法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，小象学院和主讲老师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、仿造其中的创意及内容，我们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
  - 新浪微博：ChinaHadoop

Python库的使用

邹博

本次说明

- 本PPT后面仅列举使用Python库的效果截图，详细内容请参考该PPT的配套代码。

Python库

- Pip
  - 安装Python包的推荐工具：https://pypi.python.org/pypi/pip
- Numpy
  - 为Python提供快速的多维数组处理能力
- Scipy
  - 在NumPy基础上添加了众多科学计算工具包
- Matplotlib
  - Python丰富的绘图库
- 官网：
  - http://www.scipy.org
  - http://www.matplotlib.org

数据生成

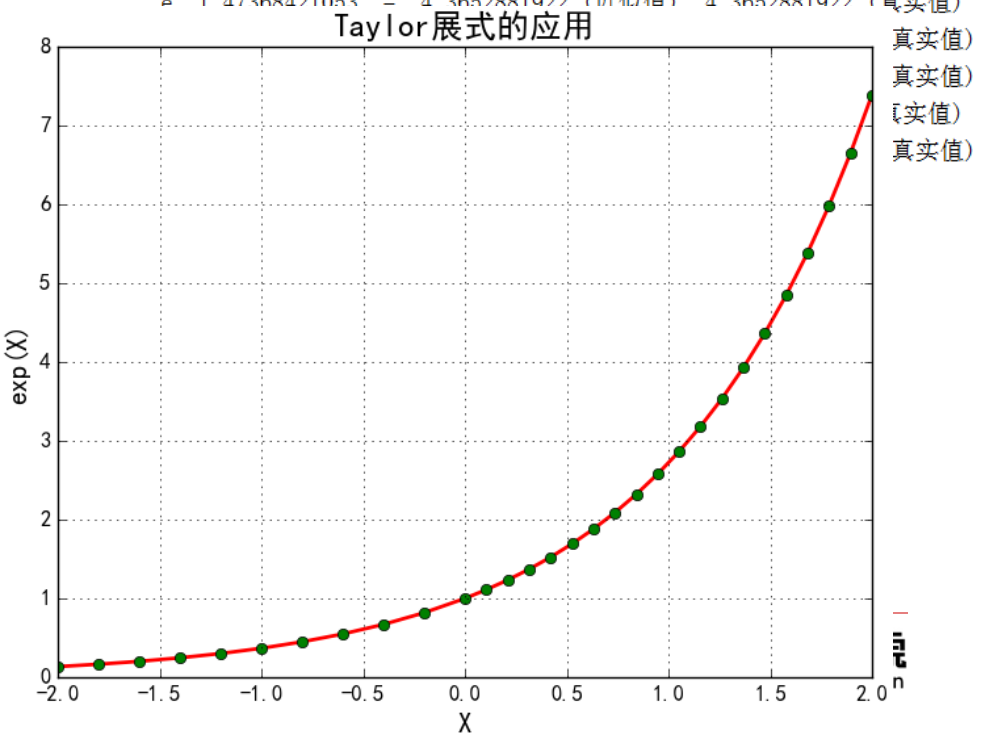
- *a = np.arange(0, 60, 10).reshape((-1, 1)) + np.arange(6)*
- A=

```
[[ 0  1  2  3  4  5]
[10 11 12 13 14 15]
[20 21 22 23 24 25]
[30 31 32 33 34 35]
[40 41 42 43 44 45]
[50 51 52 53 54 55]]
```

