Assignment 2

Mathematics for Big Data

Sergi Vilardell

Contents


```r
library(mnormt)
library(fBasics)
```

These exercises cover matrix decompositions: SD and SVD.

Exercise 2.1

Generate a multivariate Gaussian sample, `data`, of size 10^4:

```r
#Mean
mu <- c(5, 1, -1, -4)

#Covariance
sigma <- matrix(c(3,  1.385, -2.939, 0,
              1.385,      1, -1.979, 0,
             -2.939, -1.979,      8, 0,
                  0,      0,      0, 8),nrow = 4,ncol = 4)

#Data
set.seed(137)
data <- rmnorm(n=10^4,mu, sigma)
data <- as.matrix(data)
```

Exercise 2.2

Get the sample mean vector of the generated data:

```r
#Mean
m <- apply(data, 2, mean)
m
```

```
## [1]  5.022736  1.012500 -1.021740 -4.001446
```

```r
mu
```

```
## [1]  5  1 -1 -4
```

```r
#Covariance
S <- cov(data)
S
```

```
##              [,1]        [,2]        [,3]         [,4]
## [1,]  2.917889946  1.36376690 -2.90782389  0.001756047
## [2,]  1.363766897  1.00199595 -1.97970806  0.031201150
## [3,] -2.907823891 -1.97970806  8.09416087 -0.025994727
## [4,]  0.001756047  0.03120115 -0.02599473  8.002386700
```

```r
sigma
```

```
##        [,1]   [,2]   [,3] [,4]
## [1,]  3.000  1.385 -2.939    0
## [2,]  1.385  1.000 -1.979    0
## [3,] -2.939 -1.979  8.000    0
## [4,]  0.000  0.000  0.000    8
```

The results are as expected, the data is large enough to produce the expected mean and covariance but not large enough to be almost equal.

Exercise 2.3

Obtain the dimension and the rank of the file `data`, and the same for matrix S (use functions `dim()` and `rk()`)

```r
#Dimension
dim(data)
```

```
## [1] 10000     4
```

```r
dim(S)
```

```
## [1] 4 4
```

```r
#Rank
rk(data)
```

```
## [1] 4
```

```r
rk(S)
```

```
## [1] 4
```

Exercise 2.4

Now we center the data. The centering is done by substracting the mean of each column in their corresponding rows:

```r
#Center the data
X <- sweep(data, 2, m, "-")
X <- as.matrix(X)
head(X)
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.6415403 1.1150161 -2.739775 3.8174299
## [2,] 0.5020686 0.5424759  1.268530 0.6130546
## [3,] 1.9346595 0.5307002 -1.567181 4.0178013
```

```
## [4,] -0.9287143 -0.2423676  2.442640  1.6412756
## [5,] -1.1566803 -0.6883247  1.644937  0.8717114
## [6,] -3.2848933 -1.7643278  4.573853 -2.4849477
```

```r
tail(X)
```

```
##                [,1]       [,2]       [,3]       [,4]
##  [9995,] -1.8573880 -1.1655174  5.6440268 -0.2534020
##  [9996,] -2.0642419 -1.2254864  2.2020316  3.8250590
##  [9997,] -0.9098976  0.3572214 -1.5897127  1.5378890
##  [9998,]  1.8477577  1.3032174 -8.1093954  0.4607504
##  [9999,]  0.1341955 -0.9156558 -0.4047195 -1.8550333
## [10000,]  1.6322046  1.2519248 -0.7405977 -2.2336369
```

```r
rk(X)
```

```
## [1] 4
```

Exercise 2.5

Check the equality $S = \frac{1}{n-1} X^t X$:

```r
X.t <- t(X)
c <- 1/(10^4-1)
S.new <- c*X.t%*%X
```

Compare objects with `all.equal()`, which compare `R` objects and test 'near equality'

```r
all.equal(S, S.new)
```

```
## [1] TRUE
```

Exercise 2.6

Spectral decomposition of $A = S \times (n-1)$, where S is the sample covariance matrix:

```r
A <- as.matrix(S.new/c)
sd.A <- eigen(A)
diag(sd.A$values)
```

```
##        [,1]     [,2]     [,3]     [,4]
## [1,] 100198     0.00     0.00    0.000
## [2,]      0 80011.82     0.00    0.000
## [3,]      0     0.00 17411.48    0.000
## [4,]      0     0.00     0.00 2522.996
```

```r
sd.A$vectors
```

```
##             [,1]          [,2]         [,3]         [,4]
## [1,] -0.40777689 -0.0078313238 0.8360454893  0.366994025
## [2,] -0.25419119 -0.0009819457 0.2820956274 -0.925098877
## [3,]  0.87684788  0.0138357262 0.4705846349 -0.097449825
## [4,] -0.01557684  0.9998731314 0.0003135051  0.003314362
```

a) Compare the **SD** of A with the **SD** of S:

```r
sd.S <- eigen(S)
sd.S
```

```
## $values
## [1] 10.0208037  8.0019826  1.7413224  0.2523248
##
## $vectors
##             [,1]          [,2]         [,3]         [,4]
## [1,] -0.40777689 -0.0078313238 0.8360454893  0.366994025
## [2,] -0.25419119 -0.0009819457 0.2820956274 -0.925098877
## [3,]  0.87684788  0.0138357262 0.4705846349 -0.097449825
## [4,] -0.01557684  0.9998731314 0.0003135051  0.003314362
```

```
all.equal(sd.S, sd.A)
```

```
## [1] "Component \"values\": Mean relative difference: 9998"
```

