TDA231 - Algorithms for Machine Learning & Inference

Chalmers University of Technology

LP3– 2017-02-27

By:
Karabo Ikaneng, 891024-8239, kara.ikaneng@gmail.com
Elias Svensson, 920406-3052, elias.svensson.1992@gmail.com

Goal

Support Vector Machines

1 Theoretical problems

1.1 SVM, 5 points

The hyperplane expressed and weights (w) and the bias (b)

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{1}$$

Where the weights of the respective two classes w_1 and w_2 are equal to 1 and b equals -3. Therefore:

$$x_1 + x_2 - 3 = 0$$

The distance from the hyperplane to an arbitarty point (x_1 and x_2) is given as:

$$\gamma = \frac{|w_1 x_1 + w_2 x_2 + b|}{\sqrt{w_1^2 + w_2^2}} \tag{2}$$

Applying this equation to the support vectors (4 data points closest to the hyperplane) gives $\gamma = \frac{1}{\sqrt{2}}$. Therefore the margin can be given as $2\gamma = \sqrt{2}$.

Code representation of calculation:

```
%% 1.1)
clc;
clear all;
close all;

% Define points
xPositive = [2,2;4,4;4,0];
xNegative = [0,0;2,0;0,2];
yPositive = ones(size(xPositive,1),1);
yNegative = -1*ones(size(xNegative,1),1);
X = [xPositive; xNegative];
Y = [yPositive; yNegative];

% Calculate bias, beta, and margin
Mdl = fitcsvm(X,Y);
b = Mdl.Bias;
w = Mdl.Beta;
m = margin(Mdl,X,Y)/norm(w);

```

```
% Plot
hold on;
scatter(xPositive(:,1),xPositive(:,2),'r+');
scatter(xNegative(:,1),xNegative(:,2),'b.');
fplot(@(x1) -(w(1,1)*x1+b)/w(2,1));
plot([xPositive(1,1),(-b)/2],[xPositive(1,2),(-b)/2],'m:');
axis([(min(X(:,1))-1) (max(X(:,1))+1) (min(X(:,2))-1) (max(X
    (:,2))+1)]);
legend('1','-1','Boundary');
xlabel('x1');
ylabel('x2');
title(sprintf('Hyperplane: %d*x1+%d*x2+(%d)=0',w(1,1),w(2,1),b))
    ;
hold off;

% Save figure
print('11figure','-dpng')
```

1.2 SVM continued, 5 points

a

The primal formulation is of the form:

$$min_{w,b} = \frac{1}{2}||w||^2 \tag{3}$$

$$y_i(w^T x_i + b) \geq 1 \ \forall i \tag{4}$$

For the specific case we have:

$$min_{w,b} = \frac{1}{2}(\sqrt{w_1^2 + w_2^2})^2 = \frac{1}{2}(w_1^2 + w_2^2)$$

Consider the training points (X) and responses (Y) given in array below. This data is substituted into equation 4 and the results of the matrix multiplication is given below.

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \begin{array}{c} \begin{array}{cc} X1 & X2 \end{array} \\ \begin{pmatrix} 2 & 2 \\ 4 & 4 \\ 4 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 2 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{array} \begin{array}{c} \begin{array}{c} Y \end{array} \\ \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \end{array}$$

$$x_1, y_1 : 2w_1 + 2w_2 + b \geq 1$$

$$x_2, y_2 : 4w_1 + 4w_2 + b \geq 1$$

$$x_3, y_3 : 4w_1 + b \geq 1$$

$$x_4, y_4 : b \geq -1$$

$$x_5, y_5 : 2w_1 + b \geq -1$$

$$x_6, y_6 : 2w_2 + b \geq -1$$

b

Solving this Quadratic optimization problem yields the results gained in Problem 1.1, where the weights for each class was equal to 1 and b = -3.
The results are were calculated using the quadprog function in Matlab. quadprog(H,f,A,b), where:

H = eye(3);
len = size(X,1);
f = zeros(3,1);
b = -ones(len,1);
A = -diag(Y)*[X, ones(len,1)];

```
%% 1.2)b)
clc;
clear all;
close all;

% Define points
xPositive = [2,2;4,4;4,0];
xNegative = [0,0;2,0;0,2];
yPositive = ones(size(xPositive,1),1);
yNegative = -1*ones(size(xNegative,1),1);
X = [xPositive;xNegative];
Y = [yPositive;yNegative];

% Find a minimum for problem
l = size(X,1);
H = eye(3);
f = zeros(3,1);
A = -diag(Y)*[X, ones(l,1)];
b = -ones(l,1);
min = quadprog(H,f,A,b)
```

min = []

c

The support vector dual formation has the form:

$$max_\alpha = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i x_j) \tag{5}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \ \ and \ \ \alpha_i \geq 0 \ \ \forall i \tag{6}$$

