Computational Statistics Assignment 2

Kyriacos Xanthos
CID: 01389741

November 22, 2021

Question 1

For this question we consider a density of the form:

$$f(x) = \frac{k}{1+ \mid x-2 \mid^3}, \quad x \in [0,5] \tag{1}$$

where k is a normalising constant. To construct a Metropolis-Hastings sampler to sample from this density f using a random walk proposal with additive noise following a normal distribution with standard deviation σ, we consider:

$$Y \mid X_{t-1} \sim \mathcal{N}\left(X_{t-1}, \sigma^2\right) \tag{2}$$

for a time step t.

a)

Now we estimate $E(X^3)$ with Monte Carlo Markov Chain algorithm (MCMC) to be 13.9773. Comparing this value with the true value using integration, we see that the true value is 14.1732. Note that to determine the true value we needed to find the normalizing constant $k = 0.4455$ by equating the integral of the function to 1 and integrating over all space. Using the same procedure we estimate $P(X < 1)$ using MCMC to be 0.1210 and comparing with the true value in numerical integration, 0.1133. This shows that both of our estimates with MCMC were good estimates.

b)

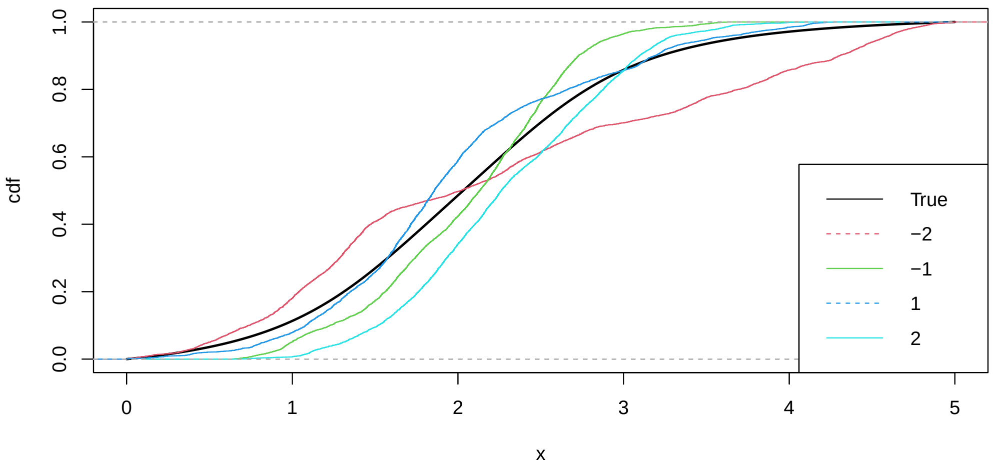

Figure 1: True CDF and estimated CDF using MCMC for function defined in 1. The legend shows the colors for each starting value.

Now we create four independent Markov chains for different initial values, namely 1,2,3,4. We run the sampler for 5000 iterations and compare the empirical Cumulative Distribution Function (CDF) obtained through integration with these samplers.

Figure 1 shows that all of the starting values for MCMC although they roughly follow the trend of the true CDF, there seems to be a large difference between them. To explore this more, we created traceplots for each starting value to examine the convergence. This is seen in figure 2 and it is clear that in 5000 iterations the MCMC is not converging and there is a large deviation between different starting values. This reinforces our understanding of the CDF figure since the lines were a bit far off. This problem can be addressed by running the MCMC for more iterations with the hope of getting better convergence.

