

# 本文使用高斯牛顿法实现曲线拟合

## 目的

一方面出于学习的目的完成这个小项目，另一方面出于为其他的同学提供一个相关的代码学习资料

## 拟合方程

$$y = e^{ax^3+bx^2+cx+d} + w$$

## 数据产生（随机数产生，误差服从高斯分布）

$$w \sim (0, \sigma^2)$$

## 误差定义

$$er_i = y_i - e^{ax_i^3+bx_i^2+cx_i+d}$$

## 求偏导（最关键）

$$\begin{cases} \frac{\partial er_i}{\partial a} = -x_i^3 e^{ax_i^3+bx_i^2+cx_i+d} \\ \frac{\partial er_i}{\partial b} = -x_i^2 e^{ax_i^3+bx_i^2+cx_i+d} \\ \frac{\partial er_i}{\partial c} = -x_i e^{ax_i^3+bx_i^2+cx_i+d} \\ \frac{\partial er_i}{\partial d} = -e^{ax_i^3+bx_i^2+cx_i+d} \end{cases}$$

```
1 # 产生固定间隔的数组
2 import numpy as np
3
4 start = 0 # 起始值
5 stop = 1.5 # 结束值（不包含在数组中）
6 step = 0.01 # 间隔
7
```

```

8  arr = np.arange(start, stop, step)
9  # print(arr)
10
11
12  mean = 0    # 均值
13  std = 10    # 标准差
14  size = int((stop-start)/step) # 数组大小
15
16  noise = np.random.normal(mean, std, size)
17  # print(noise)
18
19
20  # 生成观测数据x,y
21
22  # 定义函数参数
23  a = 0.5
24  b = 1
25  c = 2.0
26  d = -1
27  x=arr
28  obs_y=np.exp(a*x**3 + b*x**2 + c*x + d)+noise

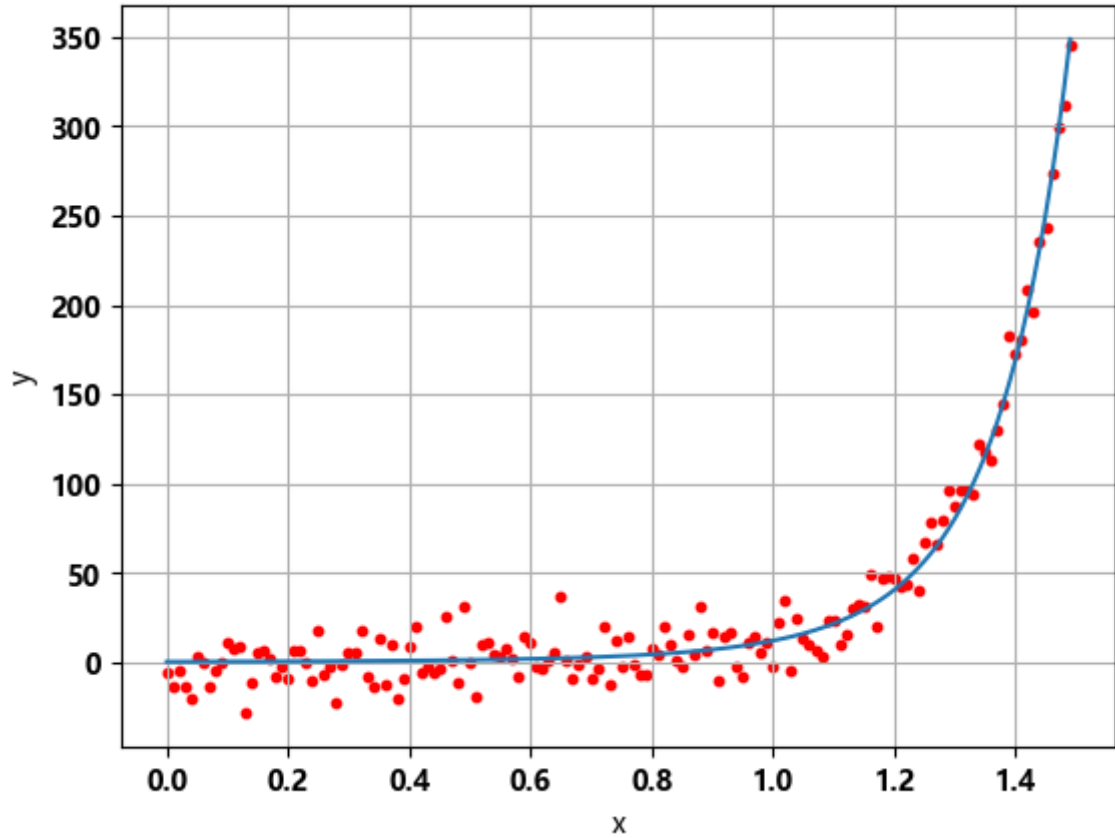
```

```

1  import matplotlib.pyplot as plt
2  import matplotlib
3  matplotlib.rc("font",family='MicroSoft YaHei',weight="bold")
4  # 绘制图像
5
6  # 定义函数
7  def f(x,A):
8      return np.exp(A[0]*x**3 + A[1]*x**2 + A[2]*x + A[3])
9
10 # 计算对应的 y 值
11 y = f(x,[a,b,c,d])
12
13 plt.plot(x, y)
14 plt.scatter(x, obs_y, color='red', s=10,label='Scatter Points')
15 plt.xlabel('x')
16 plt.ylabel('y')
17 plt.title('正确地函数图像')
18 plt.grid(True)
19 plt.show()

