人工智能程序设计

M2 科学计算与数据分析基础

2.2 numpy与科学计算

张 莉

Python中的数组

- 用list和tuple等数据结构表示数组
  - 一维数组 list = [1,2,3,4]
  - 二维数组 list = [[1,2,3],[4,5,6],[7,8,9]]
- array模块
  - 通过array函数创建数组， array.array("B", range(5))
  - 提供append、 insert和read等方法

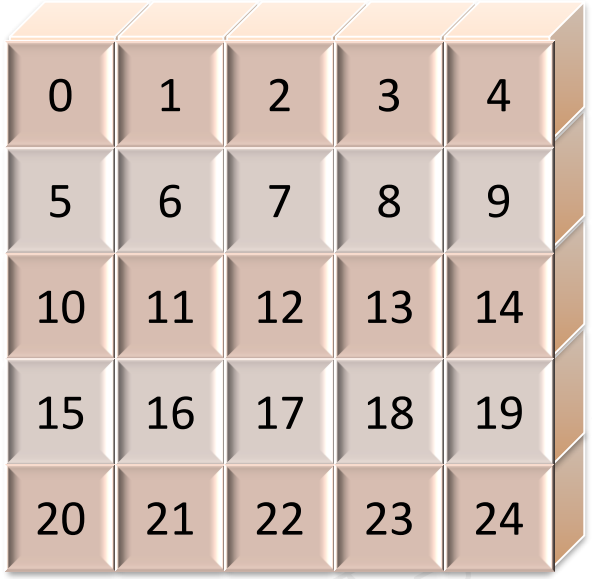
NumPy

1. ndarray的基本特性
2. 创建ndarray
3. ndarray的操作与运算
4. ufunc函数
5. 专门的应用

人工智能程序设计

1 NDARRAY的基本特性

ndarray



- ndarray是什么?

