HW3

Yan Liu

9/15/2019

1.

$$X = \begin{cases} \mathcal{N}(\mu_1, \sigma_1^2) & \text{with probability } p_1 \\ \mathcal{N}(\mu_2, \sigma_2^2) & \text{with probability } 1 - p_1 \end{cases}$$

a)

The data file specifies gender, but pretend you don't have this information. Write down the log-likelihood function $\ell(\theta)$ and $\nabla\ell(\theta)$ given the height samples, i.e. in terms of $\hat{X}_i$. Write R (or Python) functions that calculate $\ell(\theta)$ and $\nabla\ell(\theta)$

Solution

$$\begin{aligned} \theta &= (\mu_1, \mu_2, \sigma_1, \sigma_2, p1) \\ P(\hat{X}|\theta) &= P(\hat{X}_1|\theta) \cdot P(\hat{X}_2|\theta) \cdot ... \cdot P(\hat{X}_n|\theta) \\ &= \prod_{i=1}^{n} P(\hat{X}_i|\theta) \\ \implies \ell(\theta) &= log(P(\hat{X}|\theta)) \\ &= \sum_{i=1}^{n} log(P(\hat{X}_i|\theta)) \\ &= \sum_{i=1}^{n} log(p_1 \cdot \frac{1}{\sqrt{2\pi}\sigma_1^2} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi}\sigma_2^2} e^{-(x_i-\mu_2)^2/2\sigma_2^2}) \end{aligned}$$

$$\theta_{MLE} = argmax_\theta \ \ell(\theta)$$

$$\nabla\ell(\theta) = \begin{pmatrix} \frac{\partial\ell(\theta)}{\partial\mu_1} \\ \frac{\partial\ell(\theta)}{\partial\mu_2} \\ \frac{\partial\ell(\theta)}{\partial\sigma_1} \\ \frac{\partial\ell(\theta)}{\partial\sigma_2} \\ \frac{\partial\ell(\theta)}{\partial p_1} \end{pmatrix}$$

where

$$\frac{\partial \ell(\theta)}{\partial \mu_1} = \sum_{i=1}^{n} \frac{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} \cdot \frac{x_i-\mu_1}{\sigma_1^2}}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2})}$$

$$\frac{\partial \ell(\theta)}{\partial \mu_2} = \sum_{i=1}^{n} \frac{(1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2} \cdot \frac{x_i-\mu_2}{\sigma_2^2}}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2}}$$

$$\frac{\partial \ell(\theta)}{\partial \sigma_1} = \sum_{i=1}^{n} \frac{\frac{p_1}{\sqrt{2\pi}} \cdot (-2{\sigma_1}^{-3}) \cdot e^{-(x_i-\mu_1)^2/2\sigma_1^2} \cdot (-\frac{(x_i-\mu_1)^2}{2}) \cdot (-2{\sigma_1}^{-3})}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2}}$$

$$= \sum_{i=1}^{n} \frac{\frac{-2p_1(x_i-\mu_1)^2}{\sqrt{2\pi}} \cdot \frac{e^{-(x_i-\mu_1)^2/2\sigma_1^2}}{\sigma_1^6}}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2}}$$

$$\frac{\partial \ell(\theta)}{\partial \sigma_2} = \sum_{i=1}^{n} \frac{\frac{1-p_1}{\sqrt{2\pi}} \cdot (-2{\sigma_2}^{-3}) \cdot e^{-(x_i-\mu_2)^2/2\sigma_2^2} \cdot (-\frac{(x_i-\mu_2)^2}{2}) \cdot (-2{\sigma_2}^{-3})}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2}}$$

$$= \sum_{i=1}^{n} \frac{\frac{-2(1-p_1)(x_i-\mu_2)^2}{\sqrt{2\pi}} \cdot \frac{e^{-(x_i-\mu_2)^2/2\sigma_2^2}}{\sigma_2^6}}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2}}$$

$$\frac{\partial \ell(\theta)}{\partial p_1} = \sum_{i=1}^{n} \frac{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} - \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2})}{p_1 \cdot \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-(x_i-\mu_1)^2/2\sigma_1^2} + (1-p_1) \cdot \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-(x_i-\mu_2)^2/2\sigma_2^2}}$$

b)

Find the MLE for θ by

1)

