Table of Contents

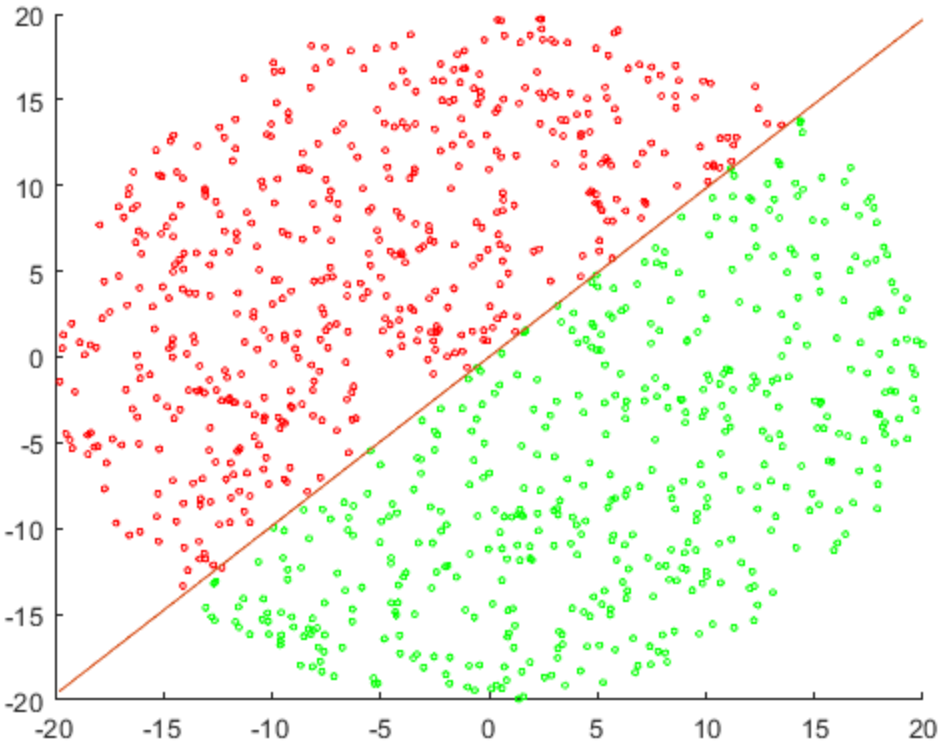

```
clc; clear;
```

Part 1

Implement softsvm. In order to test, I made linear points which classifies points based on a line. So it is fully linearly seperable. We see that it works (yay!). The colors are the different classes, and the line is our approximate decision boundary.

```
[X, l] = linearPoints(1,0);  % make points with boundary y=x

[w, b, xi] = softsvm(X,l, 0.005);

% plot the results
figure();
hold on;
scatter(X(1,:), X(2,:), 5, l);  % plot the points
plot(X(1,:), -w(1)/w(2) * X(1,:));
colormap(gca,'prism')
hold off;

clear; % get rid of testing junk
```

```
Minimum found that satisfies the constraints.

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



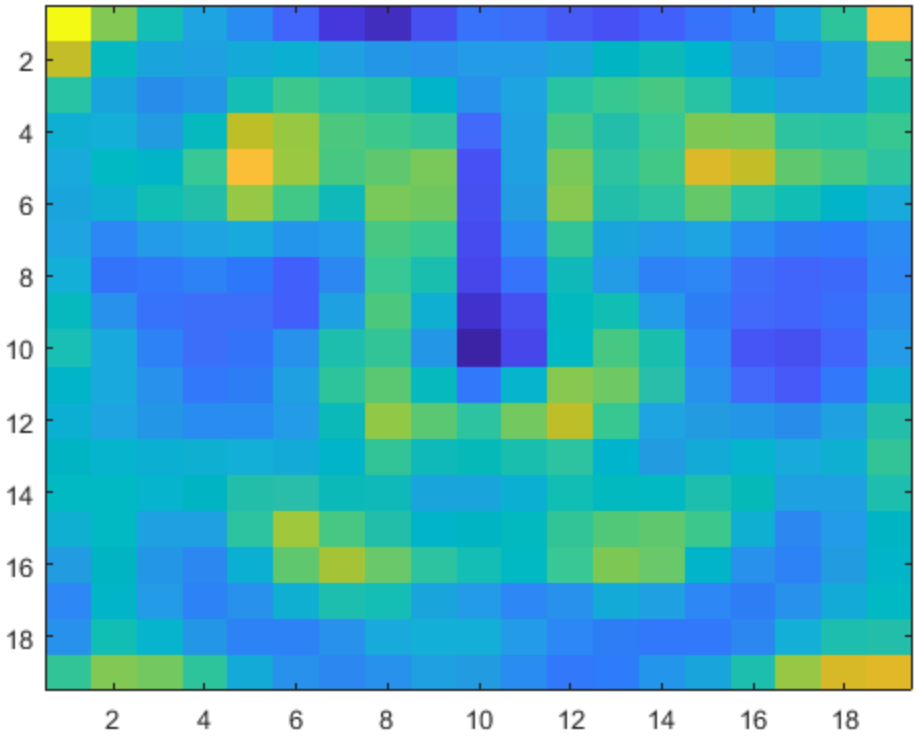
Part 2

Load the CBCL dataset and apply the soft SVM classifier with a penalty $\gamma = 0.005$. Generate and turn in a visualization of w, as found by the SVM function, using the command imagesc(reshape(w, dims)) (here dims comes from the original data file). What does this picture of w represent? How do you interpret it?
****

It looks like a face :0 Here, this picture of w represents the weights of each pixel of the face. It could be interpreted as how important each part of the face is in the classification.

