# 【python数据挖掘课程】十四.Scipy调用curve_fit实现曲线拟合



前面系列文章讲过各种知识，包括绘制曲线、散点图、幂分布等，而如何在在散点图一堆点中拟合一条直线，也变得非常重要。这篇文章主要讲述调用Scipy扩展包的curve_fit函数实现曲线拟合，同时计算出拟合的函数、参数等。希望文章对你有所帮助，如果文章中存在错误或不足之处，还请海涵~

前文推荐：

【Python数据挖掘课程】一.安装Python及爬虫入门介绍
【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制
【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐
【Python数据挖掘课程】九.回归模型LinearRegression简单分析氧化物数据
【python数据挖掘课程】十.Pandas、Matplotlib、PCA绘图实用代码补充
【python数据挖掘课程】十一.Pandas、Matplotlib结合SQL语句可视化分析
【python数据挖掘课程】十二.Pandas、Matplotlib结合SQL语句对比图分析
【python数据挖掘课程】十三.WordCloud词云配置过程及词频分析

## 一. Scipy介绍

SciPy (pronounced "Sigh Pie") 是一个开源的数学、科学和工程计算包。它是一款方便、易于使用、专为科学和工程设计的Python工具包，包括统计、优化、整合、线性代数模块、傅里叶变换、信号和图像处理、常微分方程求解器等等。

官方地址：https://www.scipy.org/


Scipy常用的模块及功能如下图所示：
强烈推荐刘神的文章：Scipy高端科学计算 - 刘一痕

| 模块 | 功能 |
| --- | --- |
| scipy.cluster | 矢量量化 / K-均值 |
| scipy.constants | 物理和数学常数 |
| scipy.fftpack | 傅里叶变换 |
| scipy.integrate | 积分程序 |
| scipy.interpolate | 插值 |
| scipy.io | 数据输入输出 |
| scipy.linalg | 线性代数程序 |
| scipy.ndimage | n维图像包 |
| scipy.odr | 正交距离回归 |
| scipy.optimize | 优化 |
| scipy.signal | 信号处理 |
| scipy.sparse | 稀疏矩阵 |
| scipy.spatial | 空间数据结构和算法 |
| scipy.special | 任何特殊数学函数 |
| scipy.stats | 统计 |

Scipy优化和拟合采用的是optimize模块，该模块提供了函数最小值(标量或多维)、曲线拟合和寻找等式的根的有用算法。

## scipy.optimize.curve_fit

**scipy.optimize.curve_fit**(*f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False, check_finite=True, bounds=(-inf, inf), method=None, jac=None, **kwargs*)¶ [source]

Use non-linear least squares to fit a function, f, to data.

Assumes `ydata = f(xdata, *params) + eps`

官方介绍：scipy.optimize.curve_fit

下面将从实例进行详细介绍，包括：

1.调用 numpy.polyfit() 函数实现一次二次多项式拟合；

2.Pandas导入数据后，调用Scipy实现次方拟合；

3.实现np.exp()形式e的次方拟合；

4.实现三个参数的形式拟合；

5.最后通过幂率图形分析介绍自己的一些想法和问题。

# 二. 曲线拟合

### 1.多项式拟合

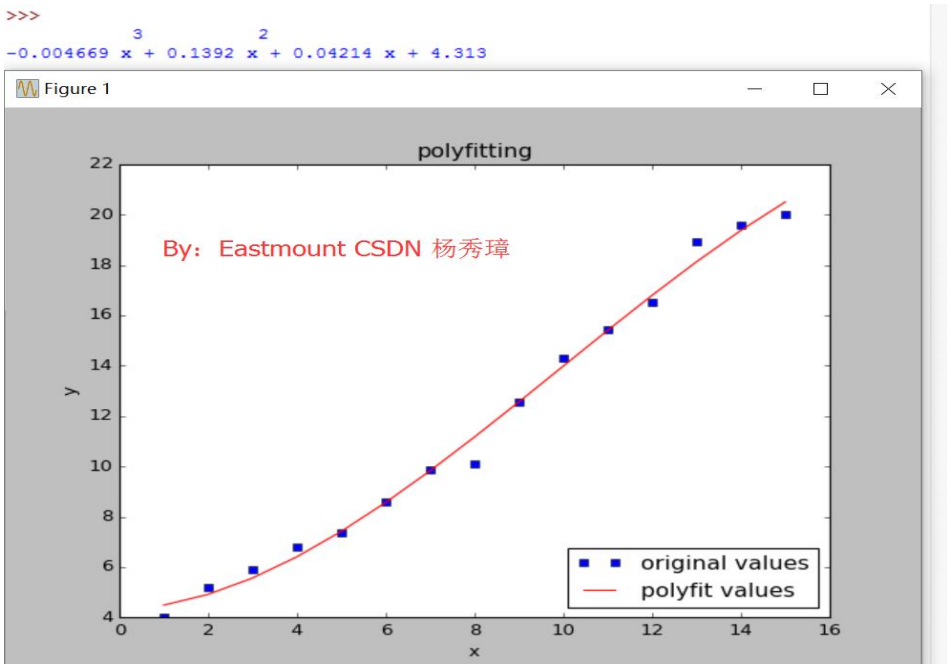
首先通过numpy.arange定义x、y坐标，然后调用polyfit()函数进行3次多项式拟合，最后调用Matplotlib函数进行散点图绘制(x,y)坐标，并绘制预测的曲线。

