PRACTICE 2 - OPTIMIZATION TECHNIQUES

Andrés Herencia y Antonio Fernández

MUTECI 2023-2024

Table of Contents

Exercise 1

Given the data from the text file titled 'isomerizacion.txt,' determine the parameters that fit the model:

$$y = \frac{\theta_1 \theta_2 (x_2 - x_3)/1.632}{1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_3}$$

Request:

a. Linearize the model and obtain an initial estimate.

b. With the previous initial estimate, solve the non-linear regression problem by minimizing the squared error.

Solution

```
clc
[x1,x2,x3,y]=textread('isomerizacion.txt','%f %f %f %f','headerlines',8);
```

a) Linearize the model and obtain an initial estimate.

With $y' = \frac{1}{y}(x_2 - x_3)$ we can operate to achieve this expression:
$y' = (x_2 - x_3)\frac{1+\theta_2 x_1+\theta_3 x_2+\theta_4 x_3}{\theta_1\theta_2(x_2-x_3)/1.632} = \frac{1.632}{\theta_1\theta_2}(1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_3) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$.

Where: $\theta_1 = \frac{1.632}{\beta_1}$, $\theta_2 = \frac{\beta_1}{\beta_0}$, $\theta_3 = \frac{\beta_2}{\beta_0}$, $\theta_4 = \frac{\beta_3}{\beta_0}$.

```
X1 = [ones(length(x1),1),x1,x2,x3];
Y1 = [(x2-x3)./y];
beta = inv(X1.'*X1)*X1.'*Y1
```

```
beta = 4x1
   -5.3459
    0.0396
    0.1798
   -0.3108
```

```
O(1)=1.632/beta(2);
O(2)=beta(2)/beta(1);
O(3)=beta(3)/beta(1);
O(4)=beta(4)/beta(1);
display(O)
```

```
O = 1x4
   41.1949   -0.0074   -0.0336    0.0581
```

b) With the previous initial estimate, solve the non-linear regression problem by minimizing the squared error.

```
f = @(x1,x2,x3,y,E) sum((y - (((x2 - x3)./1.632).*(E(1).*E(2)))./(1 + E(2).*x1 + E(3).*x2 +
myfunc = @(E) f(x1,x2,x3,y,E);
[beta_opt, fval, exitflag, output] = fminsearch(myfunc,O)
```

```
beta_opt = 1x4
    4.7969   -0.0071   -0.0059    0.0200

fval = 210.4719
exitflag = 1
output = 
    iterations: 241
```

```
      funcCount: 409
      algorithm: 'Nelder-Mead simplex direct search'
        message: 'Optimization terminated:↩ the current x satisfies the termination criteria
```

We can check that since the problem is non-linear, is very sensitive to the initial conditions and the optimization method, so, we can not interpretate the solution with ease.

```
[beta_opt2, fval2, exitflag2, output2] = fminunc(myfunc,0)
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
beta_opt2 = 1x4
   40.1275   -9.7344  -11.7309   -6.8991

fval2 = 16.1631
exitflag2 = 1
output2 =
       iterations: 66
        funcCount: 350
         stepsize: 1.5932
     lssteplength: 1
    firstorderopt: 0.5222
        algorithm: 'quasi-newton'
          message: 'Local minimum found.↩↩Optimization completed because the size of the g
```

If we want to minimize the quadratic sum of errors, we could use this method, because the fval parameter is the lowest.

```
[beta_opt3, fval3, exitflag3, output3] = fminsearch(myfunc,[0,0,0,0])
```

```
beta_opt3 = 1x4
    0.0056    0.0049   -0.0043   -0.0092

fval3 = 497.8179
exitflag3 = 1
output3 =
    iterations: 171
     funcCount: 310
     algorithm: 'Nelder-Mead simplex direct search'
       message: 'Optimization terminated:↩ the current x satisfies the termination criteria
```

```
[beta_opt4, fval4, exitflag4, output4] = fminunc(myfunc,[0,0,0,0])
```

```
Initial point is a local minimum.

Optimization completed because the size of the gradient at the initial point
is less than the value of the optimality tolerance.

<stopping criteria details>
beta_opt4 = 1x4
     0     0     0     0

fval4 = 640.4885
exitflag4 = 1
output4 =
       iterations: 0
        funcCount: 5
         stepsize: []
     lssteplength: []
    firstorderopt: 0
        algorithm: 'quasi-newton'
          message: 'Initial point is a local minimum.↩↩Optimization completed because the
```

Exercise 2

Given the data in the following table:

```
clc
x = [4.5, 5.0, 5.1, 5.3, 6.2, 7.1]'
```

```
x = 6x1
    4.5000
    5.0000
    5.1000
    5.3000
    6.2000
    7.1000
```

```
y = [5.0, 3.8, 4.9, 3.7, 3.6, 15.0]'
```

```
y = 6x1
    5.0000
    3.8000
    4.9000
    3.7000
    3.6000
   15.0000
```

Determine the regression line $Y = \alpha + \beta X$ when the criterion to be minimezed is:

a. The sum of the squared error, being$\epsilon_i = y_i - \alpha - \beta x_i$ the error of the i-th observation `i=1,2,...,6`.

b. The sum of the absolute value of the errors.

c. The maximum absolute value of the errors.

