CFerrari_Assignment4

Charley Ferrari

Wednesday, February 18, 2015

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

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

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

write code in R to compute $X = AA^T$

```
X <- A %*% t(A)
```

and $Y = A^T A$

```
Y <- t(A) %*% A
```

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.

$$A = U\Sigma V^T$$
$$A^T A = V\Sigma^T \Sigma V^T = Y$$
$$AA^T = U\Sigma\Sigma^T U^T = X$$

U

```
usvd <- svd(A)$u

ueig <- eigen(X)$vectors
```

V

```
vsvd <- cbind(svd(A)$v,c(0,0,0))

veig <- cbind(eigen(Y)$vectors[,1:2],c(0,0,0))
```

Σ

```
ssvd <- matrix(c(svd(A)$d[1],0,0,svd(A)$d[2],0,0),nrow=2)

ssueig <- matrix(c(sqrt(eigen(X)$values[1]),0,0,sqrt(eigen(X)$values[2]),0,0),nrow=2)

ssveig <- matrix(c(sqrt(eigen(Y)$values[1]),0,0,sqrt(eigen(Y)$values[2]),0,0),nrow=2)
```

$$Full Decomposition$$

```
usvd %*% ssvd %*% t(vsvd)
```

```
##      [,1]         [,2] [,3]
## [1,]    1 2.000000e+00    3
## [2,]   -1 1.110223e-16    4
```

```
ueig %*% ssueig %*% t(veig)
```

```
##      [,1]          [,2] [,3]
## [1,]    1  2.000000e+00    3
## [2,]   -1 -9.992007e-16    4
```

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:

$B = \text{myinverse}(A)$

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.

First, I'm going to create a function called cofactorize.

This will take in a matrix M, indices i and j, and return the cofactor as a numeric

```
cofactorize <- function(M, i, j){

  return(((-1)^(i+j)) * det(M[-i,-j]))

}
```

Next, I'll use a for loop to build my cofactor matrix: C

I really wanted to avoid using a for loop for this... but I just could not find a way to get my functions to iterate their way through vectors rather than accept the vectors as argumnts.

For example: Ideally I wanted to just use cofactorize like this:

cofactor(M, 1:nrow(M), 1:ncol(M))

but, when I do this, my function would interpret det(M[-i,-j]) as an empty matrix, rather than iterating through i and j to give me what I wanted. I'll see what the forums have to say on this... I spent quite a bit of time trying to get some sort of apply or ply function to work to no avail.

```
# Example M

M <- matrix(c(1,2,3,4,9,6,7,8,9),nrow=3)

C <- matrix(nrow=nrow(M), ncol=ncol(M))

for(i in 1:nrow(M)){
  for(j in 1:ncol(M)){
    C[i,j] <- cofactorize(M, i, j)
  }
}
```

After this, I can just divide the result by the determinant of M to get my inverse:

```
MInv <- t(C)/det(M)
```

Now, I can put this all together in a function:

```
myinverse <- function(M){

  C <- matrix(nrow=nrow(M), ncol=ncol(M))

  for(i in 1:nrow(M)){
    for(j in 1:ncol(M)){
      C[i,j] <- cofactorize(M, i, j)
    }
  }

  return(t(C)/det(M))

}
```