Homework 2

Due Sept. 28, 2020.

Unless otherwise stated,

- all 1-D vectors, such as x and w below, are column vectors by default.
- the classification problem is binary.
- a lower case letter variable is a scalar or vector, where a upper case letter (in any font) is a matrix.

Theory [0.8 pt each]

1. Given a sample with feature vector $x = [1.1, 2.2, 3.3]^T$, what is its augmented feature vector?

2. If the weight vector of a linear classifier is $w = [1, 0, 1, 0]^T$, and we define that a sample belongs to class $+1$ if $w^T x > 0$ and -1 if $w^T x < 0$ where x is the augmented feature vector of the sample, what is the class of the sample?

3. In our discussion at the class, we used augmented but not normalized augmented (normalized and augmented) feature vectors to derive the solution for a linear classifier that minimizes the sum of squared error. What is the equation for computing the same loss function if using **normalized augmented** feature vectors? Let x''_i be the normalized augmented feature vector of the i-th sample, and w be the weight vector of the classifier. A correct prediction shall satisfy $w^T x''_i > 0$ regardless of the class of the sample because x''_i has been normalized. You may use a computational algebra system to help – but it is not required. It might be easier by hand on scratch paper.

4. Please derive the solution for minimizing the new loss function. Keep variables and font style consistent with those in the class notes/slides. Denote $\mathbb{X}$ as a matrix, each row of which corresponds to a sample and each column of which corresponds to a feature value except the rightmost one (i.e., the bias term).

5. The loss we discussed in this class is called L2 loss because of the squared error. We could also use the absolute error, by replacing the square operation with the absolute value operation. This is also known as L1 loss. Now define such a loss function without using the absolute value operation (suppose your are on a computer that doesn't have the abolute value operator, branch operators, or even square root operator, i.e., you cannot get absolute value using if-statements). The fact that class labels are either $+1$ and -1 gives you an edge. ("Anti"-Hint: if you are looking for hinge loss or cross-entropy loss, you are wrong. If you don't know what they are, it's no problem for now.)

Programming

6. [3pts] The demo `2_Linear_classifiers.ipynb` provides snippets of code to achieve a linear classifier using the minimization of the sum of the squared errors, and the visualization of the classifier. Now please define a function `plot_mse` that visualizes the training samples and the classifier on the sample plot.

Inputs/arguments:

- `X`: a 2-D numpy array such that `X[i]`, which is 1-D numpy array, corresponds to the i-th sample's feature vector (not augmented). The shape of `X[i]` is 1-by-2.
- `y`: a 1-D numpy array such that `y[i]`, which is a scalar $\in [+1, -1]$, is the label of the i-th sample.
- `filename`: a string, the path to save the plot.

Output/return:

- `w`: the weight vector of the classifier, where the 1st element corresponds to the 1st dimension of a feature vector, the 2nd element corresponds to the 2nd dimension of a feature vector, and the last term is the bias.

For plot, please plot class $+1$ samples in blue dots and class -1 samples in red dots. For all other objects and properties (e.g., line width, line color, marker size, marker border color, etc.), use default settings.

Be sure that the plot range does not go beyond the sample range. To do that, use `xlim` and `ylim` of matplotlib to set:

```
matplotlib.pyplot.xlim(numpy.min(X[:,0]), numpy.max(X[:,0]))
matplotlib.pyplot.ylim(numpy.min(X[:,1]), numpy.max(X[:,1]))
```

7. [3pts] Redo the function above using Fisher's linear discriminant. Save the function as `plot_fisher`.

Note that on the slides, when discussion Fisher's, the **w** does not contain the bias. But when returning **w** here, please add the bias as the last element.

How to submit

For theory part, submit as a PDF file. For programming part, work on the template `hw2.py` and submit it.

How to view this in nice PDF

```
pandoc -s hw2.md -o hw2.pdf
```

Programming hints

1. To do the $\sum(\mathbf{x}-\mathbf{m_i})(\mathbf{x}-\mathbf{m_i})^T$, first subtract the mean on each feature dimention from each column of X: