PShaji_Assignment4

Priya Shaji

9/21/2019

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.

Answer 1)

1) Create matrix A

```
A = matrix(c(1,-1,2,0,3,4),nrow = 2,ncol = 3)
A
```

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

2) Let us 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.

```
X = A%*% t(A)
X
```

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

```
Y = t(A)%*%A
Y
```

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

3) Eigen Vectors and Eigen Value of X

```
eigen_X = eigen(X)
print(list('EigenValues',eigen_X$values,'EigenVectors',eigen_X$vectors))
```

```
## [[1]]
## [1] "EigenValues"
##
## [[2]]
## [1] 26.601802  4.398198
##
## [[3]]
## [1] "EigenVectors"
##
## [[4]]
##           [,1]       [,2]
## [1,] 0.6576043 -0.7533635
## [2,] 0.7533635  0.6576043
```

4) Eigen Vectors and Eigen Value of Y

```
eigen_Y = eigen(Y)
print(list('EigenValues',eigen_Y$values,'EigenVectors',eigen_Y$vectors))
```

```
## [[1]]
## [1] "EigenValues"
##
## [[2]]
## [1] 2.660180e+01 4.398198e+00 1.058982e-16
##
## [[3]]
## [1] "EigenVectors"
##
## [[4]]
##             [,1]       [,2]       [,3]
## [1,] -0.01856629 -0.6727903  0.7396003
## [2,]  0.25499937 -0.7184510 -0.6471502
## [3,]  0.96676296  0.1765824  0.1849001
```

5) Compute the left-singular, singular values, and right-singular vectors of A using the svd command.

```
ValueDecomposition = svd(A)
```

```
print(c(list(ValueDecomposition$u,ValueDecomposition$v,ValueDecomposition$d)))
```

```
## [[1]]
##            [,1]       [,2]
## [1,] -0.6576043 -0.7533635
## [2,] -0.7533635  0.6576043
##
## [[2]]
##              [,1]       [,2]
## [1,]  0.01856629 -0.6727903
## [2,] -0.25499937 -0.7184510
## [3,] -0.96676296  0.1765824
##
## [[3]]
## [1] 5.157693 2.097188
```

Therefore, eigen vectors of X are the left-singular matrix(u) of A. eigen vectors of Y are the left-singular matrix(v) of A.

7) Square of non-singular values of A and the eigen values of X

```
list(ValueDecomposition$d*ValueDecomposition$d,eigen_X$values)
```

```
## [[1]]
## [1] 26.601802  4.398198
##
## [[2]]
## [1] 26.601802  4.398198
```

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×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.

Answer 2)

1) Define the function
2) Verify if the matrix is square matrix
3) Verify if the determinant of matrix A is zero
4) Create Co-factors of matrix A
5) Initiate a loop and iterate through the rows and columns of the matrix, calculating the cofactors
6) Invoke the function

```
# Define the function 'myInverse'
myInverse = function(A){

  if(dim(A)[1] != dim(A)[2]){ return('ERROR : Matrix is not square') }

  if(det(A) == 0){ return('ERROR : Matrix is singular') }

  CMatrix = A * 0
```

```r
  for (i in 1:ncol(A)) {
    for (j in 1:nrow(A)) {
      CMatrix[i,j] = det(A[-i,-j]) * (-1)^(i+j)
    }}
  inversed = t((CMatrix)/det(A))
  return(inversed)
}

# Invoke the function 'myInverse'

A = matrix(c(1,0,-1,2,4,1,0,0,2,4,1,4,3,4,0,1),nrow=4)

B = myInverse(A)
C = solve(A)

round(B,4)==round(C,4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,] TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE
```