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 12 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 #define FILENAME "./data/sample_circle.dat"
10 using namespace std;構造体宣言
11
12 //
13 double GaussianKernel(double x_i_0,double x_i_1,double x_j_0,double x_j_1)
14 struct Dataset{
15   double input_first,input_second,y;
16
17 };プロトタイプ宣言
18
19 //
20 //int read_data(Dataset*  data);2点間の距離を出力する関数。
21 //カーネルで利用Gauss
22 double norm(double x_i_0,double x_i_1,double x_j_0,double x_j_1);
23 double PolinomicalKernel(double x_i_0,double x_i_1,double x_j_0,double x_j
24 int read_data(Dataset*  data);
25 void print_alpha(double alpha[MATRIX_DIM],int *alpha_max_number);
26 //int print_theta()
27 void print_theta(string argv1,int alpha_max_number,Dataset* data, double w
28 void  f(string argv1,double weight[2],int x0,int x1,double theta);
29
30
31
32 int main (int argc, char *const argv[]) {
33   double G[MATRIX_DIM][MATRIX_DIM], g0[MATRIX_DIM],
34     CE[MATRIX_DIM][MATRIX_DIM], ce0[MATRIX_DIM],
35     CI[MATRIX_DIM][MATRIX_DIM], ci0[MATRIX_DIM],
36     alpha[MATRIX_DIM];
37   double weight[2],theta;
38
39
40   int n,m,alpha_max_number=1,x0,x1;
41
42
43
44   int i=0;
45   double Kernel;
46   n = DATA_NUM;
47   string argv1;
48
49   struct Dataset data[DATA_NUM];
50
51   read_data(data);
```

```
52
53    {
54      {
55        /*の
56          G(i,j要素は)y_i*y_j*(x_i,x_j)
57          (x_i,x_j)=(1+x_i*x_j)^2=((x[i][0]  ,x[i][1]  ),(x[j][0],x[j][1]))
58
59
60          (6 は37)ベクトルx1
61          (48  27) はベクトルx2
62          ..は
63
64          −1ベクトルy1は
65          −1ベクトルy2
66
67          x[i][0]=data[i].input_first;
68          x[i][1]=data[i].input_second;
69          y[i]=data[i].y;
70          //G[i][j]=y_i*y_j*(x_i,x_j)
71          //(x_i,x_j)=(1+x_i*x_j)^2
72
73          */
74        for (int i = 0; i < n; i++){のクラス出力
75          //0
76          if(data[i].y==−1){
77
78
79          }
80          for (int j = 0; j < n; j++){
81
82            //C/Cでは++argvは[]型char
83
84            if(argv[1]){
85              argv1=argv[1];
86            }else{
87              argv1="No Kernel";
88            }
89
90            if(argv1=="P"){
91
92              Kernel=PolinomicalKernel(data[i].input_first,data[i].input_seco
93
94
```

```
95              }else if(argv1=="G"){
96
97                Kernel=GaussianKernel(data[i].input_first ,data[i].input_second
98
99              }else{内積でカーネルトリックなし
100
101               //
102               Kernel=(data[i].input_first*data[j].input_first+data[i].input_
103
104
105             }
106
107
108             G[i][j] =data[i].y*data[j].y*Kernel;
109
110             if(i==j) G[i][j]+=1.0e-7;
111
112
113
114
115
116           }
117           {/*シグモイドカーネル
118            //
119            Kernel=(double)tanh((1.0+data[i].input_first*data[j].input_first+
120            G[i][j] =(double)data[i].y*data[j].y*Kernel;
121            if(i==j) G[i][j]+=1.0e-7;*/
122           }
123         }
124
125       }
126     }
127
128     {//の要素は全てg0-1
129
130       for (int i = 0; i < n; i++){
131         g0[i] =(double) -1;
132
133       }
134     }
135     m = 1;
136     {//はCE(y1 y2 .. yn)
137       for (int i = 0; i < n; i++){
```

