

Factorizacio_LU

Imanol

13/3/2021

R

Factorizaciones LU sin permutación

```
library(matlib)

A = rbind(c(1,3,0,-1),c(2,1,-1,5),c(0,-2,3,-1),c(1,1,3,1))
luA = LU(A)

# Le puedo pedir la matriz P (será la matriz identidad porque no he permutado ninguna fila)
luA$P

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

# Le puedo pedir la matriz L
luA$L

##      [,1] [,2] [,3] [,4]
## [1,]     1   0.0    0    0
## [2,]     2   1.0    0    0
## [3,]     0   0.4    1    0
## [4,]     1   0.4    1    1

# Le puedo pedir la matriz U
luA$U

##      [,1] [,2] [,3] [,4]
## [1,]     1    3  0.0 -1.0
## [2,]     0   -5 -1.0  7.0
## [3,]     0    0  3.4 -3.8
## [4,]     0    0  0.0  3.0
```

```

# Comprobacion
luA$L%*%luA$U == A

##      [,1] [,2] [,3] [,4]
## [1,] TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE FALSE
## [4,] TRUE TRUE TRUE FALSE

```

Factorizaciones LU con permutación

```

library(matlib)

A = rbind(c(0,1,3),c(1,3,-2),c(-3,-2,-1))
luA = LU(A)

# Le puedo pedir la matriz P (se ha permutado la primera y segunda fila)
luA$P

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

# Le puedo pedir la matriz L
luA$L

##      [,1] [,2] [,3]
## [1,]     1     0     0
## [2,]     0     1     0
## [3,]    -3     7     1

# Le puedo pedir la matriz U
luA$U

##      [,1] [,2] [,3]
## [1,]     1     3    -2
## [2,]     0     1     3
## [3,]     0     0    -28

# Comprobacion
luA$P%*%luA$L%*%luA$U == A

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

```

Resolver sistemas

```
library(matlib)

A = rbind(c(0,1,3),c(1,3,-2),c(-3,-2,-1))
b = c(1,3,-2)

sistema = LU(A,b)

## Warning in if (!backword) 1L:len else len:1L: la condición tiene longitud > 1 y
## sólo el primer elemento será usado

# Le puedo pedir la matriz P (se ha permutado la primera y segunda fila)
sistema$P

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

# Le puedo pedir la matriz L
sistema$L

##      [,1] [,2] [,3]
## [1,]     1     0     0
## [2,]     0     1     0
## [3,]    -3     7     1

# Le puedo pedir la matriz U
sistema$U

##      [,1] [,2] [,3]
## [1,]     1     3    -2
## [2,]     0     1     3
## [3,]     0     0    -28

# Solucion del sistema L*d = b
sistema$d

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

# SOLucion del sistema
sistema$x

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

Python

Factorizaciones LU sin permutación

```
import scipy
import scipy.linalg

A = scipy.array([[1,3,0,-1], [2,1,-1,5], [0,-2,3,-1], [1,1,3,1]])

## <string>:1: DeprecationWarning: scipy.array is deprecated and will be removed in SciPy 2.0.0, use num

P, L, U = scipy.linalg.lu(A)

# Le puedo pedir la matriz P
P # No era necesaria la permutación pero python lo ha visto oportuno

# Le puedo pedir la matriz L

## array([[0., 1., 0., 0.],
##        [1., 0., 0., 0.],
##        [0., 0., 1., 0.],
##        [0., 0., 0., 1.]))

L

# Le puedo pedir la matriz U

## array([[ 1. ,  0. ,  0. ,  0. ],
##        [ 0.5,  1. ,  0. ,  0. ],
##        [ 0. , -0.8,  1. ,  0. ],
##        [ 0.5,  0.2,  1. ,  1. ]])

U

# Comprobación

## array([[ 2. ,  1. , -1. ,  5. ],
##        [ 0. ,  2.5,  0.5, -3.5],
##        [ 0. ,  0. ,  3.4, -3.8],
##        [ 0. ,  0. ,  0. ,  3. ]])

