CS436/536: Introduction to Machine Learning
Homework 2
(Due Thursday 2/15 **before the class**)

Instructions: To solve these problems, you are allowed to consult your classmates, as well as the class textbook (*Learning from Data* by Abu-Mostafa, Magdon-Ismail, and Lin, which we will call LFD) and the slides posted on Brightspace, but no other sources. You are encouraged to collaborate with other students, while respecting the collaboration policy (please see the module on Academic Honesty on Brightspace). Please write the names of all the other students you collaborated with on the homework. Everyone must write up their assignments separately.

Please write clearly and concisely, and use rigorous, formal arguments. Homework is due at the beginning of lecture, and homework turned in later will be considered late and will use up one of your late days. You must use Brightspace to submit the homework as a single neatly typed pdf file. Hand-drawn formulas or figures are okay and may be included as images within the pdf. If a programming assignment calls for plotting the results, axes must be clearly labeled, and its meaning must be obvious to anyone with only a rudimentary knowledge of machine learning and computer science. Emailed copies will not be accepted.

(1) [200 points] LFD Exercise 2.6.

(2) [100 points] LFD Problem 2.12.

(3) [400 points] LFD Problem 2.24.

(4) [100 points] Computing Features with the Handwritten Digits Data.

This week, you will familiarize yourself with the MNIST handwritten digits dataset. You may download the following files from Brightspace that accompany this homework: `ZipDigits.train` and `ZipDigits.test` contain training and test datasets respectively. The content and format of these files is documented in `ZipDigits.info`.

Each row of `ZipDigits.train` (or `ZipDigits.test`) corresponds to an example of a handwritten digit. Entries in a row are space separated. The first entry in a row correctly identifies the digit (0-9), and the next 256 entries are grayscale values between -1 (white) and 1 (black) corresponding to a 16×16 image read row after row.

For this problem, we will use only the digits 1 and 5, so you must first remove the other digits from the training and test datasets. Then, perform the following tasks:

(a) Familiarize yourself with the dataset by giving a plot of the first two digits in `ZipDigits.train`.

Hint: If you are using the Python programming language, you may use `matplotlib.pyplot.imshow` which takes a 2-D array as input. You may refer to the documentation on how to display a grayscale image.

(b) Develop *two* features to measure properties of the image that would be useful in distinguishing between the digits 1 and 5. You may use *average intensity* and *symmetry* (defined in LFD Example 3.1) as your two features, or define and compute any other features you think are better suited to help distinguish between 1 and 5. Provide a mathematical definition of the two features you compute using the notation discussed in class.

(c) Provide a 2-D scatter plot of the examples in `ZipDigits.train` w.r.t. the two features you defined in Part (b), similarly to the scatter plot in LFD Example 3.1 and elsewhere in LFD Chapter 3. For each example, plot the values of the two features with a red '$\times$' marker if it is a 5 and and a blue '$\circ$' marker if it is a 1. You must clearly label each axis with the feature it represents, and it should be possible to determine for each data point, the values of the two features you computed. You must also include a *legend* on the upper right corner of your scatter plot which clearly identifies that data points marked with '$\times$' represent examples of the digit 5 those marked '$\circ$' marker represent examples of the digit 1.

Hint: If you are using the Python programming language, you may use `matplotlib.pyplot.scatter` which creates a 2-D scatter plot of n data points when given as input two 1-D arrays `x` and `y` of length n which provide the positions of the data points along the first and second dimensions respectively.

Additional tips for plotting:

- You may find it useful in the long run to use `matplotlib.pyplot.subplots` to generate both a `matplotlib.figure.Figure` object and one or more `matplotlib.axes.Axes` objects.

  At a high level, you may think of the Figure object as a sort of canvas that gets displayed or saved as a file and the Axes object as a collection of plot elements that need to be printed onto the canvas. Each `Axes` object may therefore refer to a different collection of plot elements that together make up a plot and you get to pick where it gets printed on the canvas or let the library decide it for you. Therefore, it will often be useful to maintain a pointer to the Axes objects of interest so you can manipulate them and add different plot elements like axis labels, legends, and so forth.

  For more details, you may find this article and this article to be of interest.

- To add a legend to your plot, you may find this guide and this guide.

- To manually control how the values along each axis are marked, you may use `matplotlib.axes.Axes.set_xticks` and `matplotlib.axes.Axes.set_yticks`, and find this guide helpful.

- If you are having trouble printing your image, you may use `matplotlib.pyplot.tight_layout`. Its use is documented in this guide.

(5) [500 points] Classifying Handwritten Digits: 1 vs. 5. Implement the algorithm for linear regression for classification followed by the pocket algorithm.

Find the best separator *using the training data only* (using `ZipDigits.train`). For each example, use the two features you computed in HW2 to as the inputs; and the output is $+1$ if the handwritten digit is 1 and -1 if the handwritten digit is 5. Once you have found a separator using your classification algorithm:

(a) Give separate plots of the training data (`ZipDigits.train`) and test data (`ZipDigits.test`) which display the data points using the two features you computed in HW2, together with the separator.

(b) Compute E_{in} on your training data (`ZipDigits.train`) and E_{test}, the error of your separator on the test data (`ZipDigits.test`).

(c) Obtain a bound on the true out-of-sample error (E_{out}). You should get two bounds, one based on E_{in} and another based on E_{test}. Use a tolerance of $\delta = 0.05$. *Which is the better bound?*