 \phi(x) dx$  where  $\phi(x)$  denotes the standard normal density function.

```
#### Estimating int_{-2}^1 phi(x)dx
```

```
set.seed(1234)
```

```
n<-100000
```

```
x<-rnorm(n,0,1)
```

```
sum((x>=-2)*(x<=1))/n
```

```
pnorm(1)-pnorm(-2)
```

Thus, our task is to know how to generate random variables  $X_i$  from  $f(x)$ .

Remark) Assume that we know how to generate uniform random numbers.

## The inverse method

Suppose that a random variable (rv)  $X$  has a continuous cumulative distribution function (cdf)  $F$ . Then

$$F(X) \sim U[0, 1].$$

R-code example)

```
#### F(X) follows uniform distn
```

```
set.seed(1234)
```

```
x<-rexp(10000,rate=1) ##### f(x)=rate*exp(-rate*x)
```

```
cdfexp1<-function(x){  
    1-exp(-x)  
}
```

```
hist(cdfexp1(x))
```

Question) What is the cdf for  $1 - F(X)$  ?



If  $U \sim U[0, 1]$ ,

$$X \sim F^{-1}(U).$$

Q) Why does this hold ?

Remark) This method is useful when  $F^{-1}$  is easy to obtain.

Example: generating  $\text{Exp}(1)$

R-code example)

```
### Inverse method
```

```
gexp1<-function(n){  
  u<-runif(n,0,1)  
  x=-log(1-u)  
  return(x)  
}
```

```
set.seed(1234)  
res<-gexp1(100000)  
hist(res)  
xx<-seq(0,10,length=1000)  
lines(xx,100000*dexp(xx),type="l",col="red")
```

Example) Standard Gaussian distribution - easy or difficult ?

Example) Consider a three dimensional distribution - easy or difficult ?

## Acceptance-Rejection method (or Rejection sampling)

Consider the following density function:

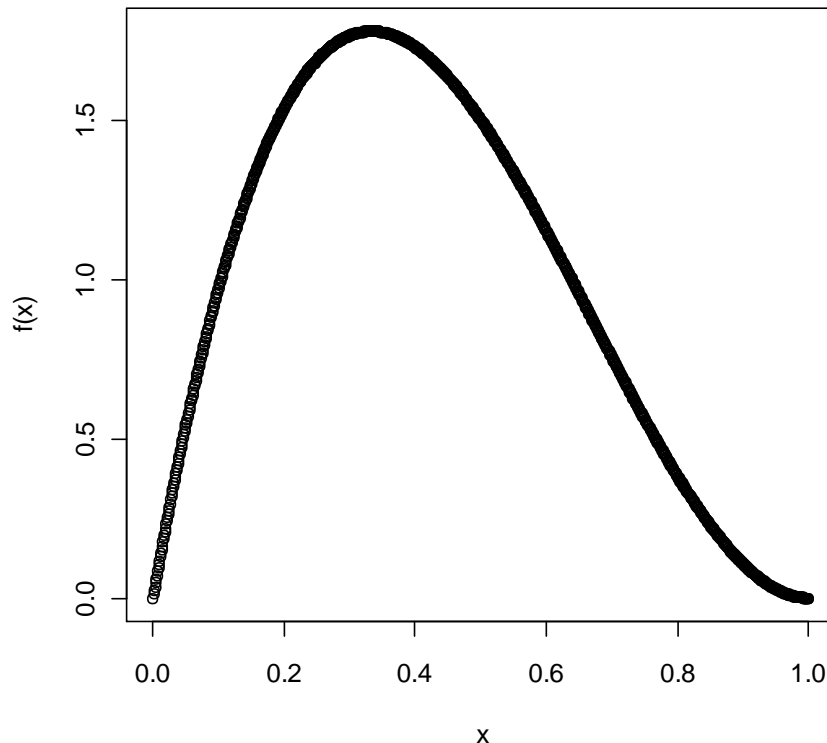
$$f(x) = 12x(1 - x)^2$$

where  $x \in (0, 1)$ .

Note that it is quite difficult to compute  $F^{-1}$  for this problem.

Key procedure: Suppose that  $g(x)$  denotes the uniform density on  $(0, 1)$

- Generate  $Y \sim g(x)$ .
- Generate  $U \sim U(0, 1)$ .
- Accept the points which satisfy  $f(Y)/(1.8 * g(Y)) \geq U$ .  
Otherwise, reject the points.
- Repeat the above procedure until you have enough sample sizes.



Q) Why does this method work ?

