科学計算研究室 Python ゼミ

～ 6. 最小二乗法[1] ～

2021-03-17 福田　浩

1　原理

1.1　最小二乗法とは

測定で得られた数値の組を、適当なモデルから想定される 1 次関数、対数曲線など特定の関数を用いて近似するときに、想定する関数が測定値に対してよい近似となるように、残差の二乗和を最小とするような係数を決定する方法、あるいはそのような方法によって近似を行うことである。(Wikipedia)

　要はこう言うこと．実験で以下のデータが得られたとする．上司が「ほとんど直線関係なんだよなぁ．1 次式で表現してくれない？」と御無体なことを言う．テキトーに答えると説教が始まるのが目に見えるので，計算を駆使して，根拠ある値を報告しなければならない．最小二乗法はこんな場合に使う方法．

Table 1: 実験で得られたデータ

x	y
1	2
2	5
3	7

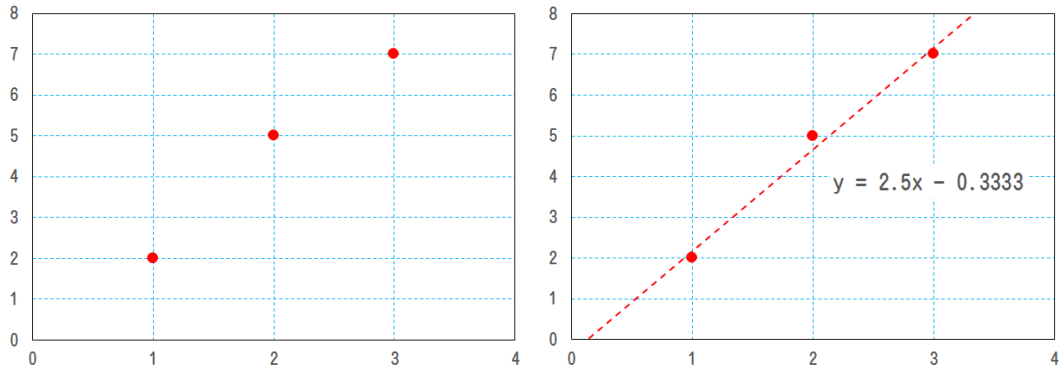

Figure 1: 実験で得られたデータ (左) とその最小二乗近似 (右)

[1] 最小自乗法とも書く

1.2 考え方

3 点 $(1,2),(2,5),(3,7)$ を近似する直線は，x の 1 次式で表すことが出来る．その式を以下のように表しても一般性を失わない．

$$y = ax + b \tag{1}$$

この時の誤差の総和 E は，

$$E = \{2-(a+b)\} + \{5-(2a+b)\} + \{7-(3a+b)\} \tag{2}$$

であるが，符号が逆になる場合はキャンセルして誤差が小さく見えてしまうので，それぞれの項を 2 乗して足す方が良い．

$$\begin{aligned} E &= \{2-(a+b)\}^2 + \{5-(2a+b)\}^2 + \{7-(3a+b)\}^2 \\ &= (2-a-b)^2 + (5-2a-b)^2 + (7-3a-b)^2 \\ &= (4+a^2+b^2-4a+2ab-4b) \\ &\quad +(25+4a^2+b^2-20a+4ab-10b) \\ &\quad +(49+9a^2+b^2-42a+6ab-14b) \\ &= 78+14a^2+3b^2-66a+12ab-28b \end{aligned} \tag{3}$$

である． 式 3 は，a,b の 2 次式であり，これが最小になるときは，a,b の偏微分係数がゼロになる時である．以上のことから，連立方程式

$$\begin{aligned} \frac{\partial E}{\partial a} &= \cancel{78} + 2\times 14a^{\cancel{1}} + \cancel{3b^2} - 66\cancel{a} + 12\cancel{a}b - \cancel{28b} \\ &= 28a - 66 + 12b \\ &= 0 \end{aligned} \tag{4}$$

$$\begin{aligned} \frac{\partial E}{\partial b} &= \cancel{78} + \cancel{14a^2} + 2\times 3b^{\cancel{1}} - \cancel{66a} + 12a\cancel{b} - 28\cancel{b} \\ &= 6b + 12a - 28 \\ &= 0 \end{aligned} \tag{5}$$

の解が a,b を与える．この場合は $a=2.5, b=-1/3$．図 2 に式 3 のグラフを示す．尚，左図では $b=-1/3$ に，右図では $a=2.5$ に固定してある．

