Introdução ao Python

3. NumPy

Rodrigo Barbosa de Santis

Sumário

- Introdução – Python puro vs NumPy
- Inicialização de Vetores
- Gradiente descendente

Python puro vs NumPy

```
# Python puro
>> import time
>> l = 10000000
>> start = time.time()
>>
>> a, b = range(l), range(l)
>> c = []
>> for i in a:
>> c.append(a[i] * b[i])
>>
>> t = time.time() - start
>> print( " Tempo: %s" % t)
Tempo: 4.49s
```

```
# NumPy
>> import time
>> import numpy as np
>> l = 10000000
>> start = time.time()
>>
>> a = np.arange(l)
>> b = np.arange(l)
>> c = a * b
>>
>> t = time.time() - start
>> print( " Tempo: %s" % t)
Tempo: 0.37 s
```

Inicialização de vetores

zeros((M,N)) ones((M,N)) empty((M,N))	vetor com zeros, M linhas, N colunas vetor com uns, M linhas, N colunas vetor vazio, M linhas, N colunas
zeros_like(A) ones_like(A) empty_like(A)	vetor com zeros, mesmo formato de A vetor com uns, mesmo formato de A vetor vazio, mesmo formato de A
random.random((M,N)) identity(N) array([[1.5,2,3],[4,5,6]])	vetor com números aleatórios, MxN matriz identidade NxN, ponto flutuante cria a partir de lista ou tupla
arange(I,F,P) linspace(I,F,N)	vetor com o início I, fim F, passo P vetor com N números no espaço de I até F

Gradiente descendente