  N维数组

  – NumPy中基本的数据结构

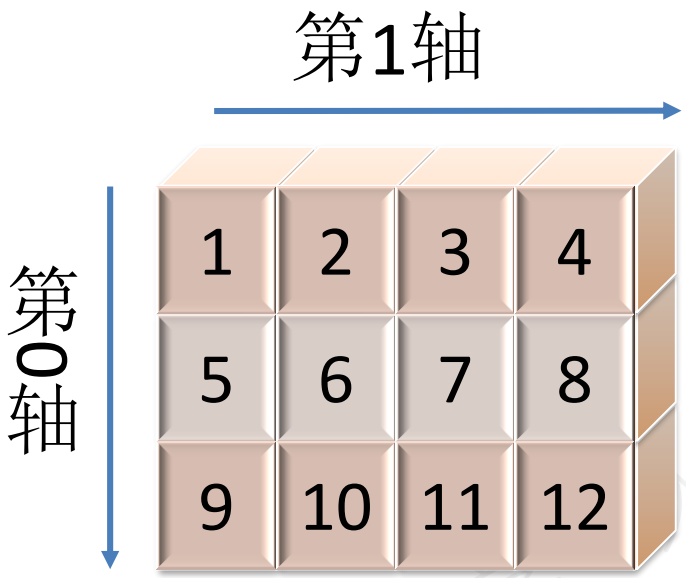
  – 所有元素是同一种类型

  – 别名为array

  – 利于节省内存和CPU计算时间

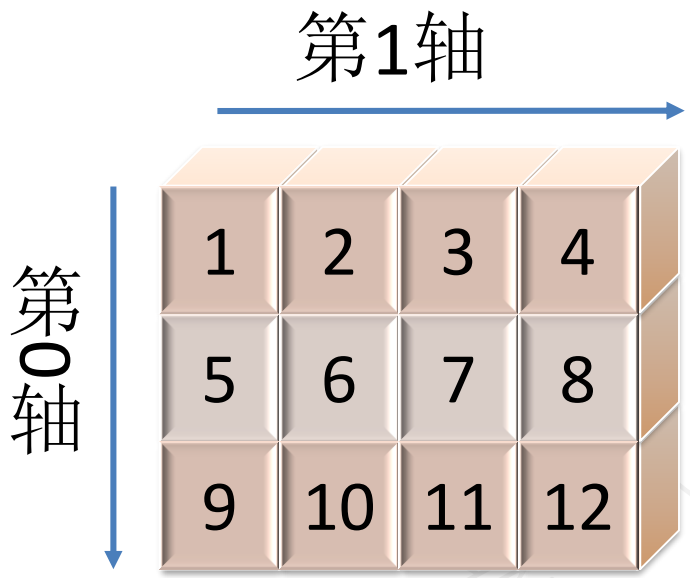
  – 有丰富的函数

ndarray基本概念



- ndarray数组属性
  - 维度(dimensions)称为轴(axes)，轴的个数称为秩(rank)
  - 沿着第0轴和第1轴操作
    - axis = 0（按列）
    - axis = 1（按行）

ndarray基本概念



- ndarray数组属性
  - 基本属性
    - ndarray.ndim（秩）
    - ndarray.shape（维度）
    - ndarray.size（元素总个数）
    - ndarray.dtype（元素类型）
    - ndarray.itemsize（元素字节大小）

人工智能程序设计

2 创建数组

ndarray的创建

array()函数

Source

```
>>> import numpy as np
>>> aArray = np.array([1,2,3])
>>> aArray
array([1, 2, 3])
>>> bArray = np.array([(1,2,3),(4,5,6)] , dtype=float)
>>> bArray
array([[1., 2., 3.],
       [4., 5., 6.]])
>>> bArray.ndim, bArray.shape, bArray.dtype
(2, (2, 3), dtype('float64'))
```

dtype: np.int*,np.uint*, np.float*
astype()方法：类型转换

ndarray的创建

arange	**array**
copy	**empty**
empty_like	**eye**
fromfile	**fromfunction**
full	**identity**
linspace	logspace
mgrid	ogrid
ones	**ones_like**
r	**zeros**
zeros_like	…

ndarray创建函数

ndarray的创建

- zeros()
- ones()
- full()
- zeros_like()
- ones_like()
- full_like()

Source

```
>>> np.zeros((2, 2))
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> np.ones([2, 3])
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> np.full((3, 3), np.pi)
...
>>>  x = np.array([[1, 2, 3], [4, 5, 6]], dtype = np.float32)
>>> np.ones_like(x)
...
```

ndarray的创建

identity()
eye()
diag()

Source

```
>>> np.identity(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
>>> np.eye(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
>>> np.eye(3,  k = 1)
...
```

ndarray的创建

arange()
linspace()

Source

```
>>> np.arange(1, 5, 0.5)
array([ 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
>>> np.linspace(1, 2, 10, endpoint = False)
array([ 1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9])
```

ndarray的创建

Source

random()
normal()
randn()
uniform()
rand()
empty()

```
>>> np.random.random((2, 2))
array([[ 0.79777004,  0.1468679 ],
       [ 0.95838379,  0.86106278]])
>>> np.random.normal(loc=3, scale=3, size=1000)
>>> np.random.normal(loc=0, scale=1, size=100)
>>> np.random.randn(100)
>>> np.random.uniform(low=-5,high=5,size=100)
>>> # N(3,3), N(0,1), U[-5,5)
>>> np.empty((2, 2))
array([[9.90263869e+067, 8.01304531e+262],
       [2.60801200e-310, 1.99392167e-077]])
```

例：采样

- 对一个二维数组，从中有放回的采样出k行数据

```
A = np.random.rand(6,3)
mask = np.random.choice(np.arange(A.shape[0]), k, replace=True)
Sample = A[mask]
```

- 对一个二维数组，从中不放回的采样出k行数据

```
mask = np.random.choice(np.arange(A.shape[0]), k, replace=False)
```

ndarray的创建

fromfunction()

