STAT 428: Homework 3: Chapter 3 and Chapters 5 Monte Carlo Methods

Du, Yuting, yutingd3 Collaborated with: Sun, Yifan, yifans8

Table of Contents

Please refer to the [**detailed homework policy document**] on Course Page for information about homework formatting, submission, and grading.

Exercise 1

Sampling discrete distributions.

(a) Design an algorithm to simulate from a `Geometric(p)` (where x is the number of failures until the first success) distribution via the inverse transform method. *(Hint: Recall that Geometric random variables are just the number of Bernoulli random variables with the same parameter p until you get a success. So just focus on simulating n Bernoulli variables and then you can transform to Geometric ones.)*

(b) Write R code to generate a sample following the Geometric distribution based on your designed algorithm in the previous part. Use $n = 1000$ sample size and $p = 0.4$. Then, estimate the expected value of this Geometric distribution via Monte Carlo integration to check if your estimates match the theoretically expected value of Geometric distribution or not.

Solution 1

(a) Design an algorithm to simulate from a `Geometric(p)` (where x is the number of failures until the first success) distribution via the inverse transform method. *(Hint: Recall that Geometric random variables are just the number of Bernoulli random variables with the same parameter p until you get a success. So just focus on simulating n Bernoulli variables and then you can transform to Geometric ones.)* generare a Bernoulli(p) rv, then the sicrete inverse-transformation method yields: 1. Generate U~unif(0,1) 2. Set X = 0 if U <= 1-p; X = 1 if U > 1-p Then repeat this process n times and record the first x hitting 1

(b) Write R code to generate a sample following the Geometric distribution based on your designed algorithm in the previous part. Use $n = 1000$ sample size and $p = 0.4$. Then, estimate the expected value of this Geometric distribution via Monte Carlo integration to check if your estimates match the theoretically expected value of Geometric distribution or not.

```
u = numeric()
k = numeric()

geo = function(n,p){
for (i in 1:n) {
  u[i] = runif(1)
  if (u[i] < p)
    k[i] = 0
  else
    k[i] = ceiling(log(1-u[i])/log(1-p))-1#inverse cdf
}
  return(k)
}

mean(geo(1000, 0.4))
```

```
## [1] 1.426
```

```
mean(rgeom(1000,0.4))
```

```
## [1] 1.557
```

Exercise 2

Monte Carlo Integration. Use Monte Carlo integration to estimate

(a)

$$\int_0^2 \cos(x * e^x) dx$$

(b)

$$\int_0^1 \int_0^1 e^{-(x+y)^2} (x^2 + y) dx dy$$

(c)

$$\int_0^4 \int_0^3 e^{-(x+y)^2} (x^2 + y) dx dy$$

Solution 2

(a)

$$\int_0^2 \cos(x * e^x) dx$$

```
u1 <- runif(1000, 0, 2)
x1 <- 2*mean(cos(exp(u1)*u1))
x1

## [1] 0.2863
```

(b)

$$\int_0^1 \int_0^1 e^{-(x+y)^2} (x^2 + y) dx dy$$

```
x <- runif(1000)
y <- runif(1000)
x2 <- mean(exp(-(x+y)^2) * (x^2+y))
x2

## [1] 0.2351
```

(c)

$$\int_0^4 \int_0^3 e^{-(x+y)^2} (x^2 + y) dx dy$$

```r
x <- runif(1000, 0, 3)
y <- runif(1000, 0, 4)
x3 <- 12*mean(exp(-(x+y)^2) * (x^2+y))
x3
```

```
## [1] 0.3857
```

Exercise 3

Variance reduction in Monte Carlo Integration: Antithetic sampling. Let us consider the simple integral

$$\int_0^1 \log(x+1)\ dx.$$

(where $\log(x) = \ln(x)$)

(a) Use Monte Carlo integration with $k = 1000$ iterations to approximate this integral. Call this approximate value I_0.

(b) Find the error of this approximation by repeating this simulation $n = 1000$ times and calculating standard error of this estimator. Call this error E_0. *(Hint: Rememeber standard error = standard deviation of the estimator from multiple simulations.)*

(c) Note that if $U \sim Unif(0,1)$, then $1 - U \sim Unif(0,1)$ also. Obtain a Monte Carlo approximation of the integral using $k_1 = 500$ iterations and uniform distribution *(This step is same as part (a))*. Call this I_{11}. Obtain another MC approximation (call this I_{12}) using $k_2 = 500$ iterations and the antithetic uniform distribution *(Which means in part (a), now use 1-U instead of U.)* Find the average of these two approximations and call it $I_1 = (I_{11} + I_{12})/2$. What is I_1 an approximation of? Compare it to I_0.

