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基础

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简介

- Python是一种面向对象的解释型语言，由荷兰人Guido van Rossum于1989年发明，第一个公开发行版于1991年发布。
  - 现在Python是由一个核心开发团队维护，Guido van Rossum仍然占据着至关重要的作用，指导其进展。
- Python语法简洁清晰，强制用空白符(white space)作为语句缩进。
- 使用Python开发的功能模块共享给他人时，必须同时提供源码本身。
  - 当然，可使用py2exe等包转换成系统能够执行的文件。
- 最大特点：简单、强大
  - Python具有丰富强大的库，常被称为胶水语言，能够把用其他语言制作的各种模块很轻松地联结在一起。
  - XGBoost/TensorFlow

Python库

- Pip
  - 安装Python包的推荐工具：https://pypi.python.org/pypi/pip
  - 更换国内源：pip install -i https://pypi.tuna.tsinghua.edu.cn/simple numpy
- Numpy：
  - 为Python提供快速的多维数组处理能力
- Pandas：PythoN Data AnalysiS Library
  - 在Numpy基础上提供了更多的数据读写工具
- 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：
  - Numpy/Scipy：http://www.scipy.org
  - Pandas：http://pandas.pydata.org/
  - Matplotlib：http://www.matplotlib.org

使用pip进行Python包的更新

```
C:\Users\zouwang>pip install pandas --upgrade
Collecting pandas
  Downloading pandas-0.19.1-cp27-cp27m-win32.whl (6.7MB)
    100% |████████████████████████████████| 6.7MB 81kB/s
Collecting python-dateutil (from pandas)
  Downloading python_dateutil-2.6.0-py2.py3-none-any.whl (194kB)
    100% |████████████████████████████████| 194kB 103kB/s
Collecting numpy>=1.7.0 (from pandas)
  Downloading numpy-1.11.2-cp27-none-win32.whl (6.5MB)
    100% |████████████████████████████████| 6.5MB 103kB/s
Collecting pytz>=2011k (from pandas)
  Downloading pytz-2016.7-py2.py3-none-any.whl (480kB)
    100% |████████████████████████████████| 481kB 244kB/s
Requirement already up-to-date: six>=1.5 in c:\python27\lib\site-packages (from python-dateutil->pandas)
Installing collected packages: python-dateutil, numpy, pytz, pandas
  Found existing installation: python-dateutil 2.5.2
    Uninstalling python-dateutil-2.5.2:
      Successfully uninstalled python-dateutil-2.5.2
  Found existing installation: numpy 1.11.1+mkl
    Uninstalling numpy-1.11.1+mkl:
      Successfully uninstalled numpy-1.11.1+mkl
  Found existing installation: pytz 2016.3
    Uninstalling pytz-2016.3:
      Successfully uninstalled pytz-2016.3
  Found existing installation: pandas 0.18.1
    Uninstalling pandas-0.18.1:
      Successfully uninstalled pandas-0.18.1
Successfully installed numpy-1.11.2 pandas-0.19.1 python-dateutil-2.6.0 pytz-2016.7
```

2to3

```
D:\Python\MachineLearning\4.Python>python 2to3.py .\Python3\4.3.calc_e.py
RefactoringTool: Skipping optional fixer: buffer
RefactoringTool: Skipping optional fixer: idioms
RefactoringTool: Skipping optional fixer: set_literal
RefactoringTool: Skipping optional fixer: ws_comma
RefactoringTool: Refactored .\Python3\4.3.calc_e.py
--- .\Python3\4.3.calc_e.py     (original)
+++ .\Python3\4.3.calc_e.py     (refactored)
@@ -33,17 +33,17 @@
     t1 = np.linspace(-2, 0, 10, endpoint=False)
     t2 = np.linspace(0, 4, 20)
     t = np.concatenate((t1, t2))
-    print t     # 横轴数据
+    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), '(真实值)'
+        print('e^', x, ' = ', y[i], '(近似值)\t', math.exp(x), '(真实值)')
         # print '误差: ', y[i] - math.exp(x)
     plt.figure(facecolor='w')
-    mpl.rcParams['font.sans-serif'] = [u'SimHei']
+    mpl.rcParams['font.sans-serif'] = ['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 - 指数函数', fontsize=18)
+    plt.title('Taylor展式的应用 - 指数函数', fontsize=18)
     plt.xlabel('X', fontsize=15)
     plt.ylabel('exp(X)', fontsize=15)
     plt.grid(True, ls=':')
RefactoringTool: Files that need to be modified:
RefactoringTool: .\Python3\4.3.calc_e.py
```

