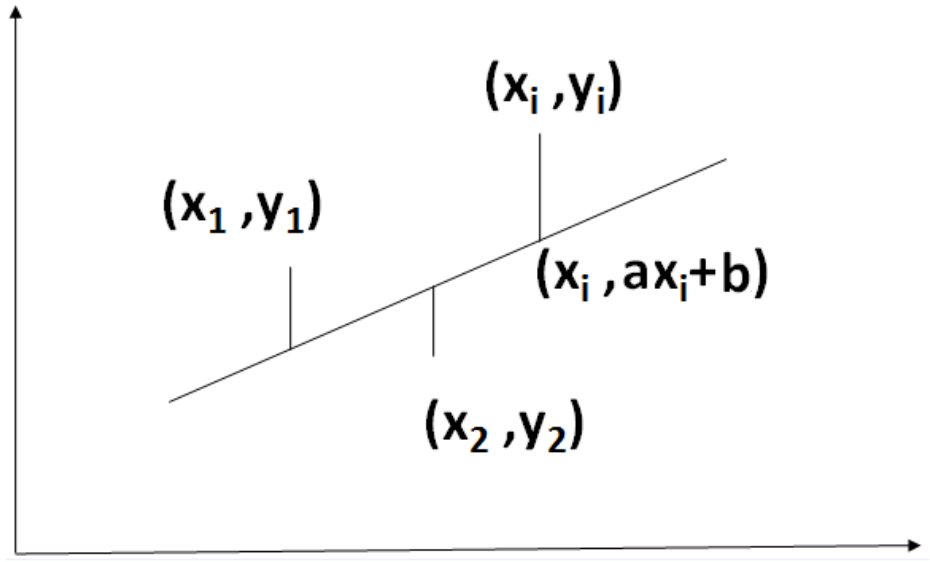
Python 多项式拟合（一元回归）

一元一阶线性拟合：

假设存在一条线性函数尽量能满足所有的点：**y=ax+b** .对所有点的的公式为：

$$y_i = a_i x_i + b_i + \beta_i, (i = 1, 2, \cdots, n, \cdots)$$

残差值β = 实际值y - 估计值y，β 应尽量小，当 β = 0 时，则完全符合一元线性方程：**y=ax+b**



通过最小二乘法计算残差和最小：

$$Q^2 = \sum \beta_i^2 = \sum (y_i - \bar{y}_i)^2 = \sum (y_i - a_i x_i - b_i)^2 = min$$

根据微积分，当 Q 对 a、b 的一阶偏导数为了0时，Q 达到最小。

$$\begin{cases} \sum (y_i - a_i x_i - b_i) = 0 \\ \sum (y_i - a_i x_i - b_i) x_i = 0 \end{cases} \Rightarrow \begin{cases} \sum y_i = a \sum x_i + nb \\ \sum y_i x_i = a \sum x_i^2 + b \sum x_i \end{cases}$$

解方程组，求 a、b 的值：

$$b = \bar{y} - a\bar{x}$$

$$a = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

示例：

ID	1	2	3	4	5	6	7	8	9	10
贷款金额(X)	31.50	134.22	200.40	244.43	300.61	320.39	345.66	449.43	524.70	544.47
还款金额(Y)	16.21	35.29	59.23	52.47	67.44	73.03	61.34	129.37	163.45	99.81

ID	11	12	13	14	15	16	17	18	19	20
贷款金额(X)	673.93	724.19	765.96	826.77	828.30	833.70	867.83	1006.42	1104.28	1237.61
还款金额(Y)	251.34	263.43	115.12	281.21	317.12	291.25	216.38	333.24	424.80	201.27