Applying a steepest ascent iteration $\theta^{(i+1)} = \theta^{(i)} + s\nabla\ell(\theta)$.

```
data <- readLines("Hope Heights.txt")
n <- length(data) - 7 # skip the first 7 comment rows
gender <- numeric(n)
height <- numeric(n)
for (i in 1:n){
  gender[i]<- as.numeric(strsplit(data[7+i], " ")[[1]][1])
  height[i] <- as.numeric(strsplit(data[7+i], " ")[[1]][3])
}
mu.0 <- mean(height)
sigma.0 <- sd(height)
```

```r
likelihood <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return (sum(p1/(sqrt(2*pi)*sigma.1^2)*exp(-(x-mu.1)^2/2*sigma.1^2)
        + (1-p1)/(sqrt(2*pi)*sigma.2^2)*exp(-(x-mu.2)^2/2*sigma.2^2)))
}
loglikelihood <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return (sum(log(p1/(sqrt(2*pi)*sigma.1^2)*exp(-(x-mu.1)^2/2*sigma.1^2)
        + (1-p1)/(sqrt(2*pi)*sigma.2^2)*exp(-(x-mu.2)^2/2*sigma.2^2))))
}
gradient.mu.1 <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return(sum((p1/sqrt(2*pi)*sigma.1^2*exp(-(x-mu.1)^2/(2*sigma.1^2))*((x-mu.1)/sigma.1^2))
           /likelihood(mu.1,mu.2, sigma.1, sigma.2,p1,x)))
}
gradient.mu.2 <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return(sum(((1-p1)/sqrt(2*pi)*sigma.2^2*exp(-(x-mu.2)^2/(2*sigma.2^2))*((x-mu.2)/sigma.2^2))
           /likelihood(mu.1,mu.2,sigma.1, sigma.2,p1,x)))
}
gradient.sigma.1 <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return(sum((-2*p1*(x-mu.1)^2*exp(-(x-mu.1)^2/2*sigma.1^2)/sqrt(2*pi)/sigma.1^6)
           /likelihood(mu.1,mu.2, sigma.1, sigma.2,p1,x)))
}
gradient.sigma.2 <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return(sum((-2*(1-p1)*(x-mu.2)^2*exp(-(x-mu.2)^2/2*sigma.2^2)/sqrt(2*pi)/sigma.2^6)
           /likelihood(mu.1,mu.2, sigma.1, sigma.2,p1,x)))
}
gradient.p1 <- function(mu.1,mu.2,sigma.1,sigma.2, p1, x){
  return(sum((exp(-(x-mu.1)^2/2*sigma.1^2)/sqrt(2*pi)/sigma.1^2
        -exp(-(x-mu.2)^2/2*sigma.2^2)/sqrt(2*pi)/sigma.2^2)
        /likelihood(mu.1,mu.2, sigma.1, sigma.2,p1,x)))
}
```

```r
# Take a look at sample mean and standard deviation
# to use as reference for parameter initialization
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(magrittr)
df <- data.frame("gender" = gender, "height" = height)
df %>% group_by(gender) %>% summarise(mean = mean(height), sd = sd(height))
```

```
## # A tibble: 2 x 3
##   gender  mean    sd
##    <dbl> <dbl> <dbl>
## 1      1  66.4  2.92
## 2      2  72.4  2.67
```

```r
steepest_ascent <-  function(mu.1.0, mu.2.0, sigma.1.0, sigma.2.0, p1.0,
                              step.size, thresh, x){
  # Initialization
  mu.1 <- mu.1.0
  mu.2 <- mu.2.0
  sigma.1 <- sigma.1.0
  sigma.2 <- sigma.2.0
  p1 <- p1.0
  s <- step.size
  l.old <- loglikelihood(mu.1, mu.2,sigma.1,sigma.2, p1, x)
  iter.counter <- 0
  # Update
  repeat {
    mu.1 <- mu.1 + s*gradient.mu.1(mu.1,mu.2,sigma.1,sigma.2, p1, x)
    mu.2 <- mu.2 + s*gradient.mu.2(mu.1,mu.2,sigma.1,sigma.2, p1, x)
    sigma.1 <- sigma.1 + s*gradient.sigma.1(mu.1,mu.2,sigma.1,sigma.2, p1, x)
    sigma.2 <- sigma.2 + s*gradient.sigma.2(mu.1,mu.2,sigma.1,sigma.2, p1, x)
    p1 <- p1 + s* gradient.p1(mu.1,mu.2,sigma.1,sigma.2, p1, x)
    l.new <- loglikelihood(mu.1, mu.2,sigma.1,sigma.2, p1, x)
    iter.counter <- iter.counter + 1
    if (iter.counter %% 1000 == 0){
      cat ( iter.counter, " Iterations  ")
      cat(" Updated likelihood", l.new, "\n")
    } #cat("l.old", l.old, "\n")
    if (l.new-l.old < thresh ){
      break
    }
    else{
      l.old <- l.new
    }
  }
  cat("Parameters: \n")
  cat("mu.1 = ", mu.1, "\n")
  cat("mu.2 = ", mu.2, "\n")
  cat("sigma.1 = ", sigma.1, " \n")
  cat("sigma.2 = ", sigma.2, " \n")
  cat("p1 = ", p1, "\n")
  return(c(mu.1, mu.2, sigma.1, sigma.2, p1))
}
```

```r
parameters <- steepest_ascent(65, 67, 2, 2, .4, step.size = .0001, thresh = .000001, x = height)
```

```
## 1000  Iterations   Updated likelihood -4063.544
## 2000  Iterations   Updated likelihood -3788.945
## 3000  Iterations   Updated likelihood -3576.983
## 4000  Iterations   Updated likelihood -3426.118
## 5000  Iterations   Updated likelihood -3322.381
## 6000  Iterations   Updated likelihood -3252.811
## 7000  Iterations   Updated likelihood -3209.266
## 8000  Iterations   Updated likelihood -3186.626
## Parameters:
## mu.1 =  66.8152
## mu.2 =  68.41419
## sigma.1 =  1.994078
```

```
## sigma.2 =  1.973996
## p1 =  0.2560473
```