In this particular case we have i= 1,...,6; X and Y:

$$\begin{array}{c} \begin{matrix} X1 & X2 \end{matrix} \\ \begin{pmatrix} 2 & 2 \\ 4 & 4 \\ 4 & 0 \\ 0 & 0 \\ 2 & 0 \\ 0 & 2 \end{pmatrix} \end{array}$$

$$\begin{array}{c} \begin{matrix} Y \end{matrix} \\ \begin{matrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{matrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \end{array}$$

d

This dual formulation can be expressed as a quadratic optimization problem and solved with quadprog in Matlab as before. The solution is calculated with a more extended expression for quadprog with more input arguments. quadprog(H,f,A,c,Aeq,ceq, LB,UB)

```
%% 1.2)d)
clc;
clear all;
close all;

% Define points
xPositive = [2,2;4,4;4,0];
xNegative = [0,0;2,0;0,2];
yPositive = ones(size(xPositive,1),1);
yNegative = -1*ones(size(xNegative,1),1);
X = [xPositive;xNegative];
Y = [yPositive;yNegative];
```

```
13
14 % Find a minimum for problem
15 H = (Y*Y').*(X*X');
16 f = -ones(1,length(H));
17 A = zeros(1,length(H));
18 c = 0;
19 Aeq = Y';
20 ceq = 0;
21 lb = zeros(length(H),1);
22 ub = inf*ones(length(H),1);
23 alpha = quadprog(H,f,A,c,Aeq,ceq,lb,ub);
24 w = X'*(alpha.*Y)
25 b = 1-(X(1,:)*w)
```

The output of the quadprog gives the weights $w = [w_1, w_2]^T = [1, 1]^T$ and b = -3. As a comment on the support vectors, when determining the dual formulation we do not need to consider all the points in the data set, but only the the support vectors. This is because the hyperplane is not dependent on points beyond the margin.

2 Practical Problems

2.1 SVM, 5 points]

a

Boxconstraint = 1.

b

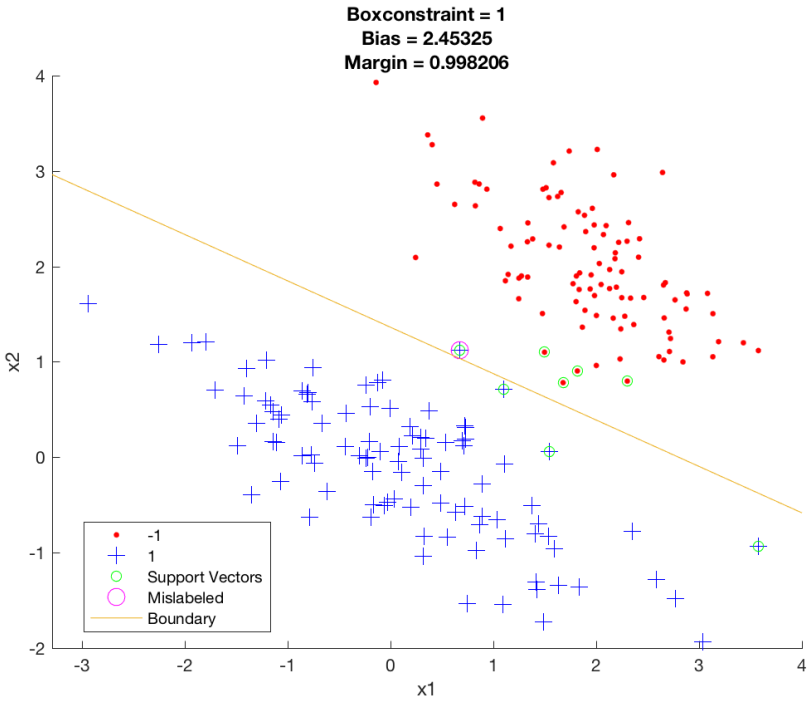

Figure 1: Soft-margin SVM (with Linear kernel), d1b.mat

c

Bias = 2.45

d

Margin = 0.998

2.2 Kernels, 5 points

a

Boxconstraint = 1.

