

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



小象学院
ChinaHadoop.cn

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，第一个公开发行人版于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>

Country	Year	Value
Algeria	2000	0.0000
Algeria	2001	0.0000
Algeria	2002	0.0000
Algeria	2003	0.0000
Algeria	2004	0.0000
Algeria	2005	0.0000
Algeria	2006	0.0000
Algeria	2007	0.0000
Algeria	2008	0.0000
Algeria	2009	0.0000
Algeria	2010	0.0000
Algeria	2011	0.0000
Algeria	2012	0.0000
Algeria	2013	0.0000
Algeria	2014	0.0000
Algeria	2015	0.0000
Algeria	2016	0.0000
Algeria	2017	0.0000
Algeria	2018	0.0000
Algeria	2019	0.0000
Algeria	2020	0.0000
Algeria	2021	0.0000
Algeria	2022	0.0000
Algeria	2023	0.0000
Algeria	2024	0.0000
Algeria	2025	0.0000
Algeria	2026	0.0000
Algeria	2027	0.0000
Algeria	2028	0.0000
Algeria	2029	0.0000
Algeria	2030	0.0000
Algeria	2031	0.0000
Algeria	2032	0.0000
Algeria	2033	0.0000
Algeria	2034	0.0000
Algeria	2035	0.0000
Algeria	2036	0.0000
Algeria	2037	0.0000
Algeria	2038	0.0000
Algeria	2039	0.0000
Algeria	2040	0.0000
Algeria	2041	0.0000
Algeria	2042	0.0000
Algeria	2043	0.0000
Algeria	2044	0.0000
Algeria	2045	0.0000
Algeria	2046	0.0000
Algeria	2047	0.0000
Algeria	2048	0.0000
Algeria	2049	0.0000
Algeria	2050	0.0000
Algeria	2051	0.0000
Algeria	2052	0.0000
Algeria	2053	0.0000
Algeria	2054	0.0000
Algeria	2055	0.0000
Algeria	2056	0.0000
Algeria	2057	0.0000
Algeria	2058	0.0000
Algeria	2059	0.0000
Algeria	2060	0.0000
Algeria	2061	0.0000
Algeria	2062	0.0000
Algeria	2063	0.0000
Algeria	2064	0.0000
Algeria	2065	0.0000
Algeria	2066	0.0000
Algeria	2067	0.0000
Algeria	2068	0.0000
Algeria	2069	0.0000
Algeria	2070	0.0000
Algeria	2071	0.0000
Algeria	2072	0.0000
Algeria	2073	0.0000
Algeria	2074	0.0000
Algeria	2075	0.0000
Algeria	2076	0.0000
Algeria	2077	0.0000
Algeria	2078	0.0000
Algeria	2079	0.0000
Algeria	2080	0.0000
Algeria	2081	0.0000
Algeria	2082	0.0000
Algeria	2083	0.0000
Algeria	2084	0.0000
Algeria	2085	0.0000
Algeria	2086	0.0000
Algeria	2087	0.0000
Algeria	2088	0.0000
Algeria	2089	0.0000
Algeria	2090	0.0000
Algeria	2091	0.0000
Algeria	2092	0.0000
Algeria	2093	0.0000
Algeria	2094	0.0000
Algeria	2095	0.0000
Algeria	2096	0.0000
Algeria	2097	0.0000
Algeria	2098	0.0000
Algeria	2099	0.0000
Algeria	2100	0.0000
Algeria	2101	0.0000
Algeria	2102	0.0000
Algeria	2103	0.0000
Algeria	2104	0.0000
Algeria	2105	0.0000
Algeria	2106	

```
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
```

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 pvtz>=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+mk1:

```
Successfully uninstalled numpy-1.11.1+mk1
```

```
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
```

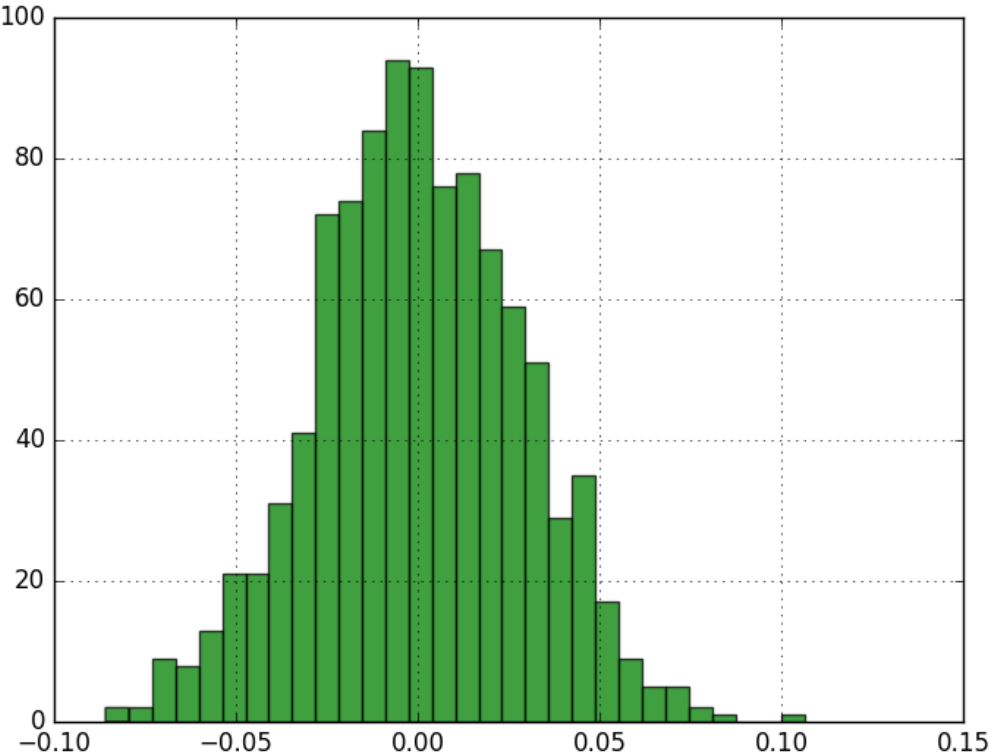
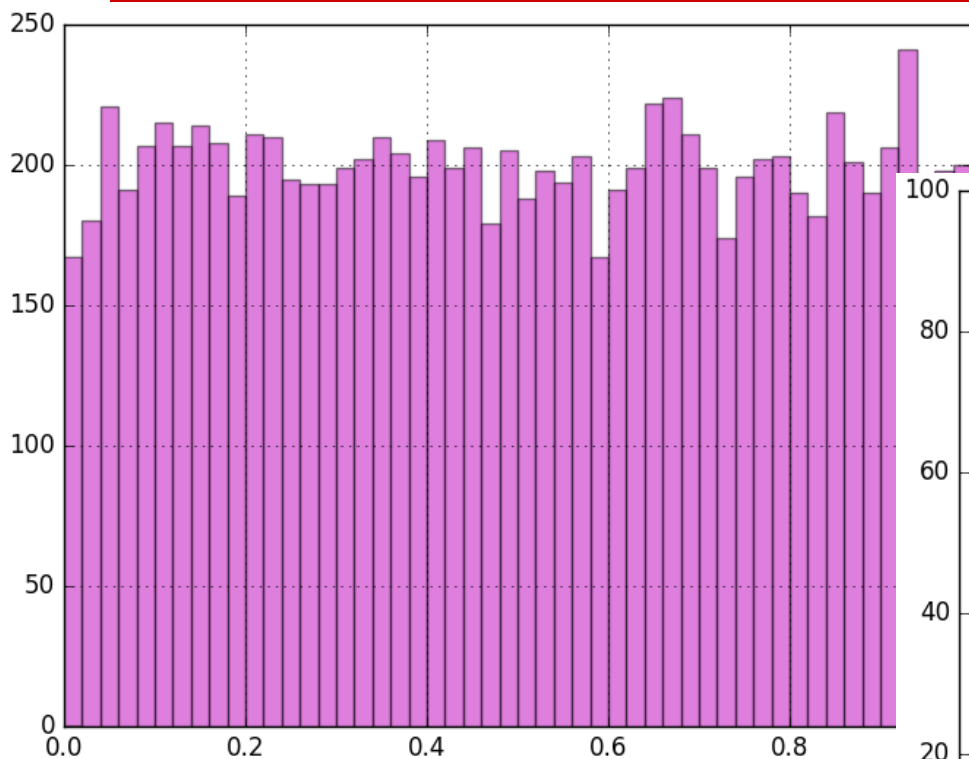
数据生成