2)

Using *nlm* or an equivalent in Python.

```
fn <- function (theta){
  # theta: a parameter vector contains (mu.1, mu.2, sigma.1, sigma.2, p1)
  return (-sum(log(theta[5]/(sqrt(2*pi)*theta[3]^2)*exp(-(height-theta[1])^2/2*theta[3]^2)
          + (1-theta[5])/(sqrt(2*pi)*theta[4]^2)*exp(-(height-theta[2])^2/2*theta[4]^2))))
}
nlm(fn, c(65, 67, 2, 2, .4), print.level = 2, hessian = TRUE)
```

```
## iteration = 0
## Step:
## [1] 0 0 0 0 0
## Parameter:
## [1] 65.0 67.0  2.0  2.0  0.4
## Function Value
## [1] 4399.023
## Gradient:
## [1]   122.37837 -1233.57374   261.22426  3952.81709    89.99643

## Warning in nlm(fn, c(65, 67, 2, 2, 0.4), print.level = 2, hessian = TRUE):
## NA/Inf replaced by maximum positive value

## Warning in nlm(fn, c(65, 67, 2, 2, 0.4), print.level = 2, hessian = TRUE):
## NA/Inf replaced by maximum positive value

## Warning in log(theta[5]/(sqrt(2 * pi) * theta[3]^2) * exp(-(height -
## theta[1])^2/2 * : NaNs produced

## Warning in nlm(fn, c(65, 67, 2, 2, 0.4), print.level = 2, hessian = TRUE):
## NA/Inf replaced by maximum positive value

## iteration = 1
## Step:
## [1] -0.12237837  1.23357374 -0.26122426 -3.95281709 -0.08999643
## Parameter:
## [1] 64.8776216 68.2335737  1.7387757 -1.9528171  0.3100036
## Function Value
## [1] 2897.099
## Gradient:
## [1]    72.57140  -823.51413   233.03117 -2580.80254    41.41791
##
## iteration = 2
## Step:
## [1] -0.02155762  0.25667097 -0.07944979  1.41134306 -0.01073890
## Parameter:
## [1] 64.8560640 68.4902447  1.6593260 -0.5414740  0.2992647
## Function Value
## [1] 260.0843
## Gradient:
## [1]     1.921032   -29.276072     4.203549 -1295.842393   133.283418
##
```

```
## iteration = 3
## Step:
## [1]  0.00957850 -0.09835587  0.02567478  0.58055143 -0.05238885
## Parameter:
## [1] 64.86564250 68.39188884  1.68500074  0.03907739  0.24687582
## Function Value
## [1] -526.8411
## Gradient:
## [1] -2.160861e-04 -1.554706e-01  2.072608e-03  5.187322e+03  1.327743e+02
##
## iteration = 4
## Step:
## [1] -0.001138827  0.012347277 -0.002982116 -0.059034320 -0.005888564
## Parameter:
## [1] 64.86450368 68.40423611  1.68201862 -0.01995693  0.24098726
## Function Value
## [1] -663.0154
## Gradient:
## [1] -5.570549e-05 -4.005649e-02  5.238188e-04 -1.005719e+04  1.317486e+02
##
## iteration = 5
## Step:
## [1]  0.0004655496 -0.0048989621  0.0011269715  0.0389011036 -0.0835994876
## Parameter:
## [1] 64.86496923 68.39933715  1.68314560  0.01894417  0.15738777
## Function Value
## [1] -683.9148
## Gradient:
## [1] -2.949041e-05 -3.626980e-02  2.768640e-04  1.059065e+04  1.186775e+02
##
## iteration = 6
## Step:
## [1] -0.0005106942  0.0055584508 -0.0013802710 -0.0194999537 -0.0432342853
## Parameter:
## [1] 64.864458534 68.404895604  1.681765324 -0.000555779  0.114153486
## Function Value
## [1] -1395.013
## Gradient:
## [1] -1.752683e-08 -3.104559e-05  2.703988e-07 -3.601804e+05  1.128864e+02
##
## iteration = 7
## Step:
## [1]  1.157045e-05 -1.215938e-04  2.785997e-05  9.915854e-04 -2.221802e-03
## Parameter:
## [1] 6.486447e+01 6.840477e+01 1.681793e+00 4.358064e-04 1.119317e-01
## Function Value
## [1] -1443.898
## Gradient:
## [1] -1.051610e-08 -1.909274e-05  1.351972e-07  4.583944e+05  1.126040e+02
##
## iteration = 8
## Step:
## [1] -0.0002683904  0.0029757447 -0.0007855070 -0.0005550005 -0.0791589229
## Parameter:
```