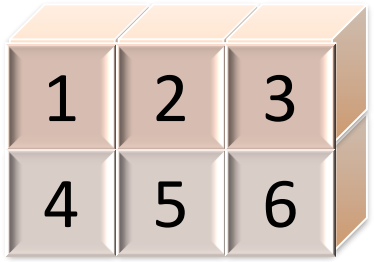
Source

```
>>> np.fromfunction(lambda i, j:(i+1)*(j+1), (9,9))
array([[ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.],
       [ 2.,  4.,  6.,  8., 10., 12., 14., 16., 18.],
       [ 3.,  6.,  9., 12., 15., 18., 21., 24., 27.],
       [ 4.,  8., 12., 16., 20., 24., 28., 32., 36.],
       [ 5., 10., 15., 20., 25., 30., 35., 40., 45.],
       [ 6., 12., 18., 24., 30., 36., 42., 48., 54.],
       [ 7., 14., 21., 28., 35., 42., 49., 56., 63.],
       [ 8., 16., 24., 32., 40., 48., 56., 64., 72.],
       [ 9., 18., 27., 36., 45., 54., 63., 72., 81.]])
```

人工智能程序设计

3 NDARRAY的操作与运算

ndarray的基本操作-切片

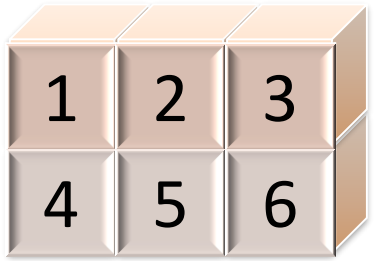


Source

```
>>> aArray = np.array([(1, 2, 3), (4, 5, 6)])
array([[1, 2, 3],
       [4, 5, 6]])
>>> print(aArray[1])
[4 5 6]
>>> print(aArray[0: 2])
[[1 2 3]
 [4 5 6]]
>>> print(aArray[:, [0, 1]])
[[1 2]
 [4 5]]
>>> print(aArray[1, 0:2])
[4 5]
```

- 行翻转，列翻转
- 边界值为1内部为0的数组
- 数组切片是创建拷贝还是副本

ndarray的基本操作-布尔索引



Source

```
>>> aArray = np.arange(1, 101)
>>> bArray = aArray[aArray <= 50]
...
>>> aArray[(aArray % 2 == 0) & (aArray > 50)]
...
>>> aArray[(aArray % 2 == 0)]  = -1
...
>>> aArray = np.arange(1, 101)
>>> cArray = np.where(aArray % 2 == 0, -1, aArray)
...
```

ndarray的基本操作-改变数组形状1

Source

```
>>> aArray = np.array([(1,2,3),(4,5,6)])
>>> aArray.shape
(2, 3)
>>> bArray = aArray.reshape(3,2)
>>> bArray
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> aArray
array([[1, 2, 3],
       [4, 5, 6]])
```

-1的功能

Source

```
>>> aArray.resize(3,2)
>>> aArray
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> aArray.shape = (3,2)
```

ndarray的基本操作-改变数组形状2

Source

```
>>> aArray = np.array([(1,2,3),(4,5,6)])
>>> # 数组展平
>>> x = aArray.ravel()
>>> y = aArray.flatten()
```

copy or view

ndarray的基本操作-数组组合

Source

```
>>> bArray = np.array([1,3,7])
>>> cArray = np.array([3,5,8])
>>> np.hstack((bArray, cArray))
array([1, 3, 7, 3, 5, 8])
>>> np.vstack((bArray, cArray))
array([[1, 3, 7],
       [3, 5, 8]])
```

水平方向
垂直方向

ndarray的基本操作-数组分割

Source

```
>>> dArray = np.arange(1,17).reshape(4,4)
>>> dArray
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
>>> np.hsplit(dArray, 2)
>>> np.vsplit(dArray, 2)
>>> np.split(dArray, 2, axis = 1)
>>> np.split(dArray, 2, axis = 0)
```

水平方向
垂直方向

例：筛法求[2,n]之间的素数

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

```
import numpy as np

def primes_searching(n):
    a = np.arange(3, n+1, 2)
    primes = [2]
    while len(a) > 0:
        p = a[0]
        a = a[a % p != 0]
        primes.append(p)
    return primes

print(primes_searching(200))
```

