Основы NumPy

Технологии и языки программирования

Юдинцев В. В.

Кафедра теоретической механики

29 марта 2019 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

NumPy

- NumPy базовая библиотека для научных вычислений в среде Python, предлагающая поддержку многомерных массивов, матриц и эффективных функция для работы с этими типами данных.
- Быстродействие кода Python с использованием NumPy в 50 раз быстрее кода на “чистом” Python[1] и сравнимо с быстродействием коммерческого пакета матричной алгебры MATLAB.

[1] http://scipy.github.io/old-wiki/pages/PerformancePython

Импорт модуля

Вариант 1

```
import numpy
v = numpy.array([1, 2])
```

Вариант 2 (рекомендуется)

```
import numpy as np
v = np.array([1, 2])
```

Вариант 3

```
from numpy import *
v = array([1, 2])
```

array

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

Свойства

- Размерность массива

```
1 >>> a.ndim
2 2
```

- Размеры (по каждому измерению)

```
1 >>> a.shape
2 (2, 3)
```

- Количество элементов (суммарное)

```
1 >>> a.size
2 6
```

Тип данных массива

```
1 import numpy as np
2 a = np.array([[1, 2, 3], [4, 5, 6]])
3 print(a.dtype)
```

```
int64
```

```
1 a = np.array([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
2 print(a.dtype)
```

```
float64
```

Функции для создания массивов

Генератор последовательностей

arange(start, stop, step, dtype)

- start: начальное значение
- stop: конечное значение (не включается в результат)
- step: шаг
- dtype: тип данных

От минус 1.0 до 1.0 с шагом 0.2:

```
1 v = np.arange(-1.0, 1.0, 0.2)
2 v
```

```
array([ -1.00000000e+00,  -8.00000000e-01,  -6.00000000e-01,
        -4.00000000e-01,  -2.00000000e-01,  -2.22044605e-16,
         2.00000000e-01,   4.00000000e-01,   6.00000000e-01,
         8.00000000e-01])
```

Массив нулевых значений

zeros(shape, dtype = float, order = 'C')

- shape: размерность массива
- dtype: тип элементов массива
- order: порядок хранения элементов

```
>>> np.zeros(5)
array([  0.,   0.,   0.,   0.,   0.])
```

Матрица-столбец нулевых значений

```
>>> np.zeros((2,1))
array([[  0.],
       [  0.]])
```

Массив единиц

ones(shape, dtype = float, order = 'C')

- shape: размерность массива
- dtype: тип элементов массива
- order: порядок хранения элементов

```
>>> np.ones(5)
array([ 1.,  1.,  1.,  1.,  1.])
```

Матрица единиц

```
>>> np.ones( (3, 2) )
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])
```

Сетка на заданном интервале

linspace(start, stop, num=50, endpoint=True, retstep=False, ...)

- start: начальное значение
- stop: конечное значение
- num: количество элементов
- endpoint: если True, то последний элемент (end) включается в результат
- retstep: возвращать и вычисленное значение шага

```
>>> np.linspace(0.0, 2.0, 4)
array([ 0.  , 0.66666667, 1.33333333, 2.  ])
```

Матрица единиц

```
>>> np.linspace(0.0, 2.0, 4, retstep = True)
(array([ 0.  , 0.66667, 1.3333, 2.  ]), 0.66666)
```

Функция от индексов массива

```
1 a = np.fromfunction(lambda i, j: i == j, (3, 3))
2 print(a)
```

```
[[ True, False, False],
 [False,  True, False],
 [False, False,  True]]
```

Копирование

Операция “присвоения” создает новую ссылку (псевдоним) на объект в памяти:

```
1 a = np.array( [ [1, 2], [3, 4] ])
2 b = a
3 print(b is a)
```

```
True
```

Для создания копии массива используется метод copy():

```
3 b = a.copy()
4 print(b is a)
```

```
False
```

Основные операции

Изменение размерности

Функция reshape изменяет размерность массива, не меняя сами данные (новый “взгляд” на массив)

```
a = np.arange(6)
print(a)
```

```
[0 1 2 3 4 5]
```

```
b = np.reshape( a, (3, 2) )
print(b)
b[0,0] = 9
print(a)
```

```
[[0 1]
 [2 3]
 [4 5]]
[9 1 2 3 4 5]
```

Изменение размерности

Одна из размерностей может быть равна “-1”. В этом случае эта размерность вычисляется:

```
a = np.reshape( np.arange(6), (3, -1) )
print(a)
```

```
[[0  1]
 [2  3]
 [4  5]]
```

