732A90: Lab 4
Computational Statistics, Group 6

Chih-Yuan Lin, Sarah Walid Alsaadi, Carles Sans Fuentes, Joshua Burrata

March 3, 2017

Question 1

In this exercise, we are supposed to compute the Metropolis-Hastings algorithm from a target distribution which is

$$f(x) \propto x^5 * exp^{-x}$$

. In order to evaluate whether we can get a good sample points from our algorithm, we are going to generate samples from the distribution by using as proposal distribution the $LN(X_t, 1)$, taking one starting point at our choice (in our case 1).

The explanation on how the data is sampled and simulated (from the algorithm) is as follows: (0) for as many points sampled we want, we are going to (1) sample a point from the proposed distribution (in our case from a lognormal with mean equal to X_t and sd equal to 1) which will be our y, and our point x0 chosen at will. (2) Then, we generate a random uniform number equal to U(0,1). (3) if

$$(lognorm(y)/lognorm(X_t)) * (p(X_t|y)/p(y|X_t)) > U(0,1)$$

Then my new X_t value sampled is the previous X_t value else my new X_t is y. A plot representing 2000 random points sampled from a lognormal distribution using the Metropolis-Hastings can be found below in figure 1.

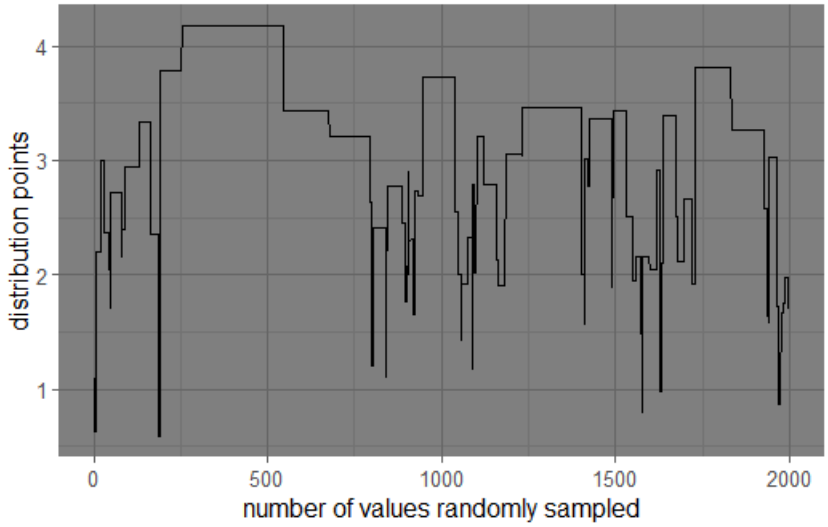

Figure 1: A plot representing 2000 random points sampled from a lognormal distribution using the Metropolis-Hastings

It can be seen that there is no convergence given the fact that the values get stack so that there is no different sampled values chosen and this shows that our lognormal model is not good to

simulate our target distribution. Also, no burning period is clear since there is no convergence on the graphic.

Now, we are asked to do the same but sampling from a new proposed distribution being a chi-squared being $Chi(\lfloor X_t + 1 \rfloor)$. A plot representing 2000 random points sampled from a chi-squared distribution using the Metropolis-Hastings can be found below in figure 2.

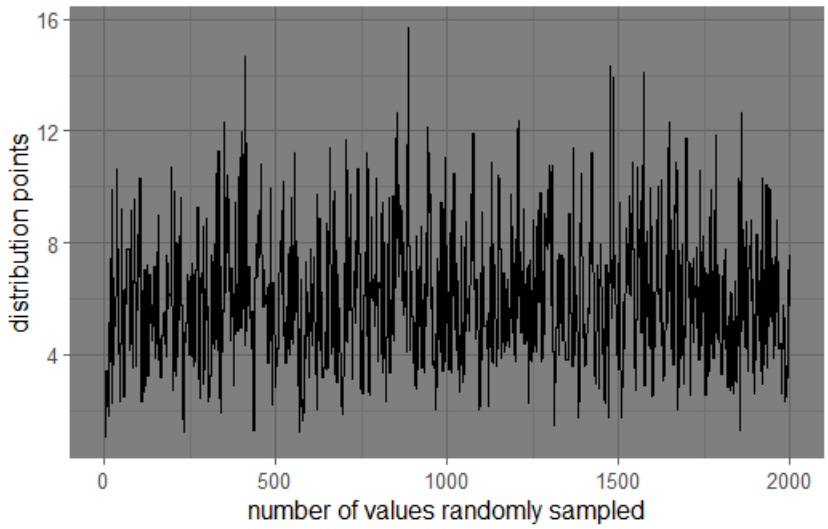

Figure 2: A plot representing 2000 random points sampled from a chi-squared distribution using the Metropolis-Hastings