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

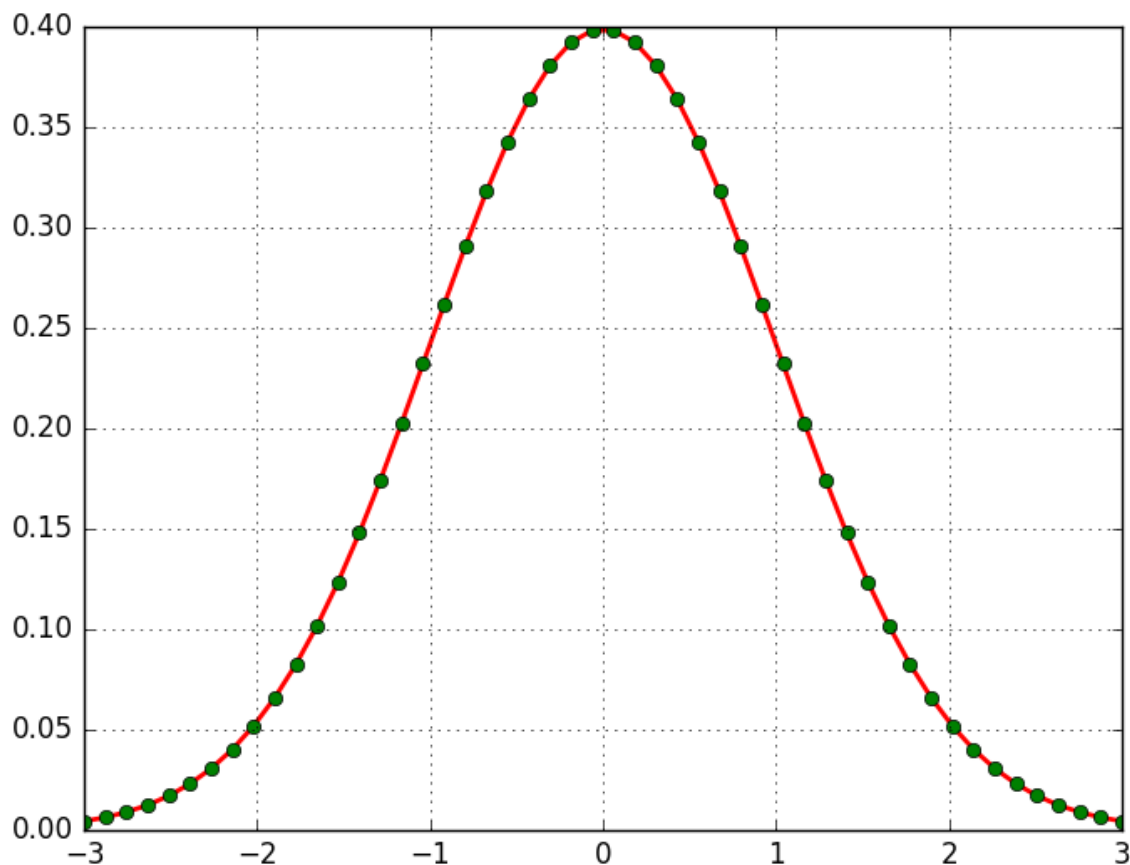
□ $A =$

```
[[ 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]]
```