完整代码：

```
#encoding=utf-8
import numpy as np
import matplotlib.pyplot as plt

#定义x、y散点坐标
x = np.arange(1, 16, 1)
num = [4.00, 5.20, 5.900, 6.80, 7.34,
       8.57, 9.86, 10.12, 12.56, 14.32,
       15.42, 16.50, 18.92, 19.58, 20.00]
y = np.array(num)

#用3次多项式拟合
f1 = np.polyfit(x, y, 3)
```

```
p1 = np.poly1d(f1)
                    print(p1)

#也可使用yvals=np.polyval(f1, x)
yvals = p1(x)  #拟合y值

#绘图
plot1 = plt.plot(x, y, 's',label='original values')
plot2 = plt.plot(x, yvals, 'r',label='polyfit values')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc=4) #指定legend的位置右下角
plt.title('polyfitting')
plt.show()
plt.savefig('test.png')
```

输出结果如下图所示，包括蓝色的正方形散点和红色的拟合曲线。
多项式函数为: y=-0.004669 x3 + 0.1392 x2 + 0.04214 x + 4.313




补充：给出函数，可以用 Origin 进行绘图的，也比较方便。

## 2.e的b/x次方拟合

下面采用Scipy的curve_fit()对上面的数据进行e的b/x次方拟合。数据集如下：

```
x = np.arange(1, 16, 1)
num = [4.00, 5.20, 5.900, 6.80, 7.34,
       8.57, 9.86, 10.12, 12.56, 14.32,
       15.42, 16.50, 18.92, 19.58, 20.00]
y = np.array(num)
```

其中，x坐标从1到15，y对应Num数组，比如第一个点(1, 4.00)、最后一个点(15, 20.00)。

然后调用curve_fit()函数，核心步骤：

(1) 定义需要拟合的函数类型，如：

```
def func(x, a, b):
    return a*np.exp(b/x)
```

(2) 调用 popt, pcov = curve_fit(func, x, y) 函数进行拟合，并将拟合系数存储在popt中，a=popt[0]、b=popt[1]进行调用；

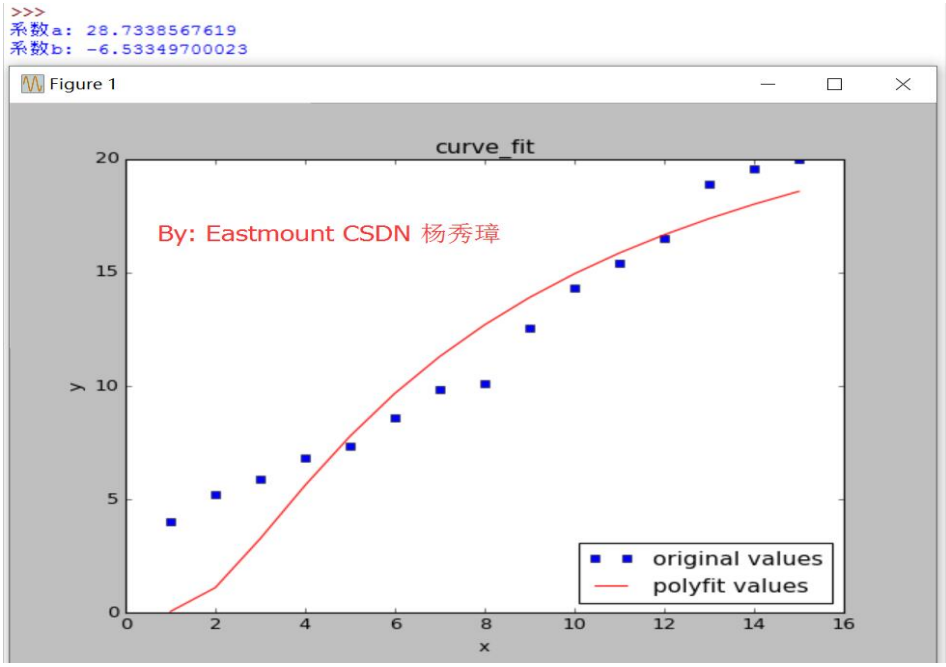
(3) 调用func(x, a, b)函数，其中x表示横轴表，a、b表示对应的参数。

