DATA 605

Week 2: Assignment

Vinayak Patel

Sept 8, 2019

Problem Set 1

1) Show that $A^T A \neq AA^T$ in general. (Proof and demonstration.)

```
A = matrix(seq(1, 9), nrow=3, byrow = T)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
AT = matrix(seq(1, 9), nrow=3, byrow = F)
AT
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
A %*% AT
```

```
##      [,1] [,2] [,3]
## [1,]   14   32   50
## [2,]   32   77  122
## [3,]   50  122  194
```

```
AT %*% A
```

```
##      [,1] [,2] [,3]
## [1,]   66   78   90
## [2,]   78   93  108
## [3,]   90  108  126
```

```
(AT %*% A) == (A %*% AT)
```

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

Hence, $A^T A \neq AA^T$

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

$A^T A = AA^T$ is true if A is a diagonal matrix.So An identity matrix when transposed and multiplied to itself, are equal to each other.

Let $A = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$ then $A^T = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$ therefore:

$$A^TA = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

$$AA^T = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

This gives a case when $A^TA = AA^T$ is true.

```
A = matrix(c(1,1,0,-1,1,3,2,4,0,2,4,8,-1,4,8,6), nrow=2, byrow = T)
AT = t(A)
(A %*% AT) == (A %*% AT)
```

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

$A^TA = AA^T$, we can see that all of them are symmetric matrices. A symmetric matrix is a matrix where $A_{ij} = A_{ji}$ for every i and j.

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 ights 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. Please submit your response in an R Markdown document using our class naming convention, E.g. LFulton_Assignment2_PS2.png You don't have to worry about permuting rows of A and you can assume that A is less than 5x5, if you need to hard-code any variables in your code. If you doing the entire assignment in R, then please submit only one markdown document for both the problems.

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

```
#matrix factorization


LU_factorization <- function(A) {
  # Check wheter matrix is square or not
  if (dim(A)[1]!=dim(A)[2]) {
    return(NA)
  }

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

```r
  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))
}
```

```r
# test matrix factorization
### Sample 1: A 2x2
A<- matrix(c(2,6,1,8), nrow = 2, ncol = 2)
LU <-LU_factorization(A)
L<-LU[[1]]
U<-LU[[2]]
A
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    6    8
```

```r
L
```

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

```r
U
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    0    5
```

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

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

```r
### Sample 2: B 3x3
B <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3)
LU <- LU_factorization(B)
L<-LU[[1]]
U<-LU[[2]]
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
L
```

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

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

```r
U
```

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

```r
B==L%*%U
```

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

```r
### Sample 3: D 3x3 with lots of negative inputs

C <- matrix(c(-3,5,-1,4,-2,-6,7,1,8), nrow = 3, ncol = 3)
LU <-LU_factorization(C)
L<-LU[[1]]
U<-LU[[2]]
C
```

```
##      [,1] [,2] [,3]
## [1,]   -3    4    7
## [2,]    5   -2    1
## [3,]   -1   -6    8
```

```r
L
```

```
##            [,1]      [,2] [,3]
## [1,]  1.0000000  0.000000    0
## [2,] -1.6666667  1.000000    0
## [3,]  0.3333333 -1.571429    1
```

```r
U
```

```
##      [,1]     [,2]     [,3]
## [1,]   -3 4.000000  7.00000
## [2,]    0 4.666667 12.66667
## [3,]    0 0.000000 25.57143
```

```r
C==L%*%U
```

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

Hence, it verifies the result