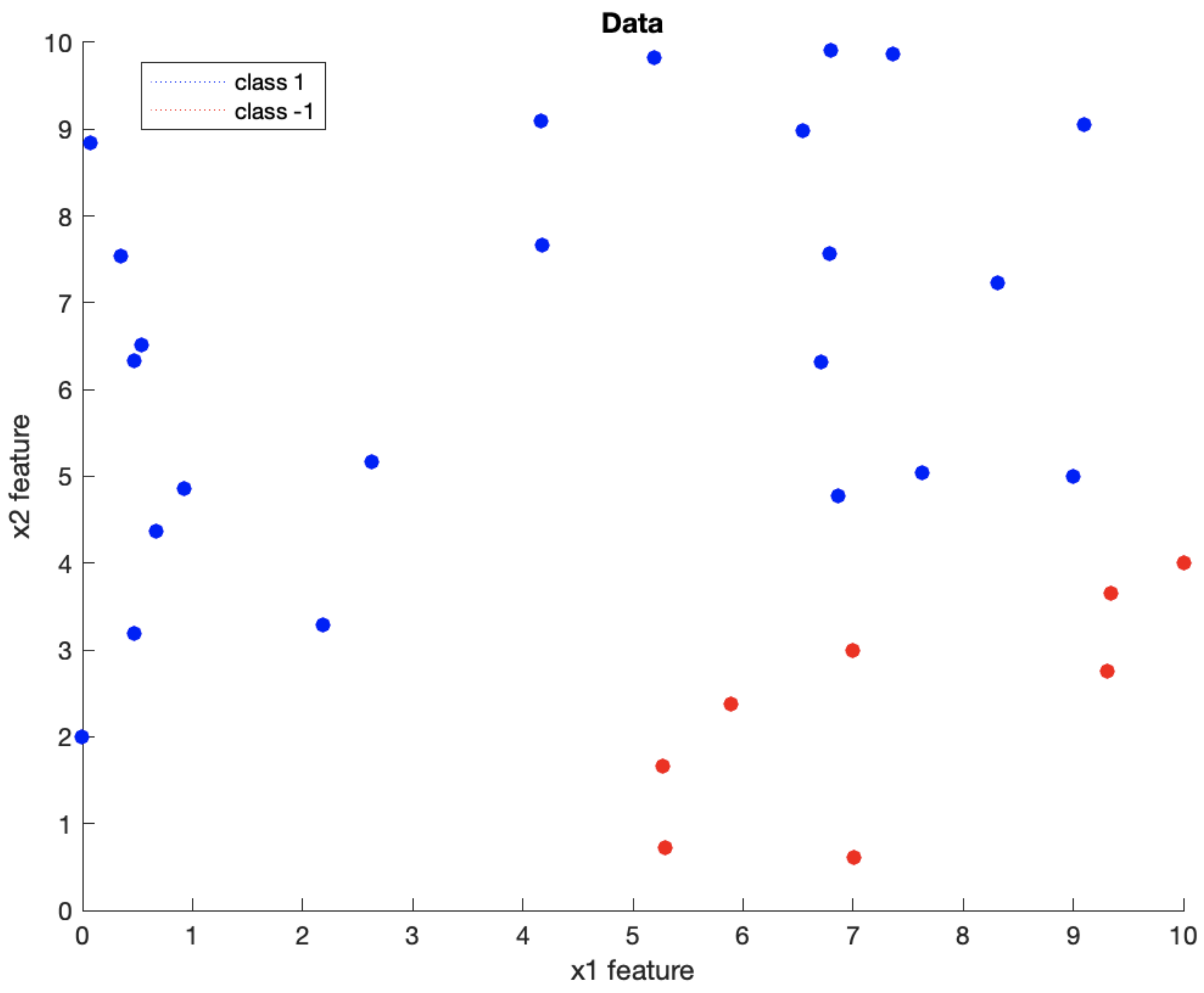
Zack Berger

Question 5

Part A - Visualization

By inspection of the following plot, the data is linearly separable…

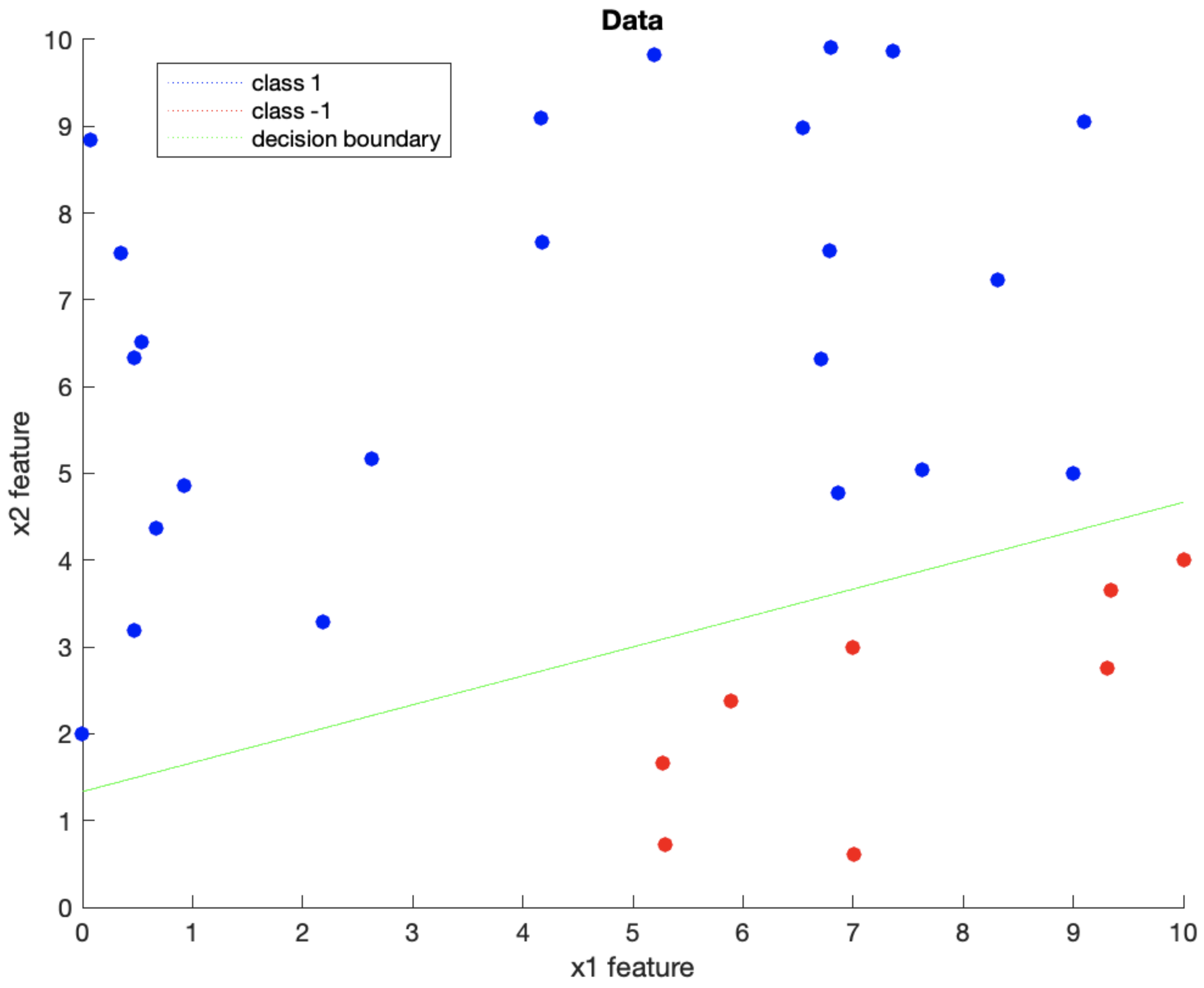


Part B

Using CVX to solve the primal problem for SVM, I found:

w = [-0.5, 1.5]
b = -2.0

The following shows a plot of the hyperplane defined by w and b overlaid with the data…



Part C

Suppose there are N data points that have feature vectors of dimension M.

As evidenced by the attached scratchwork that investigates the case of N = 3, the latter part of W(a) in the dual problem can be decomposed as (½)$a^T Pa$ where for the matrix P we have

$$P(i,j) = Y_i Y_j \langle X_i, X_j \rangle$$

Making this substitution into MATLAB then applying CVX to solve the dual problem of SVM, I found the following non-zero a's and their corresponding support vectors:

a_{28} = 0.853301
Support vector: [7.000000, 3.000000]

a_{29} = 0.396669
Support vector: [10.000000, 4.000000]

