SRout Assign 4

Subhalaxmi Rout

9/20/2020

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 X 2 matrix A

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

write code in R to compute $\mathbf{X = AAT}$ and $\mathbf{Y = ATA}$. 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. **Solution**

```
# given matrix
A <- matrix(c(1,2,3,-1,0,4), byrow = TRUE, ncol = 3)
# transpose of A
AT <- t(A)
# dot product of A and AT and assign it to X
X <- A %*% AT
# dot product of AT and A and assign it to Y
Y <- AT %*% A
# eigen values of X
eigen_x <- eigen(X)
eigen_values_x <- eigen_x$values
# round off the values till 3 decimal and put in a list
eigen_values_x <- lapply(list(eigen_values_x), round, 3)
eigen_values_x
```

```
## [[1]]
## [1] 26.602  4.398
```

```
# eigen values of Y
eigen_y <- eigen(Y)
eigen_values_y <- eigen_y$values
# round off the values till 3 decimal and put in a list
eigen_values_y <- lapply(list(eigen_values_y), round, 3)
eigen_values_y
```

```
## [[1]]
## [1] 26.602  4.398  0.000
```

```r
# eigen vectors of X
eigen_vectors_x <- eigen_x$vectors
eigen_vectors_x <- lapply(list(eigen_vectors_x), round, 3)
eigen_vectors_x
```

```
## [[1]]
##       [,1]   [,2]
## [1,] 0.658 -0.753
## [2,] 0.753  0.658
```

```r
# eigen vectors of Y
eigen_vectors_y <- eigen_y$vectors
eigen_vectors_y <- lapply(list(eigen_vectors_y), round, 3)
eigen_vectors_y
```

```
## [[1]]
##        [,1]   [,2]   [,3]
## [1,] -0.019 -0.673  0.740
## [2,]  0.255 -0.718 -0.647
## [3,]  0.967  0.177  0.185
```

Here, compute `Singular Value Decomposition(SVD).` SVD has 3 parts

- d is the vector that contain singular values of A.
- u is a matrix that contain left singular vectors of A
- v is a matrix that contain right singular vectors of A

Using `svd` function we will calculate.

```r
svd_A <- svd(A)
# round off the values till 3 decimal and put in a list
singular_d <- lapply(list(svd_A$d),round,3)
print("Singular values of A")
```

```
## [1] "Singular values of A"
```

```r
singular_d
```

```
## [[1]]
## [1] 5.158 2.097
```

```r
# round off the values till 3 decimal and put in a list
left_singular_u <- lapply(list(svd_A$u),round,3)
print("Left Singular values of A")
```

```
## [1] "Left Singular values of A"
```

```
left_singular_u
```

```
## [[1]]
##        [,1]   [,2]
## [1,] -0.658 -0.753
## [2,] -0.753  0.658
```

```
# round off the values till 3 decimal and put in a list
right_singular_v <- lapply(list(svd_A$v),round,3)
print("Right Singular values of A")
```

```
## [1] "Right Singular values of A"
```

```
right_singular_v
```

```
## [[1]]
##        [,1]   [,2]
## [1,]  0.019 -0.673
## [2,] -0.255 -0.718
## [3,] -0.967  0.177
```

Let's compare the the two sets of singular vectors and show that they are indeed eigenvectors of X and Y.

```
# Non-zero eigenvalues of X and Y = square of non-zero singular values of A?
Singular_square = round(svd_A$d * svd_A$d, digits = 3)
#Singular_square
Singular_square == round(eigen_x$values, digits = 3)
```

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

```
# Left-singular values vector u = eigen vector X?
round(svd_A$u, digits = 3) == round(eigen_x$vectors, digits = 3)
```

```
##       [,1] [,2]
## [1,] FALSE TRUE
## [2,] FALSE TRUE
```

```
round(svd_A$u, digits = 3)
```

```
##        [,1]   [,2]
## [1,] -0.658 -0.753
## [2,] -0.753  0.658
```

```
round(eigen_x$vectors, digits = 3)
```

```
##       [,1]   [,2]
## [1,] 0.658 -0.753
## [2,] 0.753  0.658
```

```r
# add '-' in 1st column
svd_A$u[ ,1] = - svd_A$u[ ,1]
# round of till 3 decimal and compare
round(svd_A$u, digits = 3) == round(eigen_x$vectors, digits = 3)
```

