DATA 605 - Assignment 4

Joshua Sturm

02/24/2018

```
library(pracma)
```

Problem Set 1

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: $\begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 4 \end{bmatrix}$ 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 commands 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 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.

```
# Input matrix A
A <- matrix(c(1,2,3,
              -1,0,4),
            nrow=2, ncol=3, byrow=T)

# Calculate X and Y using built-in commands
X <- A %*% t(A)
Y <- t(A) %*% A

# Calculate eigenvalues and eigenvectors for X and Y
X.eigenvalues <- eigen(X)$values
X.eigenvectors <- eigen(X)$vectors

Y.eigenvalues <- eigen(Y)$values
Y.eigenvectors <- eigen(Y)$vectors

# Compute singular value decomposition of A
A.leftsingular <- svd(A)$u
A.singular <- svd(A)$d
A.rightsingular <- svd(A)$v
```

```
# Compare eigenvectors of X and Y with svd of A
X.eigenvectors
##           [,1]       [,2]
## [1,] 0.6576043 -0.7533635
## [2,] 0.7533635  0.6576043
A.leftsingular
##           [,1]       [,2]
```

```
## [1,] -0.6576043 -0.7533635
## [2,] -0.7533635  0.6576043
is.element(A.leftsingular, X.eigenvectors)
## [1] FALSE  TRUE  TRUE  TRUE

Y.eigenvectors
##             [,1]       [,2]       [,3]
## [1,] -0.01856629 -0.6727903  0.7396003
## [2,]  0.25499937 -0.7184510 -0.6471502
## [3,]  0.96676296  0.1765824  0.1849001
A.rightsingular
##             [,1]       [,2]
## [1,]  0.01856629 -0.6727903
## [2,] -0.25499937 -0.7184510
## [3,] -0.96676296  0.1765824
```

From the output, we can see that the singular values of A are indeed eigenvectors of X and Y, albeit multiplied by a scalar of -1.

```
# Compare non-zero eigenvalues of X and Y with non-zero singular values of A
X.eigenvalues
## [1] 26.601802  4.398198
Y.eigenvalues
## [1] 2.660180e+01 4.398198e+00 1.058982e-16
A.singular^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 $A \times B = I$. 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.

```
myinverse <- function(A){
  # Check if the matrix A is square
  if (nrow(A) != ncol(A)){
    stop("Matrix A is not square.")
  }

  dtrmnt <- det(A)
  # Check if the matrix is of full rank
  if (dtrmnt == 0){
    stop("Matrix is not full-rank")
  }

  cofactor.matrix <- matrix(0, nrow = nrow(A), ncol = ncol(A))
```

```
  for (row in 1:nrow(A)){
    for (col in 1:ncol(A)){
      matrix.minor <- A[-row, -col] # Set up matrix of minors
      matrix.minor.det <- det(matrix.minor) # Calculate its determinant
      cofactor.matrix[row, col] <- ((-1)^(row + col)) * matrix.minor.det # Proper sign
    }
  }
  B <- (1/dtrmnt) * t(cofactor.matrix)
  return(B)
}
```

Test our function:

```
# 3x3 random matrix
random.mat.three <- matrix(rnorm(9),3,3)
#5x5
random.mat.five <- matrix(rnorm(100),5,5)

# Check if myinverse produces same result as the solve function
all.equal(solve(random.mat.three), myinverse(random.mat.three))
## [1] TRUE
all.equal(solve(random.mat.five), myinverse(random.mat.five))
## [1] TRUE
```