Table of Contents

ECG703 HW6, Support Vector Machine - James Skelly

```
close all
```

HW6. Due April 29, 2021

Derive a SVM solution for the dataset below. You must show all the details and intermediate steps. Exceptions to use library functions are matrix operations and Quadratic Programming. After finding the solution, plot the results and mark the margin as well as the support vectors.

```
% Class 1, Red Data Points
x1 = [-2;1;2;4;5;7;8;9;6];
y1 = [9;5;8;6;9;2;7;4;4];
c1 = 1;
C1Data = [x1,y1];

% Class 2, Blue Data Points
x2 = [-2;-4;-4;-5;-6;-7;-7;-8;-9];
y2 = [2;2;4;3;7;1;5;8;7];
c2 = -1;
C2Data = [x2,y2];

% Obtain size of class 1 and class 2 data for matrix indexing
[nRowsC1, nColsC1] = size(x1);
[nRowsC2, nColsC2] = size(x2);

% Compute number of data points and number of dimensions
nDataPoints = nRowsC1 + nRowsC2;
nDimensions = nColsC1 + nColsC2;
```

Solution with MATLAB's Quadratic Programmer

After getting the necessary matrices into standard form, a quadratic programmer can be used to find the final weights of the boundary line. From these weights and the calculated margin, the support vectors can be obtained using basic geometry.

```
% Get Q matrix for the standard form of QP
```

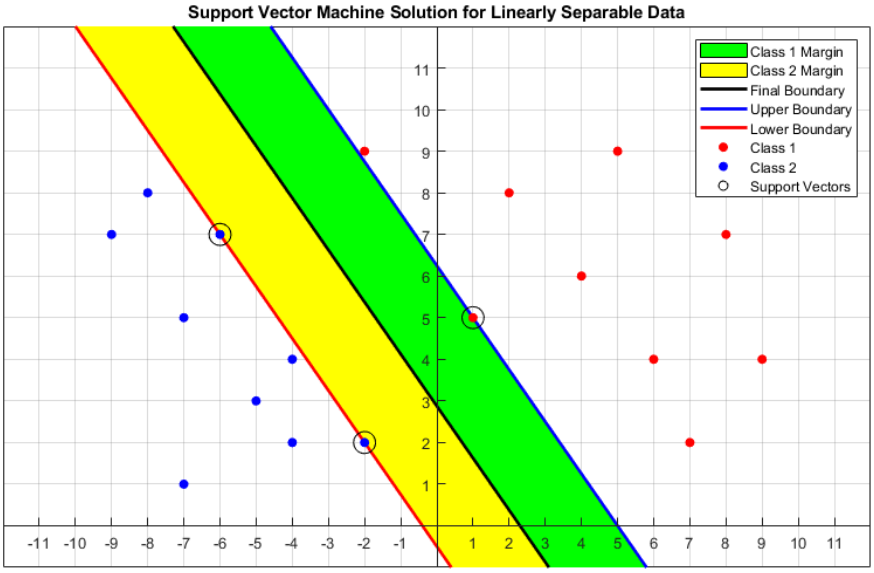
```
Q = eye(nDimensions + 1);
Q(1,1) = 0;

% Build p, c vectors for standard form of QP
p = zeros(nDimensions + 1,1);
c = ones(nDataPoints,1);

% Negate all components of c for standard form
c = c*-1;

% Allocate memory for the A matrix
A = zeros(nDataPoints,nDimensions + 1);

% Populate the A matrix for standard form of QP
for i = 1:nDataPoints

   if i <= (nDataPoints/2)
       A(i,1) = c1;
       A(i,2) = x1(i)*c1;
       A(i,3) = y1(i)*c1;
   else
       A(i,1) = c2;
       A(i,2) = x2(i-(nDataPoints/2))*c2;
       A(i,3) = y2(i-(nDataPoints/2))*c2;
   end

end

% Negate all components of A for standard form
A = A * -1;

% Obtain final boundary line weights using QP
wf = quadprog(Q,p,A,c);

% Create a separate vector excluding the bias (wo)
wf2 = [wf(3),wf(2)];

% Compute the margin
margin = 1/norm(wf2);

% Compute the offset for the support vectors from the boundary line
theta1 = rad2deg(atan(wf2(2)/wf2(1)));
theta2 = 90 - theta1;
dy = margin/(sin(deg2rad(theta2)));
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-
decreasing in
feasible directions, to within the default value of the optimality
 tolerance,
and constraints are satisfied to within the default value of the
 constraint tolerance.
```