R-code: how to obtain rv from

$$f(x) = 12x(1 - x)^2$$

```
· beta23<-function(x){  
    12*x*(1-x)^2  
}  
  
xx<-seq(0,1,length=1000)  
plot(xx,beta23(xx),ylab="f(x)",xlab="x")  
  
min(beta23(xx)) ## 0  
max(beta23(xx)) ## 1.7777 ~ 1.8  
  
set.seed(1234)  
n<-10000  
  
x<-runif(n,0,1)  
u<-runif(n,0,1)  
rej<-as.numeric(beta23(x)/1.8<u)  
  
x.OK<-x[which(rej==0)]  
  
plot(density(x.OK))  
lines(xx,beta23(xx),ylab="f(x)",xlab="x",col  
="red")
```

One key advantage of this method is that we do not need to know the proportionality constant of  $f(x)$ .

Suppose that we only know  $f(x) \sim x(x-1)^2$  on  $(0, 1)$ . Let  $\tilde{f}(x) = x(x-1)^2$ .

See the following procedure once again !

- Generate  $Y \sim g(x)$ .
- Generate  $U \sim U(0, 1)$ .
- Accept the points which satisfy  $\tilde{f}(Y)/(0.16 * g(Y)) \geq U$ .  
Otherwise, reject the points.
- Repeat the above procedure until you have enough sample size.

Q) Why does this method work ?

R-code: how to obtain rv from

$$f(x) \sim x(1-x)^2$$

.

### we do not need to have the  
proportionality constant.

```
beta00<-function(x){  
  x*(1-x)^2  
}
```

```
xx<-seq(0,1,length=1000)  
plot(xx,beta00(xx),ylab="f(x)",xlab="x")  
### max<0.16
```

```
set.seed(1234)  
n<-10000
```

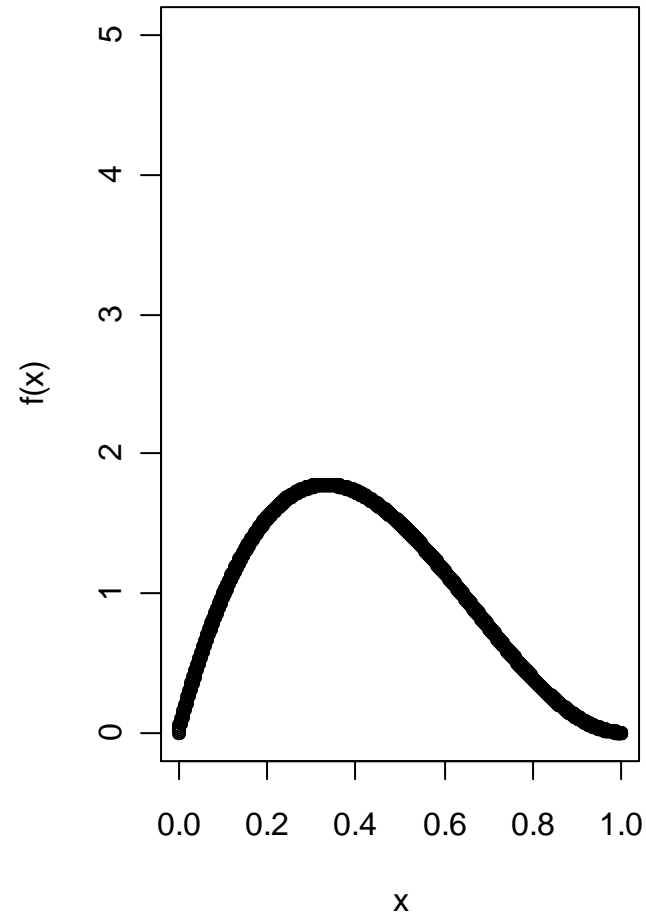
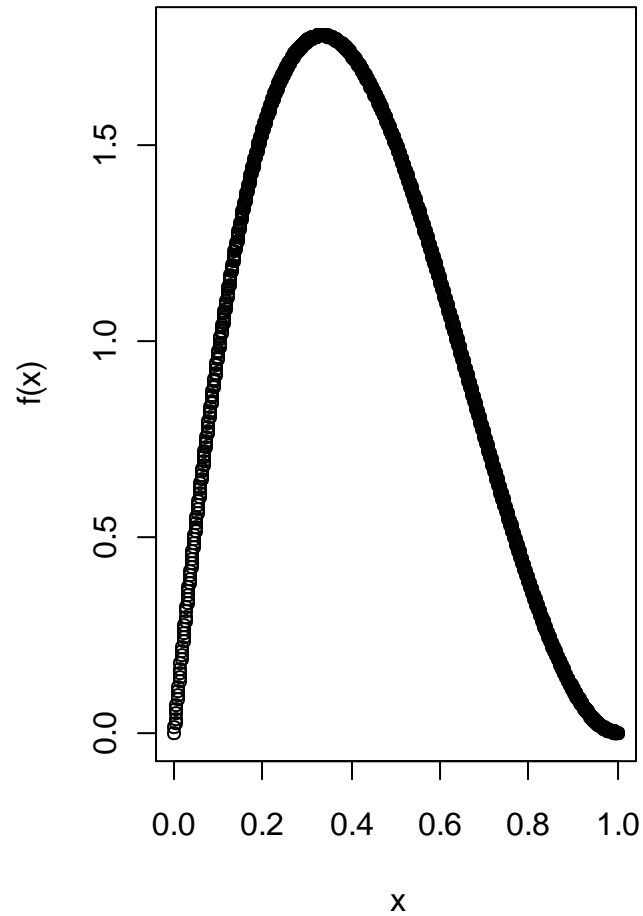
```
x<-runif(n,0,1)  
u<-runif(n,0,1)  
rej<-as.numeric(beta00(x)/0.16<u)
```

```
x.OK<-x[which(rej==0)]
```

```
plot(density(x.OK))  
lines(xx,beta23(xx),ylab="f(x)",xlab="x",col  
="red")
```

