数学建模入门作业

——18373528 杨凌华

Task1

> 用给定的多项式，如$y=x3-6x2+5x-3$，产生一组数据(xi,yi,i=1,2,...,n),再在yi上添加随机干扰(可用rand产生(0,1)均匀分布随机数,或用rands产生N(0,1)分布随机数)，然后用xi和添加了随机干扰的yi作3次多项式拟合，与原系数比较。
>
> 如果作2或4次多项式拟合，结果如何?

```
x = linspace(1,5,20);
p = [1,-6,5,-3];
y = polyval(p,x);
plot(x,y); hold on; % 绘制原始图线
delta = 2 * rand(1,20) - 1; % 添加(-1,1)范围内的扰动
y1 = y + delta;
plot(x,y1); hold on; % 绘制添加扰动后的图线
poly_2 = polyfit(x,y1,2);
disp(poly_2);
poly_3 = polyfit(x,y1,3);
disp(poly_3);
poly_4 = polyfit(x,y1,4);
disp(poly_4);
y2 = polyval(poly_2,x);
y3 = polyval(poly_3,x);
y4 = polyval(poly_4,x);
plot(x,y2); hold on; % 绘制二次拟合图线
plot(x,y3); hold on; % 绘制三次拟合图线
plot(x,y4); hold on; % 绘制四次拟合图线
legend('原始图线','添加扰动','二次拟合','三次拟合','四次拟合');

y1.append(yi + 2 * random.random() - 1)

```

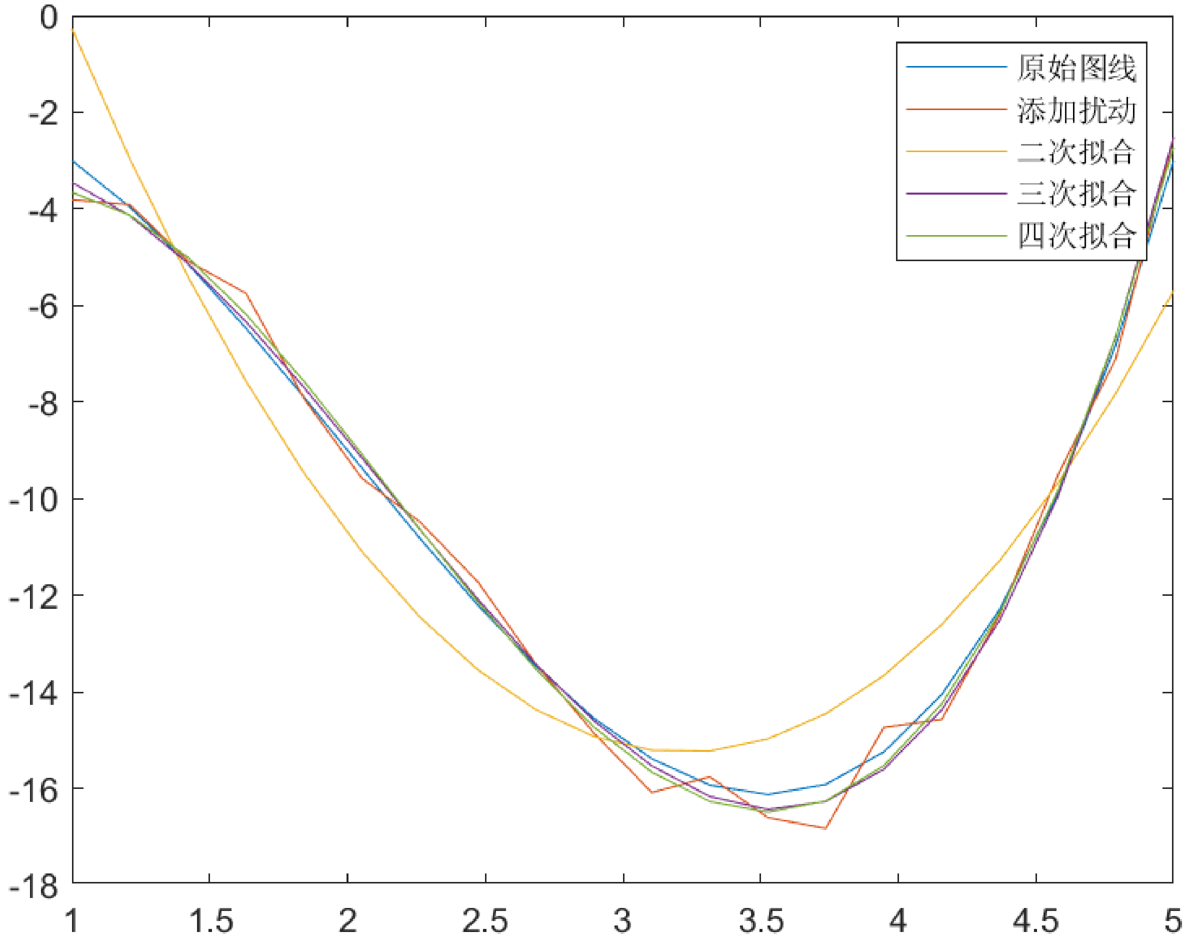
原系数：[1   -6 5 -3]

