# MATLAB实例：多元函数拟合(线性与非线性)

作者：凯鲁嘎吉 - 博客园 http://www.cnblogs.com/kailugaji/

更多请看：随笔分类 - MATLAB作图

之前写过一篇博文，是关于一元非线性曲线拟合，自定义曲线函数。

现在用最小二乘法拟合多元函数，实现线性拟合与非线性拟合，其中非线性拟合要求自定义拟合函数。

下面给出三种拟合方式，第一种是多元线性拟合(回归)，第二三种是多元非线性拟合，实际中第二三种方法是一个意思，任选一种即可，推荐第二种拟合方法。

## 1．MATLAB程序

### fit_nonlinear_data.m

```
function [beta, r]=fit_nonlinear_data(X, Y, choose)
% Input: X 自变量数据(N, D)， Y 因变量(N, 1)，choose 1-regress, 2-nlinfit 3-lsqcurvefit
if choose==1
    X1=[ones(length(X(:, 1)), 1), X];
    [beta, bint, r, rint, states]=regress(Y, X1)
    % 多元线性回归
    % y=beta(1)+beta(2)*x1+beta(3)*x2+beta(4)*x3+...
    % beta—系数估计
    % bint—系数估计的上下置信界
    % r—残差
    % rint—诊断异常值的区间
    % states—模型统计信息
    rcoplot(r, rint)
    saveas(gcf,sprintf('线性曲线拟合_残差图.jpg'),'bmp');
elseif choose==2
    beta0=ones(7, 1);
    % 初始值的选取可能会导致结果具有较大的误差。
    [beta, r, J]=nlinfit(X, Y, @myfun, beta0)
    % 非线性回归
    % beta—系数估计
```

```
    % r—残差
    % J—雅可比矩阵
    [Ypred,delta]=nlpredci(@myfun, X, beta, r, 'Jacobian', J)
    % 非线性回归预测置信区间
    % Ypred—预测响应
    % delta—置信区间半角
    plot(X(:, 1), Y, 'k.', X(:, 1), Ypred, 'r');
    saveas(gcf,sprintf('非线性曲线拟合_1.jpg'),'bmp');
elseif choose==3
    beta0=ones(7, 1);
    % 初始值的选取可能会导致结果具有较大的误差。
    [beta,resnorm,r, ~, ~, ~, J]=lsqcurvefit(@myfun,beta0,X,Y)
    % 在最小二乘意义上解决非线性曲线拟合（数据拟合）问题
    % beta—系数估计
    % resnorm—残差的平方范数 sum((fun(x,xdata)-ydata).^2)
    % r—残差 r=fun(x,xdata)-ydata
    % J—雅可比矩阵
    [Ypred,delta]=nlpredci(@myfun, X, beta, r, 'Jacobian', J)
    plot(X(:, 1), Y, 'k.', X(:, 1), Ypred, 'r');
    saveas(gcf,sprintf('非线性曲线拟合_2.jpg'),'bmp');
end
end


function yy=myfun(beta,x) %自定义拟合函数
yy=beta(1)+beta(2)*x(:, 1)+beta(3)*x(:, 2)+beta(4)*x(:, 3)+beta(5)*(x(:, 1).^2)+beta(6)*(x(:, 2).^2)+beta(7)*(x(:, 3).^2);
end
```

**demo.m**

```
clear
clc
X=[1 13 1.5; 1.4 19 3; 1.8 25 1; 2.2 10 2.5;2.6 16 0.5; 3 22 2; 3.4 28 3.5; 3.5 30 3.7];
Y=[0.330; 0.336; 0.294; 0.476; 0.209; 0.451; 0.482; 0.5];
choose=1;
fit_nonlinear_data(X, Y, choose)
```

# 2. 结果

## (1)多元线性拟合(regress)

choose=1:

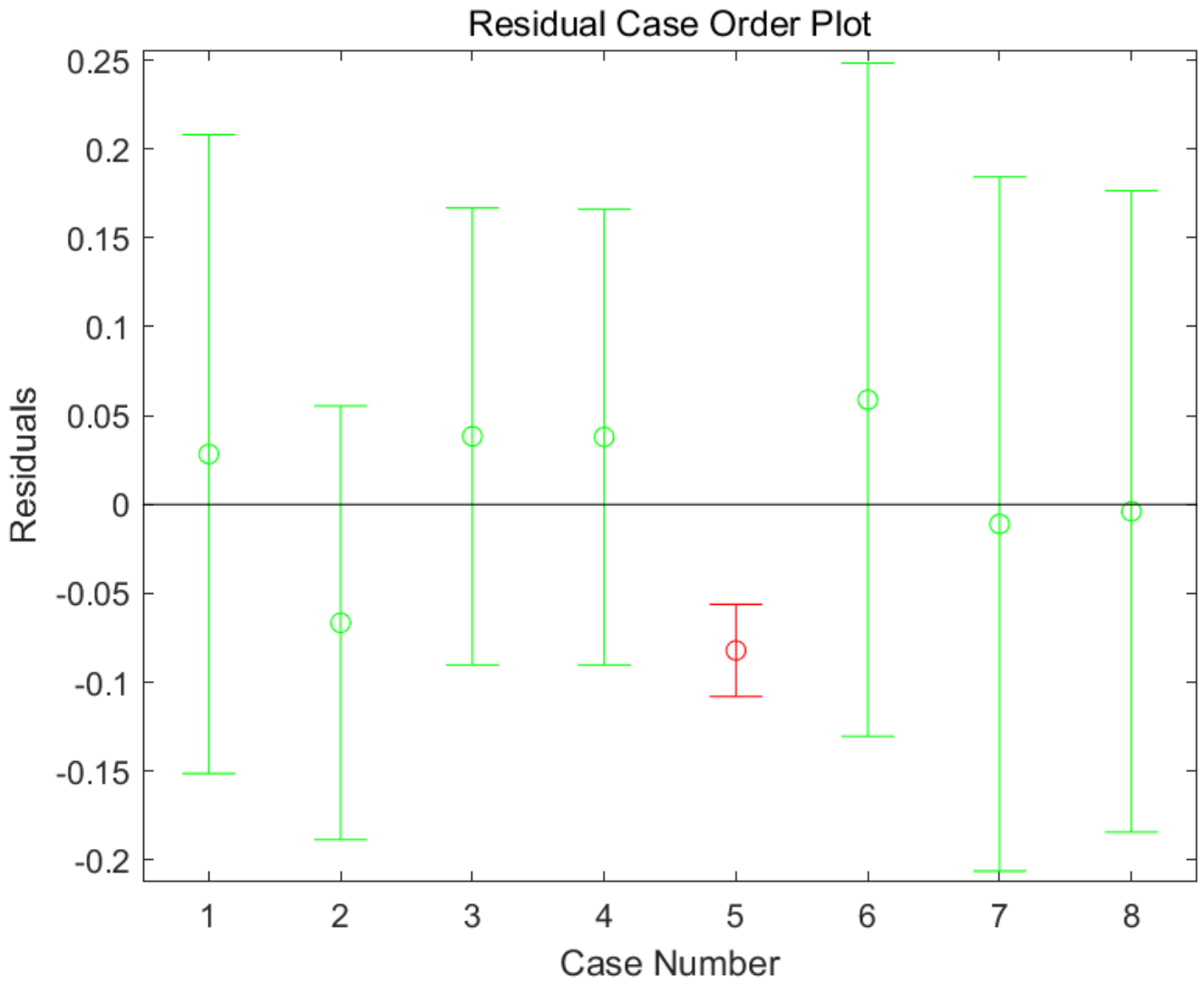
```
>> demo

beta =

   0.200908829282537
   0.044949392540298
  -0.003878606875016
   0.070813489681112


bint =

  -0.026479907290565   0.428297565855639
  -0.057656451966002   0.147555237046598
  -0.017251051845827   0.009493838095795
   0.000201918738160   0.141425060624065


r =

   0.028343433030705
  -0.066584917256987
   0.038333946339215
   0.037954851676187
  -0.082126284727611
   0.058945364984698
  -0.010982985302994
  -0.003883408743214
```

rint =

```
  -0.151352966773048   0.208039832834458
  -0.188622801533810   0.055452967019837
  -0.090283529625345   0.166951422303776
  -0.090266067743345   0.166175771095720
  -0.108068661106325  -0.056183908348897
  -0.130409602930181   0.248300332899576
  -0.206254481234707   0.184288510628719
  -0.184329400080620   0.176562582594191
```

states =

```
   0.768591079367914   4.428472778943478   0.092289917768436   0.004625488283939
```




**(2)多元非线性拟合(nlinfit)**

choose=2:

>> demo

beta =

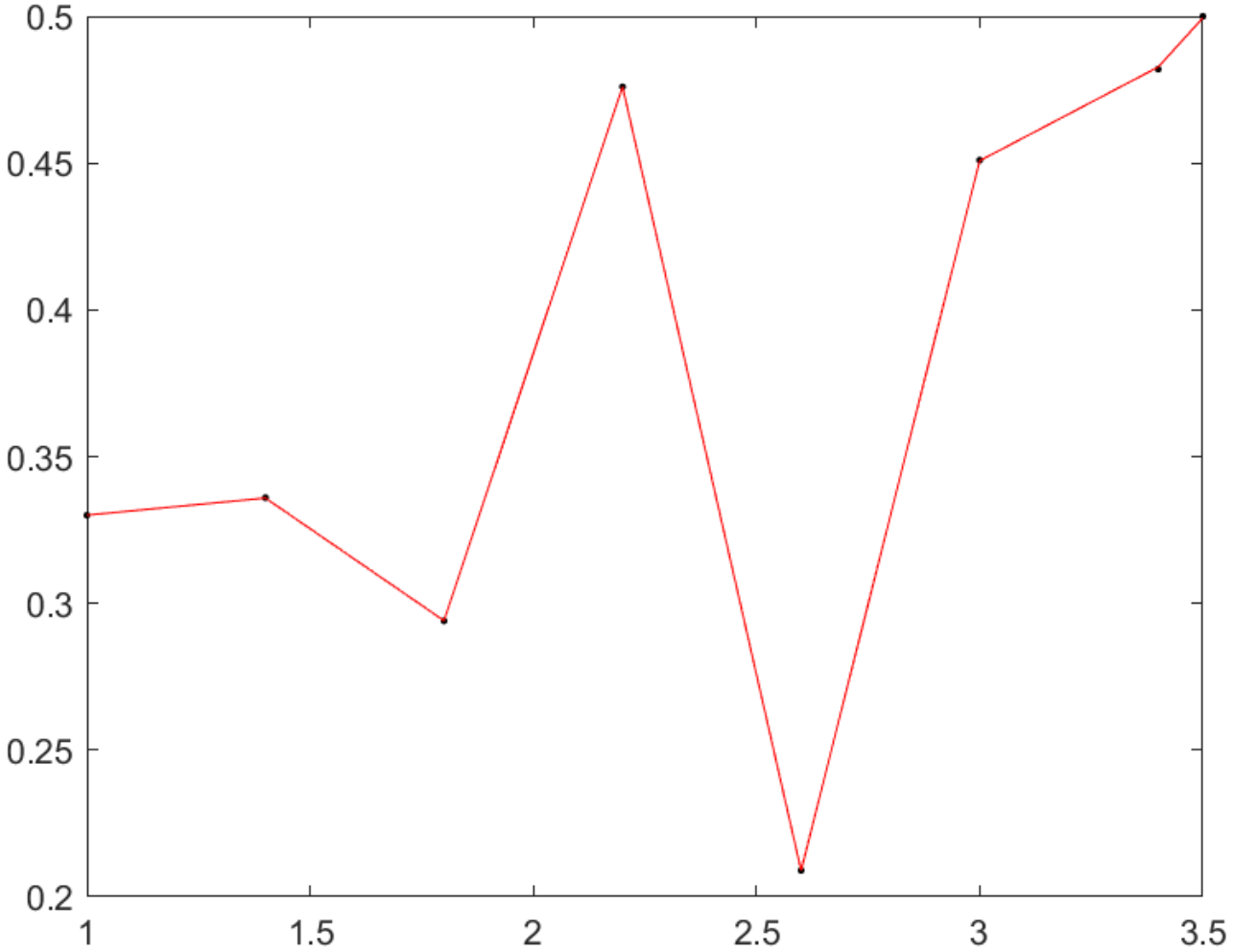
```
   0.312525876099987
   0.015300533415459
  -0.036942272680920
   0.299760796634952
   0.009412595106141
   0.000976411370591
  -0.062931846673372


r =

   1.0e-03 *

  -0.047521336834000
   0.127597019984715
  -0.092883949615763
  -0.040370056416994
   0.031209476614974
   0.211856736183458
  -0.727835090583939
   0.537947200592082


J =

   1.0e+02 *

   0.010000000000266   0.010000000001236   0.129999999998477   0.014999999999641   0.010000000007909   1.689999999969476   0.022499999999756
   0.010000000000266   0.014000000006524   0.189999999999301   0.029999999999283   0.019600000006932   3.609999999769248   0.089999999999024
   0.010000000000266   0.018000000011811   0.249999999990199   0.009999999999965   0.032399999999135   6.250000000033778   0.010000000000377
   0.009999999999679   0.022000000005116   0.099999999998065   0.025000000000218   0.048400000003999   1.000000000103046   0.062500000001264
   0.009999999999972   0.025999999998421   0.159999999998889   0.004999999999982   0.067599999997174   2.559999999999039   0.002499999999730
   0.009999999999679   0.029999999991726   0.219999999999713   0.019999999999930   0.089999999993269   4.839999999890361   0.040000000000052
   0.009999999999092   0.033999999985031   0.279999999990611   0.034999999998348   0.115599999997155   7.839999999636182   0.122500000000614
   0.010000000000266   0.034999999992344   0.299999999994194   0.037000000000420   0.122499999994626   8.999999999988553   0.136899999999292
```