例：计算前N项Fibonacci数列

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

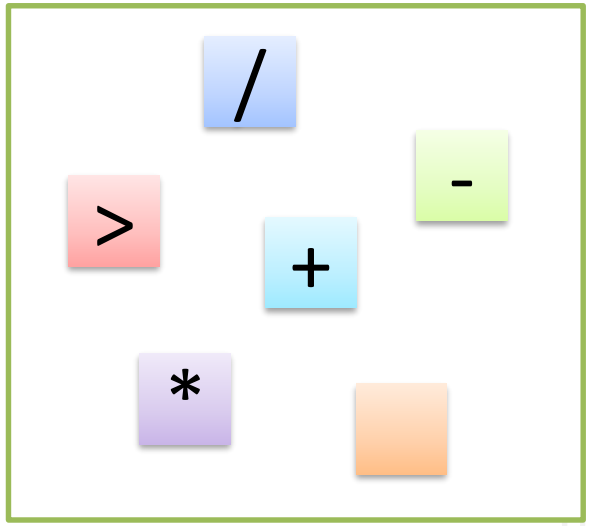
Binet's Formula

```
import numpy as np

N = 30
n = np.arange(1, N+1)
sqrt5 = np.sqrt(5)
phi = (1 + sqrt5)/2
fib = (phi**n - (-1/phi)**n)/sqrt5
fib = fib.astype(int)

print(fib)
```

ndarray的运算



利用基本运算符

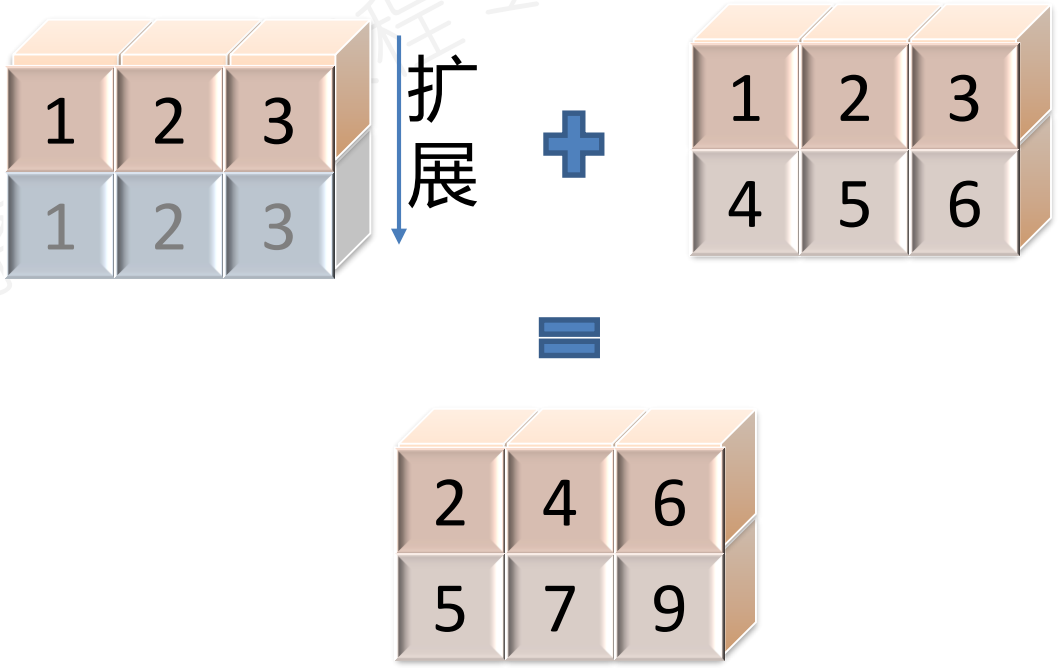
Source

```
>>> aArray = np.array([(5,5,5),(5,5,5)])
>>> bArray = np.array([(2,2,2),(2,2,2)])
>>> cArray = aArray * bArray
>>> cArray
array([[10, 10, 10],
       [10, 10, 10]])
>>> aArray += bArray
>>> aArray
array([[7, 7, 7],
       [7, 7, 7]])
```

ndarray的运算

广播功能

较小的数组会广播到较大数组的大小，使它们的形状兼容



Source

```
>>> a = np.array([1,2,3])
>>> b = np.array([[1,2,3],[4,5,6]])
>>> a + b
array([[2, 4, 6],
       [5, 7, 9]])
```

