Ajuste de curvas

Télico Oliveira

20 de agosto de 2023

1 Resumo

A partir de um problema disponível no livro Cálculo numérico e aplicações, vamos discorrer sobre ajustes de curvas e implementar um algoritmo que calcula a reta que mais se aproxima de um conjunto de pontos dados utilizando o método dos mínimos quadrados.
Problema Construir um diagrama de dispersão das medidas das variáveis x e y.

$$x_i = \{1.3, 3.4, 5.1, 6.8, 8.0\}$$
$$y_i = \{2.0, 5.2, 3.8, 6.1, 5.8\}$$

$$d_i = y_i - \hat{y_i} \tag{1}$$

$$D = \sum_{i=1}^{n} d_i^2 \tag{2}$$

A variável D calcula os desvios das ordenadas de cada ponto em relação a uma determinada reta e faz o somatório dos quadrados desses desvios.

O problema consiste em encontrar os coeficientes b_0 e b_1 dessa reta.

$$\hat{y_i} = b_0 + b_1 x_i \tag{3}$$

Considere que D é uma função dos coeficientes b_0 e b_1 $D(b_0, b_1)$, ou seja

$$D(b_0, b_1) = \sum_{i=1}^{n} d_i^2$$

$$D(b_0, b_1) = \sum_{i=1}^{n} [y_i - (b_0 + b_1 x_i)]^2$$

2 Escolha da melhor reta

A reta que mais se aproxima do conjunto de pontos dados será aquela que possuir o menor desvio . para encontrarmos esta reta, precisamos calcular as derivadas parciais de $D(b_0, b_1)$ e igualá-las a zero.

$$D(b_0, b_1) = \sum_{i=1}^{n} (y_i - b_0 + b_1 x)^2 \tag{4}$$

$$\frac{\partial D}{\partial b_0} = -2 \sum_{i=1}^{n} (y_i b_0 - b_1 x_i) \tag{5}$$

$$\frac{\partial D}{\partial b_1} = -2 \sum_{i=1}^{n} (y_i b_0 - b_1 x_1) x_i \tag{6}$$

Igualando as derivadas parciais a zero, vem:

$$-2 \sum_{i=1}^{n} (y_i b_0 - b_1 x_i) = 0$$

$$-2\sum_{i=1}^{n}(y_i b_0 - b_1 x_i) = 0$$

$$\begin{cases} \sum y_i - \sum b_0 - \sum b_1 x_i = 0 \\ \sum x_i y_i - \sum b_0 x_i - \sum b_1 x_i^2 = 0 \end{cases} \tag{7}$$

$$\begin{cases} (n) b_0 + (\sum x_i) b_1 = \sum y_i \\ (\sum x_i) b_0 + (\sum x_i^2) b_1 = \sum x_i y_i \end{cases} \tag{8}$$

Resolvendo o sistema para b_0 e b_1 por qualquer um dos métodos, obtemos como resultado

$$b_1 = \frac{n \cdot \sum x_i y_i - \sum x_i \sum y_i \sum y_i}{n \cdot \sum x_i^2 - (\sum x_i)^2} \tag{9}$$

$$b_0 = \frac{\sum y_i - (\sum x_i) b_1}{n} \tag{10}$$

3 Implementação do ajuste da melhor curva em C

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"

int main(){
 double X[5] ={1.3, 3.4, 5.1, 6.8, 8.0} ;
 double Y[5] = {2.0, 5.2, 3.8, 6.1, 5.8};
 double reta1[5];
 double reta2[5];
 double d1[5];
 double d2[5];
 double D1 = 0;
 double D2 = 0;
 double sx, sy, sxy, sx2, b0, b1;
int i;
int n = 5;
        printf("\ni  X[i], Y[i]");
 for (i = 0; i < n; i ++ ){
        reta1[i] = 0 + 1* X[i];
        d1[i] = Y[i]- reta1[i];
        reta2[i] = 4.5 + 0*X[i];
        d2[i] = Y[i]- reta2[i];


        printf("\n %i,  %.1f, %.1f",i + 1,  X[i], Y[i]);
        printf("\n");

 }
 for (i = 0; i < n; i ++ ){
        D1 += pow(d1[i], 2);
        D2 += pow(d2[i], 2);
//      printf("\n%.2f, %2.f\n", D1, D2);
 }
 //Implementa os somatorios
 for (i = 0; i < n; i++){
```

```
            sx += X[i];
            sy +=Y[i];
            sxy +=X[i]*Y[i];
            sx2 +=pow(X[i], 2);
           //printf("sx = %.1f  sy = %.1f  sxy = %.1f sx2 = %.1f\n", sx, sy, sxy, sx2);
 }
//Calcula os coeficientes da reta
b1 = (n* sxy - sx*sy)/(n*sx2- pow(sx, 2));
b0 = (sy - sx*b1)/n;
printf("\nY =  %.1f +  %.1fX\n ", b0, b1);
return 0;

}
```