

法律声明

□ 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

□ 课程详情请咨询

■ 微信公众号：小象学院

■ 新浪微博：小象AI学院



Python库



小象学院
ChinaHadoop.cn

邹博

主要内容

- 简单例子
 - 图像像素到字符的转换
 - 信息摘要与安全哈希算法MD5/SHA1
- 统计量：均值/方差/偏度/峰度
- 多元高斯分布
- 阶乘的实数域推广： Γ 函数
- Pearson相关系数的计算
- 快速傅里叶变换FFT与信号滤波
- 奇异值分解SVD与图像特征
- 股票数据相关：收盘价格曲线、滑动平均线、K线图
- 图像的卷积与卷积网络CNN
- 蝴蝶效应：Lorenz系统的曲线生成
- 绘制分形图：Mandelbrot集合

Image2Char



Code

```
import numpy as np
from PIL import Image

if __name__ == '__main__':
    image_file = 'son.png'
    height = 100

    img = Image.open(image_file)
    img_width, img_height = img.size
    width = 2 * height * img_width // img_height
    img = img.resize((width, height), Image.ANTIALIAS)
    pixels = np.array(img.convert('L'))
    chars = "MNHQ$OC?7>!:~.;. "
    N = len(chars)
    step = 256 // N
    print(N)
    result = ''
    for i in range(height):
        for j in range(width):
            result += chars[pixels[i][j] // step]
        result += '\n'
    with open('text.txt', mode='w') as f:
        f.write(result)
```

MD5/SHA1

```
import hashlib
```

```
if __name__ == "__main__":
    md5 = hashlib.md5()
    md5.update('This is a sentence.')
    md5.update('This is a second sentence.')
    print u'不出意外, 这个将是“乱码”:', md5.digest()
    print u'MD5:', md5.hexdigest()

    md5 = hashlib.md5()
    md5.update('This is a sentence.This is a second sentence.')
    print u'MD5:', md5.hexdigest()
    print md5.digest_size, md5.block_size
    print '-----'

    sha1 = hashlib.sha1()
    sha1.update('This is a sentence.')
    sha1.update('This is a second sentence.')
    print u'不出意外, 这个将是“乱码”:', sha1.digest()
    print u'SHA1:', sha1.hexdigest()

    sha1 = hashlib.sha1()
    sha1.update('This is a sentence.This is a second sentence.')
    print u'SHA1:', sha1.hexdigest()
    print sha1.digest_size, sha1.block_size
    print '====='
```

5.1.hashlib

```
↑ 不出意外, 这个将是“乱码”： + I QUL k ) $ q
↓ MD5: 2bfe14491651554c05996b1629248571
↻ MD5: 2bfe14491651554c05996b1629248571
+ 16 64
+ -----
+ 不出意外, 这个将是“乱码”： JH a e% + 1 : ,
SHA1: 99a74a48d5fb611a6525962ba3d51131cb3a1d2c
SHA1: 99a74a48d5fb611a6525962ba3d51131cb3a1d2c
20 64
```

思考

新浪微博长URL的压缩（存储、查找相关）

取消关注 | 14

...

【题目】

新浪微博发布内容要求字符不超过140，但是用户如果在发布内容中有很长的url时，会认为是很多字符。所以新浪上发布内容包含一个URL时，时把他压缩成一个TinyURL(缩小)。比如：

(因为无法发站外链接，所以去掉了链接中的http://头)

输入：zhidao.baidu.com/search?ct=17&pn=0&tn=ikaslist&rn=10&word=helloworld&ie=utf-8&fr=www

实际显示：//asdfa.cn/ak78ss

前面asdfa.cn是对应域名zhidao.baidu.com，后面长长的字符串被压缩成ak78ss。

【问题】

现在让你来设计TinyURL的实现，以下问题要怎么设计：

- (1)：域名后面的编码如何实现？
- (2)：对于已经映射过的一个URL，怎么查找已存在的TinyUrl？
- (3)：有10亿个url，一个服务上存不下，需要多台服务器，怎么设计实现
- (4)：让你来设计这样一个服务，最大的问题是什么？

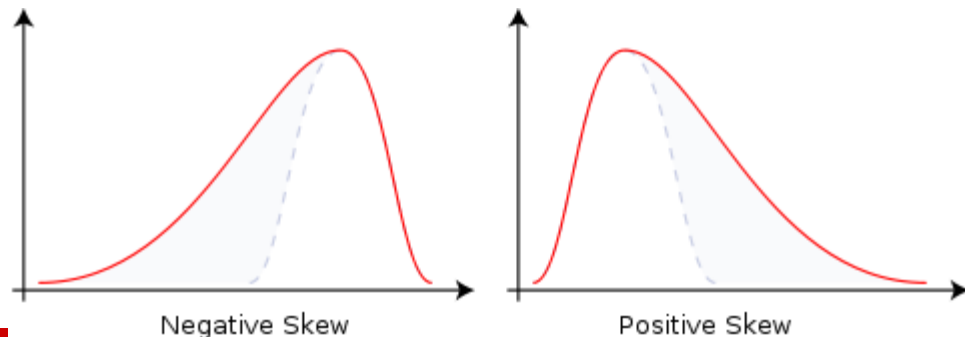
【探讨】

暂时的思路(根据网友的回答汇总)，欢迎探讨：

(1)：两种思路，一种是把真实网址存数据库，然后取自增id，做个哈希或者进制转换之类的，生成短网址，用的时候查一下就行了；另一种是直接真实网址哈希然后截取特定位数，6位的话 $(26+26+10)^6$ 种组合，应该够用了，实在不行再做一次碰撞检测

- (2)：直接key-value存储查询
- (3)：二次哈希，根据ak78ss这样的值映射到不同机器上，hash或者字母序层次下去
- (4)：查询速度？响应时间？还有过期的url在浪费存储空间？

偏度



- 偏度衡量随机变量概率分布的不对称性，是相对于均值不对称程度的度量。
- 偏度的值可以为正，可以为负或者无定义。
- 偏度为负(负偏)/正(正偏)表示在概率密度函数左侧/右侧的尾部比右侧的长，长尾在左侧/右侧。
- 偏度为零表示数值相对均匀地分布在平均值的两侧，但不一定意味着一定是对称分布。

偏度公式

□ 三阶累积量与二阶累积量的1.5次方的比率。

■ 偏度有时用Skew[X]来表示。

$$\gamma_1 = E\left[\left(\frac{X - \mu}{\sigma}\right)^3\right] = \frac{E[(X - \mu)^3]}{(E[(X - \mu)^2])^{3/2}} = \frac{\kappa_3}{\kappa_2^{3/2}}$$

$$\gamma_1 = E\left[\left(\frac{X - \mu}{\sigma}\right)^3\right] = \frac{E[X^3] - 3\mu E[X^2] + 2\mu^3}{\sigma^3} = \frac{E[X^3] - 3\mu\sigma^2 - \mu^3}{\sigma^3}$$

峰度 $\frac{\mu_4}{\sigma^4}$

- 峰度是概率密度在均值处峰值高低的特征，通常定义四阶中心矩除以方差的平方减3。

$$\gamma_2 = \frac{\kappa_4}{\kappa_2^2} = \frac{\mu_4}{\sigma^4} - 3 = \frac{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^4}{\left(\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 \right)^2} - 3$$

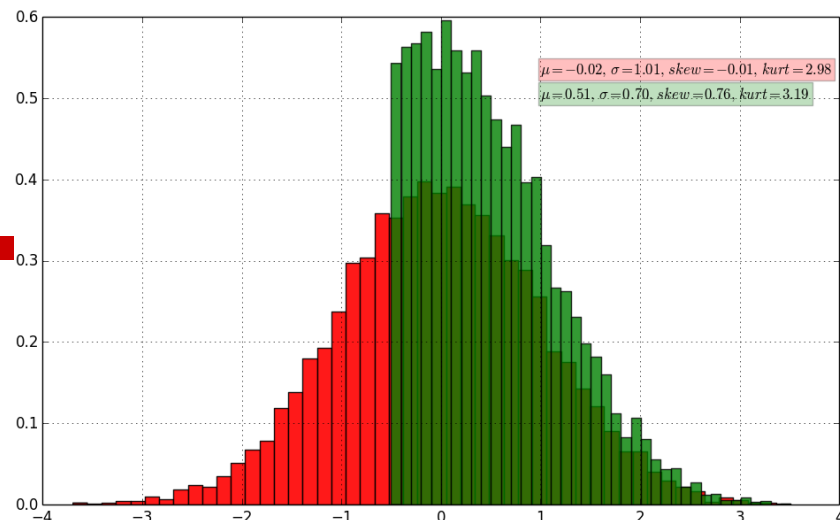
- $\frac{\mu_4}{\sigma^4}$ 也被称为超值峰度(excess kurtosis)。

- “减3”是为了让正态分布的峰度为0。
- 超值峰度为正，称为尖峰态(leptokurtic)
- 超值峰度为负，称为低峰态(platykurtic)

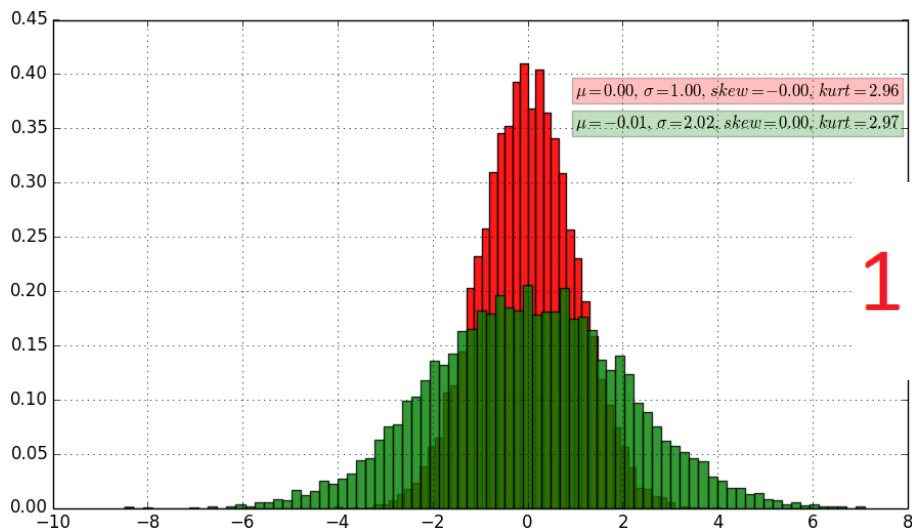
统计量Code

```
if __name__ == "__main__":
    data = list(np.random.randn(10000))
    data2 = list(2*np.random.randn(10000))
    data3 = [x for x in data if x > -0.5]
    data4 = list(np.random.uniform(0, 4, 10000))
    [niu, sigma, skew, kurt] = calc_stat(data)
    [niu2, sigma2, skew2, kurt2] = calc_stat(data2)
    [niu3, sigma3, skew3, kurt3] = calc_stat(data3)
    [niu4, sigma4, skew4, kurt4] = calc_stat(data4)
    print niu, sigma, skew, kurt
    print niu2, sigma2, skew2, kurt2
    print niu3, sigma3, skew3, kurt3
    print niu4, sigma4, skew4, kurt4

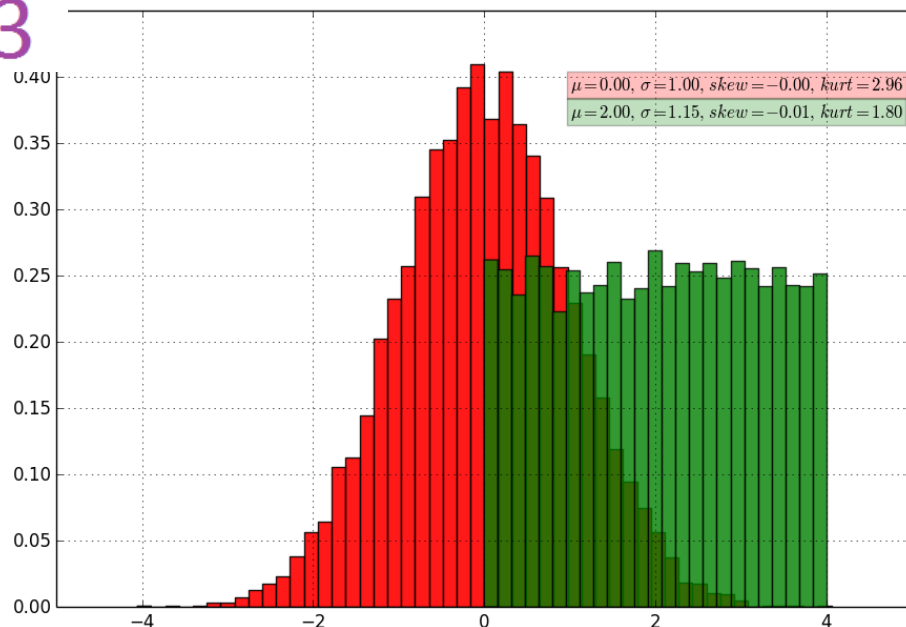
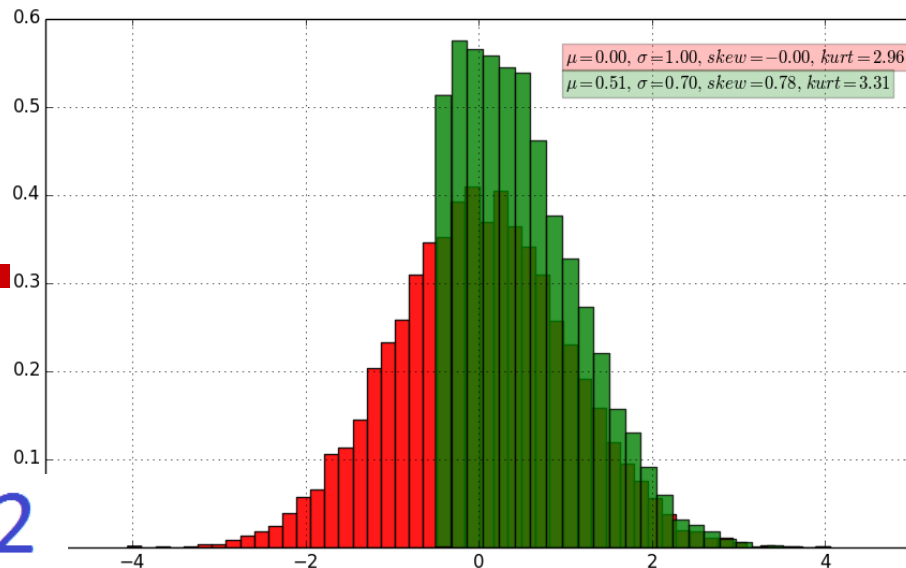
    info = r'$\mu=%.2f, \sigma=%.2f, skew=%.2f, kurt=%.2f$' % (niu, sigma, skew, kurt)
    info2 = r'$\mu=%.2f, \sigma=%.2f, skew=%.2f, kurt=%.2f$' % (niu2, sigma2, skew2, kurt2)
    plt.text(1, 0.38, info, bbox=dict(facecolor='red', alpha=0.25))
    plt.text(1, 0.35, info2, bbox=dict(facecolor='green', alpha=0.25))
    plt.hist(data, 50, normed=True, facecolor='r', alpha=0.9)
    plt.hist(data2, 80, normed=True, facecolor='g', alpha=0.8)
    plt.grid(True)
    plt.show()
```