计算每门课与平均值的差值
keepdims = True&广播

ndarray的运算—简单统计

Source

```
>>> aArray = np.array([(6,5,4),(3,2,1)])
>>> aArray.sum()
21
>>> aArray.sum(axis = 0)
array([9, 7, 5])
>>> aArray.sum(axis = 1)
array([15, 6])
>>> aArray.min()      # return value
1
>>> aArray.argmin()     # return index
5
```

np.sum(aArray >= 5)

sum	mean
std	var
min	max
argmin	argmax
cumsum	cumprod

利用基本数组统计方法

ndarray的运算—简单统计

Source

```
>>> aArray = np.array([(6,5,4),(3,2,1)])
>>> aArray.mean()
3.5
>>> aArray.var()
2.9166666666666665
>>> aArray.std()
1.707825127659933
```

sum	mean
std	var
min	max
argmin	argmax
cumsum	cumprod
sort	argsort

利用基本数组统计方法

```
array([1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800], dtype=int32)
```

ndarray的运算—统计

Source

```
>>> aArray = np.array([3,1,2])
>>> aArray.argsort()    # np.argsort(aArray)
array([1, 2, 0], dtype=int64)
>>> aArray[aArray.argsort()]
>>> aArray.argsort()[0]
1
>>> bArray = np.array([(6,4,5),(3,2,8)])
>>> bArray.argsort()
array([[1, 2, 0],
       [1, 0, 2]], dtype=int64)
>>> bArray.argsort(axis = 0)
array([[1, 1, 0],
       [0, 0, 1]], dtype=int64)
```

sum	mean
std	var
min	max
argmin	argmax
cumsum	cumprod
sort	argsort

利用基本数组统计方法

求最大元素的索引

Source

```
>>> x = np.random.rand(4,4)
>>> x
array([[0.14433369, 0.50050329, 0.88142745, 0.37142465],
       [0.02226626, 0.54092377, 0.72971328, 0.2772584 ],
       [0.5318909 , 0.78939379, 0.37960946, 0.86154654],
       [0.05792592, 0.09889635, 0.04411921, 0.78049691]])
>>> ind_max = x.argmax()
>>> ind_max
2
>>> np.unravel_index(ind_max, x.shape)
(0, 2)
```

```
np.take(x, (indi, indj, ...))
np.take(x, (indi, indj, ...), axis=1)
np.take(x, (indi, indj, ...), axis=0)
```

order : *{'C', 'F'}, optional*

Determines whether the indices should be viewed as indexing in row-major (C-style) or column-major (Fortran-style) order.

New in version 1.6.0.

NumPy文件读写及简单统计1

```
>>> x = np.arange(1,17).reshape(4,4)
>>> np.savetxt('a.txt', x,  fmt='%d')
>>> data = np.loadtxt('a.txt')
>>> dji = np.loadtxt('DJI.csv', delimiter = ',', usecols = (3,4), unpack = True, ?)
```

```
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  16
```

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	2018/8/6	25437.43	25540.02	25381.38	25502.18	25502.18	238990000
3	2018/8/7	25551.65	25692.72	25551.65	25628.91	25628.91	239910000
4	2018/8/8	25615.72	25634.11	25557.48	25583.75	25583.75	217770000
5	2018/8/9	25589.79	25613.31	25492.69	25509.23	25509.23	214970000
6	2018/8/10	25401.19	25401.19	25222.88	25313.14	25313.14	234480000

- 计算收盘价的均值
- 计算收盘价大于27000的天数

NumPy文件读写及简单统计2

```
>>> data = np.genfromtxt('DJI.csv', delimiter= ',', dtype=None, encoding=None)
>>> from io import StringIO
>>> s = StringIO("1,1.3,,abcde")
>>> data = np.genfromtxt(s, dtype=None, delimiter=",", encoding=None)
>>> data
array((1, 1.3, False, 'abcde'),
      dtype=[('f0', '<i4'), ('f1', '<f8'), ('f2', '?'), ('f3', '<U5')])
```

