NumPy – Numeric Python

Операции с ndarray

- ✓ Вычисления в NumPy векторизованы: циклы перебирающие элементы массивов не нужны
- ✓ Основные и логические операции выполняются поэлементно

```
import numpy as np

# array and scalar
v = np.arange(1,5)       # [1 2 3 4]
print v*1.1              # [ 1.1  2.2  3.3  4.4]
print v**2               # [ 1  4  9 16]
print v<3                # [ True  True False False]

# two arrays
w=np.array([2,3,5,7])    # [2 3 5 7]
print v*w                # [ 2  6 15 28]
print w**v               # [   2    9  125 2401]
```

NumPy – Numeric Python

- ✓ В NumPy имеется встроенный набор математических функций выполняющихся поэлементно

Elementwise functions

```
a=np.array([2,4,6])
b=np.array([1,3,5])
print np.sqrt(a)          # [ 1.41421356  2.          2.44948974]
print np.exp(b)           # [ 2.71828183 20.08553692  148.4131591 ]
print np.arctan2(a,b)     # [ 1.10714872  0.92729522  0.87605805]
```

NumPy – Numeric Python

Функция `dot(x,y)` в NumPy

- ✓ Для 2D матриц – произведение матриц
- ✓ Для 1D векторов – скалярное произведение
- ✓ Для размерности N – сумма произведения элементов последней «оси» `x` на предпоследнюю `y`

`dot()` определено и как метод и как функция:

```
# products of matrix
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

print x.dot(y)                  # [[19 22]
print np.dot(x,y)               #  [43 50]]

print y.dot(x)                  # [[23 34]
print np.dot(y,x)               #  [31 46]]
```

NumPy – Numeric Python

Изменение «формы» без изменения количества элементов

- ✓ `ravel()` – приведение к 1D размерности
- ✓ `reshape(new_shape)` – задание новой формы
- ✓ `transpose()` – транспонирование

reshaping

```
a = np.arange(1,7).reshape((2,3)) # modify shape
# [[1 2 3]
#  [4 5 6]]

print a.T                         # == a.transpose()
# [[1 4]
#  [2 5]
#  [3 6]]

print a.ravel()                   # flatten
# [1 2 3 4 5 6]
```

NumPy – Numeric Python

Трансляция (broadcasting)

- ✓ Механизм для выполнения операций с матрицами разных размерностей
- ✓ Меньшая матрица «транслируется» вдоль недостающих размерностей таким образом что бы получилась размерность большей матрицы

```
a = np.arange(11,17).reshape((2,3)) # [[11 12 13]
                                    #  [14 15 16]]
b = np.array([2,4,6])
print a+b                           # [[13 16 19]
                                    #  [16 19 22]]
c=np.array([1,3])
# print a+c          # Error!
print a+c.reshape(2,1)              # [[12 13 14]
                                    #  [17 18 19]]
```

NumPy – Numeric Python

Пример: outer production

```
print b*c.reshape(2,1)          # [[ 2  4  6]
                                #  [ 6 12 18]]
```

Пример: сложение x(4x4) + y(2x2)

```
x = np.arange(11,27).reshape((4,4))     # [[11 12 13 14]
                                        #  [15 16 17 18]
                                        #  [19 20 21 22]
                                        #  [23 24 25 26]]
y = np.arange(1,5).reshape((2,2))       # [[1 2]
                                        #  [3 4]]
# print x+y # Error
print (x.reshape(4,2,2)+y).reshape(4,4) # [[12 14 16 18]
                                        #  [16 18 20 22]
                                        #  [20 22 24 26]
                                        #  [24 26 28 30]]
```

NumPy – Numeric Python

Ввод-вывод в NumPy

```
import numpy as np

arr = np.arange(1,51,dtype=np.int32).reshape((5,10))

# save to a text file
# creates a space delimited file by default
np.savetxt(fname='test_array.out', X=arr)

# load saved file
read_arr = np.loadtxt(fname='test_array.out')
print np.all(read_arr == arr)   # True
```

- ✓ В функциях предусмотрены опции контроля типов данных, разделители, комментарии, заголовки и т.д.