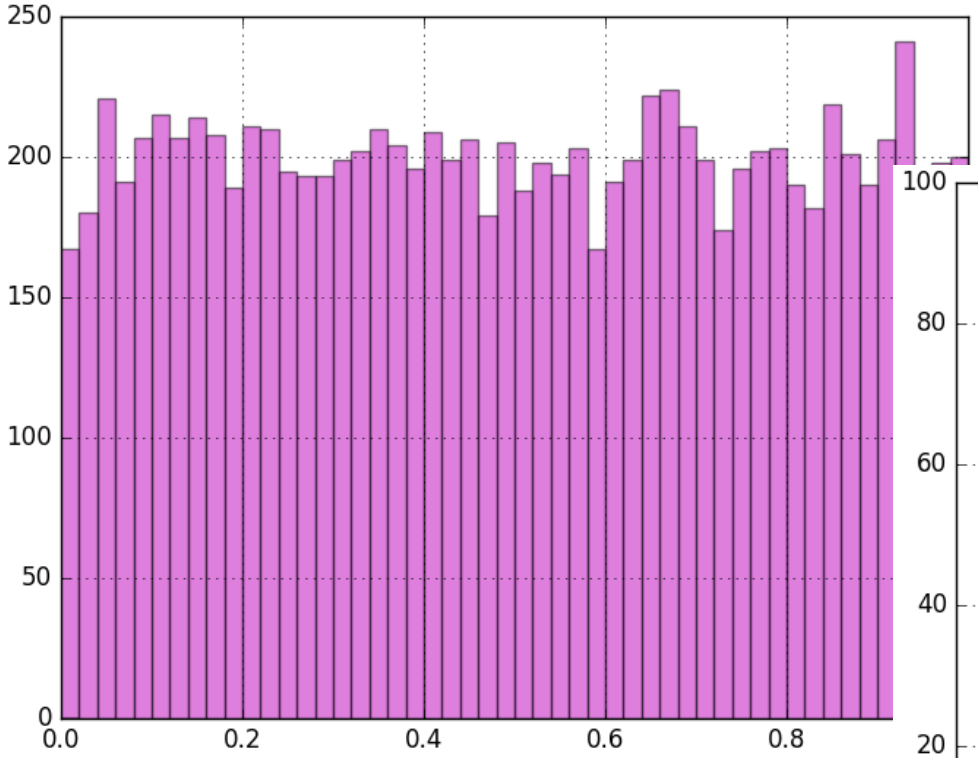
Taylor展式的应用

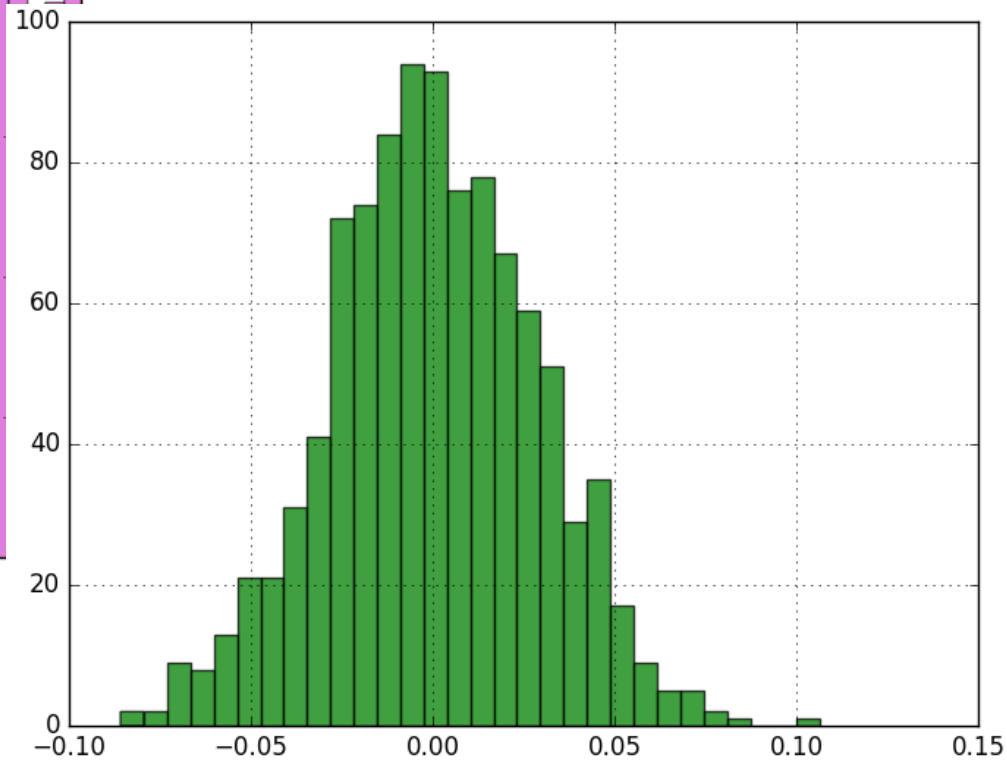
```
e^ -0.8  =  0.449328964117 (近似值) 0.449328964117 (真实值)
e^ -0.6  =  0.548811636094 (近似值) 0.548811636094 (真实值)
e^ -0.4  =  0.670320046036 (近似值) 0.670320046036 (真实值)
e^ -0.2  =  0.818730753078 (近似值) 0.818730753078 (真实值)
e^ 0.0  =  1.0 (近似值) 1.0 (真实值)
e^ 0.105263157895  =  1.11100294108 (近似值)  1.11100294108 (真实值)
e^ 0.210526315789  =  1.2343275351 (近似值) 1.2343275351 (真实值)
e^ 0.315789473684  =  1.37134152176 (近似值)  1.37134152176 (真实值)
e^ 0.421052631579  =  1.5235644639 (近似值) 1.5235644639 (真实值)
e^ 0.526315789474  =  1.69268460033 (近似值)  1.69268460033 (真实值)
e^ 0.631578947368  =  1.88057756929 (近似值)  1.88057756929 (真实值)
e^ 0.736842105263  =  2.08932721042 (近似值)  2.08932721042 (真实值)
e^ 0.842105263158  =  2.32124867566 (近似值)  2.32124867566 (真实值)
e^ 0.947368421053  =  2.57891410565 (近似值)  2.57891410565 (真实值)
e^ 1.05263157895  =  2.86518115618 (近似值) 2.86518115618 (真实值)
e^ 1.15789473684  =  3.18322469126 (近似值) 3.18322469126 (真实值)
e^ 1.26315789474  =  3.53657199412 (近似值) 3.53657199412 (真实值)
e^ 1.36842105263  =  3.92914188683 (近似值) 3.92914188683 (真实值)
e^ 1.47368421053  =  4.3652881922 (近似值)  4.3652881922 (真实值)
```

```python
def calc_e_small(x):
    n = 10
    f = np.arange(1, n+1).cumprod()
    b = np.array([x]*n).cumprod()
    return np.sum(b / f) + 1


def calc_e(x):
    reverse = False
    if x < 0:   # 处理负数
        x = -x
        reverse = True
    ln2 = 0.69314718055994530941723212145818
    c = x / ln2
    a = int(c+0.5)
    b = x - a*ln2
    y = (2 ** a) * calc_e_small(b)
    if reverse:
        return 1/y
    return y


if __name__ == "__main__":
    t1 = np.linspace(-2, 0, 10, endpoint=False)
    t2 = np.linspace(0, 2, 20)
    t = np.concatenate((t1, t2))
    print t     # 横轴数据
    y = np.empty_like(t)
    for i, x in enumerate(t):
        y[i] = calc_e(x)
        print 'e^', x, ' = ', y[i], '(近似值)\t', math.exp(x)
        # print '误差: ', y[i] - math.exp(x)
    mpl.rcParams['font.sans-serif'] = [u'SimHei']
    mpl.rcParams['axes.unicode_minus'] = False
    plt.plot(t, y, 'r-', t, y, 'go', linewidth=2)
    plt.title(u'Taylor展式的应用', fontsize=18)
    plt.xlabel('X', fontsize=15)
    plt.ylabel('exp(X)', fontsize=15)
    plt.grid(True)
    plt.show()
```

