Computational Statistics Final Solution

Md. Shahbaz Alam | ID: 011015340

2023-12-15

Solution to Q1 (a):

To apply Metropolis-Hastings algorithm, we will use a normal distribution as a proposal density, as the normal distribution is symmetric around its mean. Then we will to generate 100,000 samples of X, and visualize them.

Procedure:

$Density function$:

$f(x) = c * e^{cos(x)};\ where\ -10 \leq x \leq +10$

$N(\mu, c' = 1)$: Proposal Density

$Algorithm$:

Step 0: Initialize first sample $\theta^{(j=0)} = \theta_0$.

$Loop$:

Step 1: Draw a sample from normal proposal $\theta^* \sim N(\theta^{(j-1)}, c' = 1)$

Step 2: Calculate the Ratio $R = \dfrac{f(\theta^*)}{f(\theta^{(j-1)})}$

Step 3: Generate a uniform random number $u \sim Unif(0, 1)$

Step 4: If $R > u : set\theta^{(j)} = \theta^*$

$else \quad : set\theta^{(j)} = \theta^{(j-1)}$

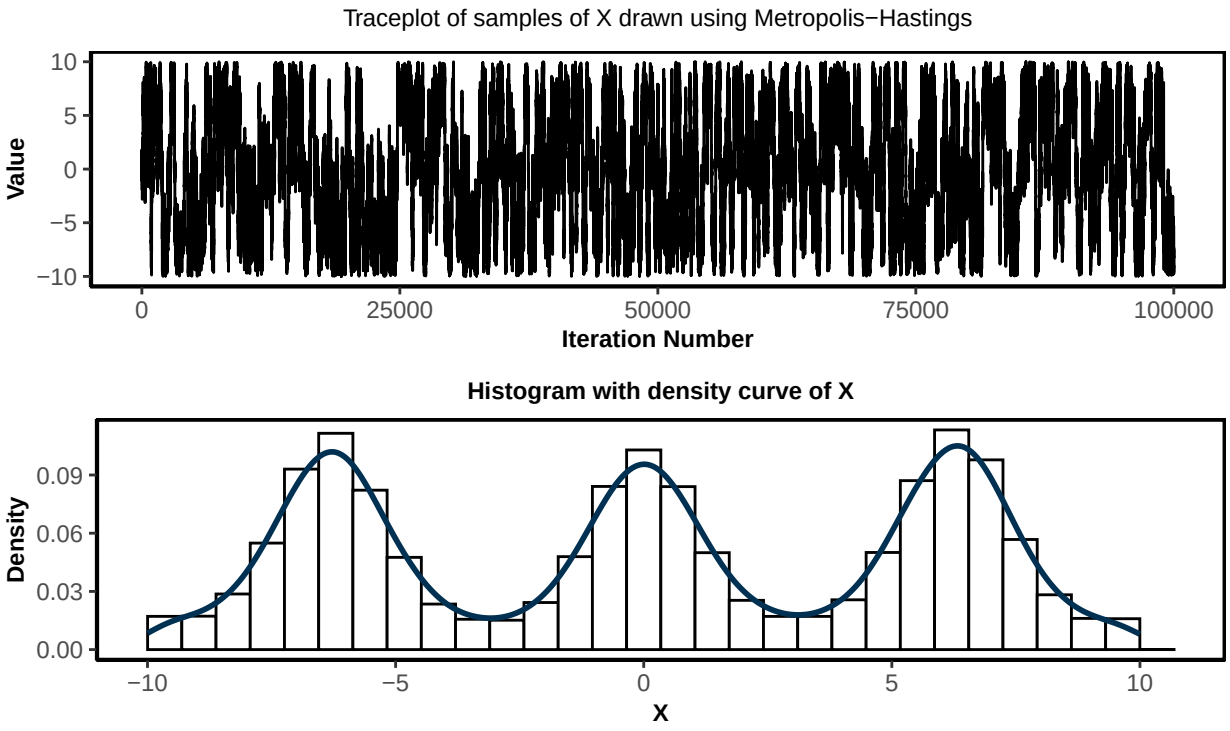
Here, for the initial value θ_0, any random value within the range (-10, +10) can be chosen. In this case, we select 0. The standard deviation of the proposal density, denoted as c', is set to 1.

