Librerías numpy y matplotlib en python

Profesor Francisco Medina

```python
import math
import numpy as np
import matplotlib.pyplot as plt
b = np.array([0, 1, 2, 3])
print ("B =" , b )
#--------------------------
# tamaño del vector
x = len(b)
print ("El tamaño del vector B es ",x)
#tamaño de la matriz
print ("El numero de elementos de B es ", b.size)

b=np.array([[2,5,1],[5,6,3],[8,9,2],[5,6,7]]) #permite crear una matriz
print (b)
print ("Numero de elementos de b", len(b)) #retorna el tamaño de la matriz
print ("Numero de elementos de b", b.size)

#-----------------------------------------------
# sumar los elementos el vector sin usar numpy
s=0
for i in range (0,len(b)):
    s=s+b[i]
print ("La suma de los elemendo de B es ",s)

# sumar los elementos el vector usando numpy
e=np.sum(b)
print ("La suma de los elemendo de B es ", e)

#---------------------------------------
#Suma de vectores
a = np.array([9.4 , 3.42])
b = np.array ([9.71 ,  7.05])
c = np.array ([-6.93 , 4])
d = np.array ([-18 , 0])
e = np.array ([-10.52, -21.57])
s = a+b+c+d+e
print (s)

```

```python
#----------------------------------------
#producto punto entre dos vectores
a=np.array([2, 4, 0])
b=np.array([-2, -10, 0])
propunto=sum(np.multiply(a , b))
print("El producto punto es", propunto)
propunto=np.dot(a,b)
print("El producto punto es", propunto)

#----------------------------------------
#producto cruz entre dos vectores
procruz=np.cross (a ,b)
print("El producto cruz es",procruz)

#----------------------------------------
# crear un vector con valores aleatorios
a = np.random.rand(5)       # uniformes in [0, 1]
print ("a =", a)

# crear un vector con valores aleatorios Gaussianos
b = np.random.randn(5)      # Gaussianos
print ("b =", b)

# Crear una vector con unos
a = np.ones((10))  # recuerde: (3, 3) es una tupla
print ("Vector de unos:", a)

# Crear una matriz con unos
a = np.ones((3, 3))  # recuerde: (3, 3) es una tupla
print ("Matriz de unos:\n", a)

# Crea una matriz de ceros
b = np.zeros((4, 3))
print ("matriz de ceros: \n", b)

```

```python
# Crear una matriz identidad
c = np.eye(3)
print ("Matriz identidad: \n", c)

# crea una matriz diagonal con los datos del vector
d = np.diag(np.array([1, 2, 3, 4]))
print ("Matriz diagonal: \n", d)

#----------------------------------------
# crear arreglos de acuerdo a un numero de puntos
b = np.linspace(0, 10, 5)   # inicio, final, número de puntos
print ("b =" , b)
c = np.linspace(0, 1, 6)   # inicio, final, número de puntos
print ("c =", c)
d = np.linspace(0, 1, 5, endpoint=False)
print ("d =", d)

 #----------------------------------
 #Crea arreglos uniformemente espaciados
 a = np.arange(20) # 0 ... n-1  (!)
 print (a)
 b = np.arange(1, 20, 2) # inicio,final(excluido),paso
 print(b)
 c = np.arange(1, 20, 3) # inicio,final(excluido),paso
 print(c)

 #-------------------------------------
 #Graficar vectores usando matplotlab (Arreglos en 1D)
 x = np.linspace(0, 3, 20)
 y = np.linspace(0, 9, 20)
 plt.plot(x, y)       # gráfica con línea continua
 plt.plot(x, y, 'o')  # gráfica con línea punteada
 plt.show()           # <-- mostrar la gráfica
                      # (en Ipython no es necesario)

 # graficar matrices (Arreglos en 2D)
 image = np.random.rand(30, 30)
 plt.imshow(image, cmap=plt.cm.hot)
 plt.colorbar()
 plt.show()
```

1.3.4. Operaciones avanzadas

Contenido

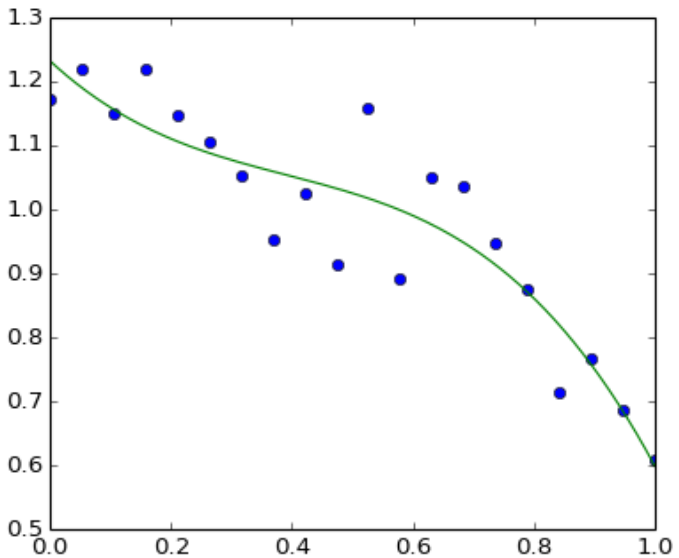
1.3.4.1. Polinomios

Numpy contiene polinomios en diferentes bases :

Por ejemplo, $3x^2 + 2x - 1$

```
>>> p = np.poly1d([3, 2, -1])
>>> p(0)
-1
>>> p.roots
array([-1.        ,  0.33333333])
>>> p.order
2
```

```
>>> x = np.linspace(0, 1, 20)
>>> y = np.cos(x) + 0.3*np.random.rand(20)
>>> p = np.poly1d(np.polyfit(x, y, 3))
>>> t = np.linspace(0, 1, 200)
>>> plt.plot(x, y, 'o', t, p(t), '-')
[<matplotlib.lines.Line2D object at ...>, <matplotlib.lines.Line2D object at
      ...>]
```



Ver http://docs.scipy.org/doc/numpy/reference/routines.polynomials.poly1d.html para más información.

1.3.4.2. Cargando archivos de datos

1.3.4.2.1. Archivos de texto

Ejemplo: `populations.txt`:

```
# year  hare    lynx    carrot
1900    30e3    4e3     48300
1901    47.2e3  6.1e3   48200
1902    70.2e3  9.8e3   41500
1903    77.4e3  35.2e3  38200
```

```
>>> data = np.loadtxt('data/populations.txt')
>>> data
array([[  1900.,  30000.,   4000.,  48300.],
       [  1901.,  47200.,   6100.,  48200.],
       [  1902.,  70200.,   9800.,  41500.],
...
```

```
>>> np.savetxt('pop2.txt', data)
>>> data2 = np.loadtxt('pop2.txt')
```

Nota: Si usted tiene un archivo de texto complicado, puede probar con:

- `np.genfromtxt`
- Usar las funciones de E/S de Python, por ejemplo regexps para parsear (Python es bastante adecuado para esto)