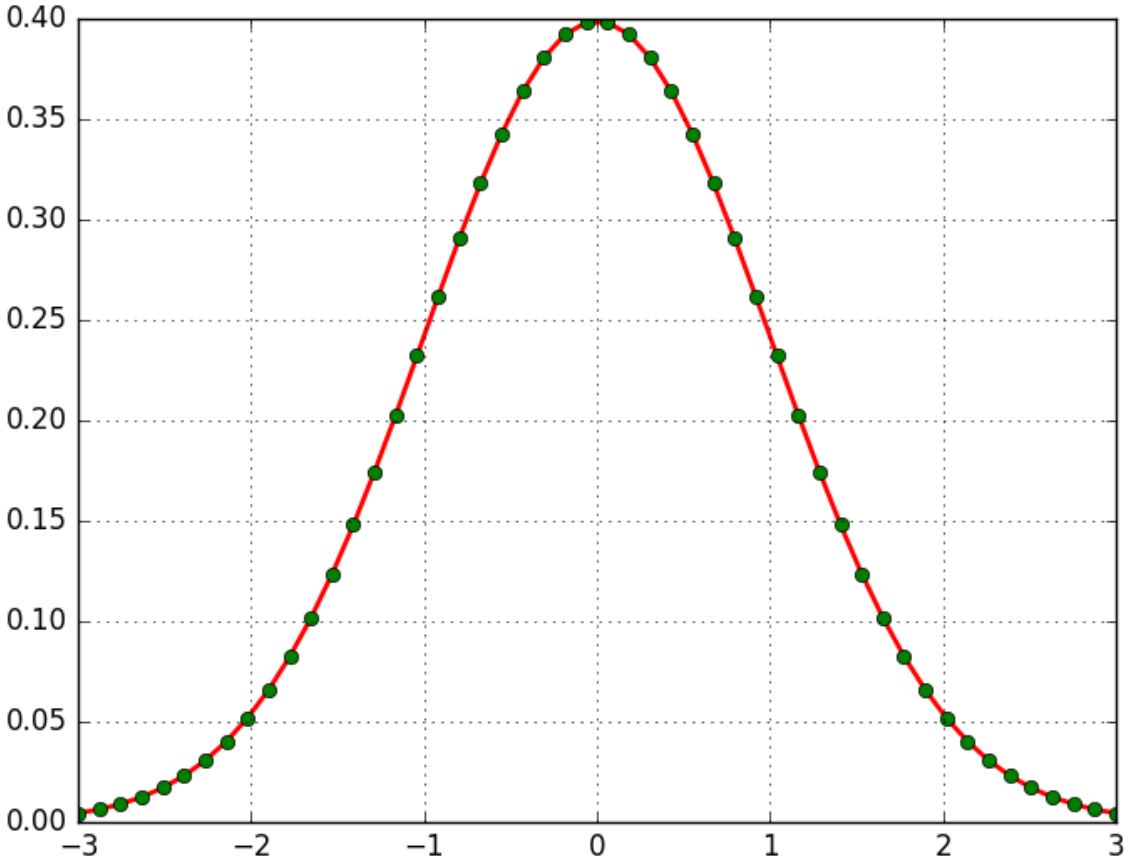


验证中心极限定理

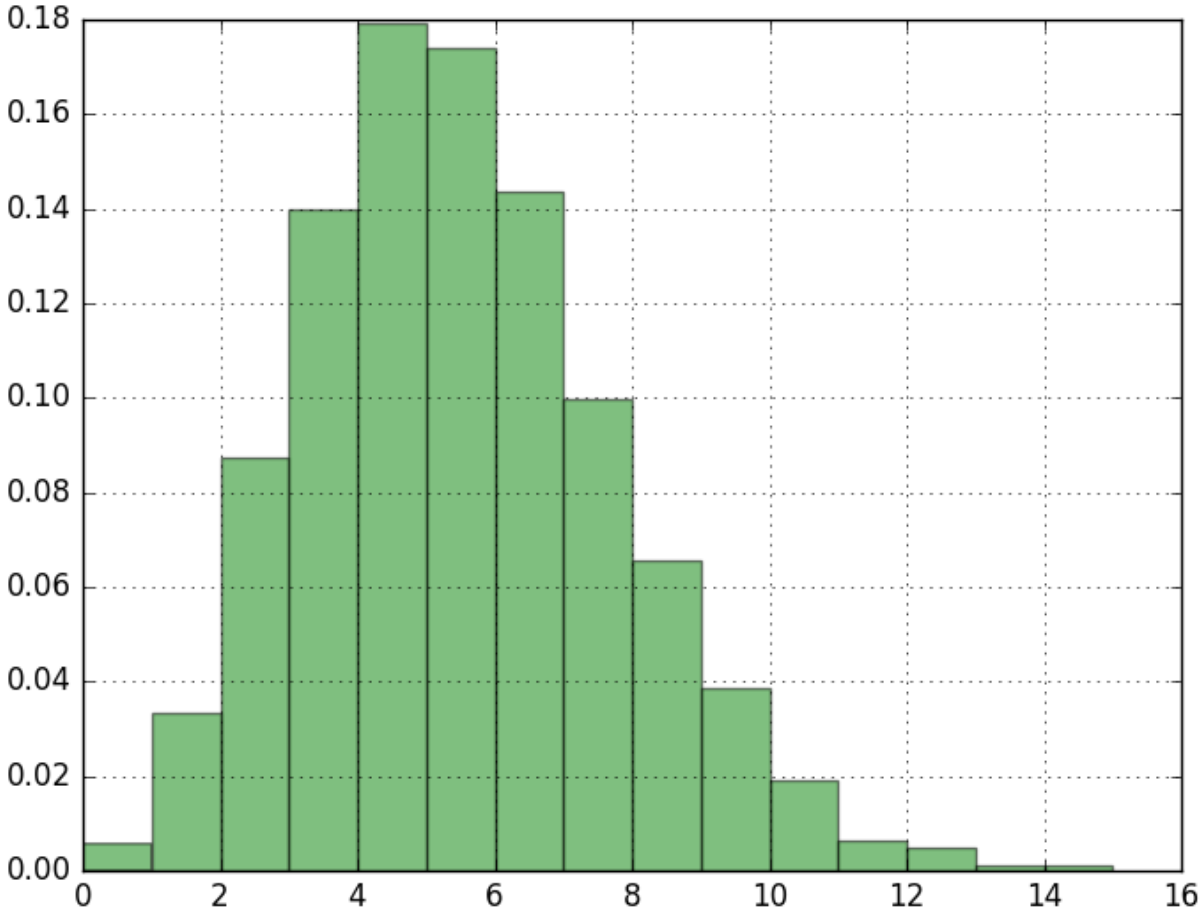

250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8


100
80
60
40
20
0
−0.10
−0.05
0.00
