STAT 428 Homework 2

Yiming Gao (yimingg2)

2017/2/10

Contents

Exercise 1

(a)

$$\theta = \int_0^1 cos(\pi x/2)dx$$

We find a Monte Carlo estimate $\hat{\theta}$ and its standard error using $m = 1000$ random draws with respect to a uniform density (seed = 27).

```
set.seed(27)
m = 1000
x = runif(m, 0, 1)
gx = cos(pi*x/2)
thetahat = mean(gx)
thetahat
```

```
## [1] 0.6281901
```

```
# standard error of thetahat
se = sd(gx)/sqrt(m)
se
```

```
## [1] 0.01000364
```

So we obtain $\hat{\theta} = 0.6282, se(\hat{\theta}) = 0.0100$.

(b)

We construct a stratified Monte Carlo estimate of θ using a total of $m = 1000$ random draws and 4 strata of equal width.

```
set.seed(27)
m = 1000 # number of replicates
k = 4 # number of strata
r = m/k # replicates per stratum
T2 = numeric(k)
g <- function(x){
  cos(pi*x/2)
}

for (i in 1:k){
  T2[i] = (1/k)*mean(g(runif(r, (i-1)/k, i/k)))
```

```
}

# mean of thetahat
estimate = sum(T2)
estimate
```

```
## [1] 0.636028
```

In this stratified Monte Carlo integration, we got

$$\hat{\theta} = 0.6360.$$

(c)

We want to use **numerical integration** to apprximate θ. The result is

$$\hat{\theta} = 0.6366,$$

which is close to that Monte Carlo gives.

```
m = 1000000
xvals = seq(0,1, length.out = m)
midpoints = (xvals[1:m-1]+ xvals[2:m])/2
delta = xvals[2] - xvals[1]
gx = cos(pi*midpoints/2)
thetatilde = sum(gx*delta)
thetatilde
```

```
## [1] 0.6366198
```

(d)

In part c, we have $m - 1$ degree of freedom. Acutually it doesn't matter as long as we have enough sample size ($m = 1000000$).

```
# m-1 intervals
se = sd(gx)/sqrt(m - 1)
U = thetatilde + se*qnorm(1-0.025, 0, 1)
U
```

```
## [1] 0.637223
```

An upper bound for the error of the approximation of θ in part c is 0.6372.

Exercise 2

(a)

We will construct an importance sampling with importance function $\phi(x) = 3(1-x^2)/2$. To draw from this distribution we use the inverse transformation method

$$\Phi(x) = \int_0^x \phi(t)dt = \frac{3}{2}\int_0^x (1-t^2)dt = \frac{3}{2}x - \frac{1}{2}x^3$$

For this CDF, we could **NOT** calculate its inverse function by hand, so I use `uniroot` for this. Then we generate a random sample from $Unif(0,1)$ and transform that according to the inverse CDF of $\phi(x)$.

```
set.seed(27)
# Phi(x)
F <- function(x) (3/2)*x-(1/2)*x^3

# Inverse Phi(x)
F.inv <- function(y){uniroot(function(x){F(x)-y}, interval = c(0, 1))$root}
F.inv <- Vectorize(F.inv)

# generate U~unif(0,1) and X = F.inv(U)
m = 1000
u = runif(m, 0, 1)
x = F.inv(u)

gf <- function(x){
  g = cos(pi*x/2)
  f = (x>0)*(x<1)
  g*f
}

gfphi = gf(x)/(3*(1-x^2)/2)
theta.hat = mean(gfphi)
theta.hat
```

