数値計算 Class-11 演習

21T2166D 渡辺大樹

2023 年 7 月 28 日

1　演習内容

Class-11 では前回行った基本関数の複数の組み合わせモデルによる補間関数の作成を、誤差の小さいものを自動的に選択して作成する、前方選択について演習していく。

今回新規に用いたコードをソースコード 1,2,3 に示す。長くなるため pdf 末尾に記載している。

以下ではどういった方法で誤差の小さい補間関数を自動で作成していくかを解説していく。

切り口としては誤差の小さい補間関数をどのように取得すべきか、というところにある。基本関数の組み合わせ総数が少ない場合、全探索 (すべてのパターンで試す) すればよいが、基本関数の組み合わせが多くなってしまうと計算量は階乗のオーダーになり爆発的に大きくなってしまう。これを解決し、かつ最適な補間関数を作成するための手法が前方選択になる。この手法は機械学習などにも使われるものである。

この手法はシンプルで、n 個の基本関数をそれぞれ 1 つずつ選び、最小二乗法でデータの補間関数を作る。(図 1)

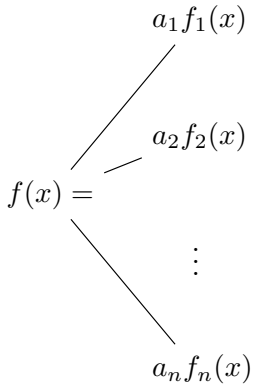

図 1　n 個の基本関数からそれぞれ補間関数を作成

出来た補間関数の誤差をそれぞれ計算し、一番小さくなった基本関数を選択する。(図 2)

ここからこの関数の続きを今と同じように生成していく。f_i 以外の基本関数で前回生成した関数の続きを最小二乗法で補間し誤差の小さい補間関数を選択する (図 3)。