0.05
0.10
0.15

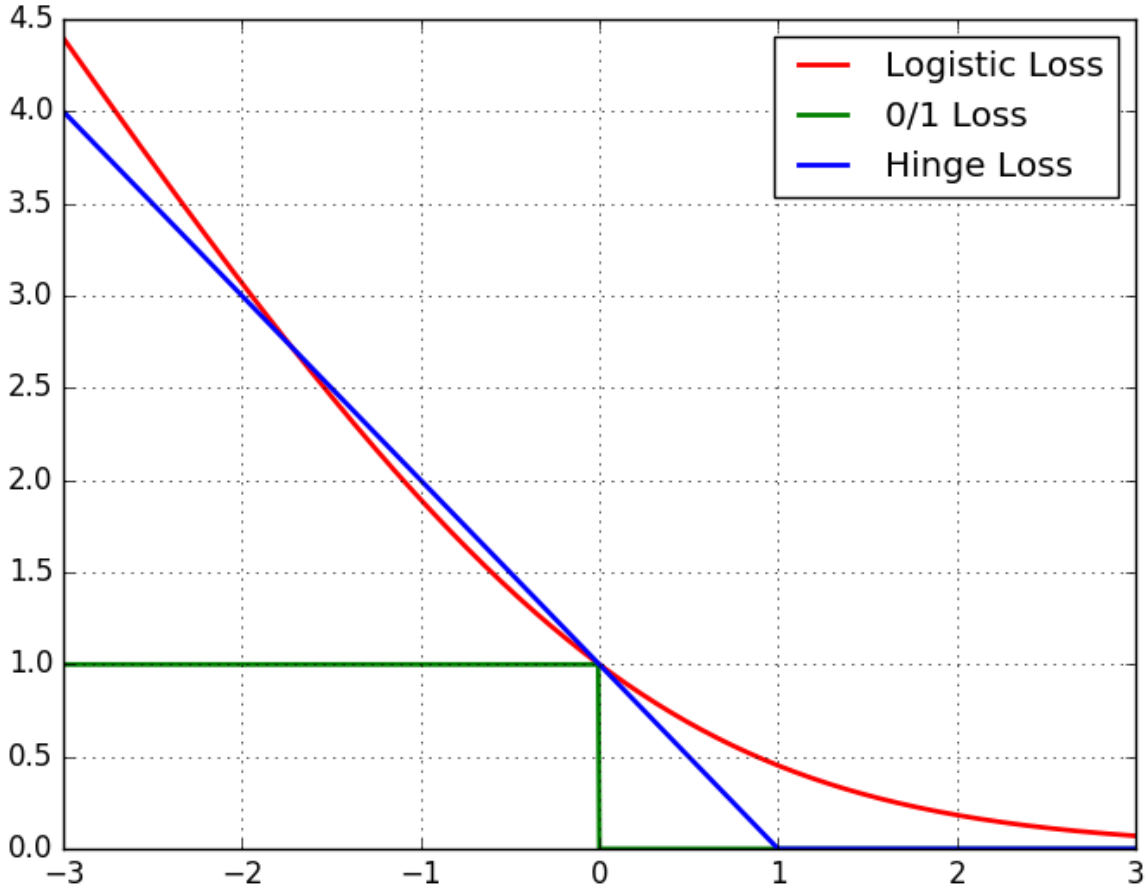
正态分布的概率密度函数



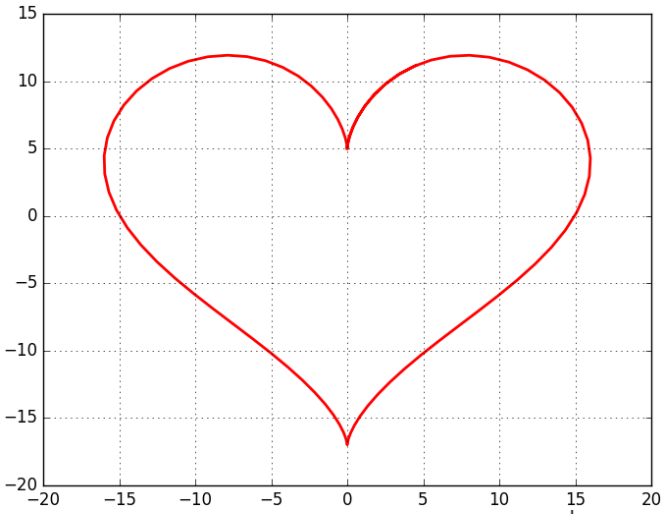
Poisson分布的概率质量函数

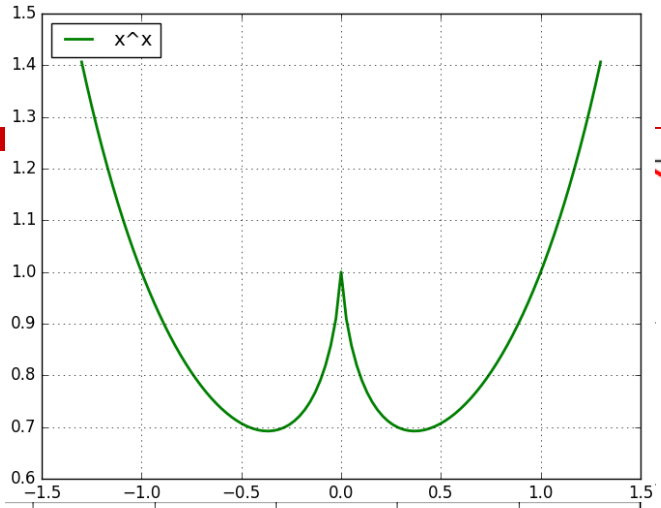