It can be seen that there is convergence in this case given that our graphic looks totally as with a random walk. Nevertheless, a burning period is not clear to be found given the data. Using the Gelman-Rubin method, now we are trying to analyze the convergence of these chi-squared generated sequence. 10 MCMC sequences (from 1 to 10 by 1) have been generated using the "as.mcmc" function from the coda package, getting the following result:

```
attr(,"class")
[1] "mcmc.list"
> print(gelman.diag(mylist))
Potential scale reduction factors:

     Point est. Upper C.I.
[1,]          1       1.01
```

The Gelman-Rubin method states that approximated convergence is diagnosed when the upper limit is close to 1. Therefore, we confirm convergence on our Metropolis-Hastings algorithm for our second Multi target distribution from which we simulate points. Nevertheless, the mean of our data is evaluated, being the means for our sampled distributions the following ones:

```
 Elognormal
[1] 3.179977
> Echi
[1] 6.007571
```

The pdf of a gamma distribution with parameters (α, β) is:

$$f(x; \alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)}$$

The expectation of such a distribution is simply $E(X) = \alpha/\beta$.

Judging from the formulation of its pdf, the distribution we simulated here is clearly a gamma distribution with parameters $\alpha = 6, \beta = 1$. Therefore the expected value of this distribution and the actual value of the integral, is 6. So the estimated value of the integral from the step 2 sample is very close to this actual value, while the Step one sample estimate is much smaller.

Question 2

Exercise 2.1

In this exercise, we are asked to import some data to R and plot the dependence of Y on X. Here below in figure 3 it can be found the plot of the data:

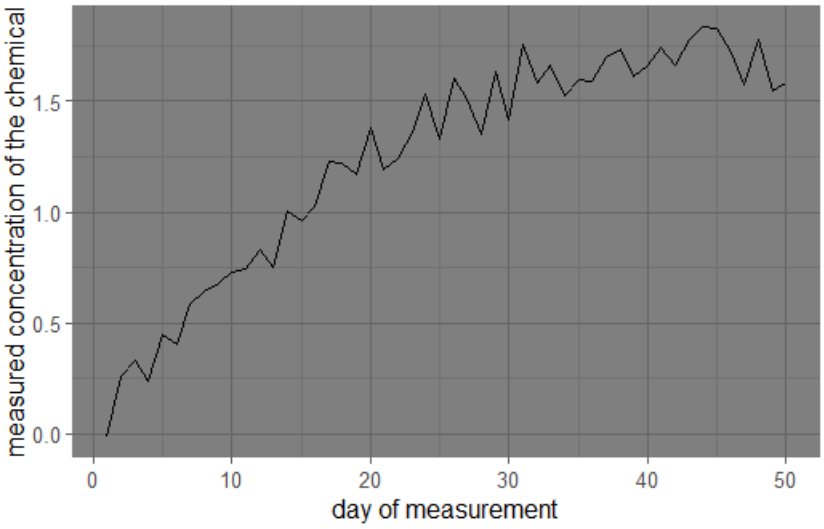

Figure 3: Plot the dependence of Y on X

The data seems to follow a logarithmic distribution.

Exercise 2.2

The likelihood is given by:

$$p(Y|\mu) = \prod_{i=1}^{n} p(y_i|\mu) \propto exp(-\frac{1}{2*0.2^2}\sum_{i=1}^{n}(y_i-\mu_i)^2)$$

and the prior is given by:

$$p(\mu) = \prod_{i=1}^{n} p(\mu_i) \propto exp(-\frac{1}{2*0.2^2}\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2)$$

Thus, the posterior is given by:

$$p(\mu|Y) \propto exp(-\frac{1}{2*0.2^2}(\sum_{i=1}^{n}(y_i-\mu_i)^2+\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2))$$

.