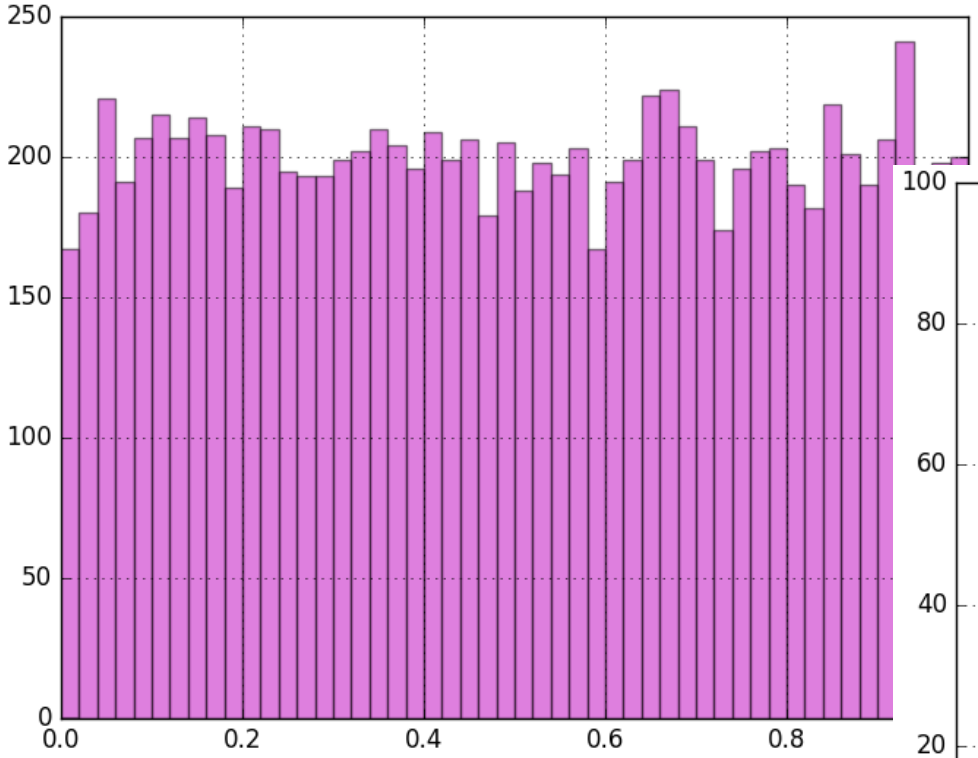

https://docs.python.org/2/library/2to3.html#module-lib2to3

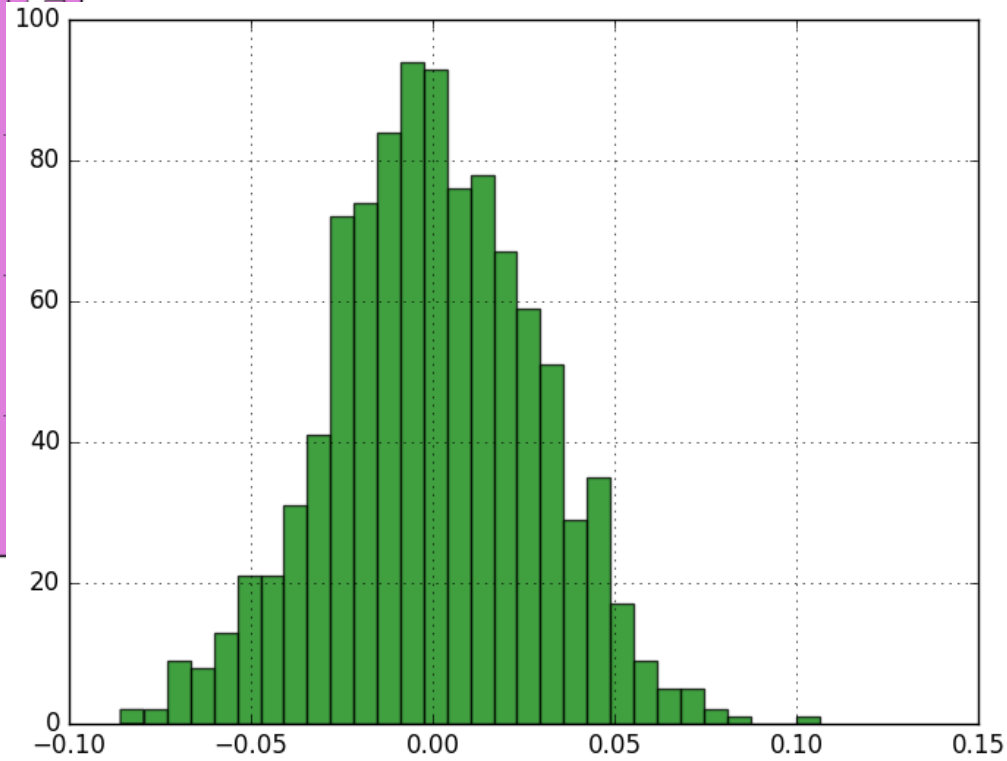
数据生成

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

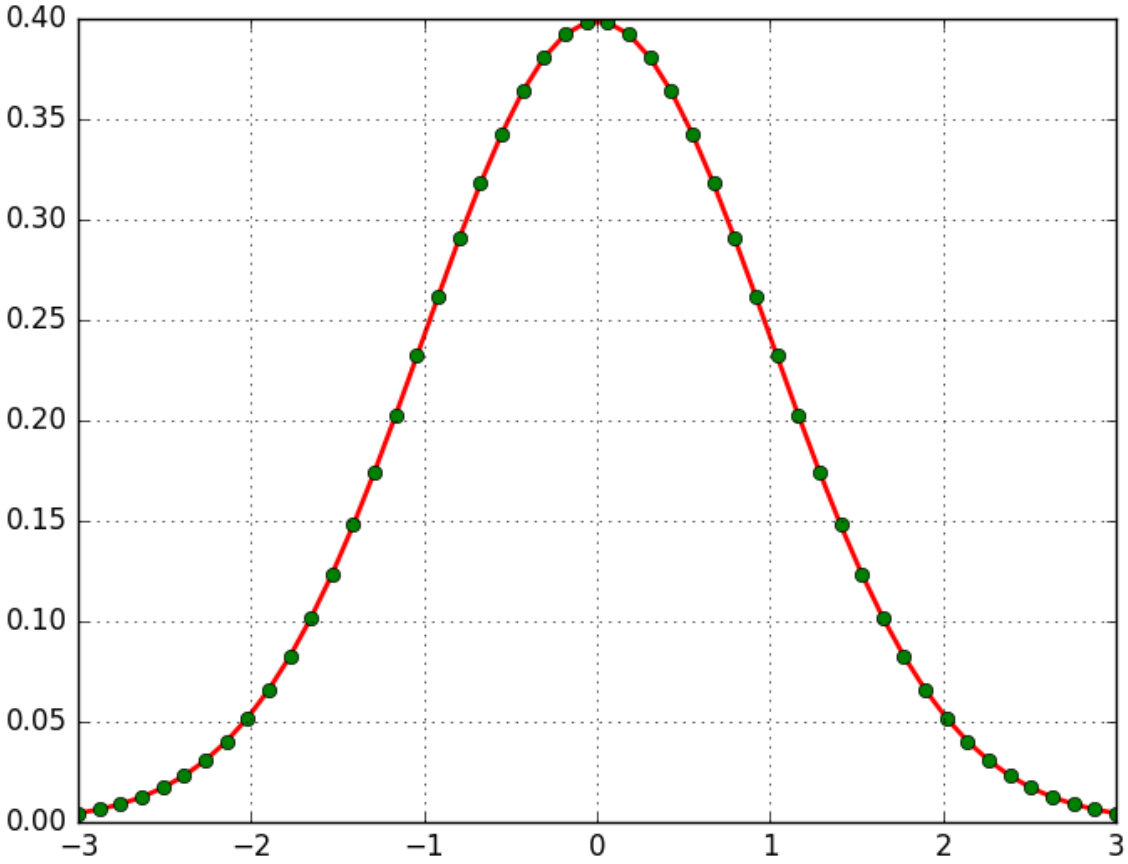
```
[[ 0  1  2  3  4  5]
[10 11 12 13 14 15]
[20 21 22 23 24 25]
[30 31 32 33 34 35]
[40 41 42 43 44 45]
[50 51 52 53 54 55]]
```