Плоский список

Преобразование многомерного массива в “плоский” список (стиль Си: order=’C’ по умолчанию):

```
3 b = np.reshape(a,-1)
4 print(b)
```

```
[0, 1, 2, 3, 4, 5]
```

Плоский список

Преобразование многомерного массива в “плоский” список (стиль Фортран: order=’F’)

```
3 b = np.reshape(a, -1 , order='F')
4 print(b)
```

```
[0, 2, 4, 1, 3, 5]
```

Формирование плоского списка: ravel

Преобразование многомерного массива в “плоский” список (стиль Си: order=’C’ по умолчанию):

```
a = np.reshape( np.arange(6), (3, -1) )
print(a)
```

```
[[0  1]
 [2  3]
 [4  5]]
```

```
b = np.ravel(a)
print(b)
```

```
[0  1  2  3  4  5]
```

```
b = np.ravel(a, order='F')
print(b)
```

```
[0  2  4  1  3  5]
```

vstack: склейка строк

```
1 a = np.array( [[1, 2],[3, 4]] )
2 b = np.array( [[5, 6],[7, 8]] )
3 c = np.vstack((a,b))
4 print(c)
```

```
[[1  2]
 [3  4]
 [5  6]
 [7  8]]
```

hstack: склейка столбцов

```
1 a = np.array( [[1, 2],[3, 4]] )
2 b = np.array( [[5, 6],[7, 8]] )
3 c = np.hstack((a,b))
4 print(c)
```

```
[[1 2 5 6]
 [3 4 7 8]]
```

hsplit: разделение на части по горизонтали

```
1 a = np.floor( 10 * np.random.random( (2,12) ) )
2 print(a)
```

```
[[ 6.  9.  2.  8.  6.  8.]
 [ 2.  1.  6.  8.  8.  6.]]
```

```
3 np.hsplit(a,3)
```

```
[array([[ 6.,  9.],
        [ 2.,  1.]]),
 array([[ 2.,  8.],
        [ 6.,  8.]]),
 array([[ 6.,  8.],
        [ 8.,  6.]])]
```

vsplit: разделение на части вертикали

```
1 a = np.floor( 10*np.random.random( (4,2) ) )
2 print(a)
```

```
[[ 5.  6.]
 [ 1.  5.]
 [ 5.  6.]
 [ 4.  3.]]
```

```
3 np.vsplit(a,2)
```

```
[array([[ 5.,  6.],
        [ 1.,  5.]]),
 array([[ 5.,  6.],
        [ 4.,  3.]])]
```

Функции

Арифметические операции

Арифметические операции выполняются поэлементно

```
a = np.array( [[1, 2],[3, 4]] )
b = np.array( [[5, 6],[7, 8]] )
c = a + b
print(c)
```

```
[[ 6  8]
 [10 12]]
```

```
c = 2*(a + b)
print(c)
```

```
[[12 16]
 [20 24]]
```

Математические функции

```
1 a = np.reshape(np.arange(6), (2, -1) )
2 print(a)
```

```
[[0 1 2]
 [3 4 5]]
```

Возведение в степень:

```
3 print( np.power(a,2) )
```

```
[[ 0,  1,  4],
 [ 9, 16, 25]]
```

```
3 print( np.power(a,a) )
```

```
[[   1    1    4]
 [  27  256 3125]]
```

Математические функции

```
1 a = np.reshape(np.arange(6), (2, -1) )
2 print(a)
```

```
[[0 1 2]
 [3 4 5]]
```

```
3 print( np.exp(a) )
```

```
[[   1.            2.71828183    7.3890561 ]
 [  20.08553692   54.59815003  148.4131591 ]]
```

```
3 print( a/(a+1) )
```

```
[[ 0.          0.5         0.66666667]
 [ 0.75        0.8         0.83333333]]
```

dot: скалярное произведение

```
1 a = np.array( [[1, 2],[3, 4]] )
2 b = np.array( [[5, 6],[7, 8]] )
3
4 c = np.dot(a,b)
5
6 print(c)
```

```
[[19 22]
 [43 50]]
```

Индексы и срезы

Индексация

```
1 a = np.arange(8)**2
2 print(a)
```

```
[ 0  1  4  9 16 25 36 49]
```

Индексация начинается с нуля. Третий элемент массива имеет индекс “2”:

```
2 print(a[2])
```

```
4
```

Срезы

```
1 a = np.arange(8)**2
2 print(a)
```

```
[ 0  1  4  9 16 25 36 49]
```