R code:

Here is the R code for generating 100,000 sample from $f(x) = c * e^{cos(x)}$; $where \ -10 \leq x \leq +10$.

```r
# Define the function
fn_x <- function(x) {
  if_else(-10 < x & x < 10, exp(cos(x)), 0)
}

# Initialize variables
set.seed(729)
vec_x <- numeric(10e4)

# loop for MH
for (i in 2:10e4) {
  candidate <- rnorm(1, mean = vec_x[i - 1], sd = 1)
  ratio <- fn_x(candidate) / fn_x(vec_x[i - 1])
  if (ratio > runif(1)) {
    vec_x[i] <- candidate
  } else {
    vec_x[i] <- vec_x[i - 1]
    }
  }
```

Visualization:



The histogram of the generated plot aligns closely with the density plot, indicating the correctness of the data generation method. Also, the trace plot shows convergence

Solution to Q1 (b):

Density function:

$$f(x) = c \cdot e^{\cos(x)};\ \text{where}\ -10 \leq x \leq 10$$
$$V(X) = E(X^2) - E(X) : \text{Quantity of interest}$$
$$P \sim \text{Unif}(\min = -10, \max = 10) : \text{Proposal density}$$

Algorithm:

Loop begin: $for number of samples$
Step 1: Draw a sample $\theta^* \sim \text{Unif}(\min = -10, \max = +10)$
Step 2: Calculate weight $w_i = \dfrac{f(\theta^*)}{P(\theta^*)}$
Step 3: Store them in array: $W = [w_1, w_2, \ldots, w_i]$ and $\Theta = [\theta_1^*, \theta_2^*, \ldots, \theta_i^*]$
End of loop

Calculate:

$$\widehat{E(X^2)} = \frac{\sum_{i=1}^{N} (\theta^*)^2 \cdot w_i}{\sum_{i=1}^{N} w_i}$$
$$\widehat{E(X)} = \frac{\sum_{i=1}^{N} \theta^* \cdot w_i}{\sum_{i=1}^{N} w_i}$$
$$\widehat{V(X)} = \widehat{E(X^2)} - \widehat{E(X)}^2$$

R code:

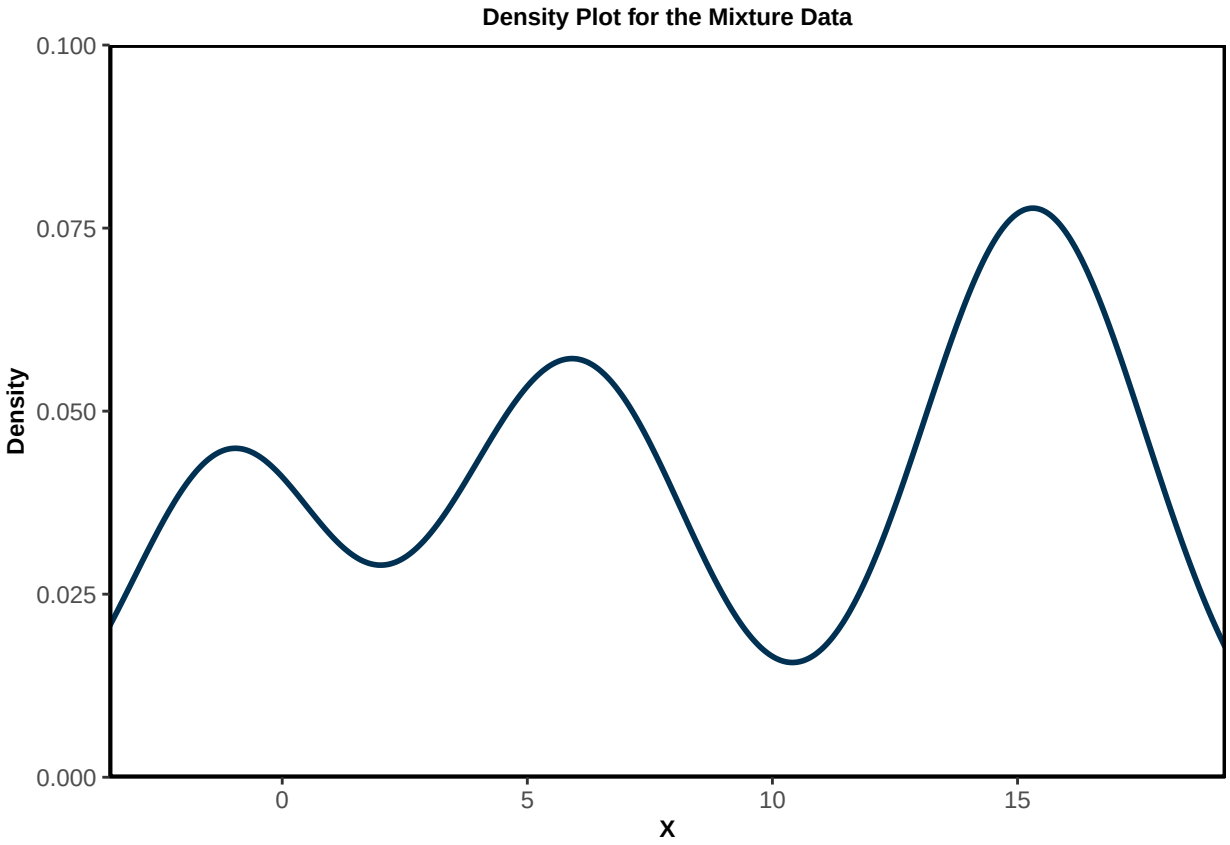
The following R code estimates $E(X^2)$ using self-normalized importance sampling from a distribution where $X \sim f(x) = c \cdot e^{\sin(x)}$. For estimation we will use 100,000 samples.

```
fn_x <- function(x) {

  if_else(-10 < x & x < 10, exp(cos(x)), 0)
}

# Initialize variables
cand_w <- numeric(10e4)
cand_x <- numeric(10e4)
cand_xsq <- numeric(10e4)
d_unif <- dunif(1, -10, 10)
set.seed(729)

# Perform the loop
for (i in 1:10e4) {

  cand <- runif(1, -10, 10)

  cand_w[i] <- fn_x(cand) / d_unif
  cand_x[i] <- cand * cand_w[i]
  cand_xsq[i] <- cand^2 * cand_w[i]
  }
```

