Laboratório de Sistemas Inteligentes
Escola Superior de Tecnologia
Universidade do Estado do Amazonas

03 de Outubro de 2020

Programação para Ciência dos Dados

Biblioteca Numpy

Elloa B. Guedes, **Tiago de Melo**
{ebgcosta, tmelo@uea.edu.br}@uea.edu.br
Pós-Graduação *Lato Sensu* em Ciência dos Dados

Outline

1. Introdução
2. Aquecimento
3. Biblioteca `numpy`
4. Referências

Introdução

Linguagem de Programação Python (Versão 3.0+)

- "*Gaste seu tempo com a resolução de seus problemas, não com a linguagem de programação que você utiliza*"
- Zen do Python
- Características:

Introdução

Linguagem de Programação Python (Versão 3.0+)

- "*Gaste seu tempo com a resolução de seus problemas, não com a linguagem de programação que você utiliza*"
- Zen do Python
- Características:
  - Alto-nível
  - Interpretada
  - Executa em máquina virtual
  - Shell Python
  - Portável, resumida, rápida, flexível, *open-source*

Introdução

- TIOBE Index: indicador da popularidade de linguagens de programação
- `https://www.tiobe.com/tiobe-index/`

Sep 2020	Sep 2019	Change	Programming Language	Ratings	Change
1	2	^	C	15.95%	+0.74%
2	**1**	**v**	**Java**	**13.48%**	**-3.18%**
3	3		Python	10.47%	+0.59%
4	4		C++	7.11%	+1.48%
5	5		C#	4.58%	+1.18%
6	6		Visual Basic	4.12%	+0.83%
7	7		JavaScript	2.54%	+0.41%
8	9	^	PHP	2.49%	+0.62%
9	19	^^	R	2.37%	+1.33%

Introdução

- Visão índice TIOBE ao longo do tempo

Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	20	20	-	-
C++	4	3	3	3	2	1	2	9
C#	5	5	5	7	9	-	-	-

Outline

1. Introdução
2. Aquecimento
3. Biblioteca `numpy`
4. Referências

Warm Up

- Atividade *hands-on* para prática de habilidades
- `http://github.com/elloa/numpy`
  - Fork no repositório
  - Clone em sua máquina local
  - Pré-requisitos: numpy, pandas, jupyter
  - Contêiner Docker com uma versão minimalista dos requisitos
- Notebook: **Warm Up - Manipulando Arquivos e Listas em Python**
- Desenvolver as atividades individualmente ou em duplas
- Não usar pandas DataFrame
- Tempo estimado: 1h (plantão de dúvidas + discussão dos resultados)

Outline

1. Introdução
2. Aquecimento
3. Biblioteca `numpy`
4. Referências

Biblioteca `numpy`

- Rápida e versátil
- Performance similar à de linguagens compiladas como C e Fortran, mas com sintaxe de alto-nível do Python
- Suporte à computação distribuída e GPUs
- *Open-source*
- `https://numpy.org/`

Biblioteca `numpy` – Histórico

- Demanda: realização de cálculos numéricos para a comunidade de Computação Científica
- 1995: `Numeric` + `Numarray` (1995)
- 1996: Versão 1.0
- Versão atual: 1.19.0

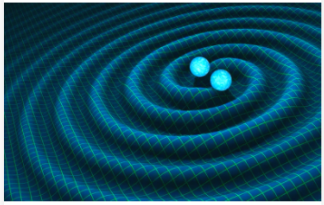
Biblioteca numpy – Aplicações

CASE STUDIES

FIRST IMAGE OF A BLACK HOLE

How NumPy, together with libraries like SciPy and Matplotlib that depend on NumPy, enabled the Event Horizon Telescope to produce the first ever image of a black hole

DETECTION OF GRAVITATIONAL WAVES



In 1916, Albert Einstein predicted gravitational waves; 100 years later their existence was confirmed by LIGO scientists using NumPy.

