SVM and Perceptron HW:

Table of Contents

q3 - svm_test and svm_train:

```
function [accuaracy] = svm_test( theta, X_test, y_test)

             predicted_y = theta' * X_test;
             comparison_y = sign(predicted_y) - y_test;
             accuaracy = sum(comparison_y ~= 0) / length(y_test);

end

function [theta] = svm_train( X, y )

d = size(X,2);
H = eye(d);
f = zeros(d,1);
Dy = diag(y);
A = Dy*X;
b = ones(size(y));

theta = quadprog(H, f, -A, -b);

end
```

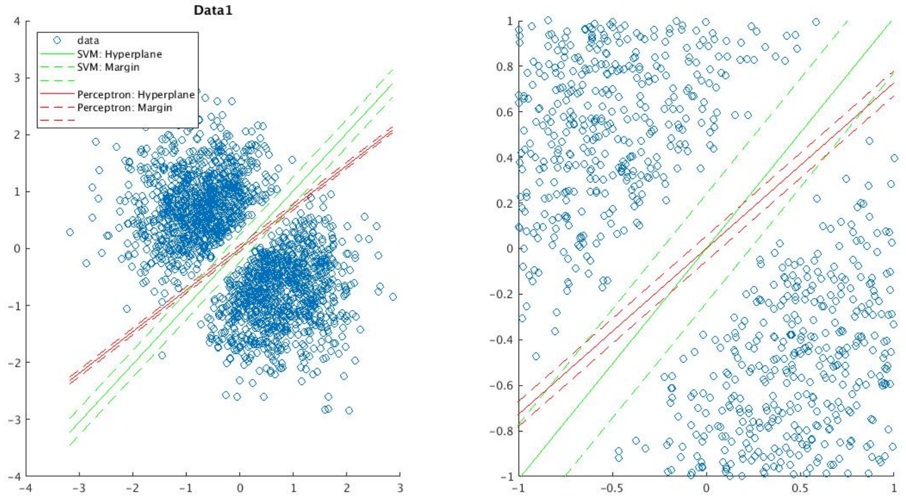
q4-5 - Visualization of HyperPlane and Margin:

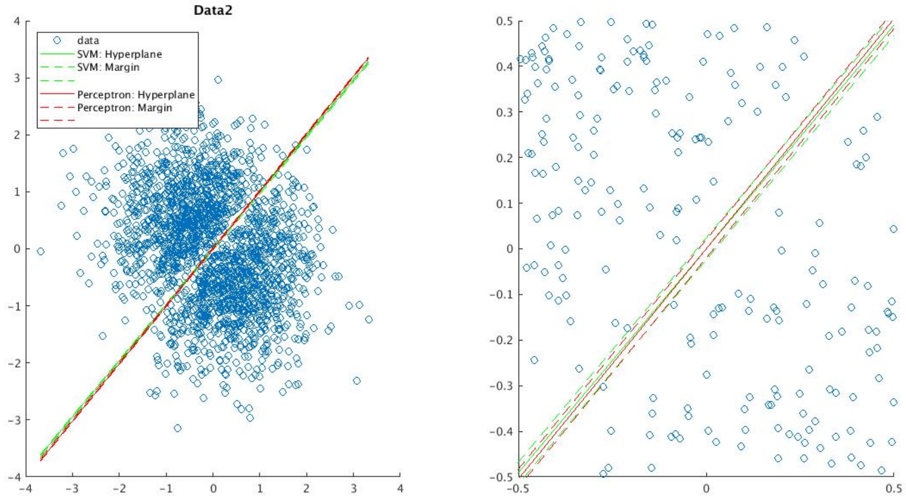
For both Perceptron and SVM.

```
clc; clear; close all;
addpath([genpath('./materials') ...
    genpath('libsvm/libsvm-3.22/matlab')])

load data1.mat
[normX, sDev, means] = data_normalization(X);

[perc_theta, k] = my_perceptron_train(normX, y);
svm_theta = svm_train(normX, y);

figure('Name', 'Data1')
set(gcf, 'Position', [300, 300, 1300, 650]);
```

```
subplot(1,2,1)
title('Data1'); hold on;
plot_data_svm_percep(normX, svm_theta, perc_theta);
subplot(1,2,2)
plot_data_svm_percep(normX, svm_theta, perc_theta);
legend('off');
xlim([-1 1]); ylim([-1 1]);

load data2.mat
[normX, sDev, means] = data_normalization(X);

[perc_theta, k] = my_perceptron_train(normX, y);
svm_theta = svm_train(normX, y);

figure('Name', 'Data2')
set(gcf, 'Position', [300, 300, 1300, 650]);
subplot(1,2,1)
title('Data2'); hold on;
plot_data_svm_percep(normX, svm_theta, perc_theta);
subplot(1,2,2)
plot_data_svm_percep(normX, svm_theta, perc_theta);
legend('off');
xlim([-0.5 0.5]); ylim([-0.5 0.5]);
```

```
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.




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.
```



