計算機科学実験及演習 4 (エージェント) レポート 1

佐竹誠

2018/10/12

1　プログラム概要

サポートベクターマシンによってデータ点が書かれたファイルを読み込み、識別器を出力します。

2　外部仕様

2.1　プログラム名

svm.cpp がプログラムファイルです。

2.2　ファイルの説明

2.2.1　svm.cpp

サポートベクターマシンのプログラムファイルです。

2.2.2　quadprog++.cc

二次計画問題を解くライブラリです。svm.cpp で使用しています。

2.2.3　quadprog++.hh

quadprog++.cc のヘッダーファイルです。svm.cpp でインクルードしています。

2.2.4　sample_linear.dat

線形識別可能なサンプルデータです。

2.2.5　sample_circle.dat

線形識別不可能なサンプルデータです。

2.2.6　Makefile

コンパイルするための make コマンドを使えるようにするファイルです。

2.3　入力方法

プログラムを実行すると、まず「データファイルを指定してください：」と表示されるので、使用するデータファイルを指定してください。ただしデータ数の上限は 100 個になっています。
その後「データの次元数を指定してください：」と表示されるので、使用するデータの次元数を指定してください。
次に「使用するカーネルを指定してください (0:カーネルトリックなし　1:多項式カーネル　2:ガウスカーネル)：」と表示されるので、使用したいカーネルを数字で指定してください。

2.4 出力

2.4.1 w

各ベクトルの重みを表します。次元の数だけ空白を挟んで出力されます。

2.4.2 alpha[]

各データの α ベクトルを表します。データの数だけ配列の形式で出力されます。

2.4.3 θ

閾値を表します。

2.4.4 計算結果画像

学習データが二次元のときのみ出力されます。データと識別機を xy 平面上に可視化したものです。

2.4.5

2.5 コンパイル方法

プログラムファイルがあるディレクトリで make コマンドを実行すると quadprog++.cc と svm.cpp のコンパイルが行われ、svm という実行可能ファイルができます。実行後は画面の指示に従って条件を入力してください。

2.6 実行例

学習データを sample_linear、次元数を 2、カーネルトリックなしで学習させた結果は次のようになります。出力される画像は図 1 です。

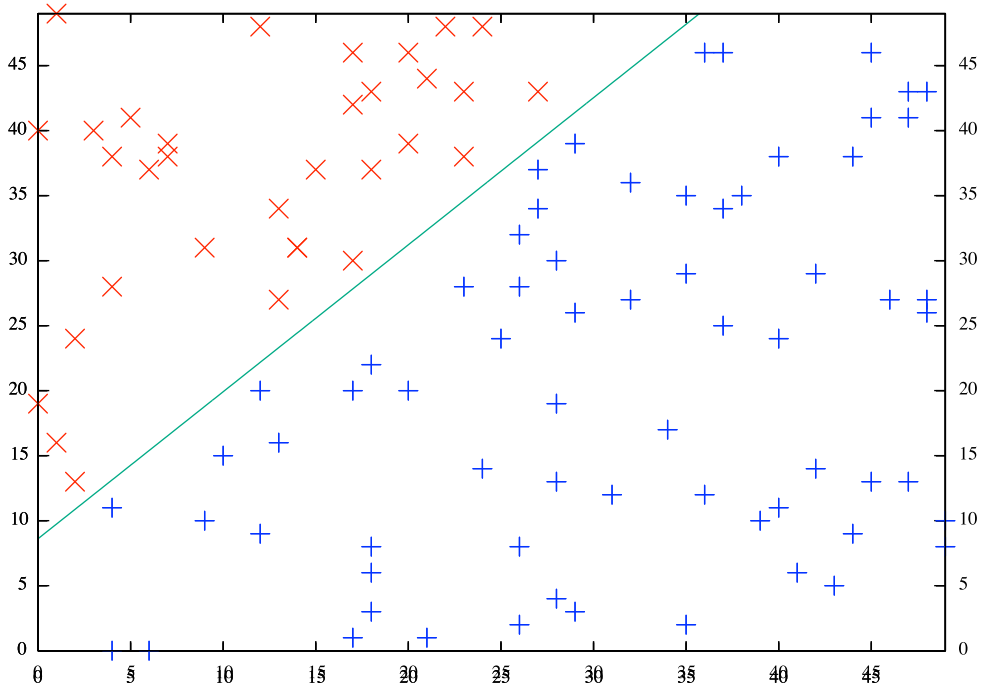

図 1　sample_liner をカーネルトリックなしで計算させた場合の出力画像

$ ./svm
データファイルを指定してください：sample_linear.dat
データの次元数を指定してください：2
使用するカーネルを指定してください (0:カーネルトリックなし　1:多項式カーネル　2:ガウスカーネル)：0
alpha[0]：-9.19286e-08
alpha[1]：-2.01237e-07
alpha[2]：9.94292e-11
alpha[3]：2.58381e-10
(途中省略)
alpha[97]：-4.5659e-07
alpha[98]：1.05053e-07
alpha[99]：6.20515e-07
w: 0.530678 -0.469417
θ：-4.04106