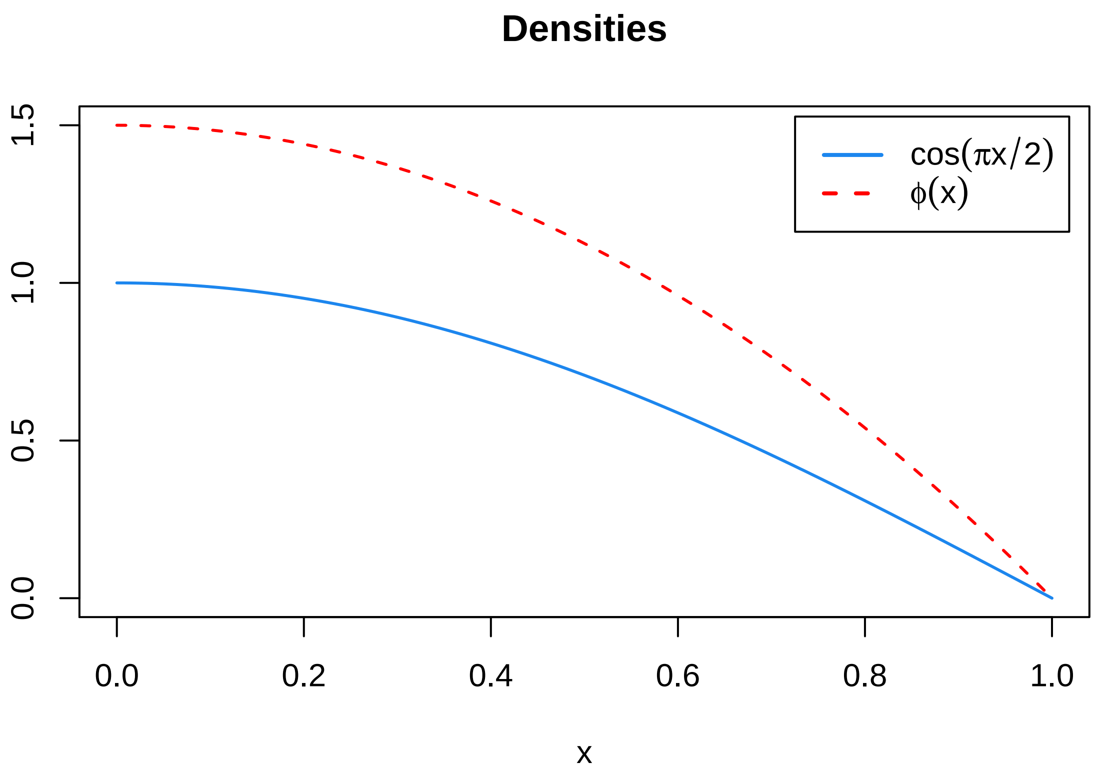
```
## [1] 0.635433
```

The estimate θ^* given by **importance sampling** is 0.6354.

(b)

We plot $\phi(x)$ and $cos(\pi x/2)$ over the interval $(0, 1)$ for comparison with $g(x)$.

```
x = seq(0, 1, 0.001)
phi = 3*(1-x^2)/2
g = cos(pi*x/2)
plot(x, g, type = "l", main = "Densities", ylab = "", ylim = c(0, 1.5), lwd = 1.5,
     col = "dodgerblue2")
lines(x, phi, lty = 2, lwd = 1.5, col = "red")
legend("topright", legend = c(expression(cos(pi *x/2)), expression(paste(phi(x)))),
       lty = 1: 2, lwd = 2, inset = 0.02, col = c("dodgerblue2", "red"))
```



We notice that the importance function $\phi(x) = 3(1 - x^2)/2$ is roughly proportional to $cos(\pi x/2)$ on interval $(0, 1)$ and satisfies $\phi(x) > 0$ for $x \in (0, 1)$.

Exercise 3

In order to tell the difference clearly, we transform the data into scientific notation. Then we repeat (1a) and (2a) 100 times, respectively, to compare the sample variances of the two samples.

```r
m = 1000 # number of replicates
n = 100 # number of experiments
estimates = matrix (0, n, 2)
g <- function(x){
  cos(pi*x/2)
}

set.seed(27)
for (i in 1:n){
  u = runif(m, 0, 1)
  x = F.inv(u)
  estimates[i, 1] = mean(g(u))
  estimates[i ,2] = mean(gf(x)/(3*(1-x^2)/2))
}

# var of thetahat
variances = format(apply(estimates, 2, var), scientific = TRUE)
```

Method	Sample variance
`Monte Carlo`	9.974568e-05
`Importance Sampling`	1.048658e-06

We know that the variance of the estimator computed from importance sampling method is a little smaller than standard Monte Carlo estimator gives, which is an improvement.

Exercise 4

(a)

We use Monte Carlo integration with a uniform distribution in two dimensions to calculate the integral

$$\theta = \int_c^d \int_a^b f(x_1, x_2) dx_1 dx_2$$

by writing the function `function 4a`.

```r
function4a <- function(a, b, c, d, mu, sigma, m){
  x1 = runif(m, a, b)
  x2 = runif(m, c, d)
```

```
  gx = rep(0, m)
  for (i in 1:m){
    # x is a two dimension uniform distribution
    x = c(x1[i], x2[i])
    gx[i] = (1/(b - a)*(d - c))*(2*pi)^(-1)*(det(sigma))^
      (-1/2)*exp((-1/2)*t(x - mu)%*%solve(sigma)%*%(x - mu))
  }
  mean(gx)
}
```

(b)

We use `function 4a` to compute the estimator when $a = 0, b = 1, c = 0, d = 1$,$\mu_1 = \mu_2 = 0, \Sigma = I$ and $m = 10000$. Then we compare it with the true value of θ by using `omxMnor` function in `OpenMx` package.

```
set.seed(27)
a = function4a(a = 0, b = 1, c = 0, d = 1, mu = c(0, 0), sigma = diag(2), m = 10000)

# calculate true integration with OpenMx package
library(OpenMx)
truevalue = omxMnor(diag(2),c(0,0), c(0,0), c(1,1))
data.frame("Simulation" = a, "True value" = truevalue)
```

```
##   Simulation True.value
## 1  0.1168181  0.1165162
```

From the table above, we notice that the estimator gives 0.1168 while true value of θ is 0.1165, which means Monte Carlo integration is very close to the true value.