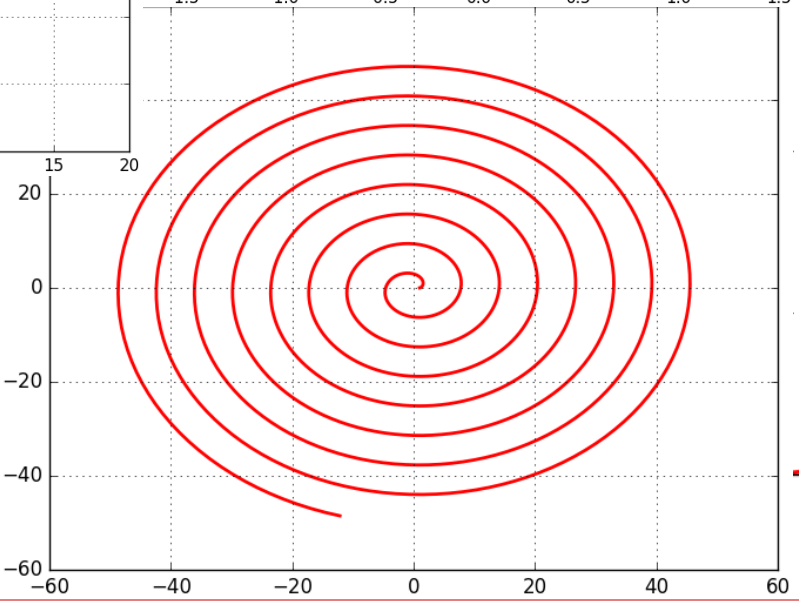
机器学习中的损失函数

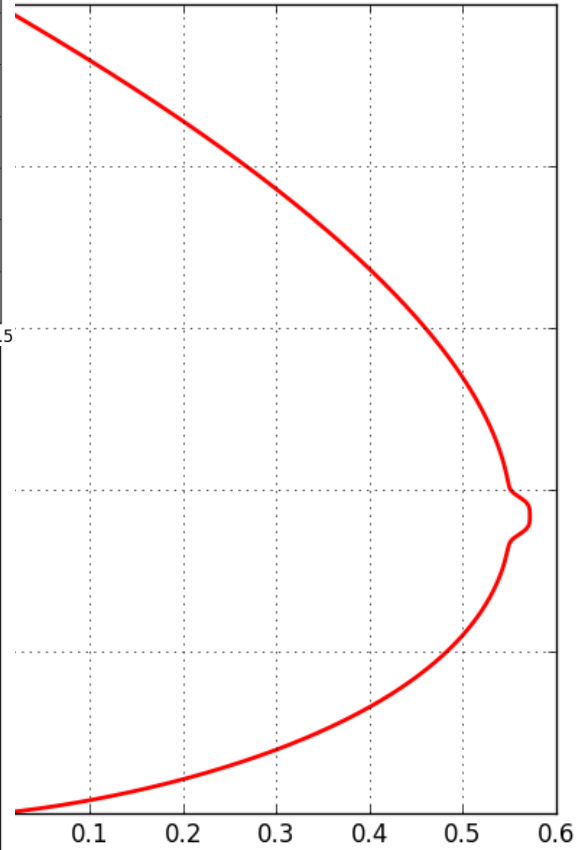

Logistic Loss
0/1 Loss
Hinge Loss
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−3
−2
−1
0
1
2
3

