Python para Data Science

Acesso rápido NumPy básico

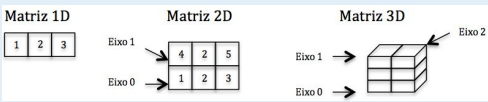
Biblioteca

A biblioteca NumPY é a biblioteca mais importante para computação científica no Python. Ela fornece uma matriz-objeto multidimensional de alta performance, e as ferramentas para trabalhar com essas matrizes.

Em geral, importa-se a biblioteca dessa forma:

```
>>> import numpy as np
```

Matrizes NumPy



Criando matrizes

```
>>> a = np.array([1, 2, 3])
>>> b = np.array([(1.5, 2, 3), (4, 5, 6,)], dtype=float)
>>> c = np.array([[(1.5, 2, 3), (4, 5, 6,)], [(3, 2, 1), (4, 5, 6)], dtype=float)
```

Marcadores iniciais

`>>>np.zeros((3,4))`	Cria matriz de zeros
`>>>np.ones((2,3,4), dtype=np.int16)`	Cria matriz de uns
`>>>d=np.arrange(10,25,5)`	Cria matriz de valores arranjados por valor incremental
`>>> np.linspace(0, 2, 9)`	Cria matriz de valores arranjados por número de amostras
`>>> e = np.full((2, 2), 7)`	Cria matriz constante
`>>> f = np.eye(2)`	Cria matriz identidade 2x2
`>>>np.random.random((2, 2))`	Cria matriz com valores aleatórios
`>>> np.empty((3, 2))`	Cria matriz vazia

I/O

Salvando e carregando no disco

```
>>> np.save('minhamatriz', a)
>>> np.savez('matriz.npz', a, b)
>>> np.load('minhamatriz.npy')
```

Salvando e carregando arquivos de texto

```
>>> np.loadtxt('meuarquivo.txt')
>>> np.genfromtxt('meuarquivo.csv', delimiter=',')
>>> np.savetxt('minhamatriz.txt', a, delimiter=',')
```

Tipos de dados

`>>> np.int64`	Inteiros de 64-bit
`>>> np.float32`	Ponto flutuante de dupla precisão
`>>> np.complex`	Números complexos
`>>> np.bool`	Valores booleanos: TRUE, FALSE
`>>> np.object`	Objeto Python
`>>> np.string_`	String de largura fixa
`>>> np.unicode_`	Unicode de largura fixa

Inspecionando sua matriz

`>>> a.shape`	Dimensões
`>>> len(a)`	Largura da matriz
`>>> b.ndim`	Número de dimensões
`>>> e.size`	Número de elementos
`>>> b.dtype`	Tipo de dados dos elementos
`>>> b.dtype.name`	Nome dos tipos de dados
`>>> b.astype(int)`	Converte uma matriz noutro tipo de dado

Pedindo ajuda

```
>>> np.info(np.ndarray.dtype)
```

Matemática com matrizes

Operações aritméticas

`>>> g = a - b` `array([[-0.5, 0., 0.], [-3., -3., -3.]])`	Subtração
`>>> np.subtract(a, b)`	Subtração
`>>> b + a` `array([[2.5, 4., 6.], [5., 7., 9.]])`	Adição
`>>> np.add(b, a)`	Adição

`>>> a / b`
`array([[0.6667, 1., 1.],`
`[0.25, 0.4, 0.5]])`
	Divisão

`>>> np.divide(a, b)`	Divisão
`>>> a * b` `array([[1.5, 4., 9.], [4., 10., 18.]])`	Multiplicação
`>>> np.multiply(a, b)`	Multiplicação
`>>> np.exp(b)`	Exponenciação
`>>> np.sqrt(b)`	Raiz quadrada
`>>> np.sin(a)`	Função seno
`>>> np.cos(b)`	Função cosseno
`>>> np.log(a, b)`	Função logarítmica
`>>> e.dot(f)` `array([[7., 7.], [7., 7.]])`	Função ponto

Comparações

`>>> a == b`
`array([[False, True, True],`
`[False, False, False]])`
	Comparação de elementos

`>>> a < 2` `array([True, False, False])`	Comparação de elementos
`>>> np.array_equal(a, b)`	Comparação de matrizes

