Numpy作业及答案

1. 依照课件3中的内容给出测算三角波（triangle\_wave（））y1、y2、y3、y4四种方式的计算速度与结果比较的代码，并对其运算显示结果。

源代码（参考）：

# -\*- coding: utf-8 -\*-

import numpy as np

import time

def triangle\_wave(x, c, c0, hc):

x = x - int(x) # 三角波的周期为1，因此只取x坐标的小数部分进行计算

if x >= c: r = 0.0

elif x < c0: r = x / c0 \* hc

else: r = (c-x) / (c-c0) \* hc

return r

x = np.linspace(0, 2, 1000)

start = time.clock()

y1 = np.array([triangle\_wave(t, 0.6, 0.4, 1.0) for t in x])

print "y1:", time.clock() - start

start = time.clock()

triangle\_ufunc1 = np.frompyfunc(triangle\_wave, 4, 1)

y2 = triangle\_ufunc1(x, 0.6, 0.4, 1.0)

y2 = y2.astype(np.float)

print "y2:", time.clock() - start

start = time.clock()

triangle\_ufunc2 = np.frompyfunc( lambda x: triangle\_wave(x, 0.6, 0.4, 1.0), 1, 1)

y3 = triangle\_ufunc2(x)

y3 = y3.astype(np.float)

print "y3", time.clock() - start

start = time.clock()

triangle\_ufunc3 = np.vectorize(triangle\_wave, otypes=[np.float])

y4 = triangle\_ufunc3(x, 0.6, 0.4, 1.0)

print "y4", time.clock() - start

print np.all(y1==y2)

print np.all(y2==y3)

print np.all(y3==y4)

结果（参考）：

y1: 0.00326345254734

y2: 0.00148136592202

y3 0.00163189733212

y4 0.00151934089149

True

True

True

1. arr11 = 5-np.arange(1,13).reshape(4,3), 计算所有元素、每一列的和；对每一个元素、每一列求累积和；计算每一行的累计积；计算所有元素的最小值；计算每一列的最大值；计算所有元素、每一行的均值；计算所有元素、每一列的中位数；计算所有元素的方差，每一行的标准差。

>>>arr11 = 5-np.arange(1,13).reshape(4,3)

>>>arr11

array([[ 4, 3, 2],

[ 1, 0, -1],

[-2, -3, -4],

[-5, -6, -7]])

>>> np.sum(arr11)   #计算所有元素的和

-18

>>>np.sum(arr11,axis = 0)    #对每一列求和

 array([ -2,  -6, -10])

>>>np.sum(arr11, axis = 1) #对每一行求和

 array([  9,   0,  -9, -18])

>>>np.cumsum(arr11) #对每一个元素求累积和（从上到下，从左到右的元素顺序）

array([  4,   7,   9,  10,  10,   9,   7,   4,   0,  -5, -11, -18], dtype=int32)

>>>np.cumsum(arr11, axis = 0) #计算每一列的累积和，并返回二维数组

array([[  4,   3,   2],

[  5,   3,   1],

[  3,   0,  -3],

[ -2,  -6, -10]], dtype=int32)

>>>np.cumprod(arr11, axis = 1) #计算每一行的累计积，并返回二维数组

array([[   4,   12,   24],

[   1,    0,    0],

[  -2,    6,  -24],

[  -5,   30, -210]], dtype=int32)

>>> np.min(arr11)   #计算所有元素的最小值

-7

>>> np.max(arr11, axis = 0) #计算每一列的最大值

array([4, 3, 2])

>>> np.mean(arr11)  #计算所有元素的均值

-1.5

>>> np.mean(arr11, axis = 1) #计算每一行的均值

array([ 3.,  0., -3., -6.])

>>> np.median(arr11)   #计算所有元素的中位数

 -1.5

>>> np.median(arr11, axis = 0)   #计算每一列的中位数

array([-0.5, -1.5, -2.5])

>>> np.var(arr12)   #计算所有元素的方差

 5.354166666666667

>>>np.std(arr12, axis = 1)   #计算每一行的标准差

array([ 2.49443826,  1.88561808,  1.69967317,  2.1602469 ])

1. 在数组[1, 2, 3, 4, 5]中每相邻两个数字中间插入两个0。

>>> Z = np.array([1,2,3,4,5])

>>> nz = 2

>>> Z0 = np.zeros(len(Z) + (len(Z)-1)\*(nz))

>>> Z0[::nz+1] = Z

>>> print Z0

array([ 1., 0., 0., 2., 0., 0., 3., 0., 0., 4., 0., 0., 5.])