(d) Repeat step (c) 1000 times and find the standard error of the estimator I_1. Compare this error with the error E_0 from (b).

Solution 3

(a) Use Monte Carlo integration with $k = 1000$ iterations to approximate this integral. Call this approximate value I_0.

```r
set.seed(123)
u = runif(1000)
I0 = mean(log(u+1))
I0
```

```
## [1] 0.3846
```

(b) Find the error of this approximation by repeating this simulation $n = 1000$ times and calculating standard error of this estimator. Call this error E_0. *(Hint:*

Rememeber standard error = standard deviation of the estimator from multiple simulations.)

```
set.seed(123)
u = runif(1000)
x = numeric()
for(i in 1: 1000){
  x[i] = mean(log(u[i]+1))
}

E0 <- sd(x)/sqrt(1000)
E0

## [1] 0.006231
```

(c) Note that if $U \sim Unif(0,1)$, then $1 - U \sim Unif(0,1)$ also. Obtain a Monte Carlo approximation of the integral using $k_1 = 500$ iterations and uniform distribution *(This step is same as part (a))*. Call this I_{11}. Obtain another MC approximation (call this I_{12}) using $k_2 = 500$ iterations and the antithetic uniform distribution *(Which means in part (a), now use 1-U instead of U.)* Find the average of these two approximations and call it $I_1 = (I_{11} + I_{12})/2$. What is I_1 an approximation of? Compare it to I_0.

```
set.seed(123)
k1 <- 500
u1 <- runif(k1)
I11 <- mean(log(u1+1))

set.seed(124)
k2 <- 500
u2 <- runif(k2)
I12 <- mean(log(1-u2 + 1))

I1 <- (I11 + I12)/2
I1

## [1] 0.3856
```

I1 is similar to I0. I0 can be explained as the expectation of the average of two results in Monte Carlo integration

(d) Repeat step (c) 1000 times and find the standard error of the estimator I_1. Compare this error with the error E_0 from (b).

```
I1 = numeric()
I11 = numeric()
I12 = numeric()


for(i in 1:1000){
```

```
  k1 <- 500
  u1 <- runif(500)
  I11[i] <- mean(log(u1 + 1))

  k2 <- 500
  u2 <- runif(500)
  I12[i] <- mean(log(1-u2+1))


  I1[i] <- (I11[i] + I12[i])/2
}

E1 <- sd(I1)
E1

## [1] 0.006387
```

E1 is similar to E0. Based on the answers on Piazza, no need to divide by sqrt(n)

Exercise 4

Variance reduction in Monte Carlo Integration: Importance Sampling. Note that we can write any integral as

$$\int_0^1 f\,(x)\;dx = \int_0^1 \frac{f(x)}{g(x)}\;g(x)\;dx = E\left[\frac{f(X)}{g(X)}\right],$$

where the last expectation is taken over the reference distribution g. Let $f\,(x) = \log(x+1)$ (where $\log(x) = \ln(x)$). I propose the following reference distribution, $g(x) = (1+\alpha)x^{\alpha},\ 0 \le x \le 1$ with $\alpha = 1.5$. This is sometimes refererred to as the power-law distribution.

(a) Describe an inverse transform algorithm to sample from the proposed distribution $g(x)$.

(b) Implmement your algorithm to get 1000 samples from $g(x)$, $(X_1, X_2, \dots, X_{1000})$.

(c) Compute the quantity $I_2 = \frac{1}{1000}\sum_{i=1}^{1000}\frac{f(X_i)}{g(X_i)}$ for your sample from part (b). What is this quantity I_2 as estimate of? Compare it with I_1.

(d) Repeat steps (b) and (c) 1000 times and find the standard error of the estimator I2. Compare this error with the error E0 from Exercise 3(b).