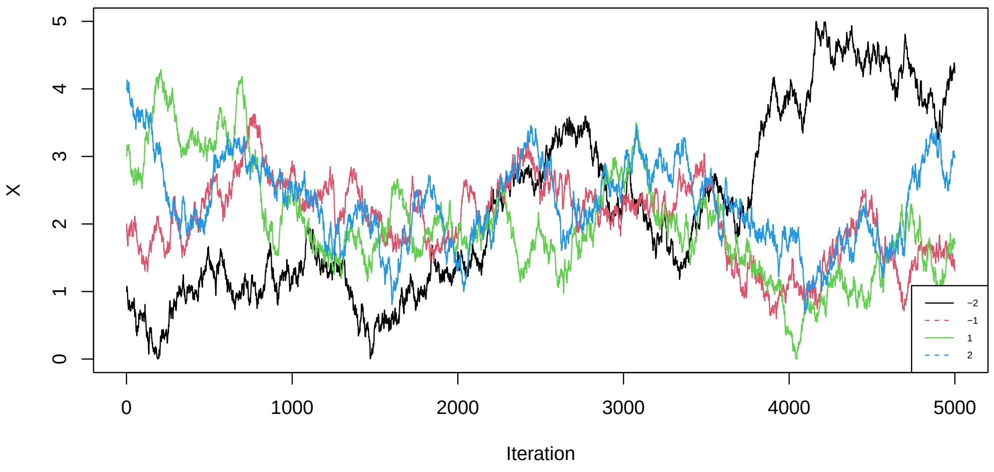

Figure 2: Trace plots of different starting values for an MCMC estimation of function 1.The legend shows the colors for each starting value.

c)

Now we explore the acceptance rates for different σs. The acceptance rate is the fraction of proposal moves that are accepted in the MCMC. For $\sigma \in \{0.05, 1, 5\}$ we get the acceptance rates 0.978, 0.663, 0.213 respectively. For the random walk proposal we interpret that if we have an acceptance rate too high, like 0.978, the support of f is not explored quickly since we are accepting almost all the moves. For low dimensional models like our 1-dimensional model in this question we expect a heuristic acceptance rate of around 1/2. [1]. The resulting stationary distributions as seen from figure 2 are not really converging so we need more iterations to explore the stationary distributions.

Question 2

In this question we explore the iid sample from the random variable X:

$$X = [-0.71, -1.30, -0.13, -2.03, 1.62, 2.38, 0.48, 0.51, -0.69, -2.32, -2.02, 1.23, -0.25, 0.76, 0.65]$$

a)

We will use the plug-in principle [1], to estimate the confidence intervals of $E[\cos x]$ for the X iid sample.

For the standard 95% confidence interval we need c_1, c_2 such that:

$$\mathrm{P}\left(c_1 \leq T(\hat{\mathrm{P}}) - T(\mathrm{P}) \leq c_2\right) = 1 - \alpha \tag{3}$$

where in our example P is the unknown probability distribution and $\hat{P}$ the estimated probability distribution from the sample X, $T = E[\cos x]$, $\alpha = 0.05$. Since we do not know the exact values of of c_1, c_2 (because we do not have access to the full data) we approximate them with c_1^*, c_2^* which we obtain using bootstrap. The resulting confidence interval will be of the form: $\left[T(\hat{P}) - c_2^*, T(\hat{P}) - c_1^*\right]$. The numerical values we get are [0.0257,0.6887].

For the Studentised 95% confidence interval we need c_1, c_2 such that:

$$\mathrm{P}\left(c_1 \leq \frac{T(\hat{\mathrm{P}}) - T(\mathrm{P})}{\sigma(\hat{\mathrm{P}})} \leq c_2\right) = 1 - \alpha \tag{4}$$

where $P, \hat{P}, \alpha$ and T are defined as above, and $\sigma(\hat{P})$ is $\frac{\sqrt{Var(\cos x)}}{\sqrt{N}}$ where N is the number of samples. So the confidence interval will be of the form $\left[T(\hat{P}) - c_2^*\sigma(\hat{P}), T(\hat{P}) - c_1^*\sigma(\hat{P})\right]$. The numerical result is [-0.0628 0.6708] which is wider than the standard confidence interval as expected.

b)

To examine the coverage probability of these confidence intervals, we first need to compute the true value. For the Student's t distribution with ν degrees of freedom this is:

$$E(\cos x) = \int_{-\infty}^{\infty} \cos(x) \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)} \left(1 + \frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}} \tag{5}$$

Now looking at the number of times the confidence interval includes the true value (0.507 using integration) when computing 500 different confidence intervals we can draw conclusions for the coverage probability. For the standard confidence interval we get a coverage probability of 86.8% which is much lower than the 95% that we wanted but for the studentised confidence interval we get 95.4% which is much closer to the 95% target.

c)

For this question we now change our test statistic to $T\left(X_1 \ldots X_n\right) = \text{median}\left(\cos\left(X_1\right) \ldots \cos\left(X_n\right)\right)$. The standard two sided confidence interval will be calculated in the same way as part b) with this new test statistic as the only difference. Computing the confidence interval we get $[0.654, 1.883]$.