如客户的贷款金额及还款金额情况，设 x 为贷款金额，预测Y为还款金额。（也可以当做收入与消费的情况）

通过公式计算相应的值：

ID	催收金额(X)	还款金额(Y)	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
1	31.50	16.21	-566.74	-156.43	88655.14	321194.23
2	134.22	35.29	-464.02	-137.35	63733.15	215314.56
3	200.40	59.23	-397.84	-113.41	45119.03	158276.67
4	244.43	52.47	-353.81	-120.17	42517.35	125181.52
5	300.61	67.44	-297.63	-105.20	31310.68	88583.62
6	320.39	73.03	-277.85	-99.61	27676.64	77200.62
7	345.66	61.34	-252.58	-111.30	28112.15	63796.66
8	449.43	129.37	-148.81	-43.27	6439.01	22144.42
9	524.70	163.45	-73.54	-9.19	675.83	5408.13
10	544.47	99.81	-53.77	-72.83	3916.07	2891.21
11	673.93	251.34	75.69	78.70	5956.80	5728.98
12	724.19	263.43	125.95	90.79	11435.00	15863.40
13	765.96	115.12	167.72	-57.52	-9647.25	28130.00
14	826.77	281.21	228.53	108.57	24811.50	52225.96
15	828.30	317.12	230.06	144.48	33239.07	52927.60
16	833.70	291.25	235.46	118.61	27927.91	55441.41
17	867.83	216.38	269.59	43.74	11791.87	72678.77
18	1006.42	333.24	408.18	160.60	65553.71	166610.91
19	1104.28	424.80	506.04	252.16	127603.05	256076.48
20	1237.61	201.27	639.37	28.63	18305.16	408794.00
合计	**11964.80**	**3452.80**			**655131.86**	**2194469.14**
均值	$\bar{x} =$ 598.24	$\bar{y} =$ 172.64				

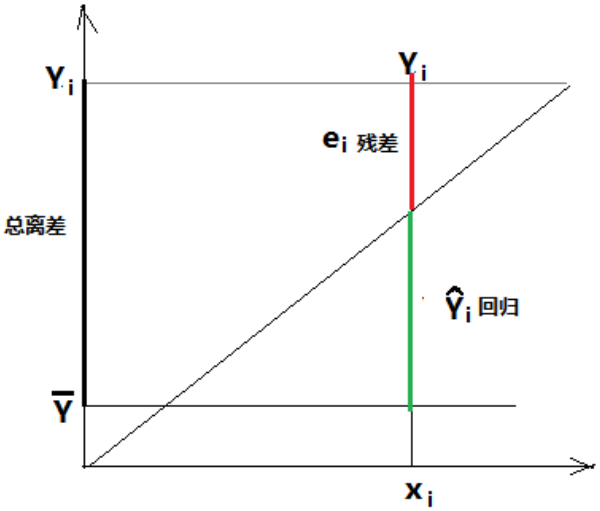
解得：

a = 655131.86/2194469.14 = 0.2985

b = 172.64-a*598.24 = -5.93464

即得回归方程为： **Ÿ = 0.2985*x - 5.93464**

回归方程验证：



Y 的第 i 个观察值与样本值的离差， 点与回归线在 Y 轴上的距离。总离差分解为两部分为：

$$y_i = Y_i - \bar{Y} = (Y_i - \hat{Y}_i) + (\hat{Y}_i - \bar{Y}) = e_i + \hat{y}_i$$

实际观测值与回归拟合值之差，为回归直线不能解释的部分：

$$e_i = (Y_i - \hat{Y}_i)$$

样本回归拟合值与观测值的平均值之差，为回归直线可解释的部分：

$$\hat{y}_i = (\hat{Y}_i - \bar{Y})$$

其中，设总体平方和为：

$$\begin{aligned} TSS &= \sum \left(Y_i - \bar{Y}\right)^2 \\ &= \sum \left(Y_i - \hat{Y}_i + \hat{Y}_i - \bar{Y}\right)^2 \\ &= \sum \left(Y_i - \hat{Y}_i\right)^2 + 2\sum \left(Y_i - \hat{Y}_i\right)\left(\hat{Y}_i - \bar{Y}\right) + \sum \left(\hat{Y}_i - \bar{Y}\right)^2 \\ &= \sum \left(Y_i - \hat{Y}_i\right)^2 + \sum \left(\hat{Y}_i - \bar{Y}\right)^2 = \sum e_i^2 + \sum \hat{y}_i^2 \\ &= RSS + ESS \end{aligned}$$

即得 回归平方和为：

$$ESS = \sum \hat{y}_i^2 = \sum (\hat{Y}_i - \bar{Y})^2$$

残差平方和为：

$$RSS = \sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2$$

即： **TSS=ESS+RSS**

样本中，TSS不变， 如果实际观测点离样本回归线越近，则ESS在TSS中占的比重越大.

