Computational Statistics Coursework 1
MSc in Statistics, Imperial College London

CID: 01855742

Contents

I, CID: 01855742, certify that this assessed coursework is my own work, unless otherwise acknowledged, and includes no plagiarism. I have not discussed my coursework with anyone else except when seeking clarification with the module lecturer via email or on MS Teams. I have not shared any code underlying my coursework with anyone else prior to submission.

1 Question 1

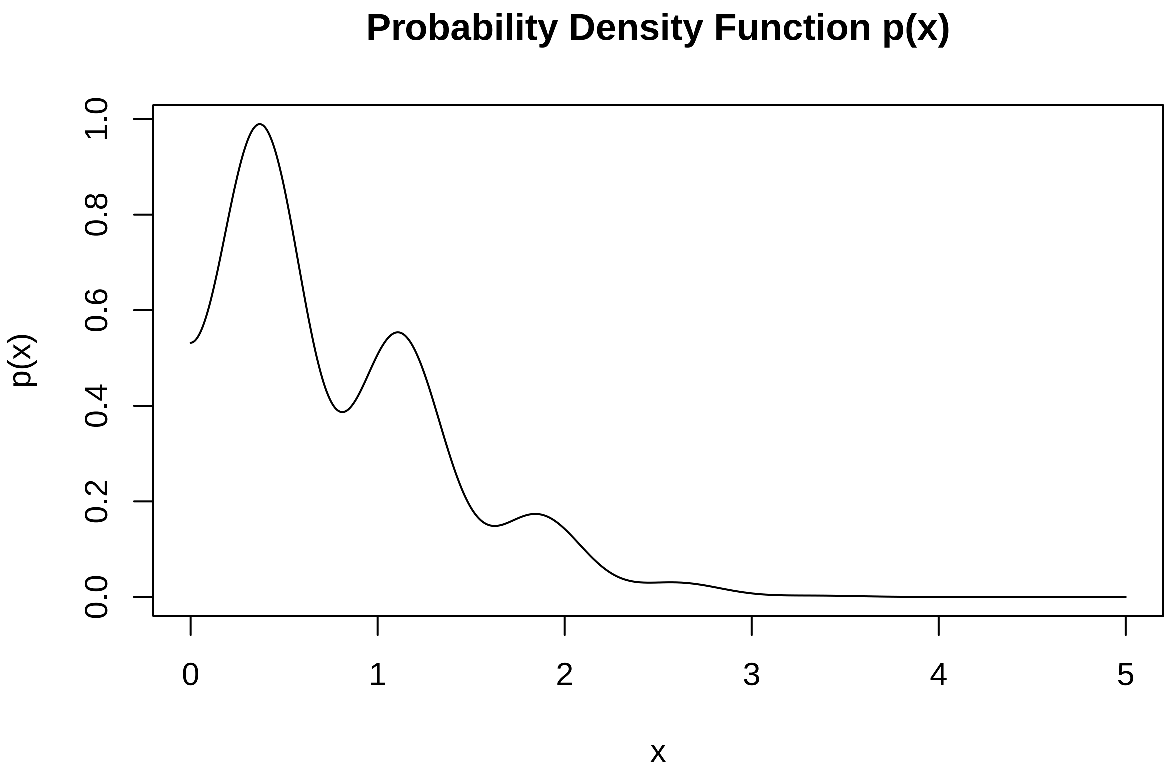
Part a.

Since $p(x)$ is a density function, we know that it should integrate to 1. Therefore, by integrating unnormalised expression (without k) using the integrate() function in R, we can find the value of k so that the $p(x)$ does in fact integrate to 1.

```
#> [1] 0.5319234
```

Therefore we see our k is (approximately) 0.5319234.