Solve the three problems with Matlab and represent the three regression lines for illustrate the differences.

Solution

Simple Regression Line and Original Data

```
X = [ones(length(x),1),x]
```

```
X = 6x2
    1.0000    4.5000
    1.0000    5.0000
    1.0000    5.1000
    1.0000    5.3000
    1.0000    6.2000
    1.0000    7.1000
```

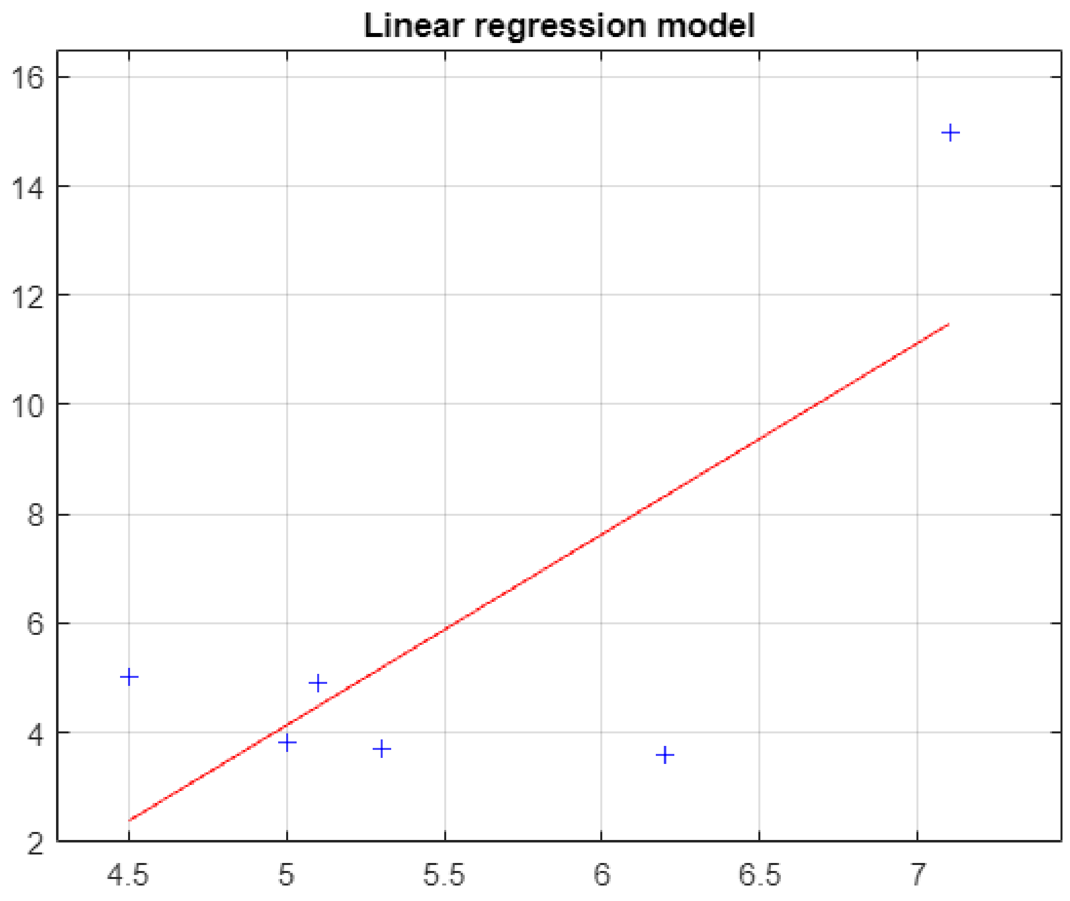
```
b = regress(y,X)
```

```
b = 2x1
  -13.3585
    3.4985
```

```
Y = b(1) + b(2)*x
```

```
Y = 6x1
    2.3849
    4.1341
    4.4840
    5.1837
    8.3323
   11.4810
```

```
plot(x,y,'b+',x,Y,'r-')
xlim([0.95*min(x),1.05*max(x)])
ylim([2,max(y)*1.1])
title('Linear regression model')
grid on
```



a) The sum of the squared error

Firstly, we define an anonymous function with the unconstrained optimization problem we are trying to solve. We set the parameters that are not objects of the problem by overriding them in a new function. Finally, we solve the optimization problem by defining a pair of initial values as a starting point.

```
f = @(x,y,beta) sum((y - beta(1) - beta(2).*x).^2)
```

```
f =
    @(x,y,beta)sum((y-beta(1)-beta(2).*x).^2)
```

```
myfunc = @(beta) f(x,y,beta)
```

```
myfunc =
    @(beta)f(x,y,beta)
```