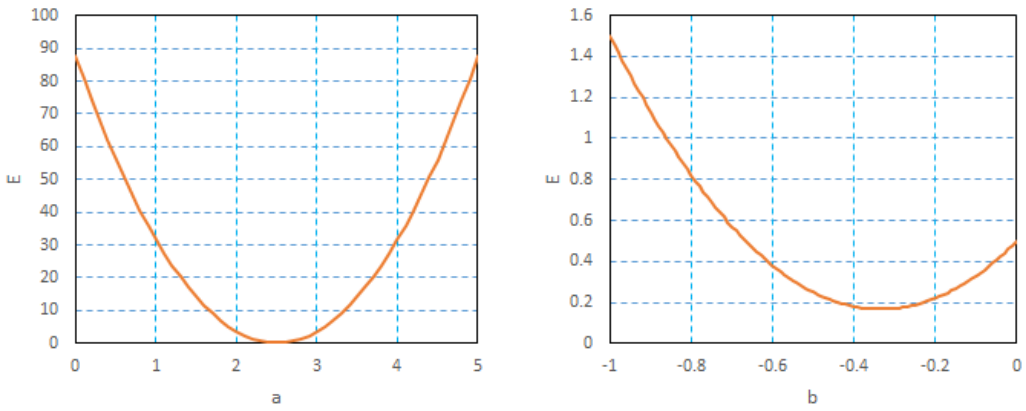

Figure 2: 式 3 の可視化

1.3 アルゴリズム

これから m 次多項式による近似式

$$y = a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_m x^m \tag{6}$$

に拡大する．データ点数が n 個の時の二乗誤差 E は

$$\begin{cases} E &= \displaystyle\sum_{k=0}^{n} (\Delta y_k)^2 \\ \Delta y_k &= y_k - (a_0 + a_1 x_k^1 + a_2 x_k^2 + \cdots + a_m x_k^m) \end{cases} \tag{7}$$

であり，

$$E = \sum_{k=0}^{n} \{y_k - (a_0 + a_1 x_k^1 + a_2 x_k^2 + \cdots + a_i x_k^i + \cdots + a_m x_k^m)\}^2 \tag{8}$$

となる．ここですべての係数 $a_i (i = 0 \sim m)$ の偏微分係数がゼロであることが必要となる．式 8 を一旦展開すると，

$$\begin{aligned} E &= \{y_0 - (a_0 + a_1 x_0^1 + a_2 x_0^2 + \cdots + a_i x_0^i + \cdots + a_m x_0^m)\}^2 \\ &+ \{y_1 - (a_0 + a_1 x_1^1 + a_2 x_1^2 + \cdots + a_i x_1^i + \cdots + a_m x_1^m)\}^2 \\ &+ \{y_2 - (a_0 + a_1 x_2^1 + a_2 x_2^2 + \cdots + a_i x_2^i + \cdots + a_m x_2^m)\}^2 \\ &+ \cdots \\ &+ \{y_k - (a_0 + a_1 x_k^1 + a_2 x_k^2 + \cdots + a_i x_k^i + \cdots + a_m x_k^m)\}^2 \\ &+ \cdots \\ &+ \{y_n - (a_0 + a_1 x_n^1 + a_2 x_n^2 + \cdots + a_i x_n^i + \cdots + a_m x_n^m)\}^2 \end{aligned} \tag{9}$$

となる．E の a_i による偏微分は，合成関数の微分を用いることで下記のように表すことが出来る．

$$\begin{aligned} \frac{\partial E}{\partial a_i} &= 2x_0^i \{y_0 - (a_0 + a_1 x_0^1 + a_2 x_0^2 + \cdots + a_i x_0^i + \cdots + a_m x_0^m)\} \\ &+ 2x_1^i \{y_1 - (a_0 + a_1 x_1^1 + a_2 x_1^2 + \cdots + a_i x_1^i + \cdots + a_m x_1^m)\} \\ &+ 2x_2^i \{y_2 - (a_0 + a_1 x_2^1 + a_2 x_2^2 + \cdots + a_i x_2^i + \cdots + a_m x_2^m)\} \\ &+ \cdots \\ &+ 2x_k^i \{y_k - (a_0 + a_1 x_k^1 + a_2 x_k^2 + \cdots + a_i x_k^i + \cdots + a_m x_k^m)\} \\ &+ \cdots \\ &+ 2x_n^i \{y_n - (a_0 + a_1 x_n^1 + a_2 x_n^2 + \cdots + a_i x_n^i + \cdots + a_m x_n^m)\} \\ &= 0 \end{aligned} \tag{10}$$

両辺を定数 2 で割り，$x_k^i y_k$ の項を左辺に移項し，添え字 k で整理すると

$$\begin{aligned} \sum_{k=0}^{n} x_k^i y_k &= \sum_{k=0}^{n} x_k^i (a_0 + a_1 x_k^2 + a_2 x_k^2 + \cdots + a_i x_k^i + \cdots a_m x_k^m) \\ &= a_0 \sum_{k=0}^{n} x_k^i + a_1 \sum_{k=0}^{n} x_k^{i+1} + a_2 \sum_{k=0}^{n} x_k^{i+2} + \cdots + a_i \sum_{k=0}^{n} x_k^{i+i} + a_m \sum_{k=0}^{n} x_k^{i+m} \end{aligned} \tag{11}$$

