NumPy 基础

金　林
中南财经政法大学统计系
jinlin82@qq.com

2020 年 2 月 8 日

Facts

1. Initial release: As Numeric, 1995; as NumPy, 2006
2. Stable release: 1.11.2 / 3 October 2016;
3. Website: `http://www.numpy.org`
4. History: `https://en.wikipedia.org/wiki/NumPy`

What is NumPy?

1. NumPy is the fundamental package for scientific computing in Python.
2. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including
   1. mathematical,
   2. logical,
   3. shape manipulation,
   4. sorting, selecting, I/O,
   5. discrete Fourier transforms, basic linear algebra,
   6. basic statistical operations, random simulation and much more.

NumPy

1. At the core of the NumPy package, is the ndarray object.
2. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.
3. The points about sequence size and speed are particularly important in scientific computing.

ndarray object

1. Vectorization:Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just "behind the scenes" in optimized, pre-compiled C code.
2. Broadcasting:Broadcasting is the term used to describe the implicit element-by-element behavior of operations.
3. NumPy fully supports an object-oriented approach with ndarray. ndarray is a class, possessing numerous methods and attributes.

- Array Creation
- Basic Operations
- Indexing, Slicing and Iterating

`array` function

1. create an array from a regular Python list or tuple using the `array` function. The type of the resulting array is deduced from the type of the elements in the sequences.
2. array transforms sequences of sequences into two-dimensional arrays, sequences of sequences of sequences into three-dimensional arrays, and so on.
3. Often, the elements of an array are originally unknown, but its size is known.

arrays with initial placeholder content

1. The function `zeros` creates an array full of zeros,
2. the function `ones` creates an array full of ones,
3. the function `empty` creates an array whose initial content is random and depends on the state of the memory.
4. the function `identity` creates the diagonal array,
5. the function `eye` creates an array with ones on the diagonal and zeros elsewhere.

`arange` and `linspace` function

1. `arange` :
2. `linspace` :

attributes of an ndarray object

1. ndarray.ndim
2. ndarray.shape
3. ndarray.size
4. ndarray.dtype
5. ndarray.itemsize
6. ndarray.data

- Array Creation
- Basic Operations
- Indexing, Slicing and Iterating

Basic Operations

1. Arithmetic operators on arrays apply elementwise. A new array is created and filled with the result.
2. The matrix product can be performed using the dot function or method:
3. Many unary operations, such as computing the sum of all the elements in the array, are implemented as methods of the ndarray class.
4. by specifying the axis parameter you can apply an operation along the specified axis of an array(类似于 R 中的 apply 函数)

例子

```
import numpy as np
a=np.arange(4)
b=np.array([2,5,8,9])
a*b

A=np.arange(12).reshape(3,4)
B=np.arange(13,25).reshape(4,3)
np.dot(A, B)
A.dot(B)
A.sum()
A.sum(axis=0)
A.sum(axis=1)

"""END"""
```

Universal Functions

1. NumPy provides familiar mathematical functions such as sin, cos, and exp.
2. In NumPy, these are called "universal functions"(ufunc).
3. Within NumPy, these functions operate elementwise on an array, producing an array as output.

```
A=np.arange(12).reshape(3,4)
np.exp(A)
np.sqrt(A)

"""END"""
```

- Array Creation
- Basic Operations
- Indexing, Slicing and Iterating

Indexing, Slicing and Iterating

1. One-dimensional arrays can be indexed, sliced and iterated over, much like lists and other Python sequences.

```
x=np.arange(12)**2
x[3]
x[2:6]
x[7:]
x[::-1]
x[9:2:-3]

"""END"""
```

Indexing, Slicing and Iterating

2. Multidimensional arrays can have one index per axis. These indices are given in a tuple separated by commas.
3. When fewer indices are provided than the number of axes, the missing indices are considered complete slices.

```
A=np.arange(24).reshape(4,6)
A[2,3]
A[1:3, 2:4]
A[1]
A[:, 2:4]
A[..., 3]

"""END"""
```

Indexing, Slicing and Iterating

4. The dots (...) represent as many colons as needed to produce a complete indexing tuple.
5. For example, if x is an array with 5 axes, then
   - `x[1, 2, ...]` is equivalent to `x[1, 2, :, :, :]`,
   - `x[..., 3]` to `x[:, : ,: ,:, 3]`
   - `x[4, ..., 5, :]` to `x[4, :, :, 5, :]`

