LU Factorization Algorithm

ShenHengheng

October 27, 2017

Brief Introduction

The algorithm is written by R language, and implement lu algorithms.

Source Code

Structure

- `findpiv()` function
- `switch2rows()` function
- `plu()` function

Code

- `findpiv <- function(A, k, p, tol)`
  - Description: `findpiv` Used by plu to find a pivot for Gaussian elimination.
  - Arguments:
    - `A` full Matrix, and such that k and p
    - `k,p` specified submatrix begin ar k-th and p-th
    - `tol` machine number
  - Returns `list(r,p)` represent row and column of the first element
  - Usage `[r, p] = findpiv(A(k:m, p:n), k, p, tol)` finds the first element in the specified submatrix which is larger than tol in absolute value. It returns a list of indices r and p so that A(r, p) is the pivot.

this is source of findpiv function

```
findpiv <- function(A, k, p, tol) {
  shape <- dim(A) # shape of A matrix, and return a vector, and contains two elements
  m <- shape[1]
  n <- shape[2]
  r <- which(abs(A) > tol)
  if (length(r) == 0)
    return(list(k=k, p=p))

  #
  r <- r[1]
  j <- as.integer(as.double(r-1)/m )+ 1
  p <- p + j - 1

  k <- k + r - (j - 1) * m - 1
  return(list(k=k, p=p))
}
```

- `switch2rows <- function(A, m, n, i, j)`
  - Description: switch two rows and spcified from `i` to `j` column.
  - Arguments:
    - `A` full Matrix,
    - `m,n` rows that changed
    - `i,j` columns specified
  - Returns new `A` changed
  - Usage `switch2rows <- function(A, m, n, i, j)` change two rows in the specified submatrix It returns new matrix `A`.

```
switch2rows <- function(A, m, n, i, j) {
  B <- A
  B[m, i:j] <- A[n, i:j]
  B[n, i:j] <- A[m, i:j]
  return(B)
}
```

- `switch2rows <- function(A, m, n, i, j)`
  - Description: plu Rectangular PA=LU factorization with row exchanges.
  - Arguments:
    - `A` a rectangular Matrix,
  - Returns `list(P, L, U)`
    - P a permutation matrix
    - L a lower trinagular matrix
    - U an upper triangular matrix, size same as raw matrix

```
plu <- function(A) {
  shape <- dim(A)
  m <- shape[1]
  n <- shape[2]
  P <- as.matrix(diag(m))
  L <- as.matrix(diag(m))
  U <- matrix(0, m, n)
  pivcol <- c()
  tol <- sqrt(.Machine$double.eps)
  sign <- 1
  p <- 1
  for (k in 1:min(m, n)) {
    xy = findpiv(A[k:m, p:n], k, p, tol)
    r <- xy[[1]]
    p <- xy[[2]]
    if(length(r)==0)return(list(P=P,L=L,U=U))
    if (r!= k) {
      A <- switch2rows(A, k, r, 1, n)
      print(dim(A))
      if (k > 1)
        L <- switch2rows(L, k, r, 1, k-1)

      P <- switch2rows(P, k, r, 1, m)
      sign <- -sign
    }
    if (abs(A[k, p]) >= tol) {
      pivcol[length(pivcol) + 1] <- p
      for (i in (k+1):m) {
```

```
        L[i, k] <- A[i, p] / A[k, p]
        for (j in (k +1):n) {
          A[i, j] <- A[i, j] - L[i, k]*A[k, j]
        }
      }
    }
    for (j in k:n) {
      U[k, j] <- A[k, j] * (abs(A[k, j]) >= tol)
    }
    if (p < n)
      p <- p + 1

  }
  #
  return(list(P=P,L=L,U=U))
}
```

example

Quickly generate matrix 3x3, and column first, e.g.

```
A <- matrix(data = 1:9, 3, 3)
A
```

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

And apply `plu` function and return `P,L,U`

```
findpiv(A, 1, 1, .Machine$double.eps)
```

```
## $k
## [1] 1
##
## $p
## [1] 1
```

```
plu(A)
```

```
## $P
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
##
## $L
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    2    1    0
## [3,]    3    2    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    0   -3   -6
```

```
## [3,]    0    0    0
```

another LU factorization example of rectangular 2x3 matrix

```
A <- matrix(data = 0:5, 2, 3)
A
```

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

```
plu(A)
```

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

```
## $P
##      [,1] [,2]
## [1,]    0    1
## [2,]    1    0
##
## $L
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
##
## $U
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    0    0    0
```