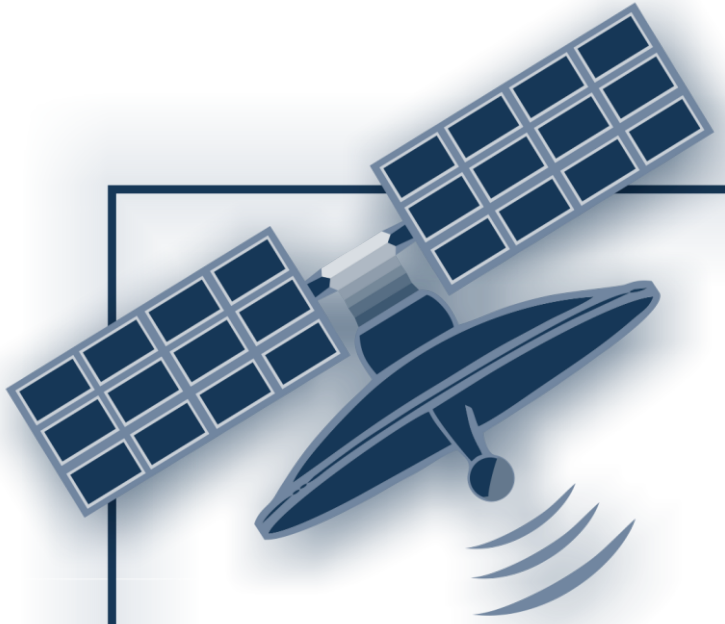

Introdução à Programação para Sensoriamento Remoto

Aula 11 – Introdução ao NumPy

Gilberto Ribeiro de Queiroz
Thales Sehn Körting
Fabiano Morelli

18 de Abril de 2018

Funções matemáticas

```
# exemplos de funções que se aplicam a cada elemento de um ndarray

>>> B = np.arange(3)

>>> B

array([0, 1, 2])

>>> np.exp(B)

array([ 1. , 2.71828183, 7.3890561])

# raiz quadrada

>>> np.sqrt(B)

array([ 0. , 1. , 1.41421356])
```

```
# para acessar os elementos do array, podemos usar um índice por eixo

>>> b = np.arange(12).reshape(3,4)

>>> b

array([[ 0,  1,  2,  3],

       [ 4,  5,  6,  7],

       [ 8,  9, 10, 11]])

>>> b[1,2]

6

>>> b[3,4]

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

IndexError: index 3 is out of bounds for axis 0 with size 3

>>> b[-1]

array([ 8,  9, 10, 11])
```

Números aleatórios

- NumPy oferece uma série de funções para geração de números aleatórios

```
Random values in a given shape.

rand(d0, d1, ..., dn)

Return a sample (or samples) from the “standard normal” distribution.

randn(d0, d1, ..., dn)
```

```
Return random integers from low (inclusive) to high (exclusive).

randint(low[, high, size, dtype])

Random integers of type np.int between low and high, inclusive.

random_integers(low[, high, size])

Return random floats in the half-open interval [0.0, 1.0).

random([size])
```

Números aleatórios

```
# importando as bibliotecas
import numpy as np
import matplotlib.pyplot as plt

# criando conjuntos de pontos aleatórios (500 pontos 2D)
azuis = np.random.random(size=(2,500))
verdes = np.random.random(size=(2,300)) + 0.5

# criando gráfico 2D
plt.figure()
plt.scatter(azuis[0], azuis[1], c = 'blue')
plt.scatter(verdes[0], verdes[1], c = 'green')
plt.show()
```

Hands on 1

- ***offset*** é um número real que é somado a todos os pixels de uma banda.
- **ganho** é um número real que é multiplicado por todos os pixels da banda.
- Suponha uma banda de uma imagem armazenada em uma matriz de 10 linhas e 10 colunas

```
>>> banda = np.zeros([10,10])
# modificar alguns pixels da banda
>>> banda[2,2] = banda[2,-3]
   = banda[4,4] = banda[4,5]
    = banda[6,1] = banda[6,-2]
     = banda[7,2:8] = 0.25
# para mostrar a matriz como uma imagem
>>> import matplotlib.pyplot as plt
>>> plt.imshow(banda, vmax=255)
>>> plt.show()
```

Hands on 1

- ***offset*** é um número real que é somado a todos os pixels de uma banda.
- **ganho** é um número real que é multiplicado por todos os pixels da banda.
- Suponha uma banda de uma imagem armazenada em uma matriz de 10 linhas e 10 colunas

```
>>> banda = np.zeros([10,10])
# modificar alguns pixels da banda
>>> banda[2,2] = banda[2,-3]
   = banda[4,4] = banda[4,5]
    = banda[6,1] = banda[6,-2]
     = banda[7,2:8] = 0.25
# para mostrar a matriz como uma
imagem
>>> import matplotlib.pyplot as
plt
>>> plt.imshow(banda, vmax=255)
>>> plt.show()
# ao não perceber nenhum conteúdo
na imagem, pense num ganho ou
offset para aplicar na imagem
>>> banda *= 500
>>> plt.imshow(banda, vmax=255)
>>> plt.show()
```

Hands on 2

- ***offset*** é um número real que é somado a todos os pixels da banda
- Suponha uma banda de uma imagem armazenada em uma matriz de 10 linhas e 10 colunas
- Faça um programa que aplique um ***offset*** informado pelo usuário em uma banda
- Em alguns casos, o resultado pode ultrapassar o range de uma imagem de 8 bits (entre 0 e 255)

  force para ficar dentro do range (e.g. -50 → 0, e 270 → 255)

Hands on 2

```
# criar a banda
banda = np.arange(100).reshape([10,10])
# definir o offset
offset = 200
# aplicar o offset
banda += offset
banda[banda > 255] = 255
banda[banda < 0] = 0
# desenhar na tela
import matplotlib.pyplot as plt
plt.imshow(banda, cmap='gray')
plt.show()
```

Hands on 3

- Dadas as seguintes séries temporais

```
serie_1 → 168, 398, 451, 337, 186,
232, 262, 349, 189, 204, 220, 220,
207, 239, 259, 258, 242, 331, 251,
323, 106, 1055, 170
serie_2 → 168, 400, 451, 300, 186,
200, 262, 349, 189, 204, 220, 220,
207, 239, 259, 258, 242, 331, 251,
180, 106, 1055, 200
```

- Utilize funções NumPy para calcular a distância euclidiana entre as séries

Hands on 3

```
# definir as series
serie_1 = numpy.array((168, 398, 451, 337, 186,
232, 262, 349, 189, 204, 220, 220, 207, 239,
259, 258, 242, 331, 251, 323, 106, 1055, 170))
serie_2 = numpy.array((168, 400, 451, 300,
186, 200, 262, 349, 189, 204, 220, 220,
207, 239, 259, 258, 242, 331, 251, 180,
106, 1055, 200))
# aplicar o calculo
subtracao = serie_1 - serie_2
quadrado = subtracao * subtracao
somatorio = numpy.sum(quadrado)
distancia_euclidiana = numpy.sqrt(somatorio)
```

Referências Bibliográficas

- NumPy. Acesso: Abril, 2018.