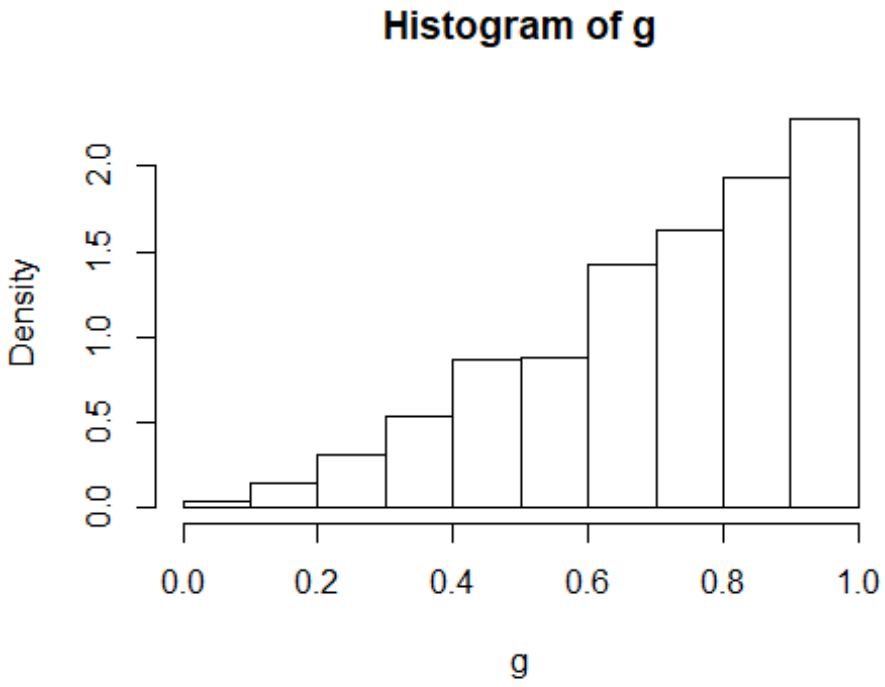
Solution 4

(a) Describe an inverse transform algorithm to sample from the proposed distribution $g(x)$. Inverse transform sampling is a method for generating random numbers from any probability distribution by using its inverse cumulative distribution $F^{-1}(x)$. Recall that the cumulative distribution for a random variable X

is $F_x(x)=P(Xx) $. In what follows, we assume that our computer can, on demand, generate independent realizations of a random variable U uniformly distributed on [0,1]. For continuous distribution, the algorithm is simple as: 1. Generate U ~ Unif(0,1) 2. Let X = F−1X(U) For discrete distribution, the alogorthm is as: 1. Generate U ~ Unif(0,1) 2. Determine the index k such that $\sum_{j=1}^{k-1} p_j \leq U \leq \sum_{j=1}^{k} p_j$, and return $X = x_k$

(b) Implmement your algorithm to get 1000 samples from $g(x)$, $(X_1, X_2, \dots, X_{1000})$.

```
#inverse cdf
set.seed(123)
alpha <- 1.5
u <- runif(1000)
g <- u^(1/(alpha+1))
hist(g, probability = TRUE)
```



(c) Compute the quantity $I_2 = \frac{1}{1000}\sum_{i=1}^{1000} \frac{f(X_i)}{g(X_i)}$ for your sample from part (b). What is this quantity I_2 as estimate of? Compare it with I_1.

```
m = 10000
alpha = 1.5

g = function(x){
  g = log(x+1)

}
```

```
u = runif(m)
x = u^(1/(alpha+1))
gfphi = g(x)/(2.5*x^1.5)
I2 = mean(gfphi)
I2

## [1] 0.3861
```

I2 is estimating $ _0^1 f(x) ; dx$. **(d)** Repeat steps (b) and (c) 1000 times and find the standard error of the estimator I2. Compare this error with the error E0 from Exercise 3(b).

```
m = 10000
alpha = 1.5

fg = function(x){
  f = log(x+1)
  g = (x>0)*(x<1)
  f*g
}


estimates = numeric()

for(i in 1:m){
  x = runif(1000)^(1/(alpha+1))
  gfphi = g(x)/((1+alpha)*x^alpha)
  estimates[i] = mean(gfphi)
}

E2 = sd(estimates)
E2

## [1] 0.004383
```

E2 is smaller.

Exercise 5

Stratified Sampling. Sometimes, it can be beneficial to divide the interval into pieces (strata). Consider

$$\int_{-3}^{3} x^3 \left(e^x - 1\right)\, dx$$

.

(a) Graph the function over the range of interest. Is the function fairly constant? Does it vary? What patterns do you see?

(b) Use standard Monte Carlo integration with $k = 10{,}000$ iterations to approximate this integral. Call this approximate value S_0.