二次拟合系数：[3.0299 -19.5346 16.2347]

三次拟合系数：[1.1727   -7.5244   9.0279   -6.1271]

四次拟合系数：[-0.0804   2.1374 -11.5634   15.8975 -10.0519]

由拟合后的图像可知，二次拟合效果并不好，而三次、四次拟合的的结果非常接近，并且与原曲线非常吻合，效果较好。



Task2

对原公式

$$v(t) = V - (V - V_0)e^{-\frac{t}{\tau}} = 10 - (10 - V_0)e^{-\frac{t}{\tau}}$$

移项，得：

$$10 - v(t) = (10 - V_0)e^{-\frac{t}{\tau}}$$

对两边取对数，得：

$$ln(10 - v(t)) = -\frac{t}{\tau} + ln(10 - V_0)$$

取：

$$y = ln(10 - v(t))$$
$$a_1 = -\frac{1}{\tau}$$
$$a_2 = ln(10 - V_0)$$

有

$$y = a_1 t + a_2$$
$$\tau = -\frac{1}{a_1}$$
$$V_0 = 10 - e^{a_2}$$

代码如下：

```
t = [0.5 1 2 3 4 5 7 9];
v = [6.36 6.48 7.26 8.22 8.66 8.99 9.43 9.63];
x = t;
y = log(10 - v);
p = polyfit(x,y,1);
tau = -1./p(1);
v0 = 10 - exp(p(2));
disp(tau);
disp(v0);
```

结果:

$$\tau = 3.5269$$
$$v0 = 5.6221$$

Task3

以 x1,x2,x3 分别代表第一、二、三季度生产的台数，则总费用满足:

$$F(x_1,x_2,x_3) = ax_1 + b{x_1}^2 + ax_2 + b{x_2}^2 + ax_3 + b{x_3}^2 + (x_1 - 40) * c + (x_1 + x_2 - 40 - 60) * c$$

最终所要求的非线性规划方程为:

$$\begin{gathered} minF(x_1,x_2,x_3) = b{x_1}^2 + b{x_2}^2 + b{x_3}^2 + (2c+a)x_1 + (c+a)x_2 + ax_3 - 140c \\ 40 \le x_1 \le 100 \\ 100 \le x_1 + x_2 \le 200 \\ 180 \le x_1 + x_2 + x_3 \le 300 \\ 0 \le x_1, x_2, x_3 \le 100 \end{gathered}$$

代码如下:

```
fun = @(x)0.2*x(1)^2+0.2*x(2)^2+0.2*x(3)^2+(2*4+50)*x(1)+
(50+4)*x(2)+50*x(3)-140*4;

x0 = [40,60,80];
A = [1 0 0
     1 1 0
     1 1 1
     -1 0 0
     -1 -1 0
     -1 -1 -1];
b = [100;200;300;-40;-100;-180];
Aeq = [];
beq = [];
lb = [0;0;0];
ub = [100;100;100];
x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub);
disp(x);
disp(fun(x));
```

输出结果:

x = [50.0000   60.0000   70.0000]