It can be resolved with two different methods natively in MATLAB: `fminunc` and `fminsearch`. We will get the same result.

```
fminunc(myfunc,[0,0]) % using a gradient boot search method
```

```
Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.

<stopping criteria details>
ans = 1x2
  -13.3585    3.4985
```

```
bopt1 = fminsearch(myfunc,[0,0]) % using derivative-free method
```

```
bopt1 = 1x2
  -13.3585    3.4985
```

We have obtained the following values:

$$\alpha = -13.3585$$
$$\beta = 3.4985$$

b) The sum of the absolute value of the errors.

```
f2 = @(x,y,beta) sum(abs((y - beta(1) - beta(2).*x)))
```

```
f2 =
    @(x,y,beta)sum(abs((y-beta(1)-beta(2).*x)))
```

```
myfunc2 = @(beta) f2(x,y,beta)
```

```
myfunc2 =
    @(beta)f2(x,y,beta)
```

```
bopt2 = fminsearch(myfunc2,[0,0])
```

```
bopt2 = 1x2
  -22.8665    5.3333
```

With this method, we have obtained:

$$\alpha = -22.8665$$
$$\beta = 5.3333$$

Using the linear programming method

See in the appendix the general structure of a LP problem.

The optimization problem can be defined as

$$\min_{\alpha,\beta} \sum_{i}^{n} |\epsilon_i|$$

Being

$$\epsilon_i = y_i - \hat{y}_i = y_i - \alpha - \beta x_i$$

Since we want to formulate it as a linear programming problem, we have to make a change of variable to leave it in linear terms. This implies that we have to define two new constraints:

$$u_i \leq \ |\epsilon_i| = \begin{cases} u_i = (y_i - \hat{y}_i) = y_i - \alpha - \beta x_i \leq \epsilon_i & i = 1, \ldots, n \\ u_i = -\epsilon_i = -(y_i - \hat{y}_i) = -(y_i - \alpha - \beta x_i) \geq -\epsilon_i & i = 1, \ldots, n \end{cases}$$

With the condition that the absolute value is always positive, our LP problem is:

$$\begin{array}{l} \min_{u_1,\ldots,u_n,\alpha,\beta} \sum_i^n u_i \\ \text{s. to} \\ i) - u_i - \alpha - \beta x_i \leq -y_i, \quad \forall i = 1, ..., n \\ ii) - u_i + \alpha + \beta x_i \leq y_i, \quad \forall i = 1, ..., n \\ iii) \ u_i \geq 0 \end{array}$$

In our concrete case, $n = 6$.

Therefore, we have 8 decision variables $(u_1, u_2, u_3, u_4, u_5, u_6, \alpha, \beta)$, 12 inequations, and an inferior bound for each u_i.

```
n = length(x)
```

```
n = 6
```

```
u = (-1)*diag(ones(n,1));
Aub = zeros(12,8);
for i = 1:n
    Aub(i,:) = [u(i,:),-1,-x(i)];
    Aub(6+i,:) = [u(i,:),1,x(i)];
end
Aub
```

```
Aub = 12x8
   -1.0000         0         0         0         0         0   -1.0000   -4.5000
         0   -1.0000         0         0         0         0   -1.0000   -5.0000
         0         0   -1.0000         0         0         0   -1.0000   -5.1000
         0         0         0   -1.0000         0         0   -1.0000   -5.3000
         0         0         0         0   -1.0000         0   -1.0000   -6.2000
         0         0         0         0         0   -1.0000   -1.0000   -7.1000
   -1.0000         0         0         0         0         0    1.0000    4.5000
         0   -1.0000         0         0         0         0    1.0000    5.0000
         0         0   -1.0000         0         0         0    1.0000    5.1000
         0         0         0   -1.0000         0         0    1.0000    5.3000
```

```
b = [-y;y]
```

```
b = 12x1
   -5.0000
   -3.8000
   -4.9000
   -3.7000
   -3.6000
  -15.0000
    5.0000
    3.8000
    4.9000
    3.7000
```

```
c = [ones(1,n),[0,0]]
```

```
c = 1x8
     1     1     1     1     1     1     0     0
```

```
lin_pr = linprog(c,Aub,b);
```

```
Optimal solution found.
```

```
bopt2lin = lin_pr(end-1:end)
```

```
bopt2lin = 2x1
  -22.8667
    5.3333
```