```
load('cbcl1.mat')
[w, b, xi] = softsvm(X,L, 0.005);

figure();
imagesc(reshape(w, dims))
```

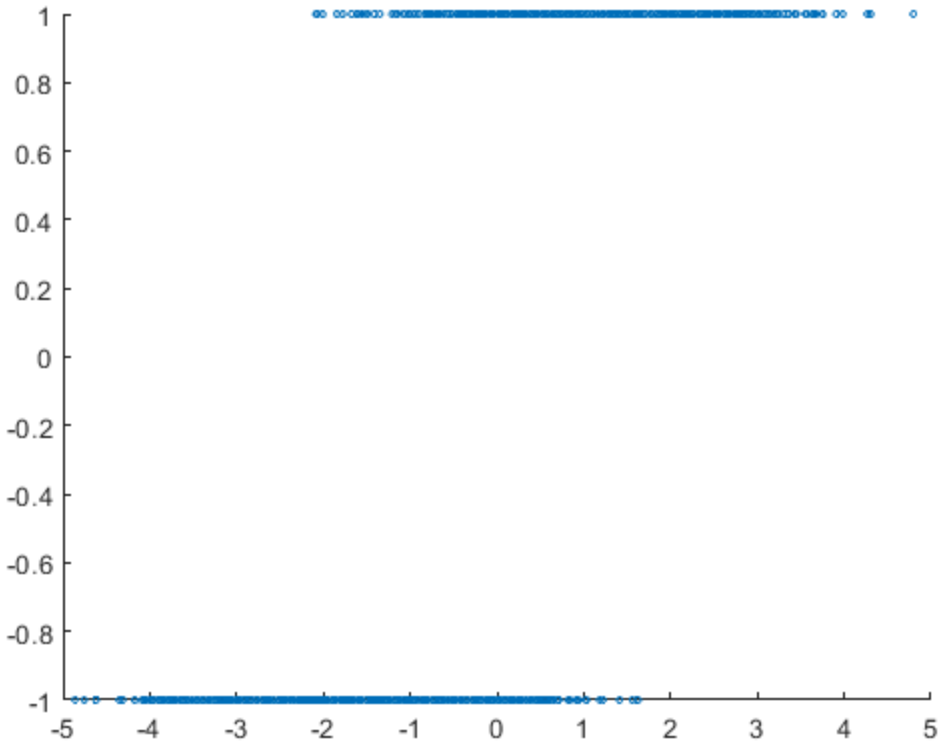
```
Minimum found that satisfies the constraints.

