C 言語による最小二乗法の値の計算

学籍番号: XXkcXXX
坂本 直志

平成 18 年 6 月 12 日

1 課題

与えられたデータ x_i, y_i に対して、最小二乗法を用いて x_i, y_i の関係を近似する直線の方程式 $y = ax + b$ の a, b を求めるプログラムを C 言語で作りなさい。そして、与えられたデータ (C 参照) に対して実際に値を求めなさい。

2 最小二乗法とは

二つ値の組となるデータ $(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)$ に対して、これらのデータの間に $y = ax + b$ という関係が成り立つと仮定して、a, b を求めることを考える。

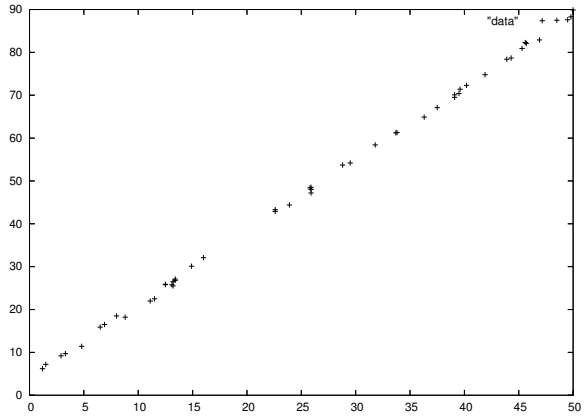

図 1: データ例

もし、得られたデータが正確に $y = ax + b$ の関係を満たしている時は、どのデータもその $y = ax + b$ で表される直線の上に乗っている。従って、データの中から任意に二点を選んで、その二点を通る直線を求めれば良い。

しかし、得られたデータが $y = ax + b$ を正確に満たしてない場合、つまり、得られたデータのどの二点を結ぶ直線に対して、他のデータ点がその直線に乗らない場合は、このような解法は使えない。このような状況では、なるべく各点に近い直線の式を求めることが必要となる。そのためには、一つ直線を決めた時にその直線が各点にどの程度近いか考える必要がある。

a, b に良い値が定まっている場合、各 $x_i. y_i$ に対して $ax_i + b$ と y_i は近い値になっているはずである。つまり、$y_i - ax_i - b$ は 0 にはならなくても 0 に近い値になるはずである。従って、各点に対してこの値の二乗を求め和をとると、各点に対する直線の近さを表すことができる。この二乗の和の値は a と b により定まるので、関数 $f(a, b)$ と考えることができる。

$$\begin{aligned} f(a,b) &= \sum_{i=1}^{n}(y_i - ax_i - b)^2 \\ &= \sum_{i=1}^{n}(y_i^2 + a^2x_i^2 + b^2 - 2ax_iy_i - 2by_i + 2abx_i) \end{aligned} \tag{1}$$

この式を最小にするような a, b は、与えられた各点を良く近似する直線の式 $y = ax + b$ となる。以後はこの $f(a, b)$ の値を小さくする a, b の求め方を考える。$f(a, b)$ は偏微分可能なので、この値を最小にする a, b に対して、$f(a, b)$ は同時に極値をとる。従って、 a, b に対して、偏微分を行い極値になる a, b を求める。

$$\begin{aligned} \frac{\partial f}{\partial a} &= \sum_{i=1}^{n}(2ax_i^2 - 2x_iy_i + 2bx_i) \\ &= 2a\sum_{i=1}^{n}x_i^2 - 2\sum_{i=1}^{n}x_iy_i + 2b\sum_{i=1}^{n}x_i \end{aligned} \tag{2}$$

$$\begin{aligned}\frac{\partial f}{\partial b} &= \sum_{i=1}^{n}(2b - 2y_i + 2ax_i) \\ &= 2nb - 2\sum_{i=1}^{n} y_i + 2a\sum_{i=1}^{n} x_i \end{aligned} \tag{3}$$

式 (2), (3) の値を 0 にする a, b が $f(a, b)$ の値を最小にする。つまり、次の連立方程式を解けば求める a, b が得られる。

$$\begin{cases} \left(\sum_{i=1}^{n} x_i^2\right) & a & + & \left(\sum_{i=1}^{n} x_i\right) & b & = & \sum_{i=1}^{n} x_i y_i \\ \left(\sum_{i=1}^{n} x_i\right) & a & + & n & b & = & \sum_{i=1}^{n} y_i \end{cases} \tag{4}$$