NumPy – Numeric Python

Линейная алгебра np.linalg

```
import numpy as np

A = np.array([[1, 2], [3, 4]])
b = np.array([10, 20])

# Matrix determinant
print np.linalg.det(A)  # -2.0

# Invers matrix
invA = np.linalg.inv(A)
print np.dot(A,invA)    # [[  1.00000000e+00   1.11022302e-16]
                        #  [  0.00000000e+00   1.00000000e+00]]

# Solve for Ax = b
x = np.linalg.solve(A, b)
print x                 # [ 0.  5.]
```

NumPy – Numeric Python

...продолжение

```
# Eigenvalues and Eigenvectors:
# eig() returns tuple; the first element is the eigen values
# and the second one is a matrix with eigen vectors

egVal,egVec = np.linalg.eig(A)
print egVal                 # [-0.37228132  5.37228132]
print egVec                 # [[-0.82456484 -0.41597356]
                            #  [ 0.56576746 -0.90937671]]
```

- ✓ Модуль линейной алгебры в NumPy находится в развитии
- ✓ Пакет SciPy имеет дополнительные функции

SciPy: Численное интегрирование

$\int_0^2 x^3 \, dx = 4$

```
from scipy.integrate import quad

res,err = quad(lambda x: x**3, 0, 2)
print '{0} estimated error {1:.1e}'.format(res,err)
```

```
4.0 estimated error 4.4e-14
```

$\int_0^1 sin(x)/x \, dx = Si(1) \sim 0.946083$

```
import numpy as np
from scipy.integrate import quad

res,err = quad(lambda x: np.sin(x)/x, 0,1)
print '{0} estimated error {1:.1e}'.format(res,err)
```

```
0.946083070367 estimated error 1.1e-14
```

SciPy: Численное интегрирование

Двойной интеграл: $\int_{x=\pi}^{2\pi} \int_{y=0}^{\pi} y \sin(x) + x \cos(y)\ dxdy$

```
import numpy as np
from scipy.integrate import dblquad

# We have to provide functions for the range of the y-variable
# and y must be the first argument, x the second in integrand
res, err = dblquad(lambda y,x: y * np.sin(x) + x * np.cos(y),
                   np.pi, 2*np.pi,
                   lambda x: 0, lambda x: np.pi)
print '{0} estimated error {1:.1e}'.format(res,err)
```

```
-9.86960440109 estimated error 1.4e-13
```

SciPy: Статистика

Генерация случайных чисел по заданному распределению

```
import numpy as np
import scipy as sp
from scipy import stats
Nmax = 1000000
gs = stats.norm(0,1).rvs(Nmax) # normal distribution N(0,1)
# gs = stats.uniform().rvs(Nmax) # uniform distribution

print 'Mean : {0:8.6f}'.format(sp.mean(gs))
print 'Variance : {0:8.6f}'.format(sp.var(gs))
print 'Std. deviation : {0:8.6f}'.format(sp.std(gs))
print 'Median : {0:8.6f}'.format(sp.median(gs))
```

```
   Normal                          Uniform
Mean : -0.000008                Mean : 0.499965
Variance : 1.000192             Variance : 0.083471
Std. deviation : 1.000096       Std. deviation : 0.288913
Median : 0.000132               Median : 0.500133
```

SciPy: Статистика

...и вычисление характеристик распределения

```
n, min_max, mean, var, skew, kurt = stats.describe(gs)

print 'Number of elements: {0:d}'.format(n)
print 'Min: {0:8.6f} Max: {1:8.6f}'.format(min_max[0],min_max[1])
print 'Mean: {0:8.6f}'.format(mean)
print 'Variance: {0:8.6f}'.format(var)
print 'Skew : {0:8.6f}'.format(skew)
print 'Kurtosis: {0:8.6f}'.format(kurt)
```

Normal	Uniform
Number of elements: 1000000	Number of elements: 1000000
Min: -4.805859 Max: 4.654846	Min: 0.000000 Max: 0.999998
Mean: -0.000008	Mean: 0.499965
Variance: 1.000193	Variance: 0.083471
Skew : 0.000920	Skew : -0.001275
Kurtosis: -0.002568	Kurtosis: -1.201883