DATA605 Homework2

Jonathan Hernandez

September 9, 2018

Problem set 1. (1): Show that $A^T A \neq AA^T$

Proof: Let A be a m x n matrix. Then A^T is a n x m matrix

Let A[i,j] be the entry in the ith row and jth column in A and

A^T[i,j] be the entry in the jth row and ith column that is A[j, i]

by matrix multiplication definition,

$AA^T = C$ where c[i,j] is a m x m matrix defined by

$c_{i,j} = \Sigma_{k=1}^{m} a_{i,k} * a_{j,k}$

and $A^T A = D$ where d[i,j] is a n x n matrix defined by

$d_{i,j} = \Sigma_{k=1}^{n} a_{k,i} * a_{k,j}$

These summations are not the same for every entry and this shows that

$A^T A \neq AA^T$

Problem set 1. (2): For a special type of square matrix A, we get $A^T A = AA^T$

Under what conditions could this be true?

Answer: if a square matrix A is symmetric that is $A^T = A$ then the condition holds true

as $A * A = A * A$ and $A^T A^T = A^T A^T$

The identity matrix is such as a matrix as I^T = I no matter what.

Problem set 2.

Write a function to factorize a square matrix A into LU or LDU.

```
# matrix factorization A a square matrix
LU_factorization <- function(A){
  # check if square matrix or not
  if (nrow(A) != ncol(A)){
    return(-1)
  }
  else{
    n_rows <- nrow(A)
    # L and U matricies; pre-populate them
    L <- matrix(0, nrow = n_rows, ncol = n_rows)
    U <- matrix(0, nrow = n_rows, ncol = n_rows)
    # using doolittle algorithm for finding L and U without gaussian elimination
    for(i in 1:n_rows){
```

```r
      # upper triangular matrix
      for(k in i:n_rows){
        sum <- 0
        for (j in 1:i){
          # compute sum in U[i,k] = A[i,k] - (L[i,j]*U[i,j] from j = 1 to i)
          sum <- sum + (L[i, j] * U[j, k])
        }
        # computing U[i,k] = A[i,k] - sum
        U[i, k] <- A[i, k] - sum
      }
      # Computer Lower triangular matrix
      for (k in i:n_rows){
        # compute sum of L[k,j] * U[j,i]
        sum <- 0
        for(j in 1:i){
          sum <- sum + (L[k, j] * U[j, i])
        }
        # compute L[k,i]
        L[k, i] <- (A[k, i] - sum) / U[i, i]
      }
    }
  }
  return(list(L=L, U=U))
}
# test matrix factorization
A <- matrix(c(1, 2, 4, 1, 3, 6, 1, 5, 8), nrow = 3, ncol = 3)
LU_factorization(A)
```

```
## $L
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    2    1    0
## [3,]    4    2    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    0    1    3
## [3,]    0    0   -2
```