验证中心极限定理

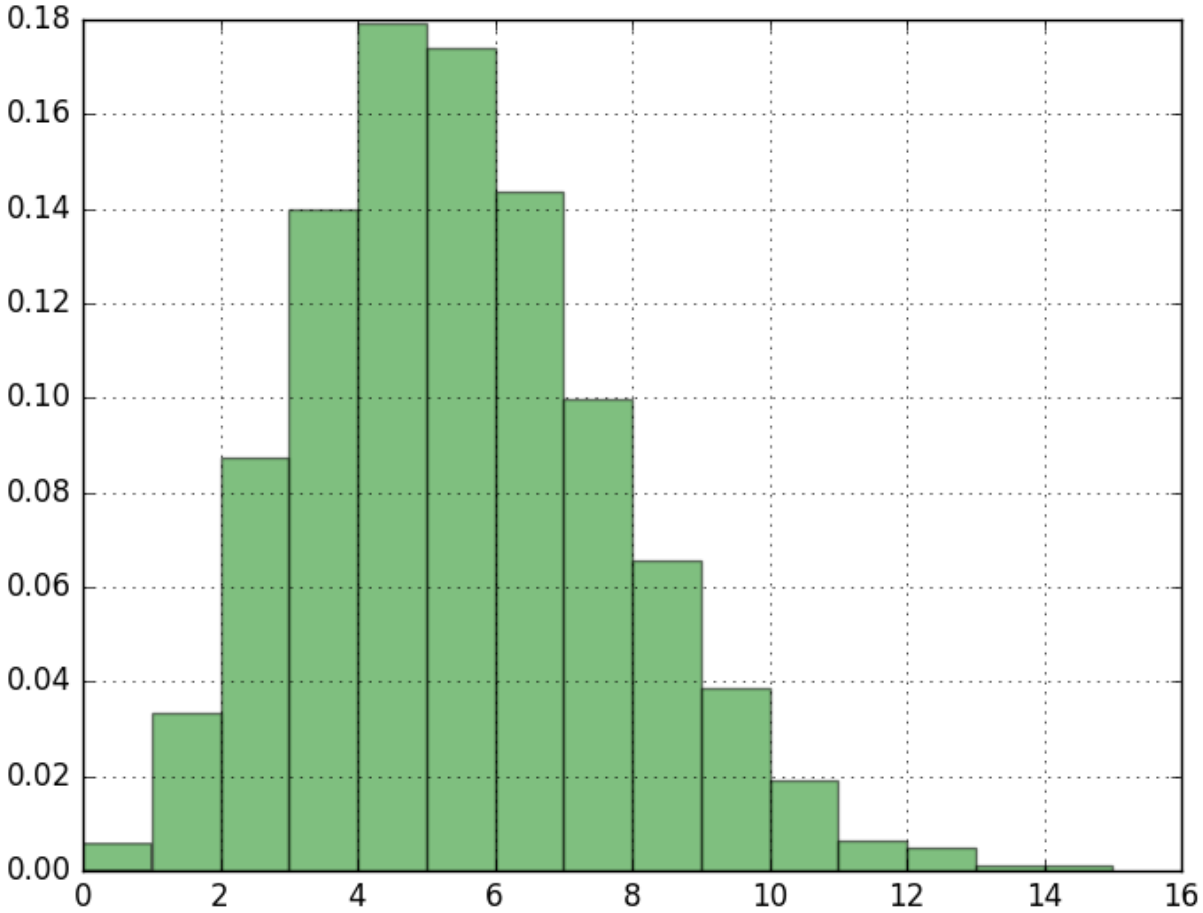

250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8


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

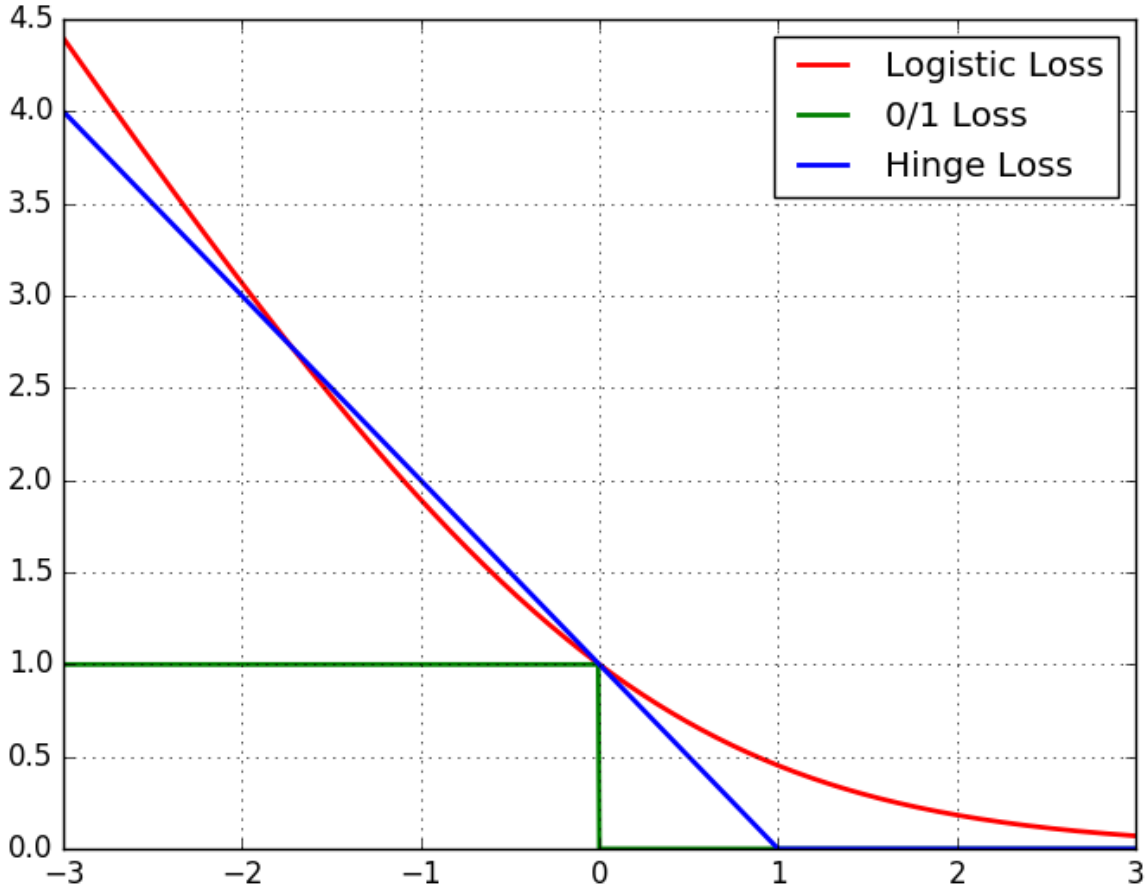
正态分布的概率密度函数



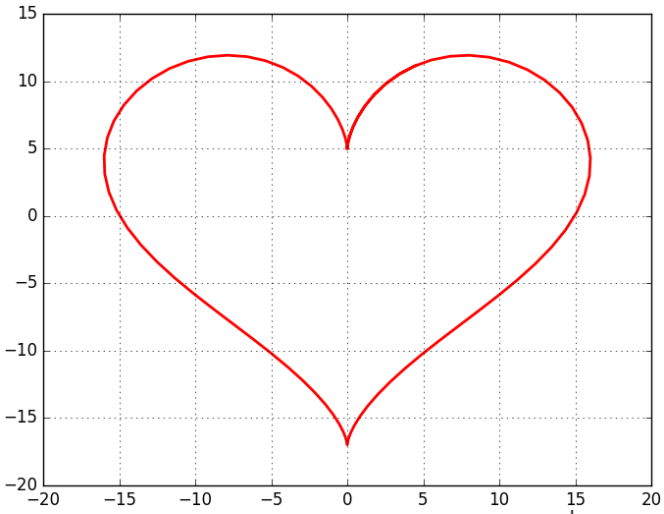
Poisson分布的概率质量函数

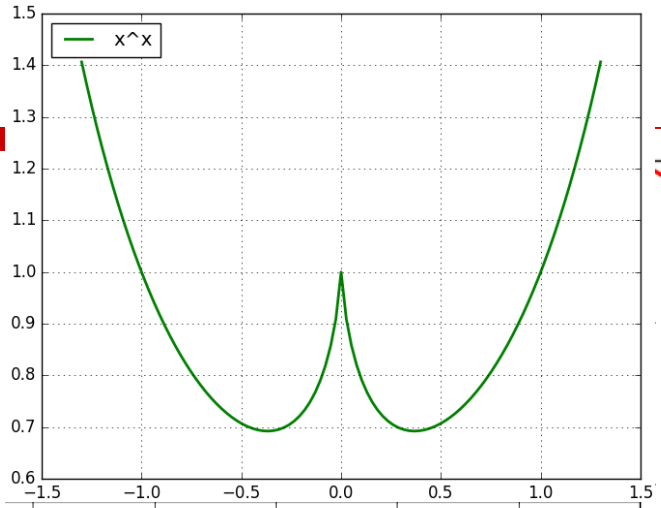


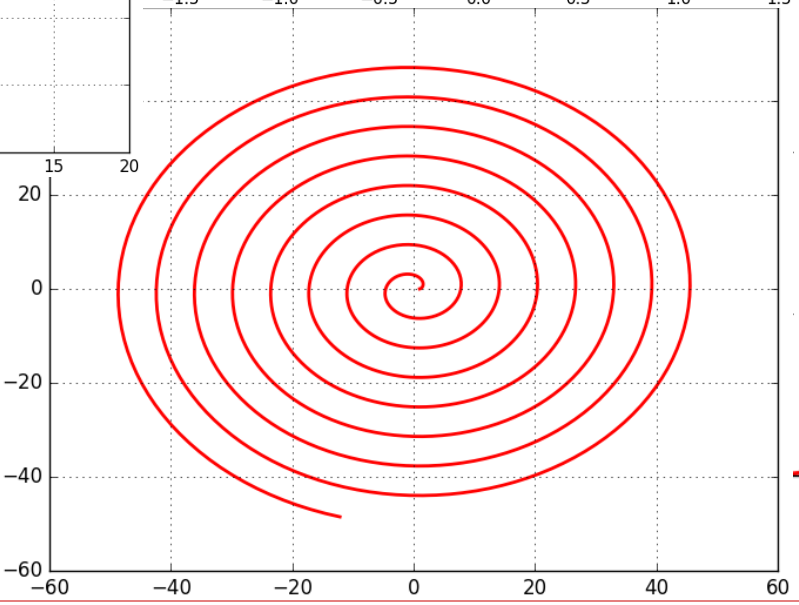
机器学习中的损失函数

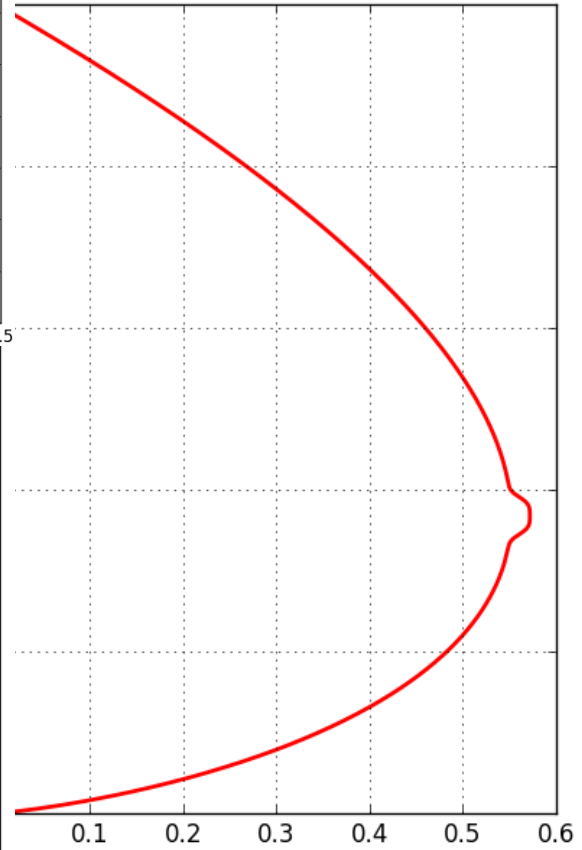

Logistic Loss
0/1 Loss
Hinge Loss
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−3
−2
−1
0
1
2
3