Indexing, Slicing and Iterating

6. Iterating over multidimensional arrays is done with respect to the first axis
7. if one wants to perform an operation on each element in the array, one can use the flat attribute which is an iterator over all the elements of the array

```
1  import numpy as np
2  A=np.arange(24).reshape(4,6)
3  for i in A:
4      """打印A的各行"""
5      print(i)
6
7  for i in A.flat:
8      """打印A中的每个元素"""
9      print(i)
10
11 """END"""
```

Changing the shape of an array

1. An array has a shape given by the number of elements along each axis
2. The shape of an array can be changed with various commands.
3. Note that the following three commands all return a modified array, but do not change the original array:
   1. `ndarray.ravel()`, `ndarray.T`, `ndarry.reshape`
4. the `ndarray.resize` method modifies the array itself

```
1  import numpy as np
2  a = np.floor(10 * np.random.random((3,4)))
3  a.shape
4
5
6
7  a.ravel()
8  a.T
9  a.reshape(2,6)
10 a.resize(2,6)
11
12 """END"""
```

Stacking together different arrays

1. `hstack, vstack`
2. `column_stack, row_stack`
3. `concatenate`
4. `c_, r_`

Splitting one array into several smaller ones

1. `hsplit`
2. `vsplit`
3. `array_split`

Indexing with Arrays of Indices-1D

1. 使用 np.array 对象作为索引下标可以取非连续元素

```
a = np.arange(12) ** 2 # the first 12 square numbers
i = np.array( [ 1,1,3,8,5 ] ) # an array of indices
a[i] # the elements of a at the positions i
array([ 1, 1, 9, 64, 25])

j = np.array( [ [ 3, 4], [ 9, 7 ] ] )
a[j]



"""END"""
```

Indexing with Arrays of Indices-2D

2. We can also give indexes for more than one dimension. The arrays of indices for each dimension must have the same shape.
3. Naturally, we can put i and j in a sequence (say a list) and then do the indexing with the list.

```
a = np.arange(12).reshape(3,4)
i = np.array([[0,1], [1,2]])
j = np.array([[2,1], [3,3]])

a[i]
a[i,j]
a[i, 2]
a[:,j]

L = [i,j]
a[L]

"""END"""
```

Indexing with Boolean Arrays

1. use boolean arrays that have the same shape as the original array
2. for each dimension of the array we give a 1D boolean array selecting the slices we want
3. Note that the length of the 1D boolean array must coincide with the length of the dimension (or axis) you want to slice.

```
a = np.arange(12).reshape(3,4)
b=a>4
a[b]
a[b]=0

a = np.arange(12).reshape(3,4)
b1 = np.array([False,True,True])
b2 = np.array([True,False,True,False])
a[b1,:]
a[b1]
a[:,b2]
a[b1,b2]

"""END"""
```

Indexing with strings

1. Structured arrays are ndarrays whose datatype is a composition of simpler datatypes organized as a sequence of named fields.
2. You can access and modify individual fields of a structured array by indexing with the field name.
3. One can index and assign to a structured array with a multi-field index, where the index is a list of field names.

```
x = np.array([('Rex', 9, 81.0), ('Fido', 3, 27.0)],
^^I     dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')
    ])
x['name']
x[['name', 'age']]

"""END"""
```

The ix_() function

1. The ix_ function can be used to combine different vectors so as to obtain the result for each n-uplet.(类似于 R 中的 `expand.grid` 函数)
2. For example, if you want to compute all the a+b*c for all the triplets taken from each of the vectors a, b and c:

```
a = np.array([2,3,4,5])
b = np.array([8,5,4])
c = np.array([5,4,6,8,3])
ax,bx,cx = np.ix_(a,b,c)

result = ax+bx * cx
result

"""END"""
```

简介

1. A universal function (or ufunc for short) is a function that operates on ndarrays in an element-by-element fashion.
2. That is, a ufunc is a "vectorized" wrapper for a function that takes a fixed number of specific inputs and produces a fixed number of specific outputs.
3. In NumPy, universal functions are instances of the numpy.ufunc class.
4. Many of the built-in functions are implemented in compiled C code.