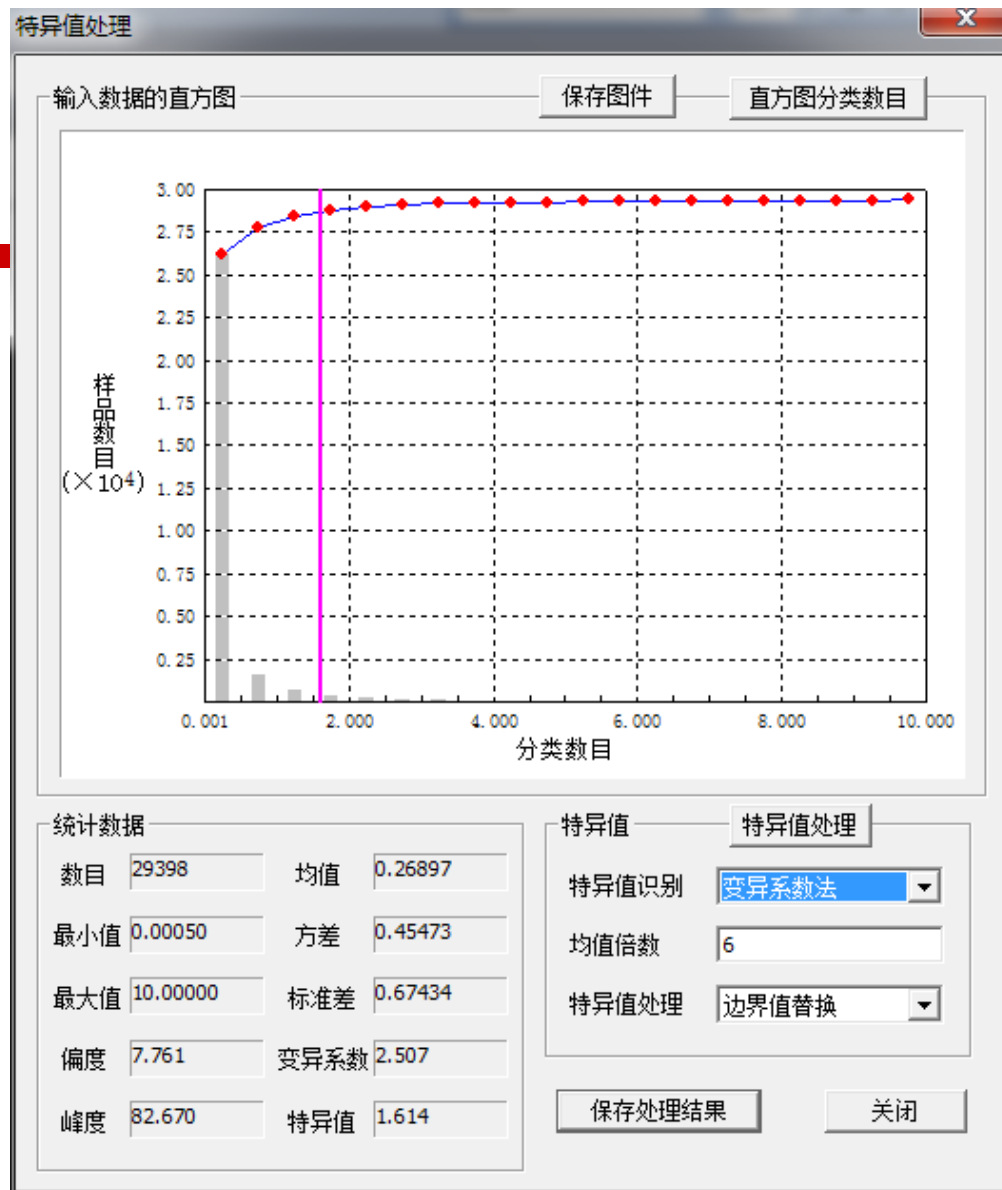
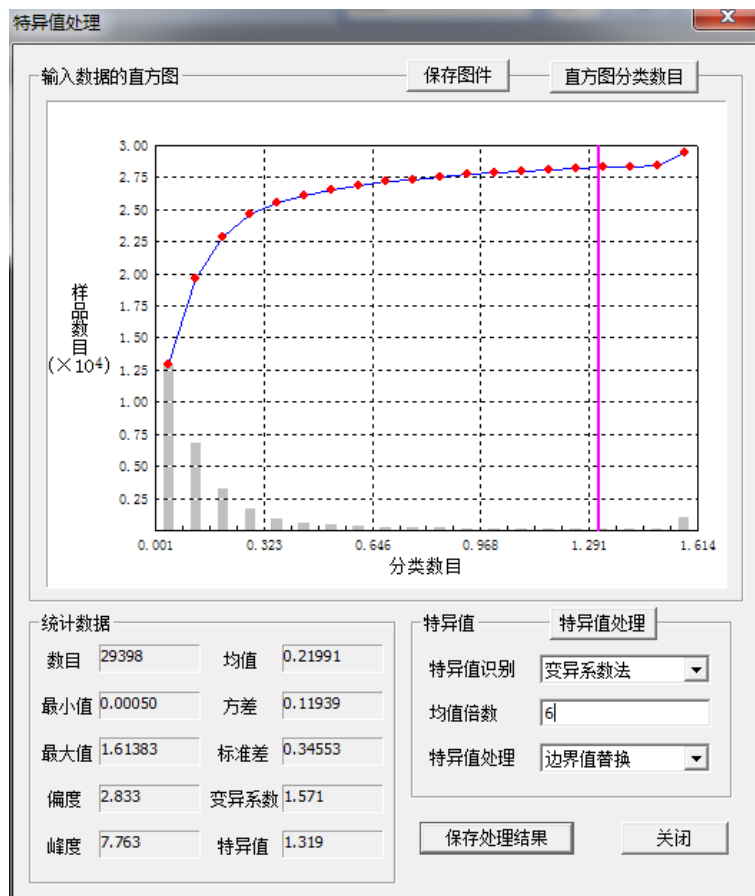
数据分析



0.0029 1.0001 -0.0040 2.9643
-0.0137 2.0208 0.0046 2.9748
0.5134 0.6962 0.7825 3.3052
2.0016 1.1548 -0.0107 1.8012



实践中的例子



统计量

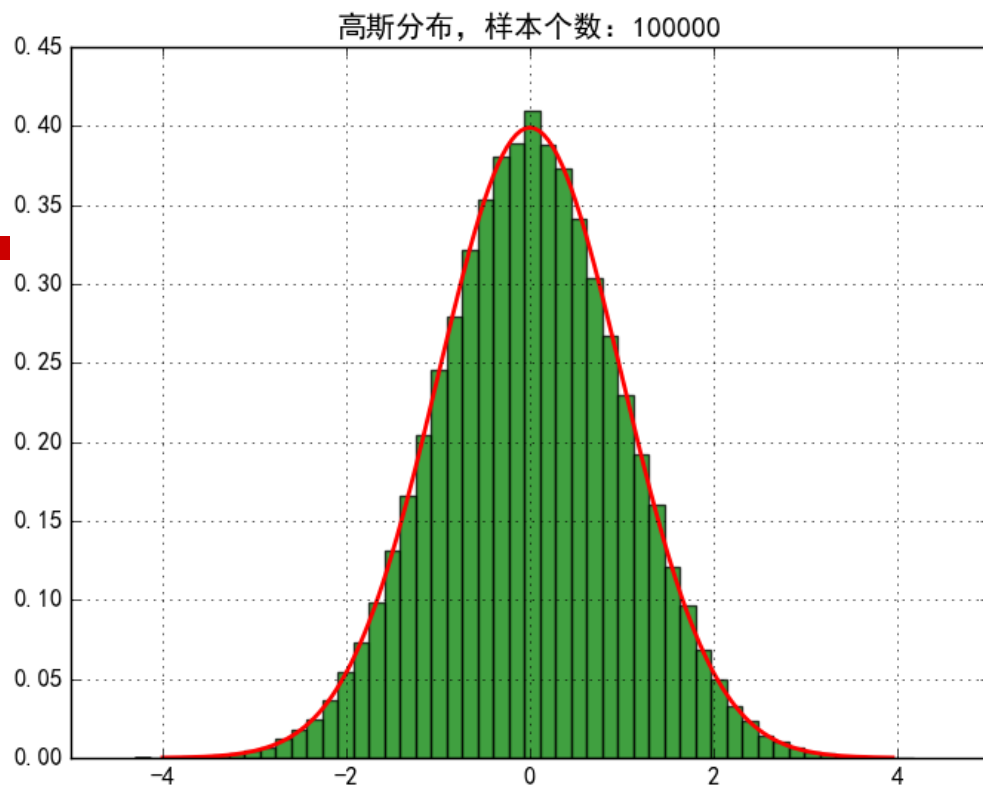
```
def calc_statistics(x):
    n = x.shape[0] # 样本个数

    # 手动计算
    m = 0
    m2 = 0
    m3 = 0
    m4 = 0
    for t in x:
        m += t
        m2 += t*t
        m3 += t**3
        m4 += t**4

    m /= n
    m2 /= n
    m3 /= n
    m4 /= n

    mu = m
    sigma = np.sqrt(m2 - mu*mu)
    skew = (m3 - 3*mu*m2 + 2*mu**3) / sigma**3
    kurtosis = (m4 - 4*mu*m3 + 6*mu*mu*m2 - 4*mu**3*mu + mu**4) / sigma**4 - 3
    print '手动计算均值、标准差、偏度、峰度: ', mu, sigma, skew, kurtosis

    # 使用系统函数验证
    mu = np.mean(x, axis=0)
    sigma = np.std(x, axis=0)
    skew = stats.skew(x)
    kurtosis = stats.kurtosis(x)
    return mu, sigma, skew, kurtosis
```

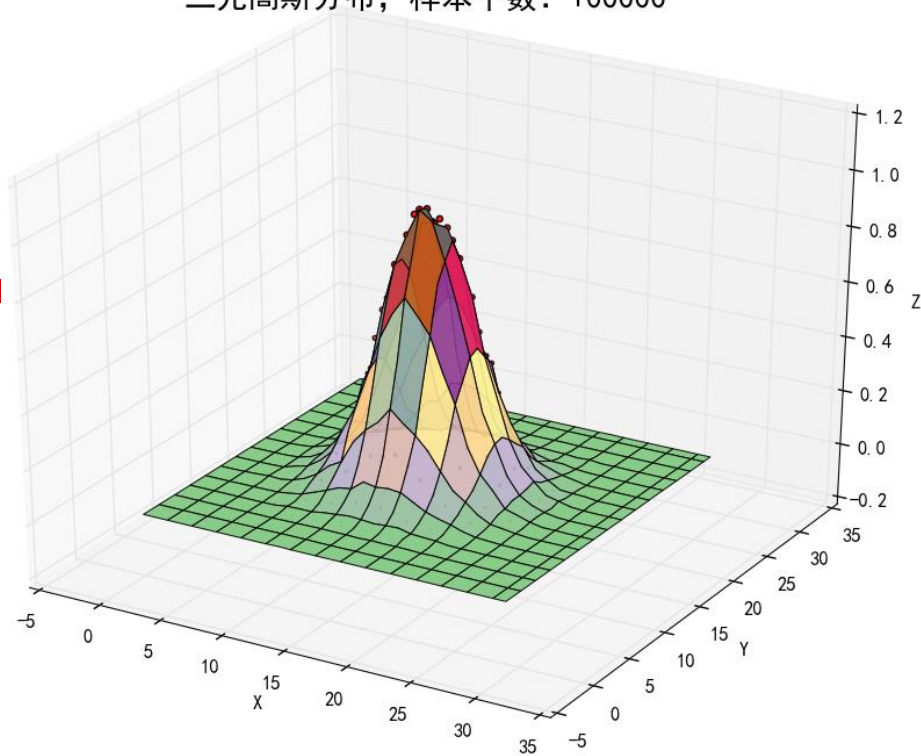


手动计算均值、标准差、偏度、峰度： -0.00232018730484 1.00220229337 0.0070687774347 0.0174102810253