The eigenvectors are the same and the difference between the two **SD** is in the eigenvalues, but the are proportional between them, this is because the factor $1/(n-1)$ that is present in the **SD** of S.

b) Check the ortogonality of V:

```
V <- as.matrix(sd.A$vectors)
V.t <- t(V)
I <- diag(4)
all.equal(I, V.t%*%V)
```

```
## [1] TRUE
```

```
all.equal(I, V%*%V.t)
```

```
## [1] TRUE
```

The orthogonality is satisfied as expected.

c) Check the Jordan theorem equality (**SD**): $A = V \Lambda V^t$:

```
D <- diag(sd.A$values)
A.new <- V%*%D%*%V.t
all.equal(A, A.new)
```

```
## [1] TRUE
```

The Jordan Theorem equality is satisfied as expected.

d) Compute the matrix Q that factorizes A and check the factorization property. Q is defined as $Q := V\Lambda^{1/2}$:

```
#Factorization
sqrt.D <- diag(sqrt(sd.A$values))
Q <- V%*%sqrt.D
Q
```

```
##             [,1]        [,2]         [,3]        [,4]
## [1,] -129.077984  -2.2151965 110.31835615  18.4339017
## [2,]  -80.461858  -0.2777567  37.22324478 -46.4671917
## [3,]  277.558043   3.9136235  62.09485490  -4.8948494
## [4,]   -4.930703 282.8277273   0.04136781   0.1664785
```

```
Q.t <- t(Q)
all.equal(A,Q%*%Q.t)
```

```
## [1] TRUE
```

The factorization property is satisfied.

e) Compare all the possible rank-2 approximations to A. We can see that the next operation is equal to A:

```
test <- cbind(Q[,1])%*%t(Q[,1])+
        cbind(Q[,2])%*%t(Q[,2])+
        cbind(Q[,3])%*%t(Q[,3])+
        cbind(Q[,4])%*%t(Q[,4])

all.equal(A, test)
```

```
## [1] TRUE
```

We take just two elements from that sum to make a rank-2 approximation, and see which is the best using the norm $(\sum_{i,j}(a_{ij} - a_{ij}^{su})^2)^{1/2}$. First we create the function that computes the norm:

```
#Norm function
norm.matrix <- function(a,b){
  c <- a-b
  c <- c*c
  norm <- sum(c)
  norm <- sqrt(norm)
  return(norm)
}
```

Make another function that computes the new matrix A given the two indexs for the columns:

```
#Reduced matrix function
A.reduc <- function(a,b){
  cbind(Q[,a])%*%t(Q[,a])+cbind(Q[,b])%*%t(Q[,b])
}
```

Now compute all the combinations for the norms. Here we use the function `combn()` which generates all the combinations of pairs of numbers between 1 and 4, each pair with two different numbers:

```
#Rank-2 approx
comb <- combn(4,2)
norms <- rep(0,6)
for(i in 1:6){
  norms[i] <- norm.matrix(A.reduc(comb[1,i],comb[2,i]),A)
}
norms <- rbind(comb, norms)
norms
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
##           1.00     1.00     1.00      2.0      2.0      3.0
##           2.00     3.00     4.00      3.0      4.0      4.0
## norms 17593.33 80051.59 81884.38 100229.8 101699.6 128224.5
```

The lowest norm corresponds to the reduction using the first two vector columns from Q, which should not be surprising, as they are the vectors for the two largest eigenvalues. We can check the rank of the matrix `A.reduc`:

```
#Ranks
rk(A.reduc(1,2))
```

```
## [1] 2
```

```
rk(A)
```

```
## [1] 4
```

We succesfully reduced the rank from 4 to 2 of the matrix A.

Exercise 2.7

Compute the Singular Value Decomposition (**SVD**), i.e., the matrices U, D, V, of the centered data X, using function `svd()`.

```
#Singular Value Decomposition
svd <- svd(X)
```

a) Compare U, V and D with the matrices appearing in the **SD** of $A = X^t X$ and $B = X X^t$.

```
#SVD and SD comparison
A.sd <- X.t%*%X
B.sd <- X%*%X.t
all.equal(A.sd, svd$v)
```

```
## [1] "Mean relative difference: 0.9999988"
```

```
all.equal(B.sd, svd$u)
```

```
## [1] "Attributes: < Component \"dim\": Mean relative difference: 0.9996 >"
## [2] "Numeric: lengths (100000000, 40000) differ"
```

b) Check the equality: $X = \sum d_j u_j v_j^t$.

```
sum <- 0
for (i in 1:4){
  sum = sum + svd$d[i]*cbind(svd$u[,i])%*%t(svd$v[,i])
}
all.equal(sum, X)
```

```
## [1] TRUE
```

c) Obtain the best rank-3 approximation to X, say $\tilde{X}$. Then, compute $(\sum_{i,j}(x_{ij} - \tilde{x}_{ij})^2)^{1/2}$ and $\max_{ij} |x_{ij} - \tilde{x}_{ij}|$ to quantify the approximation error.

The best rank-3 approximation will be the vectors corresponding to the 3 biggest values from the diagonal matrix D:

```
#Rank-3 approx
X.tilde <- 0
for (i in 1:3){
  X.tilde = X.tilde + svd$d[i]*cbind(svd$u[,i])%*%t(svd$v[,i])
}
X.norm <- norm.matrix(X, X.tilde)
X.norm
```

```
## [1] 50.22943
```

Now find $\max_{ij} |x_{ij} - \tilde{x}_{ij}|$.

```
#Max value
M <- abs(X-X.tilde)
max(M)
```

```
## [1] 1.968629
```