2.7　エラー処理

2.7.1　指定されたデータファイルがないとき

全ての入力が終わった時点で”ファイルが見つかりません”と表示し、プログラムを終了します。

2.7.2　指定されたカーネルが正しくないとき

全ての入力が終わった時点で”正しくカーネルを指定してください (0:カーネルトリックなし　1:多項式カーネル　2:ガウスカーネル)”と表示し、プログラムを終了します。

3　内部仕様

3.1　主要な大域変数の説明

3.1.1　double G[][]

二次計画問題を解く際に使用するデータを格納する double 二次元配列です。

3.1.2　double g0[]

二次計画問題を解く際に使用する double 配列です。関数の一階微分を表しており、本プログラムでは全ての要素に-1 が格納されています。

3.1.3　double CE[][]

二次計画問題を解く際に使用する double 二次元配列です。本プログラムではラベルの配列を設定しています。

3.1.4　double ce0[]

二次計画問題を解く際に使用する double 配列です。本プログラムでは全て 0 に設定しています。

3.1.5　double CI[][]

二次計画問題を解く際に使用する double 二次元配列です。本プログラムでは単位行列に設定しています。

3.1.6　double ci0[]

二次計画問題を解く際に使用する double 配列です。本プログラムでは全て 0 に設定しています。

3.1.7　double alpha[]

二次計画問題の計算結果として出力される α ベクトルを格納する double 配列です。

3.1.8　int n

二次計画問題を解く際に使用する int 変数です。本プログラムではデータの数が設定されています。

3.1.9 int m

二次計画問題を解く際に使用する int 変数です。CE と ce0 の配列の要素数を指定しており、本プログラムでは n と同じ数に設定しています。

3.1.10 int p

二次計画問題を解く際に使用する int 変数です。CI と ci0 の要素数を指定しており、本プログラムでは 1 に設定しています。

3.1.11 int N

データの次元数を表す int 型の変数です。入力から受け取った値をそのまま格納します。

3.1.12 int kernel

使用するカーネルを表す int 型の変数です。0,1,2 の三値しか取らず、それぞれカーネルトリックなし、多項式カーネル、ガウスカーネルを表します。入力から受け取った値をそのまま格納します。

3.1.13 std::string file_name

使用するデータファイルの名前を表す string 型の変数です。入力から受け取った値をそのまま格納します。

3.1.14 double sigma

ガウスカーネルを使用する際に使用する double 型の変数です。本プログラムでは 10 に設定しています。

3.1.15 double maxX, maxY, minX, minY

データが二次元の時のみ使用し、データの x 要素,y 要素それぞれの最大値最小値を格納する変数です。画像を出力する際の表示サイズの決定に使用します。初期値には maxX と maxY には小さな数を、minX と minY には大きな数を設定しています。

3.1.16 double data[][]

学習データの生のデータを格納する double 配列です。

3.1.17 double label[]

学習データのラベルを格納する double 配列です。

3.1.18 int index

学習データの数を格納する int 型の配列です。

3.1.19 double theta

得られた識別器の閾値を格納する double 型の変数です。

3.2 各関数の説明

3.2.1 double kernel_result(double* x, double* y, int kernel, int degree, double sigma)

カーネルの計算結果を得るための関数です。計算結果を double 型で返します。各引数の意味は以下の通りです。

- double* x,y
  計算する二つの配列の引数です。
- int kernel
  使用するカーネルを表す引数です。(0:内積 1:多項式カーネル 2:ガウスカーネル)
- int degree
  データの次元数を表す引数です。
- double sigma
  ガウスカーネル計算時に使用するシグマ定数を表す引数です。

3.2.2 double get_norm(double* x, double* y, int degree)