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



Part 3

Generate a plot of X'*w + b (overlay with and compare to the plot of L, the correct labels). What do the extremes (minimum/maximum) of this plot represent? Were any data points classified incorrectly, and how can you tell?
****

```
preds = X' * w + b;
figure();
hold on;
scatter(preds, L, 5);  % plot the points
colormap(gca,'prism')
hold off;
```



Part 4

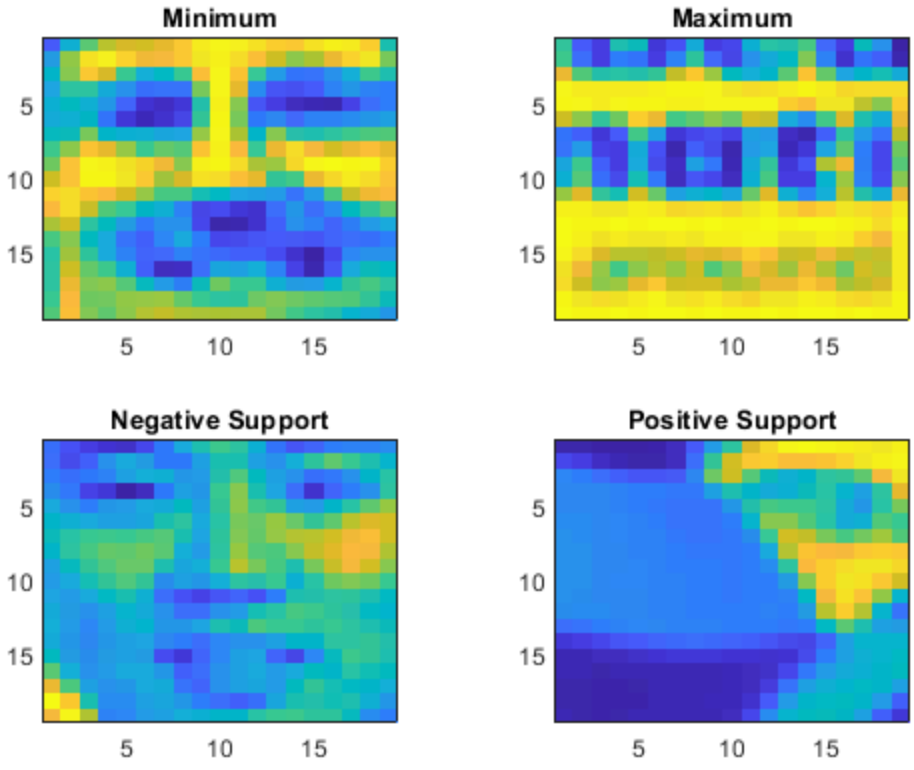
How can we determine that a data point was a support vector?
****
If its value is on the 'wrong' side of zero or if it is very close to zero (being the closest to zero of any of the points without being negative).

Part 5

Turn in two images corresponding to the extreme points of this plot (most positive/negative scoring column of X shown as images), and two more images corresponding to example support vectors from each class. Discuss what you observe!
****
The ones labeled as -1 look very much like faces. This is quite clear. It is interesting that the support looks most like a face. I would have expected the minimum to look most like a face, and the support to be something that is harder to distinguish. The positive examples just look like noise to me. I have honestly no idea what it is.

```
[~, min_idx] = min(preds);
[~, max_idx] = max(preds);

idx = 1:length(preds);
pos_supports = idx(and((preds < 0), (L > 0)));
pos_sup_idx = randsample(pos_supports, 1);
```

```
neg_supports = idx(and((preds > 0), (L < 0)));
neg_sup_idx = randsample(neg_supports, 1);

figure()
tiledlayout(2, 2);
% Minimum
y = X(:, min_idx);
nexttile;
imagesc(reshape(y, dims))
title('Minimum')
% max
y = X(:, max_idx);
nexttile;
imagesc(reshape(y, dims))
title('Maximum')
% negative support
y = X(:, neg_sup_idx);
nexttile;
imagesc(reshape(y, dims))
title('Negative Support')
% positive support
y = X(:, pos_sup_idx);
nexttile;
imagesc(reshape(y, dims))
title('Positive Support')
```



Part 6

Load the 20 Newsgroups data set and apply the soft SVM with $\gamma = 0.005$
****

```
clear; % get rid of CBCL stuff
load('news.mat');
[w, b, xi] = softsvm(X,L, 0.005);
```

```
Minimum found that satisfies the constraints.

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

Part 7

By examination of the vector w, which words are the most important for separating the two classes of documents? Which words are most distinctly space-related? What about cryptography-related? Give at least five important words for each case.
****

The most important words are the ones with the most extreme values of w. So we give the top (10) words with the most extreme values of w for each category and display it below.