SPORTS ANALYTICS

Cricket Analytics is changing the game by improving player and team performance through statistical modelling and predictive analytics. NumPy enables many of these analyses.

Biblioteca `numpy` – Conceitos Elementares

- `ndarray`: array *n*-dimensional, homogêneo e de tamanho fixo
- Criação de ndarrays a partir de listas
- Conceitos: `type`, `dtype`, `shape`, `ndim`

Biblioteca `numpy` – Conceitos Elementares

- `ndarray`: array *n*-dimensional, homogêneo e de tamanho fixo
- Criação de ndarrays a partir de listas
- Conceitos: `type`, `dtype`, `shape`, `ndim`
- Quais as diferenças com relação às listas em Python?
- Justifique: é possível criar um ndarray a partir da lista a seguir?

$$lista = [[1,2],[3,4,5],[6,7,8,9]]$$

Biblioteca numpy – Criação de arrays

```
import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

Biblioteca `numpy` – Criação de arrays

- Criação de `ndarrays`:
  - Listas
  - Listas aninhadas
  - Tuplas
  - Tuplas aninhadas
  - Tuplas e listas aninhadas
- Principal preocupação: consistência na dimensão
- Promoção de tipos
- Tipos dependem do sistema

Biblioteca `numpy` – Tipos de Dados

Table 3-1. *Data Types Supported by NumPy*

Data Type	Description
`bool_`	Boolean (true or false) stored as a byte
`int_`	Default integer type (same as C long; normally either `int64` or `int32`)
`intc`	Identical to C int (normally `int32` or `int64`)
`intp`	Integer used for indexing (same as C `size_t`; normally either `int32` or `int64`)
`int8`	Byte (–128 to 127)
`int16`	Integer (–32768 to 32767)
`int32`	Integer (–2147483648 to 2147483647)
`int64`	Integer (–9223372036854775808 to 9223372036854775807)
`uint8`	Unsigned integer (0 to 255)
`uint16`	Unsigned integer (0 to 65535)
`uint32`	Unsigned integer (0 to 4294967295)
`uint64`	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`
`float16`	Half precision float: sign bit, 5-bit exponent, 10-bit mantissa
`float32`	Single precision float: sign bit, 8-bit exponent, 23-bit mantissa
`float64`	Double precision float: sign bit, 11-bit exponent, 52-bit mantissa
`complex_`	Shorthand for complex128
`complex64`	Complex number, represented by two 32-bit floats (real and imaginary components)
`complex128`	Complex number, represented by two 64-bit floats (real and imaginary components)

Biblioteca numpy – Criação de arays

- Criação intrínseca de arrays

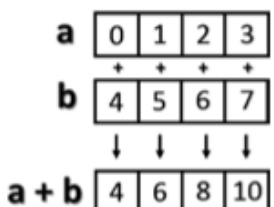
- `np.zeros(shape)`
- `np.ones(shape)`
- `np.random.*`
- `np.arange(<similar ao range>)`
- `np.linspace(inicio,fim, quantidade)`

Biblioteca numpy – Criação de arrays

```
import numpy as np

a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7)  # Create a constant array
print(c)               # Prints "[[ 7.  7.]
                       #          [ 7.  7.]]"

d = np.eye(2)         # Create a 2x2 identity matrix
print(d)              # Prints "[[ 1.  0.]
                      #          [ 0.  1.]]"

e = np.random.random((2,2))  # Create an array filled with random values
print(e)                     # Might print "[[ 0.91940167  0.08143941]
                             #               [ 0.68744134  0.87236687]]"
```

Biblioteca `numpy` – Operações Elemento-a-Elemento

- Operações elemento-a-elemento (*element-wise*)
- Soma, multiplicação, subtração, divisão e exponenciação por escalar
- Soma, multiplicação, subtração e divisão com outro vetor de igual *shape*
- Incremento e Decremento (unários)

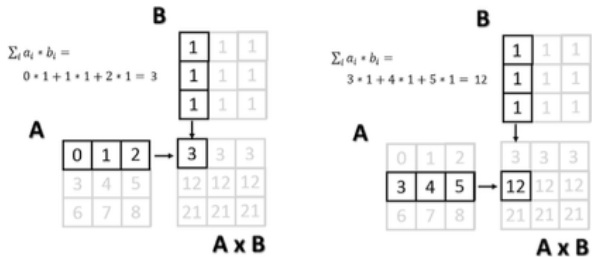


Biblioteca `numpy` – Operações Elemento-a-Elemento

- Cuidado com a multiplicação!!
- O asterisco indica a multiplicação elemento-a-elemento

Biblioteca numpy – Operações Elemento-a-Elemento

- Cuidado com a multiplicação!!
- O asterisco indica a multiplicação elemento-a-elemento
- E a multiplicação de matrizes que aprendi em álgebra? **np.dot**



Biblioteca numpy – Operações Elemento-a-Elemento

```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

bool_idx = (a > 2)   # Find the elements of a that are bigger than 2;
                     # this returns a numpy array of Booleans of the same
                     # shape as a, where each slot of bool_idx tells
                     # whether that element of a is > 2.

print(bool_idx)      # Prints "[[False False]
                     #          [ True  True]
                     #          [ True  True]]"

# We use boolean array indexing to construct a rank 1 array
# consisting of the elements of a corresponding to the True values
# of bool_idx
print(a[bool_idx])  # Prints "[3 4 5 6]"

# We can do all of the above in a single concise statement:
print(a[a > 2])     # Prints "[3 4 5 6]"
```

Biblioteca `numpy` – Funções

- Funções universais podem ser aplicadas em todos os elementos do ndarray
- Operações *bult-in*:
  - np.sin, np.cos, np.exp, np.sqrt, np.log, ...
- `ufunc`: Qualquer função que opere de modo elemento a elemento

Biblioteca `numpy` – Funções Agregadoras

- Funções agregadas produzem um único resultado a partir de todo o array
- Muito úteis para exploração de dados
- sum, min, max, mean, std
- Amplamente usadas no pandas!

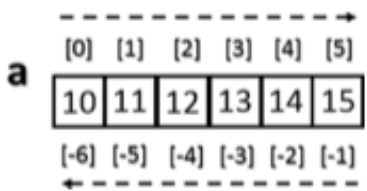
Biblioteca numpy – Funções Agregadoras

```
import numpy as np

x = np.array([[1,2],[3,4]])

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"
```

Biblioteca numpy – Indexação, Fatiamento e Iteração

- Similar às operações com listas e uso do range
- Múltipla indexação

- Cuidado: Visão do array ou cópia



Biblioteca numpy – Indexação, Fatiamento e Iteração

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print(a[0, 1])   # Prints "2"
b[0, 0] = 77     # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])   # Prints "77"
```

Biblioteca numpy – Indexação, Fatiamento e Iteração

```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

# An example of integer array indexing.
# The returned array will have shape (3,) and
print(a[[0, 1, 2], [0, 1, 0]])  # Prints "[1 4 5]"

# The above example of integer array indexing is equivalent to this:
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))  # Prints "[1 4 5]"

# When using integer array indexing, you can reuse the same
# element from the source array:
print(a[[0, 0], [1, 1]])  # Prints "[2 2]"

# Equivalent to the previous integer array indexing example
print(np.array([a[0, 1], a[0, 1]]))  # Prints "[2 2]"
```

Biblioteca `numpy` – Boleanos e Condicionais

- Permitem checar quais elementos de um ndarray satisfazem certas condições
- Facilitam a seleção de partes do array
- Base para implementar “consultas” do pandas
- Indexação Condicional