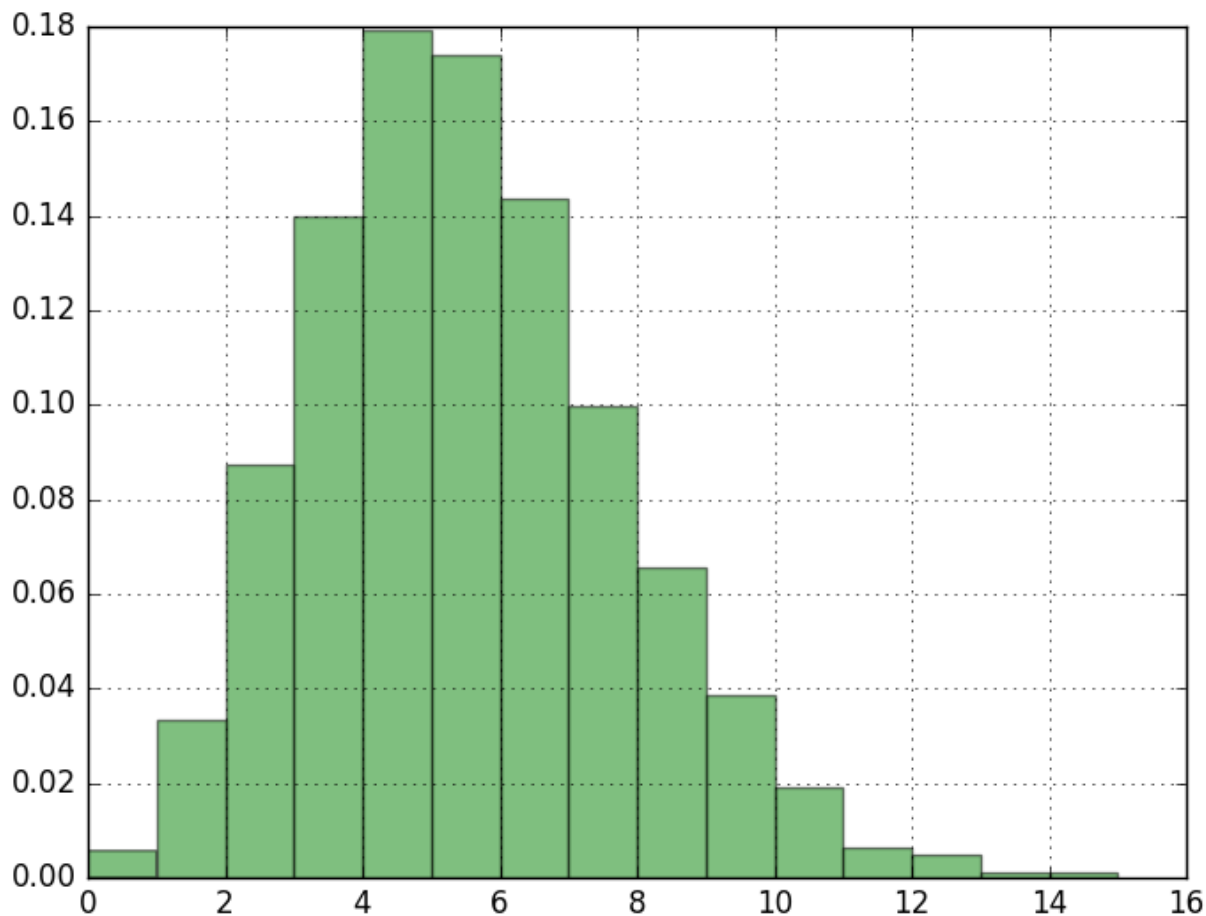

验证中心极限定理



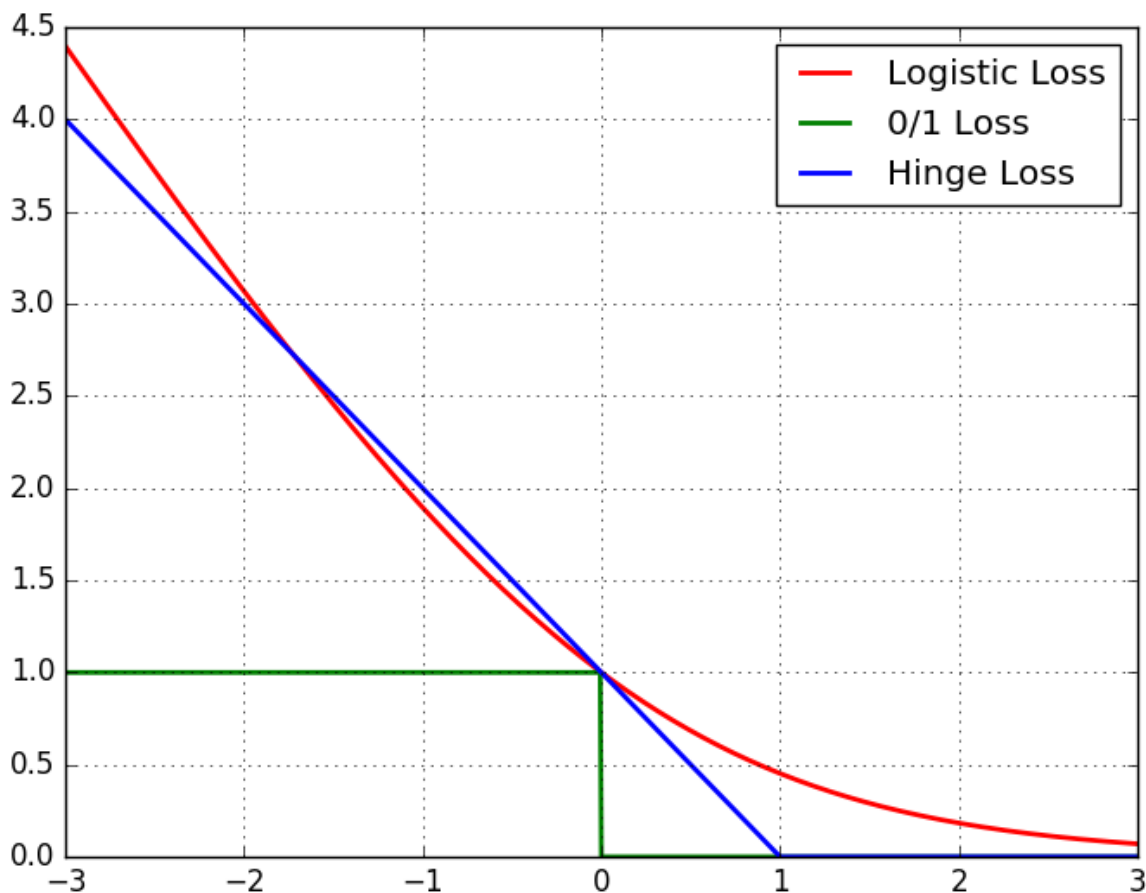
正态分布的概率密度函数



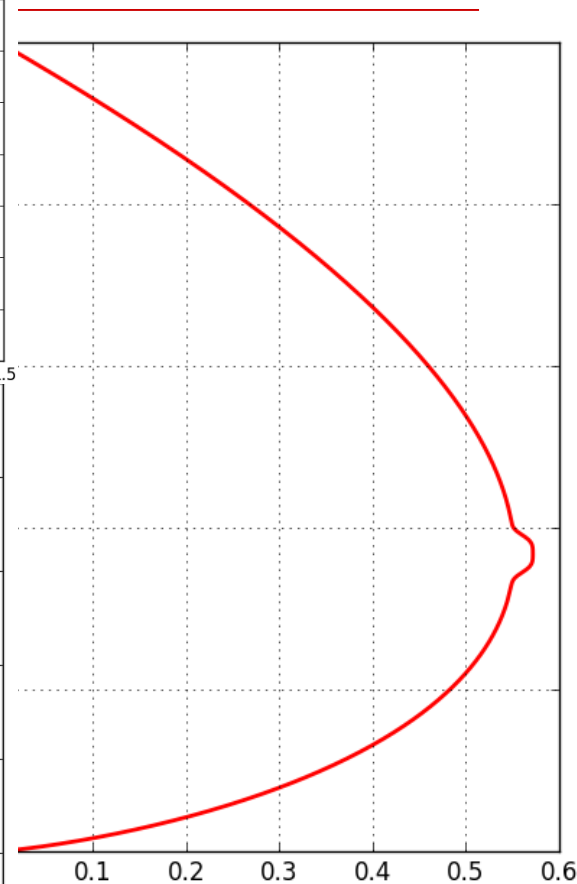
