E.Roman_Assignment_2_PS1_PS2.rmd

Enid Roman

2023-02-05

1. Problem set 1:

(1) Show that ATA does not equal to AAT in general. (Proof and demonstration.)

As per the text, A First Course of Linear Algebra, page 180, even if the sizes are right, matrix multiplication is not commutative — order matters.

Please see proof in word document.

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

```
# Matrix A
A
```

Below I will demonstrate from the result that AT does not equal AAT.

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

```
# Matrix A Transpose At
At
```

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

```
# Multiplying A by its transpose At
A %*% At
```

```
##      [,1] [,2] [,3]
## [1,]    6    3    3
## [2,]    3   17   14
## [3,]    3   14   14
```

```
# Multiplying transposed At by A
At %*% A
```

```
##      [,1] [,2] [,3]
## [1,]   26    5   -9
## [2,]    5    5    0
## [3,]   -9    0    6
```

```
# Check using logical comparison
A %*% At == At %*% A
```

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

(2) 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).

The matrix is symmetrical and thus the transpose equals that matrix itself. Thus, their product will be equal because both cases (AAT and ATA) will result in the matrix simply being squared.

```
A <- matrix(c(1,3,0,3,1,0,0,0,1), nrow=3, byrow=TRUE)
```

```
# Matrix A
A
```

Below I will demonstrate from the result that the condition will be true if the matrices are symstrical or identity matrices.

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

```
# Matix A transpose is the same matrix
t(A)
```

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

```
# Check using logical comparison
A == t(A)
```

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

```
# Check using logical comparison
(A %*% t(A)) == (t(A) %*% A)
```

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

2. 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.

Below chose to perform LU decomposition and find two matrices (L and U) who are lower and upper triangular, respectively, and whose product (LU) equals A.

```
factorize_LU <- function(A) {
  # Check wheter matrix A is square
  if (dim(A)[1]!=dim(A)[2]) {
    return(NA)
  }

  U <- A
  n <- dim(A)[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))
}
```

```
# Test the factorization function

A <- matrix(c(2,1,1,1,1,3,7,3,4), nrow=3, byrow=TRUE)
LU <- factorize_LU(A)
L<-LU[[1]]
U<-LU[[2]]
```

```
A
```

Below you can see our matrix factorization via LU decomposition of A, a 3x3 matrix was successful.

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

```
L
```

```
##      [,1] [,2] [,3]
## [1,]  1.0    0    0
## [2,]  0.5    1    0
## [3,]  3.5   -1    1
```

```
U
```

```
##      [,1] [,2] [,3]
## [1,]    2  1.0  1.0
## [2,]    0  0.5  2.5
## [3,]    0  0.0  3.0
```

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

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