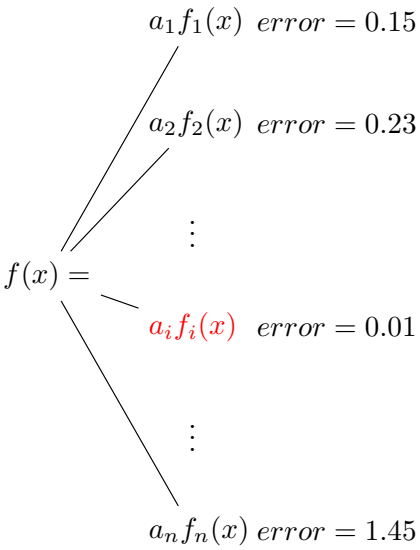

図 2　n 個の補間関数の誤差から一番小さいものを選択

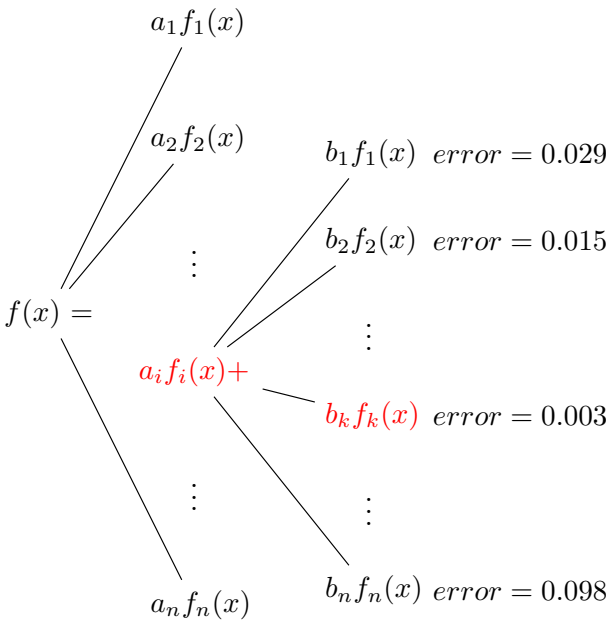

図 3　n-1 個 (f_i 以外) の基本関数でそれぞれ補間し誤差の小さいものを選択

これを繰り返していくことで、計算量は抑えられつつも最適な補間関数を得ることができる。
実際に 2 にはこのアルゴリズムが実装されている。

2　演習結果-考察

以下では前回も用いた 6 つのデータセットでこの前方選択と補間を行った結果を示す。

2.1 example2.txt

初めにデータセット example2.txt で行った結果を図 4 に示す。

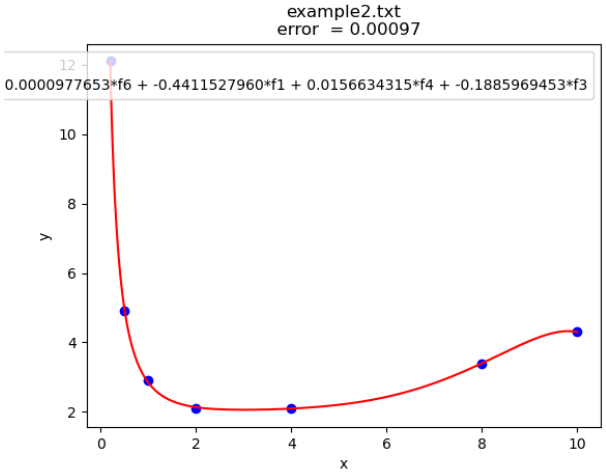

図 4　example2.txt でのデータ点と補間関数

見切れてしまっているが図 4 での補間モデルは

$$f(x) = 0.016x^3 - 0.189x^2 + 0.984x + \frac{2.469}{x} - 0.000e^x - 0.441$$

となった。e^x の係数は表記上 0 であるが 10^{-5} 程の係数である。誤差は 0.00097 となった。

前回行ったモデルセットによる選択的な補間では、一番誤差の小さい場合でも補間モデルが $f(x) = 0.001x^2 + 0.381x + \frac{2.403}{x}$ であり誤差は 0.0072 と今回の補間モデルではその関数が複雑になったことで誤差も縮まっていることが分かる。