Ypred =

0.330047521336834
0.335872402980015
0.294092883949616
0.476040370056417
0.208968790523385
0.450788143263817
0.482727835090584
0.499462052799408

delta =

0.011997285626178
0.011902559677366
0.011954353934643
0.012001513980794
0.012005923574387
0.011706970437467
0.007666390995581
0.009878186927507




**(3)多元非线性拟合(lsqcurvefit)**

choose=3:

>> demo

beta =

```
   0.312525876070457
   0.015300533464733
  -0.036942272680581
   0.299760796608728
   0.009412595094407
   0.000976411370579
  -0.062931846666179


resnorm =

    8.937848643213721e-07


r =

   1.0e-03 *

   0.047521324135769
  -0.127597015215197
   0.092883952947764
   0.040370060121864
  -0.031209466218374
  -0.211856745335304
   0.727835089662676
  -0.537947200236699


J =

   1.0e+02 *

   (1,1)      0.010000000000000
   (2,1)      0.010000000000000
   (3,1)      0.010000000000000
```

(4,1) 0.010000000000000
(5,1) 0.010000000000000
(6,1) 0.010000000000000
(7,1) 0.010000000000000
(8,1) 0.010000000000000
(1,2) 0.010000000000000
(2,2) 0.014000000059605
(3,2) 0.017999999970198
(4,2) 0.022000000029802
(5,2) 0.026000000014901
(6,2) 0.030000000000000
(7,2) 0.034000000059605
(8,2) 0.035000000000000
(1,3) 0.130000000000000
(2,3) 0.190000000000000
(3,3) 0.250000000000000
(4,3) 0.100000000000000
(5,3) 0.160000000000000
(6,3) 0.220000000000000
(7,3) 0.280000000000000
(8,3) 0.300000000000000
(1,4) 0.015000000000000
(2,4) 0.030000000000000
(3,4) 0.010000000000000
(4,4) 0.025000000000000
(5,4) 0.005000000000000
(6,4) 0.020000000000000
(7,4) 0.035000000000000
(8,4) 0.036999999880791
(1,5) 0.010000000000000
(2,5) 0.019599999934435
(3,5) 0.032399999983609
(4,5) 0.048400000035763
(5,5) 0.067599999997765
(6,5) 0.090000000000000
(7,5) 0.115600000023842

(8,5) 0.122500000000000
(1,6) 1.690000000000000
(2,6) 3.610000000000000
(3,6) 6.250000000000000
(4,6) 1.000000000000000
(5,6) 2.560000000000000
(6,6) 4.840000000000000
(7,6) 7.840000000000000
(8,6) 9.000000000000000
(1,7) 0.022500000000000
(2,7) 0.090000000000000
(3,7) 0.010000000000000
(4,7) 0.062500000000000
(5,7) 0.002500000000000
(6,7) 0.040000000000000
(7,7) 0.122500000000000
(8,7) 0.136899999976158