[Начальное значение: граница: шаг]

```
3 print(a[2:6:2])
```

```
4, 16
```

Многомерные массивы

```
a = np.reshape(np.arange(8),(2,-1))
print(a)
```

```
[[0 1 2 3]
 [4 5 6 7]]
```

```
print( a[0,1] )
```

```
1
```

```
print( a[:,1] )
```

```
[1 5]
```

```
print( a[:, 1:3] )
```

```
[[1 2]
 [5 6]]
```

Многомерные массивы

```
a = np.reshape(np.arange(8),(2,-1))
print(a)
```

```
[[0  1  2  3]
 [4  5  6  7]]
```

Последняя строка

```
print( a[-1,:] )
```

```
[4  5  6  7]
```

Второй столбец с конца

```
print( a[:,-2] )
```

```
[2  6]
```

Использование ... вместо :

```
1 a = np.array( [ [[  0,  1,  2],
2                  [ 10, 12, 13]],
3                 [[100,101,102],
4                  [110,112,113]]])
5 print(a.shape)
```

```
(2, 2, 3)
```

```
3 print( a[1 ,...] )
```

```
[[  0,  1,  2],
 [ 10, 12, 13]]
```

```
4 print( a[...,2] )
```

```
[[  2  13]
 [102 113]]
```

Массив, как итератор

При использовании массивов в конструкциях типа for ... in, итерация выполняется только по первой размерности:

```
a = np.array( [ [[  0,  1,  2],
                 [ 10, 12, 13]],
                [[100,101,102],
                 [110,112,113]]])

for row in a:
    print('Element: ',row)
```

```
Element: [[ 0  1  2]
 [10 12 13]]
Element: [[100 101 102]
 [110 112 113]]
```

Массив, как итератор

Итерация по всем элементам (последовательно по каждой размерности):

```
a = np.array( [ [[  0,  1,  2], [ 10, 12, 13]],
                [[100,101,102], [110,112,113]]])
for i in np.ravel(a):
    print('Element', i)
```

```
Element 0
Element 1
Element 2
Element 10
...
Element 101
Element 102
Element 110
Element 112
Element 113
```

Индексация при помощи массива индексов

```
1 a = np.arange(6)*2
2 i = np.array ([1, 3, 5, 2])
3
4 print(a[i])
```

```
[ 2  6 10  4]
```

```
5 i = np.array ([ [1, 3], [5, 2] ])
6
7 print(a[i])
```

```
[[ 2  6]
 [10  4]]
```

Индексация при помощи массива типа bool

```
1 a = np.arange(6)*2
2 print(a)
```

```
[ 0  2  4  6  8 10]
```

Булев массив (каждый элемент сравнивается с 4):

```
3 print( a>4 )
```

```
[False False False  True  True  True]
```

Использование массива для извлечения элементов:

```
4 print( a[a>4] )
```

```
[ 6  8 10]
```

Статистические функции

min, max, sum

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

Минимальное значение всего массива

```
print( a.min() )
```

```
1
```

Минимальные значения столбцов

```
print( a.min(axis=0) )
```

```
[1 2]
```

Минимальные значения строк

```
print( a.min(axis=1) )
```

```
[1 3]
```

min, max, sum

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

Сумма всех значений

```
2 print( a.sum() )
```

```
10
```

Сумма строк

```
3 print( a.sum(axis=0) )
```

```
[4 6]
```

Сумма столбцов

```
4 print( a.sum(axis=1) )
```

```
[3 7]
```

Среднее значение

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

Среднее значение всех элементов

```
2 print( a.mean() )
```

```
2.5
```

Среднее по первой размерности (в столбцах)

```
3 print( a.mean(axis=0) )
```

```
[2. 3.]
```

Среднее по второй размерности (в строках)

```
4 print( a.mean(axis=1) )
```

```
[ 1.5  3.5]
```

Многочлены

Коэффициенты $\Longleftrightarrow$ корни

Многочлен

$$x^4 - 11x^3 + 9x^2 + 11x - 10$$

Корни

$$x_1 = -1,\ x_2 = 1,\ x_3 = 1,\ x_4 = 10$$

np.poly

```
1 np.poly([-1, 1, 1, 10])
2
3 array([  1, -11,   9,  11, -10])
```

np.roots

```
1 np.roots([  1, -11,   9,  11, -10])
2
3 [10.+0.0000000e+00j  -1.+0.0000000e+00j
4    1.+9.6357437e-09j   1.-9.6357437e-09j ]
```

Интегрирование и дифференцирование

Интегрирование

$$\int (x^3 + x^2 + x + 1)\,dx = \frac{x^4}{4} + \frac{x^3}{3} + \frac{x^2}{2} + x + C$$

```
np.polyint([1, 1, 1, 1])
array([ 0.25, 0.33333333, 0.5, 1., 0. ])
```