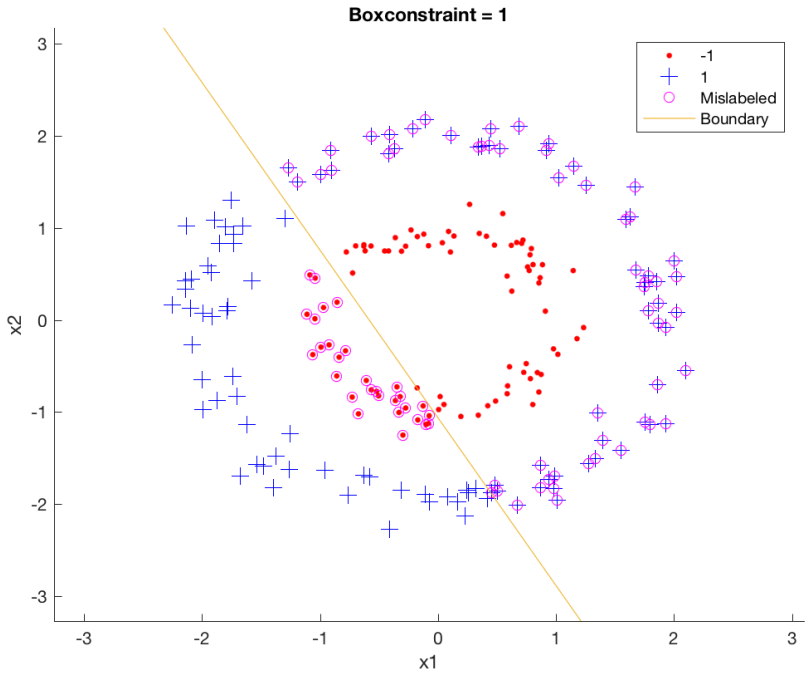

Figure 2: Soft-margin SVM (with Linear kernel), d2.mat

b

Command window output:

```
Kernel: linear, Method: smo
Elapsed time is 5.826053 seconds.
Error rate: 0.445

Kernel: linear, Method: qp
Elapsed time is 7.697023 seconds.
Error rate: 0.515

Kernel: quadratic, Method: smo
Elapsed time is 4.236103 seconds.
Error rate: 0

Kernel: quadratic, Method: qp
Elapsed time is 19.376386 seconds.
Error rate: 0
```

```
Kernel: rbf, Method: smo
Elapsed time is 2.288212 seconds.
Error rate: 0

Kernel: rbf, Method: qp
Elapsed time is 15.371009 seconds.
Error rate: 0
```

The Linear kernel fails rather miserably to classify, with either the SMO or QP optimization methods. This is because it tries to separate two concentric circles of datapoints with a straight line, which is impossible. The Quadratic and RBF kernel both classifies all data correctly with either optimization method. However, the choice optimization influences the running time of the algorithms. The SMO optimization method turns out to be faster than the QP method for all kernels. The Quadratic kernel turns out to be faster than the RBF kernel. To summarize, the SMO optimization method and the Quadratic kernel has the fastest running time. (Note: Average Classification Accuracy = 1 - Error Rate.)

c

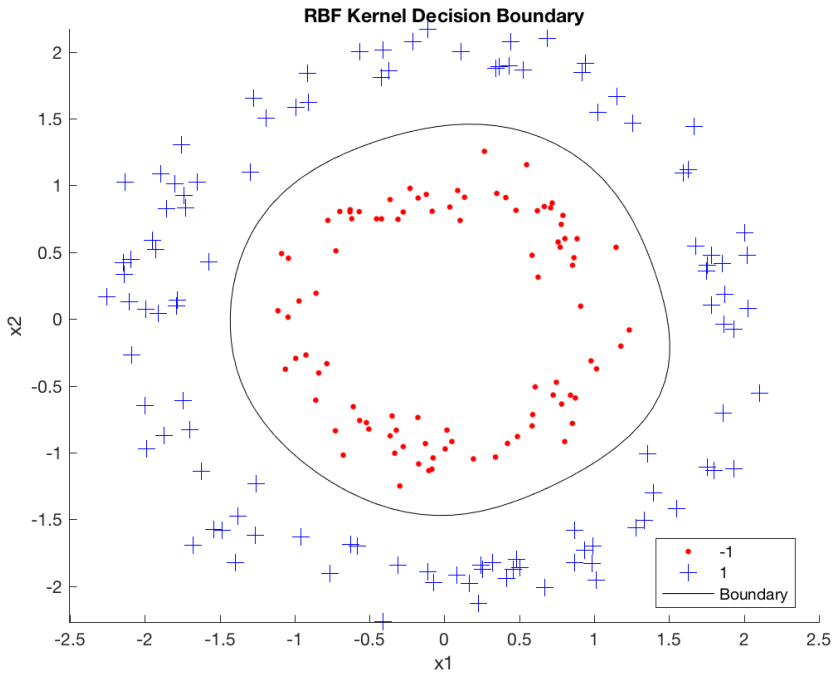

Figure 3: RBF kernel