Plotting

```
xplot2 = -30:0.05:30;
yplot2 = (-1*(wf(2)/wf(3))*xplot2)-(wf(1)/wf(3));

% Plot filler between support vectors and boundary line setup
xp2 = [xplot2, fliplr(xplot2)];
inBetween = [yplot2, fliplr(yplot2+dy)];
inBetween2 = [yplot2 - dy, fliplr(yplot2)];

% Initialize support vector matrix, point counter, offset for doubles
SV0 = zeros(1,2);
ptCounter = 1;
del = 0.0001;

% Find support vectors
for i = 1:nRowsC1

    if  (C1Data(i,2)+del) == round(((-1*(wf(2)/wf(3))*C1Data(i,1))-
(wf(1)/wf(3)))+ dy + del,4)
        SV0(ptCounter,1) = C1Data(i,1);
        SV0(ptCounter,2) = C1Data(i,2);
        ptCounter = ptCounter + 1;
    end

    if  (C2Data(i,2)+del) == round(((-1*(wf(2)/wf(3))*C2Data(i,1))-
(wf(1)/wf(3)))- dy + del,4)
        SV0(ptCounter,1) = C2Data(i,1);
        SV0(ptCounter,2) = C2Data(i,2);
        ptCounter = ptCounter + 1;
    end

end

figure('Name', 'ECG703 HW6 Support Vector Machine Solution');
set(gcf,'units','normalized','position',[0.314,0.191,0.499,0.525]);
p4 = fill(xp2, inBetween, 'g');
hold on
p5 = fill(xp2, inBetween2, 'y');
p1 = plot(xplot2, yplot2, 'black', 'LineWidth', 2);
p2 = plot(xplot2, yplot2 + dy, 'blue', 'LineWidth',2);
p3 = plot(xplot2, yplot2 - dy, 'red', 'LineWidth',2);
title('Support Vector Machine Solution for Linearly Separable Data')
xlim([-12 12])
ylim([-1 12])
xticks(-11:1:11)
yticks(-1:11)
ax = gca;
ax.XAxisLocation = 'origin';
```

```
ax.YAxisLocation = 'origin';
grid on
p6 = scatter(x1,y1,'red', 'filled');
p7 = scatter(x2,y2,'blue', 'filled');
p8 = scatter(SV0(:,1),SV0(:,2),200, 'black');


legend('Class 1 Margin', 'Class 2 Margin', 'Final Boundary',...
'Upper Boundary', 'Lower Boundary', 'Class 1', 'Class 2', 'Support
 Vectors')
```



Results and Concluding Remarks

The Support Vector Machine is a perfect application to use a quadratic programmer. Since MATLAB has a built in quadratic programmer, it is not difficult to solve the SVM using the **quadprog** function and the correct matrices/vectors in standard form. The final plot shows the boundary line, the support vectors, all of the data points, and the margins shaded.

```
fprintf('\n')
fprintf('Final Weights of Boundary Line:\n')
disp(wf)
fprintf('\n')

fprintf('\n')
fprintf('Margin:\n')
disp(margin)
fprintf('\n')

fprintf('\n')
fprintf('Support Vectors:\n')
disp(SV0)
```

```
fprintf('\n')
```

```
Final Weights of Boundary Line:
   -0.8519
    0.3704
    0.2963



Margin:
    2.1083



Support Vectors:
    -2     2
     1     5
    -6     7
```

Published with MATLAB® R2018b