各种2D曲线

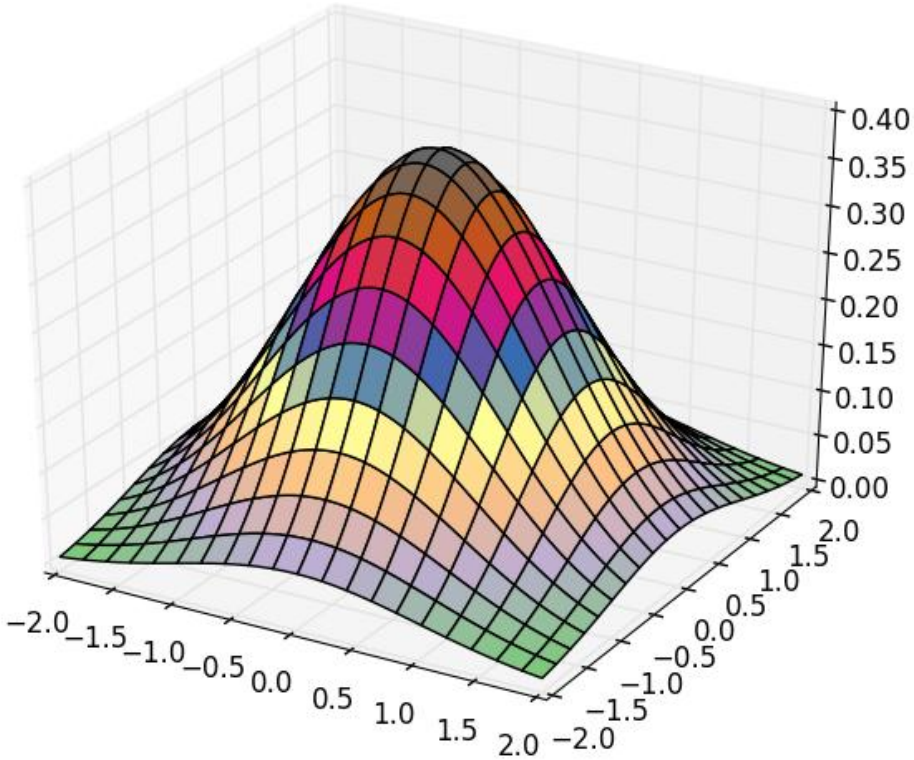

15
10
5
0
−5
−10
−15
−20
−20
−15
−10
−5
0
5
10
15
20

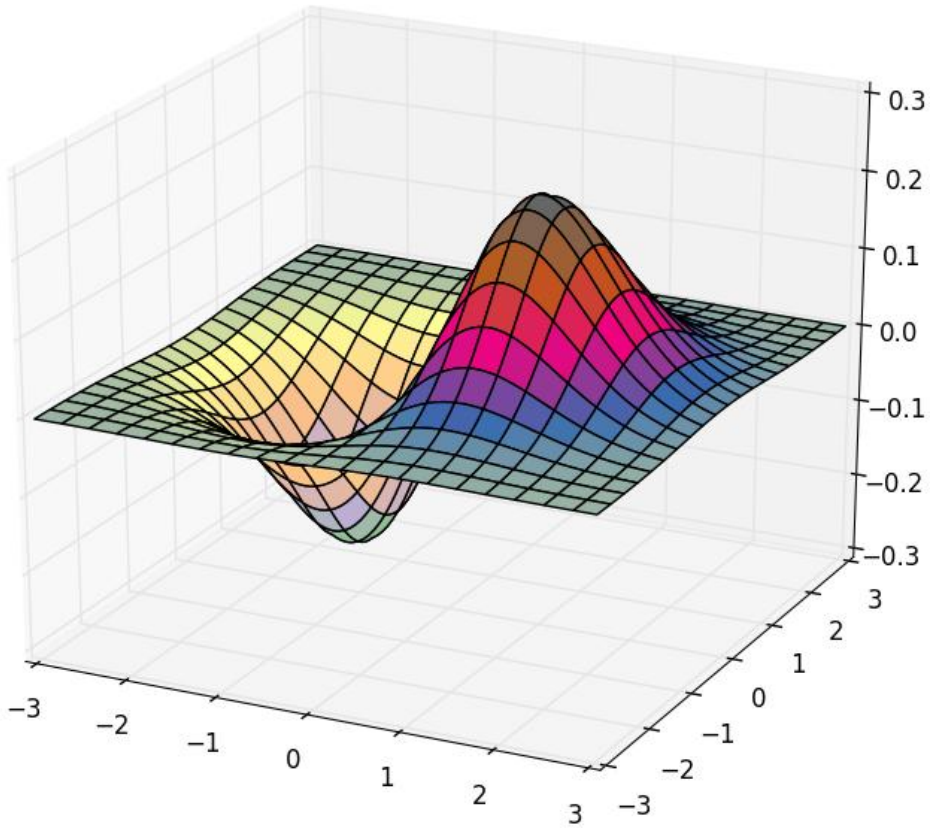

x^x
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


20
0
−20
−40
−60
−60
−40
−20
0
20
40
60


0.1
0.2
0.3
0.4
0.5
0.6

3D


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

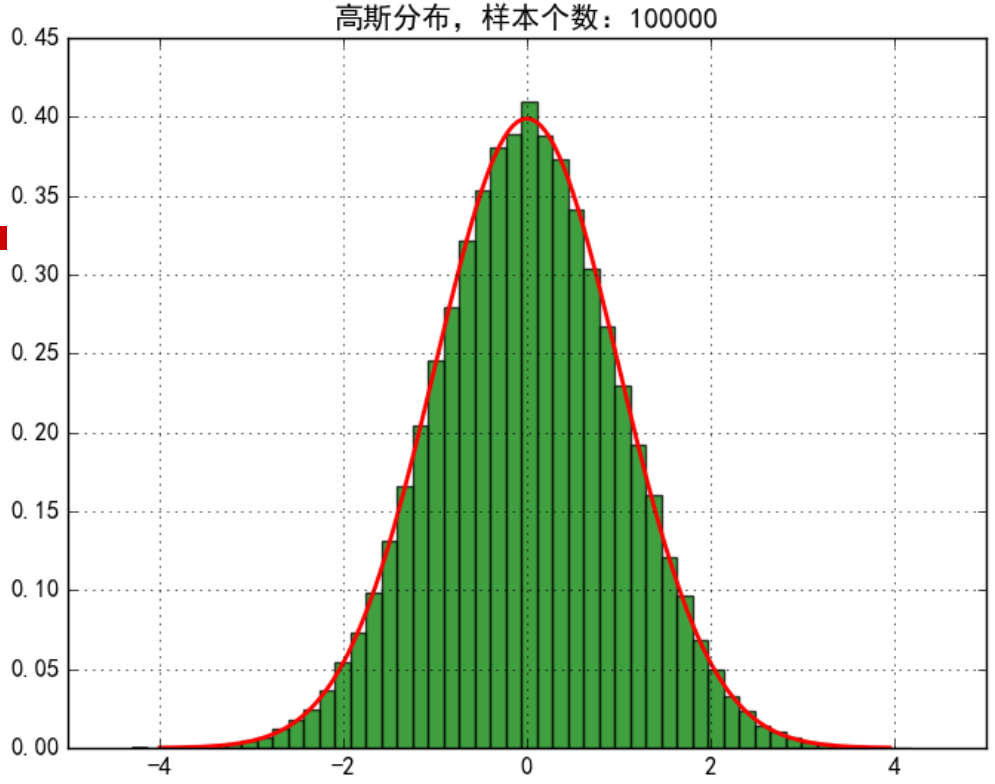

0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−3
−2
−1
0
1
2
3

类/继承类

```
class People:
    def __init__(self, n, a, s):
        self.name = n
        self.age = a
        self.__score = s
        self.print_people()
        # self.__print_people()   # 私有函数的作用

    def print_people(self):
        str = u'%s的年龄：%d，成绩为：%.2f' % (self.name, self.age, self.__score)
        print str

    __print_people = print_people


class Student(People):
    def __init__(self, n, a, w):
        People.__init__(self, n, a, w)