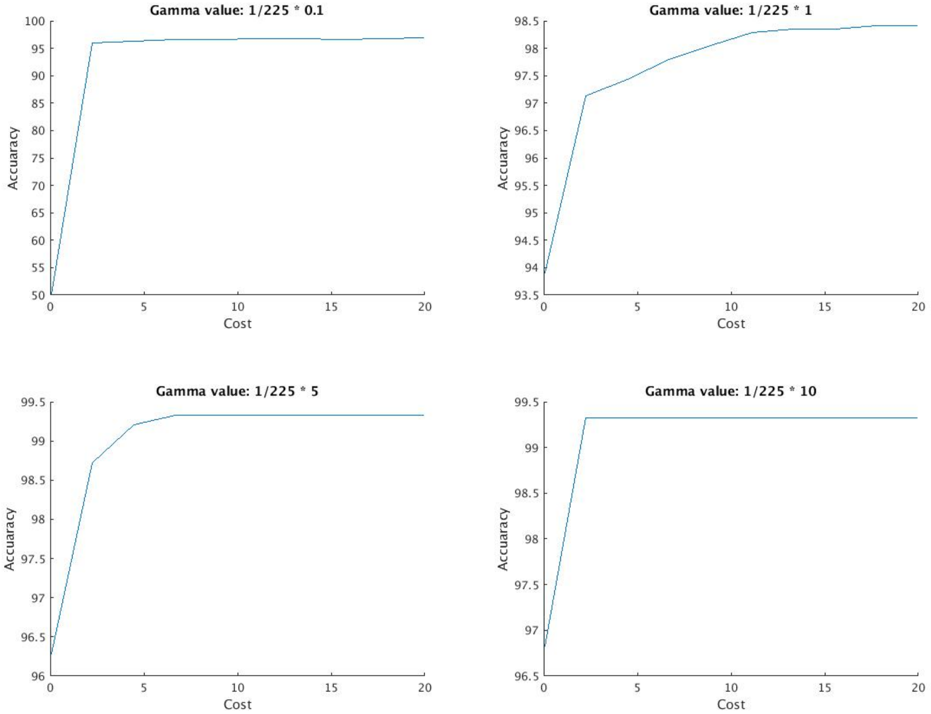


q6

```
img2 = imread('test2.jpg');
digit2 = image2blocks(img2);
img3 = imread('test3.jpg');
digit3 = image2blocks(img3);

X = [digit2; digit3];
y = [ones(size(digit2,1),1) ; -1 * ones(size(digit3,1),1)];

p = cvpartition( y, 'Holdout', 0.20);
X_train = X(p.training, :);
y_train = y(p.training);
X_test = X(p.test, :);
y_test = y(p.test);
```

```
[X_train_n, X_test_n] = normalizeTrainAndTest(X_train, X_test);

svmStruct = svmtrain(y_train, X_train_n, '-q');
[predicted_labels, accuracy, dec_values] = svmpredict(y_train,
 X_train_n, svmStruct,'-q');

fprintf('\nThe accuracy for the training data is: %.4g%%\n',
 accuracy(1))
fprintf('Its not 100%% because data is not linear seprable.\n')

[predicted_labels, accuracy, dec_values] = svmpredict(y_test,
 X_test_n, svmStruct,'-q');
fprintf('The accuracy for the test data is: %.4g%%\n', accuracy(1))


fprintf('\n***** GAMMA C PLAYGROUND *******\n')

def_c = 1;
def_gamma = 1/size(X,2);
gamma_coef = [1/10 1 5 10];
c_vals = linspace(def_c/20, def_c*20, 10);

min_error = [0 0 0];   % min_error = [error-value gamma-value cost-
value]

ACC = zeros(size(c_vals));
figure('Name', 'SVM Accuracy to cost');
set(gcf, 'Position', [200, 200, 1300, 1300]);
for j = 1:length(gamma_coef)
    for i = 1:length(c_vals)
        svmoptions = ['-c ', num2str(c_vals(i)), ' -g ',
 num2str(gamma_coef(j) * def_gamma) , ' -q'];
        svmStruct = svmtrain(y_train, X_train_n, svmoptions);
        [predicted_labels, accuracy, dec_values] = svmpredict(y_train,
 X_train_n, svmStruct,'-q');
        ACC(i) = accuracy(1);
        if accuracy(1) > min_error(1)
            min_error = [ accuracy(1) gamma_coef(j) * def_gamma
 c_vals(i) ];
        end
    end
    subplot(2,2,j); hold on;
    title(sprintf('Gamma value: 1/%g * %.4g', size(X,2),
 gamma_coef(j)))
    plot(c_vals, ACC); xlabel('Cost'), ylabel('Accuaracy')
end


fprintf('Minimal Error Parameters:\n')
fprintf('Gamma = %g | Cost = %g:\n', min_error(2:3))
fprintf('We achived training accuaracy of: %g\n', min_error(1))

fprintf('\nWe can use the Gradiant Decent method to optimize the
 parameters\n')
```

```
The accuracy for the training data is: 96.64%
Its not 100% because data is not linear seprable.
The accuracy for the test data is: 97.07%

***** GAMMA C PLAYGROUND *******
Minimal Error Parameters:
Gamma = 0.0222222 | Cost = 6.7:
We achived training accuaracy of: 99.3289

We can use the Gradiant Decent method to optimize the parameters
```



Published with MATLAB® R2017a