二乗ノルムを得るための関数です。計算結果を double 型で返します。各引数の意味は以下の通りです。

- double* x,y
  計算する二つの配列の引数です。
- int degree
  データの次元数を表す引数です。

4 評価結果

正しく識別器が得られていることを出力された画像を提示して示します。図 2, 図 3, 図 4 がそれぞれのカーネルを使用して計算させたときの出力画像です。おおよそ正しく識別できていることがわかります。

5 考察

5.1 識別

sample_linear をカーネルトリックなしで識別させることはうまくできています。

sample_circle を多項式カーネルで識別させることは概ねできているようです。しかし境界付近の識別がかなり曖昧で、マージン最大化を実現できているとはいえません。

sample_circle をガウスカーネルで識別させることはうまくできています。

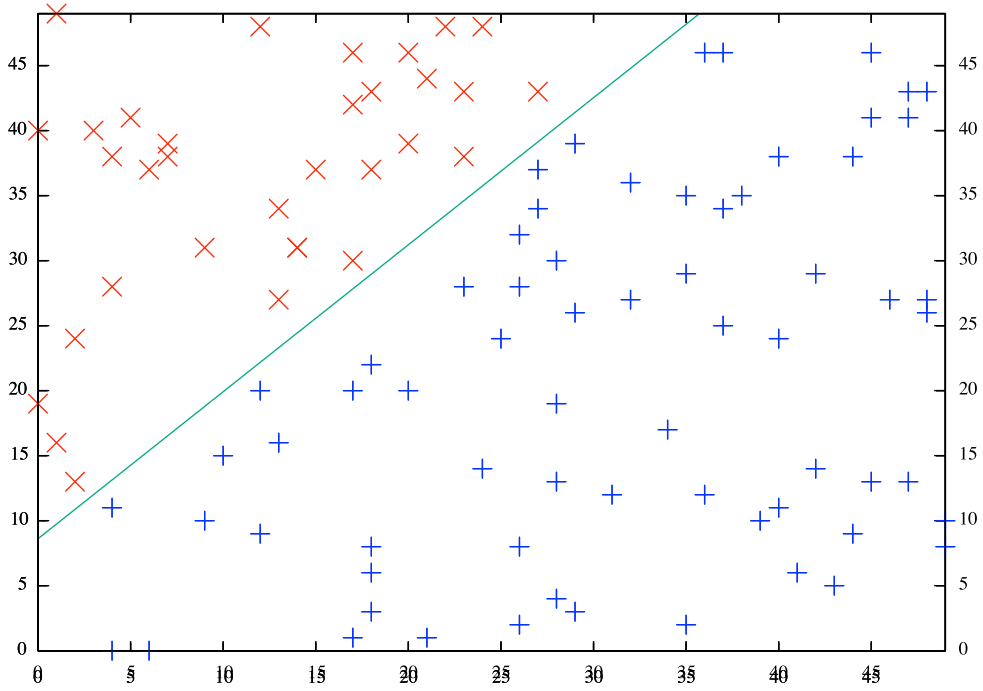

図 2　sample_liner をカーネルトリックなしで計算させた場合の出力画像

5.2　実行時間

多項式カーネルとガウスカーネルの違いのひとつに計算量があり、この部分では多項式カーネルが勝ると考え、プログラムの実行時間を計測してみました。データはどちらも同じ sample_circle を用い、二次計画問題を計算する時間を計測したところ、多項式カーネルは 20ms、ガウスカーネルは 22ms かかっていることがわかりました。二次元のデータ数 100 という非常に少ない数で 2ms という差が生まれていることからも、多項式カーネルはやはり速いカーネルだと言えます。

5.3　シグマ定数

本プログラムでは課題に設定されている通りシグマ定数を 10 で計算していましたが、このハイパーパラメータを変化させると実際に識別器が変化するのか気になり実験してみました。理論上はシグマ定数を小さくすると境界が入り組んで過学習に近くなり、大きくするとより直線的になるはずです。結果は図 5,6 のようになりました。それぞれシグマ定数は 5,30 に設定しました。

シグマ定数を小さくしたときは理論通りデータ点に影響されすぎて過学習気味になり、大きくした時はマージンが細くなってしまっていることがわかりました。したがってシグマ定数は今回のデータだと 10 くらいがちょうどよい数値だと言えます。