最小花费为：11280元

$$F(x_1,x_2,x_3)=a(x_1+x_2+x_3)+b({x_1}^2+{x_2}^2+{x_3}^2)+c(2x_1+x_2-140)$$

因为在满足合同且总费用最低时，x1 + x2 +x3 是总生产量为恒定值180，因此

$$F(x_1,x_2,x_3)=180a+b({x_1}^2+{x_2}^2+{x_3}^2)+c(2x_1+x_2-140)$$

a增大时，会导致总费用增加，但不会改变生产计划。

b增大时，因为b是二次项的系数，因此会使总花费显著增加，而对于 ${x_1}^2+{x_2}^2+{x_3}^2$ 则在x1、x2、x3越接近时越小，因此会使得三季度生产量趋向于平均。

c增大时，因为c是一次项的系数，因此使总花费增加的幅度比b小，相应的计划会调整使第一、二季度的生产有所减少，第三季度的生产有所上升，也就是使得在 x1 + x2 +x3 = 180 不变的情况下，适当减少前两个季度的剩余量。

Task4

> 假设数据挖掘的任务是将8个点聚类成3个簇，A1(2,10),A2(2,5),A3(8,4), A4 (5,8), A5 (7,5), A6 (6,4), A7 (1,2), A8 (4,9),距离函数是欧几里得距离。假设初始选择A1，A4，A7分别作为每个聚类的中心，用k-means算法来给出：
>
> 1.第一次循环执行后的三个聚类中心；
>
> 2.最后的三个簇。

代码如下：

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


# 获取两点之间的欧几里得距离
def getDist(A, B):
    return np.sqrt((A[0] - B[0]) ** 2 + (A[1] - B[1]) ** 2)


# 判断聚类中心是否改变，来决定是否退出迭代
def isCentersUnchanged(old_centers, new_centers):
    for i in range(len(old_centers)):
        if old_centers[i][0] != new_centers[i][0] or old_centers[i][1] != new_centers[i][1]:
            return False
    return True


def myKmeans(k, dataset, start_points):
    # 第一次迭代，获取第一次循环后的聚类中心 first_centers
    centers = start_points
    clusters = []
    for i in range(k): clusters.append([])
    for point in dataset:
        min_dist = getDist(point, centers[0])
        index = 0
        for i in range(1, k):
            dist = getDist(point, centers[i])
```

```python
            if min_dist > dist:
                min_dist = dist
                index = i
        clusters[index].append(point)
    for i in range(k):
        sumx = sumy = 0
        for point in clusters[i]:
            sumx += point[0]
            sumy += point[1]
        centers[i] = [sumx / len(clusters[i]), sumy / len(clusters[i])]
    first_centers = centers.copy()
    # 输出第一次循环后的聚类中心
    print('first_centers:\t' + str(first_centers))

    # 多次循环迭代，直到聚类中心不再改变
    while True:
        old_centers = centers.copy()
        clusters = []
        for i in range(k): clusters.append([])
        for point in dataset:
            min_dist = getDist(point, centers[0])
            index = 0
            for i in range(1, k):
                dist = getDist(point, centers[i])
                if min_dist > dist:
                    min_dist = dist
                    index = i
            clusters[index].append(point)
        for i in range(k):
            sumx = sumy = 0
            for point in clusters[i]:
                sumx += point[0]
                sumy += point[1]
            centers[i] = [sumx / len(clusters[i]), sumy / len(clusters[i])]
        if isCentersUnchanged(old_centers, centers): break

    return centers, clusters


if __name__ == '__main__':
    dataset = [[2, 10], [2, 5], [8, 4], [5, 8], [7, 5], [6, 4], [1, 2], [4, 9]]
    start_points = [[2, 10], [5, 8], [1, 2]]

    k = 3
    final_centers, final_clusters = myKmeans(k, dataset, start_points)
    # 输出最终的聚类中心
    print('final_centers:\t' + str(final_centers))
    # 输出最终的簇
    print('final_clusters:\t' + str(final_clusters))

    # 绘制散点图可视化最终的簇
    for points in final_clusters:
        points = np.array(points)
        plt.scatter(np.transpose(points)[0], np.transpose(points)[1])
    plt.show()
```

第一次循环后得到的三个聚类中心为：

（2.0, 10.0）（6.0, 6.0）（1.5， 3.5）

最后的三个簇分别为：

1) center: (3.67, 9.0)

clusters: (2, 10), (5, 8), (4, 9)

2) center: (7.0, 4.33)

clusters: (8, 4), (7, 5), (6, 4)

3) center: (1.5, 3.5)

clusters: (2, 5), (1, 2)

对应散点图如下：