Exercise 2.3

From the previous exercise, we get the full conditional distribution as:

part 1:

$$p(\mu_1|\mu_{-1},Y) \propto exp(-\frac{1}{2*0.2^2}((\mu_1-y_1)^2+(\mu_1-\mu_2)^2)) \propto exp(-\frac{1}{\frac{2*0.2^2}{2}}(\mu_1-\frac{(y_1+\mu_2)}{2})^2)$$

part 2:

$$p(\mu_i|\mu_{-i}, Y) \propto exp(-\frac{1}{2*0.2^2}((\mu_i-y_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_i-\mu_{i+1})^2)) \propto exp(-\frac{1}{\frac{2*0.2^2}{3}}(\mu_i-\frac{(y_i+\mu_{i-1}+\mu_{i+1})}{3})^2)$$

for $i = 2, 3, \ldots, n-1$.

part 3:

$$p(\mu_n|\mu_{-n}, Y) \propto exp(-\frac{1}{2*0.2^2}((\mu_n-y_n)^2+(\mu_n-\mu_{n-1})^2)) \propto exp(-\frac{1}{\frac{2*0.2^2}{2}}(\mu_n-\frac{(y_n+\mu_{n-1})}{2})^2)$$

Exercise 2.4

We then used 3 previously derived expressions to implement a Gibbs sampler with starting point $\overrightarrow{\mu}^0$. We ran the Gibbs sampler to obtain 1000 $\overrightarrow{\mu}$ values. We took the mean over these 1000 for each component of $\overrightarrow{\mu}$ and plotted them against X, alongside Y vs X, as shown in Figure 4

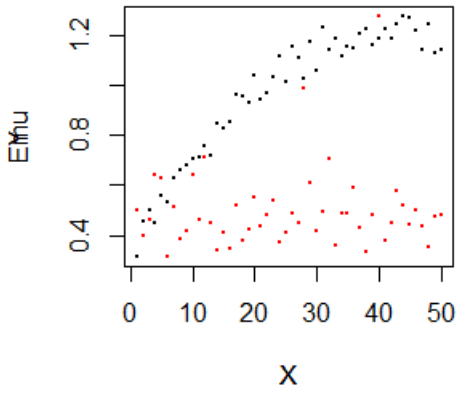

Figure 4: Plot showing Y (in black) and the expected Mu values (in red) vs X

Exercise 2.5

Figure 5 is a trace plot of μ_n over its 1000 iterations.

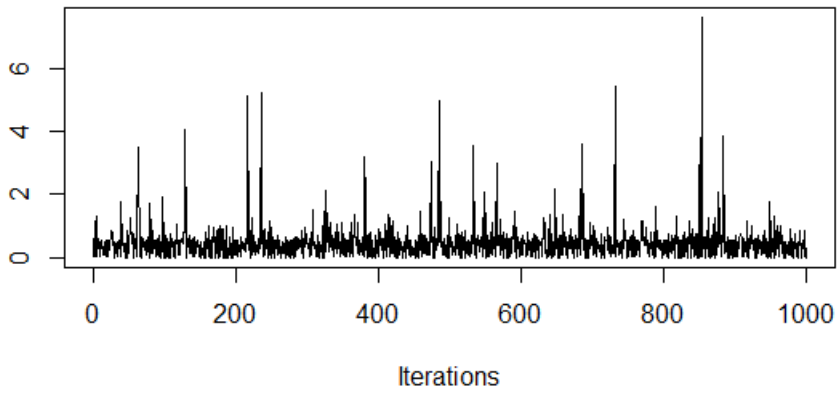

Figure 5: Trace Plot of Mu for i=n

Comment: The results in 2.4 and 2.5 seem a little strange, as Mu is not close to Y at all, so we think that our Gibbs sampler is incorrect.

Contributions

All results and comments presented have been developed and discussed together by the members of the group.

