DATA605-Week2 - Trace, Determinants,and Inverse

Christian Thieme

9/3/2020

Problem Set 1

Question 1:

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

Proof:

The rules of linear algebra state that multiplication of matrices is NOT commutative. The full proof comes from the fact that Matrices are members of the non-commutative ring theory with respect to multiplication - which is beyond the scope of this course. However, at its core, for all matrices besides two by twos, when you transpose them, it changes the number of rows and columns they have, and depending on which matrix you choose to have first in your multiplication, it will determine how many rows and columns your resulting matrix has. Take for example matrix A which is a 2x3 matrix. When we take it's transpose, it will be a 3x2 matrix, A^T. Because the number of columns from matrix A is the same as the number of rows in A^T, this multiplication is allowed. We can also tell from these values that the resulting matrix will be a 2x2 matrix since matrix A has two rows and A^T has 2 columns. If we were to reverse this and multiply A^T by A we would multiply a 3x2 by a 2x3. Looking at their row and column structure, we can tell that the resulting matrix would be a 3x3 matrix NOT a 2x2. Now, this proof is obvious when you aren't working with a square matrix, but does this non-commutative theory hold up with a square matrix? Yes, it does. Matrix multiplication is non-commutative except in extremely rare situations which will be described in the next question. Below is an example demonstrating multiplication of a 2x2 matrix with it's inverse, and showing that $A^TA \neq AA^T$:

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

$$A^T = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}$$

$$A^TA = \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{matrix} (1)(1)+(1)(1) & (1)(2)+(1)(3) \\ (2)(1)+(3)(1) & (2)(2)+(3)(3) \end{matrix} = \begin{bmatrix} 1 & 5 \\ 5 & 13 \end{bmatrix}$$

A^T A A^TA

$$AA^T = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} = \begin{matrix} (1)(1)+(2)(2) & (1)(1)+(3)(2) \\ (1)(1)+(3)(2) & (1)(1)+(3)(3) \end{matrix} = \begin{bmatrix} 5 & 7 \\ 7 & 10 \end{bmatrix}$$

A A^T AA^T

$$\begin{bmatrix} 1 & 5 \\ 5 & 13 \end{bmatrix} \neq \begin{bmatrix} 5 & 7 \\ 7 & 10 \end{bmatrix}$$

A^TA AA^T

Question 2:

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

Two matrices can be commutative under multiplication if they are simultaneously diagonalizable, such as the identity matrix.

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 flight 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.

The below function `lu_decomposition_function` will decompose a square matrix into is lower and upper matrices. The below function is completely dynamic, with no hard coding, and will work for any square matrix. If a non-sqaure matrix is entered, it will throw and error.

```
# LU Decomposition function

lu_decomposition_function <- function(A) {

if (dim(A)[1] == dim(A)[2]) {

    B <- A

    #row <- 1
    num_rows <- dim(A)[1]
    num_cols <- dim(A)[2]

    starter_row <- 2
    start_row <- 2
    start_col <- 1

    L <- diag(dim(A)[1])

    while (starter_row <= num_rows && start_col < num_cols) {


      if (start_row <= num_rows) {

        if (A[start_row, start_col] == 0) {

            start_row <- start_row + 1

        } else {

          num_under_diag <- A[start_row, start_col]
          answer <- num_under_diag / A[starter_row-1, start_col]

          next_row <- A[start_row,] + (-answer * A[starter_row-1,])

            A[start_row,] <- next_row

            L[start_row, start_col] <- answer


            start_row <- start_row + 1

        }
        } else {

          start_col <- start_col + 1
          starter_row <- starter_row + 1
          start_row <- starter_row
        }


    }
    print("------------------------")
    print("The original matrix is: ")
    print(B)
    print("------------------------")
    print("The upper decomposed matrix is: ")
    print(A)
    print("------------------------")
    print("The lower decomposed matrix is: ")
    print(L)
    print("------------------------")

} else {

  print("Not a square matrix, please enter a square matrix where
        the number of rows and columns match. ")
}

}
```

Let's run this function on a 3x3 matrix:

```
A <- matrix(c(2,1,-6,4,-4,-9,-4,3,5), nrow = 3)

lu_decomposition_function(A)
```

```
## [1] "------------------------"
## [1] "The original matrix is: "
##      [,1] [,2] [,3]
## [1,]    2    4   -4
## [2,]    1   -4    3
## [3,]   -6   -9    5
## [1] "------------------------"
## [1] "The upper decomposed matrix is: "
##      [,1] [,2] [,3]
## [1,]    2    4 -4.0
## [2,]    0   -6  5.0
## [3,]    0    0 -4.5
## [1] "------------------------"
## [1] "The lower decomposed matrix is: "
##      [,1] [,2] [,3]
## [1,]  1.0  0.0    0
## [2,]  0.5  1.0    0
## [3,] -3.0 -0.5    1
## [1] "------------------------"
```

Now, let's run this function on a 4x4 matrix:

```
A <- matrix(c(1,2,0,0,5,12,4,0,0,5,13,6,0,0,5,11), nrow = 4)

lu_decomposition_function(A)
```

```
## [1] "------------------------"
## [1] "The original matrix is: "
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    0    0
## [2,]    2   12    5    0
## [3,]    0    4   13    5
## [4,]    0    0    6   11
## [1] "------------------------"
## [1] "The upper decomposed matrix is: "
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    0    0
## [2,]    0    2    5    0
## [3,]    0    0    3    5
## [4,]    0    0    0    1
## [1] "------------------------"
## [1] "The lower decomposed matrix is: "
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    2    1    0    0
## [3,]    0    2    1    0
## [4,]    0    0    2    1
## [1] "------------------------"
```

As you can see the function is able to decompose any square matrix into its upper and lower matrices.