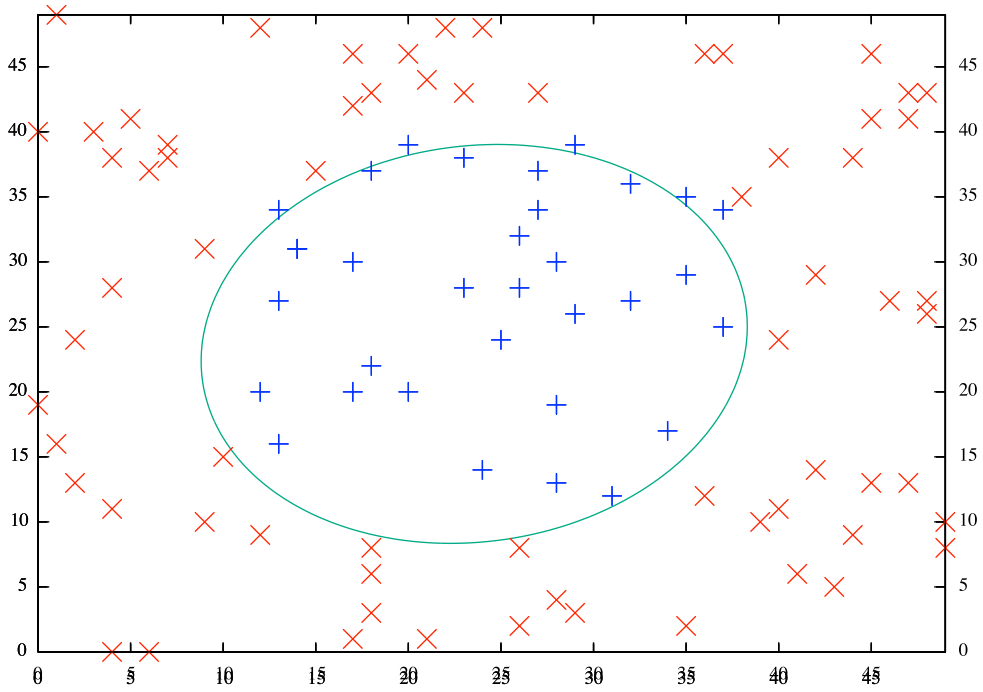

図 3　sample_circle を多項式カーネルで計算させた場合の出力画像

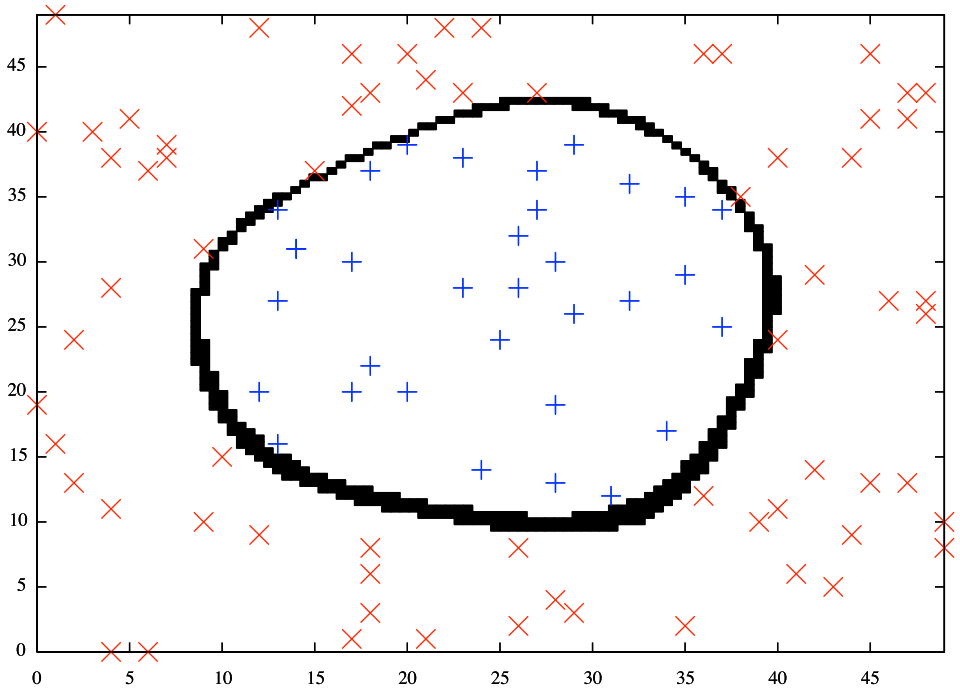

図 4　sample_circle をガウスカーネルで計算させた場合の出力画像