各种2D曲线

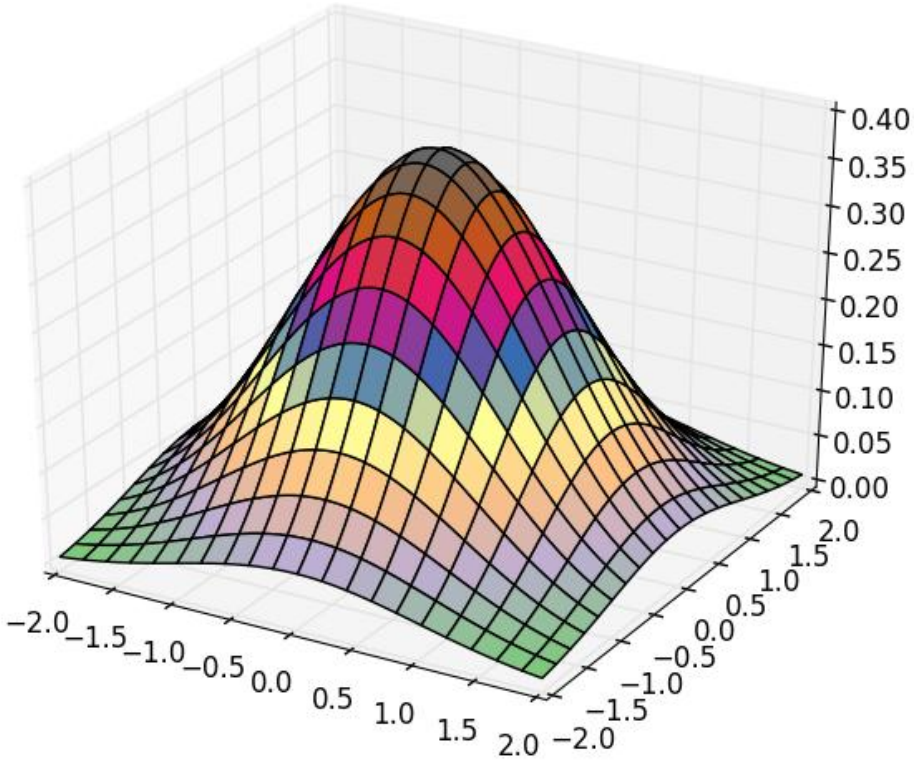

15
10
5
0
−5
−10
−15
−20
−20
−15
−10
−5
0
5
10
15
20

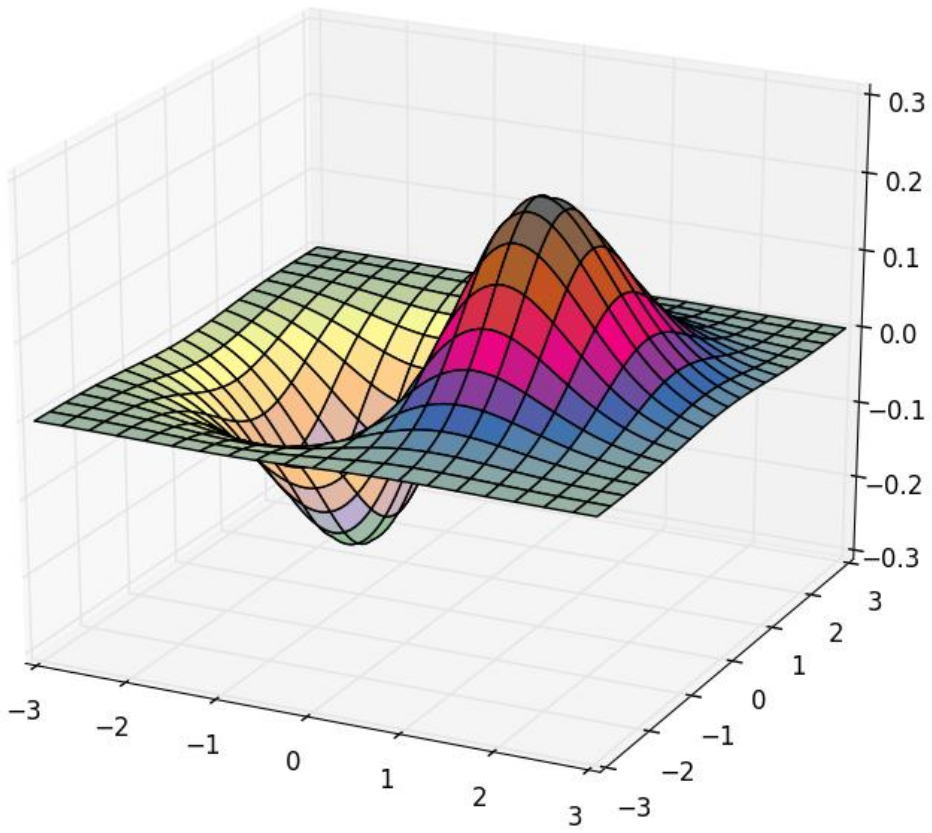

x^x
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


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

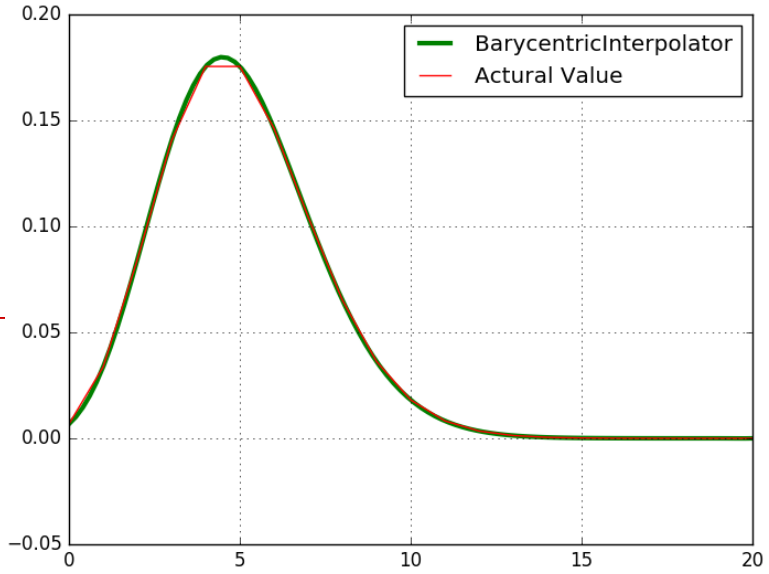

0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−3
−2
−1
0
1
2
3

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
```

重心插值



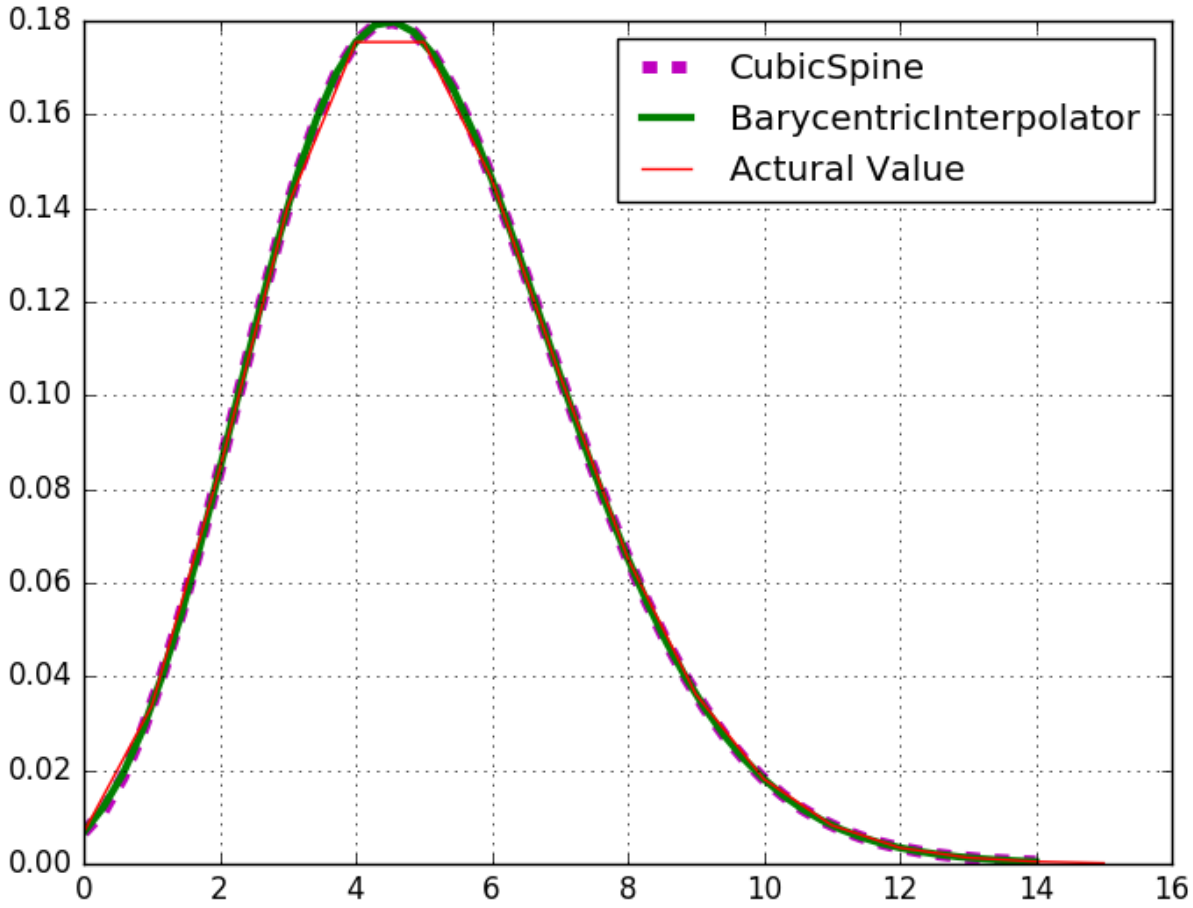
- 给定实数对 $\{(x_j, y_j),\ j = 0,1,\cdots n\}$
  - x_j互不相同。
- 对于给定的n+1个权值 $\{u_j \neq 0,\ j = 0,1,\cdots n\}$

  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

Taylor展式

- 数值计算：初等函数值的计算(在原点展开)

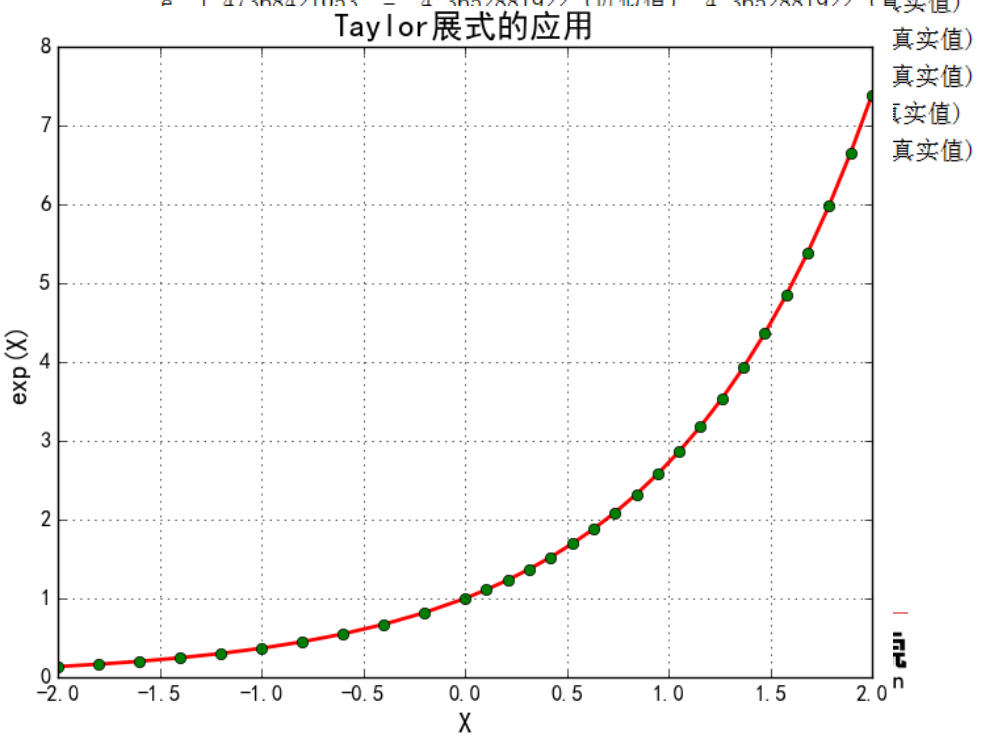
$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \cdots + (-1)^{m-1}\frac{x^{2m-1}}{(2m-1)!} + R_{2m}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + R_n$$

Taylor展式的应用

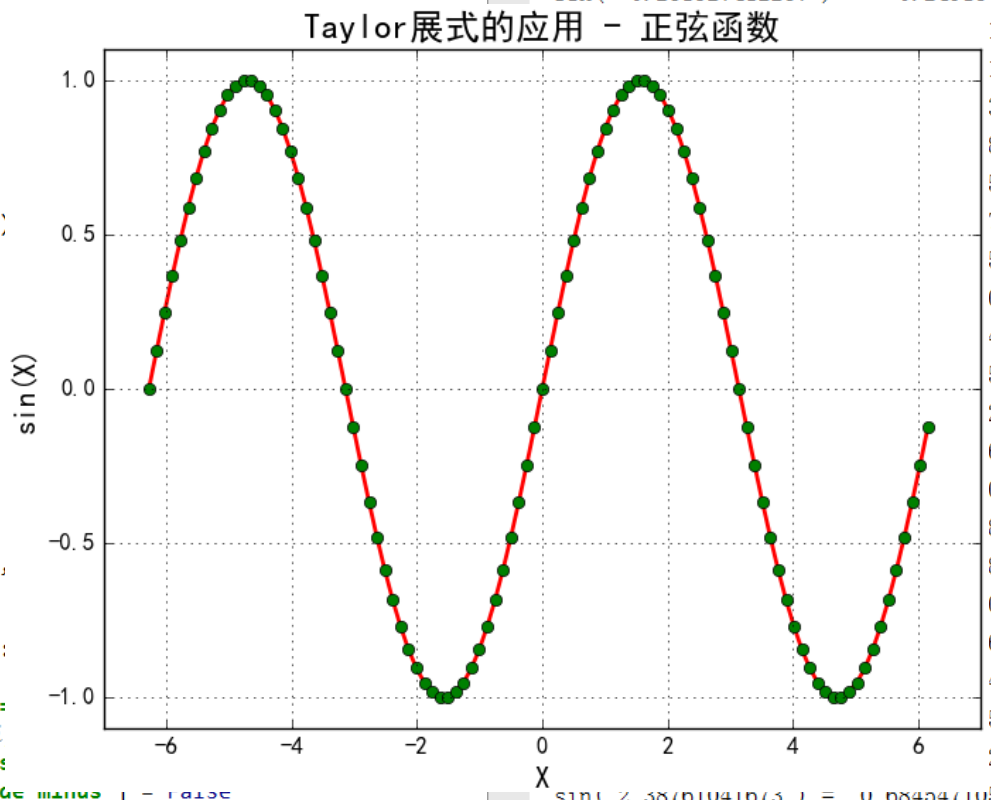
```
e^ -0.8  =  0.449328964117 (近似值) 0.449328964117 (真实值)
e^ -0.6  =  0.548811636094 (近似值) 0.548811636094 (真实值)
e^ -0.4  =  0.670320046036 (近似值) 0.670320046036 (真实值)