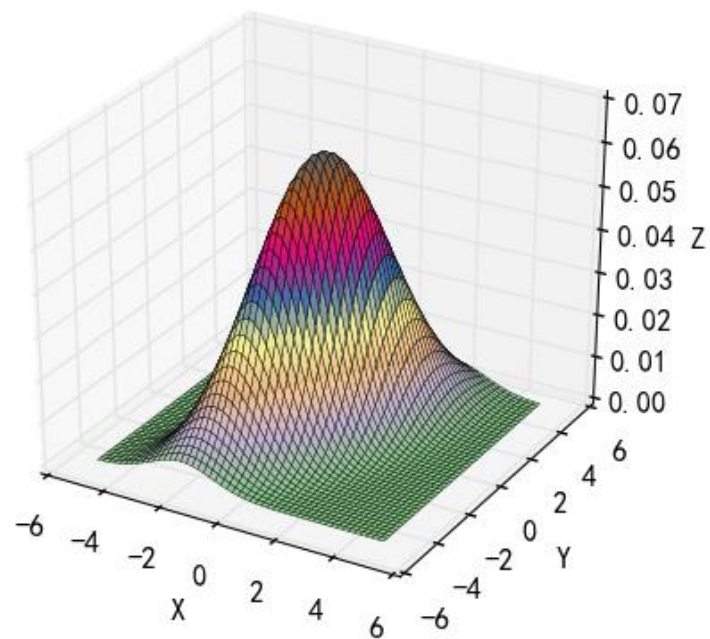
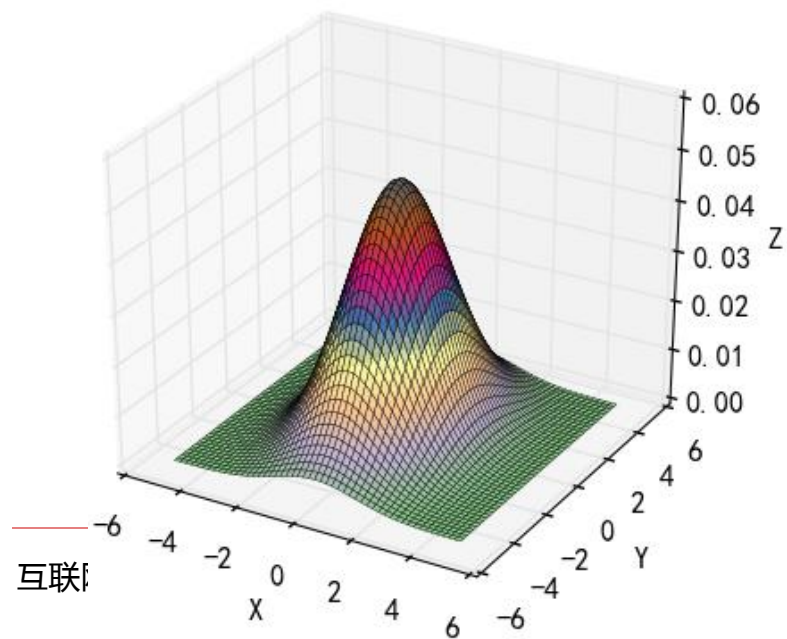
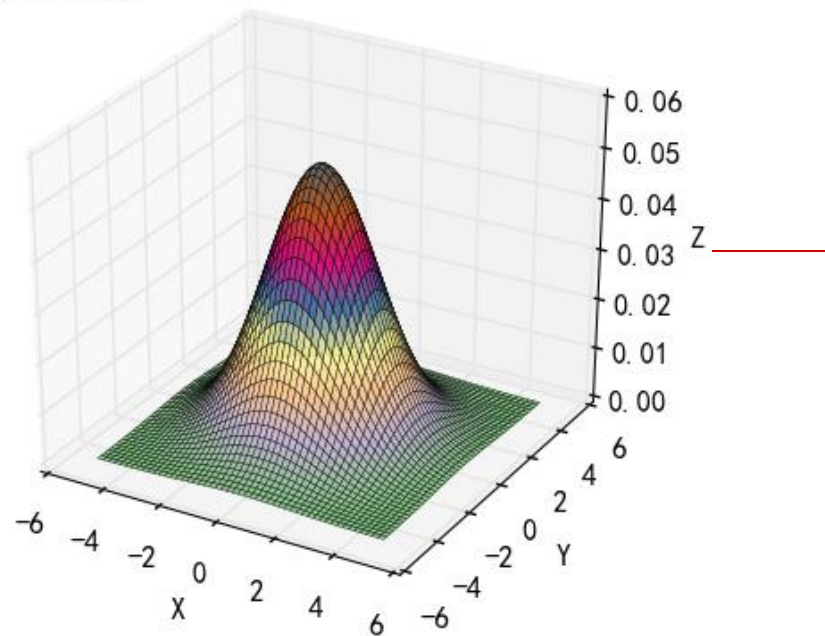
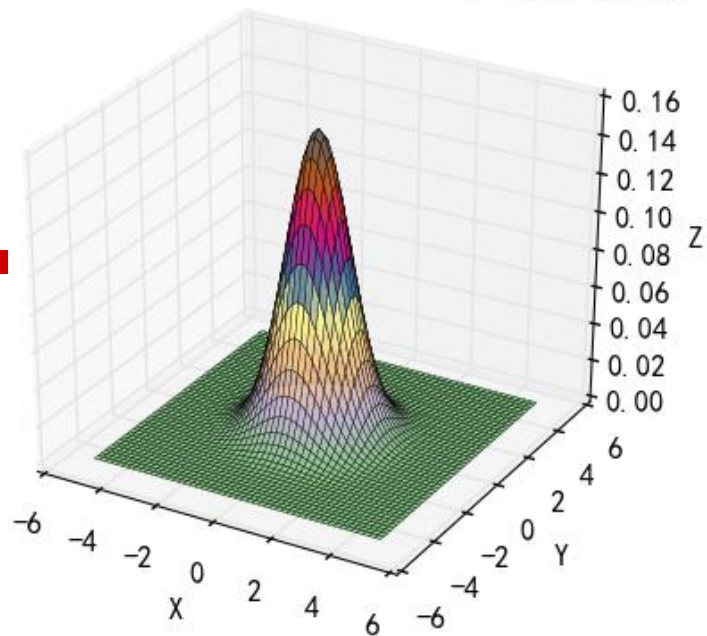
函数库计算均值、标准差、偏度、峰度： -0.00232018730484 1.00220229337 0.0070687774347 0.0174102810253

二元高斯分布

```
d = np.random.randn(100000, 2)
mu, sigma, skew, kurtosis = calc_statistics(d)
print '函数库计算均值、标准差、偏度、峰度: ', mu,
# 二维图像
N = 30
density, edges = np.histogramdd(d, bins=[N, N])
print '样本总数: ', np.sum(density)
density /= density.max()
x = y = np.arange(N)
t = np.meshgrid(x, y)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, projection='3d')
ax.scatter(t[0], t[1], density, c='r', s=15*density, marker='o', depthshade=True)
ax.plot_surface(t[0], t[1], density, cmap=cm.Accent, rstride=2, cstride=2, alpha=0.9, lw=0.75)
ax.set_xlabel(u'X')
ax.set_ylabel(u'Y')
ax.set_zlabel(u'Z')
plt.title(u'二元高斯分布，样本个数: %d' % d.shape[0], fontsize=20)
plt.tight_layout(0.1)
plt.show()
```



二元高斯分布方差比较

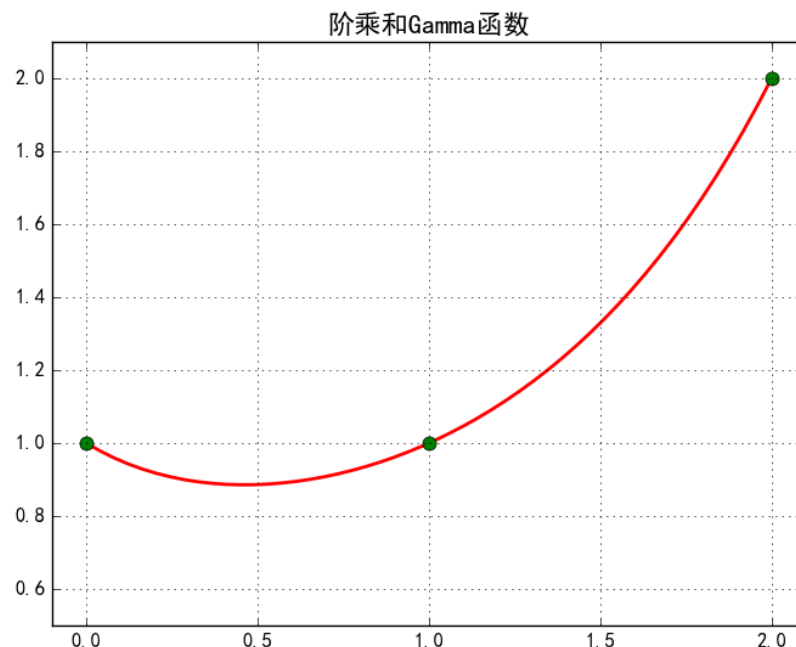
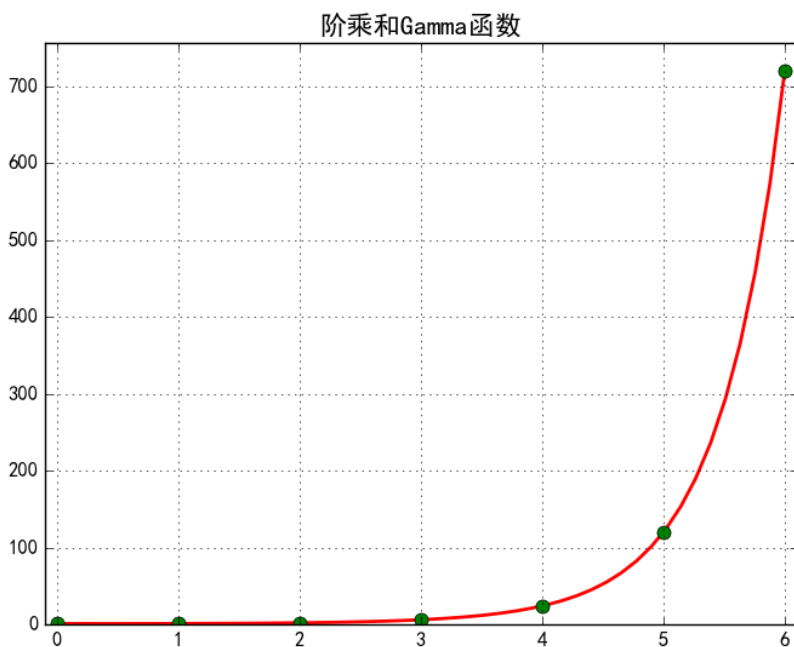


Γ函数

$$\Gamma(x) = (x-1) \cdot \Gamma(x-1) \Rightarrow \frac{\Gamma(x)}{\Gamma(x-1)} = x-1$$