from (b)	from (a)
29.338	29.752

The calculated variance obtained using those 100,000 generated samples earlier closely aligns with the variance estimated using these 100,000 self-normalized importance samples.

Solution to Q2:



From the plot, it's evident that there are three distinct peaks, suggesting the presence of three different groups. The mean of these groups appear to be approximately 0, 6, and 15. Therefore, the initial estimates for the distribution are $\mu^T = [0\ 6\ 15]$ with $\sigma^T = [2\ 2\ 2]$.

R code (for initial step):

```r
n = length(data_x)
mu.init = c(0, 5, 15)
sig.init = c(2, 2, 2)
pi.init = c(0.2, 0.3, 0.5)

# EM initialization
step = 0
eps = 10^(-6)
L = 1

dens.mat = matrix(c(dnorm(data_x, mu.init[1], sig.init[1]),
                    dnorm(data_x, mu.init[2], sig.init[2]),
                    dnorm(data_x, mu.init[3], sig.init[3])),
                  nrow = length(data_x), ncol = 3)

post.prob.0 = pi.init[1]*dens.mat[, 1] / c(dens.mat %*% pi.init)
post.prob.1 = pi.init[2]*dens.mat[, 2] / c(dens.mat %*% pi.init)
post.prob.2 = pi.init[3]*dens.mat[, 3] / c(dens.mat %*% pi.init)


mu.new = c(sum(post.prob.0*data_x)/sum(post.prob.0),
           sum(post.prob.1*data_x)/sum(post.prob.1),
           sum(post.prob.2*data_x)/sum(post.prob.2)
           )

sig.new = sqrt(c(sum(post.prob.0*(data_x-mu.new[1])^2)/sum(post.prob.0),
                 sum(post.prob.1*(data_x-mu.new[2])^2)/sum(post.prob.1),
                 sum(post.prob.2*(data_x-mu.new[3])^2)/sum(post.prob.2)
                 ))

pi.new = c(mean(post.prob.0), mean(post.prob.1), mean(post.prob.2))
```

R code (for EM algorithm):

```
while(L > eps) {

  mu.old = mu.new
  sig.old = sig.new
  pi.old = pi.new

  dens.mat = matrix(c(dnorm(data_x, mu.init[1], sig.init[1]),
                      dnorm(data_x, mu.init[2], sig.init[2]),
                      dnorm(data_x, mu.init[3], sig.init[3])),
                    nrow = length(data_x), ncol = 3)

  post.prob.0 = pi.init[1]*dens.mat[, 1] / c(dens.mat %*% pi.init)
  post.prob.1 = pi.init[2]*dens.mat[, 2] / c(dens.mat %*% pi.init)
  post.prob.2 = pi.init[3]*dens.mat[, 3] / c(dens.mat %*% pi.init)

  mu.new = c(sum(post.prob.0*data_x)/sum(post.prob.0),
             sum(post.prob.1*data_x)/sum(post.prob.1),
             sum(post.prob.2*data_x)/sum(post.prob.2)
             )

  sig.new = sqrt(c(sum(post.prob.0 * (data_x-mu.new[1])^2) / sum(post.prob.0),
                   sum(post.prob.1 * (data_x-mu.new[2])^2) / sum(post.prob.1),
                   sum(post.prob.2 * (data_x-mu.new[3])^2) / sum(post.prob.2)
                   ))

  pi.new = c(mean(post.prob.0), mean(post.prob.1), mean(post.prob.2))
  L = max(abs(mu.old - mu.new))
  step = step + 1

  }
```