拟合优度：

$$R^2 = \frac{ESS}{TSS} = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

R^2 为（样本）可决系数/判定系数（coefficient of determination），取值范围：[0，1]。R^2越接近1，说明实际观测点离样本线越近，拟合优度越高。一般地要求$R^2 \geq 0.7$。

ID	催收金额 X_i	还款金额 Y_i	$\hat{Y}=0.2985*x-5.93464$	$Y_i-\bar{Y}$	$(Y_i-\bar{Y})^2$	$Y_i-\hat{Y}_i$	$(Y_i-\hat{Y}_i)^2$
1	31.50	16.21	3.468110	-156.43	24470.34	12.74	162.36
2	134.22	35.29	34.130030	-137.35	18865.02	1.16	1.35
3	200.40	59.23	53.884760	-113.41	12861.83	5.35	28.57
4	244.43	52.47	67.027715	-120.17	14440.83	-14.56	211.93
5	300.61	67.44	83.797445	-105.20	11067.04	-16.36	267.57
6	320.39	73.03	89.701775	-99.61	9922.15	-16.67	277.95
7	345.66	61.34	97.244870	-111.30	12387.69	-35.90	1289.16
8	449.43	129.37	128.220215	-43.27	1872.29	1.15	1.32
9	524.70	163.45	150.688310	-9.19	84.46	12.76	162.86
10	544.47	99.81	156.589655	-72.83	5304.21	-56.78	3223.93
11	673.93	251.34	195.233465	78.70	6193.69	56.11	3147.94
12	724.19	263.43	210.236075	90.79	8242.82	53.19	2829.59
13	765.96	115.12	222.704420	-57.52	3308.55	-107.58	11574.41
14	826.77	281.21	240.856205	108.57	11787.44	40.35	1628.43
15	828.30	317.12	241.312910	144.48	20874.47	75.81	5746.71
16	833.70	291.25	242.924810	118.61	14068.33	48.33	2335.32
17	867.83	216.38	253.112615	43.74	1913.19	-36.73	1349.29
18	1006.42	333.24	294.481730	160.60	25792.36	38.76	1502.20
19	1104.28	424.80	323.692940	252.16	63584.67	101.11	10222.64
20	1237.61	201.27	363.491945	28.63	819.68	-162.22	26315.96
均值		$\bar{Y}$ =172.64			TSS= 267861.07		RSS= 72279.48

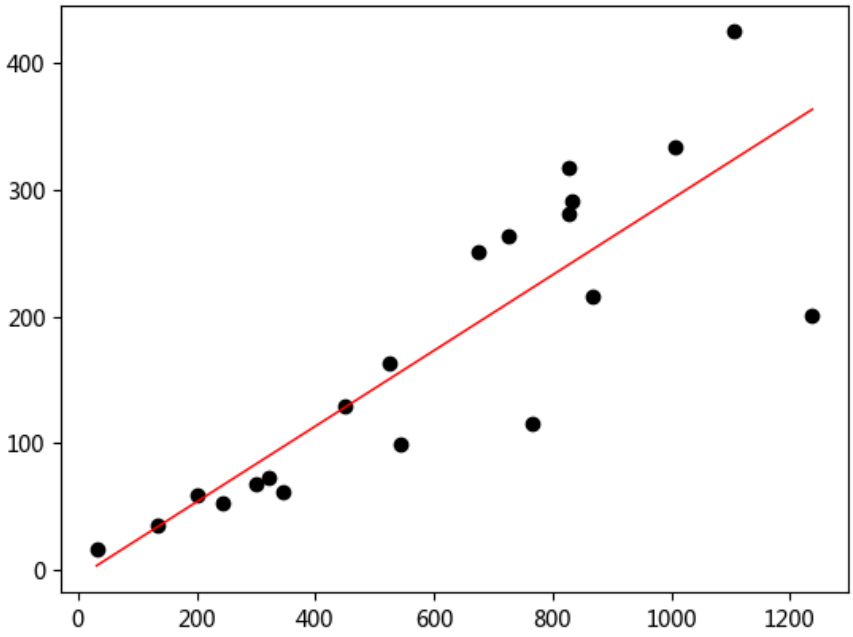
计算结果;