3 解法のあらまし

入力の値は x と y が組になり、標準入力から与えられるとする。そして、それらの値は scanf で読み込むとする。読み込みながら必要な集計を行う。さらに、読み込みが終了したら、上記の連立方程式 (4) を解く。

4 C 言語で使用した機能の説明

本節では今回の課題の解決に使用した C 言語の機能の説明をする。

scanf scanf は標準入力から値を得る関数である。値は関数の値として得るのではなく、変数に直接入る。関数の値は読み込みに成功した変数の個数となる。但し、ファイルの最後に達すると EOF という特別な値になる。

scanf の書式は次の通りである。

変数 = scanf(制御文字列,&変数名 ,...)

制御文字列は printf と同様に int 型変数であれば %d, float 型変数では%f である。但し、 double 型は %lf となることに注意しなければならない。

なお、不正な文字などが入力されると scanf は読み込みに失敗してしまうが、その際、読み込んだ不正な文字は読み飛ばしされず残ったままになる。その際、 scanf(”%∗c”) と一文字読み飛ばさなければ次の入力を得ることができない。

while,break while は条件式とブロックを指定すると、条件が正しいうちはブロックを実行し続ける。そのため、常に一番先に終了条件をチェックしなければならないが、 break を使うとそれが避けられる。

```
while(1){
  処理 1;
  if(終了条件){ break; }
  処理 2;
}
```

このようにすると、 while の条件式は常に成立するため、基本的には無限ループになるが、終了条件が成立すると、 break により、ループを一段抜け出す。つまり、 while の外に制御が移る。

なお、このような書式だと、ループの終了条件の書かれる位置が固定されないので、プログラムが読みづらくなるので、多用は控えるべきである。

int main と return C 言語では実際に実行したいプログラムは main 関数の中に書く。main 関数の本来の書式は次のようになる。

```
int main(int argc, char* argv[]){
  処理;
  return 戻り値;
}
```

但し、次のような省略のルールがある。

1. 関数の定義で戻り値 int は省略可能
2. 関数の終りには return を書く。すると return で指定した値が関数の値として返される。main の戻り値は、 OS がプログラムの正常、異常終了を区別するために使われる。通常 0 を返されると OS は正常終了だと認識する。なお、 return そのものが省略されると、本来は何も返さない。したがって、コンパイラによっては main 関数中で return を省略するとエラーが出る物もある。
3. main 関数の引数 argc と argv はコマンドラインからの引数指定に使うが、それらを使用しない場合は書かなくても問題ない。

このようなルールのため上記のプログラムは、引数も指定せず、戻り値も返さない場合は以下のようにも書ける。

```
main(){
  処理;
}
```

但し、今回のプログラムでは入力により異常終了する必要があるため、 int や return を指定した。

fprintf, stderr 標準出力へは printf で出力できるが、エラーの出力は標準出力にすべきではない。そこで、 UNIX や Windows では標準出力の他に標準エラー出力と言うエラー専用の出力チャネルが存在する。プログラム中でエラーが発生した場合は標準エラー出力に出力し、 0 以外の値を main 関数から返すのが正式なエラーの終了方法である。

fprintf はファイルなどに printf と同様の出力を与えるものである。第一引数にファイルハンドル、第二引数に制御文字列、第三引数以降は変数の列が入る。標準エラー出力にメッセージを出すにはファイルハンドル stderr を使い、fprintf で出力する。なお、標準出力は stdout というファイルハンドルなので、 fprintf に stdout を指定すると、 printf と同じ意味になる。

5 プログラムの説明

5.1 素朴なプログラム

まず、素朴なプログラムを付録 A に示した。

3 行目から 5 行目では変数の宣言を行っているが、ここで和を求める変数に対してはあらかじめ 0 で初期化している。和を求める変数の意味は次のとおりである。

sx2: $\sum_{i=1}^{n} x_i^2$. x の二乗和。

sx: $\sum_{i=1}^{n} x_i$. x の和。

sxy: $\sum_{i=1}^{n} x_i y_i$. x と y の積の和。

sn: n. データの個数。

sy: $\sum_{i=1}^{n} y_i$. y の和。

6 行目から 12 行目が while による繰り返しになっている。終了条件は 2 つの値 x と y を scanf で読み込んだとき、 EOF（データの終わり）で読み込みに失敗することである。データの終わりまで繰り返される動作は、上記の変数の意味にしたがってそれぞれの和を求める操作である。