(c)

This time, we want to directly use random draws from $f(x_1, x_2)$ and integrate the appropriate indicator function. We use `mvrnorm` function in `mvtnorm` package to draw samples from multivariate normal distribution.

When we write the function, we define those points out of the support of indicator functions as **NA**, so that we could calculate the sum of non-NAs' more easily.

```
library(mvtnorm)
function4c <- function(a, b, c, d, mu, sigma, m){
```

```r
  # draw sample from multivatiate normal distribution
  x = mvrnorm(m, mu, sigma)
  for (i in 1:m){
    # for those don't satisty a<x1<b and c<x2<d, we give them NA value
    x[i,] = ifelse((x[i,1]>a)&&(x[i,1]<b)&&(x[i,2]>c)&&(x[i,2]<d), x[i, ], NA)
  }
  (sum(!is.na(x))/2)/m
}
```

(d)

We set the seed 27, the estimation of θ is 0.1138 with this method.

```r
set.seed(27)
function4c(a = 0, b = 1, c = 0, d = 1, mu = c(0, 0), sigma = diag(2), m = 10000)
```

```
## [1] 0.1138
```

Exercise 5 (Rizzo 5.2)

The standard normal cdf

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

First we compute a Monte Carlo estimate of $\Phi(x)$ by generating from the $Uniform(0, x)$, then compare it with the normal cdf function `pnorm`. Meanwhile, we construct a 95% confidence interval for $\Phi(2)$.

```r
function5.2 <- function(x){
  m = 10000
  u = runif(m, 0, x)
  g = (1/sqrt(2*pi))*exp(-u^2/2)
  # exploit the symmetry of normal distribution
  cdf = mean(g)*x + (1/2)
  return(cdf)
}

# set seed 27, x = 2
set.seed(27)
cdf = function5.2(2)
```

```
# true value
phi = pnorm(2)
print(round(rbind(2, cdf, phi), 3))
```

```
##       [,1]
##      2.000
## cdf 0.980
## phi 0.977
```

The Monte Carlo estimate is $\hat{\Phi}(2) = 0.980$, which appears to be very close to the `pnorm` value 0.977.

95% Confidence interval

```
set.seed(27)
m = 10000
u = runif(m, 0, x)
g = (1/sqrt(2*pi))*exp(-u^2/2)
se = sd(g)/sqrt(m)
zval = qnorm(1-0.025, 0, 1)
L = cdf - zval*2*se
U = cdf + zval*2*se
print(round(c(L, U), 3))
```

```
## [1] 0.979 0.981
```

The 95% confidence interval for $\Phi(2)$ is $(0.979, 0.981)$.

Exercise 6 (Rizzo 5.3)

We want to compute a Monte Carlo estimate $\hat{\theta}$ of

$$\theta = \int_0^{0.5} e^{-x} dx$$

by sampling from two distributions and estimate the variance of $\hat{\theta}$.

(a) Uniform(0, 0.5)

```
set.seed(27)
m = 10000
# from Unif(0,0.5)
```

```r
x = runif(m, 0, 0.5)
gx = exp(-x)
thetahat = mean(gx)*0.5
a = thetahat
b = exp(0) - exp(-0.5)

data.frame("Simulation" = a, "True value" = b)
```

```
##   Simulation True.value
## 1  0.3941513  0.3934693
```

```r
# variance of thetahat
format((0.5^2)*var(gx)/m, scientific = TRUE)
```

```
## [1] "3.224917e-07"
```

We get $\hat{\theta} = 0.3942, \hat{Var}(\hat{\theta}) = 3.22e^{-07}$ by sampling from $Unif(0, 0.5)$.

(b) Exponential distribution

```r
set.seed(27)
m = 10000
# mean of exponential distribution is 0.25
y = rexp(m, rate = 1/0.25)
gy = exp(-y)
thetahat.y = mean(gy)*0.5

data.frame("Simulation" = thetahat.y, "True value" = b)
```

```
##   Simulation True.value
## 1   0.400438  0.3934693
```

```r
# variance of thetahat*
format((0.5^2)*var(gy)/m, scientific = TRUE)
```

```
## [1] "6.707444e-07"
```

We get $\hat{\theta}^* = 0.4004, \hat{Var}(\hat{\theta}) = 6.707e^{-07}$ by sampling from $Exp(4)$. Variance of $\hat{\theta}$ (sampling from **Uniform(0, 0.5)**) is smaller.