If the number of rejections is large, the efficiency of this algorithm will be low.

Which bounding function should be used ?





This implies that the rectangular shape may not be optimal as a bounding function, so we should consider a tighter function to  $f(x)$ .

Acceptance-Rejection algorithm is:

- Find a "easy-to- use" density function  $g(x)$  satisfying  $f(x) \leq Kg(x)$  for all  $x$  and some constant  $K$ .
- Simulate  $Y \sim g(x)$ .
- Simulate  $U \sim U(0, 1)$ .
- Accept the points  $Y$  only if  $f(Y)/(Kg(Y)) \geq U$ .
- Set  $X = Y$ . Repeat this until we have enough samples.
- Then,  $X \sim f(x)$ .

Q) Advantages ?

Q) Disadvantages ?

Q) Why does " $X \sim f(x)$ " hold ?

## Rejection sampling in more than 1 dimension

Suppose that we want to generate random variables from a bivariate distribution:

$$f(x, y) = x + y$$

for  $x, y \in (0, 1)$

Q) Suggest your rejection sampling procedure.

## The sampling importance resampling (SIR) algorithm

The target density:  $f(x)$

The importance sampling function:  $g(x)$

The standardized importance weights:  $w(x_i) = \frac{f(x_i)/g(x_i)}{\sum_{i=1}^m (f(x_i)/g(x_i))}$

SIR algorithm is as follows:

- Simulate  $Y_1, \dots, Y_m \sim g(x)$ .
- Compute the standardized importance weights:  $w(Y_1), \dots, w(Y_m)$
- Resample  $X_1, \dots, X_n$  from  $Y_1, \dots, Y_m$  with replacement with probabilities  $w(Y_1), \dots, w(Y_m)$

Remark) When implementing SIR, the relative size  $n/m$  is very important. In principle, we require  $n/m \rightarrow 0$ . A rule of thumb says that  $n/m \sim 0.1$  is tolerable.

Some criteria for the choice of  $g$

- The support of  $g \supset$  the support of  $f$

Q) Why ?

- $g$  should have heavier tail than  $f$ . More generally,  $f(x)/g(x)$  should not grow too large.

Q) Why ?

→ The distribution of weights should not have high skewness.

R-code: how to obtain rv from

$$f(x) = (2\pi)^{-1/2} \exp(-x^2)$$

·Suppose that  $g(x) = \frac{1}{\pi(1+x^2)}$

```
#### SIR I
#### f(target)=Normal , g(x)=Cauchy

set.seed(1234)

SW<-function(x){
  ff<-dnorm(x)
  gg<-dcauchy(x)
  weight<-(ff/gg)/sum(ff/gg)
  return(weight)
}

m<-10000
n<-100

xcau<-rcauchy(m,location=0,scale=1)
ww<-SW(xcau)

x<-
sample(xcau,size=n,replace=TRUE,prob=w
w)

hist(x,probability=TRUE,xlim=c(-5,5))
xx<-seq(-5,5,length=100)
lines(xx,dnorm(xx),type="l",col="red")
```

R-code: how to obtain rv from

$$f(x) = \frac{1}{\pi(1+x^2)}$$

.

Suppose that  $g(x) = (2\pi)^{-1/2} \exp(-x^2)$ .

Then,  $X$  obtained from the SIR algorithm has the distribution that converges to  $f(x)$  as  $m \rightarrow \infty$ .

Q) Why does this hold ?

Q) What is the biggest difference between Rejection algorithm and SIR algorithm ?