Factorización LU en Python

Daniel Eduardo Macias Estrada

2/12/2020

Introducción

En el siguiente documento se explicará de manera breve, el manejo de la factorización LU con el lenguaje de programación Python.

Toda la información recabada está basado enteramente de la obra de Juan Gabrial Gomila Salas, CEO de Frogames, Matemático, Data Scientist & Game Designer.

Uso de la Factorización LU en Python

Con ayuda de la función **scipy.linalg.lu()** de la libreria **scipy**, podemos hacer uso de la factorización LU. Ésta toma como argumento a una matriz cuadrada.

Tres matrices serán devueltas al usar la función: P, L y U

Ejemplo 1

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

La salida P, sería:

```
P
```

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

La salida de L, es

```
L
```

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

La salida de U, es

```
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

Econntraremos 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)
```

La salida de P, es

```
P
```

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

La salida de L, es

```
L
```

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

La salida de U, es

```
U
```

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

Otra forma de usar la factoriación LU

Existe otra función la cual requiere menos memoria a comparación de la mostrada anteriormente. Está función es **scipy.linalg.lu_factor()**, la cual aceptaba como argumento una matriz cuadrada.

Los elementos que devolverá son:

- Una matriz 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]**

Retomando los ejemplos anteriores

Ejemplo 2

Encontraremos 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)
```

Como se ve a continuación, en la variable LU, se almaceno tanto L como U, para ahorrar espacio de memoria. Para acceder a cada uno, se hará lo siguiente

```
LU
```

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

```
L
#Matriz L
```

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

```
U
#Matriz U
```

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

En le caso del vector de indices, se ve lo siguiente

```
piv
```

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

```
P
#Resultado anterior
```

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

Se observa 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

Resolución de sistemas de ecuaciones lineales con factorización LU en Python

Haciendo uso de la función anterior, podremos resolver los sistemas lineales en Python. La función **scipy.linalg.lu_solve**, recibe como parámetros a una tupla que contenga **(LU, piv)** y el vector de términos independinetes **b**

Lo único que devuelve es el vector de 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([-0.00000000e+00,  1.00000000e+00, -1.98254112e-18])
```