2.2 nh_bb_age_length.txt

続いてデータセット nh_bb_age_length.txt で行った結果を図 5 に示す。

これも図では見切れているが補間モデルは

$$f(x) = 0.003x^3 - 0.276x^2 + 4.367x - \frac{0.036}{x} - 0.001e^x + 50.748$$

となった。誤差は 0.0003 となった。

これも前回行った補間では誤差が 0.039 ほどだったためおよそ $\frac{1}{100}$ も誤差が小さくなったことが分かる。

2.3 nh_bb_age_weigth.txt

続いてデータセット nh_bb_age_weigth.txt で行った結果を図 6 に示す。

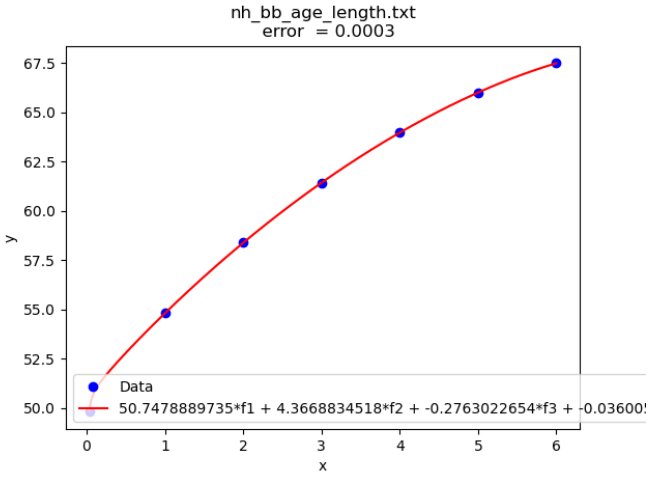

図 5　nh_bb_age_length.txt でのデータ点と補間関数

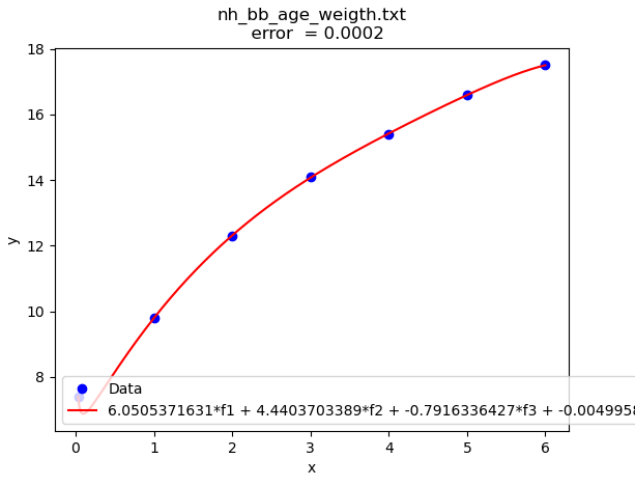

図 6　nh_bb_age_weigth.txt でのデータ点と補間関数

補間モデルは

$$f(x) = 0.071x^3 - 0.792x^2 + 4.440x + \frac{0.040}{x} - 0.005e^x + 6.051$$

となり、誤差は 0.0002 となった。

これも前回行った補間では誤差が 0.009 だったため前回の補間よりもより精度のよい補間ができていることが分かる。

2.4　nh_covid-italy.txt

続いてデータセット nh_covid-italy.txt で補間を行った結果を図 7 に示す。

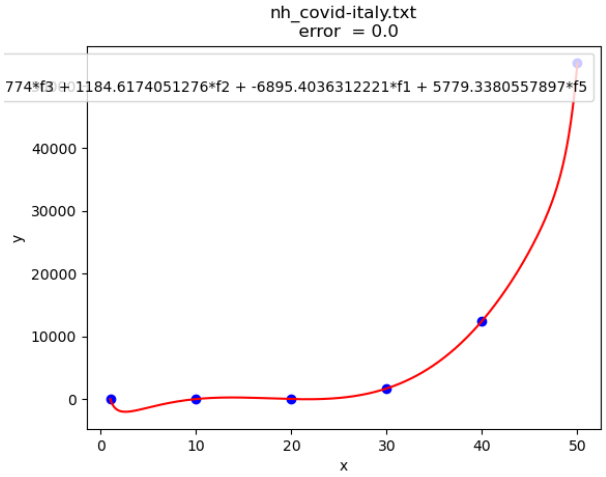

図 7　nh_covid-italy.txt でのデータ点と補間関数

図 7 の補間モデルは

$$f(x) = 1.255x^3 + -67.807x^2 + 1184.617x + \frac{5779.338}{x} + 0.000e^x - 6895.404$$

となった。誤差は 0.0 と非常に小さく、1 を実行したときに出力されたデータファイルを見てもほとんど 0 に近い値になっている。

これも前回行った補間では誤差は $3.1 \cdot 10^6$ ほどだったため一気に精度が上がったことが分かる。

ただこの図 7 を見ると一つ目のデータ点のすぐ後の補間関数が少し下に下がっている。このデータはおそらくイタリアのコロナ感染者についてのデータであると思うが実際に感染者の増加がこの通りであるわけではない可能性が高い。

補間モデルの精度を上げることはもちろんメリットもあるが、この例のように必ずしも誤差の小さな補間モデルが優れているわけではないと考えられる。

2.5　nh_fish.txt

続いてデータセット nh_fish.txt で補間を行った結果を図 8 に示す。

図 8 の補間モデルは

$$f(x) = 0.000x^3 - 0.003x^2 + 1.902x + \frac{42273.604}{x} - 450.301$$

となった。データ点のばらつきのせいか e^x の項は最小二乗法の結果がうまく出なかったため項が入っていない。誤差は 0.534 となった。

前回行った補間での誤差は 0.564 で、確かに今回は小さくなっているがこれまでの演習結果と比較するとあまり変わらないように見える。これはおそらくデータのばらつきが大きいため正確な関数の近似が難しいことが要因に挙げられる。こういったばらつきのあるデータは、その関係を示し