```
## [1] 64.864201714 68.407749755  1.681007677 -0.000119194  0.032772761
## Function Value
## [1] -1711.726
## Gradient:
## [1]  0.000000e+00 -1.422586e-06  0.000000e+00 -1.685015e+06  1.033884e+02
##
## iteration = 9
## Step:
## [1] -2.249183e-05  2.508799e-04 -6.717972e-05  1.714250e-04 -7.902701e-03
## Parameter:
## [1] 6.486418e+01 6.840800e+01 1.680940e+00 5.223092e-05 2.487006e-02
## Function Value
## [1] -1877.555
## Gradient:
## [1]  0.000000e+00 -2.725506e-07  0.000000e+00  3.792955e+06  1.025505e+02
##
## iteration = 10
## Step:
## [1] -1.079806e-04  1.198383e-03 -3.170781e-04 -5.443071e-05 -3.283097e-02
## Parameter:
## [1]  6.486407e+01  6.840920e+01  1.680623e+00 -2.199792e-06 -7.960913e-03
## Function Value
## [1] -2514.329
## Gradient:
## [1]  0.000000e+00  6.647459e-09  0.000000e+00 -1.212429e+08  9.921025e+01
##
## iteration = 11
## Step:
## [1] -8.520394e-06  9.461099e-05 -2.506498e-05  3.020964e-06 -2.633182e-03
## Parameter:
## [1]  6.486406e+01  6.840929e+01  1.680598e+00  8.211718e-07 -1.059410e-02
## Function Value
## [1] -2711.667
## Gradient:
## [1] 0.000000e+00 0.000000e+00 0.000000e+00 1.593006e+08 9.895175e+01
##
## iteration = 12
## Step:
## [1] -1.162446e-04  1.290497e-03 -3.417032e-04 -8.283250e-07 -3.567997e-02
## Parameter:
## [1]  6.486395e+01  6.841058e+01  1.680257e+00 -7.153142e-09 -4.627406e-02
## Function Value
## [1] -3663.773
## Gradient:
## [1] 0.000000e+00 0.000000e+00 0.000000e+00 9.866049e+08 9.557730e+01
##
## iteration = 13
## Step:
## [1] -2.367283e-04  2.628066e-03 -6.958778e-04  8.316681e-10 -7.267090e-02
## Parameter:
## [1]  6.486371e+01  6.841321e+01  1.679561e+00 -6.321474e-09 -1.189450e-01
## Function Value
## [1] -3695.207
## Gradient:
```

```
## [1]    0.0000    0.0000    0.0000 -126.4295   89.3699
##
## iteration = 14
## Step:
## [1] -3.408279e-03  3.783739e-02 -1.001885e-02  1.197266e-08 -1.046274e+00
## Parameter:
## [1]  6.486030e+01  6.845105e+01  1.669542e+00  5.651189e-09 -1.165219e+00
## Function Value
## [1] -3783.638
## Gradient:
## [1]   0.0000   0.0000   0.0000 113.0238  46.1847
##
## iteration = 15
## Parameter:
## [1]  6.486030e+01  6.845105e+01  1.669542e+00  5.651189e-09 -1.165219e+00
## Function Value
## [1] -3783.638
## Gradient:
## [1]   0.0000   0.0000   0.0000 113.0238  46.1847
##
## Last global step failed to locate a point lower than x.
## Either x is an approximate local minimum of the function,
## the function is too non-linear for this algorithm,
## or steptol is too large.

## $minimum
## [1] -3783.638
##
## $estimate
## [1]  6.486030e+01  6.845105e+01  1.669542e+00  5.651189e-09 -1.165219e+00
##
## $gradient
## [1]   0.0000   0.0000   0.0000 113.0238  46.1847
##
## $hessian
##              [,1]         [,2]          [,3]          [,4]          [,5]
## [1,] 0.000000e+00  0.000000000  0.0000000000  2.453914e-05  0.000000e+00
## [2,] 0.000000e+00  0.000000000  0.0000000000 -9.556191e-03  0.000000e+00
## [3,] 0.000000e+00  0.000000000  0.0000000000 -2.179028e-04  0.000000e+00
## [4,] 2.453914e-05 -0.009556191 -0.0002179028 -1.817599e+11 -2.273737e-05
## [5,] 0.000000e+00  0.000000000  0.0000000000 -2.273737e-05  2.133229e+01
##
## $code
## [1] 3
##
## $iterations
## [1] 15
```