□ Γ函数是阶乘在实数上的推广

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt = (x-1)!$$

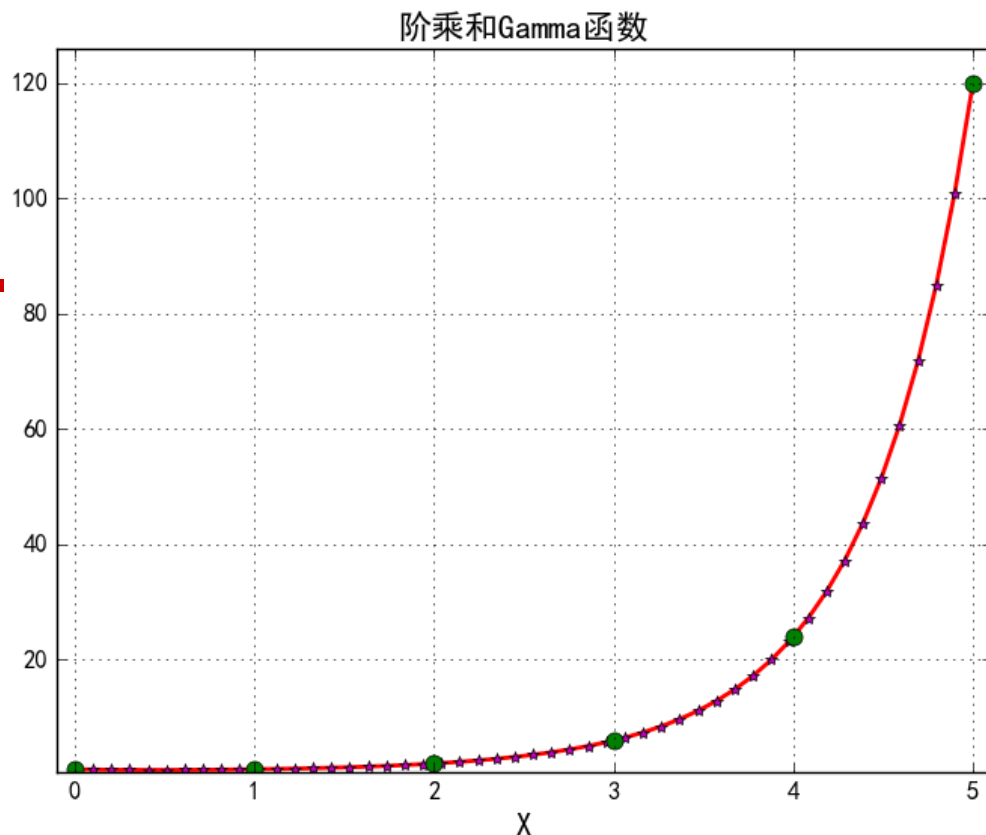


Γ 函数

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.special import gamma
from scipy.special import factorial

mpl.rcParams['axes.unicode_minus'] = False
mpl.rcParams['font.sans-serif'] = 'SimHei'

if __name__ == '__main__':
    N = 5
    x = np.linspace(0, N, 50)
    y = gamma(x+1)
    plt.figure(facecolor='w')
    plt.plot(x, y, 'r-', x, y, 'm*', lw=2)
    z = np.arange(0, N+1)
    f = factorial(z, exact=True)    # 阶乘
    print f
    plt.plot(z, f, 'go', markersize=8)
    plt.grid(b=True)
    plt.xlim(-0.1, N+0.1)
    plt.ylim(0.5, np.max(y)*1.05)
    plt.xlabel(u'X', fontsize=15)
    plt.ylabel(u'Gamma(X) - 阶乘', fontsize=15)
    plt.title(u'阶乘和Gamma函数', fontsize=16)
    plt.show()
```



Pearson相关系数

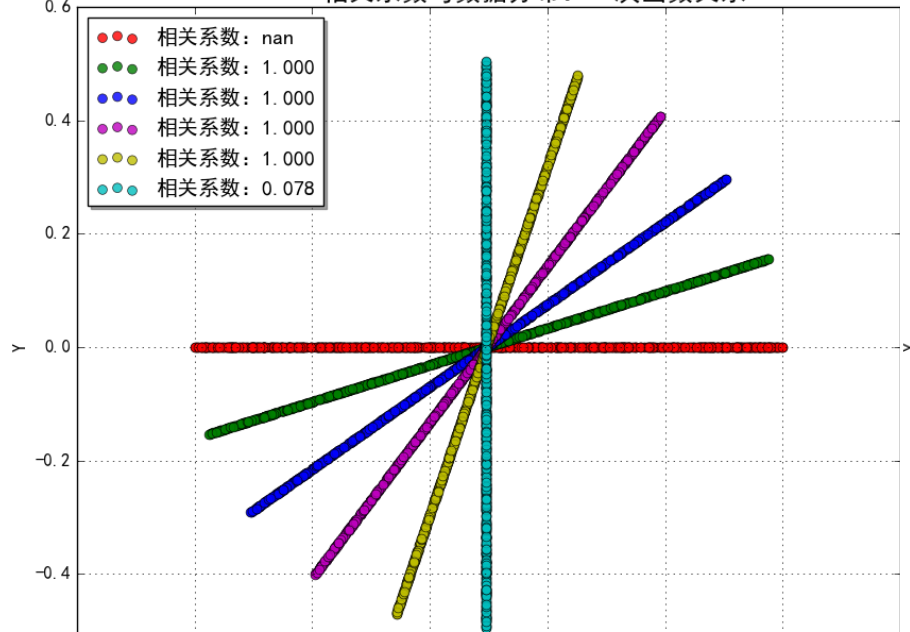
□ 定义相关系数:

$$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}}$$

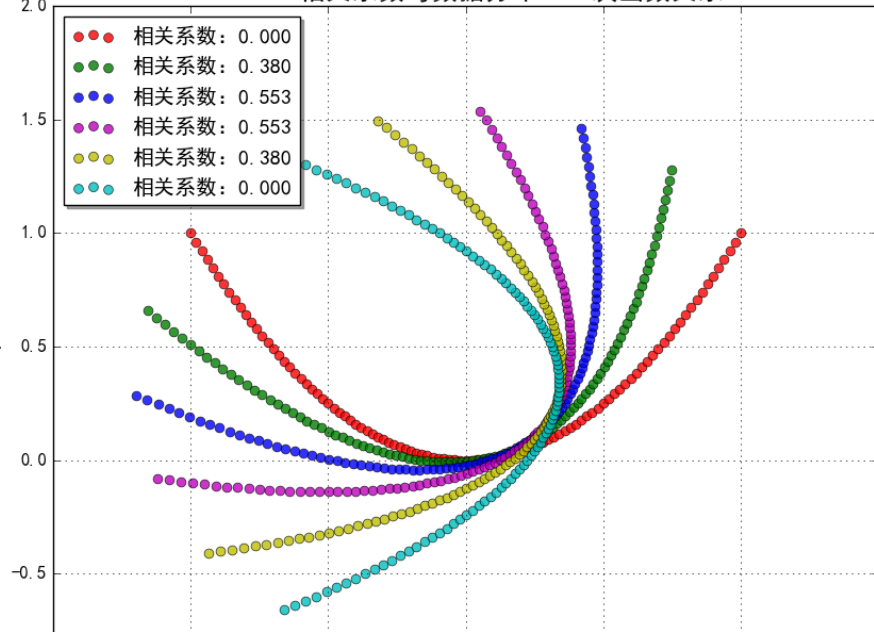
□ 由协方差上界定理可知, $|\rho| \leq 1$

■ 当且仅当X与Y有线性关系时, 等号成立

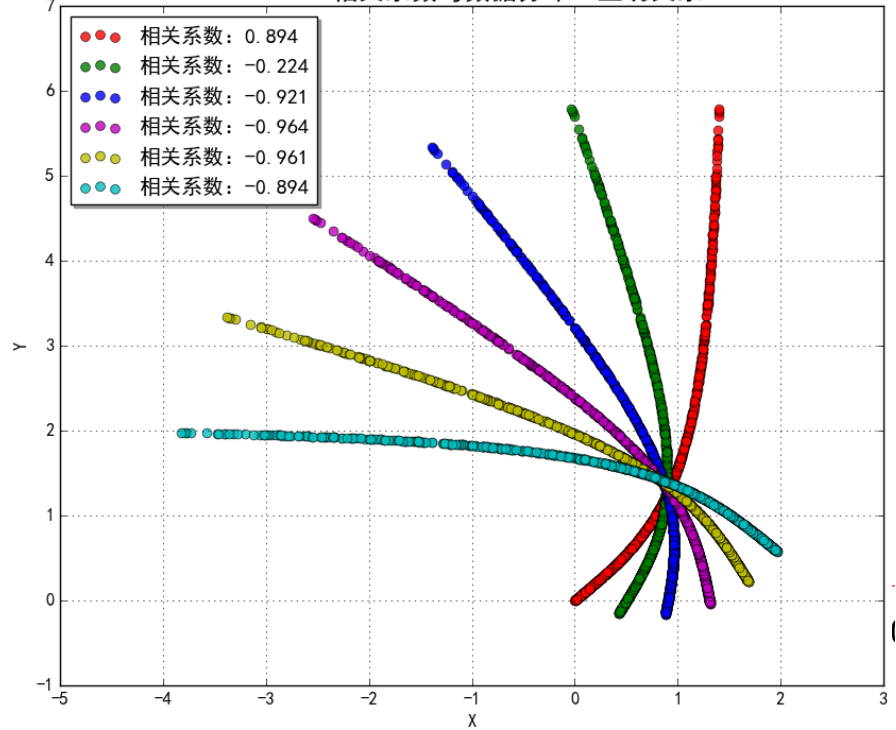
Pearson相关系数与数据分布：一次函数关系



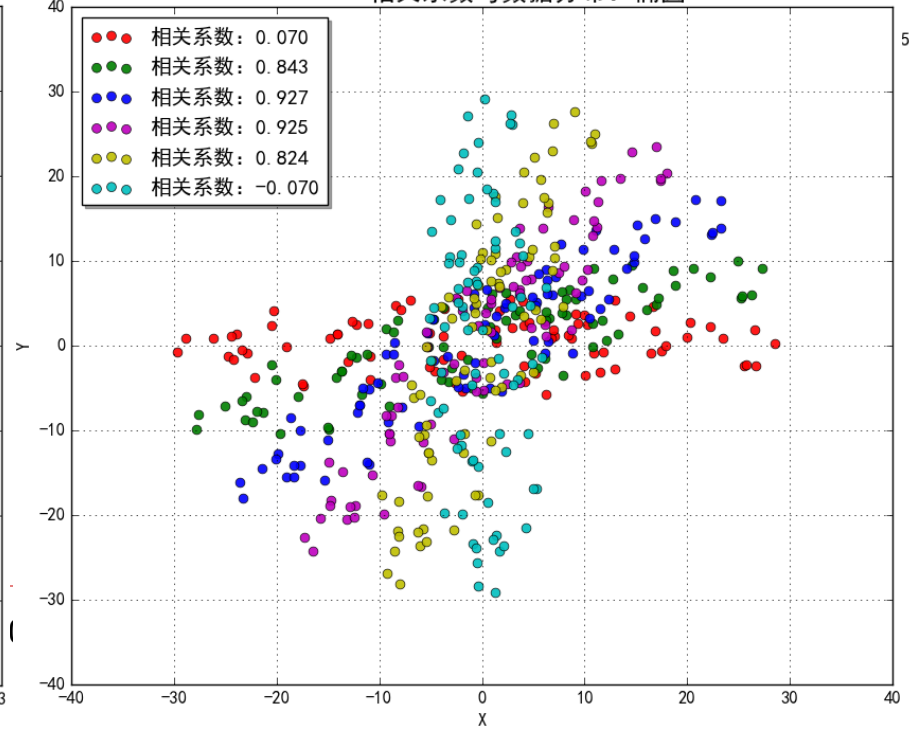
Pearson相关系数与数据分布：二次函数关系



Pearson相关系数与数据分布：正切关系



Pearson相关系数与数据分布：椭圆

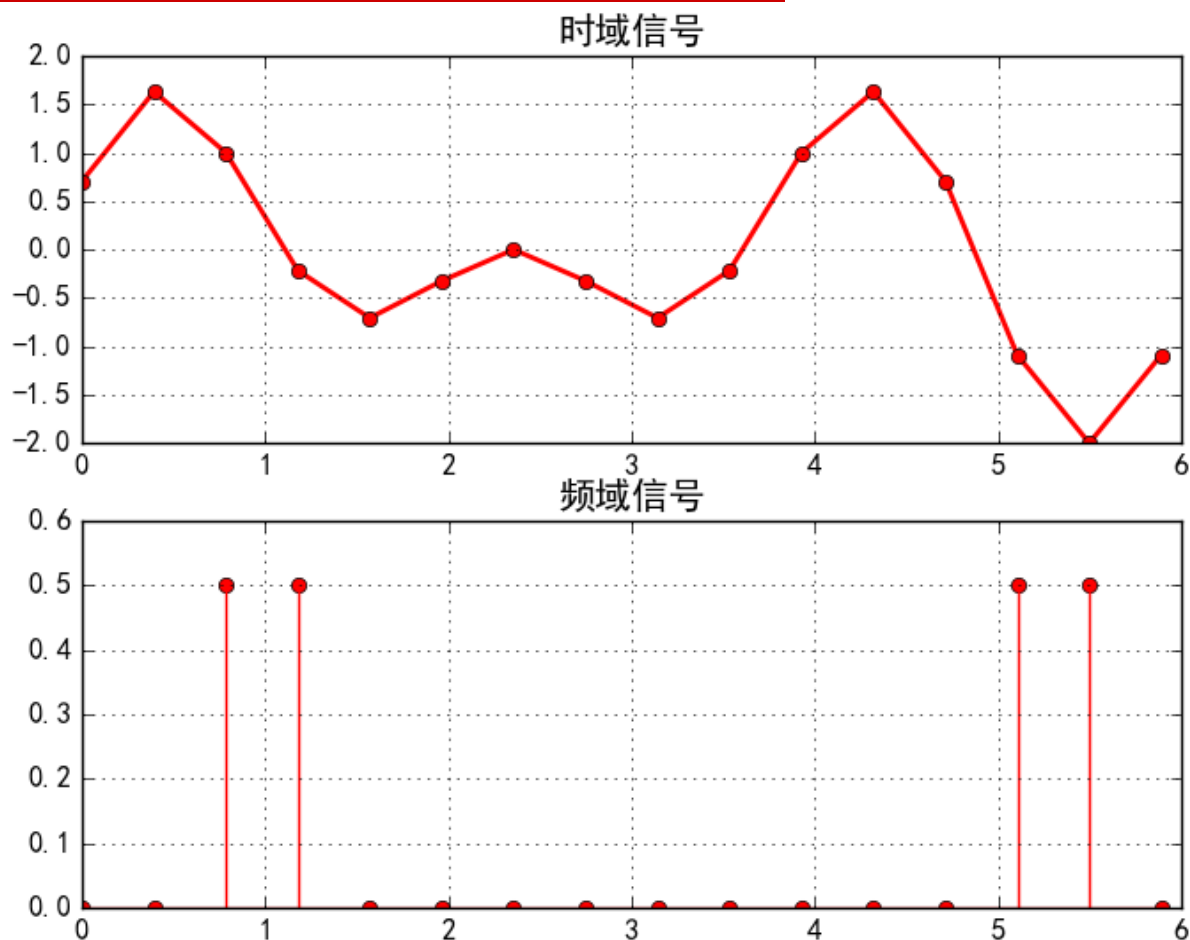


Code