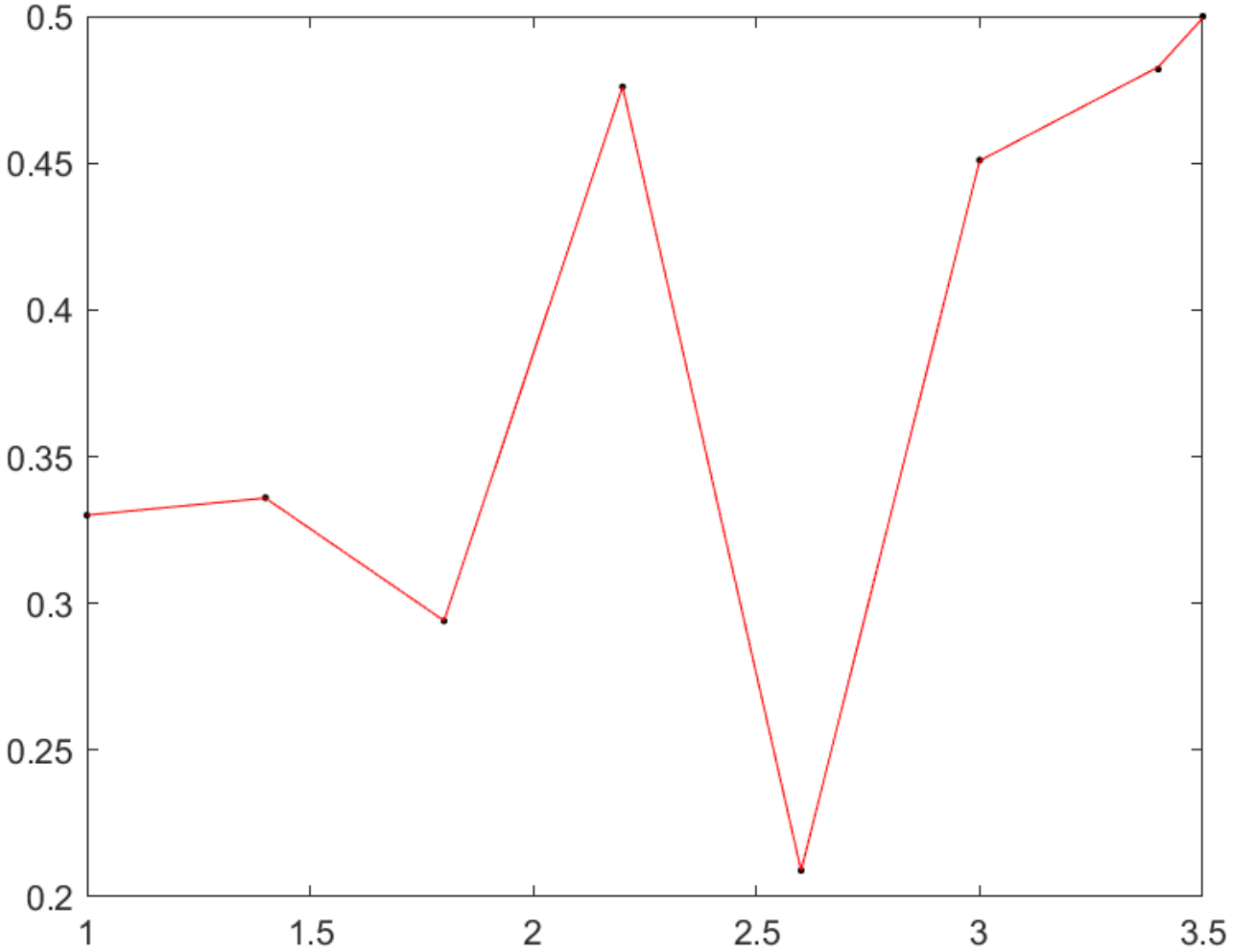
Ypred =

0.330047521324136
0.335872402984785
0.294092883952948
0.476040370060122
0.208968790533782
0.450788143254665
0.482727835089663
0.499462052799763

delta =

0.011997285618724
0.011902559623756
0.011954353977139

0.012001513949620
0.012005923574975
0.011706970418735
0.007666391016173
0.009878186931566



# 注意

1）多元非线性函数拟合中参数的初始值需要提前设置，有些情况下，参数的初始选取对函数拟合结果影响极大，需要谨慎处理。

2）第二三种方法中，由于数据是多维的，因此只展示了第一个维度的拟合函数图。如有需要，可自行修改。

3）自定义拟合函数要看清楚数据X的维度，我这里是三维的，因此有x(:, 3)，如果是D维，要写到x(:, D)。同时，参数beta的尺寸也要相应更新。

4）数据归一化方法自行选择，可能有些数据集不适合最大-最小归一化。

5）很多时候拟合函数很难构造，线性拟合效果又不理想，在这种情况下，可以尝试使用神经网络，深度学习，支持向量机等工具进行拟合非线性函数。这里是BP神经网络来进行拟合(回归)的一个例子。MATLAB实例：BP神经网络用于回归任务 - 凯鲁嘎吉 - 博客园