完整代码如下：

```
#encoding=utf-8
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

#自定义函数 e指数形式
def func(x, a, b):
    return a*np.exp(b/x)

#定义x、y散点坐标
x = np.arange(1, 16, 1)
num = [4.00, 5.20, 5.900, 6.80, 7.34,
       8.57, 9.86, 10.12, 12.56, 14.32,
       15.42, 16.50, 18.92, 19.58, 20.00]
y = np.array(num)

#非线性最小二乘法拟合
popt, pcov = curve_fit(func, x, y)
#获取popt里面是拟合系数
a = popt[0]
b = popt[1]
yvals = func(x,a,b) #拟合y值
print u'系数a:', a
print u'系数b:', b
```

```
#绘图
     plot1 = plt.plot(x, y, 's',label='original values')
plot2 = plt.plot(x, yvals, 'r',label='polyfit values')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc=4) #指定legend的位置右下角
plt.title('curve_fit')
plt.show()
plt.savefig('test2.png')
```

绘制的图形如下所示，拟合效果没有多项式的好。




## 3.aX的b次方拟合

第三种方法是通过Pandas导入数据，因为通常数据都会存储在csv、excel或数据库中，所以这里结合读写数据绘制a*x的b次方形式。

假设本地存在一个data.csv文件，数据集如下图所示：

| | A | B | C | D |
|---|---|---|---|---|
| 1 | x | y | | |
| 2 | 1 | 4 | | |
| 3 | 2 | 5.2 | | |
| 4 | 3 | 5.9 | | |
| 5 | 4 | 6.8 | | |
| 6 | 5 | 7.34 | | |
| 7 | 6 | 8.57 | | |
| 8 | 7 | 9.86 | | |
| 9 | 8 | 10.12 | | |
| 10 | 9 | 12.56 | | |
| 11 | 10 | 14.32 | | |
| 12 | 11 | 15.42 | | |
| 13 | 12 | 16.5 | | |
| 14 | 13 | 18.92 | | |
| 15 | 14 | 19.58 | | |
| 16 | 15 | 20 | | |

然后调用Pandas扩展包读取数据，并获取x、y值显示，这段代码如下：

```
#导入数据及x、y散点坐标
data = pd.read_csv("data.csv")
print data
print(data.shape)
print(data.head(5)) #显示前5行数据
x = data['x'] #获取x列
y = data['y'] #获取y列
print x
print y
```

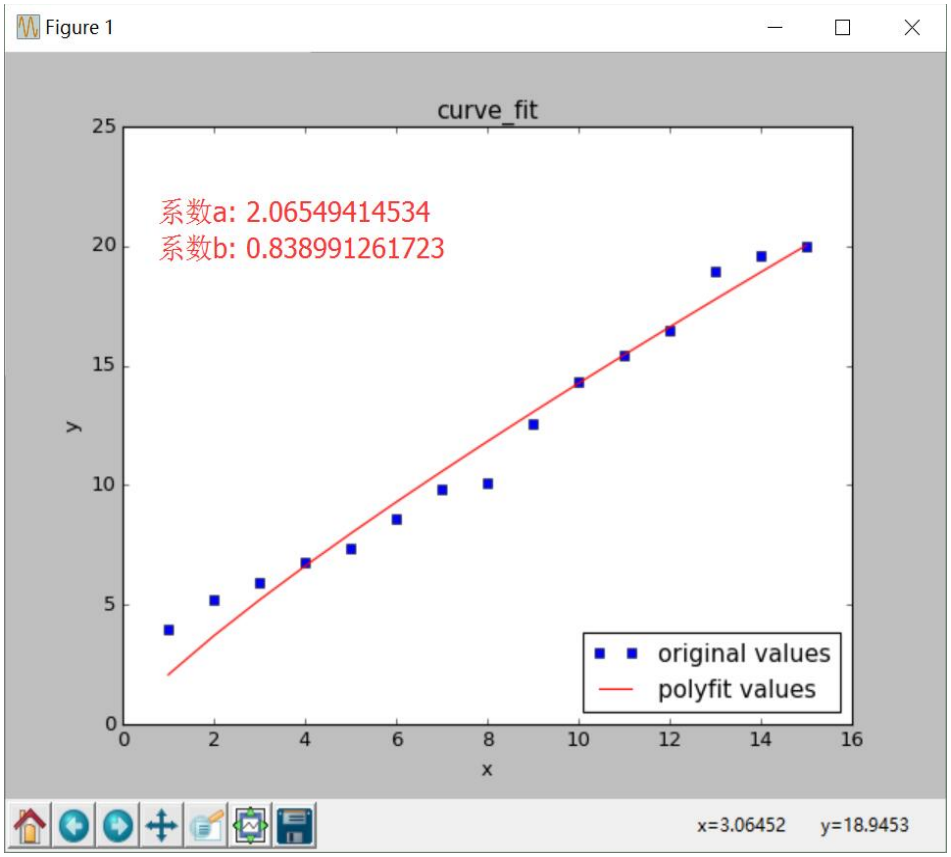
比如 print y 输出结果：

```
0      4.00
1      5.20
2      5.90
3      6.80
4      7.34
5      8.57
6      9.86
7     10.12
8     12.56
9     14.32
10    15.42
11    16.50
12    18.92
13    19.58
14    20.00
Name: y, dtype: float64
```

