```
% IDS/ACM/CS 158: Fundamentals of Statistical Learning
% PS5, Problem 3: Support Vector Machine
% Author: Michael Li, mlli@caltech.edu
%--------------------------------------------------------------------------
clear;

D = readmatrix('dataset9.csv');
X = D(:, 1:end-1);
ys = D(:,end);
g_plus = D(ys == 1, 1:end-1);
g_minus = D(ys == -1, 1:end-1);

% C = 0
C = 0;
N = size(D,1);
p = size(D(1,1:end-1), 2);
K = ones(size(ys*transpose(ys)));

% generate kernel matrix and find lambdas
for i=1:length(X)
    for j=1:length(X)
        K(i, j) = gaussKernel(X(i,:), X(j,:));
    end
end

H = (ys*transpose(ys)) .* K;
svm = quadprog(H, -1*ones(1,N), zeros(1,N), 0, transpose(ys), 0,
 zeros(N,1), (1/C)*ones(N,1));

% find support vectors
support_vecs = D(abs(svm) > 10^-5, :);
support_plus = support_vecs(support_vecs(:,3)==1, 1:end-1);
support_minus = support_vecs(support_vecs(:,3)==-1, 1:end-1);
nonzero_lambs = svm(abs(svm) > 10^-5, :);
lambs_plus = nonzero_lambs(support_vecs(:,3)==1);
lambs_minus = nonzero_lambs(support_vecs(:,3)==-1);

% find beta0
b0_max = 0;
for i=1:length(support_plus)
    tot = 0;
    for j=1:length(support_plus)
        tot = tot + lambs_plus(j) * gaussKernel(support_plus(j,:),
 support_plus(i,:));
    end

    if tot > b0_max
        b0_max = tot;
    end
end

b0_min = 10^100;
```

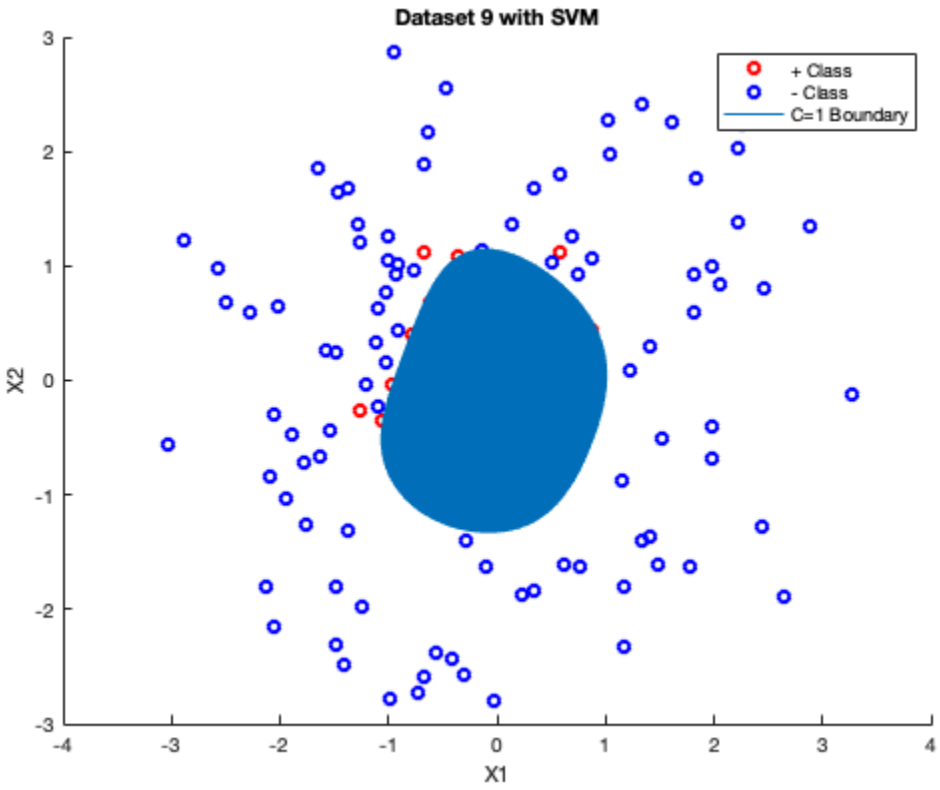
```
for i=1:length(support_minus)
    tot = 0;
    for j=1:length(support_minus)
        tot = tot + -lambs_minus(j) * gaussKernel(support_minus(j,:),
 support_minus(i,:));
    end

    if tot < b0_max
        b0_min = tot;
    end
end

b0 = -1/2*(b0_max + b0_min);

fprintf("\nNumber of Support Vectors C=0 are %i\n", size(support_vecs,
 1))
% For C=0, there are 30 Support Vectors

% get points for boundary
x1 = -2:.004:2;
x2 = -2:.004:2;
boundary_x_0 = [];
boundary_y_0 = [];

% test each point
for i=x1
    for j=x2
        tot = b0;
        for k=1:length(support_vecs)
            tot = tot + support_vecs(k,3) * nonzero_lambs(k) *
 gaussKernel(support_vecs(k, 1:end-1), [i j]);
        end

        if abs(tot) < .04
            boundary_x_0 = [boundary_x_0 i];
            boundary_y_0 = [boundary_y_0 j];
        end
    end
end

% C=1
C = 1;
N = size(D,1);
p = size(D(1,1:end-1), 2);
K = ones(size(ys*transpose(ys)));

% generate kernel matrix and find lambdas
for i=1:length(X)
    for j=1:length(X)
        K(i, j) = gaussKernel(X(i,:), X(j,:));
    end
end

H = (ys*transpose(ys)) .* K;
```

```
svm = quadprog(H, -1*ones(1,N), zeros(1,N), 0, transpose(ys), 0,