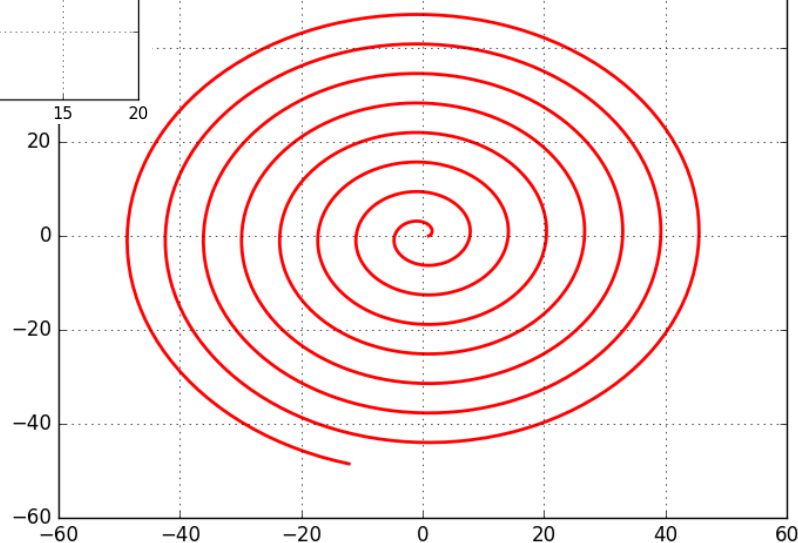
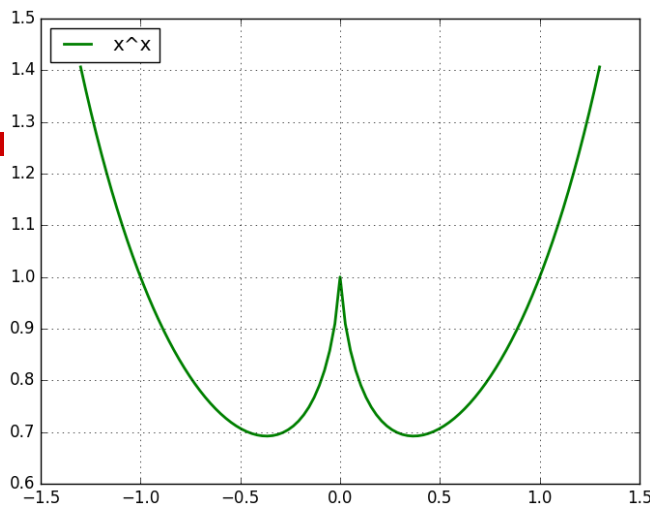
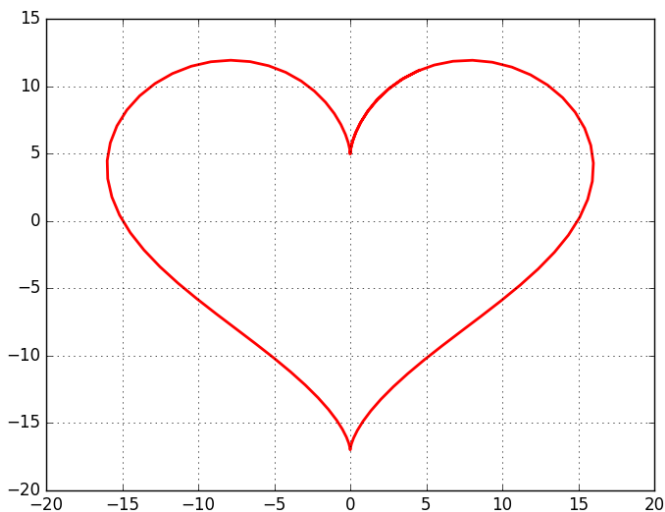
Poisson分布的概率质量函数