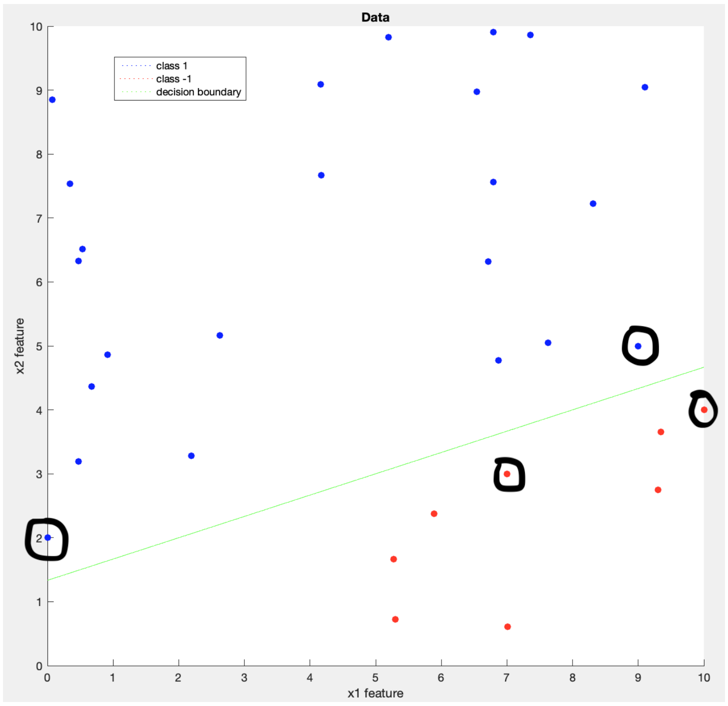
a_{30} = 0.201101
Support vector: [0.000000, 2.000000]

a_{31} = 1.048869
Support vector: [9.000000, 5.000000]

I used these values to calculate w and b, which are reported as follows:

w = [-0.5, 1.5]
b = -2.0

There are 4 support vectors, circled on the following graph…



Code Attachment

```
% Import data
data = readtable('/Users/zackberger/Desktop/ML/HW/HW_4/Q5/Data.csv');
data = data{:,:};


% Separate data into column vectors
X = data(:,[1,2]);
Y = data(:,3);

x1 = data(:,1);
x2 = data(:,2);

[num_data, data_dim] = size(X);


% Optimize primal SVM problem with CVX
cvx_begin

  variable w_primal(2);
  variable b_primal;

  minimize( norm(w_primal) );

  subject to
    Y.*(X*w_primal + b_primal) >= 1;

cvx_end

w_primal;
b_primal;


% Calculate the matrix P, where P(i,j) = YiYj<Xi,Xj>
P = zeros(num_data,num_data);
for i = 1:num_data
  for j = 1:num_data
    P(i,j) = X(i,:)*X(j,:)'*Y(i)*Y(j);
  end
end

one_vector = ones(num_data, 1);

% Optimize dual SVM problem with CVX
cvx_begin

  variable alpha_dual(num_data);

  minimize( (1/2)*alpha_dual.'*P*alpha_dual - (one_vector.'*alpha_dual) );
```

```
    subject to
        alpha_dual >= 0;
        Y.'*alpha_dual == 0;
cvx_end

% Loop through alpha_dual and for non-zero alpha, print corresponding support vector
num_support_vectors = 0;
for i = 1:num_data
    if alpha_dual(i) > .0001
        fprintf('alpha %i = %f\n', i, alpha_dual(i));
        fprintf('Support vector: [%f, %f]\n\n', X(i,1), X(i,2));
        num_support_vectors = num_support_vectors + 1;
    end
end

% Use alpha_dual to compute w and b
w_dual = [0, 0];
for i = 1:num_data
    if alpha_dual(i) > .0001
        w_dual = w_dual + alpha_dual(i)*X(i,:)*Y(i);
    end
end

b_dual = 0
for i = 1:num_data
    if alpha_dual(i) > .0001
        b_dual = b_dual + (1/num_support_vectors) * (Y(i) - w_dual*X(i,:).');
    end
end

w_dual
b_dual


% Graph results!
hold on

% Scatter plot feature vectors
for i = 1: numel(Y)
    if Y(i) == 1
        scatter(x1(i), x2(i), 'filled', 'b')
    else
        scatter(x1(i), x2(i), 'filled', 'r')
    end
end

% Plot primal decision boundary (w1*x1 + w2*x2 + b = 0)
% x1_axis = 0:1/100:10;
% x2_axis = (-b_primal - w_primal(1)*x1_axis) / w_primal(2);
% plot(x1_axis, x2_axis, 'g');

% Plot dual decision boundary (w1*x1 + w2*x2 + b = 0)
x1_axis = 0:1/100:10;
x2_axis = (-b_dual - w_dual(1)*x1_axis) / w_dual(2);
plot(x1_axis, x2_axis, 'g');

% Create graph details
title("Data");
xlabel("x1 feature");
ylabel("x2 feature");

L(1) = plot(nan, nan, 'b:');
L(2) = plot(nan, nan, 'r:');
L(3) = plot(nan, nan, 'g:');
legend(L, {'class 1', 'class -1', 'decision boundary'})

hold off
```