常见通用函数

1. Math operations
2. Trigonometric functions
3. Floating functions
4. ...

Order statistics

函数	功能
amin(a[, axis, out, keepdims])	Return the minimum of an array or mini
amax(a[, axis, out, keepdims])	Return the maximum of an array or max
nanmin(a[, axis, out, keepdims])	Return minimum of an array or minimum
nanmax(a[, axis, out, keepdims])	Return the maximum of an array or max
ptp(a[, axis, out])	Range of values (maximum - minimum)
percentile(a, q[, axis, out, ...])	Compute the qth percentile of the data
nanpercentile(a, q[, axis, out, ...])	Compute the qth percentile of the data

Averages and variances

函数	功能
median(a[, axis, out, overwrite_input, keepdims])	Compute the median al
average(a[, axis, weights, returned])	Compute the weighted a
mean(a[, axis, dtype, out, keepdims])	Compute the arithmetic
std(a[, axis, dtype, out, ddof, keepdims])	Compute the standard d
var(a[, axis, dtype, out, ddof, keepdims])	Compute the variance a
nanmedian(a[, axis, out, overwrite_input, ...])	Compute the median al
nanmean(a[, axis, dtype, out, keepdims])	Compute the arithmetic
nanstd(a[, axis, dtype, out, ddof, keepdims])	Compute the standard d
nanvar(a[, axis, dtype, out, ddof, keepdims])	Compute the variance a

注：几何平均数，调和平均数函数在 scipy 中

Correlating

函数	功能
corrcoef(x[, y, rowvar, bias, ddof])	Return Pearson product-mome
correlate(a, v[, mode])	Cross-correlation of two 1-dime
cov(m[, y, rowvar, bias, ddof, fweights, ...])	Estimate a covariance matrix,

Histograms

常见矩阵运算- Matrix and vector products

函数	功能
dot(a, b)	Dot product of two arrays.
vdot(a, b)	Return the dot product of two vectors.
inner(a, b)	Inner product of two arrays.
outer(a, b)	Compute the outer product of two vectors.
linalg.matrix_power(M, n)	Raise a square matrix to the (integer) power n.
kron(a, b)	Kronecker product of two arrays.

常见矩阵运算- Decompositions

函数	功能
linalg.cholesky(a)	Cholesky decomposition.
linalg.qr(a)	Compute the qr factorization of a matrix.
linalg.svd(a)	Singular Value Decomposition.
linalg.eig(a)	Compute the eigenvalues and right eigenvectors of a square array.

常见矩阵运算- Norms and other numbers

函数	功能
linalg.norm(x)	Matrix or vector norm.
linalg.cond(x)	Compute the condition number of a matrix.
linalg.det(a)	Compute the determinant of an array.
linalg.matrix_rank(M)	Return matrix rank of array using SVD method
trace(a)	Return the sum along diagonals of the array.

常见矩阵运算- Solving equations and inverting matrices

函数	功能
linalg.solve(a, b)	Solve a linear matrix equation, or system of linear scalar equations.
linalg.tensorsolve(a, b)	Solve the tensor equation a x = b for x.
linalg.lstsq(a, b)	Return the least-squares solution to a linear matrix equation.
linalg.inv(a)	Compute the (multiplicative) inverse of a matrix.
linalg.pinv(a)	Compute the (Moore-Penrose) pseudo-inverse of a matrix.

简介

1. 下面这些函数主要用于随机抽样和生成随机数字，关于概率，分位点等计算见 `scipy.stats` 模块
2. 下面函数都以 `np.random.` 开始
3. 有很多功能相同名字不同的函数

Simple random data

函数	功能
rand(d0, d1, ..., dn)	Random values in a given shape.
randn(d0, d1, ..., dn)	Return a sample (or samples) from the "standard normal"
randint(low[, high, size, dtype])	Return random integers from low (inclusive) to high (exclusi
random_integers(low[, high, size])	Random integers of type np.int between low and high, inclus
random_sample([size])	Return random floats in the half-open interval [0.0, 1.0).
random([size])	Return random floats in the half-open interval [0.0, 1.0).
ranf([size])	Return random floats in the half-open interval [0.0, 1.0).
sample([size])	Return random floats in the half-open interval [0.0, 1.0).
choice(a[, size, replace, p])	Generates a random sample from a given 1-D array
bytes(length)	Return random bytes.

Permutations

函数	功能
shuffle(x)	Modify a sequence in-place by shuffling its contents.
permutation(x)	Randomly permute a sequence, or return a permuted range.