c)

Given your MLE in (b), use the distribution of X to predict whether a given sample is taken from a man or woman. Intuitively, the two normal distributions of X correspond to the male and female height distributions. Given a sample, $\hat{X}$, decide which normal the sample is most likely to come from and assign the gender accordingly. Determine what percentage of individuals are classified correctly.

```r
mu.1 <- parameters[1]
mu.2 <- parameters[2]
sigma.1 <- parameters[3]
sigma.2 <- parameters[4]
p1 <- parameters[5]
x <- height
z.score.f <- abs((x-mu.1)/sigma.1)
z.score.m <- abs((x- mu.2)/sigma.2)
```

```r
gender.pred <- ifelse(z.score.f < z.score.m, 1, 2)
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
confusion.matrix <-  as.matrix(table(gender, gender.pred))
confusion.matrix
```

```
##       gender.pred
## gender  1  2
##      1 33 17
##      2  2 48
```

```r
cat("\nThe percentage of individuals that are classified correctly is: ",
    sum(diag(confusion.matrix))/sum(confusion.matrix)*100, "%")
```

```
##
## The percentage of individuals that are classified correctly is:  81 %
```

2.

a)

Let Q be an $n \times n$ orthonormal matrix. Show that $Q^{-1} = Q^T$

Solution

We first show that $QQ^T = Q^TQ = I$ Let $v = v_1, v_2, ..., v_n$ be column vectors of Q, by definition, we have

$$v_i^T v_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

When we multiply Q^T with Q, let q_{ij} indicate the i_{th} row, j_{th} column of the result. When $i \neq j$, q_{ij} is the dot product of the i_{th} row of Q^T (or the i_{th} column of Q) with the j_th row of Q, which gives 0, since $v_1, v_2, ..., v_n$ are orthogonal;when $i = j$, $q_{ij}$is the dot product of the i_{th} row of Q^T (or the i_{th} column of Q) with the i_{th} row of Q, which gives 1, since $v_1, v_2, ..., v_n$ are normal. Thus, $Q^TQ = I$. Symmetrically, we also have $QQ^T = I$.

Since the columns vectors v of Q are orthonormal vectors, v_i, v_j are linearly independent, thus Q is invertible. In other words, Q^{-1} exists.