e^ -0.2  =  0.818730753078 (近似值) 0.818730753078 (真实值)
e^ 0.0  =  1.0 (近似值) 1.0 (真实值)
e^ 0.105263157895  =  1.11100294108 (近似值)  1.11100294108 (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()
```



Taylor展式的应用

```
def calc_sin_small(x):
    x2 = -x ** 2
    t = x
    f = 1
    sum = 0
    for i in range(10):
        sum += t / f
        t *= x2
        f *= ((2*i+2)*(2*i+3)
    return sum


def calc_sin(x):
    a = x / (2*np.pi)
    k = np.floor(a)
    a = x - k*2*np.pi
    return calc_sin_small(a)


if __name__ == "__main__":
    t = np.linspace(-2*np.pi,
    print t      # 横轴数据
    y = np.empty_like(t)
    for i, x in enumerate(t):
        y[i] = calc_sin(x)
        print 'sin(', x, ') =
        # print '误差: ', y[i
    mpl.rcParams['font.sans-s
    mpl.rcParams['axes.unicode_minus'] = False
    plt.figure(facecolor='w')
    plt.plot(t, y, 'r-', t, y, 'go', linewidth=2)
    plt.title(u'Taylor展式的应用 - 正弦函数', fontsize=18)
    plt.xlabel('X', fontsize=15)
    plt.ylabel('sin(X)', fontsize=15)
    plt.xlim((-7, 7))
    plt.ylim((-1.1, 1.1))
    plt.grid(True)
    plt.show()
```



5.2.calc_sin