よって，下記の $a_0 \sim a_m$ に関する連立方程式を Jordan 法などで解けばよい．

$$\begin{cases} a_0(n+1) & +a_1 \sum x_k & +a_2 \sum x_k^2 & +\cdots + a_m \sum x_k^m & = \sum y_k \\ a_0 \sum x_k & +a_1 \sum x_k^2 & +a_2 \sum x_k^3 & +\cdots + a_m \sum x_k^{1+m} & = \sum x_k y_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_0 \sum x_k^i & +a_1 \sum x_k^{i+1} & +a_2 \sum x_k^{i+2} & +\cdots + a_m \sum x_k^{i+m} & = \sum x_k^i y_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_0 \sum x_k^m & +a_1 \sum x_k^{m+1} & +a_2 \sum x_k^{m+2} & +\cdots + a_m \sum x_k^{2m} & = \sum x_k^m y_k \end{cases} \tag{12}$$

2 課題

最小二乗法プログラムを Python で実装せよ． 式 12 は厳つく見えるが，コードはいたってシンプルにできる．

- データ点数 `n` と，近似多項式の次数 `m` を標準入力から入力する機能を備えること
- `n` 点の座標 (x, y) の組を標準入力から入力する機能を備えること
- データ点の数は，最大 10 点としてよい
- 最小二乗法による近似多項式の係数を標準出力に出力する機能を備えること
- 以下の点を 1 次式で最小二乗近似し，その 1 次式の係数を求めよ
  - $(-2, 1), (0, -1), (2, 5)$
- 以下の点を 3 次式で最小二乗近似し，その 3 次式の係数を求めよ
  - $(1, 2), (2, 5), (3, 7), (5, 10), (6, 17)$

3 参考：C++ソースコード

```
 1  #include <iostream>
 2  #include <math.h>
 3  using namespace std;
 4
 5  int main (void){
 6
 7       int n, m;
 8       double a[10][11], x[10], y[10];
 9
10       cin >> n >> m;
11       m++;
12       cout << m << endl;
13       for (int i=0;i<n;i++) cin >> x[i] >> y[i];
14       for(int i=0;i<m;i++){
15           for(int j=0;j<m;j++){
16               a[i][j]=0;
17               for(int k=0;k<n;k++) a[i][j] += pow(x[k],i+j);
18               cout << a[i][j] << " ";
19           }
20           a[i][n]=0;
21           for(int k=0;k<n;k++) a[i][n] += y[k]*pow(x[k],i);
22           cout << a[i][n] << endl;
23       }
24       return 0;
25  }
```

このソースコードの実行形式ファイルを `lsm`，Jordan 法の実行形式ファイルを `jordan` とすると，下記のインプットカード `in.txt` を用意しておけば，Linux シェルや Windows PowerShell のパイプ機能を用いて，

```
$ ./lsm < in.txt | ./jordan
```

で解が出力される．同様に，Python の場合は

```
$ python3 ./lsm.py < in.txt | python3 ./jordan.py
```

で過去の Jordan 法の Pyhton スクリプトを再利用できる．

インプットカードの例
データ 5 点に対し，近似多項式 3 次式で最小二乗法を適用する．データ点は (x, y) = (1, 2), (2, 5), (3, 7), (5, 10), (6, 17) である．

```
5 3
1 2
2 5
3 7
5 10
6 17
```

4 更なる検討

最小二乗法で使うデータと，その近似多項式をグラフで示せ． matplotlib を使うと，最小二乗法の結果を視覚的に確認することが出来る．この際，パイプ機能を使って処理するには限界がある．一方で，これまでのプログラム資産である jordan.py は流用したいところである．こんな時は jordan.py をライブラリ化して，それを lsm.py から呼び出すようにする．C 言語で#include < ··· > のような表現が python でも用意されている．

1. 呼び出し先のプログラムの内部に　def 呼び出す関数名　を記述し，
2. 呼び出し元のプログラムで　import 呼び出す関数のあるファイル　でインポートし，
3. 呼び出し元のプログラムで関数を呼び出せばよい

引数や返り値に注意のこと．具体的には，下記の Joudan 法で連立方程式を解くプログラムを，jordan_lib.py というファイル名で保存する．

```
-------------------------------------------------
def solve_jordan(n, a):
     # ここに jordan 法で連立方程式を解くプログラムが入る
     ans=[]
     for i in range(n):
         ans.append(a[i][n])
     return ans
-------------------------------------------------
```

続いて，課題で作成した最小二乗法プログラムを改良する．import jordan_lib でインポートし，返り値 = jordan_lib.solve_jordan(引数) で関数を呼び出す．あとは，グラフ描画のコードを追加する．

```
-------------------------------------------------
import jordan_lib
import numpy as np
import matplotlib.pyplot as plt
N = 400
# ここに最小二乗法を解くプログラムが入る
ans = jordan_lib.solve_jordan(m,a)  # ここで jordan 法を呼び出す

# 以下でグラフを描画する
xp = np.linspace(x[0],x[n-1],N+1)
z = []
for j in range(N+1):
     zz = 0
     for i in range(m):
         zz += ans[i]*xp[j]**i
     z.append(zz)
plt.scatter(x,y, s=100, c="red") # 最小二乗法の元になるデータの散布図プロット
plt.plot(xp,z)                   # 最小二乗法で求めた近似多項式のラインプロッ
ト
plt.show()
-------------------------------------------------
```