```
def calc_pearson(x, y):
    std1 = np.std(x)
    # np.sqrt(np.mean(x**2) - np.mean(x)**2)
    std2 = np.std(y)
    cov = np.cov(x, y, bias=True)[0,1]
    return cov / (std1 * std2)

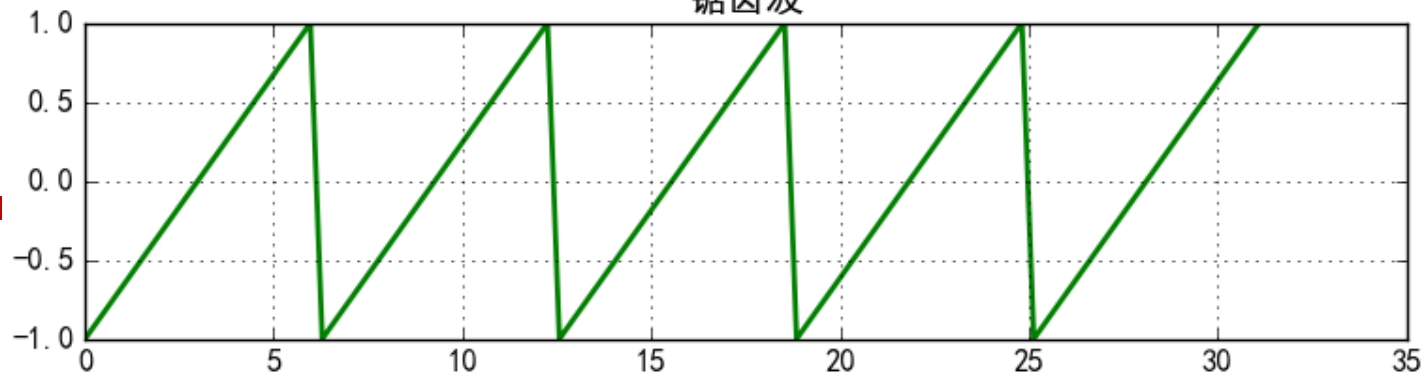
def pearson(x, y, tip):
    clrs = list('rgbmyc')
    plt.figure(figsize=(10, 8), facecolor='w')
    for i, theta in enumerate(np.linspace(0, 90, 6)):
        xr, yr = rotate(x, y, theta)
        p = stats.pearsonr(xr, yr)[0]
        # print calc_pearson(xr, yr)
        print '旋转角度: ', theta, 'Pearson相关系数: ', p
        str = u'相关系数: %.3f' % p
        plt.scatter(xr, yr, s=40, alpha=0.9, linewidths=0.5, c=clrs[i])
    plt.legend(loc='upper left', shadow=True)
    plt.xlabel(u'X')
    plt.ylabel(u'Y')
    plt.title(u'Pearson相关系数与数据分布: %s' % tip, fontsize=18)
    plt.grid(b=True)
    plt.show()
```

时域与频域信号

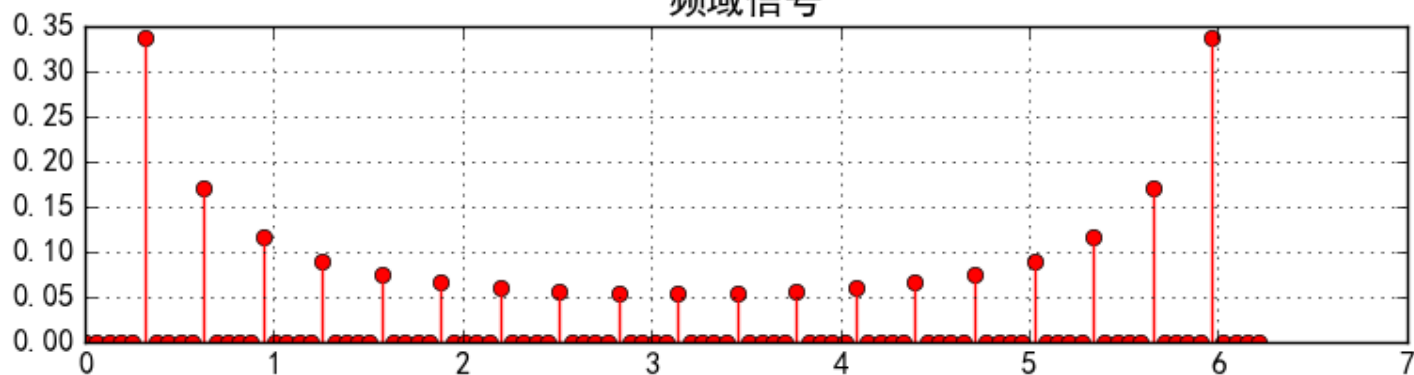


快速傅里叶变换FFT与频域滤波

锯齿波



频域信号

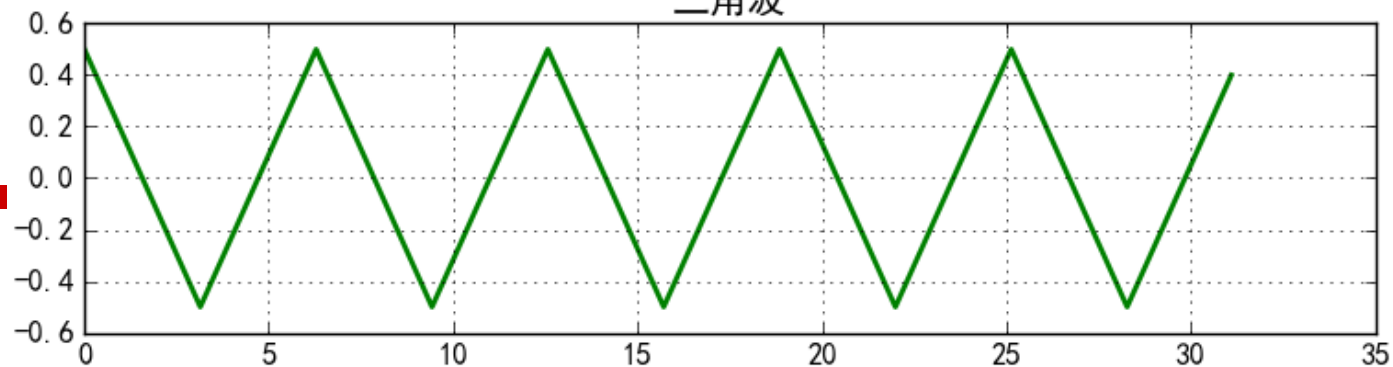


锯齿波恢复信号



快速傅里叶变换FFT与频域滤波

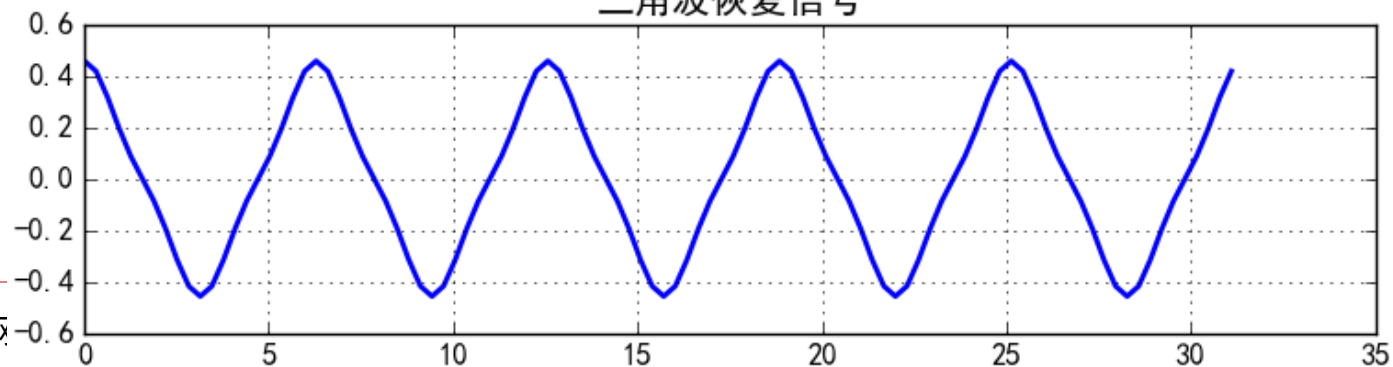
三角波



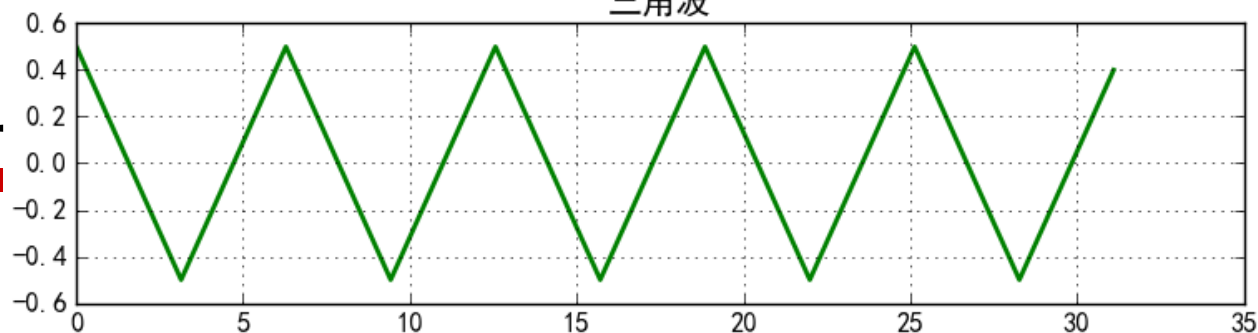
频域信号



三角波恢复信号



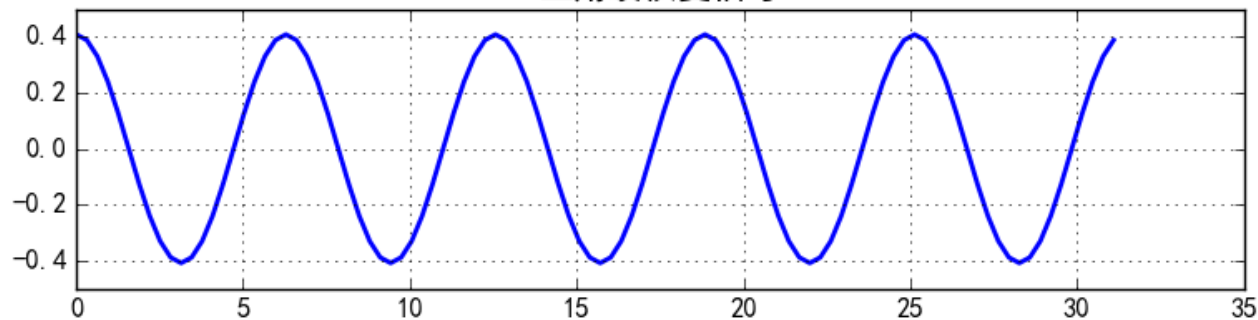
三角波



频域信号



三角波恢复信号



不同的阈值

SVD的提法

$$(A^T \cdot A)v_i = \lambda_i v_i \Rightarrow \begin{cases} \sigma_i = \sqrt{\lambda_i} \\ u_i = \frac{1}{\sigma_i} A \cdot v_i \end{cases} \Rightarrow A = U \Sigma V^T$$

□ 奇异值分解(Singular Value Decomposition)是一种重要的矩阵分解方法，可以看做对称方阵在任意矩阵上的推广。

■ Singular: 突出的、奇特的、非凡的

■ 似乎更应该称之为“**优值分解**”

□ 假设A是一个 $m \times n$ 阶实矩阵，则存在一个分解使得：

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T$$

■ 通常将奇异值由大而小排列。这样， Σ 便能由A唯一确定了。

□ 与特征值、特征向量的概念相对应：

■ Σ 对角线上的元素称为矩阵A的奇异值；

■ U的第i列称为A的关于 σ_i 的左奇异向量；

■ V的第i列称为A的关于 σ_i 的右奇异向量。

奇异值分解-效果