Plotting $p(x)$:



Part b.

$$\frac{p(x)}{g(x)} = \frac{ke^{-\frac{x^2}{2}}(sin(4x)^2+1)}{e^{-x}} \leq \frac{2ke^{-\frac{x^2}{2}}}{e^{-x}} \leq 2ke^{-\frac{x^2}{2}+x} \leq 2ke^{-\frac{x^2}{2}+x-\frac{1}{2}+\frac{1}{2}} \leq 2ke^{\frac{1}{2}}e^{-\frac{1}{2}(x^2-2x+1)} \leq 2ke^{\frac{1}{2}}e^{-\frac{1}{2}(x-1)^2} \leq 2ke^{\frac{1}{2}}$$

Note we use the results $(sin(4x)^2+1) \leq 2$ and $e^{-\frac{1}{2}(x-1)^2} \leq 1$ for $0 \leq x \leq 5$

Therefore we have shown there exists some $M = 2ke^{\frac{1}{2}}$ such that $p(x) \leq Mg(x)$

This may however not be optimal, so we can try to use the optim() function in R to find the optimal M, which would be the largest value that $\frac{p(x)}{g(x)}$ takes in the interval $0 \leq x \leq 5$

```
#> [1] 1.753987
#> [1] 1.728001
```

Our analytical value for M is 1.753987, and using the optim function in R, we get the numerical estimate of $M_{optim} = 1.728001$

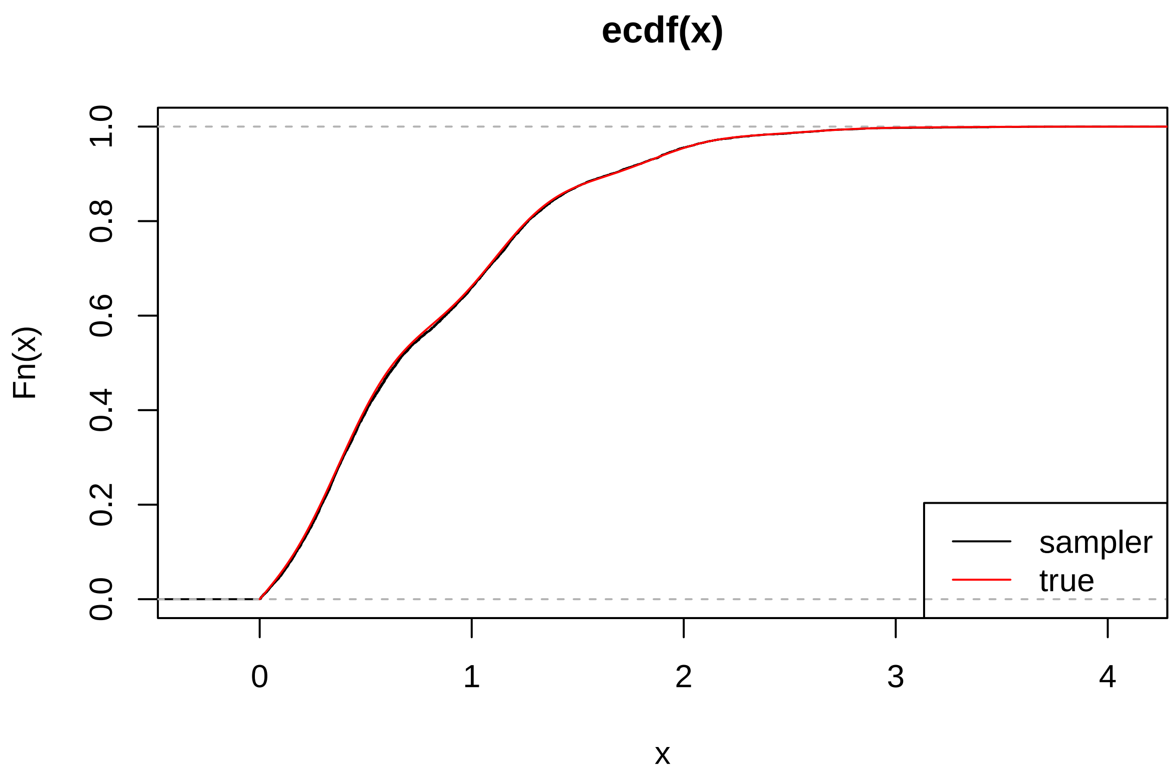
Part c.

