EE 559 **Homework 1** Posted: Friday, 1/20/2023
Jenkins Due: Fri., 1/27/2023, 11:59 PM PST

Submit by uploading two pdf files to the D2L Dropbox "Homework 1". This can be found in "Content" area, "Homework submission" tab on left side (before "Week 1"). Upload:

(1) A single pdf file of all your answers to the HW problems. Your solutions can be handwritten or typewritten. If handwritten, you can convert to pdf by scanning or using a smartphone camera; if you use a smartphone camera then use a pdf scanner app to convert the pictures to scan-like pdf (e.g., CamScanner).

(2) A single pdf file of your code. The code file must be computer readable (not a scan or a screenshot) – you can check for this by trying to select and copy text in the pdf file.

Always, when submitting files of your work by upload to D2L, after uploading it is good practice to check the files that you uploaded to make sure they are the correct files and are complete.

Python 3 is the preferred language for solving this homework assignment. However, you may use MATLAB if you prefer. Please keep in mind that MATLAB will only be accepted for Homework 1 and Homework 2; after that everyone must use Python 3.

Resources for help with coding in Python: help files, internet resources including Python documentation and tutorials, Python resources listed in the syllabus, discussions with other students (via piazza or direct communication), and TA office hours.

In this homework assignment, **your Python coding should use only standard functions**, methods, etc. in Python, NumPy, and matplotlib; and you can use pandas only for reading and writing csv files. Use of other libraries is not permitted.
MATLAB users may use csvread function to read the csv file into a matrix. Functions such as size, mean, min, sqrt, and unique should be sufficient for this homework. All other standard MATLAB operations such as slicing, concatenation of matrices are allowed.

1. Write code to implement a 2-class nearest-means classifier for data that has 2 features. (In part (e) below you will also need to run the case of data that has just 1 feature.)

   You are given 3 datasets, each of which is divided into a training set and test set.

   dataset1_train.csv, dataset1_test.csv,
   dataset2_train.csv, dataset2_test.csv,
   dataset3_train.csv, dataset3_test.csv

   Each csv file has one row for each data point, and one column for each feature; except the last column contains the class labels (1 or 2).

   Note that for a nearest-means classifier, the training or learning phase consists merely of computing the class means; and the classification of each data point can be done by finding which class mean is closer.

Tips: • For plotting, **PlotDecBoundaries.py** or **PlotDecBoundaries.m** function helps you plot the data points, means, decision boundary and regions. (Or you may write your own plotting code if you prefer.)

The PlotDecBoundaries() in PlotDecBoundaries.py requires three arguments,

i. Training data points of all classes (of shape [n_train, n_feats])
ii. Corresponding Train labels (of shape [n_train])
iii. Class sample means (k^{th} row needs to be set as the sample mean of the k^{th} class)

The PlotDecBoundaries.m function in MATLAB requires three arguments,

i. Training data points of all classes (of size [n_train, n_feats])
ii. Corresponding Train labels (of size [n_train,1])
iii. Class sample means (k^{th} row needs to be set as the sample mean of the k^{th} class)

• As a good coding practice, it is recommended that your code should be generalized to work for any given number of classes and data which has any number of features. Accordingly, you can derive the number of classes from the training labels using unique() function from NumPy or MATLAB. The number of features can be derived from the number of columns in the csv file (number of features is one less than the number of columns). This will also ensure the same code works for part (e).

(a) *Learning (training) and classification*. Use unnormalized data as supplied in the datasets.

For each dataset (1, 2, and 3), do the following.

Compute the class means on the training data.

(i) Plot the training data (using different colors or symbols for the different classes), the class means, the decision boundary, and decision regions.

Classify all data points in the training set and in the test set, using the class means computed above.

(ii) Report the classification error rate on the training set, and separately on the test set. Classification error rate = (Number of misclassified points) / (total number of points), expressed as percentage.

(b) Compare and comment on the results: how do the test error rates on datasets 1, 2, and 3 compare? Try to explain why.

(c) *Preprocessing: normalization.* Standardize the data (so that each feature, across both classes combined, has sample mean = 0 and sample variance = 1). For each dataset, compute the normalizing parameters from the training data, and then use those parameter values to standardize the training data and test data. The result is the (standardized) data you will use for this part.

Repeat part (a), except on the normalized data.

(d) Compare and comment on the results of part (a) to those of part (c): how do the error rates on normalized (standardized) and unnormalized data compare for each given dataset? Try to explain why.

(e) *Feature transformation: feature reduction by projection.* For this part start with the standardized data. Reduce the number of features to 1 by using a projection transformation, as described below.

A projection transformation will project all data points onto the same line, to result in a new, 1D feature space.

For each (standardized) dataset, do the following:

To get a somewhat optimal result for the feature transformation, try different directions for the line by using the following brute-force technique on the training set:

Try 40 different vector directions $\underline{r}_m$ to project onto, as follows:

$\underline{r}_m = (10, m)$, $\mathrm{m} = 0,1,2,\cdots,9$

$\underline{r}_\mathrm{m} = (20 - \mathrm{m}, 10)$, $\mathrm{m} = 10,11,12,\cdots,29$

$\underline{r}_m = (-10,\ 40 - m)$, $\mathrm{m} = 30,31,32,\cdots,39$

For each value of m, project all the training data points onto $\underline{r}_m$. The result will be a set data points in 1D (projected) feature space. Then train the 1D classifier by computing the class means. Next, classify all the training data points in the 1D feature space.

To optimize the feature transformation, plot the training error rate vs. m. Report which value(s) of m give(s) the lowest training error. Call it m^*; if more than one value of m give the lowest training error, then let m^* be the lowest such m.

Also report $\underline{r}_{m^*}$.

Then classify all the points in the test set in the 1D space for $m = m^*$. Show the projected points, as well as the decision boundaries and regions, on a 2D plot (using the same axes as the other 2D plots), all for $m = m^*$. Report the resulting test error.

(f) Compare and comment on the results: how do the test error rates on (optimized) 1D data of part (e) compare with test error rates on 2D normalized data of part (c), for each given dataset? Try to explain why.