Tarea 8

Factorización LU

Daniel Eduardo Macias Estrada

3/12/2020

Pregunta 1

Encuentra las factorizaciones $A = LU$ o $PA = LU$ de las siguientas matrices

A)

$$A_1 = \begin{pmatrix} 0 & 2 & -3 & 4 \\ 0 & 0 & -5 & -1 \\ 5 & -1 & -2 & 0 \\ -2 & 0 & 4 & 6 \end{pmatrix}$$

Resultado con R

```
library(matlib)
A = rbind(c(0,2,-3,4), c(0,0,-5,-1), c(5,-1,-2,0), c(-2,0,4,6))
luA = LU(A)
luA$L
```

```
     [,1] [,2]      [,3] [,4]
[1,]    1  0.0  0.000000    0
[2,]    0  1.0  0.000000    0
[3,]    0 -0.5  1.000000    0
[4,]    0  0.0 -1.142857    1
```

```
luA$U
```

```
         [,1] [,2] [,3]      [,4]
[1,] 0.000000    0 -5.0 -1.000000
[2,] 0.000000    2 -3.0  4.000000
[3,] 5.000000    0 -3.5  2.000000
[4,] 3.714286    0  0.0  8.285714
```

```
luA$P
```

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

```
# Como el resultado de U no es una matriz superior, se comprueba que al menos el resultado
# del producto da la matriz A
luA$L%*%luA$U
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    0   -5   -1
[2,]    0    2   -3    4
[3,]    5   -1   -2    0
[4,]   -2    0    4    6
```

Resultado con Python

```
import numpy as np
import scipy.linalg
A = np.array([[0,2,-3,4],[0,0,-5,-1],[5,-1,-2,0],[-2,0,4,6]])
P,L,U = scipy.linalg.lu(A)
L
```

```
array([[ 1.  ,  0.  ,  0.  ,  0.  ],
       [ 0.  ,  1.  ,  0.  ,  0.  ],
       [ 0.  ,  0.  ,  1.  ,  0.  ],
       [-0.4 , -0.2 , -0.52,  1.  ]])
```

```
U
```

```
array([[ 5.  , -1.  , -2.  ,  0.  ],
       [ 0.  ,  2.  , -3.  ,  4.  ],
       [ 0.  ,  0.  , -5.  , -1.  ],
       [ 0.  ,  0.  ,  0.  ,  6.28]])
```

Resultado con Octave

```
A = [0 2 -3 4; 0 0 -5 -1; 5 -1 -2 0; -2 0 4 6];
[L U P] = lu(A)
```

```
/usr/libexec/octave/5.2.0/exec/x86_64-pc-linux-gnu/octave-gui: /home/dan22/anaconda3/envs/curso_AL/lib/
L =

   1.00000   0.00000   0.00000   0.00000
   0.00000   1.00000   0.00000   0.00000
   0.00000   0.00000   1.00000   0.00000
  -0.40000  -0.20000  -0.52000   1.00000

U =

   5.00000  -1.00000  -2.00000   0.00000
   0.00000   2.00000  -3.00000   4.00000
   0.00000   0.00000  -5.00000  -1.00000
   0.00000   0.00000   0.00000   6.28000

P =
```

```
Permutation Matrix

   0   0   1   0
   1   0   0   0
   0   1   0   0
   0   0   0   1
```

Comprobación de los resultados a mano con R

```
U = rbind(c(1,0,-2,-3),c(0,1,-8,-15),c(0,0,1,1/5),c(0,0,0,1))
L = rbind(c(-2,0,0,0), c(5,-1,0,0), c(0,0,-5,0), c(0,2,13,157/5))
PA = rbind(c(-2,0,4,6), c(5,-1,-2,0), c(0,0,-5,-1), c(0,2,-3,4))
PA == L%*%U
```

```
     [,1] [,2] [,3] [,4]
[1,] TRUE TRUE TRUE TRUE
[2,] TRUE TRUE TRUE TRUE
[3,] TRUE TRUE TRUE TRUE
[4,] TRUE TRUE TRUE TRUE
```

B)

$$A_2 = \begin{pmatrix} 1 & 2 & -1 & 4 \\ 0 & -1 & 5 & 8 \\ 2 & 3 & 1 & 4 \\ 1 & -1 & 6 & 4 \end{pmatrix}$$

Resultado con R

```
A = rbind(c(1,2,-1,4), c(0,-1,5,8), c(2,3,1,4), c(1,-1,6,4))
luA = LU(A)
luA$L
```

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

```
luA$U
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2   -1    4
[2,]    0   -1    5    8
[3,]    0    0   -2  -12
[4,]    0    0    0   24
```

```
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
```