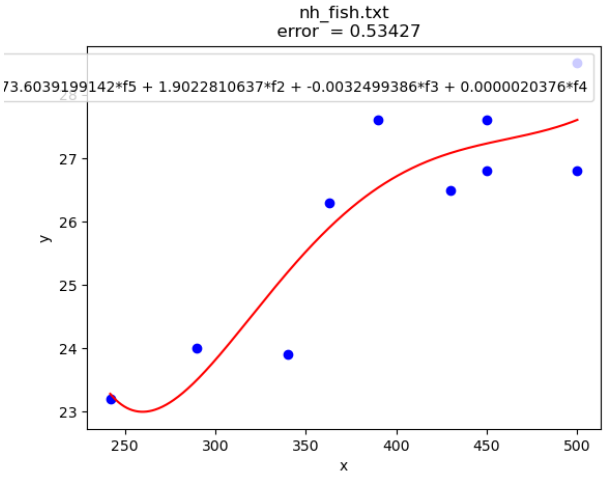

図 8　nh_fish.txt でのデータ点と補間関数

たいのなら正確で誤差のない複雑なモデルの補間よりも、シンプルな一次関数や対数関数での最小二乗法などでの補間が適していると考えられる。

2.6　nh_wine.txt

最後にデータセット nh_wine.txt で補間を行った結果を図 9 に示す。

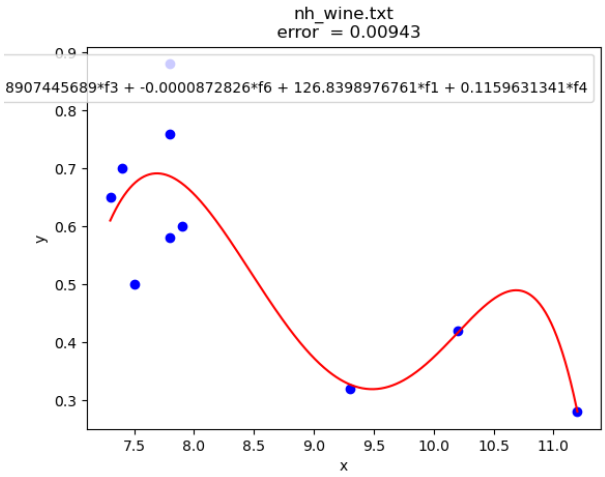

図 9　nh_wine.txt でのデータ点と補間関数

図 9 の補間モデルは

$$f(x) = -1.891x^3 - 0.000x^2 - \frac{514.318}{x} - 0.000e^x - 126.840$$

となった。こちらも 6 回目の選択で LU 分解出来ない係数行列が出来てしまったようで x の項が

なくなっている。誤差は 0.0094 となった。

前回の補間での誤差は 0.0131 であり 2.5 でのデータセット同様あまり変化の少ない結果となった。理由も 2.5 と同じく、データのばらつきが起因しているものと考えられる。

ソースコード 1　main_forward_selection.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <float.h>
#define N 220
#define NF 6 /* 基本関数の数*/
#include "my_library_v3.h"
#include "minjijo_lusolve_extended.h"
#include "forward_selection.h"
/* main_forward_selection */
int main(void)
{
    int n; /* n 個データ点数, */
    double x[N], y[N]; /* データ */
    // char *data = "example2.txt";
    // char *data="nh_covid-italy.txt";
    // char *data="nh_bb_age_weigth.txt";
    // char *data="nh_bb_age_length.txt";
    // char *data="nh_fish.txt";
    char *data="nh_wine.txt";
    char *path =
        "C:/Program_code/NumMeth/Class11STU";
        char fname_data[200];
    printf("このプログラムは前方選択を使用して\n");
    printf("誤差が小さいモデルを見つ出す\n");
    /* データの入力 */
    sprintf(fname_data, "%s/data/%s", path, data);
    n = data_input(fname_data, x, y);
    /* 前方選択 */
    forward_step_wise_selection(NF, x, y, n, path, data);
    printf("モデル誤差と求めた基本関数の係数はファイル out_%s に保存されます\n",
         data);
    printf("補間結果はファイル mk#_est_%s に保存されます\n", data);
    printf("処理の終了\n");
    return 0;
}
```

ソースコード 2　forward_selection.h

```
typedef struct struct_model
{
    int k; /* 基本関数の数k */
    int f_id[N]; /* 基本関数のid 1,...,NF*/
    double sol[N]; /* モデルの求めたa1,...ak の係数 */
    double err; /* モデルの誤差 */
} struct_model;
enum result
{
    TRUE,
    FALSE
};
int f_in_model(struct_model model, int fi)
{
    int i;
    int found = FALSE;
    for (i = 1; i <= model.k; i++)
    {
        if (fi == model.f_id[i])
        {
            found = TRUE;
            break;
        }
    }
    return found;
}
void copy_model(struct_model *p_current_m, struct_model *p_best_m)
{
    int j;
    (*p_best_m).err = (*p_current_m).err;
    (*p_best_m).k = (*p_current_m).k;
    for (j = 1; j <= (*p_best_m).k; j++)
    {
        (*p_best_m).f_id[j] = (*p_current_m).f_id[j];
        (*p_best_m).sol[j] = (*p_current_m).sol[j];
    }
}
void forward_step_wise_selection(int nf, double x[N], double y[N], int n,
                                 char *path, char *data)
{
    int i, k, j;
```