R^2 = 1 - 72279.48 / 267861.07 = 0.73016

python 方法实现:

```
#-*- coding: utf-8 -*-
# python 3.5.0

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

df = pd.read_table('D:/Python35/mypy/test.txt')
x = np.asarray(df[['x']])
y = np.asarray(df[['y']])
reg = LinearRegression().fit(x, y)
print("一元回归方程为:  Y = %.5fX + (%.5f)" % (reg.coef_[0][0], reg.intercept_[0]))
print("R平方为: %s" % reg.score(x, y))

plt.scatter(x, y,  color='black')
plt.plot(x, reg.predict(x), color='red', linewidth=1)
plt.show()
```

```
D:\Python35\mypy>python aa.py
一元回归方程为:  Y = 0.29854X + (-5.95722)
R平方为: 0.730160560258
```



一元多阶线性拟合（多项式拟合）：

假设存在一个函数，只有一个自变量，即只有一个特征属性，满足多项式函数如下：

$$f_M(x,w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

损失函数：损失函数越小，就代表模型拟合的越好。

$$L(w) = \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=0}^{M} w_j x_i^j - y_i\right)^2$$

通过对损失函数偏导为0时，得到最终解方程的函数：

$$\begin{bmatrix} N & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^M \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{M+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \cdots & \sum x_i^{M+2} \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ \sum x_i^M & \sum x_i^{M+1} & \sum x_i^{M+2} & \cdots & \sum x_i^{2M} \end{bmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \\ \vdots \\ \sum x_i^M y_i \end{bmatrix}$$

公式推导参考：

https://www.zhihu.com/question/23483726

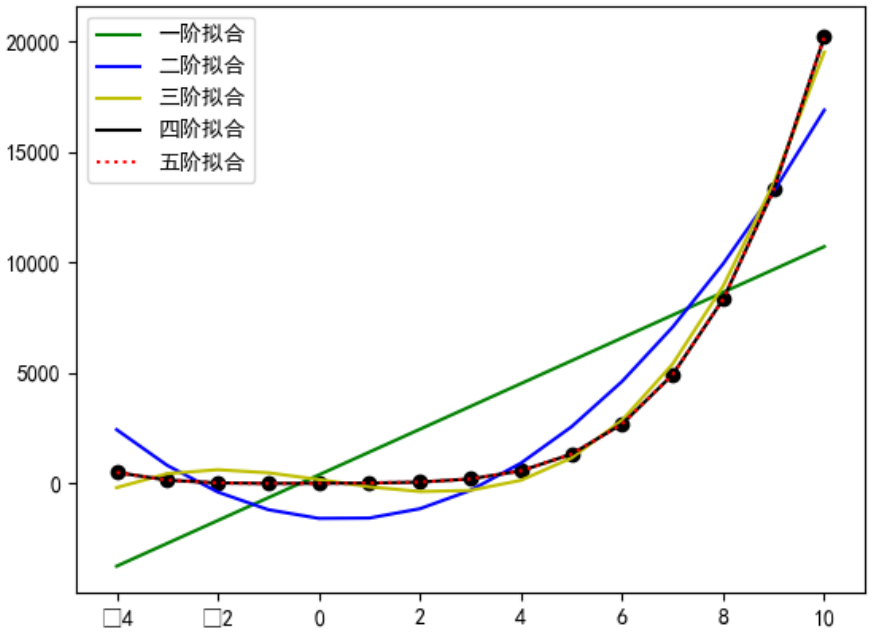
http://blog.csdn.net/xiaolewennofollow/article/details/46757657

https://wenku.baidu.com/view/f20f3e0da8956bec0875e343.html?from=search

python numpy.polyfit 实现：（此处 x 只有一个特征，属于一元多阶函数）

假设因变量 y 刚好符合该公式。

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10])
y = np.array(2*(x**4) + x**2 + 9*x + 2) #假设因变量y刚好符合该公式
#y = np.array([300,500,0,-10,0,20,200,300,1000,800,4000,5000,10000,9000,22000])