最后完整的拟合代码如下所示：

```
#encoding=utf-8
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import pandas as pd

#自定义函数 e指数形式
def func(x, a, b):
    return a*pow(x,b)

#导入数据及x、y散点坐标
data = pd.read_csv("data.csv")
print data
print(data.shape)
print(data.head(5)) #显示前5行数据
x = data['x']
y = data['y']
print x
print y

#非线性最小二乘法拟合
popt, pcov = curve_fit(func, x, y)
#获取popt里面是拟合系数
a = popt[0]
b = popt[1]
yvals = func(x,a,b) #拟合y值
print u'系数a:', a
print u'系数b:', b

#绘图
plot1 = plt.plot(x, y, 's',label='original values')
plot2 = plt.plot(x, yvals, 'r',label='polyfit values')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc=4) #指定legend的位置右下角
plt.title('curve_fit')
plt.savefig('test3.png')
plt.show()
```

输出结果如下图所示：



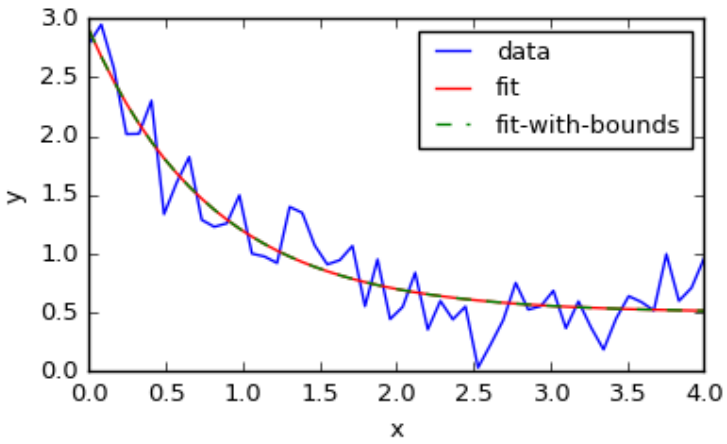

## 4.三个参数拟合

最后介绍官方给出的实例，讲述传递三个参数，通常为 a*e(b/x)+c形式。

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def func(x, a, b, c):
    return a * np.exp(-b * x) + c

# define the data to be fit with some noise
xdata = np.linspace(0, 4, 50)
y = func(xdata, 2.5, 1.3, 0.5)
y_noise = 0.2 * np.random.normal(size=xdata.size)
ydata = y + y_noise
plt.plot(xdata, ydata, 'b-', label='data')

# Fit for the parameters a, b, c of the function `func`
popt, pcov = curve_fit(func, xdata, ydata)
```

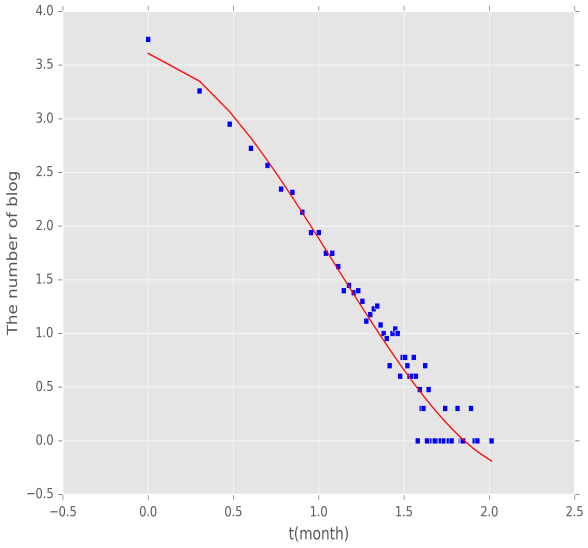
```
plt.plot(xdata, func(xdata, *popt), 'r-', label='fit')