Дифференцирование

```
np.polyder([1./4., 1./3., 1./2., 1., 0.])
array([ 1., 1., 1., 1.])
```

Аппроксимация

Таблица значений

```
x    = [1, 2, 3, 4, 5, 6, 7, 8]
y    = [0, 2, 1, 3, 7, 10, 11, 19]
```

Аппроксимация полиномом второй степени

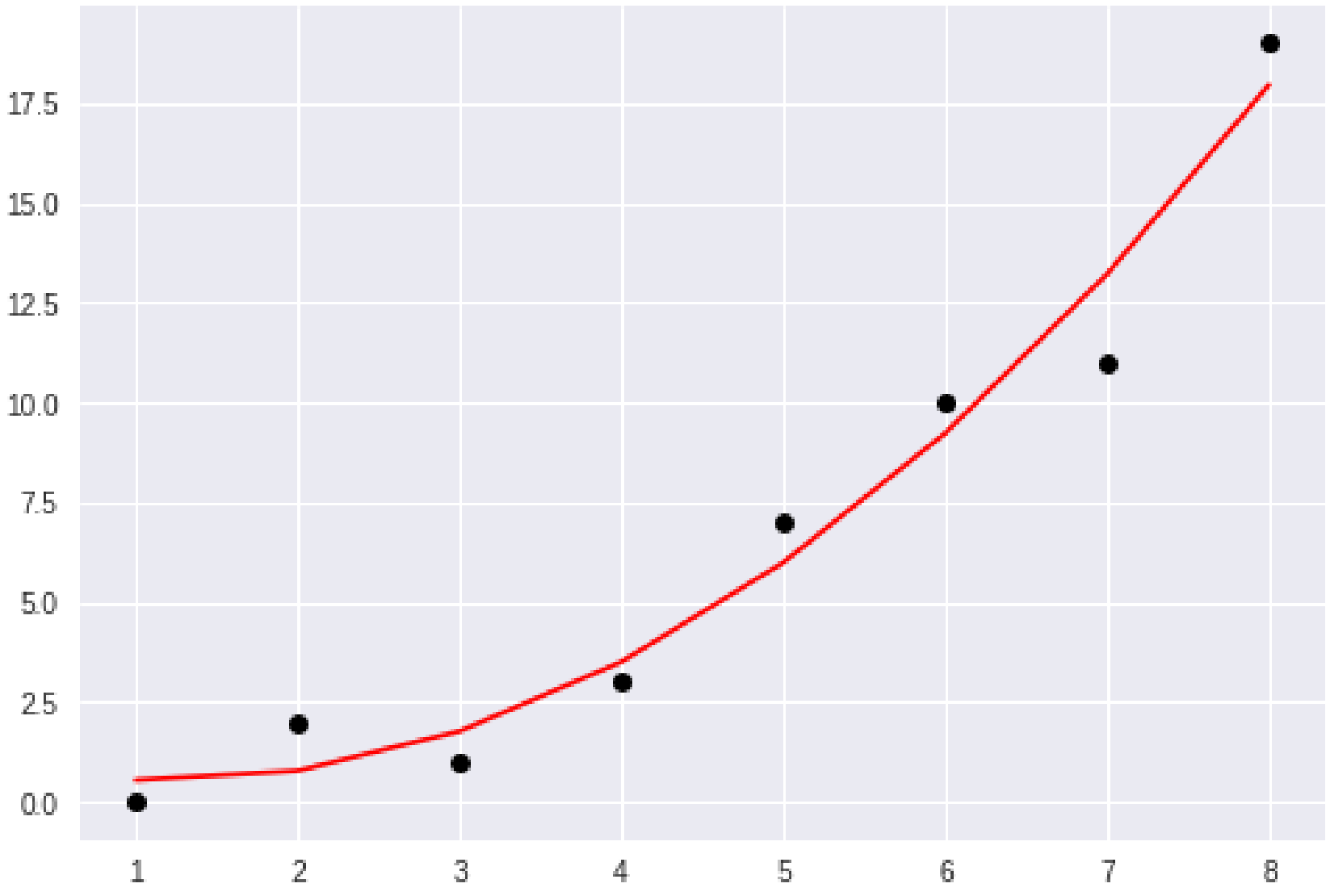
```
fit = np.polyfit(x, y, 2)
array([ 0.375  , -0.88690476, 1.05357143])
```

Значения аппроксимирующего полинома

```
yf = np.array([ np.polyval(fit , xi) for xi in x ])

array([ 0.54166667,  0.7797619 ,  1.76785714,
        3.50595238,  5.99404762,  9.23214286,
        13.2202381,  17.95833333])
```