Solving as an LP problem, we have obtained the following values:

$$\alpha = -22.8667$$
$$\beta = 5.3333$$

c) The maximum of the absolute values of the errors.

```
f3 = @(x,y,beta) max(abs((y - beta(1) - beta(2).*x)))
```

```
f3 =
    @(x,y,beta)max(abs((y-beta(1)-beta(2).*x)))
```

```
myfunc3 = @(beta) f3(x,y,beta)
```

```
myfunc3 =
    @(beta)f3(x,y,beta)
```

```
bopt3 = fminsearch(myfunc3,[0,0])
```

```
bopt3 = 1x2
  -16.2769    3.8462
```

Obtaining:

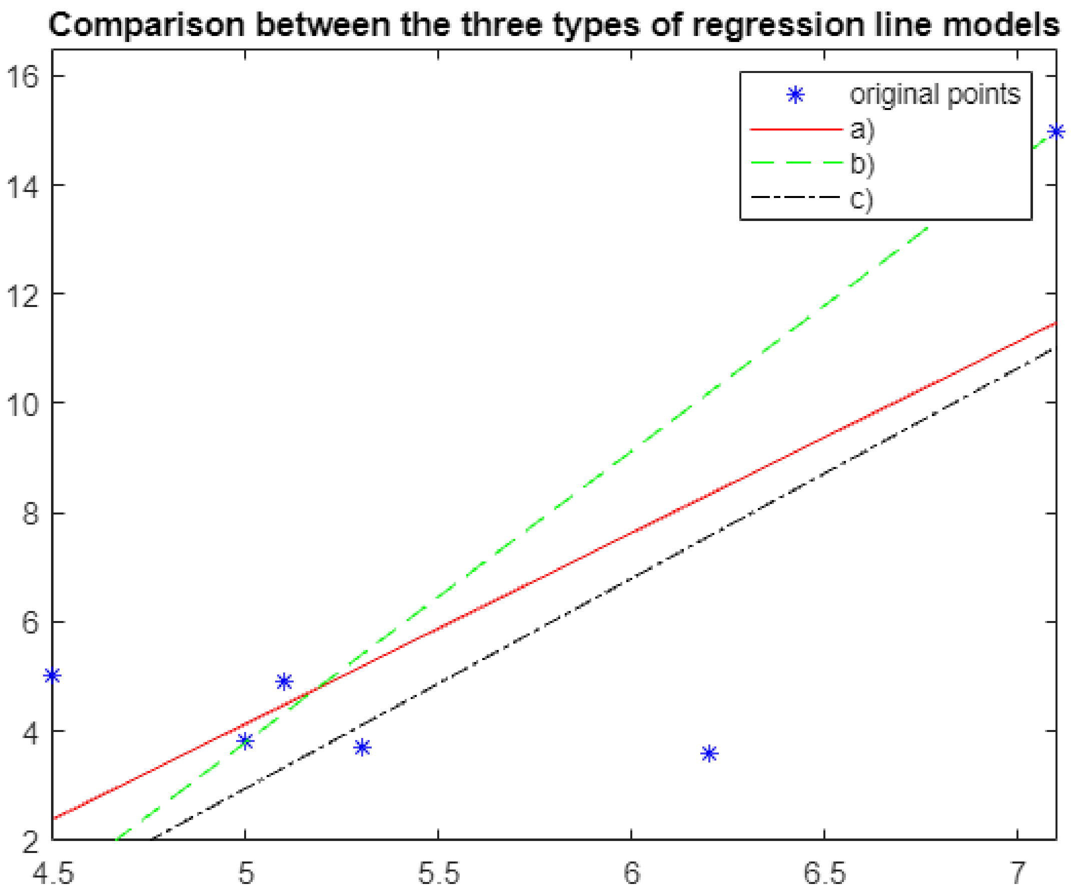
$$\alpha = -16.2769$$
$$\beta = 3.8462$$

Plotting the solutions

```
reg1 = bopt1(1) + bopt1(2)*x;
reg2 = bopt2(1) + bopt2(2)*x;
reg3 = bopt3(1) + bopt3(2)*x;
```

$$\begin{cases} y_1 = -13.36 + 3.50x \\ y_2 = -22.87 + 5.33x \\ y_3 = -16.28 + 3.85x \end{cases}$$

```
plot(x,y,'b*',x,reg1,'-r',x,reg2,'--g',x,reg3,'-.k')
legend('original points','a)','b)','c)')
xlim([min(x),max(x)])
ylim([2,max(y)*1.1])
title('Comparison between the three types of regression line models')
```



We can observe that the regression lines are strongly influenced by a point that could be considered an outlier. Depending on the optimization method, the influence of this point is lower or higher. The other points are much nearer; if we had analyzed only those points, the regression line would have been a less pronounced slope.