DATA 605 Week 4 Homework

Ilya Kats

September 23, 2017

Problem Set 1

The following block includes all relevant R code. Results are presented below.

```
# Initial matrix A
A <- matrix(c(1,-1,2,0,3,4), nrow=2)

# Calculate X and Y
X <- A %*% t(A)
Y <- t(A) %*% A

# Calculate eigenvalues and eigenvectors for X and Y
library(pracma)
eigenX <- eigen(X)
eigenY <- eigen(Y)

# Calculate SVD
svdA <- svd(A)
```

Matrix A

```
A
```

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

Matrix $X = AA^T$

```
X
```

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

Matrix $Y = A^TA$

```
Y
```

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

Eigenvalues of X and Y: We can see that not counting the zero value, the eigenvalues are the same. The output is rounded to 10 decimal places for display and to account for possible minor floating point limitations.

```
round(eigenX$values, digits=10)
```

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

```
round(eigenY$values, digits=10)
```

```
## [1] 26.601802  4.398198  0.000000
```

Eigenvectors of X and Y

```
eigenX$vectors
```

```
##           [,1]       [,2]
## [1,] 0.6576043 -0.7533635
## [2,] 0.7533635  0.6576043
```

```
eigenY$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
```

Singular Value Decomposition Results

```
svdA
```

```
## $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
```

Comparison: Eigenvalues and Singular Values of A: We can confirm that eigenvalues (σ) are equal to square of singular values derived from SVD (σ^2). Again to account for minor possible floating point fluctuations, the output is rounded to 10 decimal placed.

```
round(eigenX$values, digits=10) == round(svdA$d^2, digits=10)
```

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

Comparison: Eigenvectors and Singular Vectors of A

First, let us compare eigenvectors of X and singular vectors U.

```
eigenX$vectors
```

```
##           [,1]       [,2]
## [1,] 0.6576043 -0.7533635
## [2,] 0.7533635  0.6576043
```

```
svdA$u
```

```
##            [,1]       [,2]
## [1,] -0.6576043 -0.7533635
## [2,] -0.7533635  0.6576043
```

Immediately we see that first vectors only differ by their sign. This is similar to the problem encountered by Prof. Gilbert Strang in his Linear Algebra / SVD lecture (Lecture 29, MIT 18.06, https://www.youtube.com/watch?v=Nx0lRBaXoz4). Prof. Strang explains that it is possible to invert the sign of an eigenvector and resulting vector will also be an eigenvector. Let us invert the sign of the first eigenvector (multiply it by -1).

```
eigenX$vectors[,1] <- eigenX$vectors[,1] * (-1)
round(eigenX$vectors, digits=10) == round(svdA$u, digits=10)
```

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

Similarly we need to invert the sign of the first eigenvector of Y before we compare it to v_1. We only compare the first two eigenvectors since the eigenvector corresponding to eigenvalue of 0 is not used in the Singular Value Decomposition.

```
eigenY$vectors[,1] <- eigenY$vectors[,1] * (-1)
round(eigenY$vectors[,1:2], digits=10) == round(svdA$v, digits=10)
```

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

Problem Set 2

A function to compute the inverse of a well-conditioned full-rank square matrix using co-factors.

```
myinverse <- function(A) {
  # Check that A is square
  n <- dim(A)[1]
  if (n!=dim(A)[2]) {
    return(NA)
  }

  # Check that A is inversible
  # Determinant should not be zero
  if (det(A)==0) {
    return(NA)
  }

  # Loop through all elements and compute co-factor matrix C
  C <- matrix(0, n, n)
  for(i in 1:n) {
    for(j in 1:n) {
      # The submatrix A[-i,-j] is forced into matrix type in case of 2x2 matrix
      # Otherwise, if 2x2 matrix, then A[-i,-j] would produce a single element
      C[i,j] <- (-1)^(i+j)*det(matrix(A[-i,-j], n-1))
    }
  }

  # Inverse of A is equal to transpose of C divided by determinant of A
  B <- t(C)/det(A)

  return(B)
}
```

The function works by computing the co-factor matrix using the following formula: $C_{ij} = (-1)^{i+j} det(M_{ij})$. The inverse matrix is then computed as $A^{-1} = \frac{C^T}{det(A)}$.

Demonstration

Example 1

Let $A = \begin{bmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$.

```
A <- matrix(c(1,2,3,6,5,6,7,8,9), nrow=3)

# Calculate inverse using 'myinverse' function
B <- myinverse(A)
B
```

```
##       [,1] [,2]       [,3]
## [1,] -0.25   -1  1.0833333
## [2,]  0.50   -1  0.5000000
## [3,] -0.25    1 -0.5833333
```

```
# Double-check using 'solve' function
solve(A)
```

```
##       [,1] [,2]       [,3]
## [1,] -0.25   -1  1.0833333
## [2,]  0.50   -1  0.5000000
## [3,] -0.25    1 -0.5833333
```

```
round(solve(A), digits=10) == round(B, digits=10)
```

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

```
# Check that AB=I
round(A %*% B, digits=10)
```

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

Example 2

Let $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

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

# Calculate inverse using 'myinverse' function
B <- myinverse(A)
B
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

```
# Double-check using 'solve' function
solve(A)
```

```
##      [,1] [,2]
## [1,]   -2  1.5
## [2,]    1 -0.5
```

```
round(solve(A), digits=10) == round(B, digits=10)
```

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

```
# Check that AB=I
round(A %*% B, digits=10)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

Notes

Throughout this homework rounding numbers to some decimal places (I have used 10) was necessary because of small precision discrepancies.