Аппроксимация



Линейная алгебра

Скалярное произведение

Числа

```
np.dot(3, 4)
```

Векторы

```
np.dot([2, 3], [2, 3])
13
```

Матрицы

```
a = [[1, 0],
     [0, 1]]
b = [[4, 1],
     [2, 2]]

np.dot(a, b)
array([[4, 1],
       [2, 2]])
```

Норма

```
from numpy import linalg as LA

a = np.arange(9) - 4

array([-4, -3, -2, -1,  0,  1,  2,  3,  4])
```

$$|a| = \sqrt{\sum_i a_i^2}$$

```
LA.norm(a)
7.745966692414834
```

Это эквивалентно

```
np.sqrt(np.sum(a*a))
```

Норма

```
b = a.reshape((3, 3))
array([[-4, -3, -2],
       [-1,  0,  1],
       [ 2,  3,  4]])
```

$$|b|_F = \sqrt{\sum_i \sum_j |b_{ij}|^2}$$

```
LA.norm(a)
7.745966692414834
```

$$|b|_\infty = \max_i \sum_j |b_{ij}|$$

```
LA.norm(b, np.inf)
9
```

Решение СЛУ

$$\begin{cases} 3x_1 + x_2 = 9 \\ x_1 + 2x_2 = 8 \end{cases}$$

```
a = np.array([[3,1], [1,2]])
b = np.array([9,8])
x = np.linalg.solve(a, b)
```

```
x
array([ 2.,  3.])
```

Файловые функции

Чтение массива из текстового файла

File1.txt

```
# Results
1 23 5
2 65 6
4 55 4
```

Прочитать значения из файла File1.txt в массив res

```
1 res = np.loadtxt("file1.txt", delimiter=" ")
2 print(res)
```

```
[[  1.  23.   5.]
 [  2.  65.   6.]
 [  4.  55.   4.]]
```

Чтение массива из файла: usecols

File1.txt

```
# Results
1 23 5
2 65 6
4 55 4
```

Прочитать значения из файла File1.txt в массив res столбцы с индексами 1 и 2:

```
1 res = np.loadtxt("file1.txt", usecols = (1, 2) )
2 print(res)
```

```
[[ 23.   5.]
 [ 65.   6.]
 [ 55.   4.]]
```

Чтение массива из файла: skiprows

File1.txt

```
# Results
1 23 5
2 65 6
4 55 4
```

Прочитать значения из файла File1.txt в массив res, пропустив первые две строки

```
1 res = np.loadtxt("file1.txt", skiprows = 2 )
2 print(res)
```

```
[[  2.  65.   6.]
 [  4.  55.   4.]]
```

Запись массива в текстовый файл

```
x = np.linspace(0, np.pi, 5).reshape((5,1))

table = np.hstack( (x, np.sin(x)) )

np.savetxt('sin.txt', table, delimiter=',')
```

Содержимое файла sin.txt

```
0.0000000000000000e+00,0.0000000000000000e+00
7.8539816339744827e-01,7.0710678118654746e-01
1.5707963267948966e+00,1.0000000000000000e+00
2.3561944901923448e+00,7.0710678118654757e-01
3.1415926535897931e+00,1.2246467991473532e-16
```

Форматирование вывода

```
x = np.linspace(0, np.pi, 5).reshape((5,1))

table = np.hstack( (x, np.sin(x)) )

np.savetxt('sin.txt', table, fmt='%7.4f')
```

Содержимое файла sin.txt

```
 0.0000  0.0000
 0.7854  0.7071
 1.5708  1.0000
 2.3562  0.7071
 3.1416  0.0000
```

Форматирование вывода

```
x = np.linspace(0, np.pi, 5).reshape((5,1))

table = np.hstack( (x, np.sin(x)) )

np.savetxt('sin.txt', table, fmt = '%9.4g',
                             header = '--data start--',
                             footer = '-- data end --'))
```

Содержимое файла sin.txt

```
# --data start--
        0           0
   0.7854      0.7071
    1.571           1
    2.356      0.7071
    3.142 1.225e-16
# -- data end --
```

Справка

1. Quickstart tutorial

   `https://docs.scipy.org/doc/numpy-dev/user/quickstart.html`

2. Numpy User Guide

   `https://docs.scipy.org/doc/numpy/user/`

3. Numpy Reference Guide

   `https://docs.scipy.org/doc/numpy/reference/`

4. NumPy в Python. Часть 4

   `https://habr.com/ru/post/415373/`