```

正确地函数图像



高斯牛顿迭代进行曲线拟合（我理解为间接平差升级版）

直接求得解析解

$$V = Bx - L$$

$$x = (B^T B)^{-1} (B^T L)$$

$$\begin{bmatrix} er_1 \\ er_2 \\ \vdots \\ er_{n-1} \\ er_n \end{bmatrix} = B \begin{bmatrix} \Delta a \\ \Delta b \\ \Delta c \\ \Delta d \end{bmatrix} - L$$

$$B = \begin{bmatrix} -x_1^3 e^{a_0 x_1^3 + b_0 x_1^2 + c_0 x_1 + d_0} & \dots & \dots & -e^{a_0 x_1^3 + b_0 x_1^2 + c_0 x_1 + d_0} \\ -x_2^3 e^{a_0 x_2^3 + b_0 x_2^2 + c_0 x_2 + d_0} & \vdots & \vdots & -e^{a_0 x_2^3 + b_0 x_2^2 + c_0 x_2 + d_0} \\ \vdots & \vdots & \vdots & \vdots \\ -x_{n-1}^3 e^{a_0 x_{n-1}^3 + b_0 x_{n-1}^2 + c_0 x_{n-1} + d_0} & \vdots & \vdots & -e^{a_0 x_{n-1}^3 + b_0 x_{n-1}^2 + c_0 x_{n-1} + d_0} \\ -x_n^3 e^{a_0 x_n^3 + b_0 x_n^2 + c_0 x_n + d_0} & \dots & \dots & -e^{a_0 x_n^3 + b_0 x_n^2 + c_0 x_n + d_0} \end{bmatrix}$$