The proportion of samples accepted is $\frac{1}{M}$.

```
#> [1] 0.5787034
```

Using $M_{optim} = 1.728001$, we get that the proportion of samples accepted is 0.5787034

Plotting the Empirical Cdf below and comparing it to the true cdf:



Part d.

Computing: $I = P[X > 1.5]$ using Monte Carlo:

```
#> [1] 0.1263
```

We estimate that $\hat{I}_{MC} = 0.205$.

Computing a 90% confidence interval using the formula given in lectures:

$$\left[\hat{I}_{MC} + \frac{S}{\sqrt{n}} Z_{\alpha/2}, \hat{I}_{MC} + \frac{S}{\sqrt{n}} Z_{1-\alpha/2}\right]$$

with $\alpha = 0.1, n = 10000$

```
#> [1] 0.1208357 0.1317643
```

Our 90% confidence interval is [0.1208357,0.1317643]

Part e.

To determine an optimal α, we want α to be such that P(X>1.5) is large, where X is Gamma(α, 1) distributed. However, we also don't want the density close to 1.5 to be very small as this could make our standard errors large. Trying the values $\alpha = 1, 3, 5, 7, 10$ in that order give:

```
#> [1] 0.2231302
#> [1] 0.8088468
#> [1] 0.9814241
#> [1] 0.999074
#> [1] 0.9999959
```

We decide to choose the value $\alpha = 3$, as $P[X > 1.5] = 0.8088468$, which is large and larger values of α may have small density values around $X = 1.5$

```
#> [1] 0.1273861
```

Our Importance Sampling estimate is: $\hat{I}_{IS} = 0.1273861$

```
#> [1] 0.1238852 0.1308870
```

The 90% confidence interval for $\hat{I}_{IS}$ is [0.1238852 0.1308870].

```
#> [1] 0.0109286
#> [1] 0.0070018
```

Note that the length of the confidence interval for $\hat{I}_{IS}$ is 0.0070018, which is lower than the length of $\hat{I}_{MC}$ which suggests the Importance Sampling estimate is more efficient.

2 Question 2

Part a.

The log likelihood for the complete data $(\mathcal{Y}, \mathcal{X})$is:

$$\mathcal{L}(\mu \,|\, \mathcal{Y}, \mathcal{X}) = -\sum_{i=1}^{n} \left(\frac{1}{2} \log(2\pi\sigma^2) + \frac{(x_i - \mu)^2}{2\sigma^2} \right)$$

Conditioning on observed data $\mathcal{Y}$ and μ_n, we get the minorant function:

$$Q(\mu \,|\, \mu_n) = -\sum_{i=1}^{n} \left(\frac{1}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \mathbb{E}\left[(X_i - \mu)^2 \middle|\, \mathcal{Y}, \mu_n \right] \right)$$

Note that the property of conditional expectation to get that:

$$\mathbb{E}\left[(X_i - \mu)^2 \middle|\, \mathcal{Y}, \mu_n \right] = \mathbb{E}\left[(X_i - \mu)^2 \middle|\, X_i \geq 0, \mathcal{Y}, \mu_n \right] \cdot P(X_i \geq 0 | \mathcal{Y}, \mu_n) + \mathbb{E}\left[(X_i - \mu)^2 \middle|\, X_i < 0, \mathcal{Y}, \mu_n \right] \cdot P(X_i < 0 | \mathcal{Y}, \mu_n)$$

and we know that

$$\mathbb{E}\left[(X_i - \mu)^2 \middle|\, X_i \geq 0, \mathcal{Y}, \mu_n \right] = (Y_i - \mu)^2$$

and

$$\mathbb{E}\left[(X_i - \mu)^2 \middle|\, X_i < 0, \mathcal{Y}, \mu_n \right] = (-Y_i - \mu)^2 = (Y_i + \mu)^2$$

