Factorizaciones LU con Python

Ramon Ceballos

27/2/2021

Factorizaciones LU con Python

1. Obtener las matrices LU en Python

Método A con scipy.linalg.lu()

Para realizar una factorización LU con `Python`, podemos utilizar la función **`scipy.linalg.lu()`** introduciendo por parámetro una matriz cuadrada. Para ello, habrá que instalar la librería `scipy` mediante `py_install("scipy")`.

La función `scipy.linalg.lu()` devolverá tres matrices: P, L **y** U.

Veámoslo con un ejemplo.

Ejemplo 1

Encontremos la factorización LU de la siguiente matriz:

$$A = \begin{pmatrix} 1 & 3 & 0 & -1 \\ 2 & 1 & -1 & 5 \\ 0 & -2 & 3 & -1 \\ 1 & 1 & 3 & 1 \end{pmatrix}$$

```
import scipy
import scipy.linalg
import numpy as np

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

En este caso, aunque no fuese necesaria la permutación, `Python` la ha realizado.

```
P
```

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

Las matrices L y U son:

```
L
```

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

```
U
```

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

Ejemplo 2.1

Encontremos ahora la factorización LU de la matriz:

$$A = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 3 & -2 \\ -3 & -2 & -1 \end{pmatrix}$$

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

En este caso, podemos ver como también se han permutado filas, ya que la matriz P no es la matriz identidad.

```
P
```

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

En este caso, podemos ver como también se han permutado filas, ya que la matriz P no es la matriz identidad

Las matrices L y U son:

```
L
```

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

```
U
```

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

Método B con scipy.linalg.lu_factor()

Para realizar una factorización LU con `Python`, también podemos utilizar la función **`scipy.linalg.lu_factor()`** introduciendo por parámetro una matriz cuadrada.

La función devolverá dos elementos matrices:

- Una matriz en cuya parte inferior se corresponde con la matriz triangular inferior L y cuya parte superior se corresponde con la matriz triangular superior U.
- Un vector de índices, **`piv`** que indica que la fila i se ha intercambiado con la fila **`piv[i]`**.

Veámoslo con un ejemplo.

Ejemplo 2.2

Encontremos ahora la factorización LU de la matriz:

$$A = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 3 & -2 \\ -3 & -2 & -1 \end{pmatrix}$$

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

Matriz obtenida por este método.

```
LU
```

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

```
L #Resultado anterior
```

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

```
LU
```

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

```
U #Resultado anterior
```

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

Aquí observamos que la primera fila (la 0), se ha cambiado con la tercera; la segunda se ha quedado tal cual; y la tercera, una vez realizado el primer intercambio, se ha quedado en el sitio.

```
piv
```

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

```
P #Resultado anterior
```

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

2. Resolver sistemas de ecuaciones con matrices LU en Python

Con lo visto anteriormente, ahora somos capaces de resolver sistemas utilizando factorización LU con `Python`. Esto lo podemos hacer con la función **`scipy.linalg.lu_solve()`** a la cual introducimos por parámetros la tupla **`(LU,piv)`** y el vector de términos independientes del sistema, **`b`**.

Esta función devuelve únicamente el vector solución.

Ejemplo 3

Consideremos el sistema:

$$\begin{cases} & x_2 & + & 3x_3 & = & 1 \\ x_1 & + & 3x_2 & - & 2x_3 & = & 3 \\ -3x_1 & - & 2x_2 & - & x_3 & = & -2 \end{cases}$$

```
A = np.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.,  1.,  0.])
```