# Constrain the optimization to the region of ``0 < a < 3``, ``0 < b < 2``
# and ``0 < c < 1``:
popt, pcov = curve_fit(func, xdata, ydata, bounds=(0, [3., 2., 1.]))
plt.plot(xdata, func(xdata, *popt), 'g--', label='fit-with-bounds')

plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

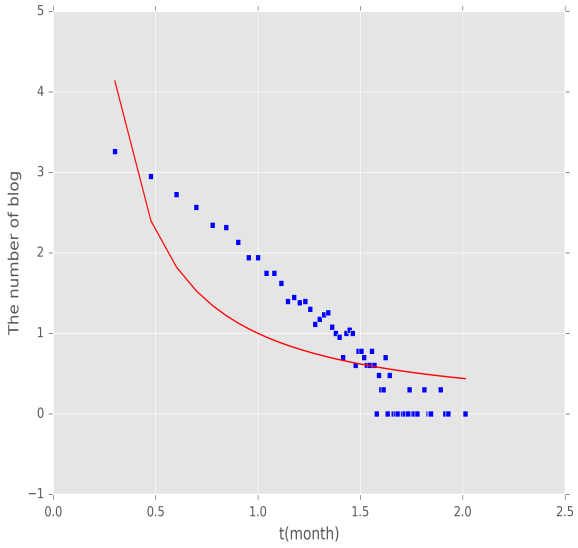
输出结果如下图所示：




## 三. 幂律分布拟合及疑问

下面是我幂率分布的实验，因为涉及到保密，所以只提出几个问题。
图1是多项式的拟合结果，基本符合图形趋势。
图2是幂指数拟合结果，幂指数为-1.18也符合人类的基本活动规律。




问题：

1.为什么幂律分布拟合的图形不太好，而指数却很好；

2.计算幂指数及拟合是否只对中间那部分效果好的进行拟合；

3.e的b/x次方、多项方程、x的b次方哪个效果好？

间隔时间 $\tau$ 定义为用户两次连续活动之间的时间差，例如用户在 $t_1$ 时刻被记录到活动一次，在 $t_2$ 时刻又被记录到活动一次，则用户的间隔时间定义为 $(t_2-t_1)$。间隔时间作为反映事件发生快慢的一个重要概念，在现实中有着极其重要的作用，常被作为刻画人类活动模式的度量之一。

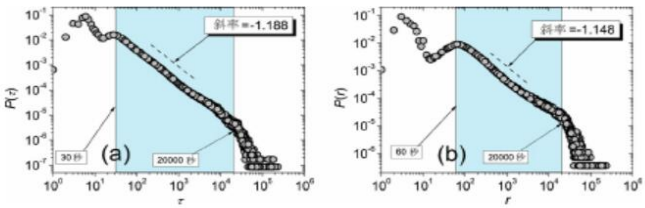



图 2–1 短信用户间隔时间和响应时间分布

图 2–1 (a) 表示用户发送短信的间隔时间分布，该图结果证实了人类短信通信行为的间隔时间分布符合幂指数 $\alpha \simeq 1.188$ 的幂律分布 $P(\tau) \sim \tau^{-\alpha}$，该幂指数是通过最大似然估计得到，并通过了 $K-S$ 检验[1 38]（该文章所有幂律分布结果幂指数都经类似处理）。其它的人类通讯活动的实证结果也证实了类似的结果，虽

最后希望这篇文章对你有所帮助，尤其是我的学生和接触数据挖掘、**机器学习**的博友。这篇文字主要是介绍拟合，记录一些代码片段，作为在线笔记，也希望对你有所帮助。同时，后面论文写完会opensource系列文章。

一醉一轻舞，一梦一轮回。一曲一人生，一世一心愿。

(By:Eastmount 2017-05-07 下午3点半 http://blog.csdn.net/eastmount/ )

Eastmount 博客专家
发布了444 篇原创文章 · 获赞 5908 · 访问量 484万+