Using Bayes Rule, we have that:

$$P(X_i \geq 0 | \mathcal{Y}, \mu_n) = \frac{\phi(Y_i; \mu_n, \sigma^2)}{\phi(Y_i; \mu_n, \sigma^2) + \phi(-Y_i; \mu_n, \sigma^2)} = w_i(\mu_n)$$

and we can easily calculate that

$$P(X_i < 0|\mathcal{Y}, \mu_n) = 1 - P(X_i \geq 0|\mathcal{Y}, \mu_n) = 1 - w_i(\mu_n)$$

Therefore, using these results, we have that

$$\mathbb{E}\left[(X_i - \mu)^2 \middle| \mathcal{Y}, \mu_n\right] = (Y_i - \mu)^2 \cdot w_i(\mu_n) + (Y_i + \mu)^2 \cdot (1 - w_i(\mu_n))$$

and so

$$Q(\mu \,|\, \mu_n) = -\sum_{i=1}^{N} w_i(\mu_n)\frac{(Y_i - \mu)^2}{2\sigma^2} + (1 - w_i(\mu_n))\frac{(Y_i + \mu)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$$

as required.

Part b.

Differentiating $Q(\mu \,|\, \mu_n)$ with respect to μ we get:

$$\frac{\partial Q(\mu \,|\, \mu_n)}{\partial \mu} = -\sum_{i=1}^{N} -w_i(\mu_n)\frac{(Y_i - \mu)}{\sigma^2} + (1 - w_i(\mu_n))\frac{(Y_i + \mu)}{\sigma^2}$$

Setting the derivative to 0 and multiplying both sides by $-\sigma^2$, we get:

$$\sum_{i=1}^{N} -w_i(\mu_n)(Y_i - \mu_{n+1}) + (1 - w_i(\mu_n))(Y_i + \mu_{n+1}) = 0$$

Multiplying out the brackets and simplifying, we get:

$$\sum_{i=1}^{N} \mu_{n+1} - Y_i \cdot (2w_i(\mu_n) - 1) = 0$$

and since : $\sum_{i=1}^{N} \mu_{n+1} = N\mu_{n+1}$, we have:

$$N\mu_{n+1} = \sum_{i=1}^{N} Y_i \cdot (2w_i(\mu_n) - 1)$$

and therefore our update rule for μ_{n+1} is:

$$\mu_{n+1} = \frac{1}{N}\sum_{i=1}^{N} Y_i \cdot (2w_i(\mu_n) - 1)$$

Part c. Implementing EM Algorithm for the true distribution $N(5, 10^2)$

```
#> [1] 5.718861
```

We can see that our estimate is $\hat{\mu} = 5.718861$, which is an overestimate of our true value $\mu = 5$

Part d. Trying different values $\mu_0 = -6, -3, 0, 3, 6$ in that order, we get:

```
#> [1] -5.718861
#> [1] -5.718861
#> [1] 0
#> [1] 5.718861
#> [1] 5.718861
```

Note that in the case $\mu_0 = 0$, we get the estimate $\mu = 0$.

This is because $w_i(\mu_0) = 1/2$ and so $2w_i(\mu_0) - 1 = 0$ meaning

$$\mu_1 = \frac{1}{N} \sum_{i=1}^{N} Y_i \cdot (2w_i(\mu_0) - 1) = 0$$

and the algorithm will stop.

Note that when $\mu_0 > 0$, we seem to converge close to the true μ.

However, when $\mu_0 < 0$, we seem to converge close to $-\mu$.

Therefore, it appears that our EM algorithm can estimate the magnitude of the true μ, but not the sign.