For the studentised two sided confidence interval we will need to construct a nested bootstrap algorithm since we cannot now directly calculate $\sigma(\hat{\mathrm{P}})$. This means that for each bootstrap sample x^* we need to estimate a distribution for the variance again using bootstrap samples. I did not have time for the implementation but I included the unfinished code in the appendix.

Question 3

We consider a function of the form:

$$f(x, y) = ky^{1/2} \exp\left\{-\frac{x^2}{2} - \frac{y}{2}\left(1 + (a_1 - x)^2 + (a_2 - x)^2\right)\right\} \text{ for } y \geq 0 \tag{6}$$

where k > 0 is a normalising constant and $a_1 = 4, a_2 = 1$. Now in order to use Gibbs sampling we need to consider the conditional distributions of 6. Note that

$$f_{X|Y}(x) = \frac{f_{X,Y}(x, y)}{f_Y(y)} \propto f_{X,Y}(x, y) \tag{7}$$

So we will just be concerned with the conditional distributions up to a constant. We calculate:

$$f_{Y|X}(y) \propto y^{1/2} \exp\left\{\frac{y}{2}\left(1 + (a_1 - x)^2 + (a_2 - x)^2\right)\right\} \tag{8}$$

and

$$f_{X|Y}(x) \propto \exp\left\{-\frac{x^2}{2} - \frac{y}{2}\left((a_1-x)^2 + (a_2-x)^2\right)\right\} \tag{9}$$

We can observe that the conditional distribution in 8 $\sim \Gamma(\text{shape}=\frac{3}{2}, \text{rate}=\frac{1}{2}(1+(a_1-x)^2+(a_2-x)^2))$. Also, by completing the square in x in equation 9 and dropping all constants, we see that 9 $\sim \mathcal{N}\left(\mu = \frac{(a_1+a_2)y}{1+2y}, \sigma = \frac{1}{1+2y}\right)$.

This suggests the two following Gibbs samplers:

$$A_1 \sim \Gamma\left(\frac{3}{2}, \frac{1}{2}(1+(a_1-x)^2+(a_2-x)^2)\right) \quad A_2 \sim \mathcal{N}\left(\frac{(a_1+a_2)y}{1+2y}, \frac{1}{1+2y}\right) \tag{10}$$

With $a_1 = 4, a_2 = 1$, we obtain the following scatterplot:

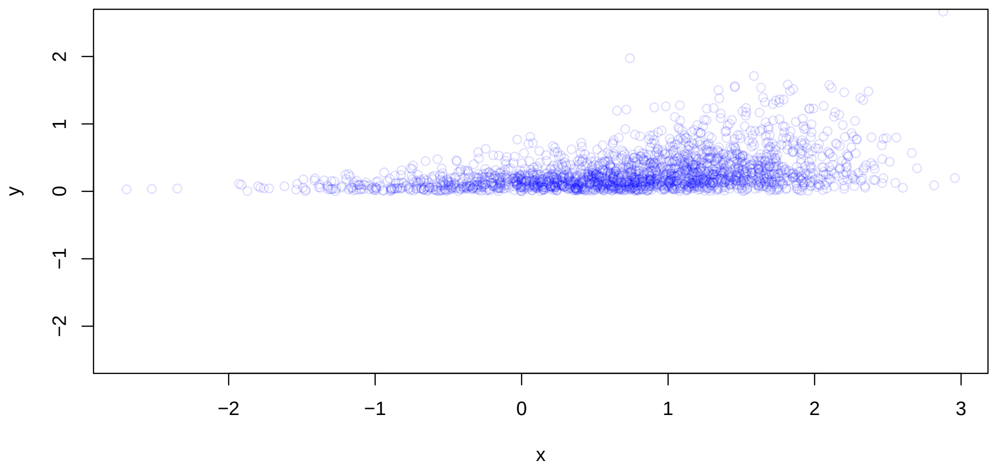

Figure 3: Scatterplot of the approximate sample from the joint distribution in 6.The opacity of the points shows where most of them are concentrated.

Note that as expected in figure 3 we have no values for $Y < 0$, and we observe a clear pattern for this joint distribution.