Appendix

Assignment 1

```
####LAB 4
## Activity 1

set.seed(12345)
###1.1
posterior_PDFlognorm<- function(value,param){
  return(dlnorm(x=value, meanlog = param, sdlog = 1))
}

##target distribution
targetDist<- function(x){
  return(x**5*exp(-x))
}

MetropolisHastings<- function(times, init, posterior,ydist="rlnorm"){
  result<- integer(0)
  result[1]<-init
  counter<-1
  init<- init

  while(counter<times){

    x<-init
    if(ydist == "chi"){
      y0<- rchisq(1,floor(x+1))
      }
    else{
      y0<- rlnorm(1, x,1)
    }
    #Sample a candidate value X* from a proposal distribution g(.|x)

    myrunif<- runif(1,0,1)

    alpha<-(posterior(x, y0)/posterior(y0,x))*(targetDist(y0)/targetDist(x))
    if(myrunif<alpha){
      init<- y0

    }else{
      init<-init
    }
    counter<-counter+1
    result[counter]<- init

  }
  return(result)
}

MCMH<-MetropolisHastings(2000,1, posterior_PDFlognorm)

library(ggplot2)
myplot<- function(data){
  mydata<- as.data.frame(data)
  mplot<- ggplot2::ggplot(data = mydata)+geom_line(aes( x= 1:nrow(mydata), y = data))+
          ylab("distribution points")+xlab("number of values randomly sampled")+
          theme_dark()
  return(mplot)
}
myplot(MCMH)


###1.2
set.seed(12345)
chisquarePDF<- function(value,param){
  return(dchisq(value,floor(param+1)))
}

MCMH_chi<-MetropolisHastings(2000, 1, chisquarePDF, "chi")
myplot(MCMH_chi)

###1.3 EXplanation

##1.4
#install.packages("coda")
library(coda)
mylist<-mcmc.list()

times<-2000
n<- 10

for( i in 1:n){
  mylist[[i]]<-as.mcmc(MetropolisHastings(times, i, chisquarePDF, "chi"))
```

```
}
mylist
print(gelman.diag(mylist))


##1.5
Elognormal<- mean(MCMH)
Echi<- mean(MCMH_chi)

##1.6
alpha <- 6
beta<- 1
Ebeta <- alpha/Beta
```

Assignment 2

```
##Activity 2
library(ggplot2)
load("C:/Users/M/Desktop/Statistics and Data Mining Master/Semester 2/Computationa Statistics/
     Lecture 4/chemical.RData")

##2.1
mydata<- data.frame(X=X, Y = Y)

ggplot2::ggplot(mydata)+ geom_line(aes(x = X, y = Y))+
          xlab("day of measurement")+ ylab("measured concentration of the chemical")+
          theme_dark()

##2.4
gibbs <- function(tmax, mu0, var=0.2) {
  d <- length(mu0)
  mu <- matrix(0, nrow=tmax, ncol = d)
  mu[1, ] <- mu0
  Y <- matrix(0, nrow=tmax, ncol = d)
  Y[1, ] <- sapply(X=mu0, FUN=function(mu){
    y <- rnorm(1, mean=mu, sd=0.2)
    return(y)
  })
  t = 1
  while (t < tmax) {
    #update mu_t
    #i=1
    mu[t+1, 1] <- exp(-1/0.04*(mu[t, 1]-(Y[t, 1]+mu[t, 2])/2)^2)/(dnorm(Y[t, 1], mean=mu[t, 1],
          sd=0.2))
    #i =2,3,...,d-1
    dmin1 <- d-1
    for (i in 2:dmin1) {
      mu[t+1, i] <- exp(-1/(0.08/3)*(mu[t, i]-(Y[t, i]+mu[t+1, i-1]+mu[t, i+1])/3)^2)/(dnorm(Y[
           t, i], mean=mu[t, i], sd=0.2))
    }
    #i=d
    mu[t, d] <- exp(-1/0.04*(mu[t, d]-(Y[t, d]+mu[t+1, d-1])/2)^2)/(dnorm(Y[t, d], mean=mu[t, d
         ], sd=0.2))

    #update Y_t
    Y[t+1, ] <- sapply(X=mu[t+1, ], FUN=function(mu){
    y <- rnorm(1, mean=mu, sd=0.2)
    return(y)
  })

    t <- t+1
  }
  return(mu)
}

mu0 <- rep(0, 50)
tmax <- 1000
mu <- gibbs(tmax=tmax, mu0=mu0)
Emu <- colMeans(mu)
plot(X, Emu,pch=19, cex=0.3, col="red")
par(new=TRUE)
plot(X, Y,pch=19, cex=0.3, col="black", axes = FALSE)

##2.5
mun <- as.mcmc(mu[, 50])
library(coda)
traceplot(mun, type="p", pch=19, cex=0.3)
```