Biblioteca numpy – Booleanos e Condicionais

```
a = np.random.randint(low = 1, high= 100, size=16).reshape(4,4)
a
```

```
array([[40, 84, 99, 70],
       [62,  2, 16, 25],
       [92, 37, 10, 52],
       [90, 47, 27,  1]])
```

```
a < 50
```

```
array([[ True, False, False, False],
       [False,  True,  True,  True],
       [False,  True,  True, False],
       [False,  True,  True,  True]])
```

```
a[a<50]
```

```
array([40,  2, 16, 25, 37, 10, 47, 27,  1])
```

Biblioteca numpy – Booleanos e Condicionais

```
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

bool_idx = (a > 2)   # Find the elements of a that are bigger than 2;
                     # this returns a numpy array of Booleans of the same
                     # shape as a, where each slot of bool_idx tells
                     # whether that element of a is > 2.

print(bool_idx)      # Prints "[[False False]
                     #          [ True  True]
                     #          [ True  True]]"

# We use boolean array indexing to construct a rank 1 array
# consisting of the elements of a corresponding to the True values
# of bool_idx
print(a[bool_idx])  # Prints "[3 4 5 6]"

# We can do all of the above in a single concise statement:
print(a[a > 2])     # Prints "[3 4 5 6]"
```

Biblioteca `numpy` – Manipulação de Dimensões

- reshape
- ravel, flatten
- transpose

Biblioteca `numpy` – Broadcasting

- *Broadcasting* permite que um operador ou função atue em um ou mais arrays mesmo quando eles não possuem o mesmo *shape*

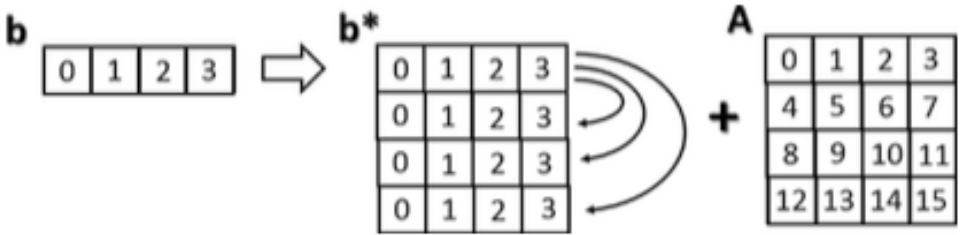
- Dois arrays estão sujeitos ao broadcasting quando todas as suas dimensões são compatíveis
  - O comprimento de cada dimensão é igual
  - Ou um dos comprimentos é igual a 1

- Regras:
  1. Regra 1: Adicionar 1 a cada dimensão faltante
  2. Regra 2: A dimensão faltante é preenchida com réplicas dos valores pré-existentes

Biblioteca numpy – Broadcasting

```
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)   # Create an empty matrix with the same shape as x

# Add the vector v to each row of the matrix x with an explicit loop
for i in range(4):
    y[i, :] = x[i, :] + v

# Now y is the following
# [[ 2  2  4]
#  [ 5  5  7]
#  [ 8  8 10]
#  [11 11 13]]
print(y)
```

Biblioteca numpy – Broadcasting



Biblioteca `numpy` – Arquivos

- Possibilidade de leitura e escrita em arquivos
- Arquivos binários ou textuais
- `np.save` e `np.load`
- Prática com imagens!

Outline

1. Introdução
2. Aquecimento
3. Biblioteca `numpy`
4. Referências

Referências

1. `https://cs231n.github.io/python-numpy-tutorial/`
2. `https://numpy.org/doc/stable/user/tutorial-svd.html`
3. `https://numpy.org/`
4. NELI, F. (2018) **Python Data Analytics**. Editora Apress, 1a. edição. Estados Unidos. Capítulo 3 – *The Numpy Library*

03 de Outubro de 2020

Programação para Ciência dos Dados

Biblioteca Numpy

Elloa B. Guedes, **Tiago de Melo**
{ebgcosta, tmelo@uea.edu.br}@uea.edu.br
Pós-Graduação *Lato Sensu* em Ciência dos Dados