 zeros(N,1), (1/C)*ones(N,1));

% find support vectors
support_vecs = D(abs(svm) > 10^-5, :);
support_plus = support_vecs(support_vecs(:,3)==1, 1:end-1);
support_minus = support_vecs(support_vecs(:,3)==-1, 1:end-1);
nonzero_lambs = svm(abs(svm) > 10^-5, :);
lambs_plus = nonzero_lambs(support_vecs(:,3)==1);
lambs_minus = nonzero_lambs(support_vecs(:,3)==-1);

% find beta0
b0_max = 0;
for i=1:length(support_plus)
    tot = 0;
    for j=1:length(support_plus)
        tot = tot + lambs_plus(j) * gaussKernel(support_plus(j,:),
 support_plus(i,:));
    end

    if tot > b0_max
        b0_max = tot;
    end
end

b0_min = 10^100;
for i=1:length(support_minus)
    tot = 0;
    for j=1:length(support_minus)
        tot = tot + -lambs_minus(j) * gaussKernel(support_minus(j,:),
 support_minus(i,:));
    end

    if tot < b0_max
        b0_min = tot;
    end
end

b0 = -1/2*(b0_max + b0_min);

fprintf("\nNumber of Support Vectors C=1 are %i\n", size(support_vecs,
 1))
% For C=1, there are 61 Support Vectors

% get points for boundary
x1 = -2:.004:2;
x2 = -2:.004:2;
boundary_x = [];
boundary_y = [];

% test each point
for i=x1
    for j=x2
        tot = b0;
```

```
        for k=1:length(support_vecs)
            tot = tot + support_vecs(k,3) * nonzero_lambs(k) *
 gaussKernel(support_vecs(k, 1:end-1), [i j]);
        end

        if abs(tot) < .04
            boundary_x = [boundary_x i];
            boundary_y = [boundary_y j];
        end
    end
end

% plot
figure
hold on
plot(g_plus(:,1), g_plus(:, 2), 'or')
plot(g_minus(:,1), g_minus(:, 2), 'ob')
plot(boundary_x_0, boundary_y_0)
plot(boundary_x, boundary_y)
legend('+ Class', '- Class', 'C=1 Boundary', 'C=0 Boundary')
title('Dataset 9 with SVM')
xlabel('X1')
ylabel('X2')

function res = gaussKernel(x, y)
    res = exp(-(norm(x-y))^2);
end

% Clearly there's something wrong with my code. I'm not sure if the
 lambdas
% are just incorrect or if i'm calculating the decision boundaries
% incorrectly, but I think if I were to guess, that using the decision
% boundary for C=1 makes more sense because the training data is
% definitely not all encompassing. C=0 overfits the data for sure and
 would
% not generalize well to other cases not seen in this data.
```

```
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.


Number of Support Vectors C=0 are 30

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.


Number of Support Vectors C=1 are 61
Warning: Ignoring extra legend entries.
```



Published with MATLAB® R2019a