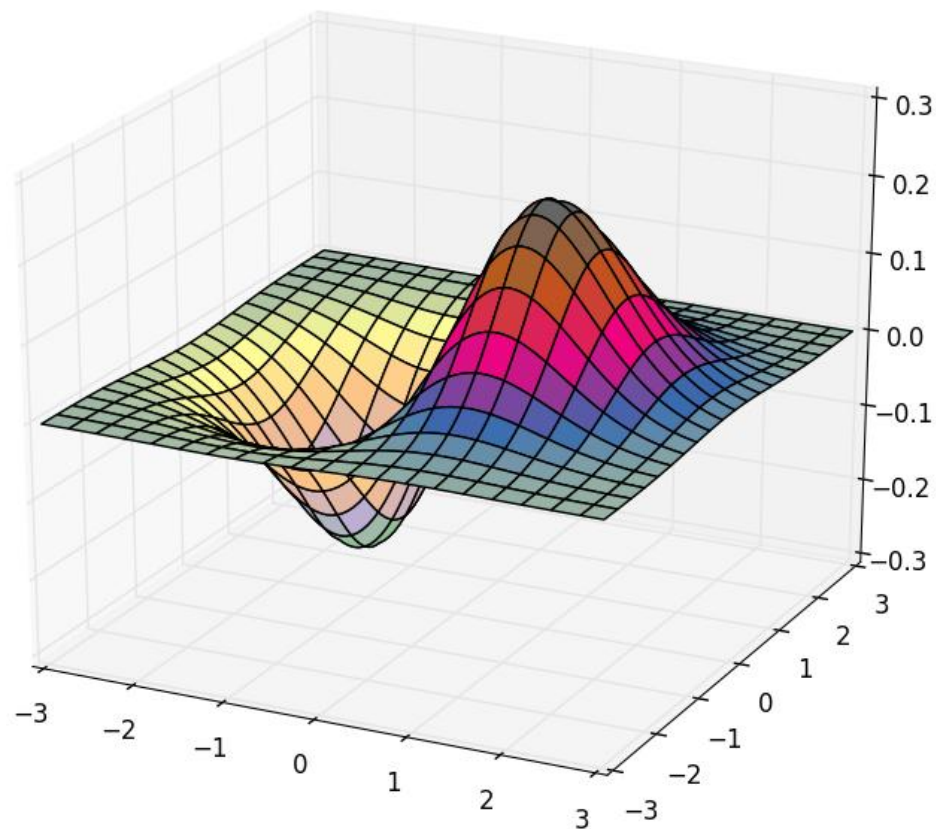
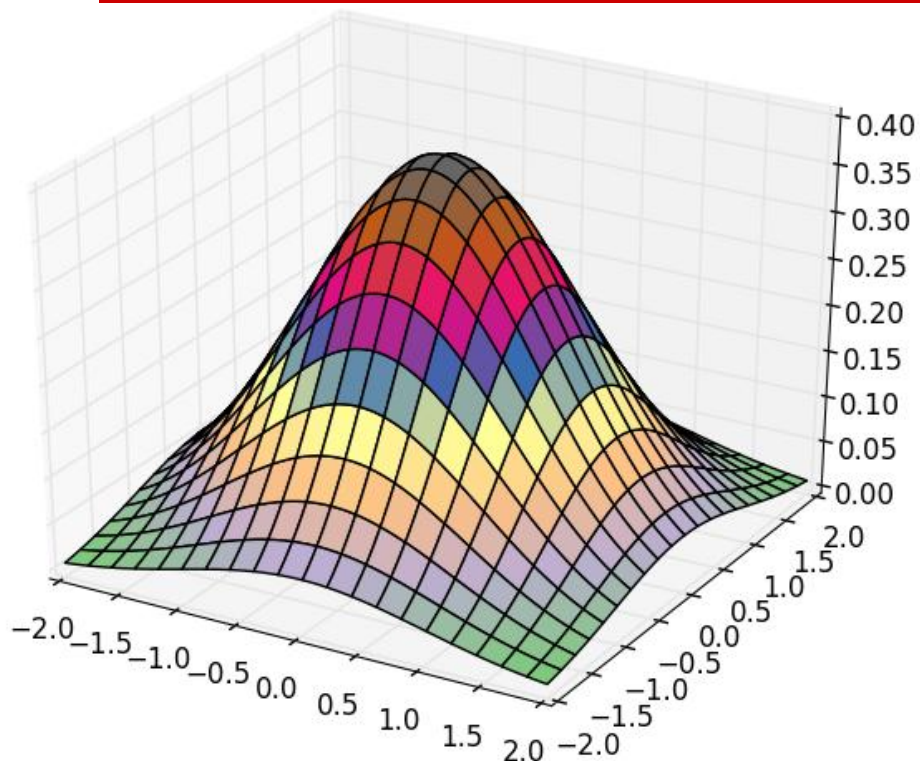
机器学习中的损失函数



各种2D曲线



3D



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 $\{(x_j, y_j), j=0,1,\dots,n\}$

■ x_j 互不相同。

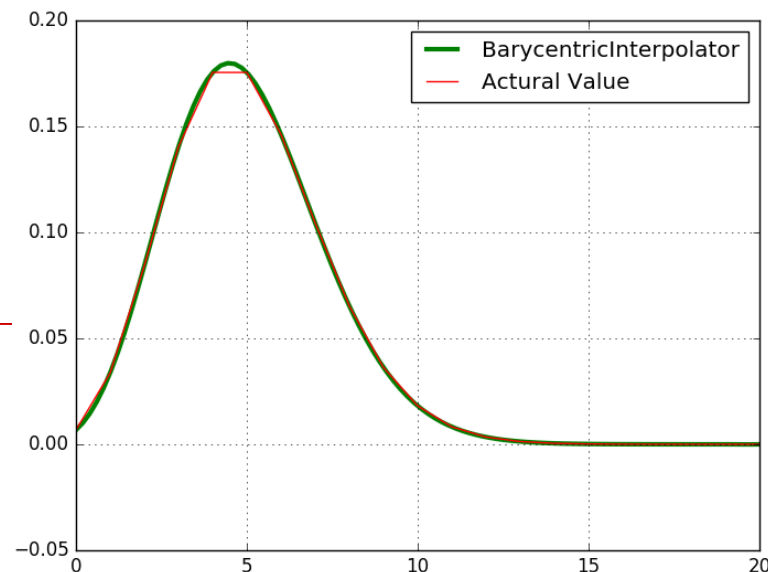
□ 对于给定的 $n+1$ 个权值 $\{u_j \neq 0, j=0,1,\dots,n\}$