Resultado con Python

```
A = np.array([[1,2,-1,4],[0,-1,5,8],[2,3,1,4],[1,-1,6,4]])
P,L,U = scipy.linalg.lu(A)
L
```

```
array([[ 1.        ,  0.        ,  0.        ,  0.        ],
       [ 0.5       ,  1.        ,  0.        ,  0.        ],
       [ 0.        ,  0.4       ,  1.        ,  0.        ],
       [ 0.5       , -0.2       , -0.14285714,  1.        ]])
```

```
U
```

```
array([[ 2.        ,  3.        ,  1.        ,  4.        ],
       [ 0.        , -2.5       ,  5.5       ,  2.        ],
       [ 0.        ,  0.        ,  2.8       ,  7.2       ],
       [ 0.        ,  0.        ,  0.        ,  3.42857143]])
```

```
P
```

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

Resultado con Octave

```
A = [1 2 -1 4; 0 -1 5 8; 2 3 1 4; 1 -1 6 4];
[L U P] = lu(A)
```

```
/usr/libexec/octave/5.2.0/exec/x86_64-pc-linux-gnu/octave-gui: /home/dan22/anaconda3/envs/curso_AL/lib/
L =

   1.00000   0.00000   0.00000   0.00000
   0.50000   1.00000   0.00000   0.00000
   0.00000   0.40000   1.00000   0.00000
   0.50000  -0.20000  -0.14286   1.00000

U =

   2.00000   3.00000   1.00000   4.00000
   0.00000  -2.50000   5.50000   2.00000
   0.00000   0.00000   2.80000   7.20000
   0.00000   0.00000   0.00000   3.42857
```

```
P =

Permutation Matrix

   0   0   1   0
   0   0   0   1
   0   1   0   0
   1   0   0   0
```

Comprobación de los resultados a mano con Python

```
U = np.array([[1,2,-1,4],[0,1,-5,-8],[0,0,1,6],[0,0,0,1]])
L = np.array([[1,0,0,0],[0,-1,0,0],[2,-1,-2,0],[1,-3,-8,24]])
A == L.dot(U)
```

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

C)

$$A_3 = \begin{pmatrix} 2 & 4 & -2 & 0 \\ 3 & 7 & 5 & -4 \\ -1 & 2 & -2 & 5 \\ 6 & 1 & 0 & 2 \end{pmatrix}$$

Resultado con R

```
A = rbind(c(2,4,-2,0), c(3,7,5,-4), c(-1,2,-2,5), c(6,1,0,2))
luA = LU(A)
luA$L
```

```
     [,1] [,2]      [,3] [,4]
[1,]  1.0    0  0.000000    0
[2,]  1.5    1  0.000000    0
[3,] -0.5    4  1.000000    0
[4,]  3.0  -11 -2.685714    1
```

```
luA$U
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    4   -2  0.0
[2,]    0    1    8 -4.0
[3,]    0    0  -35 21.0
[4,]    0    0    0 14.4
```

```
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
```

Resultado con Python

```
A = np.array([[2,4,-2,0], [3,7,5,-4], [-1,2,-2,5], [6,1,0,2]])
P,L,U = scipy.linalg.lu(A)
L
```

```
array([[ 1.        ,  0.        ,  0.        ,  0.        ],
       [ 0.5       ,  1.        ,  0.        ,  0.        ],
       [ 0.33333333,  0.56410256,  1.        ,  0.        ],
       [-0.16666667,  0.33333333,  0.7606383 ,  1.        ]])
```

```
U
```

```
array([[ 6.        ,  1.        ,  0.        ,  2.        ],
       [ 0.        ,  6.5       ,  5.        , -5.        ],
       [ 0.        ,  0.        , -4.82051282,  2.15384615],
       [ 0.        ,  0.        ,  0.        ,  5.36170213]])
```

```
P
```

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

Resultado con Octave

```
A = [2 4 -2 0; 3 7 5 -4; -1 2 -2 5; 6 1 0 2]
[L U P] = lu(A)
```

```
/usr/libexec/octave/5.2.0/exec/x86_64-pc-linux-gnu/octave-gui: /home/dan22/anaconda3/envs/curso_AL/lib/
A =

   2   4  -2   0
   3   7   5  -4
  -1   2  -2   5
   6   1   0   2

L =

   1.00000   0.00000   0.00000   0.00000
   0.50000   1.00000   0.00000   0.00000
   0.33333   0.56410   1.00000   0.00000
  -0.16667   0.33333   0.76064   1.00000
```

```
U =

   6.00000   1.00000   0.00000   2.00000
   0.00000   6.50000   5.00000  -5.00000
   0.00000   0.00000  -4.82051   2.15385
   0.00000   0.00000   0.00000   5.36170

P =

Permutation Matrix

   0   0   0   1
   0   1   0   0
   1   0   0   0
   0   0   1   0
```

