Matrices en python

Daniel Eduardo Macias Estrada

8/8/2020

Introducción

El contenido de este documento se realizó con la finalidad de recolectar información básica sobre el manejo de matrices con python. Con ayuda de la libreria **numpy**, pues es una librería de funciones matemáticas de alto nivel.

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

Declaración de una matriz La manera de representar las matrices en python es mediante filas, que son estructura de datos de este lenguaje.

Para crear un vector fila

```
row = [1,5,3]
row
```

```
## [1, 5, 3]
```

Para crear un vector columna

```
col = [[1],[5],[3]]
col
```

```
## [[1], [5], [3]]
```

En caso de una matriz cualquiera

```
M = [[1,2,3], [4,5,6], [7,8,9]]
M
```

```
## [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Acceder a un elmento de la matriz y array de numpy Para obtener un elemento, se utiliza la sintáxis siguiente

```
M[1][1]
```

```
## 5
```

El primer corchete indica la fila y el segundo la columna del elemento. Además es necesario considerar que, en lugar de R, python posiciona sus elementos empezando desde el 0. Por ello el elemento m_{11} es igual a

```
M[0][0]
```

```
## 1
```

Para obtener una fila, solo basta colocar un corcheta indicando la posición de la fila que queremos.

```
M[2]
```

```
## [7, 8, 9]
```

En el caso de querer obtener una columna, el proceso es un poco más complejo. Lo anterior visto es una forma de declaración de matrices de manera sencilla. Se volverán a realizar, usando plenamente **numpy**, para tener una mayor manipulación de la matriz.

```
import numpy as np
M = np.array([[1,2,3],[4,5,6],[7,8,9]])
M
```

```
## array([[1, 2, 3],
##        [4, 5, 6],
##        [7, 8, 9]])
```

Habiendo realizado las líneas anteriores, M, de ser una lista de python, pasa a ser un array de numpy.

De la función **np.array()** podemos resaltar que tiene un parámetro llamado **dtype** en el cual podemos indicar el tipo de dato de la matriz: int, float, complex...

```
M = np.array([[1,2,3],[4,5,6],[7,8,9]], dtype = complex)
M
```

```
## array([[1.+0.j, 2.+0.j, 3.+0.j],
##        [4.+0.j, 5.+0.j, 6.+0.j],
##        [7.+0.j, 8.+0.j, 9.+0.j]])
```

```
M = np.array([[1,2,3],[4,5,6],[7,8,9]], dtype = float)
M
```

```
## array([[1., 2., 3.],
##        [4., 5., 6.],
##        [7., 8., 9.]])
```

Con esta sintáxis, al momento de querer extraer un elemento o fila, podemos usar las expresiones anteriores, sin embargo existe una manera distinta para obtener las filas y columnas de una matriz

```
M = np.array([[1,2,3],[4,5,6],[7,8,9]])

M[1,:] # Obtiene la segunda fila
```

```
## array([4, 5, 6])
```

```
M[:,0] # Obtiene la primer columna
```

```
## array([1, 4, 7])
```

Matriz nula y matriz de unos Para crear una matriz nula se usa la función **np.zeros((fil,col))**. Para indicar el número de filas y columnas se pasa como parámetro una tupla con ambos valores, siendo el primero el número de filas

```
print(np.zeros((3,4)))
```

```
## [[0. 0. 0. 0.]
##  [0. 0. 0. 0.]
##  [0. 0. 0. 0.]]
```

Para crear una matriz con solamente valores de uno, se usa la función **np.ones((fil,col))**

```
print(np.ones((5,4)))
```

```
## [[1. 1. 1. 1.]
##  [1. 1. 1. 1.]
##  [1. 1. 1. 1.]
##  [1. 1. 1. 1.]
##  [1. 1. 1. 1.]]
```

Matrices diagonales Declarar una matriz diagonal en python es parecido a hacerlo en R. Se hará uso de la función **np.diag()**

