Data 605 - Week2_ HW

Vishal Arora

September 7, 2019

1. Problem set 1

(a) Show that AT A != A AT in general. (Proof and demonstration.)

```
#Let's create a basic 3 x 3 matrix, which we'll call mat1.

mat1 <- matrix(c(1,3,0,4,1,5,2,0,7), nrow = 3, byrow = TRUE)
mat1
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    0
## [2,]    4    1    5
## [3,]    2    0    7
```

```
# using R get the transpose for mat1 matrices.

mat1T <- t(mat1)
mat1T
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    2
## [2,]    3    1    0
## [3,]    0    5    7
```

```
#calculate both  ATA and AAT.
ATA <- mat1T %*% mat1
ATA
```

```
##      [,1] [,2] [,3]
## [1,]   21    7   34
## [2,]    7   10    5
## [3,]   34    5   74
```

```
AAT <- mat1 %*% mat1T
AAT
```

```
##      [,1] [,2] [,3]
## [1,]   10    7    2
## [2,]    7   42   43
## [3,]    2   43   53
```

```
#It's clear that ATA???AAT for this "general" matrix.
# check programmaticaly AAT == ATA

AAT == ATA
```

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

(b) For a special type of square matrix A, we get AT A = AAT . Under what conditions could this be true? (Hint: The Identity matrix I is an example of such a matrix).

```
#Identity Matrix is a square matrix in which all the elements of the principal diagonal are ones and al
#Creating Identity matrix using R

A <- diag(3)
A
```

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

```
#transpose of A
AT <- t(A)
AT
```

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

```
# checking AT A = A AT

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

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

Problem Set 2

Matrix factorization is a very important problem. There are supercomputers built just to do matrix factorizations. Every second you are on an airplane, matrices are being factorized. Radars that track flights use a technique called Kalman filtering. At the heart of Kalman Filtering is a Matrix Factorization operation. Kalman Filters are solving linear systems of equations when they track your ight using radars.

Write an R function to factorize a square matrix A into LU or LDU, whichever you prefer.

```
library(matrixcalc)
```

```
## Warning: package 'matrixcalc' was built under R version 3.5.2
```

```
matrixFactorization_LU <- function(mat) {
  # Checking whether matrix is a square matrix
 if(is.square.matrix(mat) == FALSE) {
    return("Only square matrix are allowed!")
  }
```

```r
  U <- mat
  n <- dim(mat)[1]
  L <- diag(n)


  if (n==1) {
    return(list(L,U))
  }


  for(i in 2:n) {
    for(j in 1:(i-1)) {
      multiplier <- -U[i,j] / U[j,j]
      U[i, ] <- multiplier * U[j, ] + U[i, ]
      L[i,j] <- -multiplier
    }
  }
  return(list(L,U))
}


#using the matrixFactorization_LU
A <- matrix(c(2,0,5,6,1,4,7,3,0), nrow=3, byrow=TRUE)
LU <- matrixFactorization_LU(A)
L<-LU[[1]]
U<-LU[[2]]

A
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    5
## [2,]    6    1    4
## [3,]    7    3    0
```

```r
L
```

```
##      [,1] [,2] [,3]
## [1,]  1.0    0    0
## [2,]  3.0    1    0
## [3,]  3.5    3    1
```

```r
U
```

```
##      [,1] [,2]  [,3]
## [1,]    2    0   5.0
## [2,]    0    1 -11.0
## [3,]    0    0  15.5
```

```r
#A == L %*% U
A == L %*% U
```

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