Comprobación de los resultados a mano con Octave

```
U = [1 2 -1 0; 0 1 8 -4; 0 0 1 -3/5; 0 0 0 1];
L = [2 0 0 0; 3 1 0 0; -1 4 -35 0; 6 -11 94 72/5];
A = [2 4 -2 0; 3 7 5 -4; -1 2 -2 5; 6 1 0 2];
L*U
```

```
/usr/libexec/octave/5.2.0/exec/x86_64-pc-linux-gnu/octave-gui: /home/dan22/anaconda3/envs/curso_AL/lib/l
ans =

   2.00000   4.00000  -2.00000   0.00000
   3.00000   7.00000   5.00000  -4.00000
  -1.00000   2.00000  -2.00000   5.00000
   6.00000   1.00000   0.00000   2.00000
```

D)

$$A_4 = \begin{pmatrix} 0 & 2 & 3 & 1 \\ 0 & 4 & -1 & 5 \\ 2 & 0 & 3 & 1 \\ 1 & -4 & 5 & 6 \end{pmatrix}$$

Resultado con R

```
A = rbind(c(0,2,3,1), c(0,4,-1,5), c(2,0,3,1), c(1,-4,5,6))
luA = LU(A)
luA$L
```

```
     [,1] [,2]     [,3] [,4]
[1,]    1    0 0.000000    0
[2,]    0    1 0.000000    0
[3,]    0    0 1.000000    0
[4,]    0   -2 3.666667    1
```

```
luA$U
```

```
          [,1] [,2] [,3]     [,4]
[1,]  0.000000    4   -1 5.000000
[2,]  0.000000    2    3 1.000000
[3,]  2.000000    0    3 1.000000
[4,] -6.333333    0    0 4.333333
```

```
luA$P
```

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

```
# Como el resultado de U no es una matriz superior, se comprueba que al menos el resultado
# del producto da la matriz A
luA$L%*%luA$U
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    4   -1    5
[2,]    0    2    3    1
[3,]    2    0    3    1
[4,]    1   -4    5    6
```

Resultado con Python

```
A = np.array([[0,2,3,1],[0,4,-1,5],[2,0,3,1],[1,-4,5,6]])
P,L,U = scipy.linalg.lu(A)
L
```

```
array([[ 1.        ,  0.        ,  0.        ,  0.        ],
       [ 0.        ,  1.        ,  0.        ,  0.        ],
       [ 0.        ,  0.5       ,  1.        ,  0.        ],
       [ 0.5       , -1.        ,  0.71428571,  1.        ]])
```

```
U
```

```
array([[ 2.        ,  0.        ,  3.        ,  1.        ],
       [ 0.        ,  4.        , -1.        ,  5.        ],
       [ 0.        ,  0.        ,  3.5       , -1.5       ],
       [ 0.        ,  0.        ,  0.        , 11.57142857]])
```

```
P
```

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

Resultado con Octave

```
A = [0 2 3 1; 0 4 -1 5; 2 0 3 1; 1 -4 5 6];
[L U P] = lu(A)
```

```
/usr/libexec/octave/5.2.0/exec/x86_64-pc-linux-gnu/octave-gui: /home/dan22/anaconda3/envs/curso_AL/lib/l
L =

   1.00000   0.00000   0.00000   0.00000
   0.00000   1.00000   0.00000   0.00000
   0.00000   0.50000   1.00000   0.00000
   0.50000  -1.00000   0.71429   1.00000

U =

    2.00000    0.00000    3.00000    1.00000
    0.00000    4.00000   -1.00000    5.00000
    0.00000    0.00000    3.50000   -1.50000
    0.00000    0.00000    0.00000   11.57143

P =

Permutation Matrix

   0   0   1   0
   0   1   0   0
   1   0   0   0
   0   0   0   1
```

Comprobación de los resultados a mano con R

```
U = rbind(c(1,-4,5,6), c(0,1,-7/8,-11/8), c(0,0,1,21/5), c(0,0,0,1))
L = rbind(c(1,0,0,0), c(2,8,0,0), c(0,4,5/2,0), c(0,2,19/4,-81/5))
PA = rbind(c(1,-4,5,6), c(2,0,3,1), c(0,4,-1,5), c(0,2,3,1))
PA == L%*%U
```

```
     [,1] [,2] [,3] [,4]
[1,] TRUE TRUE TRUE TRUE
[2,] TRUE TRUE TRUE TRUE
[3,] TRUE TRUE TRUE TRUE
[4,] TRUE TRUE TRUE TRUE
```