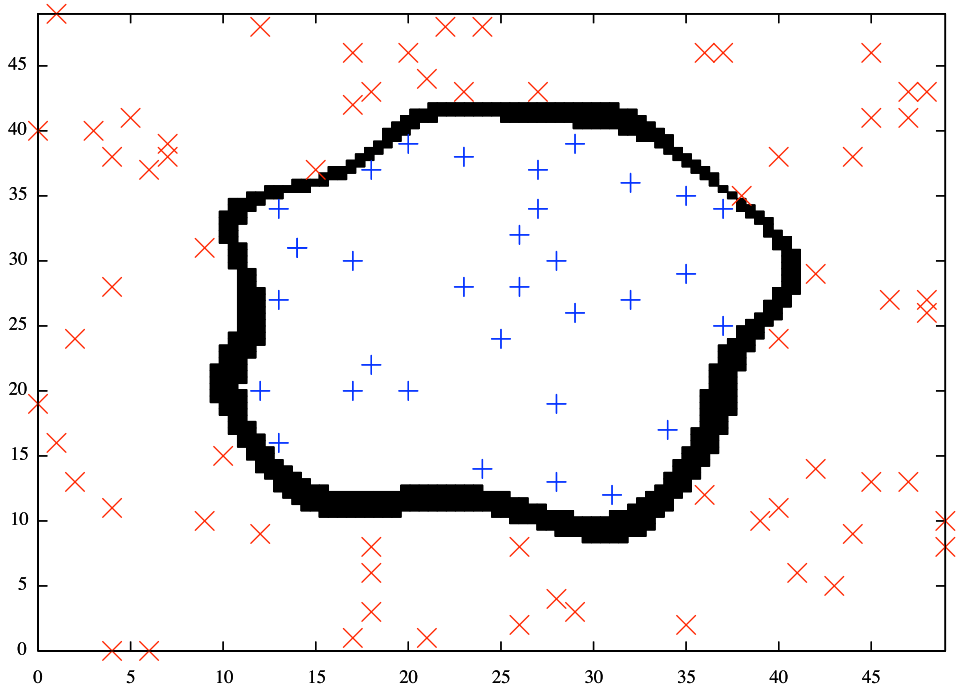

図 5　シグマ定数を 5 にしたときの識別器を可視化したもの

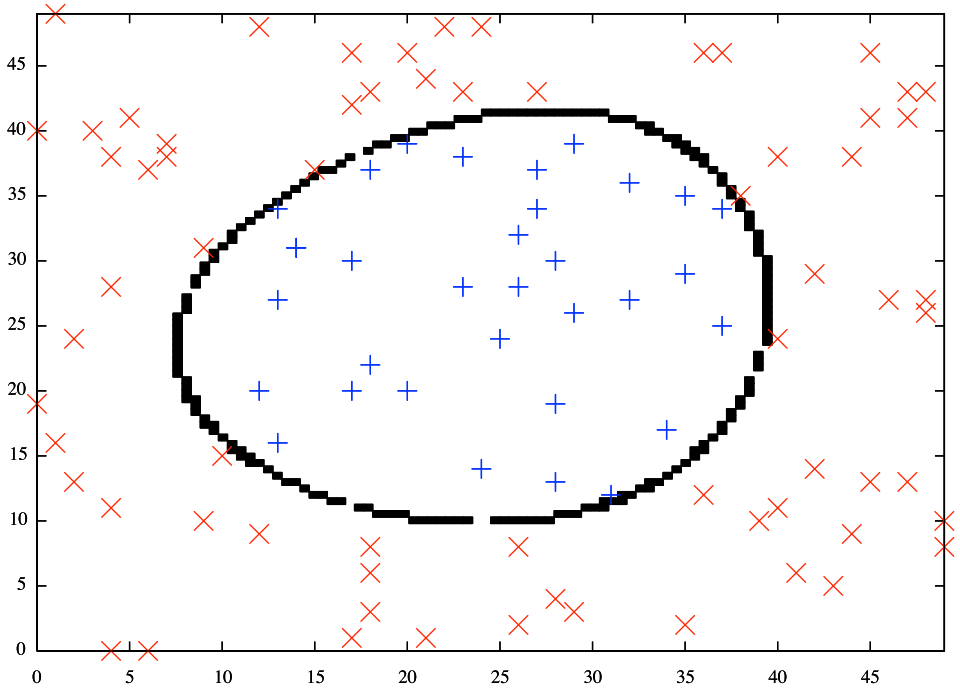

図 6　シグマ定数を 30 にしたときの識別器を可視化したもの