In this assignment you will take the provided starter code and fill in the missing details in order to create a working perceptron implementation.

To start, download the following code files:

- learn_perceptron.m
- plot_perceptron.m

And the following datasets:

- dataset1.mat
- dataset2.mat
- dataset3.mat
- dataset4.mat

Attention: some people have notified us that the provided datasets do not load under some versions of Octave. We are providing the same datasets in a different format that will hopefully work with more versions. You can find these files below.

And the following datasets:

- dataset1_ancient_octave.mat
- dataset2_ancient_octave.mat
- dataset3_ancient_octave.mat
- dataset4_ancient_octave.mat

For those who want to download all of the files together in a zip archive, you get get them here: Assignment1.zip

To run the code, you first need to load a dataset. To do so enter the following command in the Octave console (to load dataset 1):

load dataset1

This should load 4 variables:

- neg_examples_nobias - The matrix containing the examples belonging to class 0.
- pos_examples_nobias - The matrix containing the examples belonging to class 1.
- w_init - Some initial weight vector.
- w_gen_feas - A generously feasible weight vector (empty if one doesn't exist).

The variables have _nobias appended to their names because they do not have an additional column of 1's appended to them. This is done automatically in the learn_perceptron.m code already. Now that you have loaded a dataset, you can run the algorithm by entering the following at the Octave console:

w = learn_perceptron(neg_examples_nobias,pos_examples_nobias,w_init,w_gen_feas)

This will start the algorithm and plot the results as it proceeds. Until the algorithm converges you can keep pressing enter to run the next iteration. Pressing 'q' will terminate the program. At each iteration it should produce a plot that looks somethinglike this.

The top left plot shows the data points. The circles represent one class while the squares represent the other. The line shows the decision boundary of the perceptron using the current set of weights. The green examples are those that are correctly classified while the red are incorrectly classified. The top-right plot will show the number of mistakes made by the perceptron. If a generously feasible weight vector is provided (and not empty), then the bottom left plot will show the distance of the learned weight vectors to the generously feasible weight vector.

Currently, the code doesn't do any learning. It is your job to fill this part in. Specifically, you need to fill in the lines under learn_perceptron.m marked %YOUR CODE HERE (lines 114 and 122). When you are finished, use this program to help you answer the questions below.