繰り返しが終わったら、13,14 行目で連立方程式を解き、 15 行目で表示し、終了する。

5.2 問題点と修正プログラム

前節のプログラムは入力データが正常の場合、正常動作をする。しかし、異常なデータが来た場合、異常終了せず動作が止まって応答がなくなる。また、データ数が少ない場合、割算でエラーが出て停止してしまう。そこで、異常終了すべき事項を列挙し、対応することにする。

数値以外の入力 「不正なデータの入力」を表示して異常終了する。

入力データが x, y の組になってない 「データが対になってません」を表示して異常終了する。

データ数が少ない場合 「データが足りません」を表示して異常終了する。

なお、データが x の値 1 個の場合、「データが対になってません」のエラーとする。

これに対応するには、上記の場合それぞれエラーコード 1, 2, 3 を対応させ、異常終了ではそのエラーコードを return で返すことにする。またエラーメッセージは標準エラー出力 stderr に出力することにする。

さて、なぜ応答が返って来なくなるかと言うと、それは scanf は想定している入力値以外が入力された場合、その値を読み込まず、再度 scanf を行うと同じ文字を読んでしまうからである。但し、 scanf はその場合、読み込みに成功した数値の数を値として返すので、読み込みエラーは検出できる。

このようなエラーを検出するため、ファイル入力の部分では x と y は別々に読み、次のような処理を行う。

1. x を読み込む
2. x の読み込みに失敗した時、
   (a) EOF ならファイル終了で、集計処理へ進む
   (b) それ以外なら「不正なデータの入力」エラー
3. y を読み込む
4. y の読み込みに失敗した時、
   (a) EOF ならファイル終了で、「データが対になっていない」エラー
   (b) それ以外なら「不正なデータの入力」エラー
5. 加算処理

また、集計処理時にデータ数が 2 以上でなければ「データが足りません」エラーを出す。

このようにエラーに対応したプログラムを付録 B に示した。main の定義では main の型を省略してもよいが、return でエラー番号を明示するため、 int と明示した。

6 動作確認

修正プログラムが正常に動作することを調べる。実行例を図 2 に示す。

1. 入力が少ない場合
   空の入力、 一組しかない場合の実行例を示す。いずれも「データが足りません」エラーを出すため、正常に処理されている。

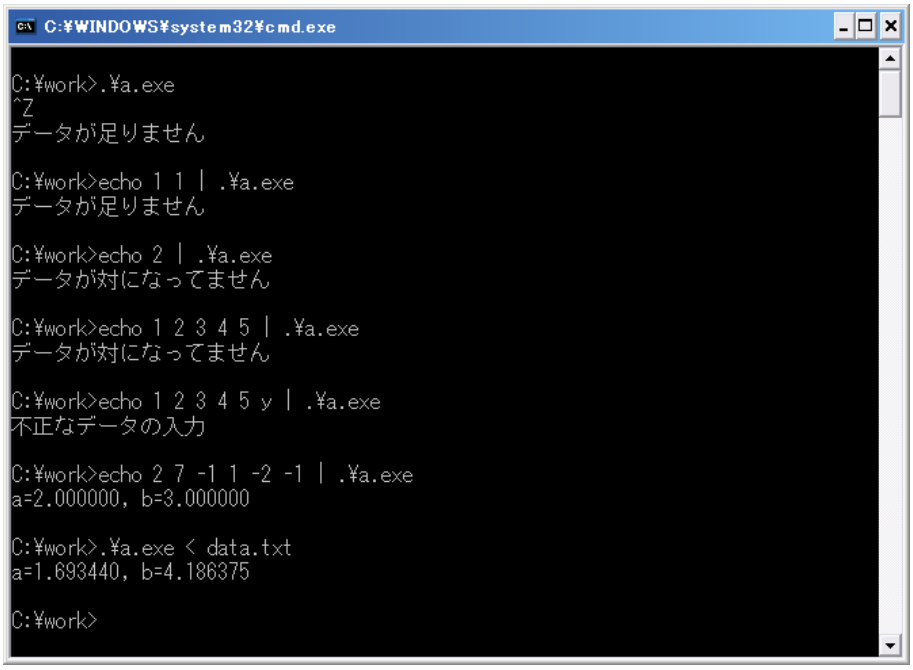

図 2: 実行例

2. 入力が対になっていない場合
   入力を 1 個と 5 個の場合の実行例を示す。いずれも「データが対になっていない」エラーを出すため、正常に処理されている。