4、归一化，将矩阵规格化到0～1，即最小的变成0，最大的变成1，最小与最大之间的等比缩放。试对Z = np.random.random((5,5))进行归一化。

>>>> Z = np.random.random((5,5))

>>> Zmax,Zmin = Z.max(), Z.min()

>>> Z = (Z - Zmin)/(Zmax - Zmin)

>>> print Z

[[ 0. 0.32173291 0.17607851 0.6270374

0.95000808]

[ 0.49153473 0.70465605 0.61930085 0.00303294 1.

]

[ 0.4680561 0.88742782 0.29899683 0.80704789

0.12300414]

[ 0.05094248 0.23065875 0.82776775 0.07873239

0.50644422]

[ 0.27417053 0.78679222 0.517819 0.5649124 0.4716856

]]

5、找出数组中与给定值最接近的数（通用方法）。（例：任一数组Z=array([[0,1,2,3],[4,5,6,7]])，给任一定值z=5.1）

>>> Z=array([[0,1,2,3],[4,5,6,7]])

>>> print Z

[[0 1 2 3]

[4 5 6 7]]

>>> z=5.1

>>> np.abs(Z - z).argmin()

5

>>> print Z.flat[np.abs(Z - z).argmin()]

5

6. python解方程：3x + 6y -5z = 12；x-3y+2z = -2；5x -y +4z = 10。

解答：  
import numpy as np  
a = np.linalg.solve([[3,6,-5],[1,-3,2],[5,-1,4]],[12,-2,10])  
print(a)

[ 1.75 1.75 0.75]

7．参见课件4第45页，，对**g**(**y**)在100个切比雪夫节点之上分別使用**Polynomial（**Polynomial.fit**）**和**Chebyshev（**Chebyshev.fit**）**进行插值，在[-1,1]区间上取1000个等距点对误差进行比较。g(x)= sin(z\*\*2) + sin(z)\*\*2,且 z=(x - 1) \* 5。

import numpy as np

from numpy.polynomial import Polynomial, Chebyshev

def g(x):

x = (x - 1) \* 5

return np.sin(x\*\*2) + np.sin(x)\*\*2

n = 100

x = Chebyshev.basis(n).roots()

xd = np.linspace(-1, 1, 1000)

p\_g = Polynomial.fit(x, g(x), n - 1, domain=[-1,1])

c\_g = Chebyshev.fit(x, g(x), n - 1, domain=[-1, 1])

print "Max Polynomial Error:", abs(g(xd) - p\_g(xd)).max()

print "Max Chebyshev Error:", abs(g(xd) - c\_g(xd)).max()

Max Polynomial Error: 1.195307121

Max Chebyshev Error: 6.47575904011e-09

D:\enthought\canopy\User\lib\site-packages\numpy\polynomial\polynomial.py:1383: RankWarning: The fit may be poorly conditioned

warnings.warn(msg, pu.RankWarning)

8、**试用bincount()函数替代histogram()函数完成统计男青少年年龄和身高的例子的计算（数据见height.csv）**

解答：

d = np.loadtxt("height.csv",delimiter=",")

x =np.array(d[:,0],dtype=int)

zz=np.bincount(x)

zz=zz[7: ]

w = np.array(d[:,1])

zz1=np.bincount(x, w)

zz1=zz1[7: ]

zz1/zz

Out:

array([ 125.96 , 132.06666667, 137.82857143, 143.8, 148.14 , 153.44, 162.15555556, 166.86666667, 172.83636364, 173.3 , 175.275 , 174.19166667, 175.075 ])

9、**使用二项分布进行赌博计算**

同时抛弃9枚硬币，如果正面朝上少于5枚，则输掉8元，否则就赢8元。如果手中有1000元作为赌资，请问赌博10000次后可能会是什么情况呢？

# -\*- coding: utf-8 -\*-

import numpy as np

import pylab

#np.random.seed(1234)

binomial = np.random.binomial(9,0.5,10000) #生成二项分布随机数

money = np.zeros(10000) #生成10000次赌资的列表

money[0] = 1000 #首次赌资为1000元

for i in range(1,10000):

if binomial[i]<5 :

money[i] = money[i-1] - 8

#如果少于5枚正面，则在上一次赌资的基础上输掉8元

else:

money[i] = money[i-1] + 8

#如果至少5枚正面，则在上一次赌资的基础上赢取8元

pylab.plot(np.arange(10000), money)

pylab.show()