Funções agregadas

`>>> a.sum()`	Soma de matrizes
`>>> a.min()`	Valor mínimo dentro da matriz
`>>> b.max(axis=0)`	Valor máximo numa linha
`>>> b.cumsum(axis=0)`	Soma acumulada dos elementos
`>>> a.mean()`	Média
`>>> b.median()`	Mediana
`>>> a.corrcoef()`	Coeficiente de correlação
`>>> np.std(b)`	Desvio padrão

Copiando matrizes

`>>> h = a.view()`	Cria uma visão da matriz com os mesmos dados
`>>> np.copy(a)`	Cria uma cópia da matriz
`>>> h = a.copy()`	Cria uma cópia profunda da matriz

Ordenando matrizes

`>>> a.sort()`	Ordena uma matriz
`>>> c.sort(axis=0)`	Ordena os elementos de um eixo da matriz

Subconjuntos, cortes, indexação

Subconjuntos	
`>>> a[2]` `3`	Seleciona o elemento no segundo índice
`>>> b[1, 2]` `6.0`	Seleciona o elemento na linha 1 e coluna 2
Cortes	
`>>> a[0:2]` `array([1, 2])`	Seleciona itens no índice 0 e 1
`>>> b[0:2, 1]` `array([2., 5.])`	Seleciona itens nas linhas 0 e 1, na coluna 1
`>>> b[:1]` `array([[1.5, 2., 3.]])`	Seleciona os itens da linha 1

`>>> c[1, ...]`
`array([[[3., 2., 1.],`
`[4., 5., 6.]]])`
	O mesmo que [1, :, :]

`>>> a[ : :-1]` `array([3, 2, 1])`	Matriz a invertida
Indexação booleana	
`>>> a[a > 2]` `array([1])`	Seleciona itens de a menores que 2
Indexação fancy	
`>>> b[[1, 0, 1, 0], [0, 1, 2, 0]]` `array([4., 2., 6., 1.5])`	Seleciona elementos (1,0), (0,1), (1,2) e (0,0)

`>>> b[[1, 0, 1, 0]] [:, [0, 1, 2, 0]]`
`array([[4., 5., 6, 4.])`
`[1.5, 2., 3., 1.5]])`
`[4., 5., 6., 4.]])`
`[1.5, 2., 3., 1.5]])`
	Seleciona um subconjunto das linhas e colunas das matrizes

Manipulação de matrizes

Transpondo uma matriz	
`>>> i = np.transpose(b)`	Permuta as dimensões da matriz
`>>> i.T`	Permuta as dimensões da matriz
Mudando o formato de uma matriz	
`>>> b.ravel()`	Torna a matriz plana
`>>> g.reshape(3, -2)`	Remodela, mas não muda os dados
Adicionando/removendo elementos	
`>>> h.resize((2, 6))`	Retorna nova matriz com forma(2,6
`>>> np.append(h, g)`	Adiciona itens a uma matriz
`>>> np.insert(a, 1, 5)`	Insere itens numa matriz
`>>> np.delete(a, [1])`	Deleta itens de uma matriz
Combinando matrizes	
`>>> np.concatenate((a, d), axis=0)` `array([1, 2, 3, 10, 15, 20])`	Concatena matrizes

`>>> np.vstack((a, b))`
`array([[1.,2.,3.],`
`[1.5,2.,3.],`
`[4.,5.,6.]])`
	Empilha matrizes verticalmente

`>>> np.r_[e,f]`	Empilha matrizes verticalmente

`>>> np.hstack((e,f))`
`array([7.,7.,1.,0.],`
`[7.,7.,0.,1.])`
	Empilha matrizes horizontalmente

`>>> np.column_stack((a,d))`
`array([[1,10],`
`[2,15],`
`[3,20]])`
	Cria matrizes empilhadas por coluna

`>>> np.c_[a,d]`	Cria matrizes empilhadas por coluna
Partindo matrizes	
`>>> np.hsplit(a,3)` `[array([1]), array([2]), array([3])]`	Parte a matriz horizontalmente no terceiro índice

`>>> np.vsplit(c,2)`
`[array([[[1.5,2.,1.],`
`[4.,5.,6.]]]),`
`array([[[3.,2.,3.],`
`[4.,5.,6.]]])]`
	Parte a matriz horizontalmente no segundo índice