3. 異常なデータの入力
   入力として、数以外の物を入れた場合、「不正なデータの入力」が出力されるため、正常に処理されている。

4. 自明な例の入力
   自明な例として、 $y = 2x + 3$ の直線上の値、(2,7), (-1,1), (-2,-1) を与えた所、 2 と 3 を正常に出力するため、正常に処理されている。

5. 与えられたデータを入力しても正しく実行され、値 $a = 1.693440, b = 4.186375$ が得られた。

7 まとめ、検討

本レポートでは、与えられた x_i, y_i データに対して直線近似する式の係数を最小二乗法で求めるプログラムを作成した。始めに作ったプログラムでは規定のフォーマットのデータ入力しか認めず、それ以外では異常終了せずに止まってしまうので、異常終了するように改めた。

プログラムでグラフを書けるようにできると良いと思った。

参考文献

[1] 森正武. 数値解析, 共立数学講座, 第 12 巻. 共立出版, 1973.

A　付録: 素朴なプログラム

```
#include <stdio.h>
main(){
  double x,y;
  double sx2=0,sx=0,sxy=0,sy=0;
  int sn=0;
  double a,b;
  while(scanf("%lf␣%lf",&x,&y)!=EOF){
    sx2+=x*x;
    sx+=x;
    sxy+=x*y;
    sn++;
    sy+=y;
  }
  a=(sn*sxy-sx*sy)/(sn*sx2-sx*sx);
  b=(sx*sxy-sy*sx2)/(sx*sx-sn*sx2);
  printf("a=%f,␣b=%f\n",a,b);
}
```

B　付録: 修正プログラム

```
#include <stdio.h>
int main(){
  double x,y;
  double sx2=0,sx=0,sxy=0,sy=0;
  double a,b;
  int sn=0;
  int count;
  while(1){
    count=scanf("%lf",&x);
    if(count!=1){
      if(count==EOF){
        break;
      }else{
        fprintf(stderr,"不正なデータの入力\n");
        return 1;
      }
    }
    count=scanf("%lf",&y);
    if(count!=1){
      if(count==EOF){
        fprintf(stderr,"データが対になってません\n");
        return 2;
      }else{
        fprintf(stderr,"不正なデータの入力\n");
```

```
            return 1;
          }
        }
        sx2+=x*x;
        sx+=x;
        sxy+=x*y;
        sn++;
        sy+=y;
    }
    if(sn<2){
        fprintf(stderr,"データが足りません\n");
        return 3;
    }
    a=(sn*sxy-sx*sy)/(sn*sx2-sx*sx);
    b=(sx*sxy-sy*sx2)/(sx*sx-sn*sx2);
    printf("a=%f, b=%f\n",a,b);
    return 0;
}
```

C 付録: 与えられたデータ

x	y	x	y
8.8	18.2	43.9	78.4
31.8	58.4	45.6	82.3
13.1	25.8	2.9	9.2
36.3	64.9	40.2	72.3
12.5	25.8	25.9	48.0
33.8	61.3	45.7	82.1
44.3	78.7	13.2	26.5
45.3	80.9	39.1	70.1
49.8	88.3	1.2	6.2
50.0	89.9	13.4	27.1
22.6	43.3	14.9	30.1
29.5	54.2	37.5	67.1
33.7	61.2	13.4	26.8
4.8	11.4	28.8	53.7
22.6	42.9	12.5	25.9
6.5	15.9	11.5	22.5
3.3	9.7	48.5	87.5

x	y
1.5	7.2
25.9	47.2
11.1	22.0
16.0	32.1
13.2	25.5
39.6	71.4
6.9	16.5
49.5	87.6
25.8	48.4
39.1	69.5
25.9	48.5
41.9	74.8
8.0	18.5
23.9	44.4
39.5	70.4
46.9	82.9