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

```r
round(svd_A$u, digits = 3)
```

```
##       [,1]   [,2]
## [1,] 0.658 -0.753
## [2,] 0.753  0.658
```

```r
round(eigen_x$vectors, digits = 3)
```

```
##       [,1]   [,2]
## [1,] 0.658 -0.753
## [2,] 0.753  0.658
```

```r
# Right-singular values vector v = eigen vector Y?
round(svd_A$v, digits = 3) == round(eigen_y$vectors[,1:2], digits = 3)
```

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

```r
round(svd_A$v, digits = 3)
```

```
##        [,1]   [,2]
## [1,]  0.019 -0.673
## [2,] -0.255 -0.718
## [3,] -0.967  0.177
```

```r
round(eigen_y$vectors[,1:2], digits = 3)
```

```
##        [,1]   [,2]
## [1,] -0.019 -0.673
## [2,]  0.255 -0.718
## [3,]  0.967  0.177
```

```r
# add '-' in 1st column
svd_A$v[,1] = - svd_A$v[,1]
# round of till 3 decimal and compare
round(svd_A$v, digits = 3) == round(eigen_y$vectors[,1:2], digits = 3)
```

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

```
round(svd_A$v, digits = 3)
```

```
##        [,1]   [,2]
## [1,] -0.019 -0.673
## [2,]  0.255 -0.718
## [3,]  0.967  0.177
```

```
round(eigen_y$vectors[,1:2], digits = 3)
```

```
##        [,1]   [,2]
## [1,] -0.019 -0.673
## [2,]  0.255 -0.718
## [3,]  0.967  0.177
```

We can see from the above comparision that the singular vectors obtained using the SVD command are the same as the eigenvectors of X and Y. The only difference is whether they are positive or negative, which is due to them being declared in unit vector form. The ratios are otherwise same.

Hence, we can conclude that computed non-zero eigenvectors of X and Y are the same as the non-zero SVD values of matrix A and square of non-zero singular values of A are same as the Non-zero eigenvalues of X and Y.

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 X 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.

Solution

```
inverse = function(A){
  # check if square
  if(nrow(A) != ncol(A))
    {
    return('ERROR : Matrix is not a square')
    }
  # check if determinant is 0
  if(round(det(A),digits = 3) == 0)
    {
    return('ERROR : Matrix is not invertible')
    }
  # create co-factor matrix
  n = ncol(A)
  co_factor = diag(n)
  #Cofactoring proceedure
```

```
  for (i in 1:ncol(A)) {
    for (j in 1:nrow(A)) {
      co_factor[i,j] = det(A[-i,-j]) * (-1)^(i+j)
    }
    }
  mat_inverse = t((co_factor)/det(A))
  return(mat_inverse)
}
# Testing - scenario 1 (not a square matrix)
A1 = matrix(c(1,2,4,6,9,5),byrow = T,nrow=2)

B = inverse(A1)
B
```

```
## [1] "ERROR : Matrix is not a square"
```

```
# Testing - scenario 2 (determinante is zero)
A2 = matrix(c(1,2,3,4,5,6,7,8,9),byrow = T,nrow=3)

D = inverse(A2)
D
```

```
## [1] "ERROR : Matrix is not invertible"
```

```
# Testing - scenario 3
A3 = matrix(c(3,2,5,2,3,2,5,2,4),byrow = T,nrow=3)
E = inverse(A3)
E
```

```
##             [,1]        [,2]       [,3]
## [1,] -0.29629630 -0.07407407  0.4074074
## [2,] -0.07407407  0.48148148 -0.1481481
## [3,]  0.40740741 -0.14814815 -0.1851852
```

```
S = solve(A3)
S
```

```
##             [,1]        [,2]       [,3]
## [1,] -0.29629630 -0.07407407  0.4074074
## [2,] -0.07407407  0.48148148 -0.1481481
## [3,]  0.40740741 -0.14814815 -0.1851852
```

```
# compare
round(E,4) == round(S,4)
```

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