```
138         for(int j=0;j<n;j++){
139
140           CE[i][0] =data[i].y;
141
142         }
143       }
144
145
146
147     }
148
149
150     {//はce00
151       for (int i = 0; i < n; i++){
152
153         ce0[i]=(double)0;
154
155
156
157       }
158     }
159
160
161     {
162       /*はCI
163         (1 0 0 0 0 )
164         (0 1 0 0 0 )
165         (0 0 1 0 0 )
166         (0 0 0 1 0 )
167         (0 0 0 0 1 )
168       */
169
170       for (int i = 0; i < n; i++)
171         for (int j = 0; j < n; j++)
172           if(i==j){
173             CI[i][j]=(double)1;
174
175           }else{
176             CI[i][j]=(double)0;
177
178           }
179
180
```

```
181     }
182
183     {
184       //はci0(0 0 0 0 0 )
185       for (int j = 0; j < n; j++){
186         ci0[j] =(double)0;
187
188       }
189     }
190
191
192
193     try{例外が発生する可能性のあるコード
194       //
195
196       solve_quadprog(G, g0,100,  CE, ce0,1, CI, ci0,100, alpha);
197     } catch (const std::exception& ex) {
198       std::cerr << "solve_quadprog_failed" << ex.what() << std::endl;
199       throw;
200     }
201
202
203     //を出力alpha
204     print_alpha(alpha,& alpha_max_number);重み
205     //出力weight
206
207     for(i=0;i<n;i++){
208       weight[0]+=alpha[i]*data[i].y*data[i].input_first;
209       weight[1]+=alpha[i]*data[i].y*data[i].input_second;
210
211
212     }
213     cout<<"weight[0]="<<weight[0]<<endl;
214     cout<<"weight[1]="<<weight[1]<<endl;
215
216
217     //(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）
218     //
219     //cout<<weight[0]<<"x+"<<weight[1]<<"y="<<theta<<endl;
220
221     if(argv[1]){
222       argv1=argv[1];
```

```
223     }else{
224       argv1="No Kernel";
225     }
226
227
228     //を出力するtheta
229     print_theta(argv1,alpha_max_number,data,weight,Kernel);
230
231     cout<<"f"<<endl;
232     for (int x0 = 0; x0 < 50; x0 ++){
233       for (int x1 = 0; x1 < 50; x1 ++){識別
234     //f
235         // f(argv1,weight,x0,x1,theta);
236       }
237     }
238
239 }
240 double norm(double x_i_0,double x_i_1,double x_j_0,double x_j_1){
241     return pow(x_i_0 -x_j_0,2.0)+ pow(x_i_1 -x_j_1,2.0);
242
243 }
244 double PolinomicalKernel(double x_i_0,double x_i_1,double x_j_0,double x_j
245     return pow((1.0+x_i_0*x_j_0+x_i_1*x_j_1),2.0);
246
247 }
248
249 double GaussianKernel(double x_i_0,double x_i_1,double x_j_0,double x_j_1)
250     return exp(-norm(x_i_0,x_i_1,x_j_0,x_j_1)/2.0/sigma/sigma);
251
252
253 }
254
255 int read_data(Dataset* data) {
256     int i=0;
257     ifstream ifs(FILENAME);
258     string str;
259     if(ifs.fail()) {
260       cerr << "File do not exist.\n";
261       exit(0);
262     }
263
264     while(getline(ifs, str)) {
265       if(i==100) break;
```

```
266       data[i].input_first=0; data[i].input_second=0; data[i].y=0;
267
268       sscanf(str.data(), "%lf %lf %lf", &data[i].input_first , &data[i].input
269
270       i++;
271     }
272
273
274
275     return 0;
276   }
277   void print_alpha(double alpha[MATRIX_DIM],int *alpha_max_number){
278     int i=0;出力
279     //
280     cout<<"alpha"<<endl;
281     for ( i = 0; i <DATA_NUM; i++){
282       // cout<<"alpha[i]:"<<alpha[i]<<"alpha[alpha_max_number]:"<<alpha[alph
283       if(alpha[i]*10>alpha[*alpha_max_number]*10){
284         (*alpha_max_number)=i;
285         // cout<<"alpha_max_number"<<*alpha_max_number<<endl;
286       }
287       printf("%d\t%f\n", i, alpha[i]);//は浮動小数点型で出力alpha
288
289     }
290
291
292   }
293
294   void print_theta(string argv1,int alpha_max_number,Dataset* data, double w
295
296
297
298
299     //はが最大の番号alpha_max_numberalpha
300      cout<<"alpha_max_number:"<<alpha_max_number<<endl;
301     if(argv1=="P"){
302
303       Kernel=PolinomicalKernel(weight[0],weight[1],data[alpha_max_number].in
304
305
306     }else if(argv1=="G"){
307
308       Kernel=GaussianKernel(weight[0],weight[1],data[alpha_max_number].input
```