```
def restore(sigma, u, v, K): # 奇异值、左特征向量、右特征向量
    print K
    m = len(u)
    n = len(v[0])
    a = np.zeros((m, n))
    for k in range(K+1):
        for i in range(m):
            a[i] += sigma[k] * u[i][k] * v[k]
    b = a.astype('uint8')
    Image.fromarray(b).save("svd_" + str(K) + ".png")
```



SVD



svd_0.png



svd_1.png



svd_2.png



svd_3.png



svd_4.png



svd_5.png



svd_6.png



svd_7.png



svd_8.png



svd_9.png



svd_10.png



svd_11.png



svd_12.png



svd_13.png



svd_14.png



svd_15.png



svd_16.png



svd_17.png



svd_18.png



svd_19.png



svd_20.png



svd_21.png



svd_22.png



svd_23.png



svd_24.png



svd_25.png



svd_26.png



svd_27.png



svd_28.png



svd_29.png



svd_30.png



svd_31.png



svd_32.png



svd_33.png



svd_34.png



svd_35.png



svd_36.png



svd_37.png



svd_38.png



svd_39.png



svd_40.png



svd_41.png



svd_42.png



svd_43.png



svd_44.png



svd_45.png



svd_46.png



svd_47.png



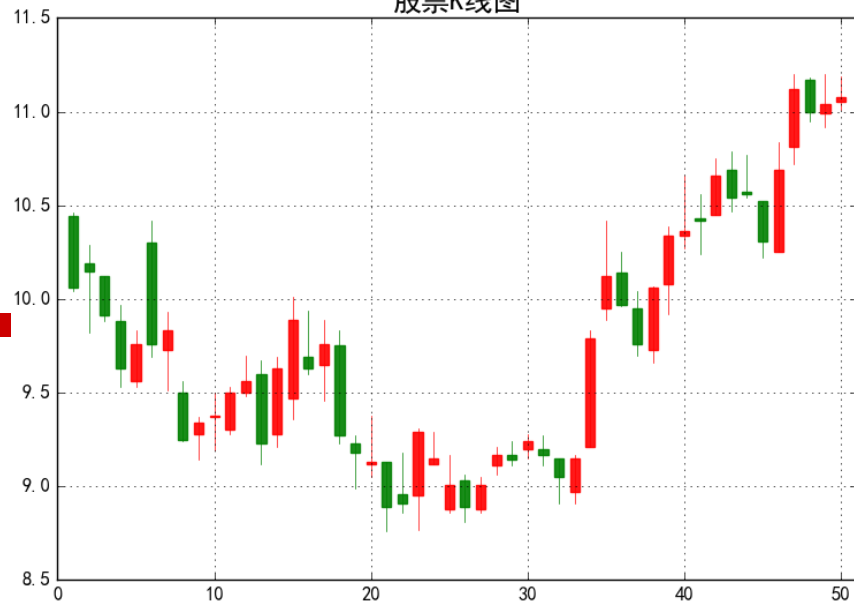
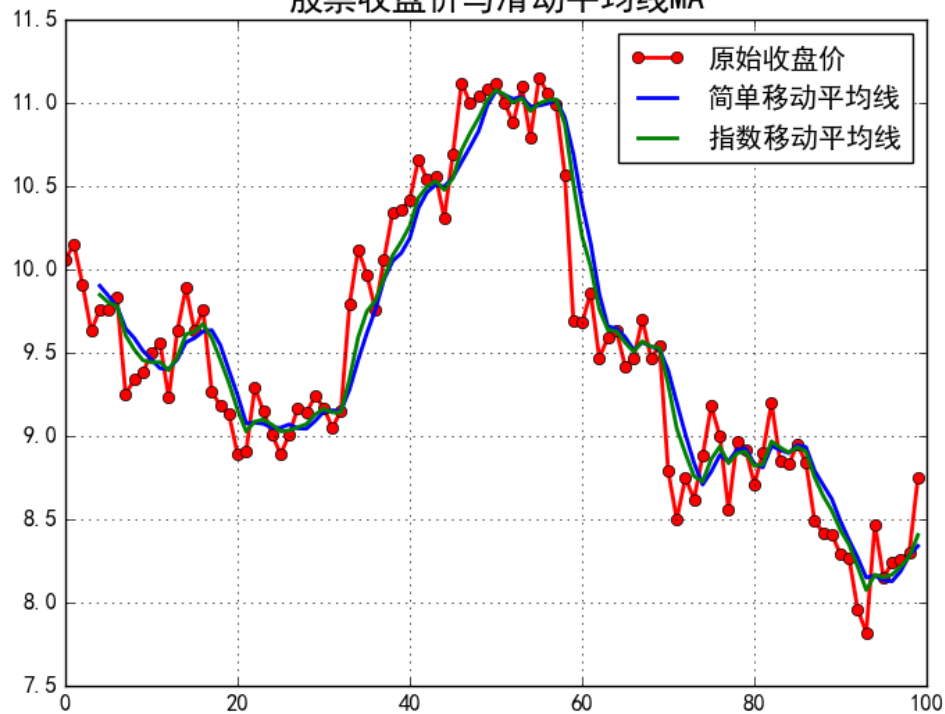
svd_48.png



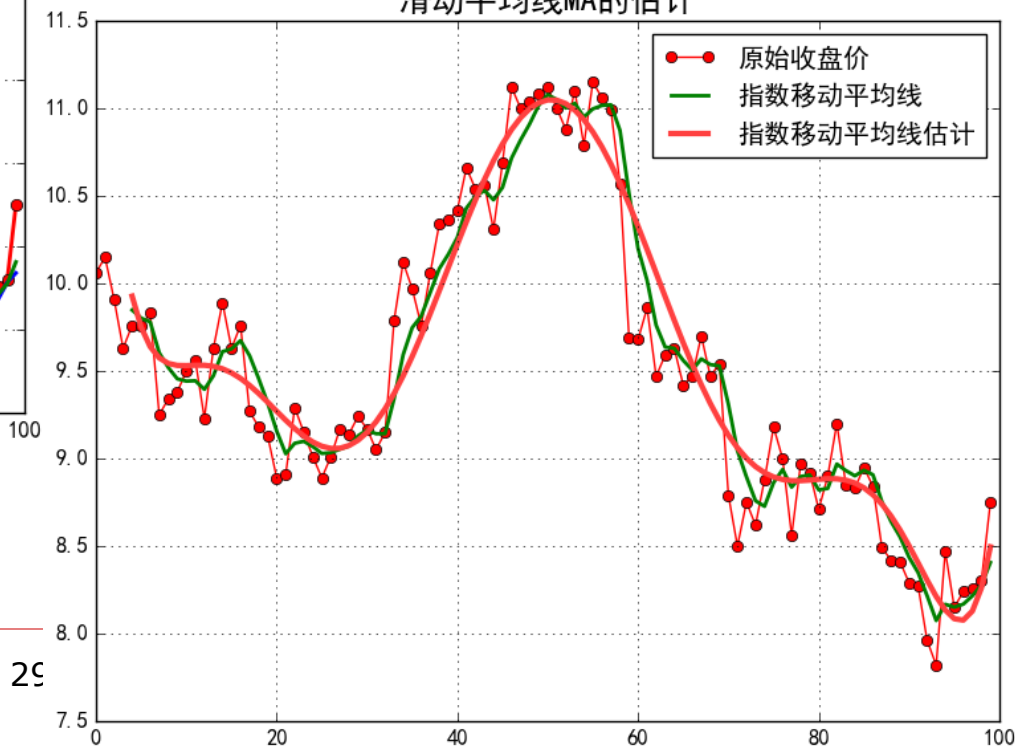
svd_49.png

某股票收盘数据处理

股票收盘价与滑动平均线MA



滑动平均线MA的估计

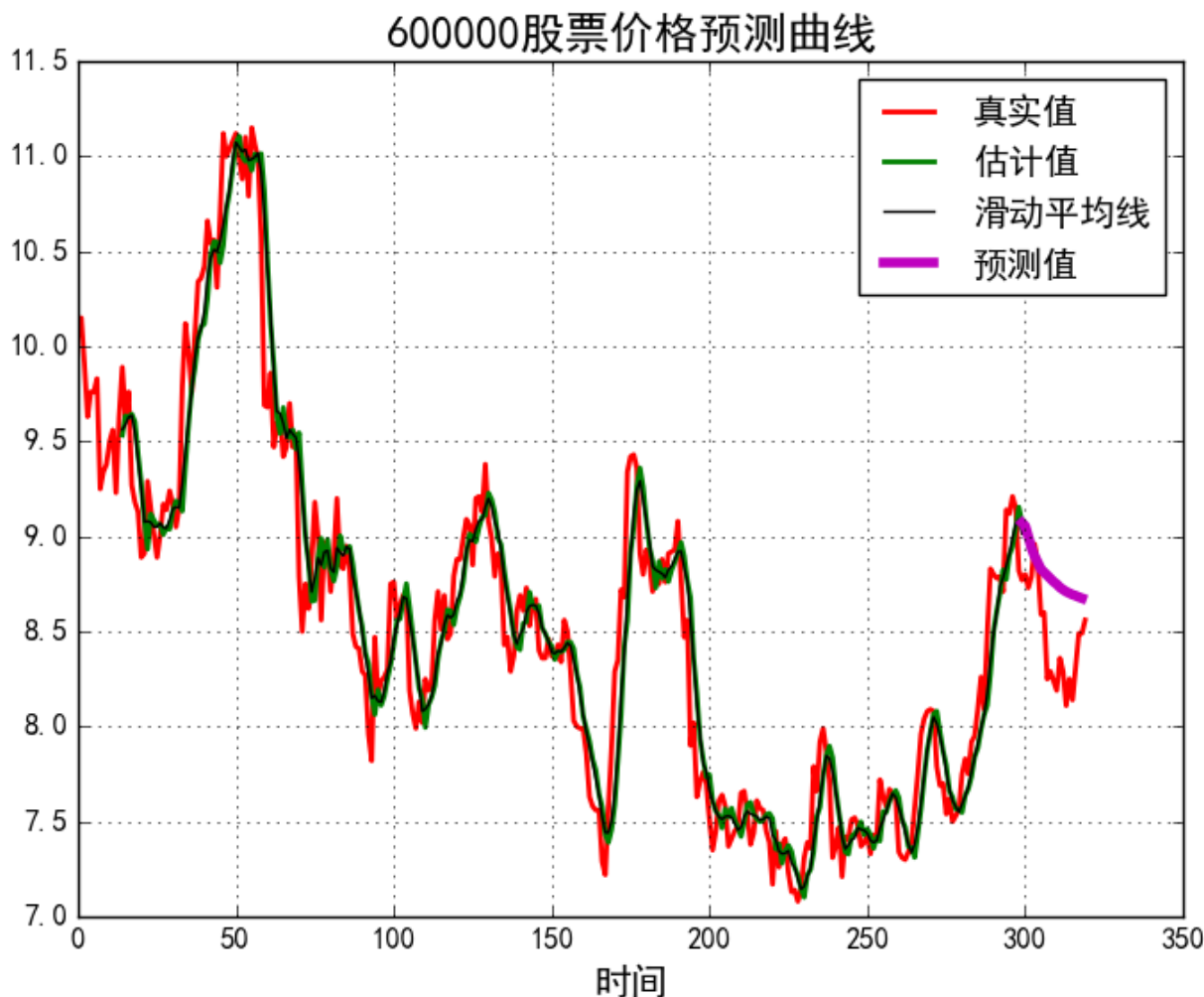


Code

```
K = 5
weight = np.ones(K)
weight /= weight.sum()
y_avg = np.convolve(y, weight)
np.set_printoptions(linewidth=100)
N = len(y_avg)
lag = 10
x = np.zeros((N-lag, lag))
for i in range(N-lag):
    x[i] = y_avg[i:i+lag]
y_avg = y_avg[lag:]
model = RidgeCV(alphas=np.logspace(-4, 4, 100))
model.fit(x, y_avg)
y_pred = model.predict(x)
```

```
window = x[-1]
m = 20
y_next = np.zeros(m)
for i in range(m):
    y_next[i] = model.predict(window.reshape(1, -1))
    window = np.delete(window, 0)
    window = np.append(window, y_next[i])
y_next = np.insert(y_next, 0, y_pred[-1])
```

股票价格



图像的卷积



laplacian.png



laplacian2.png



prewitt.png



prewitt_x.png



prewitt_y.png



soble.png



soble_x.png



soble_y.png

图像的卷积



laplacian.png



laplacian2.png



prewitt.png



prewitt_x.png



prewitt_y.png



soble.png



soble_x.png



soble_y.png

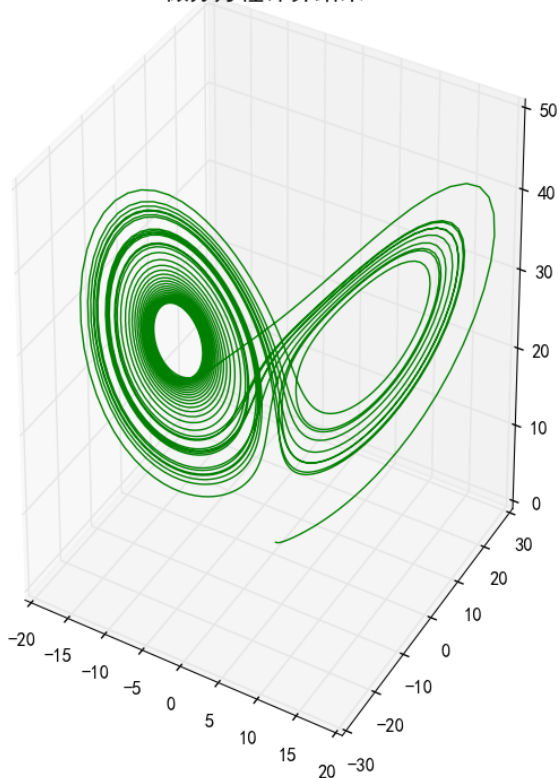
Code