```
clc;
idx = (1:length(w))';
A = [w, idx];
B = sortrows(A, 1);

n = 10;
top_negative = B(1:n, 2);
top_positive = B(end-n+1:end, 2);

disp("Top Positive (space):");
disp(dict(top_positive, :));

fprintf("\n\n");

disp("Top Negative (cryptography):");
disp(dict(top_negative, :));
```

```
Top Positive (space):
dc

nasa

sky
```

launch

prb

science

orbit

pat

moon

space

Top Negative (cryptography):
clipper

encryption

key

chip

steve

security

code

your

na

nsa

softsvm.m

```
%SOFTSVM    Learns an approximately separating hyperplane for the
 provided data.
% [w, b, xi] = softsvm( X, l, gamma )
%
% Input:
% X : D x N matrix of data points
% l : N x 1 vector with class labels (+/- 1)
% gamma : scalar slack variable penalty
%
% Output:
% w : D x 1 vector normal to the separating hyperplane
% b : scalar offset
```

```
% xi : N x 1 vector of slack variables
%
% classify data using sign( X'*w + b )

function [w, b, xi] = softsvm( X, l, gamma )

[D,N] = size(X);

% construct H, f, A, b, and lb


gamma = 0.005;
% Quadratic Objective
H = spdiags([zeros(N,1); ones(D,1); 0] , 0, N + D + 1, N + D + 1);
% Linear Objective
f =[gamma * ones(N, 1); zeros(D,1); 0];  % gamma N times, then D+1
 zeros for right shape
% Linear Inequality Constraints Ax <= b
L = spdiags(l, 0, N, N );
A = -1 * [speye(N), L*X', l];
b = -1 * ones(N, 1);
% Lower bounds
lb = [zeros(N, 1); -Inf(D+1,1)];

% Solve
x = quadprog( H, f, A, b, [], [], lb );

% distribute components of x into w, b, and xi:
xi = x(1:N);
w = x(N+1:N+D);
b = x(end);
end
```

linearPoints.m

```
function [X,l] = linearPoints(m,b)
    n = 1000;
    R = 20;
    x0 = 0; % Center of the circle in the x direction.
    y0 = 0; % Center of the circle in the y direction.
    % Now create the set of points.
    t = 2*pi*rand(n,1);
    r = R*sqrt(rand(n,1));
    x = x0 + r.*cos(t);
    y = y0 + r.*sin(t);

    X = [x';y'];
    l = 2 * (m * x > y) - 1;
end
```

imgrid1.m

```
% IMage GRID generator
%   Given a bunch of little images, this generates a single figure
 which shows
%  a lot of them.
% Input:
%   I - dims(1)*dims(2)-by-N matrix with image data for each of N
 images.
%   dims - 2D vector with the height and width of each image.
 Reshaping the columns of
%    I to this size should produce each image.
%   gridsz - The size of the grid to show (for example, [2,3] will
 show six images in total
%    laid out in a grid with two rows).
% Output:
%   None (creates a figure).

function [] = imgrid(I, dims, gridsz)
    if dims(1) * dims(2) ~= size(I,1)
        error('Columns of I must have the same number of elements as
 given by dims.');
    end

    M = zeros(dims(1) * gridsz(1), dims(2) * gridsz(2));
    k = 1;
    for i = 1:gridsz(1)
        for j = 1:gridsz(2)
            is = (1 + (i-1)*dims(1)):((i)*dims(1));
            js = (1 + (j-1)*dims(2)):((j)*dims(2));
            M(is, js) = reshape(I(:,k), dims);
            k = k + 1;
            if k > size(I,2)
                break;
            end
        end
        if k > size(I,2)
         break;
        end
    end
    imagesc(M);
    colormap gray;
    set(gcf,'Color','w');
    set(gca,'XTickLabel',[]);
    set(gca,'YTickLabel',[]);
end
```

Published with MATLAB® R2021a