Use the transpose formula above, we have

$$Q^TQ = I \Longrightarrow Q^TQQ^{-1} = IQ^{-1} \Longrightarrow Q^TQQ^{-1} = Q^{-1} \Longrightarrow Q^T(QQ^{-1}) = Q^{-1} \Longrightarrow Q^T = Q^{-1}$$

b)

Show that R rotates vectors by an angle θ and that F reflects vectors about the x-axis.

Solution

Let A be a 2×1 matrix with length L, let the angle between vector A and the x-axis be α, then we can write A as

$$A = \begin{pmatrix} Lcos(\alpha) \\ Lsin(\alpha) \end{pmatrix}$$

Thus,

$$RA = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} Lcos(\alpha) \\ Lsin(\alpha) \end{pmatrix} = \begin{pmatrix} L(\cos(\theta)cos(\alpha) - \sin(\theta)\sin(\alpha)) \\ L(\sin(\theta)cos(\alpha) + \cos(\theta)\sin(\alpha)) \end{pmatrix}$$

Use angle addition formulas, we have

$$\cos(\theta)cos(\alpha) - \sin(\theta)\sin(\alpha) = cos(\theta + \alpha)$$
$$\sin(\theta)cos(\alpha) + \cos(\theta)\sin(\alpha) = sin(\theta + \alpha)$$

Thus,

$$RA = \begin{pmatrix} L(\cos(\theta + \alpha) \\ L(\sin(\theta + \alpha) \end{pmatrix}$$

Therefore, geometrically, left multiply by R is equivalent to rotate A by an angle θ.

As for FA, we have:

$$FA = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} Lcos(\alpha) \\ Lsin(\alpha) \end{pmatrix} = \begin{pmatrix} Lcos(\alpha) \\ -Lsin(\alpha) \end{pmatrix}$$

So after left multiply by F, x-coordinate $Lcos(\alpha)$ remains the same, while the y-coordinate $(-Lsin(\alpha))$ becomes the negative of the original value of y. In other words, left multiply by F reflects vectors about the x-axis.

c)

Show that you can form any 2×2 orthonormal matrix using R and RF.

Solution

$$RF = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} cos(\theta) & -\sin(\theta) \\ sin(\theta) & -\cos(\theta) \end{pmatrix}$$

Let A denotes an orthonormal matrix with entry a, b, c, d as below.

$$A = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$$

where a, b, c, d satisfies the following equations by the orthonormal property:

$$\begin{cases} a^2 + b^2 = 1 \\ c^2 + d^2 = 1 \\ ac + bd = 0 \end{cases}$$

Imagine a unit circle, we may find that there exists two angles $\alpha, \beta \in [0, 2\pi)$ such that

$$a = cos(\alpha), \ b = sin(\alpha), \ c = sin(\beta), \ d = cos(\beta)$$

The third equation therefore can be written as (using the angle addition formulas)

$$cos(\alpha)sin(\beta) + sin(\alpha)cos(\beta) = 0 \Longrightarrow sin(\alpha + \beta) = 0$$

By the property of $sin()$ function, we have $\alpha + \beta = k\pi, k \in N$. Due to the periodicity of the $sin()$ function, we only need to consider two situations $k = 0$ or $k = 1$.

When $k = 0$, we have $\alpha = -\beta$,

$$\begin{aligned} a &= cos(\alpha), \\ b &= sin(\alpha), \\ c &= sin(\beta) = sin(-\alpha) = -sin(\alpha), \\ d &= cos(\beta) = cos(-\alpha) = cos(\alpha). \end{aligned}$$

A can be written as R.

When $k = 1$, we have $\alpha + \beta = \pi$, which gives

$$\begin{aligned} a &= cos(\alpha), \\ b &= sin(\alpha), \\ c &= sin(\beta) = sin(\pi - \alpha) = sin(\alpha), \\ d &= cos(\beta) = cos(\pi - \alpha) = -cos(\alpha). \end{aligned}$$

A can be written as RF.

3.

Let $X \sim (\mu, \Sigma)$ where $\mu \in \mathbb{R}^n$ and Σ is an $n \times n$ covariance matrix.

a)

Let M be an $n \times n$ invertible matrix. Show that $MX \sim (M\mu, M\Sigma M^T)$.