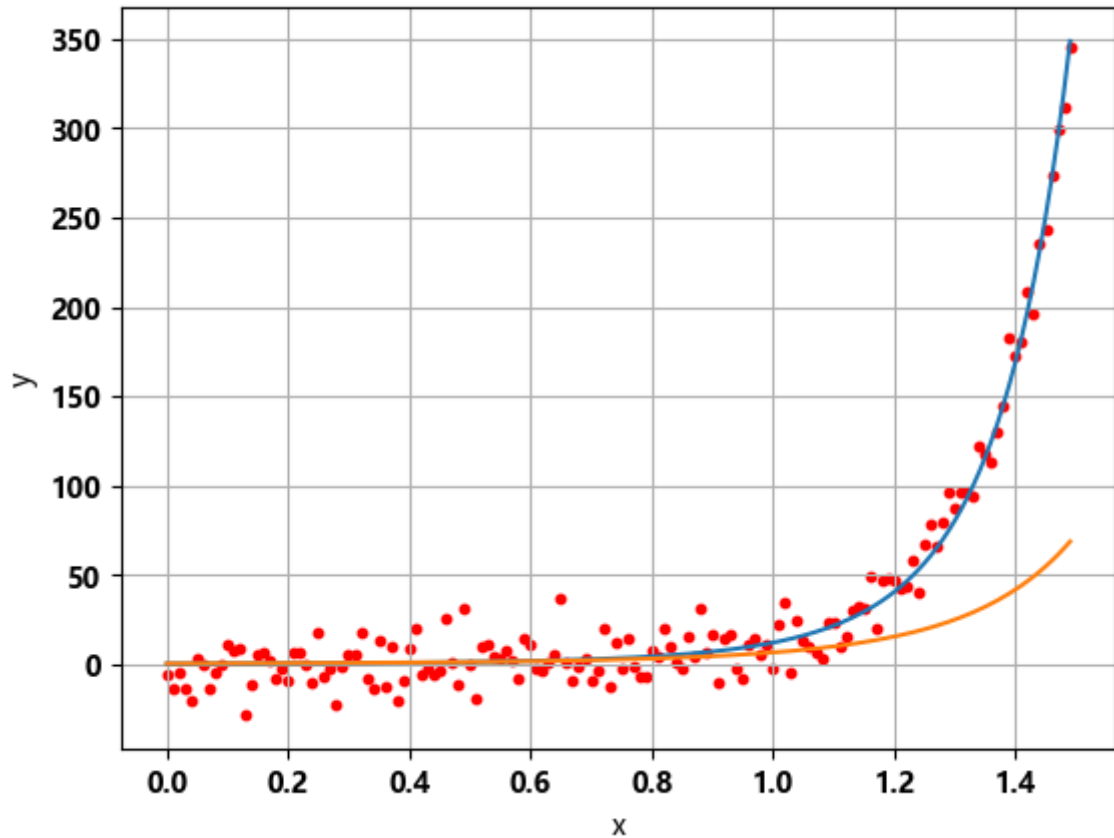
$$L = \begin{bmatrix} e^{a_0 x_1^3 + b_0 x_1^2 + c_0 x_1 + d_0} - y_1 \\ e^{a_0 x_2^3 + b_0 x_2^2 + c_0 x_2 + d_0} - y_2 \\ \vdots \\ e^{a_0 x_{n-1}^3 + b_0 x_{n-1}^2 + c_0 x_{n-1} + d_0} - y_{n-1} \\ e^{a_0 x_n^3 + b_0 x_n^2 + c_0 x_n + d_0} - y_n \end{bmatrix}$$

```

1  # 定义a,b,c,d的初始值(随便定)
2  a0=0.3
3  b0=0.7
4  c0=1.6
5  d0=-0.7
6
7  # 定义函数,A为参数数组
8  def f0(x,A):
9      return np.exp(A[0]*x**3 + A[1]*x**2 + A[2]*x + A[3])
10
11 # 计算对应的 y 值
12 y0 = f0(x,[a0,b0,c0,d0])
13
14 plt.plot(x, y)
15 plt.plot(x, y0)
16 plt.scatter(x, obs_y, color='red', s=10,label='Scatter Points')
17 plt.xlabel('x')
18 plt.ylabel('y')
19 plt.title('正确地函数图像')
20 plt.grid(True)
21 plt.show()

