EE 510: Mathematical Foundations of Machine Learning **Winter 2020**

Demo: Subspace-based Handwritten Digit Recognition

Instructor Name: John Lipor

In this demo, you will implement a nearest-subspace classifier for the MNIST handwritten digits database. This is one of the most common benchmark datasets in machine learning, and state-of-the-art algorithms achieve accuracy better than the average human. In this demo, we'll see how a simple classifier based on low-rank approximation gets quite good results. Follow the steps below in order after reading Section 6.3 of the Low-Rank Approximation notes.

Task 1: Data Exploration

Open `classifyDigits`, which loads the dataset in for all 10 digits. The variables are the images (also called *feature vectors*) `X` and the corresponding class labels `trueLabels`. There are 200 images from each class. Note that

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}$$

is the matrix whose columns are the vectorized images of the digits. The original image size is 28×28.

- First, have a look at the actual images. Use subplots to plot 3-4 images from each of 3-4 classes. Hint: Use the `reshape` command.
- Next, pick a couple of digits (classes), and plot the singular values of the matrix whose columns are the vectors *only from a single class*. For example, let $X^{(1)}$ be the matrix whose columns are all the images of the digit "1". Plot the singular values of this matrix.
- For the classes above, take note of how many singular values are needed to capture "most" of the energy in the data. This is a good way to estimate the intrinsic dimension of the data. You can evaluate this quantitatively by looking at the ratio
$$\frac{\sum_{k=1}^{K} \sigma_k}{\sum_{k=1}^{r} \sigma_i}.$$
- Raise your hand once you've finished this part, and I'll come by to look at your work.

Task 2: Train Classifier

Your second task is to complete the `trainUoS.m` function. This function takes in a set of feature vectors and labels, as well as a tuning parameter `r`, which is the desired subspace dimension. The output is a matrix of size $D \times r \times 10$, where each "page" corresponds to a $D \times r$ orthonormal basis for the learned subspace.

- Split the data into training and test sets. Use the first 100 images *from each class* to build the training set and the next 100 images to build the test set.
- Implement the `trainUoS` function as described in the `trainUoS.m` file.
- Pick a few digits/classes and display the principal components returned by `trainUoS` run on the training data as images. That is, pick some $U_k = U_{\text{full}}(:, :, k)$, and display the first 3-4 columns of U_k as images. Do this in a subplot for a few values of k. What do you see? Hint: The `squeeze` command may be useful.
- Raise your hand once you've finished this part, and I'll come by to look at your work.

Task 3: Test Classifier

Now that you have trained a set (union) of subspaces that describes the training data, we'll see if this does a good job of predicting on *new* (test) data.

- Implement the `nsClassify` function as described in the `nsClassify.m` file. See Section 6.3 of the notes.
- Pick a value of r that makes sense to you based on your evaluation in Task 1, and run your classifier. What is the resulting classification error?
- Choose a few mislabeled examples from the test data and display the corresponding images. What do you notice?
- If time allows, perform "model selection" by looping over a range of values for r. Which value works best? What is the resulting error?
- Raise your hand once you've finished this part, and I'll come by to look at your work.