computational statistic lab3

Group 12

11/16/2021

# **Question 1** **Stable distribution**

## q1: plot f(x) and fp(x)

In the following plot, we use c = 1 in ,and . The x is start from 0 to 10.

图表

中度可信度描述已自动生成

From the plot, we can see that there is a problem happened when x < 1(), so only can not be applied to generate samples. So, we decided to use another function to replace power-low function in the interval(0,).

we have

and when

we consider

and when

If we set , then:

To make a density function, we can use .

So, the is :

Plot below shows the plot of when and c=1 in *.*

# 直方图 低可信度描述已自动生成

If we want to use as majorizing density function, we have to make these two curves be similar to each other. So, we pick c = 1.1 . The plot below shows two curves with these parameters

直方图

中度可信度描述已自动生成

## q2: sampling

From question1, we use c = 1.1 .

From previous steps, we know function fmm(x) is the combination of two weighted density function. So if we want to sample from this function, we have to sample from these two density function respectively.

When x ∈ (0, 𝑇𝑚𝑖𝑛), we need to sample from

When x ∈ (0, 𝑇𝑚𝑖𝑛), we need to sample from

As for the proportion, we need to consider the area between curve and x-axis. In fmm(x), when c = 1.1 , the area of left part is:

the area of right part is

So, if we decide to generate **N** samples, then we should have from left part, and from right part. Then, we can use functions(runif() and poweRlaw::rplcon()) to get samples.

To apply acceptance/rejection method, we need to find majorizing constant (will be called mc in the following text)first.

1) when , so when x=, we have mc = 1.003

2) when , so when x=, we have mc = 1.537

## q3: sampling for different majorizing constant.

In this section, we have a list for c, 1, 1.1, 2.5 ,3.

Since the data is very large, the result will be scattered in (0,) (e.g. some data is like 0,01 and some data will reach scale), we set a boundary for plotting (0,40).

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

图表, 直方图

描述已自动生成

The table below shows the variance, mean and reject rate regarding different parameter c in f(x).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | C =1 | C =1.1 | C =2.5 | C =3 |
| Mean | 79863.653 | 5974.723 | 167184.433 | 360583.365 |
| Variance | 6.055782e+13 | 3.128015e+10 | 1.530487e+14 | 4.656850e+14 |
| Reject rate | 0.4395720 | 0.4578229 | 0.4224211 | 0.4550728 |

Mean value will increase with c increase.

Variance shares the similar trend with mean.

Reject rate, I think, will not only depends on c but also many aspects. All the parameters in the power-law function and the replace function will have great impact on reject rate.

# **Question 2** **Laplace distribution**

## q1: Generate DE distribution

step 1: get the formula

Since the target distribution is DE(0,1), so the formula is:

Let Y from Unif(0,1), according to the inverse CDF method, we set that :

Solve this formula, we can get:

We need x both negative and positive, so we remove absolute sign in this way:

step2: get x

Using function runif() to generate 10000 samples, we can take these samples as y, then get the generated samples.

After generating the samples, we can draw the histogram. Since these samples are randomly generated and computed from function Unif(), the results differ each time. The Laplace density function LA(0,1) should have mean value 0 and variance 2.

The histogram is like below(10000 samples with mean value -0.00934368, variance 1.989046)

图表, 直方图

描述已自动生成

## q2: Generate normal distribution

step 1: get constant C

Since we must make sure that for every valid x, c\*DE(x) >N(x).

we move x to right side and remain only constant c in the left side:

X = 1 is valid, and when x = 1, the right side reach the maximum value . So, we will get:

So, we will choose C = 1.3155.

step 2: get sample from DE (0,1)

We already get good result in q1, so we just pick 2000 samples from vector ‘x’ in q1.

step3: apply Acceptance/Rejection method

In this step, we will have a loop runs 2000 times. Each loop will pick a sample from what we have in step2(we will call this sample x in the following text), a random sample from function runif (like runif(1)) and a random value generated by dnorm(x). Then:

Applying the data from **q1** in these three steps, we have the reject number: 478. So, the reject rate is 0.239.By generating the sample 300 times, we learn that average reject rate is 0.242 The estimated reject rate is 0.2398 (1/C), which is very close to our result.