        self.name = 'Student ' + self.name

    def print_people(self):
        str = u'%s的年龄：%d' % (self.name, self.age)
        print str


def func(p):
    p.age = 11


if __name__ == '__main__':
    p = People('Tom', 10, 3.14159)
    func(p)     # p传入的是引用类型
    p.print_people()

    # 注意分析下面语句的打印结果，是否觉得有些“怪异”？
    j = Student('Jerry', 12, 2.71828)

    # 成员函数
    j.print_people()
    People.print_people(j)
```

```
Tom的年龄：10，成绩为：3.14
Tom的年龄：11，成绩为：3.14

Jerry的年龄：12

Tom的年龄：11，成绩为：3.14
Student Jerry的年龄：12

Tom的年龄：11，成绩为：3.14
Student Jerry的年龄：12，成绩为：2.72
```

统计量

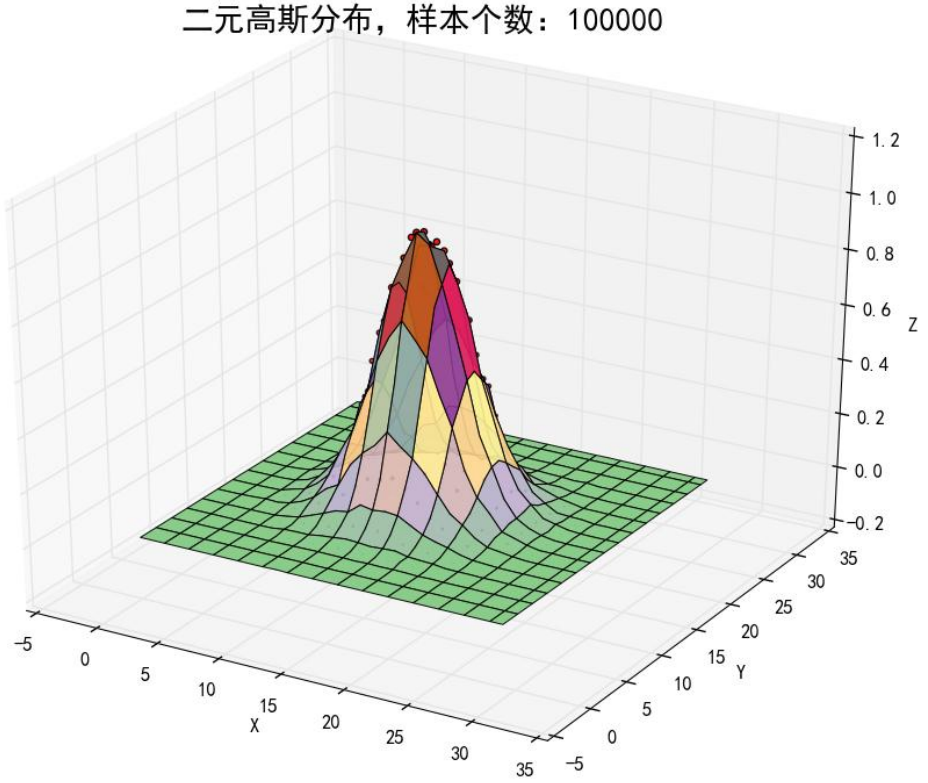



```
def calc_statistics(x):
    n = x.shape[0]  # 样本个数

    # 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

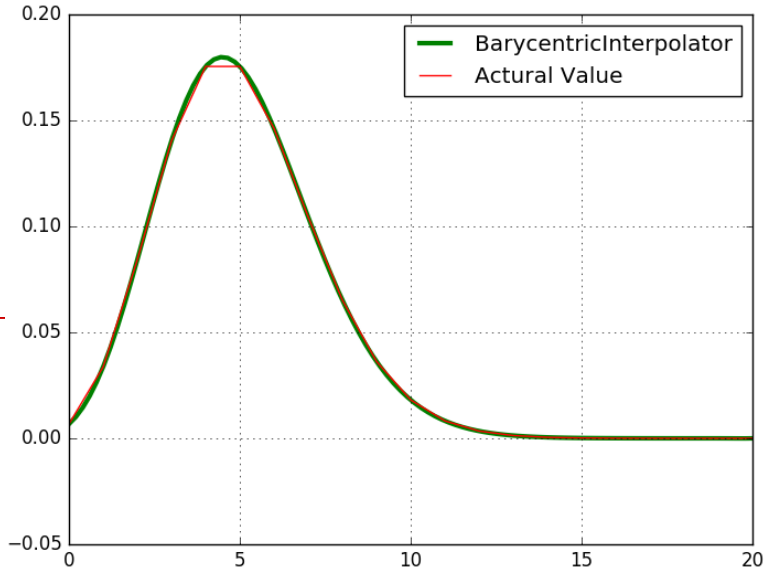
函数库计算均值、标准差、偏度、峰度： -0.00232018730484 1.00220229337 0.0070687774347 0.0174102810253

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()
```