```
N = np.diag([1,2,3,4])
N
```

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

Y para consultar los elementos de la diagonal principal de una matriz, se hace uso de la misma función

```
np.diag(N)
```

```
## array([1, 2, 3, 4])
```

Dimensión de una matriz Para poder conocer el orden de una matriz, utilizamos la función **np.shape()**

```
np.shape(M)
```

```
## (3, 3)
```

Manipulación de matrices con python

Suma de los elementos de una matriz Para obtener el resultado de sumar todos los elementos de una matriz, usamos la función **np.sum()**

```
M
```

```
## array([[1, 2, 3],
##        [4, 5, 6],
##        [7, 8, 9]])
```

```
np.sum(M)
```

```
## 45
```

En caso de requerir sumar por filas o por columnas, debemos añadir el parámetro **axis**, el cual recibirá un 0 o 1, en donde 0 indica los elementos horizontales (filas), y 1 los elementos verticales (columnas)

```
np.sum(M, axis = 0)
```

```
## array([12, 15, 18])
```

```
np.sum(M, axis = 1)
```

```
## array([ 6, 15, 24])
```

Producto de los elementos de una matriz Para calcular la multiplicación de todos los elementos de una matriz, usamos la función **np.prod()**

```
np.prod(M)
```

```
## 362880
```

Media aritmética de los elementos de una matriz La librería **numpy** nos provee una función parecida a la de R, la cual permite obtener la media aritmética

```
np.mean(M)
```

```
## 5.0
```

Además de que popdemos sacar la media de cada fila o columna.

```
np.mean(M, axis = 0) #Por filas
```

```
## array([4., 5., 6.])
```

```
np.mean(M, axis = 1) #Por columnas
```

```
## array([2., 5., 8.])
```

Operaciones con matrices

Transpuesta de una matriz Para calcular la transpuesta de una matriz se utiliza la función **.transpose()**. En lugar de que la función reciba como parámetro a la matriz, del objeto matriz se obtiene dicho método

```
print(M.transpose())
```

```
## [[1 4 7]
##  [2 5 8]
##  [3 6 9]]
```

Traza de una matriz Para obtener la traza de una matriz, es decir, la suma de los elementos de la diagonal principal, usamos la función **.trace()**

```
print(M.trace())
```

```
## 15
```

```
A = np.array([[3,1],[5,-2]])
B = np.array([[3,-6],[6,7]])
print(A+B)
```

Suma de matrices

```
## [[ 6 -5]
##  [11  5]]
```

```
print(A*5)
```

Producto por un escalar

```
## [[ 15   5]
##  [ 25 -10]]
```

Producto de matrices Para multiplicar 2 matrices, se usa la sintáxis siguiente

```
print(A.dot(B))
```

```
## [[ 15 -11]
##  [  3 -44]]
```

Si se utiliza el símbolo asterístico para indicar la multiplicación como es común, ésta se dara elemento por elemento

```
A*B
```

```
## array([[  9,  -6],
##        [ 30, -14]])
```

```
print(np.linalg.matrix_power(A,5))
```

Potencia de una matriz

```
## [[718 155]
##  [775 -57]]
```

Rango e inversa

Rango de una matriz Para calcular el rango, el cual se define como el número de filas no nulas de una matriz equivalente (escalonada o escalonada reducida), se usará la función **np.linalg.matrix_rank()**

```
np.linalg.matrix_rank(A)
```

```
## 2
```

```
np.linalg.matrix_rank(A)
```

```
## 2
```

Inversa de una matriz Esta se obtiene con a función **np.linalg.inv()**

```
print(np.linalg.inv(A))
```

```
## [[ 0.18181818  0.09090909]
##  [ 0.45454545 -0.27272727]]
```

```
print(np.linalg.inv(A).dot(A)) #Comprobar si el producto con la inversa da la identidad
```

```
## [[1. 0.]
##  [0. 1.]]
```