3 Code appendix

```
knitr::opts_chunk$set(
  collapse = TRUE,
  comment = "#>"
)
include_solutions <- TRUE
require(rmarkdown)
require(knitr)
require(kableExtra)
# Library Imports

### Question 1a

set.seed(100)

p_unnormalised <- function(x) ifelse(0<= x & x <= 5,
                         exp(-(x^2/2))*((sin(4*x))^2 + 1), 0)


k <- 1/integrate(p_unnormalised,0,5)$value

k



p <- function(x) p_unnormalised(x) * k

### Plot of p(x)

x_values <- seq(0, 5, length.out = 1000)

plot(x_values, p(x_values), type = "l", xlab = "x", ylab = "p(x)", main = "Probability Density Function



### Question 1b
```

```r
#Analytical M:
m_anal <- 2*k*exp(1/2)

m_anal
#OPTMISE M

g <- function(x)dexp(x, rate = 1)

p_g <- function(x) p(x)/g(x)


M_optim <- optim(1,p_g,lower=0,upper=5,method="Brent",control=list(fnscale=-1),hessian=TRUE)$value


M_optim

### Question 1c

M <- M_optim

1/M


#Rejection Sampling, modifying function from Problem Sheet 4
set.seed(100)

rf <- function(){
  while(1){
    x <- rexp(1, rate=1)
    if (runif(1) < p(x)/(M*g(x))){
      return(x)
    }
  }
}

x <- replicate(10000,rf())

plot(ecdf(x))


P <- function(x) integrate(p,0,x)$value
t <- seq(0,5,length.out=10000)
lines(t,Vectorize(P)(t),col="red")
legend("bottomright",col=c("black","red"), c("sampler","true"),lty=1)





### Question 1d

#Monte Carlo estimate
```

```r
MC <- mean((x>1.5))

MC


### Confidence Interval

CI <- MC + (sd(x>1.5)/sqrt(10000))*qnorm(c(0.05,0.95))

CI


### Question 1e

1 - pgamma(1.5, 1, 1)
1 - pgamma(1.5, 3, 1)
1 - pgamma(1.5, 5, 1)
1 - pgamma(1.5, 7, 1)
1 - pgamma(1.5, 10, 1)

set.seed(120)

alpha <- 3
N <- 10000

y <- rgamma(N, alpha, 1)

IS <- mean((y>1.5)*p(y)/ dgamma(y, alpha, 1))

IS


IS_CI <- IS + sd((y>1.5)*p(y)/ dgamma(y, alpha, 1))/ sqrt(10000) * qnorm(c(0.05,0.95))

IS_CI

#Length of MC confidence interval
0.1317643 - 0.1208357

#Length of IS confidence interval
0.1308870 - 0.1238852


### Question 2c

#Generate data:
set.seed(100)

N <- 1000
mu = 5
sigma = 10

X <- rnorm(N, mean=mu, sd = sigma)
```

```r
Y <- abs(X)

#EM Algorithm

mu_curr <- 1 # Starting point mu = 1
mu_path <- mu_curr
maxsteps <- 1000
step <- 1

wi <- function(mu_n, y) dnorm(y, mu_n, sigma)/(dnorm(y, mu_n, sigma) + dnorm(-y, mu_n, sigma))

while(step<=maxsteps) {
  # Update algorithm
  mu_next <- (1/N) * sum(Y * (2*sapply(Y, wi, mu_curr) - 1))

  mu_path <- c(mu_path,mu_next)
  if (abs(mu_next-mu_curr)<1e-10) break;
  mu_curr <- mu_next
  step <- step+1

  ## should stop when a certain maximum number of steps is reached
}

#mu_path

mu_path[length(mu_path)]



### Question 2d

EM <- function(mu_curr){
  while(step<=maxsteps) {
  # Update algorithm
  mu_next <- (1/N) * sum(Y * (2*sapply(Y, wi, mu_curr) - 1))

  mu_path <- c(mu_path,mu_next)
  if (abs(mu_next-mu_curr)<1e-10) break;
  mu_curr <- mu_next
  step <- step+1

  ## should stop when a certain maximum number of steps is reached
  }

  return(mu_path[length(mu_path)])

}

EM(-6)
EM(-3)
EM(0)
EM(3)
EM(6)
```

```
### Question


### Question
```

4 References