Solution

We use the characteristic function of multinormal distribution to prove the theorem. By definition of the multinormal distribution using characteristic function (see reference here: https://www.math.kth.se/matstat/gru/sf2940/sf2940lectVI3.pdf), since $X \sim (\mu, \Sigma)$, the corresponding characteristic function is

$$\phi_X(t) = exp(it^T\mu - \frac{1}{2}t^T\Sigma t)$$

where i is the imaginary unit and t is a $n \times 1$ row vector.

Assume m_i is the i_{th} row of the $n \times n$ invertible matrix M. We then prove that $m_iX \sim N(m_i\mu, m_i\Sigma m_i^T)$.

$$\begin{aligned} \phi_{m_iX}(t) &= E[exp(it(m_iX))] \\ &= E[exp(i(t^Tm_i)X] \\ &= E[exp(i(m_i^Tt)^TX)] \\ &= E[exp(i(m_i^Tt)^TX)] \end{aligned}$$

Let $t^* = m_i^T t$, we then have

$$\begin{aligned}
\phi_{m_i X}(t) &= E[exp(i(m_i^T t)^T X)] \\
&= E[exp(i(t^*)^T X)] \\
&= exp(i(t^*)^T \mu - \frac{1}{2}(t^*)^T \Sigma (t^*)) \\
&= exp(i(m_i^T t)^T \mu - \frac{1}{2}(m_i^T t)^T \Sigma (m_i^T t)) \\
&= exp(i(t^T m_i \mu - \frac{1}{2} t^T m_i \Sigma (m_i^T t)) \\
&= exp(i(t^T (m_i \mu)) - \frac{1}{2} t^T (m_i \Sigma m_i^T) t)
\end{aligned}$$

Thus, we have $m_i X \sim N(m_i\mu, m_i \Sigma m_i^T)$. Since

$$M = \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ . \\ m_n \end{pmatrix}$$

$$MX = \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ . \\ m_n \end{pmatrix} X = \begin{pmatrix} m_1 X \\ m_2 X \\ . \\ . \\ . \\ m_n X \end{pmatrix}$$

where $m_i X \sim N(m_i\mu, m_i \Sigma m_i^T),\ i = 1, 2, ..., n$. This is equivalent to $MX \sim (M\mu, M\Sigma M^T)$

b)

Show that if Σ is a diagonal matrix then the coordinates of X are independent normals.

Solution:

Since Σ is a diagonal matrix, it follows that there exists eigenvectors $q^{(1)}, q^{(2)}, ..., q^{(n)}$ and associated eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ such that $q^{(i)} \cdot q^{(j)} = 0 \ \forall \ i \neq j$ and $||q^{(i)}|| = 1$.

Thus, $\Sigma = QDQ^T$ where

$$D = \begin{pmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ . & . & . & . \\ . & . & . & . \\ 0 & ... & 0 & \lambda_n \end{pmatrix} Q = (q^{(1)}, q^{(2)}, ..., q^{(n)})$$

By the theorem in a), we have $MX \sim (M\mu, M\Sigma M^T)$. Since $X \sim N(\mu, \Sigma) \iff X \sim \mu + N(0, \Sigma)$, we may only consider the situation where $X \sim N(0, \Sigma)$. Left multiply X by Q^T, we have

$$
\begin{aligned}
Q^T X &\sim (0, Q^T \Sigma Q) \\
&= (0, Q^T Q D Q^T Q) \\
&= (0, (Q^T Q) D (Q^T Q)) \\
&= (0, IDI) \\
&= (0, D)
\end{aligned}
$$

Let $Z = Q^T X \sim (0, D)$, then $Z_i \sim N(0, D_{ii})$. Thus $f(z) = \frac{1}{\sqrt{(2\pi)^n \det D}} e^{-(z^T D^{-1} z)/2}$.
Since D is diagonal, we have $\det D = D_{11} \cdot D_{22} \cdot ... \cdot D_{nn}$.

$$
Z^T D^{-1} Z = (Z_1, Z_2, ..., Z_n) \begin{pmatrix} \frac{1}{D_{11}} & 0 & ... & 0 \\ 0 & \frac{1}{D_{22}} & ... & 0 \\ . & . & . & . \\ . & . & . & . \\ 0 & ... & 0 & \frac{1}{D_{nn}} \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \\ . \\ . \\ Z_n \end{pmatrix} = \sum_{i=1}^{n} \frac{Z_i^2}{D_{ii}}
$$