```
def convolve(image, weight):
    height, width = image.shape
    h, w = weight.shape
    height_new = height - h + 1
    width_new = width - w + 1
    image_new = np.zeros((height_new, width_new), dtype=np.float)
    for i in range(height_new):
        for j in range(width_new):
            image_new[i,j] = np.sum(image[i:i+h, j:j+w] * weight)
    image_new = image_new.clip(0, 255)
    image_new = np rint(image_new).astype('uint8')
    return image_new
```

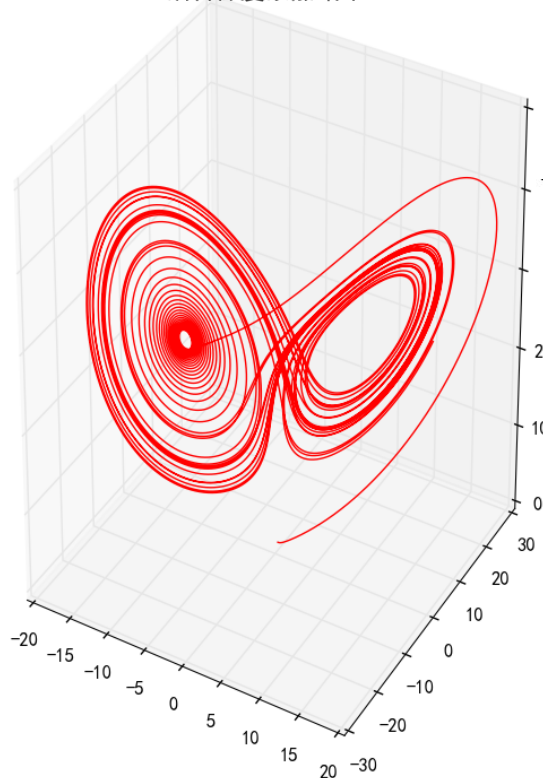
Lorenz方程

Lorenz系统

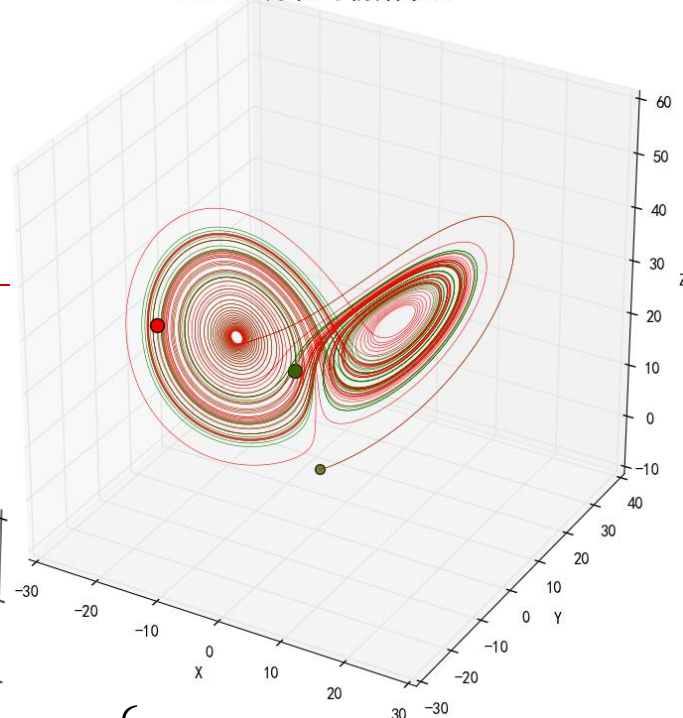
微分方程计算结果



沿着梯度累加结果



Lorenz方程与初始条件



$$\begin{cases} \frac{\partial f}{\partial x} = \sigma(y - x) \\ \frac{\partial f}{\partial y} = x(\rho - z) - y \\ \frac{\partial f}{\partial z} = x \cdot y - \beta \cdot z \end{cases}$$

Code

```
s0 = (0., 1., 0.)
t = np.arange(0, 30, 0.01)
s = odeint(lorenz, s0, t)
plt.figure(figsize=(12, 8), facecolor='w')
plt.subplot(121, projection='3d')
plt.plot(s[:, 0], s[:, 1], s[:, 2], c='g')
plt.title(u'微分方程计算结果', fontsize=16)
```

```
s = lorenz_trajectory(s0, 40000)
plt.subplot(122, projection='3d')
plt.plot(s[:, 0], s[:, 1], s[:, 2], c='r')
plt.title(u'沿着梯度累加结果', fontsize=16)
```

```
plt.tight_layout(1, rect=(0,0,1,0.98))
plt.suptitle(u'Lorenz系统', fontsize=20)
plt.show()
```

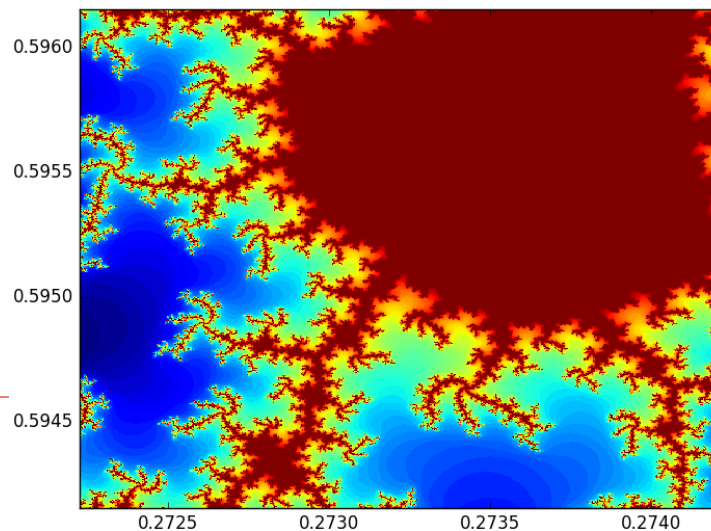
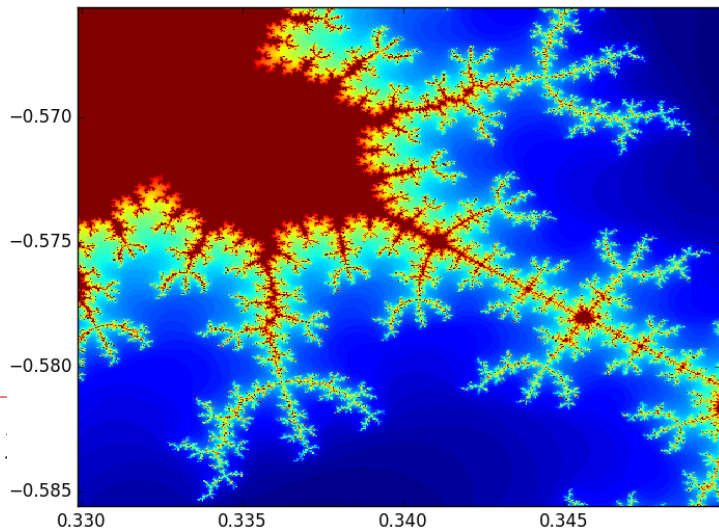
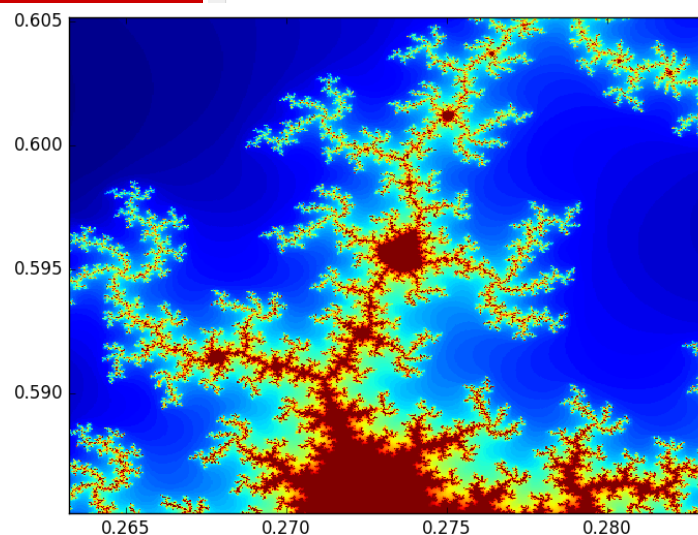
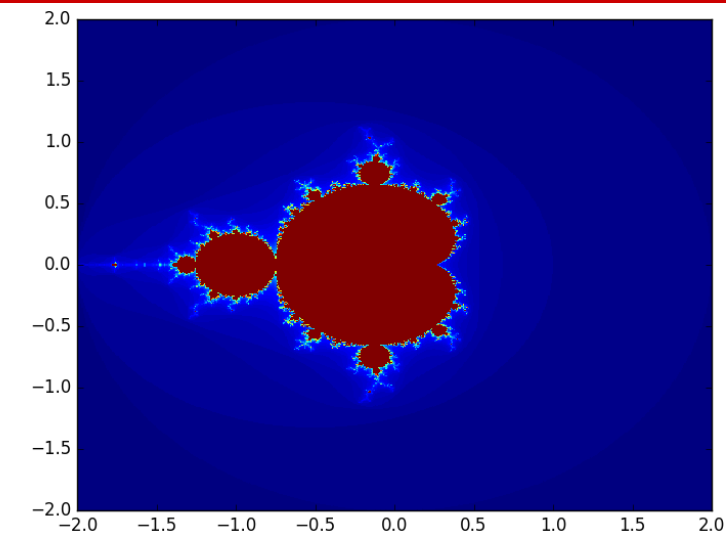
```
ax = Axes3D(plt.figure(figsize=(8, 8)))
s0 = (0., 1., 0.)
s1 = lorenz_trajectory(s0, 50000)
s0 = (0., 1.0001, 0.)
s2 = lorenz_trajectory(s0, 50000)
# 曲线
ax.plot(s1[:, 0], s1[:, 1], s1[:, 2], c='g', lw=0.4)
ax.plot(s2[:, 0], s2[:, 1], s2[:, 2], c='r', lw=0.4)
# 起点
ax.scatter(s1[0, 0], s1[0, 1], s1[0, 2], c='g', s=50, alpha=0.5)
ax.scatter(s2[0, 0], s2[0, 1], s2[0, 2], c='r', s=50, alpha=0.5)
# 终点
ax.scatter(s1[-1, 0], s1[-1, 1], s1[-1, 2], c='g', s=100)
ax.scatter(s2[-1, 0], s2[-1, 1], s2[-1, 2], c='r', s=100)
ax.set_title(u'Lorenz方程与初始条件', fontsize=20)
ax.set_xlabel(u'X')
ax.set_ylabel(u'Y')
ax.set_zlabel(u'Z')
plt.show()
```

Mandelbrot Set

- 1967年，美国数学家Mandelbrot提出著名的“英格兰的海岸线到底有多长”问题，该问题对欧氏几何提出挑战。1975年，Mandelbrot在其《自然界中的分形几何》著作中引入了分形(fractal)的概念。
- Mandelbrot集的定义：
 - 对于某常数 c ，定义函数 $f(z;c) = z^2 + c$
 - 取复数 $z=0$ ，计算序列 $a = \{0, f(0), f(f(0)), f(f(f(0))), \dots\}$
 - 若序列的模不发散到无穷大，则 c 属于Mandelbrot集。
- 例如：
 - c 为1时，序列发散，则 $1 \notin M$ $a_{c=1} = \{0, 1, 2, 5, 26, 677, \dots\}$
 - c 为 i 时，序列不发散，则 $i \in M$ $a_{c=i} = \{0, i, -1+i, -i, -1+i, -i, \dots\}$

Mandelbrot集

