平成26年度 3回生後期学生実験エージェント
課題１

竹田創)

提出日：平成 26 年 12 月 9 日

1　プログラム概要

SVM.cc は２次計画問題をとくライブラリ（QuadProg++）と、データとクラスの学習データを元にしてサポートベクタマシンを作成するプログラムである。

2　外部仕様

2.1　プログラム名とファイルの説明

QuadProg++.cc や QuadProg++.hh は２次計画問題をとくためのライブラリのプログラムである。sample_linear.dat と sample_circle.dat はデータとクラスの学習データのプログラムである。1_circle.dat,a_linear.dat,minus_1_circle.dat,minus_1
は SVM.cc はサポートベクターマシンを実現するプログラムである。

```
1 #include <iostream>
2 #include <string>
3 #include <fstream>
4 #include<stdlib.h>
5 #include "QuadProg++.hh"
6 #include <cmath>
7 #define DATA_NUM 100
8 #define sigma 10.0
```

```
 9
10 using namespace std;プロトタイプ宣言
11
12 //
13 int read_data(); 2点間の距離を出力する関数。
14 //カーネルで利用Gauss
15 double norm(double x_i_0,double x_i_1,double x_j_0,double x_j_1);
16 double PolinomicalKernel(double x_i_0,double x_i_1,double x_j_0,double x_j
17
18 double GaussianKernel(double x_i_0,double x_i_1,double x_j_0,double x_j_1)
19 struct Dataset{
20         double input_first,input_second,y;
21
22 };
23
24 int read_data() {
25         return 0;
26 }
27
28
29
30
31 int main (int argc, char *const argv[]) {
32         double G[MATRIX_DIM][MATRIX_DIM], g0[MATRIX_DIM],
33                 CE[MATRIX_DIM][MATRIX_DIM], ce0[MATRIX_DIM],
34                 CI[MATRIX_DIM][MATRIX_DIM], ci0[MATRIX_DIM],
35                 alpha[MATRIX_DIM],plot[MATRIX_DIM];
36         double weight[2],theta;
37         double x,y;
38
39         int n, m, p;
40
41         double sum = 0.0;
42
43         int i=0;
44         double Kernel;
45         n = DATA_NUM;
46
47         struct Dataset data[DATA_NUM];
48         //でファイル読み込んで、ifstreamdataから[0]dataに格納[99]
49         /*double x[100][100];
50           double y[100];
51          */
```

```
52          //cout << "ifstream" << endl;
53          ifstream ifs("sample_circle.dat");
54          string str;
55
56          if(ifs.fail()) {
57                  cerr << "File do not exist.\n";
58                  exit(0);
59          }
60
61          while(getline(ifs, str)) {
62                  if(i==100) break;
63                  data[i].input_first=0; data[i].input_second=0; data[i].y=0
64
65                  sscanf(str.data(), "%lf %lf %lf", &data[i].input_first, &d
66                  /*
67                     cout<<i<<endl;
68                     cout << "first = " << data[i].input_first << endl;
69                     cout << "second = " << data[i].input_second << endl;
70                     cout << "result = " <<  data[i].y << endl;
71                   */
72                  /*
73                     x[i][0]=data[i].input_first;
74                     x[i][1]=data[i].input_second;
75                     y[i]=data[i].y;
76                   */
77
78
79                  i++;
80          }
81          //G[i][j]
82
83
84          {
85                  {
86                          /*の
87                             G(i,j要素は)y_i*y_j*(x_i,x_j)
88                             (x_i,x_j)=(1+x_i*x_j)^2=((x[i][0] ,x[i][1] ),(x
89
90
91                             (6 は37)ベクトルx1
92                             (48 27) はベクトルx2
93                             ..は
94
```

```
95                                  −1ベクトルy1は
96                                  −1ベクトルy2
97
98                                  x[i][0]=data[i].input_first;
99                                  x[i][1]=data[i].input_second;
100                                 y[i]=data[i].y;
101                             //G[i][j]=y_i*y_j*(x_i,x_j)
102                             //(x_i,x_j)=(1+x_i*x_j)^2
103
104                              */
105                             for (int i = 0; i < n; i++){のクラ
    ス出力
106                                     //0
107                                     if(data[i].y==−1){
108                                             cout << data[i].input_first <<"\t"
109
110                                     }
111                                     for (int j = 0; j < n; j++){
112
113                                             //C/Cでは++argvは[]
    型char
114                                             string argv1;
115                                             if(argv[1]){
116                                                     argv1=argv[1];
117                                             }else{
118                                                     argv1="No Kernel";
119                                             }
120                                             cout<<argv1<<endl;
121                                             if(argv1=="P"){
122                                                     //cout<<"PolinomicalKernel
123                                                     Kernel=PolinomicalKernel(d
124
125
126                                             }else if(argv1=="G"){
127                                                     //cout<<"GaussianKernel"<<
128                                                     Kernel=GaussianKernel(data
129
130                                             }else{
131                                                     //cout<<"NO Kernel"<<endl;
    内積でカーネルトリックなし
132                                                     //
133                                                     Kernel=(data[i].input_firs
134
```

```
135
136                                                 }
137
138
139                                                 G[i][j] =data[i].y*data[j].y*Kerne
140
141                                                 if(i==j) G[i][j]+=1.0e-7;
142
143
144
