605 - Homework 4

Jose Mawyin

9/16/2019

ASSIGNMENT 4

DATA 605 FUNDAMENTALS OF COMPUTATIONAL MATHEMATICS - 2019

1. Problem Set 1

In this problem, we'll verify using R that SVD and Eigenvalues are related as worked out in the weekly module. Given a 3 × 2 matrix A:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 4 \end{bmatrix}$$

Write code in R to compute $X = AA^T$ and $Y = A^TA$. Then, compute the eigenvalues and eigenvectors of X and Y using the built-in commans in R. Then, compute the left-singular, singular values, and right-singular vectors of A using the svd command. Examine the two sets of singular vectors and show that they are indeed eigenvectors of X and Y. In addition, the two non-zero eigenvalues (the 3rd value will be very close to zero, if not zero) of both X and Y are the same and are squares of the non-zero singular values of A. Your code should compute all these vectors and scalars and store them in variables. Please add enough comments in your code to show me how to interpret your steps.

```
##Calculation of Transpose Matrix
A <- matrix(c(1,2,3,-1,0,4), ncol = 3, byrow = TRUE)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]   -1    0    4
```

```
matrix.size <- as.data.frame(dim(A))
row.size <- matrix.size[1,1]
col.size <- matrix.size[2,1]
row.size;col.size
```

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

```
## [1] 3
```

```
A.Transpose <- c()
for (i in 1:col.size) {
  A.Transpose <- rbind(A.Transpose, A[,i] )

}
A.Transpose
```

```
##      [,1] [,2]
## [1,]    1   -1
## [2,]    2    0
## [3,]    3    4
```

```
##Matrix multiplication
X <- A%*%A.Transpose
Y <- A.Transpose%*%A
X; Y
```

```
##      [,1] [,2]
## [1,]   14   11
## [2,]   11   17
```

```
##      [,1] [,2] [,3]
## [1,]    2    2   -1
## [2,]    2    4    6
## [3,]   -1    6   25
```

```
##Calculatio of eigenvalues and eigenvectors
eigen(X); eigen(Y)
```

```
## eigen() decomposition
## $values
## [1] 26.601802  4.398198
##
## $vectors
##           [,1]       [,2]
## [1,] 0.6576043 -0.7533635
## [2,] 0.7533635  0.6576043
```

```
## eigen() decomposition
## $values
## [1] 2.660180e+01 4.398198e+00 1.058982e-16
##
## $vectors
##             [,1]       [,2]       [,3]
## [1,] -0.01856629 -0.6727903  0.7396003
## [2,]  0.25499937 -0.7184510 -0.6471502
## [3,]  0.96676296  0.1765824  0.1849001
```

```
##Calculating left-singular, singular values, and right-singular vectors of A.
svd(A)
```

```
## $d
## [1] 5.157693 2.097188
##
## $u
##            [,1]       [,2]
## [1,] -0.6576043 -0.7533635
## [2,] -0.7533635  0.6576043
```

```
##
## $v
##             [,1]       [,2]
## [1,]  0.01856629 -0.6727903
## [2,] -0.25499937 -0.7184510
## [3,] -0.96676296  0.1765824
```

```
##Comparing that the two non-zero eigenvalues  of both X and Y are the same and
##are squares of the non-zero singular values of A.
X.n.0.eigenvalues <- round(eigen(X)$values[1:2], 12)
Y.n.0.eigenvalues <- round(eigen(Y)$values[1:2], 12)
A.n.0.s <- round((svd(A)$d)^2, 12)

X.n.0.eigenvalues == Y.n.0.eigenvalues
```

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

```
A.n.0.s == Y.n.0.eigenvalues
```

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

2. Problem Set 2

Using the procedure outlined in section 1 of the weekly handout, write a function to compute the inverse of a well-conditioned full-rank square matrix using co-factors. In order to compute the co-factors, you may use built-in commands to compute the determinant. Your function should have the following signature: $B = myinverse(A)$

where A is a matrix and B is its inverse and $A \times B = I$. The off-diagonal elements of I should be close to zero, if not zero. Likewise, the diagonal elements should be close to 1, if not 1. Small numerical precision errors are acceptable but the function myinverse should be correct and must use co-factors and determinant of A to compute the inverse.

Please submit PS1 and PS2 in an R-markdown document with your first initial and last name.

**The process that will use to calculate the inverse of the matrix A is through co-factors and determinants:

$$A^{-1} = \frac{C^T}{det(A)}$$

The code below creates a square matrix given the parameter m.size.

```
m.size <- 4
##Genearates a square matrix of size "m.size" filled with random numbers between 1 to 10
M <- matrix(sample.int(10, size = m.size^2, replace = T), nrow = m.size)
```

The code below

```
myinverse <- function(M) {
cofactor.list <- c()
##Finds the column and rown size of the matrix M
i.final <- nrow(M)
```

```
j.final <- ncol(M)

i <- 1
j <- 1

sub.matrix <- M[-i,-j]
sub.matrix
det(sub.matrix)
##Loops through the i and j indeces of the matrix M
for (i in 1:i.final){
  for(j in 1:j.final){
sub.matrix <- M[-i,-j]
sub.matrix.det <- det(sub.matrix)*(-1)**(i+j)
cofactor.list <- c(cofactor.list, sub.matrix.det)
}
}
##Creates a matrix of cofactors.
CF <- matrix(cofactor.list, ncol = j.final, byrow = TRUE)
##Calculates the Transpose of the cofactor Matrix
CF.Transpose <- c()
for (i in 1:(ncol(CF))) {
  CF.Transpose <- rbind(CF.Transpose, CF[,i] )
}
##Calculates the inverse of matrix M as the tranpose of the cofactor matrix divded by the determinant o
M.det <- det(M)
Inverse <- CF.Transpose/M.det
#Returns the inverse matrix
return(Inverse)
}
```

Comparing output of the myinverse() and inv() function from matlib:

```
library(matlib)
myinverse(M)
```

```
##             [,1]       [,2]        [,3]        [,4]
## [1,] -0.01327434  0.3141593 -0.07079646 -0.20796460
## [2,]  0.15486726 -0.3318584  0.15929204  0.09292035
## [3,] -0.03185841 -0.2460177  0.23008850  0.10088496
## [4,] -0.01061947  0.2513274 -0.25663717  0.03362832
```

```
inv(M)
```

```
##             [,1]       [,2]        [,3]        [,4]
## [1,] -0.01327434  0.3141593 -0.07079646 -0.20796460
## [2,]  0.15486726 -0.3318584  0.15929204  0.09292035
## [3,] -0.03185841 -0.2460177  0.23008850  0.10088496
## [4,] -0.01061947  0.2513274 -0.25663717  0.03362832
```