重心插值



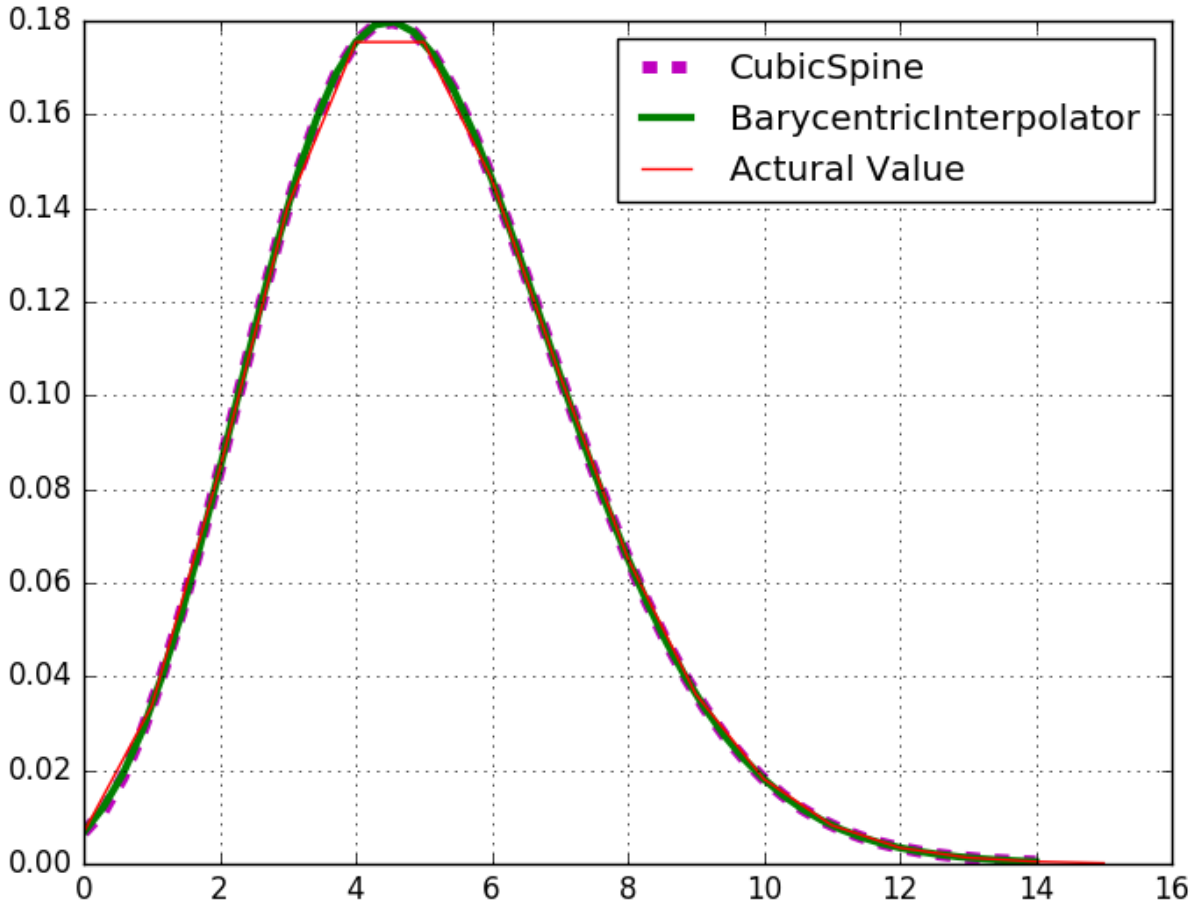
- 给定实数对 $\{(x_j, y_j),\ j = 0,1,\cdots n\}$
  - x_j互不相同。
- 对于给定的n+1个权值 $\{u_j \neq 0,\ j = 0,1,\cdots n\}$

  有：

$$f(x) = \frac{\sum_{j=0}^{n} \frac{u_j}{x - x_j} y_j}{\sum_{j=0}^{n} \frac{u_j}{x - x_j}}$$

- 则函数f(x)在x_k处的值为y_k。
  - 对于权值，可以选择 $\{u_j = (-1)^k,\ j = 0,1,\cdots n\}$

样条插值 – 重心插值


CubicSpine
BarycentricInterpolator
Actural Value
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
14
16

Demo

作业

- 实现任何一个函数曲线/曲面的Python显示。
  - Matplotlib
- 利用Python提供的SVD库函数，实现图像恢复。
- 数值计算

作业：数值计算

- 对于某二分类问题，若构造了九个正确率都是0.6的分类器，采用少数服从多数的原则进行最终分类，则最终分类正确率是多少？
  - 若构造99个分类器呢？

```
def bagging(n, p):
    p = 0.6
    s = 0
    for i in range(n / 2 + 1, n + 1):
        s += c(n, i) * p ** i * (1 - p) ** (n - i)
    return s


if __name__ == "__main__":
    for t in range(9, 100, 10):
        print t, '次采样正确率：', bagging(t, 0.6)
```

```
Ensumble
C:\Python27\python.exe D:/Python/Ensumble.py
9 次采样正确率： 0.73343232
19 次采样正确率： 0.813907978585
29 次采样正确率： 0.863787051336
39 次采样正确率： 0.897941368711
49 次采样正确率： 0.922424437652
59 次采样正确率： 0.940447995732
69 次采样正确率： 0.953949756505
79 次采样正确率： 0.964189692839
89 次采样正确率： 0.972027516007
99 次采样正确率： 0.97806955787
```

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
  - 大数据分析挖掘

感谢大家！

恳请大家批评指正！