Table for the estimated parameters:

	Dist 1	Dist 2	Dist 3
μ	-0.8023	5.772	15.29
σ	1.455	1.64	1.462
$P(Y) \in Dist\ i$	0.2282	0.3262	0.4455

The table presents the estimated parameters. Assuming three categories, the proportions for each category are 22.8%, 32.9%, and 44.5%. The group means are -0.8023, 5.772, and 15.29, with corresponding standard deviations of 1.4549, 1.64, and 1.462.

Solution to Q3 (a):

R code:

```r
# Initialize variables
med_x <- numeric(1e4)
mom_x <- numeric(1e4)

# Perform the loop
for (i in 1:1e4) {
  x <- sample(x = data_x, size = length(data_x), replace = TRUE)
  med_x[i] <- median(x)
  mom_x[i] <- mean(x^4)
  }

# Calculate means
mean_med_x <- mean(med_x)
mean_mom_x <- mean(mom_x)

# Output the results
mean_med_x
mean_mom_x
```

Median	Forth_Raw_Moment
7.326	26191.645

Solution to Q3 (b):

R code:

```
# Initialize variables
med_x <- mom_x <- 0

n <- length(data_x)
for( i in 1:n){
  med_x[i] <- median(data_x[-i])
  mom_x[i] <- mean(data_x[-i]^4)
}
n*median(data_x) - (n-1)*mean(med_x)
n*mean(data_x^4) - (n-1)*mean(mom_x)
```

Median	Forth Raw Moment
7.262	26194.031

Both Median and the expected value of the Fourth Raw Moment $E(X^4)$ estimates using Bootstrap and leave-one-out Jackknife methods shows close similarity.

Solution to Q4:

$$f(\theta) = a.exp(-\theta^5 + 5\theta - \sqrt{\theta}) + c, \quad 0 < \theta < 2$$

where a and c are positive constants. Use (a) Bisection and (b) Newton's method to find the MLE of θ.

$$\begin{aligned}
L(x) &= a \cdot e^{-x^5+5x-\sqrt{x}} + c; \quad \text{where } a, c > 0 \\
L(x) &\propto e^{-x^5+5x-\sqrt{x}} \\
l(x) &= \log L(x) \\
l(x) &\propto -x^5 + 5x - \sqrt{x}
\end{aligned}$$

$$\begin{aligned}
l'(x) &= -5x^4 + 5 - 0.5x^{-0.5} \\
l''(x) &= -20x^3 + 0.25x^{-1.5}
\end{aligned}$$

R Code (1st and 2nd derivative):

```
l_theta <- function(x, first_der = TRUE) {
  if_else(first_der, -5 * x^4 + 5 - .5 * x^(-1/2), -20 * x^3 + .25 * x^(-1.5))
}
```

part (a):

R code:

```
left_x <- 0.5
right_x <- 1

while (abs(left_x - right_x) > 1e-5) {
  mid_x <- (left_x + right_x) / 2
  if (l_theta(mid_x) * l_theta(right_x) > 0) {
    right_x <- mid_x
    } else {
    left_x <- mid_x
    }
  }

result <- l_theta(mid_x, first_der = FALSE)
result
mid_x
```

Observing the negative value of the second derivative indicates that the estimate obtained through the bisection method represents the point at which the likelihood is maximized.

MLE with Bisection	Second Derivative
0.9736	-18.1995

Part (b):

R code:

```
init_x <- 1

while (TRUE) {
  prev_x <- init_x
  init_x <- init_x - l_theta(init_x) / l_theta(init_x, first_der = FALSE)

  if (abs(prev_x - init_x) < 1e-5) {
    break
    }
  }

init_x
```

MLE with NR	Second Derivative
0.9736	-18.1995

The second derivative of estimated value is negative which suggests that the estimate obtained through Newton's method represents the point at which the likelihood is maximized.

Thus, in both methods, the obtained maximum likelihood estimates (MLEs) are identical.