Plug in the two expressions above to $f(z)$, we have

$$
\begin{aligned}
f(z) &= \frac{1}{\sqrt{(2\pi)^n \det D}} e^{-(z^T D^{-1} z)/2} \\
&= \frac{1}{\sqrt{(2\pi)^n (D_{11} \cdot D_{22} \cdot ... \cdot D_{nn})}} e^{-\frac{1}{2} \sum_{i=1}^{n} \frac{z_i^2}{D_{ii}}} \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi) D_{ii}}} e^{-\frac{z_i^2}{2D_{ii}}} \\
&= f_{Z_1}(z_1) \cdot f_{Z_2}(z_2) \cdot ... \cdot f_{Z_n}(z_n)
\end{aligned}
$$

Thus, $Z_1, Z_2, ..., Z_n$ are independent.

3)

Write a function **MultiNorm(μ, Σ)** that samples from a multivariate normal with mean μ and covariance Σ. Your function can use **rnorm**, the univariate normal sampler in R, and the function **eigen** (or their equivalent in Python).

Solution:

Based on the previous conclusion, we know that given $X \sim N(\mu, \Sigma)$, we can rewrite Σ through the spectral decompostion, i.e., $\Sigma = QDQ^T$, where Q is the eigen vector, D is a diagonal matrix with eigen values on the diagonal. Since $X \sim N(\mu, \Sigma) \iff X \sim \mu + N(0, \Sigma)$. Thus, assume $X \sim N(0, \Sigma)$, then

$$
Q^T X \sim N(0, Q^T (QDQ^T) Q) \implies Q^T X \sim N(0, D)
$$

Let $Z = Q^T X \sim N(0, D)$, we can first sample from $N(0, D)$ to get Z, then transform back to $X \sim (0, \Sigma)$ by left multiply Q as $QZ = QQ^T X = X$. After that, we can add back mu to recover the original X.

```r
# $Id: mvnorm.R 332 2016-10-27 09:17:12Z thothorn $
# Code for checking edge cases is adapted from
# https://rdrr.io/cran/mvtnorm/src/R/mvnorm.R

MultiNorm <- function(N, mu, sigma){
  if (!isSymmetric(sigma, tol = sqrt(.Machine$double.eps),
                     check.attributes = FALSE)) {
        stop("sigma must be a symmetric matrix")
    }
    if (length(mu) != nrow(sigma)){
        stop("mean and sigma have non-conforming size")
    }
    # Sigma = Q %*% D  %*% t(Q)
    ev <- eigen(sigma, symmetric = TRUE)
    # Check Positive Semidefinite
    # Compare with the machine's smallest positive floating-point number
    if (!all(ev$values >= -sqrt(.Machine$double.eps) * abs(ev$values[1]))){
        warning("sigma is numerically not positive semidefinite")
    }
    Q <- ev$vectors
    D <- diag(x = ev$values, nrow = length(ev$values), ncol =  length(ev$values))
    # When X ~ N(0, Sigma),  Z = t(Q) %*% X ~ N(0,D)
    Z <- numeric(0)
    for (i in 1:nrow(D)){
      # Z1, Z2, Z3, ... follows N(0, D11), N(0, D22), N(0, D33), ...
      Z <- rbind(Z, rnorm(N, sd = sqrt(D[i,i])))
    }
    #  Z = t(Q) %*% X
    #  Q %*% Z = X
    # Thus we could transform back to X ~ N(0, Sigma) by left multiply Q
    X <- Q %*% Z
    # Since X ~ N(mu, Sigma) is equivalent to mu +  N(0, Sigma),
    # We can simply add back the mean vector
    X <- sweep(X, MARGIN = 1, mu, "+")
    return(X)
    }
```

```r
mu <- c(1,2,3)
sigma <- matrix(c(4,2,1,2,3,2,1,2,3), ncol=3)
X <-MultiNorm (N=500, mu=mu, sigma=sigma)
cat("Sample Means: ",rowMeans(X), "\n")
```

```
## Sample Means:  1.065312 2.056025 3.12102
```

```r
cat("Sample Covariance Matrix:\n")
```

```
## Sample Covariance Matrix:
```

```r
var(t(X))
```

```
##           [,1]     [,2]      [,3]
## [1,] 3.8735499 2.014707 0.9635637
## [2,] 2.0147066 3.053519 1.9237297
## [3,] 0.9635637 1.923730 2.7163804
```

The sample means and covariance matrix are both very close to the assigned parameters.