有：

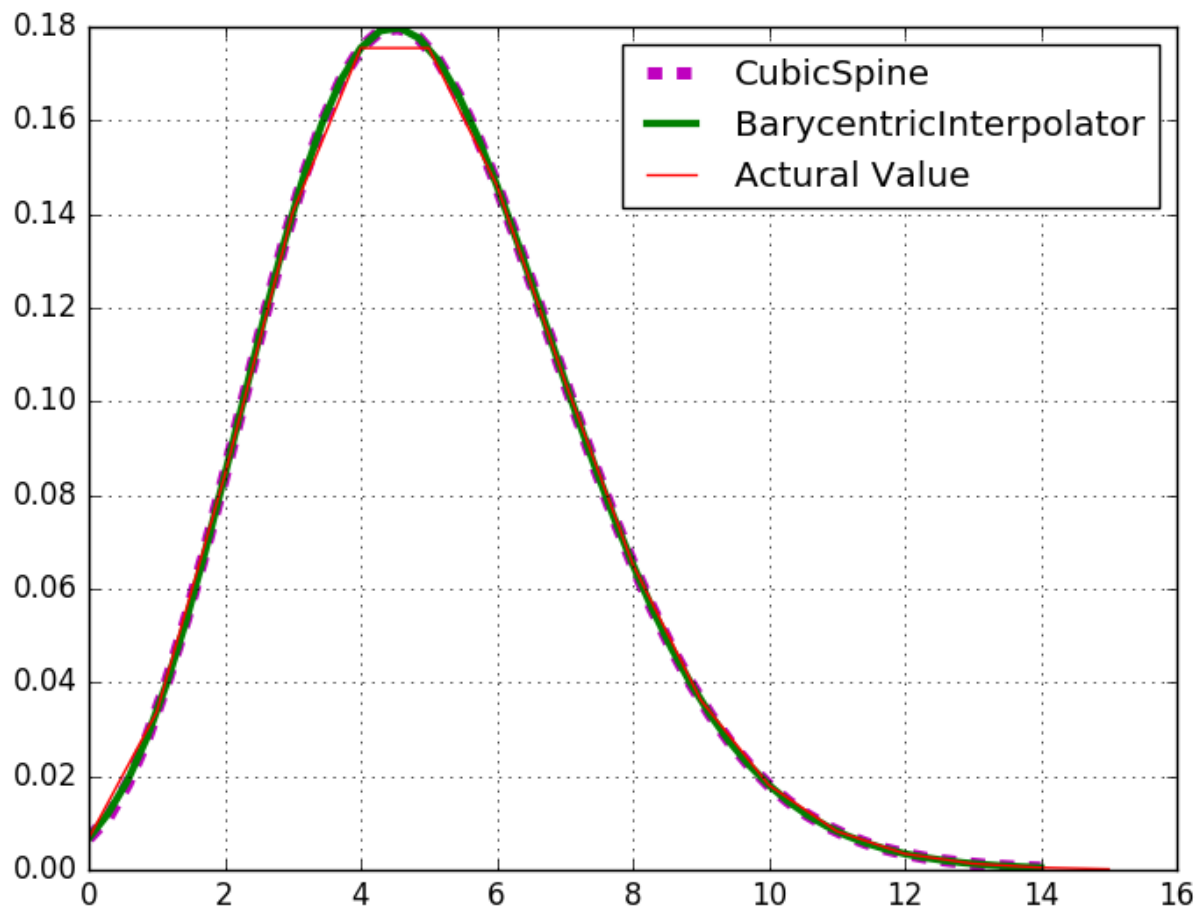
$$f(x) = \frac{\sum_{j=0}^n \frac{u_j}{x - x_j} y_j}{\sum_{j=0}^n \frac{u_j}{x - x_j}}$$

□ 则函数 $f(x)$ 在 x_k 处的值为 y_k 。

■ 对于权值，可以选择 $\{u_j = (-1)^j, j=0,1,\dots,n\}$



样条插值 – 重心插值



Taylor展式

□ 数值计算：初等函数值的计算(在原点展开)

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

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

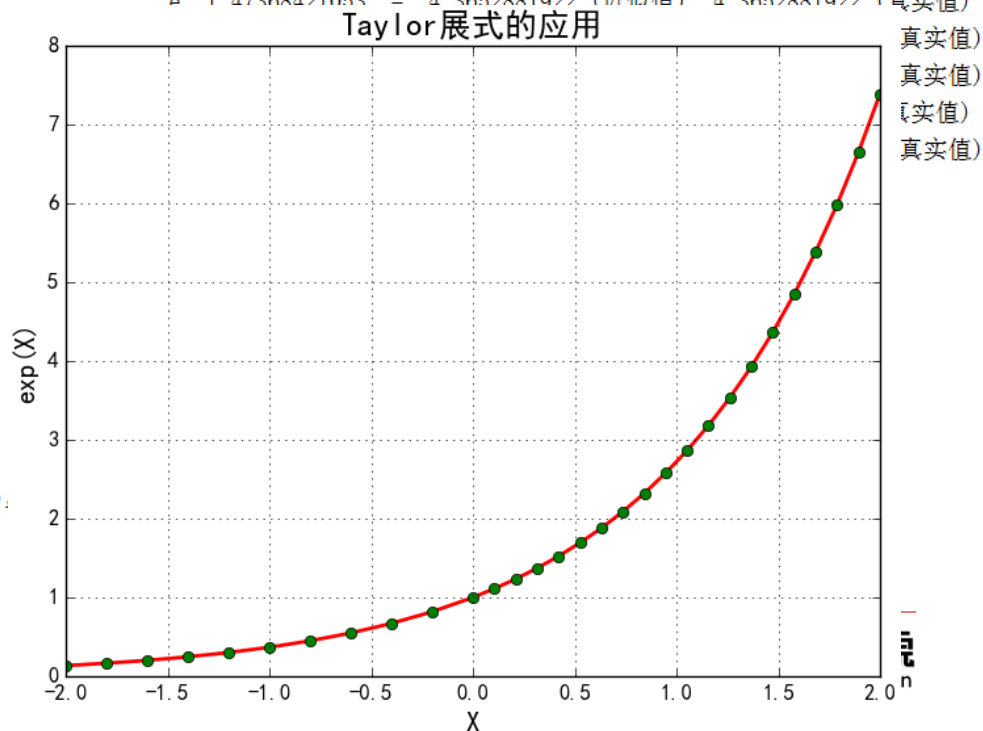
Taylor展式的应用

```
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()
```



$e^{-0.8} = 0.449328964117$ (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)

