Heaven's Light is Our Guide

RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

LAB REPORT

Course Code	: CSE 2204
Course Name	: Numerical Methods Sessional
Experiment No.	: 04
Experiment Name	: Least Squares Curve fiitting a Straight line and power function

Submitted by:	**Submitted to:**
Name : MD Shehab Sarker Roll : 2103046 Section : A Session : 2021-2022	Shyla Afroge Associate Professor Computer Science & Engineering RUET

Date Of Submission : 30.11.2024

Problem Define:

Task1: Find the best values for a0 and a1 in Least-Squares Curve Fitting a Straight line

Theroy:

Usually a mathematical equation is fitted to experimental data by plotting the data on a graph paper and then passing a straight line through the data points. The method has the obvious drawback in that the straight line drawn may not be unique. The method of least squares is probably the most systematic procedure to fit a unique curve through given data points and is widely used in practical computations. It can also be easily implemented on a digital computer

Let the set of data points be (x, y), i = 1, 2, ..., m, and let the curve given by Y= f(x) be fitted to this data. At x = x_i, the experimental (or observed) value of the ordinate is y, and the corresponding value on the fitting curve is f(x_i). If e, is the error of approximation at x = x_i, then we have

$$e_i = y - f(x_i).$$

If we write

$$S = [y_1 - f(x_1)]^2 + [y_2 - f(x_2)]^2 + \cdots + [y_m - f(x_m)]^2$$
$$= e_1^2 + e_2^2 + \cdots + e_m^2.$$

Let Y=a_0 +a_1x be the straight line to be fitted to the given data.so below the progamming code by above equation any problem solve for a0 and a1 values

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define lli long long int

int main(){
   double x[20],y[20];
   double sumX=0,sumX2=0,sumY=0,sumXY=0,a,b;

   int n;
   cout<<"Enter number of data:";
   cin>>n;

   cout<<"Enter Data: "<<endl;
   cout<<"x"<<"  "<<"y"<<endl;;
   for(int i=0;i<n;i++){
      cin>>x[i]>>y[i];
   }
```

```
    for(int i=0;i<n;i++){
        sumX+=x[i];
        sumX2+=x[i]*x[i];
        sumY+=y[i];
        sumXY+=x[i]*y[i];

    }

    b=(n*sumXY-sumX*sumY)/(n*sumX2-sumX*sumX);
    a=(sumY-b*sumX)/n;

    cout<<"a0 = "<<a<<" and a1 = "<<b<<endl;
    cout<<"curve fitting is: y = "<<a<<" + "<<b<<"X";
    return 0;
}
```

Result:

```
Enter number of data:6
Enter Data:
x  y
20 800.3
30 800.4
40 800.6
50 800.7
60 800.9
70 801.9
a0 = 799.566 and a1 = 0.0274286
curve fitting is: y = 799.566 + 0.0274286X
```

Conclusion:

The above C++ program performs linear regression using the method of least squares. It fits a straight line of the form $y=a+bX$, where a (intercept) and b (slope) are determined based on the given data points. The program calculates a and b using formulas derived from minimizing the squared differences between the observed y-values and the predicted y-values on the line.

Task 2: Find the constant a and b by the method of Non linear least squares Curve Fitting

Theory:

In this Task, we consider a power function, a polynomial of the nth degree and an exponential function to fit the given data points

$$(x_i, y_i), i=1,..., m$$

Power function Let $y=ax^c$ be the function to be fitted to the given data. Taking logarithms of both sides, we obtain the relation

$$\log y = \log a + c \log x,$$

which is of the form $Y=a_0 + a_1X$, where $Y=\log y$, $a_0= \log a$, $a_1 = c$ and $X = \log x$. Hence the procedure outlined in the previous task can be followed to evaluate ao and a_1. Then a and c can be calculated from the formulae $a_0 = \log a$ and $c = a_1$.

Code:

```
#include<bits/stdc++.h>
using namespace std;
#define lli long long int

int main(){
    double x[20],y[20];
    double sumX=0,sumX2=0,sumY=0,sumXY=0,a,b,A;

    int n;
    cout<<"Enter number of data:";
    cin>>n;

    cout<<"Enter Data: "<<endl;
    cout<<"x"<<"  "<<"y"<<endl;;
    for(int i=0;i<n;i++){
        cin>>x[i]>>y[i];
    }

    for(int i=0;i<n;i++){
        sumX+=log(x[i]);
        sumX2+=log(x[i])*log(x[i]);
        sumY+=log(y[i]);
        sumXY+=log(x[i])*log(y[i]);

    }

    b=(n*sumXY-sumX*sumY)/(n*sumX2-sumX*sumX);
    A=(sumY-b*sumX)/n;
    a=exp(A);
    cout<<"a0 = "<<a<<" and a1 = "<<b<<endl;
    cout<<"Y="<<a<<" *x^"<<b<<endl;
    return 0;
}
```

Result:

```
Enter number of data:5
Enter Data:
x  y
1 2
2 4.1
3 6.3
4 8.5
5 10.9
a0 = 1.9903 and a1 = 1.05109
Y=1.9903 *x^1.05109
```

Conclusion:

The given C++ program performs power regression by fitting a curve of the form $Y=a*x^b$ to a given dataset. It uses a logarithmic transformation to linearize the equation and apply the least-squares method to find a (scaling factor) and b (power/exponent).