```
def divergent(c):  
    z = 0  
    i = 0  
    while i < 100:  
        z = z**2 + c  
        if abs(z) > 2:  
            break  
        i += 1  
    return i
```



作业

- 实现任何一个函数曲线/曲面的Python显示。
 - 如Mandelbrot集/Julia集/Koch曲线/Sierpinski垫片
- 尝试使用SVD实现图像处理和特征提取。
- 熟悉Python的Numpy/Scipy数值计算数学库。

附：MD5

- MD5(Message Digest Algorithm)，消息摘要算法，为计算机安全领域广泛使用的一种散列函数，用以提供消息的完整性保护。
 - 文件号RFC 1321(R.Rivest,MIT Laboratory for Computer Science and RSA Data Security Inc. April 1992)
- 对于任意长度的信息，经过MD5算法，得到长度为128bit的摘要。

MD5的框架理解

□ 对于长度为512bit的信息，可以通过处理，得到长度为128bit的摘要。

□ 初始化摘要：

0x0123456789ABCDEF FEDCBA9876543210

■ A=0x01234567 B=0x89ABCDEF

■ C=0xFEDCBA98 D=0x76543210

□ 现在的工作，是要用长度为512位的信息，变换初始摘要。

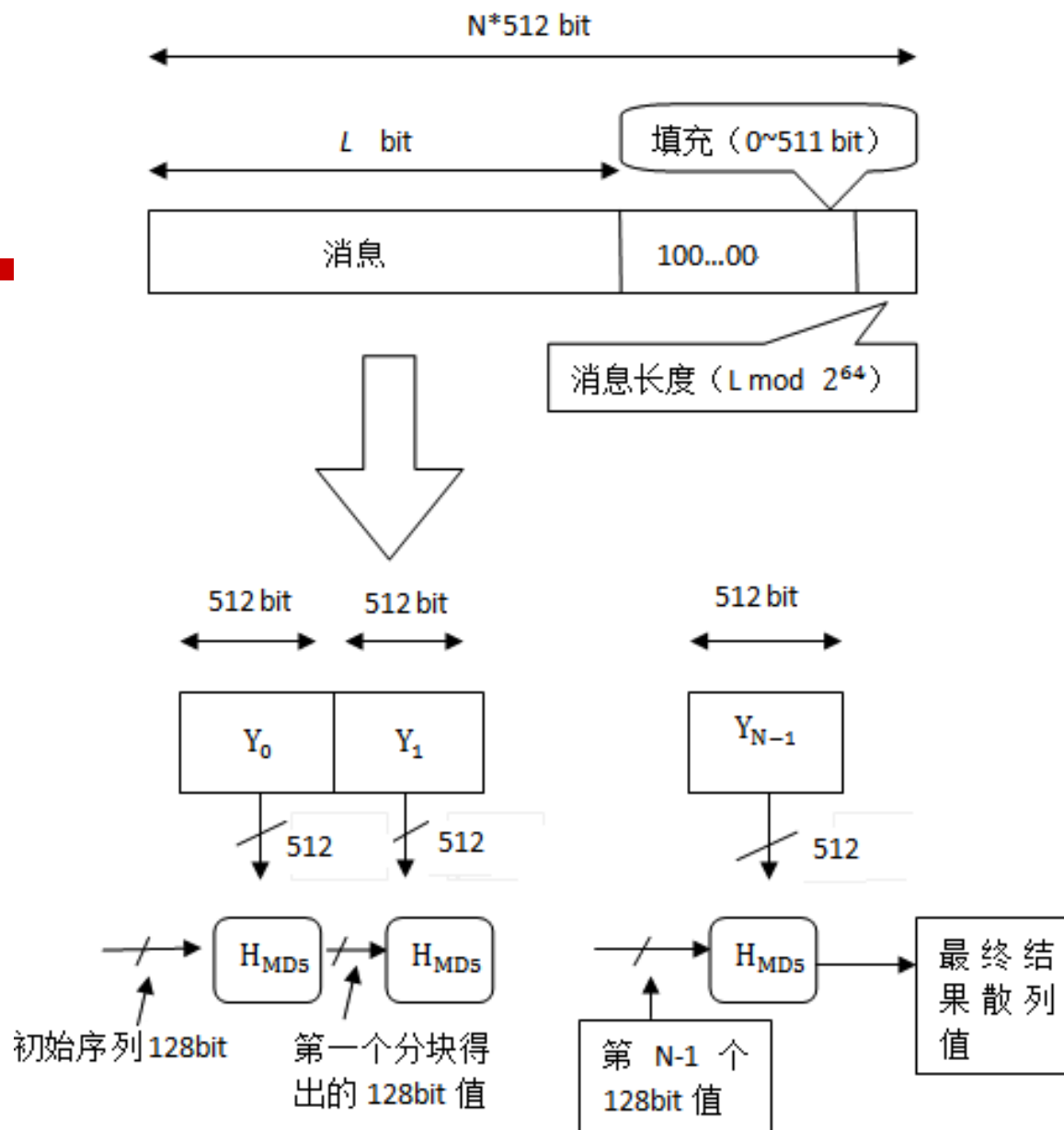
MD5的总体理解

- 定义变量 a, b, c, d , 分别记录 A, B, C, D ;
- 将512bit的信息按照32bit一组, 分成16组;
分别记为 M_j ($0 \leq j \leq 15$);
- 取某正数 s 、 t_k , 定义函数:
$$FF(a, b, c, d, M_j, s, t_k) = (a + F(b, c, d) + M_j + t_k) \ll s$$
- 利用 M_j 分别进行信息提取, 将结果保存到 a
 - 其中, $F(X, Y, Z) = (X \& Y) \mid (\sim X \& Z)$

MD5的总体理解

- 经过以上16次变换， a, b, c, d 带有了 M_j 的信息
- 事实上经过四轮这样的变换($4\text{轮} \times 16\text{次} = 64\text{次}$)
- 经过64次变换后，将 a, b, c, d 累加给 A, B, C, D
- 此时，完成了512bit信息的提取；进行下一个512bit信息的相同操作

MD5框架



MD5细致算法

- 在算法中，首先需要对信息进行填充，使其长度对512求余的结果等于448。
- 填充的方法如下，在信息的后面填充一个1和若干个0，直到满足上面的条件。然后，在这个结果后面附加一个以64位二进制表示的原始信息长度Length。
- 经过这两步的处理，数据总长度为 $N*512+448+64=(N+1)*512$ ，即长度恰好是512的整数倍。

MD5的细致算法

□ 定义四个非线性函数：

■ $F(X, Y, Z) = (X \& Y) | ((\sim X) \& Z)$ $H(X, Y, Z) = X \wedge Y \wedge Z$

■ $G(X, Y, Z) = (X \& Z) | (Y \& (\sim Z))$ $I(X, Y, Z) = Y \wedge (X | (\sim Z))$

□ M_j 表示数据的第 j 个子分组 ($0 \leq j \leq 15$, 循环表示), 常数 t_k 是 $2^{32} * \text{abs}(\sin(k))$ 的整数部分, $1 \leq k \leq 64$, 单位是弧度。

□ $FF(a, b, c, d, M_j, s, t_k) = (a + F(b, c, d) + M_j + t_k) \ll s$

□ $GG(a, b, c, d, M_j, s, t_k) = (b + G(b, c, d) + M_j + t_k) \ll s$

□ $HH(a, b, c, d, M_j, s, t_k) = (b + H(b, c, d) + M_j + t_k) \ll s$

□ $II(a, b, c, d, M_j, s, t_k) = (b + I(b, c, d) + M_j + t_k) \ll s$

64次操作

第一轮

FF(a, b, c, d, M0, 7, 0xd76aa478)
FF(d, a, b, c, M1, 12, 0xe8c7b756)
FF(c, d, a, b, M2, 17, 0x242070db)
FF(b, c, d, a, M3, 22, 0xc1bdceee)
FF(a, b, c, d, M4, 7, 0xf57c0faf)
FF(d, a, b, c, M5, 12, 0x4787c62a)
FF(c, d, a, b, M6, 17, 0xa8304613)
FF(b, c, d, a, M7, 22, 0xfd469501)
FF(a, b, c, d, M8, 7, 0x698098d8)
FF(d, a, b, c, M9, 12, 0x8b44f7af)
FF(c, d, a, b, M10, 17, 0xfffff5bb1)
FF(b, c, d, a, M11, 22, 0x895cd7be)
FF(a, b, c, d, M12, 7, 0x6b901122)
FF(d, a, b, c, M13, 12, 0xfd987193)
FF(c, d, a, b, M14, 17, 0xa679438e)
FF(b, c, d, a, M15, 22, 0x49b40821)

第二轮

GG(a, b, c, d, M1, 5, 0xf61e2562)
GG(d, a, b, c, M6, 9, 0xc040b340)
GG(c, d, a, b, M11, 14, 0x265e5a51)
GG(b, c, d, a, M0, 20, 0xe9b6c7aa)
GG(a, b, c, d, M5, 5, 0xd62f105d)
GG(d, a, b, c, M10, 9, 0x02441453)
GG(c, d, a, b, M15, 14, 0xd8a1e681)
GG(b, c, d, a, M4, 20, 0xe7d3fbc8)
GG(a, b, c, d, M9, 5, 0x21e1cde6)
GG(d, a, b, c, M14, 9, 0xc33707d6)
GG(c, d, a, b, M3, 14, 0xf4d50d87)
GG(b, c, d, a, M8, 20, 0x455a14ed)
GG(a, b, c, d, M13, 5, 0xa9e3e905)
GG(d, a, b, c, M2, 9, 0xfcefa3f8)
GG(c, d, a, b, M7, 14, 0x676f02d9)
GG(b, c, d, a, M12, 20, 0x8d2a4c8a)

第三轮

HH(a, b, c, d, M5, 4, 0xffffa3942)
HH(d, a, b, c, M8, 11, 0x8771f681)
HH(c, d, a, b, M11, 16, 0x6d9d6122)
HH(b, c, d, a, M14, 23, 0xfde5380c)
HH(a, b, c, d, M1, 4, 0xa4beea44)
HH(d, a, b, c, M4, 11, 0x4bdecfa9)
HH(c, d, a, b, M7, 16, 0xf6bb4b60)
HH(b, c, d, a, M10, 23, 0xbebfbcb70)
HH(a, b, c, d, M13, 4, 0x289b7ec6)
HH(d, a, b, c, M0, 11, 0xeaal27fa)
HH(c, d, a, b, M3, 16, 0xd4ef3085)
HH(b, c, d, a, M6, 23, 0x04881d05)
HH(a, b, c, d, M9, 4, 0xd9d4d039)
HH(d, a, b, c, M12, 11, 0xe6db99e5)
HH(c, d, a, b, M15, 16, 0x1fa27cf8)
HH(b, c, d, a, M2, 23, 0xc4ac5665)

第四轮

II(a, b, c, d, M0, 6, 0xf4292244)
II(d, a, b, c, M7, 10, 0x432aff97)
II(c, d, a, b, M14, 15, 0xab9423a7)
II(b, c, d, a, M5, 21, 0xfc93a039)
II(a, b, c, d, M12, 6, 0x655b59c3)
II(d, a, b, c, M3, 10, 0x8f0ccc92)
II(c, d, a, b, M10, 15, 0xffefff47d)
II(b, c, d, a, M1, 21, 0x85845dd1)
II(a, b, c, d, M8, 6, 0x6fa87e4f)
II(d, a, b, c, M15, 10, 0xfe2ce6e0)
II(c, d, a, b, M6, 15, 0xa3014314)
II(b, c, d, a, M13, 21, 0x4e0811a1)
II(a, b, c, d, M4, 6, 0xf7537e82)
II(d, a, b, c, M11, 10, 0xbd3af235)
II(c, d, a, b, M2, 15, 0x2ad7d2bb)
II(b, c, d, a, M9, 21, 0xeb86d391)

我们在这里

□ <http://wenda.ChinaHadoop.cn>

■ 视频/课程/社区

□ 微博

■ @ChinaHadoop

■ @邹博_机器学习

□ 微信公众号

■ 小象学院

■ 大数据分析挖掘

互联网新技术在线教育领航者

小象问答 搜索标题、用户 全站内容搜索 提问 首页 动态 发现 话题 通知

全部 招聘求职 机器学习 大数据平台技术 DCon 大数据行业应用 NoSQL数据库 数据科学 江湖救急

发现 最新 推荐 热门 等待回复

graphviz has no attribute 'write' 贡献
邹博 回复了问题 • 2 人关注 • 1 个回复 • 3 次浏览 • 2017-04-09 15:48

sklearn中如何理解Pipeline机制 贡献
数据分析与数据挖掘 邹博 回复了问题 • 2 人关注 • 1 个回复 • 28 次浏览 • 2017-04-09 15:39

关于9.Logistic回归的ppt中第9页的对数线性函数 贡献
机器学习 邹博 回复了问题 • 3 人关注 • 3 个回复 • 39 次浏览 • 2017-04-09 15:35

关于“贝叶斯估计中，最大后验概率估计就是结构化风险最小化的例子：当模型是条件概率分布，损失函数为对数损失函数，模型的复杂度由模型的先验概率表示，结构化风险最小化就等价于最大后验概率估计” 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 26 次浏览 • 2017-04-09 15:27

关于连续值的预测 贡献
咨询 邹博 回复了问题 • 2 人关注 • 1 个回复 • 31 次浏览 • 2017-04-09 15:24

拉格朗日对偶函数为什么一定是凸函数 贡献
数据科学 邹博 回复了问题 • 2 人关注 • 2 个回复 • 26 次浏览 • 2017-04-09 15:20

梯度下降公式中的斯堪J 是 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 29 次浏览 • 2017-04-09 15:17

深度学习适合做预测吗？ 贡献
深度学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 27 次浏览 • 2017-04-09 15:15

关于6.4PCA_FeatureSelection.py中plt.legend的参数疑问 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 28 次浏览 • 2017-04-09 15:04

@邹博 有哪些可以下载数据源的网站？ 贡献
数据分析与数据挖掘 邹博 回复了问题 • 4 人关注 • 1 个回复 • 31 次浏览 • 2017-04-09 14:53

LDA主题模型 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 29 次浏览 • 2017-04-09 14:45

代码10.6bagging_ridged老师提到了采样率设为0.2能够使峰值部分的数据被体现出来。这是为什么呢？ 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 22 次浏览 • 2017-04-09 14:26

GraphViz's executables not found 贡献
机器学习 邹博 回复了问题 • 3 人关注 • 2 个回复 • 23 次浏览 • 2017-04-09 13:47

决策树中关于feature_importances代码的问题 贡献
机器学习 邹博 回复了问题 • 2 人关注 • 1 个回复 • 6 次浏览 • 2017-04-09 13:11

专题
招聘求职
大数据行业应用
数据科学
系统与编程
云计算技术

热门话题 更多 >
机器学习 907 个问题, 230 人关注
spark 387 个问题, 172 人关注
hadoop 1059 个问题, 155 人关注
python数据分析 171 个问题, 28 人关注
数据分析与数据挖掘 54 个问题, 111 人关注

热门用户 更多 >
小心巴 14 个问题, 0 次赞同
叉叉V 45 个问题, 22 次赞同
铁甲无声 10 个问题, 0 次赞同
带刀锦衣卫 13 个问题, 0 次赞同

感谢大家！

恳请大家批评指正！

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象学院
- 新浪微博：小象AI学院