```
sin( -1.25663706144 ) =  -0.951066447544 (近似值)   -0.951056516295 (真实值)
sin( -1.13097335529 ) =  -0.904843692145 (近似值)   -0.904827052466 (真实值)
sin( -1.00530964915 ) =  -0.844355457307 (近似值)   -0.844327925502 (真实值)
sin( -0.879645943005 ) =  -0.770558254809 (近似值)  -0.770513242776 (真实值)
sin( -0.753982236862 ) =  -0.684619861245 (近似值)  -0.684547105929 (真实值)
sin( -0.628318530718 ) =  -0.587901575106 (近似值)  -0.587785252292 (真实值)
sin( -0.502654824574 ) =  -0.481937724358 (近似值)  -0.481753674102 (真实值)
sin( -0.376991118431 ) =  -0.368412871668 (近似值)  -0.368124552685 (真实值)
sin( -0.251327412287 ) =  -0.249137247033 (近似值)  -0.248689887165 (真实值)
                          1027336 (近似值)  -0.125333233564 (真实值)
                          197e-16 (近似值)  8.881784197e-16 (真实值)
                          33564 (近似值)  0.125333233564 (真实值)
                          87165 (近似值)  0.248689887165 (真实值)
                          52685 (近似值)  0.368124552685 (真实值)
                          74102 (近似值)  0.481753674102 (真实值)
                          52292 (近似值)  0.587785252292 (真实值)
                          05929 (近似值)  0.684547105929 (真实值)
                          42776 (近似值)  0.770513242776 (真实值)
                          5502 (近似值)  0.844327925502 (真实值)
                          2466 (近似值)  0.904827052466 (真实值)
                          6295 (近似值)  0.951056516295 (真实值)
                          0729 (近似值)  0.982287250729 (真实值)
                          8428 (近似值)  0.998026728428 (真实值)
                          8428 (近似值)  0.998026728428 (真实值)
                          0729 (近似值)  0.982287250729 (真实值)
                          6295 (近似值)  0.951056516295 (真实值)
                          466 (近似值)  0.904827052466 (真实值)
                          5502 (近似值)  0.844327925502 (真实值)
                          2775 (近似值)  0.770513242776 (真实值)
sin( 2.38761041673 ) =  0.684547105927 (近似值)  0.684547105929 (真实值)
sin( 2.51327412287 ) =  0.587785252288 (近似值)  0.587785252292 (真实值)
sin( 2.63893782902 ) =  0.481753674088 (近似值)  0.481753674102 (真实值)
sin( 2.76460153516 ) =  0.368124552648 (近似值)  0.368124552685 (真实值)
```

计算圆周率

```
import numpy as np

if __name__ == '__main__':
    print np.sqrt(6 * np.sum(1 / np.arange(1, 100000, dtype=np.float) ** 2))
```

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} + \cdots + (-1)^{m-1}\frac{x^{2m-1}}{(2m-1)!} + R_{2m}$$

$$\sin\frac{1}{x} = \frac{1}{x} - \frac{1}{3!}\left(\frac{1}{x}\right)^3 + \frac{1}{5!}\left(\frac{1}{x}\right)^5 - \frac{x^7}{7!}\left(\frac{1}{x}\right)^7 + \frac{x^9}{9!}\left(\frac{1}{x}\right)^9$$

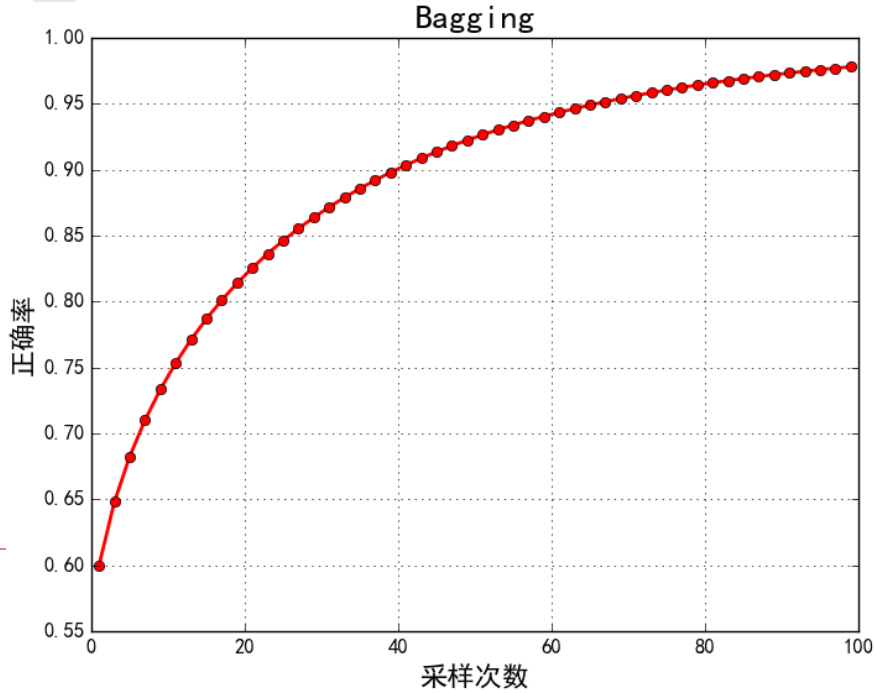
$$= \left(x - \frac{1}{\pi}\right)\left(x + \frac{1}{\pi}\right)\left(x - \frac{1}{2\pi}\right)\left(x + \frac{1}{2\pi}\right)\left(x - \frac{1}{3\pi}\right)\left(x + \frac{1}{3\pi}\right)\cdots$$

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
```



我爱乒乓球

- 福原爱与刘诗雯正在乒乓球比赛，若任何一球刘诗雯赢的概率都是60%。则对于11分制的一局，刘诗雯获胜的概率有多大？
  - 为计算简便，暂不考虑分差必须大于等于2
    - 注：如果考虑分差大于等于2，结果相差非常小
  - 0.825622133638/0.836435199842
- 如果考虑“五局三胜制”或“七局四胜制”，则刘诗雯最终获胜的概率有多大？
  - 0.966274558546
  - 0.983505058096