```
42      struct_model current_m, best_m; /* 現行モデルと最高モデル */
43      current_m.k = 0;
44      current_m.err = DBL_MAX; /* DBL_MAX <float.h> */
45      best_m.err = DBL_MAX;
46      char fname_hokan[200], fname_out[200];
47      FILE *fp_out;
48      /* モデルの誤差と基本関数と求めた係数のファイル名を設定する */
49      sprintf(fname_out, "%s/results/out_%s", path, data);
50      /* モデルの誤差と基本関数と求めた係数のファイルを開く */
51      fp_out = fopen(fname_out, "w");
52      for (k = 1; k <= nf; k++) /* k basic functions */
53      {
54          printf("--- k = %d ---\n", k);
55          for (i = 1; i <= nf; i++)
56          {
57              if (f_in_model(current_m, i) == FALSE)
58              {
59                  /* 現在のモデルで関数fi を試す */
60                  current_m.k = k;
61                  current_m.f_id[k] = i;
62                  /* モデルの係数を求める */
63                  compute_model(n, k, x, y, current_m.f_id, current_m.sol);
64                  /* モデルの誤差を求める */
65                  current_m.err = model_error(n, k, x, y, current_m.sol,
66                                              current_m.f_id);
67                  /* 最高モデルを更新する */
68                  if (current_m.err < best_m.err)
69                      copy_model(&current_m, &best_m);
70                  /* 誤差と求めた基本関数の係数を出力 */
71                  printf("%d %f ", i, current_m.err);
72                  for (j = 1; j <= k; j++)
73                  {
74                      printf("%.3f*f%d ", current_m.sol[j], current_m.f_id[j]);
75                  }
76                  printf("\n");
77              }
78          }
79          /* モデルの誤差，モデルの基本関数と求めた係数を出力 */
80          model_error_output(fp_out, k, best_m.err, best_m.k, best_m.sol,
81                             best_m.f_id);
82          printf("k=%d 最高のモデル\n", best_m.k);
83          printf(" 誤差: %f\n", best_m.err);
```

```
84          printf(" モデル: ");
85          for (j = 1; j <= best_m.k; j++)
86          {
87              printf("%.3f*f%d ", best_m.sol[j], best_m.f_id[j]);
88          }
89          printf("\n");
90          /* グラフを描くための準備（数表を出力）*/
91          sprintf(fname_hokan, "%s/results/mk%d_est_%s", path, best_m.k, data);
92          // printf("%s\n", fname_hokan);
93          data_output(fname_hokan, n, best_m.k, x, y, best_m.sol, best_m.f_id);
94          /* 現在モデルを更新する */
95          copy_model(&best_m, &current_m);
96      }
97      /* ファイルを閉じる */
98      fclose(fp_out);
99  }
```

ソースコード 3　fig.py

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Mon Jun 22 14:03:55 2020
5
6  @author: hernan
7  """
8  import pandas as pd
9  import matplotlib.pyplot as plt
10
11
12 #元のデータファイル名
13 fname_datas = ["nh_fish.txt","nh_wine.txt"]
14
15 for fname_data in fname_datas:
16     fm = "../results/out_"+fname_data #モデルの誤差，モデルの係数と基本関数
17     # モデルをエラーごとに並べ替える
18     column_names = [i for i in range(0, 5)]
19     #df_m_sort = pd.read_csv(fm, sep="\t", header=None, names = column_names)
20     #df_m_sort.sort_values(by=[1])
21
22     #調べるモデル
23     for model_number in range(1,6):
24
```

```
25        fd = "../data/"+fname_data #データ
26        fe = "../results/mk"+str(model_number)+"_est_"+fname_data #補完の結果
27
28        #出力されたモデルを取得する
29        df_m = pd.read_csv(fm, sep="\t", header=None, skiprows = model_number
              -1 , nrows = 1)
30        error = str(df_m.iloc[0,1])
31        model = ' + '.join(list(df_m.iloc[0,2:]))
32
33
34        #元のデータを読み込む
35        df_d = pd.read_csv(fd, sep="\t", header=None, skiprows=1)
36
37        #補間されたデータを読み込む
38        df_e = pd.read_csv(fe, sep="\t", header=None)#
39
40        # 元のデータと補間されたデータをプロットする
41        plt.figure()
42        plt.plot(df_d[0],df_d[1], 'ob')
43        plt.plot(df_e[0],df_e[1], '-r')
44        plt.title(fname_data + "\n error = " + str(error))
45        plt.xlabel('x')
46        plt.ylabel('y')
47        plt.legend(["Data", model])
48        plt.savefig("plot_" + fname_data + ".png")
49        plt.show()
```