(P.dot(L)).dot(U) == A

## array([[ True,  True,  True,  True],
##        [ True,  True,  True,  True],
##        [ True,  True,  True,  True],
##        [ True,  True,  True,  True]])
```

Factorizaciones LU con permutación

```

import scipy
import scipy.linalg

A = scipy.array([[0,1,3], [1,3,-2], [-3,-2,-1]])
P, L, U = scipy.linalg.lu(A)

# Le puedo pedir la matriz P
P # No era necesaria la permutacion pero python lo ha visto oportuno

# Le puedo pedir la matriz L

## array([[0., 0., 1.],
##        [0., 1., 0.],
##        [1., 0., 0.]])  

L  

# Le puedo pedir la matriz U

## array([[ 1.          ,  0.          ,  0.          ],
##        [-0.33333333,  1.          ,  0.          ],
##        [-0.          ,  0.42857143,  1.          ]])  

U  

# Comprobacion

## array([[-3.          , -2.          , -1.          ],
##        [ 0.          ,  2.33333333, -2.33333333],
##        [ 0.          ,  0.          ,  4.          ]])  

(P.dot(L)).dot(U) == A  

## array([[ True,  True,  True],
##        [ True,  True,  True],
##        [ True,  True,  True]])
```

Reducir la informacion del metodo LU

Se usa esto para ahorrar memoria teniendo muillones de datos

```

import scipy
import scipy.linalg

A = scipy.array([[0,1,3], [1,3,-2], [-3,-2,-1]])
LU, piv = scipy.linalg.lu_factor(A)

# L y U juntas
LU # Diagonal superior U y la inferior L

# Nos da que filas fueron intercambiadas
```

```

## array([[-3.          , -2.          , -1.          ],
##        [-0.33333333,  2.33333333, -2.33333333],
##        [-0.          ,  0.42857143,  4.          ]])

piv # La primera fila ha sido cambiada por la tercera (0->2), la segunda fila se mantiene (1) y la tercera (2)

## array([2, 1, 2], dtype=int32)

```

Resolver sistemas

```

import scipy
import scipy.linalg

A = scipy.array([[0,1,3], [1,3,-2], [-3,-2,-1]])

LU, piv = scipy.linalg.lu_factor(A)

b = [1, 3, -2]

x = scipy.linalg.lu_solve((LU,piv),b)
x

## array([-0.0000000e+00,  1.0000000e+00, -1.98254112e-18])

```

Matlab

Factorizaciones LU sin permutación

```

A = [1,3,0,-1; 2,1,-1,5; 0,-2,3,-1; 1,1,3,1]
[L,U,P] = lu(A);

# Le puedo pedir la matriz P
P

# Le puedo pedir la matriz L
L

# Le puedo pedir la matriz U
U

# Comprobacion
L*U == A

```

Factorizaciones LU con permutación

```

import scipy
import scipy.linalg

A = scipy.array([[0,1,3], [1,3,-2], [-3,-2,-1]])
P, L, U = scipy.linalg.lu(A)

# Le puedo pedir la matriz P
P # No era necesaria la permutacion pero python lo ha visto oportuno

# Le puedo pedir la matriz L

## array([[0., 0., 1.],
##        [0., 1., 0.],
##        [1., 0., 0.]])
```

L

Le puedo pedir la matriz U

```

## array([[ 1.          ,  0.          ,  0.          ],
##        [-0.33333333,  1.          ,  0.          ],
##        [-0.          ,  0.42857143,  1.          ]])
```

U

Comprobacion

```

## array([[-3.          , -2.          , -1.          ],
##        [ 0.          ,  2.33333333, -2.33333333],
##        [ 0.          ,   0.          ,   4.          ]])
```

(P.dot(L)).dot(U) == A

```

## array([[ True,  True,  True],
##        [ True,  True,  True],
##        [ True,  True,  True]])
```