(c) Split the function into six equal strata, (-3,-2), (-2,-1), … and (2,3). Use Monte Carlo integration to approximate the first strata using $k = 10{,}000/6$, and the do the same with the other five strata. Combine the results, and call this approximate value S_1.

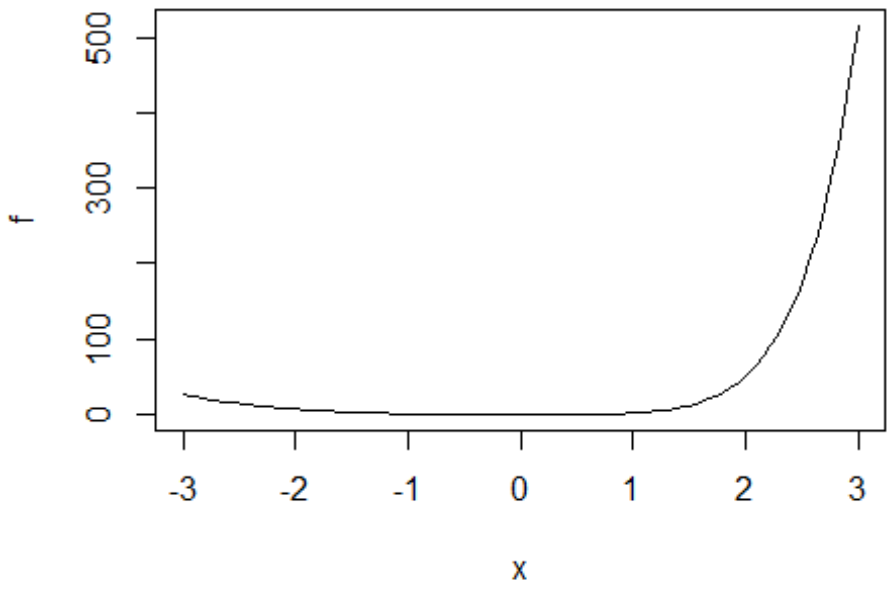
(d) Repeat steps (b) and (c) 1000 times and find the standard error of both estimates. Which estimator is more efficient? Can you think of better places to split the interval to get a more efficient estimator?

Do the following problems from the book: 5.2, 5.4, 5.13, 5.14.

Solution 5

(a) Graph the function over the range of interest. Is the function fairly constant? Does it vary? What patterns do you see?

```
f <- function(x) x^3*(exp(x)-1)
plot(f, from = -3, to = 3)
```



(b) Use standard Monte Carlo integration with $k = 10{,}000$ iterations to approximate this integral. Call this approximate value S_0.

```r
u <- runif(10000, -3, 3)
s0 <- 6*mean(u^3*(exp(u)-1))
s0
```

```
## [1] 250.3
```

(c) Split the function into six equal strata, (-3,-2), (-2,-1), ... and (2,3). Use Monte Carlo integration to approximate the first strata using $k = 10{,}000/6$, and the do the same with the other five strata. Combine the results, and call this approximate value S_1.

```
## [1] 222.3
```

(d) Repeat steps (b) and (c) 1000 times and find the standard error of both estimates. Which estimator is more efficient? Can you think of better places to split the interval to get a more efficient estimator?

```r
n <- 1000
estimates <- matrix(0, n, 2)

m = 10000
k = 6
S1 = numeric(k)

g = function(x){
  x^3*(exp(x)-1)
}

for(i in 1:n){
  estimates[i, 1] = 6*mean(g(runif(m, -3, 3)))

  for(j in -3:3){
  S1[j] = mean(g(runif(m/k, j-1, j)))
  }

  estimates[i, 2] = sum(S1)
}


apply(estimates, 2, mean)
```

```
## [1] 245.0 235.6
```

```r
apply(estimates, 2, var)
```

```
## [1] 31.23 10.59
```

The second method has a smaller variance indicating it is better. Trying to split the interval into smaller pieces may achieve a more efficient estimator.

Exercise 6 (5.2)

Refer to Example 5.3. Compute a Monte Carlo estimate of the standard normal cdf, by generating from the Uniform(0,x) distribution. Compare your estimates with the normal cdf function pnorm. Compute an estimate of the variance of your Monte Carlo estimate of $\Phi(2)$, and a 95% confidence interval for $\Phi(2)$.