145                                                 //cout<<"G["<<i<<"]["<<j<<"]=" <<
146
147                                         }
148                                         {/*シグモイドカーネル
149                                                 //
150                                                 Kernel=(double)tanh((1.0+data[i].i
151                                                 G[i][j] =(double)data[i].y*data[j]
152                                                 if(i==j) G[i][j]+=1.0e-7;*/
153                                         }
154                                 }
155
156                         }
157                 }
158
159                 {//の要素は全てg0-1
160
161                         for (int i = 0; i < n; i++){
162                                 g0[i] =(double) -1;
163                                 //      cout<<"g0["<<i<<"]="<<g0[i] <<endl;
164                         }
165                 }
166                 m = 1;
167                 {//はCE(y1 y2 .. yn)
168                         for (int i = 0; i < n; i++){
169                                 for(int j=0;j<n;j++){
170
171                                         CE[i][0] =data[i].y;
172                                         //      cout<<"CE["<<i<<"]["<<j<<"]="<<CE[
173                                 }
174                         }
175
176
177
```

```
178         }
179
180
181         {//はce00
182                 for (int i = 0; i < n; i++){
183
184                         ce0[i]=(double)0;
185
186
187
188                 }
189         }
190
191
192         {
193                 /*はCI
194                   (1 0 0 0 0 )
195                   (0 1 0 0 0 )
196                   (0 0 1 0 0 )
197                   (0 0 0 1 0 )
198                   (0 0 0 0 1 )
199                  */
200
201                 for (int i = 0; i < n; i++)
202                         for (int j = 0; j < n; j++)
203                                 if(i==j){
204                                         CI[i][j]=(double)1;
205                                         //cout<<"CI["<<i<<"]["<<j<<"]=" <<
206                                 }else{
207                                         CI[i][j]=(double)0;
208                                         //cout<<"CI["<<i<<"]["<<j<<"]=" <<
209                                 }
210
211
212         }
213
214         {
215                 //はci0(0 0 0 0 0 )
216                 for (int j = 0; j < n; j++){
217                         ci0[j] =(double)0;
218                         //cout<<"ci0["<< j<<"]="<<ci0[i] <<endl;
219                 }
220         }
```

```
221             //でとくsolve_quadprog
222
223
224             try{例外が発生する可能性のあるコード
225                     //
226
227                     solve_quadprog(G, g0,100,  CE, ce0,1, CI, ci0,100, alpha);
228             } catch (const std::exception& ex) {
229                     std::cerr << "solve_quadprog_failed" << ex.what() << std::
230                     throw;
231             }
232
233             //  std::cout << "f: " << solve_quadprog(G, g0,DATA_NUM,
    CE, ce0,1, CI, ci0,1, alpha) << std::endl;
234             //std::cout << "alpha: ";
235             //for (int i = 0; i < n; i++)
236             //    std::cout <<"alpha["<<i<<"]"<< alpha[i] << ' ';
237             //std::cout << std::endl;出力
238
239
240             //
241             cout<<"alpha"<<endl;
242             for ( i = 0; i <n; i++)
243                     printf("%d\t%f\n", i, alpha[i]);//は浮動小
    数点型で出力alpha重み
244
245
246             //出力weight
247
248             for(i=0;i<n;i++){
249                     weight[0]+=alpha[i]*data[i].y*data[i].input_first;
250                     weight[1]+=alpha[i]*data[i].y*data[i].input_second;
251
252
253             }
254             cout<<"weight[0]="<<weight[0]<<endl;
255             cout<<"weight[1]="<<weight[1]<<endl;しきい値
256
257             //出力theta本来はどの
258             //alpha[kのときでも]は同じになるはずである。ここではthetak
    をとる=5
259             theta=(weight[0]*data[5].input_first+weight[1]*data[5].input_secon
260             cout<<"theta="<<theta<<endl;
```

```
261
262             //(w,x)=となるときを考えるthetaカーネルトリックなし()カー
    ネルトリックなしのときの境界の式（一次式）
263             //
264             //cout<<weight[0]<<"x+"<<weight[1]<<"y="<<theta<<endl;
265
266
267             //Kernel(w,x)=となるときが境界となるtheta
268             cout ガウスカーネルを使ったときの境界の式<<""<<endl;
269             cout<<"exp(-("<<weight[0]<<"-x)**2-("<<weight[1]<<"-y)**2/2.0/"<<s
270
271
272
273             /*
274             //で描画できる点書き出しgnuplot2
275             printf("%f\t%f\n",0.00,theta/weight[1]);
276             printf("%f\t%f\n",theta/weight[0],0.00);
277              */
278
279
280
281 }
282 double norm(double x_i_0,double x_i_1,double x_j_0,double x_j_1){
283             return pow(x_i_0 -x_j_0,2.0)+ pow(x_i_1 -x_j_1,2.0);
284
285 }
286 double PolinomicalKernel(double x_i_0,double x_i_1,double x_j_0,double x_j
287             return pow((1.0+x_i_0*x_j_0+x_i_1*x_j_1),2.0);
288
289 }
290
291 double GaussianKernel(double x_i_0,double x_i_1,double x_j_0,double x_j_1)
292             return exp(-norm(x_i_0,x_i_1,x_j_0,x_j_1)/2.0/sigma/sigma);
293
294
295 }
296
297 /*
298 double printGaussian(weight[0],weight[1],theta){点、ガウスカー
    ネルの境界の式をプロットする。
299             //1000
300             for ( i = 0; i <10000; i++){
301                     plot[i]=;
```

```
302                     printf("%d\t%f\n", i, plot[i]);//は浮動小数
    点型で出力alpha
303             }
304 }*/
```