To estimate $P((X,Y) \in [-1,1] \times [0,0.5])$ we use the ergotic theorem [1], to obtain

$$P((X,Y) \in [-1,1] \times [0,0.5]) \approx \frac{1}{10^4}\sum_{i=1}^{10^4} 1_{\left(X^{(i)},Y^{(i)}\right)\in[-1,1]\times[0,0.5]} \approx 0.500 \tag{11}$$

To approximate $E(X^2)$ we simply find the mean of the squared samples X from the approximate distribution we have computed above. The numerical value is 1.305

Question 4

We have data from two different groups, group 1 and group 2 corresponding to 50 and 75 observations respectively. The null hypothesis is that they have the same variance.

a)

We consider a test statistic of the form:

$$T(x,y) = \mid \text{Var}(x) - \text{Var}(y) \mid \tag{12}$$

and we device a permutation test to check the null hypothesis. We create 1000 permutations to obtain different values for the test statistic and we then compare the test statistics with the test statistic of the actual observations. This gives a p value of 0.096 with a 95% confidence interval of [0.090 0.102]. This shows that we do not have sufficient evidence to reject the null hypothesis that that the two groups have the same variance at a 5% level.

b)

Now we will assume that the data have been sampled independently from a Gamma distribution with shape parameter k=5 and unknown scale parameters θ_1 and θ_2 for groups 1 and 2 respectively. This means we can use parametric bootstrap to estimate the distributions of the sample variance in each group. This will be done by approximating $\theta_{1,2}$ by the mean of the data in the groups divided by the shape parameter. Using these $\hat{\theta_{1,2}}$ we will sample from a gamma distribution with shape parameter k and obtain Bootstrap samples of the distributions for the two groups. Replicating this procedure 1,000 times we get a figure 4 for the empirical cdf of the resulting distributions.

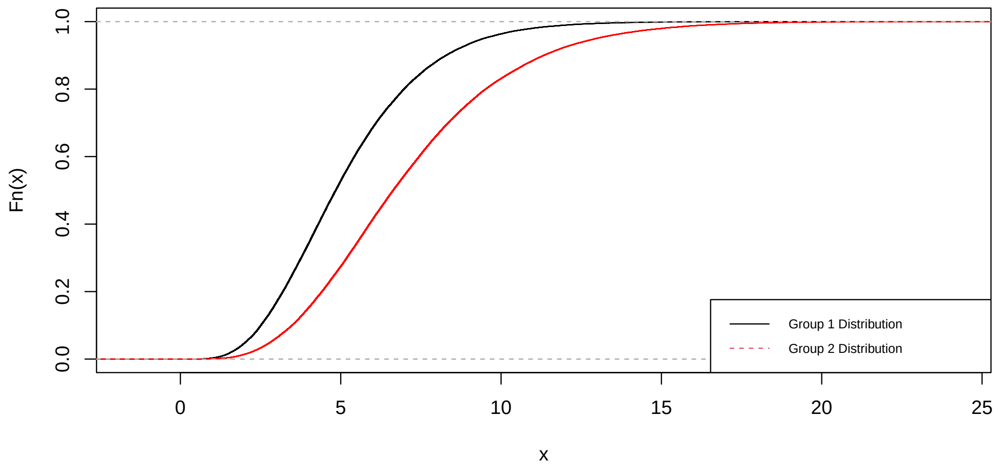

Figure 4: Empirical CDF of distributions of Group 1 and Group 2 using a bootstrap sample size 1,000.

Now to compute a parametric bootstrap test we consider a p value of the form:

$$\hat{p} = \frac{1 + \sum_{i=1}^{B} I\left(T\left(\mathbf{X}^{*i}\right) \geq T(\mathbf{x})\right)}{B+1} \tag{13}$$

where B are the number of Bootstrap samples, I is the indicator function, T is the test statistic 12 ,and $\mathbf{X}^*$ are the iid bootstrap samples generated from the distribution and plotted in 4.

We evaluate the test statistic on these generated bootstrap samples (we use 10,000 replications) and compare the test statistic out of these samples with the test statistic of the original data, just as we did in part **a)** of this question. We now obtain $\hat{p} = 0.032$ suggesting that now we have sufficient evidence to reject the null hypothesis that these two groups have the same variance.