Solution 6

```
#refer to example 3.5
x = seq(0.1, 2.5, by = 0.1)
u = runif(10000)

cdf = numeric()
se = numeric()

for (i in 1:length(x)) {
  u = runif(10000, 0, x[i])
  gx = exp(-u^2 / 2)
  cdf[i] = x[i]*mean(gx)/sqrt(2*pi) + 0.5
  se[i] = x[i]*sd(gx)/sqrt(2*pi)/sqrt(10000)
}

#cbind(x, cdf,    phi=pnorm(q = x),se)

cbind(x, cdf,    phi=pnorm(q = x),se)[20,4]^2 # variance for MC estimate
of phi(2).

##         se
## 5.303e-06

cbind(x, cdf, phi=pnorm(q = x),se)[20,2]+cbind(x, cdf, phi=pnorm(q =
x),se)[20,4]*c(-1.96,1.96) # 95% CI for MC estimate of phi(2).

## [1] 0.9735 0.9825
```

Exercise 7 (5.4)

Write a function to compute a Monte Carlo estimate of the Beta(3, 3) cdf, and use the function to estimate F (x) for $x = 0.1,0.2, \ldots 0.9$. Compare the estimates with the values returned by the pbeta function in R.

Solution 7

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

beta <- function(x){
  if(x <= 0) {return (0)}
  if(x >= 1) {return (1)}
```

```
  c <- 1/integrate(f, 0, 1)$val
  u <- runif(2000, 0, x)
  ret <- x*c*mean(f(u))
  return (ret)
}

x <- seq(0.1, 0.9, 0.1)
cdf <- numeric()

for( i in 1:9) {
  cdf[i] <- beta(x[i])
}

true_cdf <- pbeta(x, 3, 3)

data.frame(x, cdf, true_cdf, abs(true_cdf - cdf))
```

```
##     x      cdf true_cdf abs.true_cdf...cdf.
## 1 0.1 0.008308  0.00856           0.0002517
## 2 0.2 0.058873  0.05792           0.0009527
## 3 0.3 0.164424  0.16308           0.0013438
## 4 0.4 0.320383  0.31744           0.0029434
## 5 0.5 0.522018  0.50000           0.0220176
## 6 0.6 0.665760  0.68256           0.0168003
## 7 0.7 0.834583  0.83692           0.0023374
## 8 0.8 0.938652  0.94208           0.0034285
## 9 0.9 0.994620  0.99144           0.0031796
```

```
#similar result
```

Exercise 8 (5.13)

Find two importance functions f1 and f2 that are supported on $(1, \infty)$ and are 'close' to

$$g(x) = \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \quad x > 1$$

Which of your two importance functions should produce the smaller variance in estimating

$$\int_1^{\infty} \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

by importance sampling? Explain.

Solution 8

My choice for the importance fucntion is $f_1(x) = e^{-x} \quad x > 1$ and $f_2(x) = x^2 \quad x > 1$. Since those two importance functions have the similar structure as the target function.

```
m = 1000
theta.hat = numeric()
se = numeric()

gf = function(x){
  g = exp(-x^2/2)/(x^2/sqrt(2*pi))
  f = (x>1)
  g*f
}

##try f1
x = rexp(m, 1)
gfphi = gf(x)/exp(-x)
theta.hat[1] = mean(gfphi)
se[1] = sd(gfphi)/sqrt(m)

##try f2
#use inverse cdf to similarte the f2
u = runif(m)
x = (3*u)^(1/3)
gfphi = gf(x)/(x^2)
theta.hat[2] = mean(gfphi)
se[2] = sd(gfphi)/sqrt(m)



se

## [1] 0.03189 0.01221
```

It seems like $f_2(x) = x^2 \quad x > 1$ has smaller variance which indicating it is closer to the target function.

Exercise 9 (5.14)

Obtain a Monte Carlo estimate of

$$\int_1^{\infty} \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

by importance sampling.

Solution 9

We can change our variable by y = 1/x, then

$$\int_1^{\infty} \frac{x^2}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

will be

$$\int_0^{1} \frac{1}{y^4\sqrt{2\pi}} e^{-\frac{-1}{2y^2}} dy$$

```
n <- 1000
gf <- function(x){
  if(x < 0) {return (0)}
  else if (x>1) {return (0)}
  gf = exp(-1/(2*x^2)) / (sqrt(2*pi)*x^4)
  return (gf)
}

iter <- 0
x <- numeric()

while(iter < n){
  f1 <- rexp(1)
  x[iter] <- gf(f1)
  iter <- iter + 1
}

gfphi = x/exp(-x)
thetheta = mean(gfphi)
thetheta

## [1] 0.4578
```