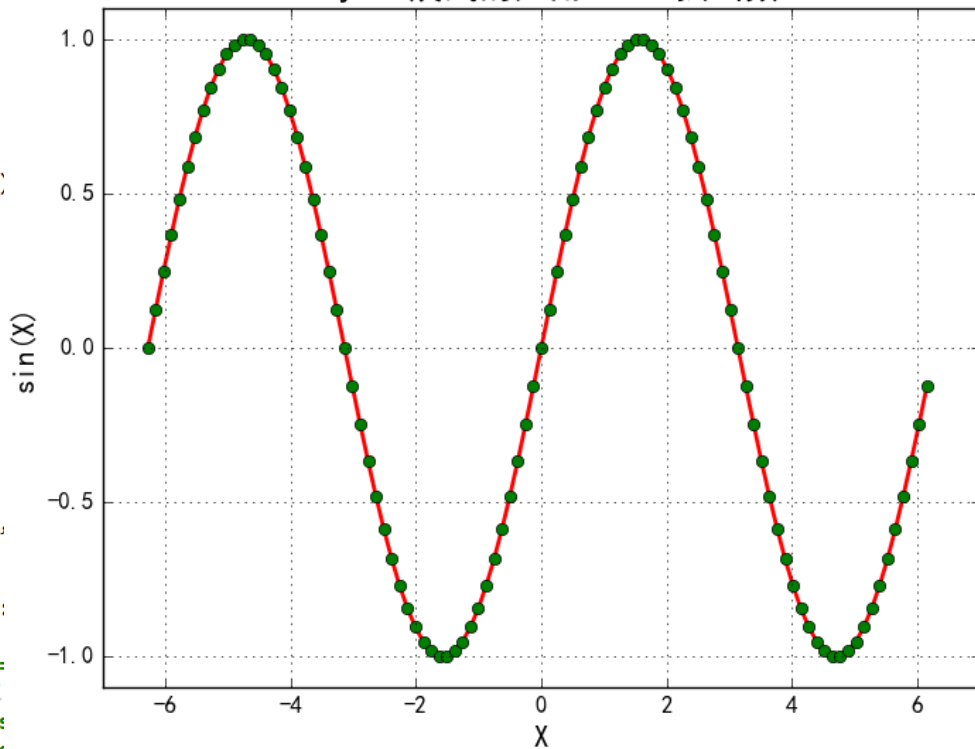
Taylor展式的应用

```
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+1))
    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, 2*np.pi, 100)
    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, ') = ', y[i]
        # print '误差: ', y[i] - sin(x)
    mpl.rcParams['font.sans-serif'] = ['SimHei']
    mpl.rcParams['axes.unicode_minus'] = False
    plt.figure(facecolor='w')
    plt.plot(t, y, 'r-', t, y, 'go', linewidth=2)
    plt.title('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()
```

Taylor展式的应用 - 正弦函数



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(2.38701041073) = 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))
```

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

$$\begin{aligned} \sin \frac{1}{x} &= \frac{1}{x} - \frac{1}{3!} \left(\frac{1}{x} \right)^3 + \frac{1}{5!} \left(\frac{1}{x} \right)^5 - \frac{1}{7!} \left(\frac{1}{x} \right)^7 + \frac{1}{9!} \left(\frac{1}{x} \right)^9 \\ &= \left(x - \frac{1}{\pi} \right) \left(x + \frac{1}{\pi} \right) \left(x - \frac{1}{2\pi} \right) \left(x + \frac{1}{2\pi} \right) \left(x - \frac{1}{3\pi} \right) \left(x + \frac{1}{3\pi} \right) \cdots \end{aligned}$$

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？

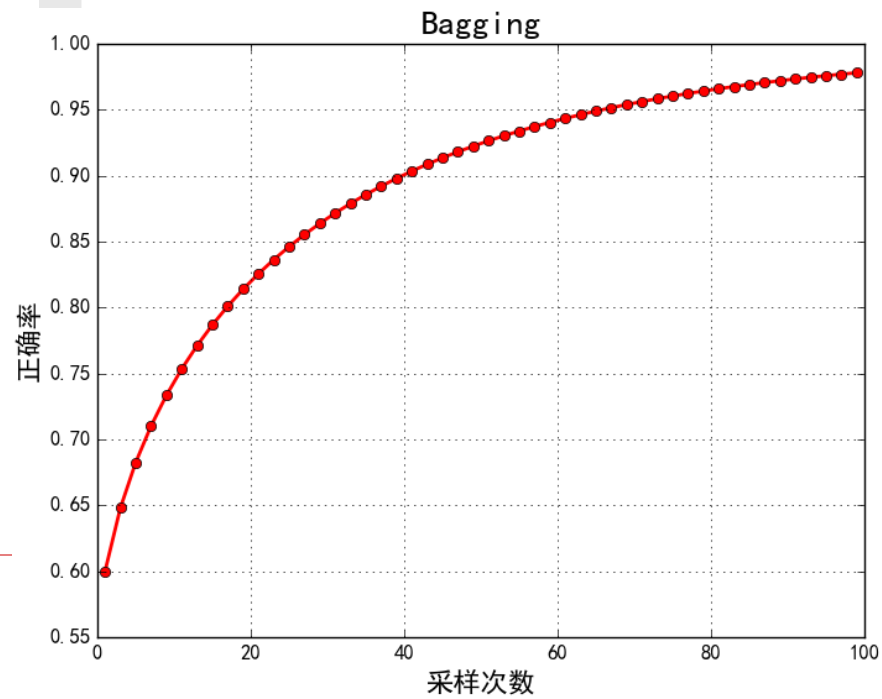
■ 若构造99个分类器呢？

```
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)
```

Ensemble

C:\Python27\python.exe D:/Python/Ensemble.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

$$P(X = x; r, p) = C_{x-1}^{r-1} \cdot p^r \cdot (1-p)^{x-r}$$
$$x \in [r, r+1, r+2, \dots, \infty)$$

- 若记 $X=k$ 为失败的次数，则有：

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

Code

```
import numpy as np
from scipy import special

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

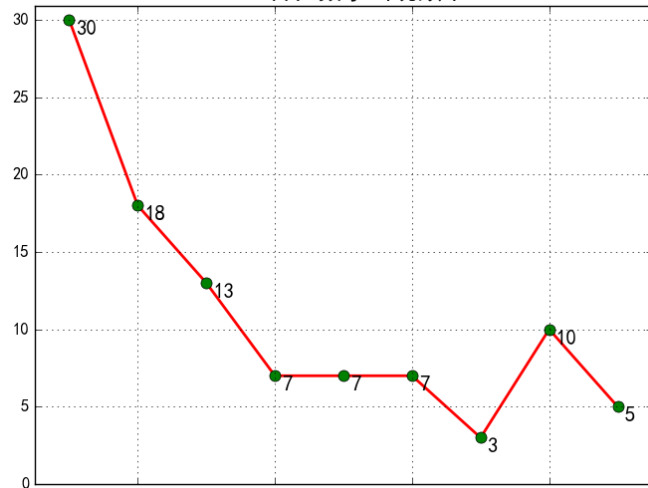
    # 1. 暴力模拟
    if method == 'simulation':
        p = 0.6
        a, b, c = 0, 0, 0
        t, T = 0, 1000000
        while t < T:
            a = b = 0
            while (a <= 11) and (b <= 11):
                if np.random.uniform() < p:
                    a += 1
                else:
                    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