In parametric bootstrap we expect the estimates to be more accurate if the parametric assumptions are correct, ie. in our case that these data have been sampled independently from the given Gamma distributions. If this is indeed correct, the estimate for the p value is more accurate than the one we obtained from permutation tests. In general, it is best to rely on permutation tests for hypothesis testing since if the basis for permuting the data is a random assignment, then the p-value obtained using permutation tests is *exact* as long as all possible permutations are considered. In our example the p value of permutation test was smaller and therefore will be considered more accurate than the bootstrap approach.

References

[1] Dr Sarah Filippi. Lecture notes. *Imperial College London*, pages 38–68, October 2021.

A Code for Question 1

```
set.seed(222)

my_f <- function(x) ifelse((x>=0 & x<=5), 1/(1+abs(x-2)^3), 0)

MH <- function(f,q,rq,n,init){
  xall <- rep(NA,n)
  x <- init
  xall[1] <- x
  n_acc <- 0 #acceptance rate
  for (t in seq(2,n)){
    y <- rq(x)
    # if (runif(1)<=(f(y)*q(y,x))/(f(x)*q(x,y))){
    if (runif(1)<=(f(y))/(f(x))){
      x <- y
      n_acc <- n_acc+1
    }
    xall[t] <- x
  }
  cat('Acceptance rate: ', n_acc/n)
  return(xall)
}

sigma=1
N=5000
q <- function(x,y) dnorm(y,mean=x,sd=sigma)
rq <- function(x) x+rnorm(1, mean=0, sd=sigma)
f <- my_f
set.seed(123)
xall <- MH(f,q,rq,N,1)

sigma=0.05
xall_1 <- MH(f,q,rq,N,1)
sigma=1
xall_2 <- MH(f,q,rq,N,1)
sigma=5
xall_3 <- MH(f,q,rq,N,1)

plot(xall_1,type='l')
lines(xall_2, col='red')
lines(xall_3, col='blue')
ex2 <- mean(xall^3)
ex2

k <- 1/integrate(f,-Inf,Inf)$value

f3 <- function(x) ifelse((x>=0 & x<=5), k*x^3/(1+abs(x-2)^3), 0)


integrate(f3,0,5)$value
```

```
mean(xall<1)

f3 <- function(x) ifelse((x>=0 & x<=5), k/(1+abs(x-2)^3), 0)

integrate(f3,0,1)$value
plot(xall, type='l')

#b
# N=1e4
set.seed(222)
my_chains <- matrix(rep(NA,N*4), nrow = N, ncol = 4)
count <- 0
for (init in c(1,2,3,4)){
  count = count +1
  sigma=0.05
  my_chains[,count] <- MH(f,q,rq,5000,init)

}

plot(my_chains[,1],type='l', ylab='X', xlab='Iteration')
for (i in 2:4){
  print(i)
  lines(my_chains[,i], col=i)
}
legend('bottomright', legend=c(-2,-1,1,2), col=c(1,2,3,4), lty=1:2, cex=0.5)

x <- seq (0 ,5 ,0.01)
cdf <- sapply(x, function(a) k*integrate(f, lower=0, upper=a)$value)

plot(x,cdf, type='l', lwd=2)
lines(ecdf(my_chains[,1]), col=2)
lines(ecdf(my_chains[,2]), col=3)
lines(ecdf(my_chains[,3]), col=4)
lines(ecdf(my_chains[,4]), col=5)
legend('bottomright', legend=c('True',-2,-1,1,2),
col=c(1,2,3,4,5), lty=1:2, cex=1)
```