The histogram of these 2000 samples is:

图表, 直方图

描述已自动生成

The histogram of samples from function rnorm() is:

图表, 直方图

描述已自动生成

These two histograms show that samples from Acceptance/Rejection method has similar distribution as samples from normal distribution.

# **Appendix:**

## Question 1

library(ggplot2)  
library(reshape2)

library(poweRlaw)

get\_fpx\_origin <- function(alpha,t\_min,x)  
{  
 x <- x[order(x)]  
 s <- x[which(x<t\_min)]  
 x <- x[-which(x<t\_min)]  
 res <- rep(0,length(s))  
 res <- c(res,(alpha-1)/t\_min\*(x/t\_min)^(-alpha))  
 return(res)  
}  
get\_fpx <- function(alpha,t\_min,x)  
{  
 x <- x[order(x)]  
 s <- x[which(x<t\_min)]  
 x <- x[-which(x<t\_min)]  
 res <- rep((alpha-1)/t\_min,length(s))  
 res <- c(res,(alpha-1)/t\_min\*(x/t\_min)^(-alpha))  
 res <- res/alpha  
 return(res)  
}  
# plot for f(x) and fp(x)  
c <- 1  
t\_min <- 1  
alpha <- 2  
x\_f <- 1:1000/200  
fx <- c/sqrt(2\*pi)\*exp(-c\*c/2/x\_f)\*x\_f^(-3/2)  
fpx <- get\_fpx\_origin(alpha,t\_min,x\_f)  
df <- data.frame(x = x\_f,fx =fx,fpx=fpx)  
df1 <- melt(df,id.vars='x')  
  
p1 <-ggplot(df1,aes(x=x,y=value))+  
 geom\_point(aes(color=variable))+  
 ggtitle('f(x) and fp(x) when alpha = 2 T\_min = 1')+  
 theme(plot.title = ggplot2::element\_text(hjust=0.5))  
print(p1)  
# plot for fmm(x) and f(x)  
c <- 1.1  
t\_min <- 0.45  
alpha <- 1.2  
x\_f <- 1:1000/200  
fx <- c/sqrt(2\*pi)\*exp(-c\*c/2/x\_f)\*x\_f^(-3/2)  
fpx2 <- get\_fpx(alpha,t\_min,x\_f)  
df <- data.frame(x = x\_f,fx =fx,fmmx=fpx2)  
df2 <- melt(df,id.vars='x')  
# plot  
p2 <-ggplot(df2,aes(x=x,y=value))+  
 geom\_point(aes(color=variable))+  
 ggtitle('f(x) and fmm(x) when alpha = 1.2 T\_min = 0.45,c = 1.1')+  
 theme(plot.title = ggplot2::element\_text(hjust=0.5))  
print(p2)  
################################################################################  
#q2  
new\_powerLaw\_sampling <- function(nsamples,alpha,t\_min)  
{  
  
 n1 <- nsamples\*(alpha-1)/(alpha)  
 n2 <- nsamples - n1  
 first\_part <- runif(n1,0,t\_min)  
 second\_part <- rplcon(n2,t\_min,alpha)  
 return(c(first\_part,second\_part))  
}  
get\_sample <- function(mc,new\_samples,alpha,t\_min,c\_in\_fx){  
 accept\_sample <- c()  
 for (i in 1:length(new\_samples))   
 {  
 y <- new\_samples[i]  
 u <- runif(1)  
 fxy <- c\_in\_fx/sqrt(2\*pi)\*exp(-c\_in\_fx\*c\_in\_fx/2/y)\*y^(-3/2)  
 if(y <= t\_min)  
 {  
 fyy <- (alpha-1)/t\_min/alpha  
 }else  
 {  
 fyy <- ((alpha-1)/alpha/t\_min)\*(y/t\_min)^-alpha  
 }  
 if(u <= fxy/(fyy\*mc))  
 {accept\_sample <- c(accept\_sample,y)}  
 }  
 return(accept\_sample)  
}  
new\_samples <- new\_powerLaw\_sampling(20000,alpha,t\_min)  
res <- get\_sample(1.8433,new\_samples,alpha,t\_min,c)  
print(1-length(res)/length(new\_samples))  
################################################################################  
#q3  
c\_in\_fx\_list <- c(1,1.1,2.5,3)  
t\_min <- 0.45  
alpha <- 1.2  
reject\_rate <- c()  
mean <- c()  
variance <- c()  
for (i in c\_in\_fx\_list) {  
 # get samples  
 new\_samples <- new\_powerLaw\_sampling(20000,alpha,t\_min)  
 x1 <- 1/(3-2\*alpha)   
 mc1 <- (i/sqrt(2\*pi)\*exp(-i\*i/2/(x1))\*(x1)^(-3/2))/((alpha-1)/t\_min/alpha\*(x1/t\_min)^(-alpha))  
 mc2 <- (i/sqrt(2\*pi)\*exp(-i\*i/2/(1/3))\*(1/3)^(-3/2))/((alpha-1)/t\_min/alpha)  
 res <- get\_sample(max(mc1,mc2,1),new\_samples,alpha,t\_min,i)  
 # get some data  
 reject\_rate <- c(reject\_rate,(length(new\_samples)-length(res))/length(new\_samples))  
 mean <- c(mean,mean(res))  
 variance <- c(variance,var(res))  
 # plot histogram   
 hist(res[which(res<=40)],xlab = 'x',main = paste0('histogram when c(parameter of f(x)) is',i),breaks = 400)  
}