```

正确地函数图像



```

1  # 进行间接平差求解
2  # 构建B矩阵
3  a0_temp=a0
4  b0_temp=b0
5  c0_temp=c0
6  d0_temp=d0
7
8  approx=np.array([a0_temp,b0_temp,c0_temp,d0_temp])
9
10 # 迭代次数
11 n=10
12 dieDaiResult = np.zeros((n, 4))
13 # 用一个
14
15 for j in range(n):
16     B=np.zeros((size,4))
17     L=np.zeros((size,1))
18     for i in range(size):
19         B[i,0]=-x[i]**3*f0(x[i],approx)
20         B[i,1]=-x[i]**2*f0(x[i],approx)
21         B[i,2]=-x[i]*f0(x[i],approx)
22         B[i,3]=-f0(x[i],approx)
23         L[i,0]=f0(x[i],approx)-obs_y[i]
24

```

```

25     arr_B=np.array(B)
26     arr_L=np.array(L)
27     tem1=np.linalg.inv(np.dot(np.transpose(arr_B),arr_B))
28     tem2=np.dot(np.transpose(arr_B),arr_L)
29
30     delta_x=np.dot(tem1,tem2)
31     # 这里进行下一次迭代
32     approx=approx+delta_x.flatten()
33     # list_approx=approx.
34     dieDaiResult[j,:]=approx
35
36 fig = plt.figure(figsize=(10, 5))
37 plt.subplot(1, 2, 1)
38 # 标准曲线
39 plt.plot(x, y)
40 # 初始曲线
41 plt.plot(x, y0)
42 # 迭代曲线
43 for i in range(n-8,n):
44     xishu=dieDaiResult[i]
45     print(xishu)
46     tem_y=f0(x,xishu)
47     plt.plot(x, tem_y,color='#aa00ff',linestyle='--')
48 # plt.plot(x, y1,color='#aa00ff')
49 plt.scatter(x, obs_y, color='red', s=10,label='Scatter Points')
50 # plt.plot(x, y1)
51 plt.xlabel('x')
52 plt.ylabel('y')
53 plt.title('正确地函数图像')
54 plt.grid(True)
55
56 # 创建第二个子图
57 plt.subplot(1, 2, 2)
58 # 标准曲线
59 plt.plot(x, y)
60 # 初始曲线
61 plt.plot(x, y0)
62 # 迭代曲线
63 for i in range(n-8,n):
64     xishu=dieDaiResult[i]
65     print(xishu)
66     tem_y=f0(x,xishu)
67     plt.plot(x, tem_y,color='#aa00ff',linestyle='--')
68 # plt.plot(x, y1,color='#aa00ff')
69 plt.scatter(x, obs_y, color='red', s=10,label='Scatter Points')

```

```

70 # plt.plot(x, y1)
71 plt.xlabel('x')
72 plt.ylabel('y')
73 plt.title('局部放大图')
74
75 # 设置局部放大范围
76 plt.xlim(1.32, 1.44) # 设置 x 轴范围
77 plt.ylim(100, 250)   # 设置 y 轴范围
78 plt.grid(True)
79 plt.show()
80

```

```

1  [ 15.18948191 -48.88914189  58.18225421 -21.87081973]
2  [ 11.06123187 -37.62392869  48.70588659 -19.61002625]
3  [ 2.98965953 -9.4558721   16.33702568 -7.39086396]
4  [-0.25717518  2.86430852   0.85419677 -0.95130386]
5  [ 2.28260343 -6.65929933  12.58667395 -5.68529249]
6  [-0.53772553  3.94032672 -0.50831067 -0.38253999]
7  [ 2.22309202 -6.36924235  12.12862682 -5.44958593]
8  [-0.55472966  4.00668407 -0.59381963 -0.34621364]
9  [ 15.18948191 -48.88914189  58.18225421 -21.87081973]
10 [ 11.06123187 -37.62392869  48.70588659 -19.61002625]
11 [ 2.98965953 -9.4558721   16.33702568 -7.39086396]
12 [-0.25717518  2.86430852   0.85419677 -0.95130386]
13 [ 2.28260343 -6.65929933  12.58667395 -5.68529249]
14 [-0.53772553  3.94032672 -0.50831067 -0.38253999]
15 [ 2.22309202 -6.36924235  12.12862682 -5.44958593]
16 [-0.55472966  4.00668407 -0.59381963 -0.34621364]

```