```
309
310     }else{内積でカーネルトリックなし
311
312         //
313         Kernel=(weight[0]*data[alpha_max_number].input_first+weight[1]*data[al
314
315
316     }
317
318     double theta=Kernel-data[alpha_max_number].y;
319     cout<<"theta="<<theta<<endl;
320
321
322
323 }
324 void   f(string argv1,double weight[2],int x0,int x1,double theta){
325
326     double Kernel;
327    if(argv1=="P"){
328         Kernel=PolinomicalKernel(weight[0],weight[1],x0,x1);
329
330
331     }else if(argv1=="G"){
332
333       Kernel=GaussianKernel(weight[0],weight[1],x0,x1);
334
335     }else{内積でカーネルトリックなし
336
337         //
338         Kernel=(weight[0]*x0+weight[1]*x1);
339
340
341     }
342
343    if(Kernel*100<theta*100){
344       printf("%d\t%d\n",x0,x1);
345
346
347     }
348
349 }
```

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

sample_linear.dat を解析した場合、つぎのように出力される。

```
alpha
0 0.000000
1 0.000000
2 0.000000
3 -0.000000
4 -0.000000
5 0.250937
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 0.000000
21 0.000000
22 -0.000000
23 -0.000000
24 0.249687
25 0.000000
26 -0.000000
27 0.000000
28 0.000000
29 0.000000
30 -0.000000
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 0.001249
41 0.000000
42 0.000000
43 -0.000000
```

```
44 -0.000000
45 -0.000000
46 0.000000
47 -0.000000
48 -0.000000
49 0.000000
50 0.000000
51 -0.000000
52 -0.000000
53 -0.000000
54 -0.000000
55 0.000000
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 -0.000000
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 0.000000
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 0.000000
97 -0.000000
98 0.000000
99 -0.000000
weight[0]=0.530611
weight[1]=-0.469388
alpha_max_number5
theta=-4.04082


\end  {verbatim}
```

sample_circle.dat をガウスカーネルで解析した場合、つぎのように出力される。

```
\begin{verbatim}
```

```
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。次に関数について説明する。

- `double norm`
  - ４つの double 型の引数をとり、それは２つの点の座標を表す
  - 　２つの点の距離の二乗を返す
- `double PolinomicalKernel`
  - ４つの double 型の引数をとり、それは２つの点の座標を表す
  - 　多項式カーネルを計算する
- `int read_data`
  - 構造体を引数にとる
  - 　指定したファイルのデータをよみこんで構造体に代入して返す
- `void print_alpha`
  - alpha の配列と、alpha の配列数を引数にとる。
  - 　alpha の値を順番に出力する
- `void print_theta`
  - コマンドライン引数と、alpha の個数、構造体、重みを引数にとる
  - 　コマンドライン引数に応じてカーネルを選び theta を出力する

5 評価結果

- sample_linear.dat のデータを解析した評価結果
  - カーネルトリックなしの場合

```
weight[0]=0.530611
weight[1]=-0.469388
alpha_max_number:5
theta=-4.04082
```

  - 多項式カーネルの場合

```
weight[0]=0.371174
weight[1]=-0.363525
alpha_max_number:40
theta=4.89865
```

  - ガウスカーネルの場合

```
weight[0]=152.11
weight[1]=-121.204
alpha_max_number:5
theta=1
```

- sample_ciecle.dat のデータを解析した評価結果
  - カーネルトリックなしの場合

```
weight[0]=-0.00109041
weight[1]=0.0047479
alpha_max_number:20
theta=-0.976828
```

  - 多項式カーネルの場合

```
weight[0]=1.87848
weight[1]=1.73443
alpha_max_number:48
theta=7120.85
```

– ガウスカーネルの場合

```
weight[0]=12.9251
weight[1]=-177.845
alpha_max_number:48
theta=-1
```

6 考察

gnuplot の使い方に苦労したが、線形と円形のサンプルデータは適切に学習させることができた。課題 1 に時間がかかってしまった。