print(mean)  
print(variance)  
print(reject\_rate)

## Question 2

library(ggplot2)  
library(reshape2)

library(poweRlaw)

# q1  
unif\_sample <- runif(10000,0,1)  
DE\_sample <- c()  
for (i in 1:10000)  
{  
 if(unif\_sample[i] > 0.5)  
 DE\_sample <-c(DE\_sample,log(2-2\*unif\_sample[i]))  
 else  
 DE\_sample <-c(DE\_sample,-log(2\*unif\_sample[i]))  
}

# interval [n\*gap,(n+1)\*gap] will be used in the histogram  
gap <- (max(DE\_sample)-min(DE\_sample))/200  
# count will be vector used to make histogram  
count <-c()  
start <- min(DE\_sample)  
end <- min(DE\_sample) + gap  
for(i in 1:200)   
{   
 temp <- DE\_sample[which(DE\_sample >= start)]  
 temp <- temp[which(temp < end)]  
 count <- c(count,length(temp))  
 start <- start + gap  
 end <- end + gap  
}  
barplot(count,xlab = 'x',ylab = 'y',main = 'generated DE sample')

# q2  
#  
c = 1.3155  
index <- 1:10000  
DE\_2000 <- DE\_sample[which(index%%5==0)]  
reject\_num <- 0  
accept\_sample <- c()  
#for(j in 1:300){  
for (i in 1:2000)   
{  
 y <- DE\_2000[i]  
 # fxx is normal density at x  
 u <- runif(1)  
 fxy <- dnorm(y)  
 fyy <- 1/2 \*exp(-abs(y))  
 if(u <= fxy/(fyy\*c))  
 {accept\_sample <- c(accept\_sample,y)}  
 else  
 {  
 reject\_num <- reject\_num + 1  
 }  
}  
#}  
#reject\_num/300/2000  
# ER is 1-1/c  
gap <- (max(accept\_sample)-min(accept\_sample))/80  
# count will be vector used to make histogram  
count <-c()  
start <- min(accept\_sample)  
end <- min(accept\_sample) + gap  
for(i in 1:80)   
{   
 temp <- accept\_sample[which(accept\_sample >= start)]  
 temp <- temp[which(temp < end)]  
 count <- c(count,length(temp))  
 start <- start + gap  
 end <- end + gap  
}  
barplot(count,xlab = 'x',ylab = 'y',main = 'generated normal sample')

r<-rnorm(2000)  
# ER is 1-1/c  
gap <- (max(r)-min(r))/80  
# count will be vector used to make histogram  
count <-c()  
start <- min(accept\_sample)  
end <- min(accept\_sample) + gap  
for(i in 1:80)   
{   
 temp <- r[which(r >= start)]  
 temp <- temp[which(temp < end)]  
 count <- c(count,length(temp))  
 start <- start + gap  
 end <- end + gap  
}  
barplot(count,xlab = 'x',ylab = 'y',main = 'generated normal sample from rnorm')