Cheat Sheet Numpy/Matplotlib Matlab

Guía apoyo

Pablo Pizarro R.
pablo.pizarro@ing.uchile.cl

Numpy y Matplotlib son librerías en Python que sirven para realizar operaciones matemáticas tanto de manejo de matrices y vectores como de representación de gráficos, respectivamente.

Para instalar dichas librerías en Linux sólo basta ingresar en el terminal los siguientes comandos:

```
sudo pip install numpy
sudo pip install matplotlib
```

En caso de estar utilizando Windows, a partir de las versiones 2.7.9 para python 2 y 3.4 para python 3, pip viene incluido y se ejecuta con las instrucciones (estando python en el respectivo PATH):

```
python -m pip install numpy
python -m pip install matplotlib
```

Finalmente para acceder a las librerías dentro de Python se deben importar haciendo uso de las siguientes instrucciones:

```
import numpy as np
import matplotlib.pylab as pl
```

A continuación, comandos básicos de numpy y matlab:

- Nan e Infinito:

```
Python: Nan = np.nan; Inf = np.inf
Matlab: _nan = NaN; _inf = Inf
```

- Crear matriz de unos.

```
Python: np.ones((2,3))
>> [[1, 1, 1],
[1, 1, 1]]
Matlab: ones(2,3)
>> 1 1 1
1 1 1
```

- Crear matriz de ceros.

```
Python: np.zeros((2,3))
>> [[0, 0, 0],
[0, 0, 0]]
Matlab: zeros(2,3)
>> 0 0 0
0 0 0
```

- Ponderar matriz.

```
Python: np.ones((2,3)) * 5
>> [[5, 5, 5],
[5, 5, 5]]
Matlab: ones(2,3) .* 5
>> 5 5 5
5 5 5
```

- Tamaño matriz.

```
Python: np.zeros((2,3)).shape
>> (2, 3)
Matlab: size(zeros(2,3))
>> 2 3
```

- Accesores.

```
Python: matriz[a,b] o bien matriz[:,b]
Matlab: matriz(a,b) o bien matriz(:,b)
```

- Chequeo contra `NaN`.

```
Python: np.isnan(np.ones((2,3)) * np.nan)
>> [[True, True, True],
[True, True, True]]
Matlab: isnan([1 1 NaN 1 1])
>> 0 0 1 0 0
```

- Chequeo contra `Inf`.

```
Python: np.isinf(np.ones((2,3)) * np.inf)
>> [[True, True, True],
[True, True, True]]
Matlab: isinf([1 1 Inf 1 1])
>> 0 0 1 0 0
```

- Elementos distintos de cero.

```
Python: np.nonzero(np.ones((2,3)))
>> (array([0, 0, 0, 1, 1, 1]), array([0, 1, 2, 0, 1, 2])) ...
Matlab: [Inf Inf 0; 0 Inf Inf]
>> 1 1 0
0 1 1
```

- Flip up-down y left-right.

```
Python: A = np.ones((2,2)) ; A[0,1] = 2; A[1,0] = 3; A[1,1] = 4;
np.flipud(A)
>> [[3, 4],
[1, 2]]
np.fliplr(A)
>> [[2, 1],
[4, 3]]
Matlab: flipud([1 2; 3 4])
>> 3 4
1 2
fliplr([1 2; 3 4])
>> 2 1
4 3
```

- Promedio matriz.

```
Python: A = np.ones((2,2)) ; A[0,1] = 2; A[1,0] = 3; A[1,1] = 4;
np.mean(A) = 2.5
Matlab: mean2([1 2; 3 4]) = 2.5
```

- Gradiente matriz.

```
Python: A = np.matrix([[5, 2, 9],[8, 5, 1],[2, 1, 8]])
[U,V] = np.gradient(A)
>> U = [
[ 3. 3. 8. ]
[1.5 0.5 0.5]
[6. 4. 7.]
]
>> V = [
[3. 2. 7. ]
[3. 3.5 4. ]
[1. 3. 7. ]
]
Matlab: [U, V] = gradient([5 2 9 ; 8 5 1; 2 1 8])
U =
3.0000 2.0000 7.0000
3.0000 3.5000 4.0000
1.0000 3.0000 7.0000
V =
3.0000 3.0000 8.0000
1.5000 0.5000 0.5000
6.0000 4.0000 7.0000
```

- Mostrar matriz.

```
Python: A = np.matrix([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
pl.matshow(A)
pl.show()
Matlab: A = [1 2 3; 4 5 6; 7 8 9]
surf(A) # o bien mesh(A)
```

- Trasponer matriz.

```
Python: (a,b) = ([0, 0, 1, 1, 2, 2],[1, 2, 1, 2, 1, 2]) ; ...
pl.transpose((a,b))
>> [
[0 1]
[0 2]
[1 1]
[1 2]
[2 1]
[2 2]
]
Matlab: transpose([0 0 1 1 2 2; 1 2 1 2 1 2])
>> 0 1
0 2
1 1
1 2
2 1
2 2
```

- Quiver.

```
# Asumiendo vectores U y V del gradiente.
Python: pl.figure() ; pl.quiver(U,V) ; pl.show()
Matlab: quiver(U,V)
```