SCIENCE | PYTHON

SOLVING LINEAR EQUATIONS USING MATRICES AND PYTHON

OCTOBER 30, 2015 | ID.STUDIO

Header image credit: Datafloq

Don't miss the second part of this series, Linear equations with Python: the QR decomposition

In a previous article, we looked at solving an LP problem, i.e. a system of linear equations with inequality constraints. If our set of linear equations has constraints that are deterministic, we can represent the problem as matrices and apply matrix algebra. Matrix methods represent multiple linear equations in a compact manner while using the existing matrix library functions.

We will be using NumPy (a good tutorial here) and SciPy (a reference guide here). For installing these amazing packages there are tons of resources on the web, we just point at Installing the SciPy Stack.

An example

As our practice, we will proceed with an example, first writing the matrix model and then using **Numpy** for a solution.

$$\begin{aligned} 2a + b + c &= 4 \\ a + 3b + 2c &= 5 \\ a &= 6 \end{aligned}$$

Now, we can formalize the problem with matrices:

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 0 & 0 \end{bmatrix}, x = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, B = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

Then, the linear equations can be written this way:

$$Ax = B$$

to solve for the vector x, we must take the inverse of matrix A and the equation is written as follows:

$$x = A^{-1}B$$

Using numpy to solve the system

```
import numpy as np
# define matrix A using Numpy arrays
A = np.array([[2, 1, 1],
 [1, 3, 2],
 [1, 0, 0]])

#define matrix B
B = np.array([4, 5, 6])

# linalg.solve is the function of NumPy to solve a system of linear
scalar equations
print "Solutions:\n",np.linalg.solve(A, B )

Solutions:
[  6.  15. -23.]
```

So the solutions are: $a = 6, b = 15, c = -23$

When matrices grow up

As the number of variables increases, the size of matrix A increases as well and it becomes computationally expensive to get the matrix inversion of A. Among the various methods, we will consider 3 procedures in order to get matrix A factorized into simpler matrices: the LU decomposition, the QR decomposition and the Jacobi iterative method.

LU decomposition

The LU decomposition, also known as upper lower factorization, is one of the methods of solving square systems of linear equations. As the name implies, the LU factorization decomposes the matrix A into A product of two matrices: a lower triangular matrix L and an upper triangular matrix U. The decomposition can be represented as follows:

$$LU = A$$

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\begin{bmatrix} l_{11}u_{11} & l_{11}u_{12} & l_{11}u_{13} \\ l_{21}u_{11} & l_{21}u_{12} + l_{22}u_{22} & l_{21}u_{13} + l_{22}u_{23} \\ l_{31}u_{11} & l_{31}u_{12} + l_{32}u_{22} & l_{31}u_{13} + l_{32}u_{23} + l_{33}u_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

LU decomposition with SciPy

```
#import linalg package of the SciPy module for the LU decomp
import scipy.linalg as linalg
#import NumPy
import numpy as np
#define A same as before
A = np.array([[2., 1., 1.],
 [1., 3., 2.],
 [1., 0., 0.]])

#define B
B = np.array([4., 5., 6.])

#call the lu_factor function
LU = linalg.lu_factor(A)

#solve given LU and B
```

```
x = linalg.lu_solve(LU, B)
print "Solutions:\n",x

#we get the same solution as before
Solutions:
[  6.  15. -23.]

#now we want to see how A has been factorized, P is the so called
Permutation matrix
P, L, U = scipy.linalg.lu(A)

print P
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
print L
[[ 1.   0.   0. ]
 [ 0.5  1.   0. ]
 [ 0.5 -0.2  1. ]]
print U
[[ 2.   1.   1. ]
 [ 0.   2.5  1.5]
 [ 0.   0.  -0.2]]
```

From L and U variables as printed in the code, we can represent A factorized into:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0.5 & -0.2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 \\ 0 & 2.5 & 1.5 \\ 0 & 0 & -0.2 \end{bmatrix} = A = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 0 & 0 \end{bmatrix}$$

In the next episode we will continue with 2 other methods of solving linear equations: QR decomposition and a some way different approach, the Jacobi method.

[ssba]

FEATURED PYTHON