# coef 为系数，poly_fit 拟合函数
coef1 = np.polyfit(x,y, 1)
poly_fit1 = np.poly1d(coef1)
plt.plot(x, poly_fit1(x), 'g',label="一阶拟合")
print(poly_fit1)

coef2 = np.polyfit(x,y, 2)
poly_fit2 = np.poly1d(coef2)
plt.plot(x, poly_fit2(x), 'b',label="二阶拟合")
print(poly_fit2)

coef3 = np.polyfit(x,y, 3)
poly_fit3 = np.poly1d(coef3)
plt.plot(x, poly_fit3(x), 'y',label="三阶拟合")
print(poly_fit3)

coef4 = np.polyfit(x,y, 4)
poly_fit4 = np.poly1d(coef4)
plt.plot(x, poly_fit4(x), 'k',label="四阶拟合")
print(poly_fit4)

coef5 = np.polyfit(x,y, 5)
poly_fit5 = np.poly1d(coef5)
plt.plot(x, poly_fit5(x), 'r:',label="五阶拟合")
print(poly_fit5)

plt.scatter(x, y, color='black')
plt.legend(loc=2)
plt.show()
```



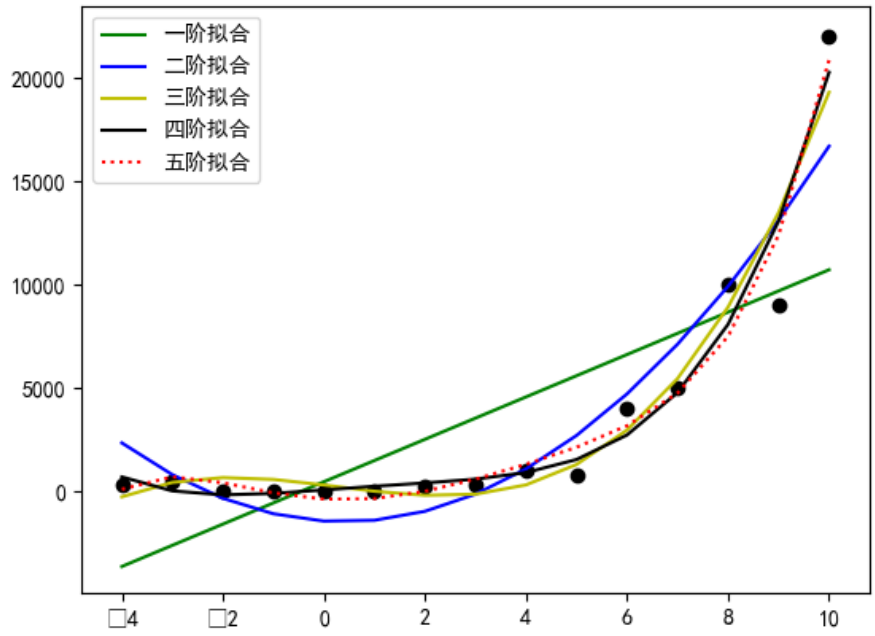
其中5个函数拟合如下：

```
1033 x + 383.8
        2
203.6 x - 188.8 x - 1584
    3          2
24 x - 12.43 x - 342.4 x + 172.7
   4              3      2
2 x + 1.055e-13 x + 1 x + 9 x + 2
             5      4                3      2
-9.575e-17 x + 2 x + 1.292e-15 x + 1 x + 9 x + 2
```

可以看到，只要最高阶为4阶以上，如 四阶拟合 和 五阶拟合，拟合函数近乎完全是符合原函数 y = 2*(x**4) + x**2 + 9*x + 2，拟合是最好的，几乎没有产生震荡，没有过拟合。

当将因变量 y 更换如下：

```
x = np.array([-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10])
y = np.array([300,500,0,-10,0,20,200,300,1000,800,4000,5000,10000,9000,22000])
```




```
1026 x + 462.5
        2
197.2 x - 157.3 x - 1444
      3         2
23.9 x - 17.85 x - 310.2 x + 305.3
        4         3         2
2.808 x - 9.799 x + 0.9996 x + 183.2 x + 65.52
         5          4          3          2
0.6263 x - 6.586 x + 8.642 x + 173.2 x - 147.2 x - 382.4
```

结果发现，四阶及以上拟合程度较高。当设置阶数越高，震荡越明显，也就过度拟合了。怎样确定拟合函数或者最高阶呢？ 参考：Python 确定多项式拟合/回归的阶数

分类 Algorithm , Python - Algorithm

驯龙高手
关注 - 2
粉丝 - 9

0 0

« 上一篇： WOE、VI 分类变量预测能力
» 下一篇： 基础公式