Exercise 7 (Rizzo 5.4)

(a) Function

We are asked to compute a Monte Carlo estimate of the $Beta(3,3)$ cdf, and estimate $F(x)$ for $x = 0.1, 0.2, ..., 0.9$.

$Beta(3,3)$ has pdf

$$x^2(1-x)^2\frac{\Gamma(6)}{\Gamma(3)\Gamma(3)} = \frac{5!}{2!2!}x^2(1-x)^2 = 30x^2(1-x)^2$$

Its support is $x \in [0,1]$. In order to estimate cdf for $Beta(3,3)$, we should estimate

$$F(x) = 30\int_0^x t^2(1-t)^2 dt$$

We notice that we **cannot** apply the direct algorithm above because the limits of integration change which results in changing the parameters of the uniform distribution for each different value of the cdf required. Suppose that we prefer an algorithm that always samples from $Uniform(0,1)$.

This can be accomplished by a change of variables. Making the substitution $y = \frac{t}{x}$, we have $dt = xdy$ and

$$F(x) = 30\int_0^1 x^3y^2(1-xy)^2 dy$$

Thus, $F(x) = 30E_Y[x^3Y^2(1-xY)^2]$, where the random variable Y has the Uniform(0,1) distribution. Generate iid Uniform(0,1) random numbers $u_1, ..., u_m$, and compute $F\hat(x)$. We write a function `cdf`, which takes `x` and `m` (number of replicates) as arguments.

```
cdf <- function(x, m){
  u = runif(m)
  cdf = numeric(length(x))
  for (i in 1:length(x)){
        g = (x[i]^3)*u^2*(1-x[i]*u)^2
        # don't forget to multiply 30
        cdf[i] = mean(g)*30
  }
  return(cdf)
}
```

(b) Comparison

Then we compare the estimates with the value of cdf computed (numerically) by the `pbeta` function.

```
set.seed(27)
x = seq(0.1, 0.9, 0.1)
m = 10000
cdf = cdf(x, m)
Phi = pbeta(x, 3, 3)
print(round(rbind(x, cdf, Phi), 3))
```

```
##      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9]
## x   0.100 0.200 0.300 0.400 0.500 0.600 0.700 0.800 0.900
## cdf 0.008 0.057 0.162 0.315 0.496 0.678 0.832 0.937 0.988
## Phi 0.009 0.058 0.163 0.317 0.500 0.683 0.837 0.942 0.991
```

Notice that the Monte Carlo estimates appear to be very close to the `pbeta` values.

Exercise 8 (Rizzo 5.14)

We want to obtain a Monte Carlo estimate of

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

by importance sampling. Let $y = \frac{1}{x}$, thus $dx = -\frac{1}{y^2} dy$, and

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

We use $\phi(y) = 1$ as importance function.

```
m = 10000
gf <- function(y){
  g = exp(-1/(2*y^2))*(1/(y^4))
  f = (y > 0)*(y < 1)
  g*f
}

# try importance function = 1
set.seed(27)
y = runif(m)
gfphi = gf(y)
thetahat = mean(gfphi)*(1/sqrt(2*pi))
thetahat
```

```
## [1] 0.3987302
```

```r
# true integral value
integrand = function(x){x^2*exp(-x^2/2)*(1/sqrt(2*pi))}
integrate(integrand, lower = 1, upper = Inf)
```

```
## 0.400626 with absolute error < 5.7e-07
```

The result is 0.3987, which is very close to the true integral value 0.4006.

Bonus

Because it is a two dimension case, we imagine separate the whole area into small squares, then calculate the value of each square's center. At last we calculate the integral by summing up those values time each square area.

```r
# I won't take m too large, it will cost too much time
m = 1000
mu = c(0, 0)
sigma = diag(2)
xvals = seq(0,1, length.out = m)
yvals = seq(0,1, length.out = m)
xmidpoints = (xvals[1:m-1]+ xvals[2:m])/2
ymidpoints = (yvals[1:m-1]+ yvals[2:m])/2

# the area of little square
delta = (xvals[2] - xvals[1])*(yvals[2] - yvals[1])

# create empty matrix
gxy = matrix(rep(0, (m-1)^2), m-1, m-1)

for (i in 1:(m-1)){
  for (j in 1:(m-1)){
    gxy[i,j] = (2*pi)^(-1)*(det(sigma))^(-1/2)*
      exp((-1/2)*((xmidpoints[i])^2 + (ymidpoints[j])^2))
  }
}

thetatilde = sum(gxy*delta)
data.frame("Numerical integration" = round(thetatilde, 8),
           "True value" = round(truevalue, 8))
```

```
##   Numerical.integration True.value
## 1             0.1165162  0.1165162
```

Note that the result of numerical integration is

$$\hat{\theta} = 0.1165,$$

which is almost the same as the true value.