B Code for Question 2

```
set.seed(222)
X <- c(-0.71, -1.30, -0.13, -2.03, 1.62, 2.38, 0.48, 0.51,
-0.69, -2.32, -2.02,
       1.23, -0.25, 0.76, 0.65)

#two sided interval
two_sided_boot <- function(X,b,alpha){
  bootsamples <- replicate(b,{
    bX <- sample(X,replace=TRUE)
    T(bX)-T(X)
  })
  cstar <- quantile(bootsamples,c(alpha/2,1-alpha/2))
  return(c(T(X)-cstar[2], T(X)-cstar[1]))
}

stud_two_sided_boot <- function(X,b, alpha){
  bootsamples <- replicate(b,{
    bX <- sample(X,replace=TRUE)
    (T(bX)-T(X))/(sdT(bX))
  })
  cstar <- quantile(bootsamples,c(alpha/2,1-alpha/2))

  return(c(T(X)-sdT(X)*cstar[2],
           T(X)-sdT(X)*cstar[1]))

}

T <- function(x) mean(cos(x))
sdT <- function(x) sd(cos(x))/sqrt(length(x))



b=1000
alpha = 0.05

set.seed(1222)
two_sided_boot(X,b,alpha)
stud_two_sided_boot(X,b,alpha)


r <- function(n){rt(n,4)}
##work out correct value
mean(cos(r(1e6)))#with Monte Carlo

ff <- function(x) cos(x)*gamma(5/3)/(sqrt(4*pi)*gamma(2))*(1+(x^2)/4)^(-5/2)

trueres <- integrate(ff,-Inf,Inf)$value
# trueres <- mean(cos(r(1e6)))

n <- 15
CI_bootnp <- replicate(500,two_sided_boot(r(n),1e3, alpha))
mean((CI_bootnp[1,]<=trueres) &(CI_bootnp[2,]>=trueres))

CI_bootnp_stud<- replicate(500,stud_two_sided_boot(r(n),1e3,alpha))
mean((CI_bootnp_stud[1,]<=trueres) &(CI_bootnp_stud[2,]>=trueres))


# C
T <- function(x) median(cos(x))
sdT <- function(x) sd(median(cos(x)))


b=1000
alpha = 0.05

set.seed(1222)
two_sided_boot(X,b,alpha)


B=10
T <- c()
for (i in range(1:b)){
  bX <- sample(X,replace=TRUE)
  for (j in range(1:100)){
    print(cos(bX[j]))
    T[j] <- median(cos(bX[j]))
    print(T[j])

  }
}
```

C Code for Question 3

```
set.seed(222)
a=4
b=1
A1 <- function(y) ifelse(y>=0, rnorm(1, mean=(a+b)*y/(1+2*y),
sd=1/(1+2*y)),0)

A2 <- function(x) rgamma(1, shape=3/2, rate=1/2*(1+(a-x)^2+(b-x)^2))

n <- 2000
xall <- matrix(NA,ncol=2, nrow=n)
x <- c(0,0)
xall[1,] <- x
for (t in seq(2,n)){
  x[1] <- A1(x[2])
  x[2] <- A2(x[1])
  xall[t,]<-x
}

count = 0
for (i in 1:dim(xall)[1]){
  if ((xall[i,1] <= 1 & xall[i,1] >=-1 ) & (xall[i,2] <= 0.5 &
  xall[i,2] >=0)){
    count = count + 1
  }
}

count/n

mean(xall[,1]^2)

plot(xall[,1], xall[,2], col=rgb(red=0, green=0, blue=1, alpha=0.14),
     xlab='x', ylab= 'y', ylim=c(-2.5,2.5))
```

D Code for Question 4

```
set.seed(222)
dat <- read.csv('qu4.csv')
x <- dat$observations[dat$group==1]
y <- dat$observations[dat$group==2]

n <- length(x)
m <- length(y)

T <- function(Z) abs(var(Z[1:n]) - var(Z[(n+1):(n+m)]))
```

```
Z <- c(x,y)
t <- T(Z)
Zperm <- sample(Z,length(Z))
Trep <- replicate(1e4, T(sample(Z,length(Z)))>=t)
pval <- mean(Trep)
pval

pval+sd(Trep)/sqrt(length(Trep))*qnorm(c(0.025,0.975)) ##Conf Int

#b
k=5
genbootsample <- function(data){
  theta_hat <- mean(data)/k
  rgamma(length(data), shape=k, scale=theta_hat)
}
m1 <- replicate(1000,genbootsample(x))
m2 <- replicate(1000,genbootsample(y))

plot(ecdf(m1), main='')
lines(ecdf(m2),col='red')
legend('bottomright', legend=c('Group 1 Distribution',
              'Group 2 Distribution'), col=c(1,2), lty=1:2, cex=0.7)
B <- 1e4
Trep <- replicate(B, {T(genbootsample(c(x,y)))})

pval_boot <- mean(c(Trep, t)>=t)
pval_boot
```