负二项分布

- 对于一系列独立的成败实验，每次实验成功的概率恒为p，持续实验直到r次成功(r为正整数)，则总实验次数X的概率为

$$P(X=x;r,p)=C_{x-1}^{r-1}\cdot p^{r}\cdot(1-p)^{x-r}$$

$$x\in[r,r+1,r+2,\cdots\infty)$$

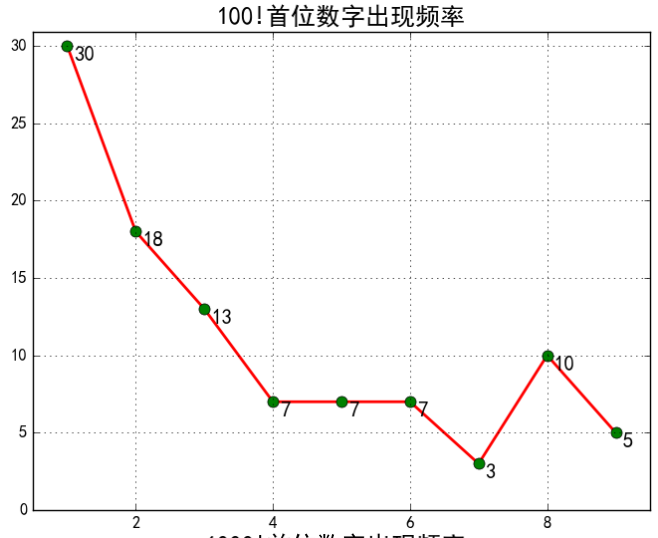
- 若记X=k为失败的次数，则有：

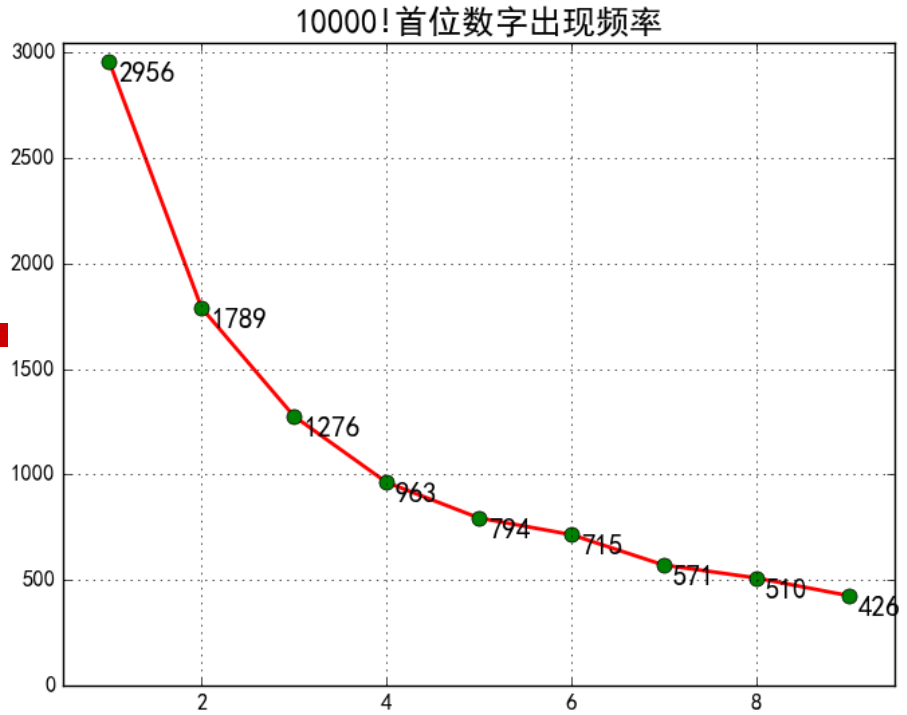
$$P(X=k;r,p)=C_{k+r-1}^{r-1}\cdot p^{r}\cdot(1-p)^{k},\ k\in N$$

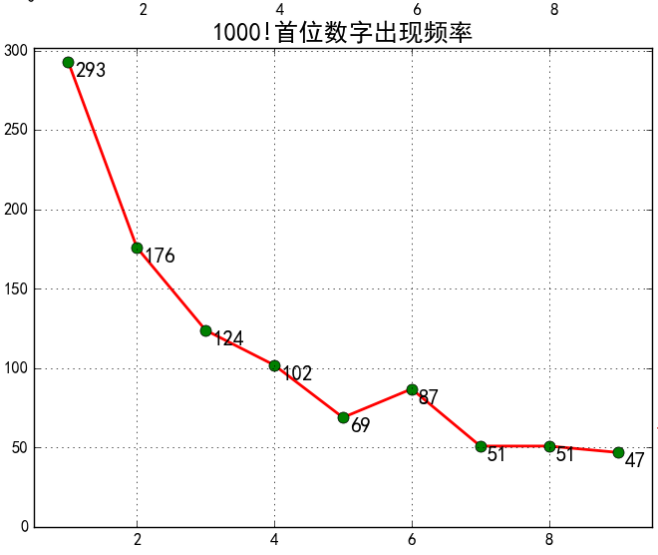
Code

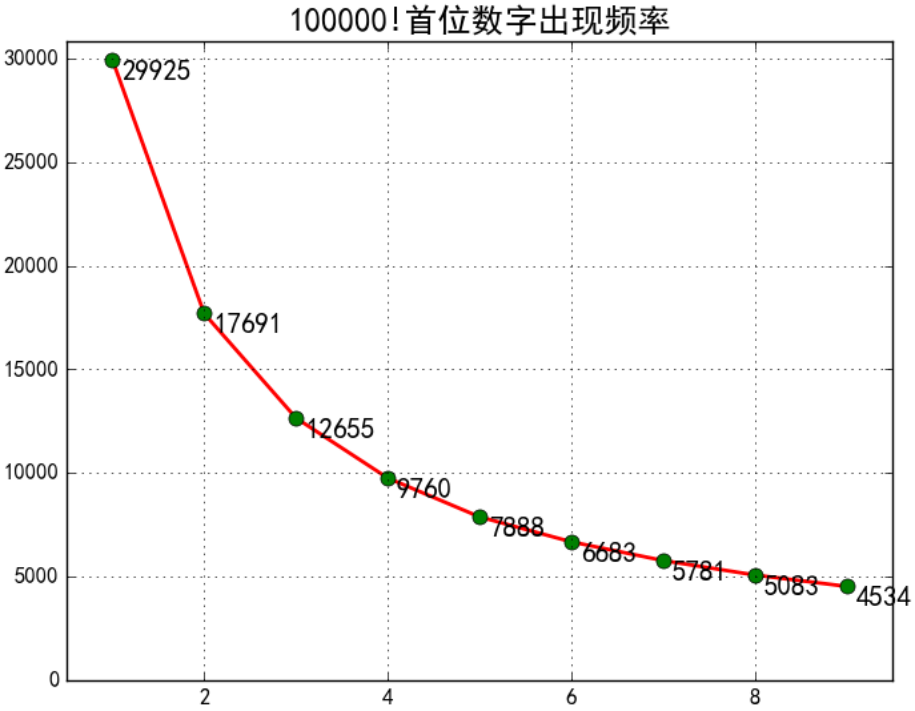
```python
import numpy as np
from scipy import special

if __name__ == '__main__':
    method = 'strict'

    # 1.暴力模拟
    if method == 'simulation':
        p = 0.6
        a, b, c = 0, 0, 0
        t, T = 0, 1000000
        while t < T:
            a = b = 0
            while (a <= 11) and (b <= 11):
                if np.random.uniform() < p:
                    a += 1
                else:
                    b += 1
            if a > b:
                c += 1
            t += 1
        print float(c) / float(T)
    # 2.直接计算
    elif method == 'simple':
        answer = 0
        p = 0.6      # 每分的胜率
        N = 11       # 每局多少分
        for x in np.arange(N):  # x为对手得分
            answer += special.comb(N + x - 1, x) * ((1-p) ** x) * (p ** N)
        print answer
    # 3.严格计算
    else:
        answer = 0
        p = 0.6  # 每分的胜率
        N = 11  # 每局多少分
        for x in np.arange(N-1):  # x为对手得分：11:9  11:8  11:7  11:6...
            answer += special.comb(N + x - 1, x) * ((1 - p) ** x) * (p ** N)
        p10 = special.comb(2*(N-1), N-1) * ((1-p)*p) ** (N-1)   # 10:10的概率
        t = 0
        for n in np.arange(100):    # {XO}(0,)|OO   思考：可以如何简化？
            t += (2*p*(1-p)) ** n * p * p
        answer += p10 * t
        print answer
```

本福特定律


100!首位数字出现频率
30
18
13
7
7
7
3
10
5


10000!首位数字出现频率
2956
1789
1276
963
794
715
571
510
426


1000!首位数字出现频率
293
176
124
102
69
87
51
51
47


100000!首位数字出现频率
29925
17691
12655
9760
7888
6683
5781
5083
4534

互联网

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

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

感谢大家！

恳请大家批评指正！