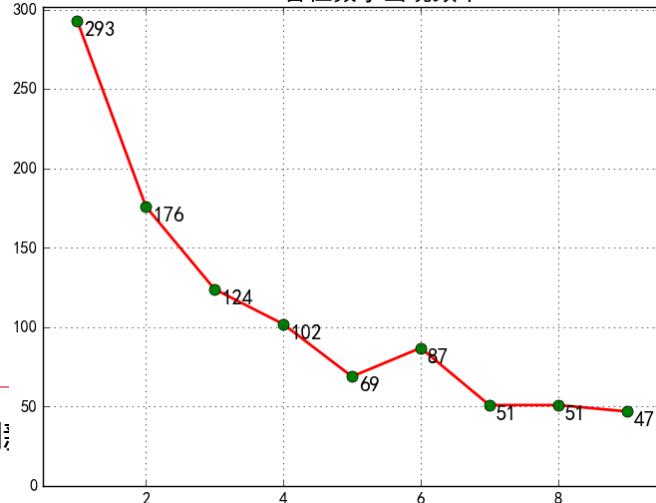
    # 3. 严格计算
    else:
        answer = 0
        p = 0.6 # 每分的胜率
        N = 11 # 每局多少分
        for x in np.arange(N-1): # x为对手得分: 11:9 11:8 11:7 11:6...
            answer += special.comb(N + x - 1, x) * ((1 - p) ** x) * (p ** N)
        p10 = special.comb(2*(N-1), N-1) * ((1-p)*p) ** (N-1) # 10:10的概率
        t = 0
        for n in np.arange(100): # {x0}(0,)|00 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

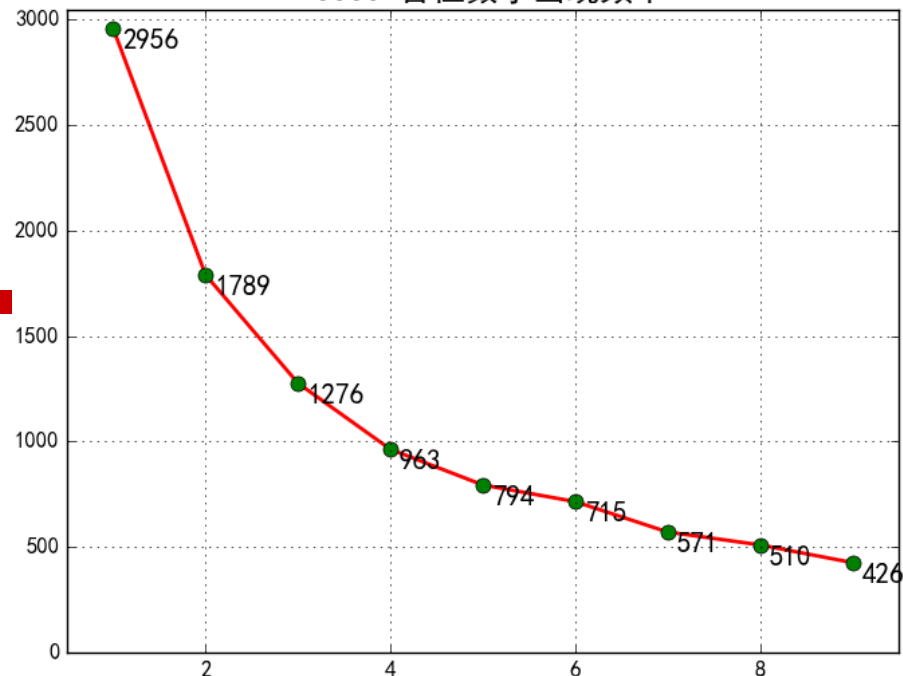
100! 首位数字出现频率



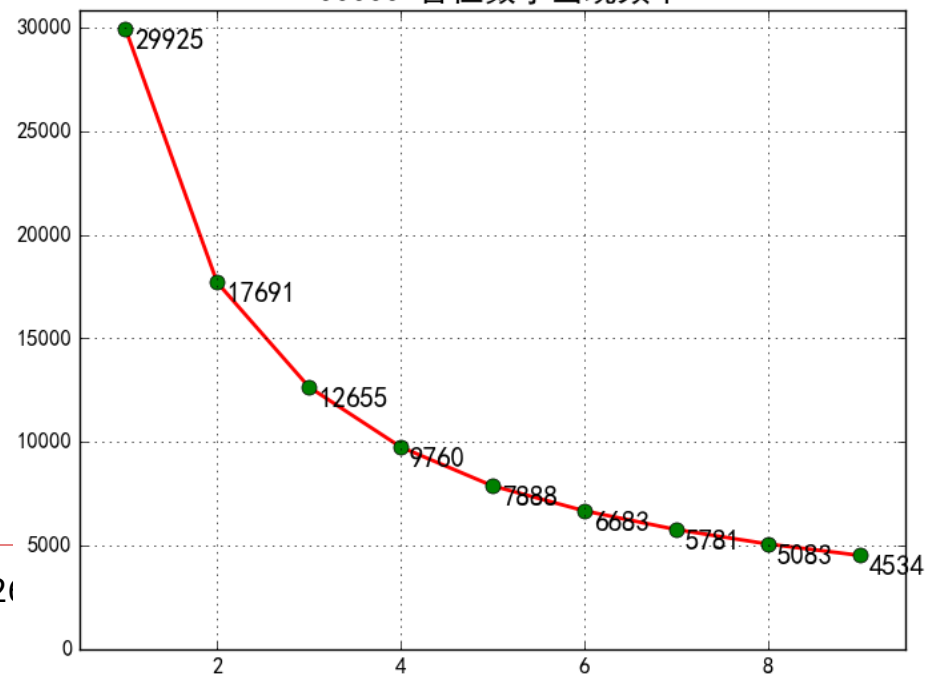
1000! 首位数字出现频率



10000! 首位数字出现频率



100000! 首位数字出现频率



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

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
又又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！