2.2 プログラム引数の説明

コンパイルしたファイルを実行するときにコマンドライン引数としてG、Pまたはなにもなしをとる。Gをとったときはガウスカーネルで計算して、Pをとったときは多項式カーネルで計算し、なにもとらなかったときはカーネルトリックなしで計算する。

2.3 入出力ファイル及び参照ファイル

sample_linear.dat と sample_circle.dat はデータとクラスの学習データのプログラムである。このデータをよみとってSVMを作成する。また出力結果としてalphaの組を出力する。

3 コンパイル方法

```
g++-4.7 -Wall QuadProg++.cc SVM.cc
./a.out
./a.out G
./a.out P
```

3.1 実行例

```
alpha
0 0.000000
1 -0.000000
2 0.000000
3 0.000000
4 -0.000000
```

```
5 0.000000
6 0.000000
7 -0.000000
8 0.000000
9 -0.000000
10 0.000000
11 -0.000000
12 0.000000
13 0.000000
14 0.000000
15 0.000000
16 0.000000
17 0.000000
18 0.000000
19 0.000000
20 14.900113
21 0.000000
22 6.462795
23 -0.000000
24 0.000000
25 126.382013
26 86.638915
27 0.000000
28 -0.000000
29 -0.000000
30 41.650582
31 0.000000
32 0.000000
33 0.000000
34 -0.000000
35 -0.000000
36 0.000000
37 0.000000
38 -0.000000
39 -0.000000
```

```
40 0.000000
41 0.000000
42 0.000000
43 -0.000000
44 0.000000
45 -0.000000
46 -0.000000
47 0.000000
48 129.805469
49 -0.000000
50 8.902141
51 0.000000
52 -0.000000
53 -0.000000
54 -0.000000
55 6.116575
56 -0.000000
57 -0.000000
58 0.000000
59 0.000000
60 -0.000000
61 -0.000000
62 -0.000000
63 0.000000
64 0.000000
65 0.000000
66 -0.000000
67 0.000000
68 -0.000000
69 -0.000000
70 0.000000
71 0.000000
72 7.242414
73 -0.000000
74 0.000000
```

```
75 0.000000
76 -0.000000
77 -0.000000
78 0.000000
79 0.000000
80 0.000000
81 80.888327
82 0.000000
83 0.000000
84 -0.000000
85 -0.000000
86 0.000000
87 0.000000
88 0.000000
89 0.000000
90 -0.000000
91 -0.000000
92 -0.000000
93 -0.000000
94 -0.000000
95 0.000000
96 43.959578
97 0.000000
98 2.340770
99 0.000000
weight[0]=12.9251
weight[1]=-177.845
theta=-2285.13
```

4 内部仕様

サポートベクタマシンを実現する SVM.cc について説明する。Quad-Prog++を利用して 2 次計画問題をとくため、G、CE、CI、g0、ce0、ci0 に適切な値を代入する。CI は単位行列、CE は (y1 y2 y3 ..) という行列、ci0 と ce0 は全ての成分が 0 のベクトル、g0 は全ての成分が-1 のベクト

[scale=0.8]linearNokernel.png

図 1: 線形のデータをカーネルトリックなしで解析

[scale=0.8]linearPolinomical.png

図 2: 線形のデータを多項式カーネルで解析

ルである。行列 G は G[i][j] が y[i]*y[j]*Kernel(xi,xj) の成分をもつ行列である。

5　評価結果

線形のデータはカーネルトリックなしのときと多項式カーネルのときでうまくとけていることが分かった。ガウスカーネルの gnuplot で出力がうまくいかなかった。

6　考察

まだガウスカーネルの場合の描画ができていません。後日実装できしだい再提出します。