Distribution

函数	功能
beta(a, b[, size])	Draw samples from a Beta distribution.
binomial(n, p[, size])	Draw samples from a binomial distribution.
chisquare(df[, size])	Draw samples from a chi-square distribution.
dirichlet(alpha[, size])	Draw samples from the Dirichlet distribution.
exponential([scale, size])	Draw samples from an exponential distribution.
f(dfnum, dfden[, size])	Draw samples from an F distribution.
gamma(shape[, scale, size])	Draw samples from a Gamma distribution.
geometric(p[, size])	Draw samples from the geometric distribution.
gumbel([loc, scale, size])	Draw samples from a Gumbel distribution.
hypergeometric(ngood, nbad, nsample[, size])	Draw samples from a Hypergeometric distributio
laplace([loc, scale, size])	Draw samples from the Laplace or double expone
logistic([loc, scale, size])	Draw samples from a logistic distribution.
lognormal([mean, sigma, size])	Draw samples from a log-normal distribution.
logseries(p[, size])	Draw samples from a logarithmic series distributi

Distributions

函数	功能
multivariate_normal(mean, cov[, size, ...])	Draw random samples from a multivariate normal di
negative_binomial(n, p[, size])	Draw samples from a negative binomial distribution.
noncentral_chisquare(df, nonc[, size])	Draw samples from a noncentral chi-square distribut
noncentral_f(dfnum, dfden, nonc[, size])	Draw samples from the noncentral F distribution.
normal([loc, scale, size])	Draw random samples from a normal (Gaussian) dist
pareto(a[, size])	Draw samples from a Pareto II or Lomax distributior
poisson([lam, size])	Draw samples from a Poisson distribution.
power(a[, size])	Draws samples in [0, 1] from a power distribution wi
rayleigh([scale, size])	Draw samples from a Rayleigh distribution.
standard_cauchy([size])	Draw samples from a standard Cauchy distribution w
standard_exponential([size])	Draw samples from the standard exponential distribu
standard_gamma(shape[, size])	Draw samples from a standard Gamma distribution.
standard_normal([size])	Draw samples from a standard Normal distribution (
standard_t(df[, size])	Draw samples from a standard Student′ s t distribu
triangular(left, mode, right[, size])	Draw samples from the triangular distribution over t
uniform([low, high, size])	Draw samples from a uniform distribution.
vonmises(mu, kappa[, size])	Draw samples from a von Mises distribution.
wald(mean, scale[, size])	Draw samples from a Wald, or inverse Gaussian, dist
weibull(a[, size])	Draw samples from a Weibull distribution.
zipf(a[, size])	Draw samples from a Zipf distribution.

Random generator

函数	功能
RandomState([seed])	Container for the Mersenne Twister pseudo-random number generator.
seed([seed])	Seed the generator.
get_state()	Return a tuple representing the internal state of the generator.
set_state(state)	Set the internal state of the generator from a tuple.

Functions and Methods Overview

1. Array Creation:
   arange, array, copy, empty, empty_like, eye, fromfile, fromfunction,identity, linspace, logspace, mgrid, ogrid, ones, ones_like, r, zeros, zeros_like
2. Conversions:
   ndarray.astype, atleast_1d, atleast_2d, atleast_3d, mat
3. Manipulations:
   array_split, column_stack, concatenate, diagonal, dsplit, dstack, hsplit,hstack, ndarray.item, newaxis, ravel, repeat, reshape, resize, squeeze, swapaxes, take, transpose, vsplit, vstack

Functions and Methods Overview

4. Questions:
   all, any, nonzero, where
5. Ordering:
   argmax, argmin, argsort, max, min, ptp, searchsorted, sort
6. Operations:
   choose, compress, cumprod, cumsum, inner, ndarray.fill, imag, prod, put, putmask,real, sum
7. Basic Statistics:
   cov, mean, std, var
8. Basic Linear Algebra:
   cross, dot, outer, linalg.svd, vdot

Numpy 中的其他常用模块

1. String operations
2. Datetime Support Functions
3. Discrete Fourier Transform (numpy.fft)
4. Financial functions
5. Functional programming
6. Logic functions
7. Mathematical functions
8. Matrix library (numpy.matlib)
9. numpy.polynomial package
10. 具体用法和更多模块可以参考 Numpy reference[numpy-ref.pdf] 中的 Routines 内容