StringIO:
内存中的“文件”

例：缺失值填充及检测

已知有数据："1, 2, 3, 4, 5\n6, , , 7, 8\n , , 9,10,11\n"
利用IO功能, 将其读取到数组中（先将上述字符串转换为IO流）,保存为整型，缺少的部分使用-999填充，最后打印-999的位置索引(x,y)。

```
import numpy as np
from io import StringIO

text = "1, 2, 3, 4, 5\n6, , , 7, 8\n , , 9,10,11\n"
f = StringIO(text)
data = np.genfromtxt(f, delimiter=",", dtype=int, filling_values=-999)
loc_x, loc_y = np.where(data==-999)

print(list(zip(loc_x, loc_y)))
```

人工智能程序设计

4 UFUNC函数

ndarray的ufunc函数

- ufunc（universal function，通用）是一种能对数组的每个元素进行操作的函数。NumPy内置的许多ufunc函数都是在C语言级别实现的，计算速度非常快，数据量大时有很大的优势。

Math operations: add, subtract, mod, power, log, …
Trigonometric functions: sin, cos, tan, tanh, …
Bit-twiddling functions: bitwise_and, invert, …
Comparison functions: greater, less, equal, logical_and, …
Floating functions: isinf, fabs, modf, floor, …

https://docs.scipy.org/doc/numpy/reference/ufuncs.html

ndarray的ufunc函数

File

```
# Filename: 1.py
import time
import math
import numpy as np
x = np.arange(0, 1000, 0.001)
t_m1 = time.process_time()
for i, t in enumerate(x):
        x[i] = math.pow((math.sin(t)), 2)
t_m2 = time.process_time()
y = np.arange(0, 1000, 0.001)
t_n1 = time.process_time()
y = np.power(np.sin(y), 2)
t_n2 = time.process_time()
```

```
Running time of math: t_m2 - t_m1
Running time of numpy: t_n2 - t_n1
```

将标量函数转换为ufunc函数

- np.frompyfunc(func, nin, nout)
  nin: 参数个数，int类型
  nout：函数返回对象个数，int类型
- np.vectorize(...)

```
>>> oct_array = np.frompyfunc(oct, 1, 1)
>>> oct_array(np.array((10, 30, 100)))
array(['0o12', '0o36', '0o144'], dtype=object)
>>> np.array((oct(10), oct(30), oct(100)))
array(['0o12', '0o36', '0o144'], dtype='<U5')
```

```
def f(x):
    return 2*x

double_array = np.frompyfunc(f,1,1)
print(double_array(np.arange(1,1000)))
```

思考：均匀采样[0,1]之间的1000个数，获得所有数与某个给定值的乘积

人工智能程序设计

5 专门的应用

ndarray的专门应用—线性代数

Source

```
>>> import numpy as np
>>> x = np.array([[1,2], [3,4]])
>>> r1 = np.linalg.det(x)
>>> print(r1)
-2.0
>>> r2 = np.linalg.inv(x)
>>> print(r2)
[[-2.   1. ]
 [ 1.5 -0.5]]
>>> r3 = np.dot(x, x)
>>> print(r3)
[[ 7 10]
 [15 22]]
```

Scipy中的linalg模块

dot	矩阵内积
linalg.det	行列式
linalg.inv	逆矩阵
linalg.solve	多元一次方程组求根
linalg.eig	求特征值和特征向量

线性方程组求解

```
import numpy as np
a = np.array([[2, -3, 1], [3, 2, 0], [1, 7, -1]])
b = np.array([1, 13, 16])
x = np.linalg.solve(a, b)
print(x)
print(np.allclose(np.dot(a, x), b))
```

$$2x_0-3x_1+x_2=1$$
$$3x_0+2x_1=13$$
$$x_0+7x_1-x_2=16$$

```
a = np.matrix([[2, -3, 1], [3, 2, 0], [1, 7, -1]])
b = np.matrix([1, 13, 16]).T
```

```
from scipy.linalg import solve
x = solve(a, b)
```