APMA E4990.002: TAKE-HOME QUIZ 1
DUE SUNDAY, OCTOBERT 23, AT 11:59PM

Please justify your answers, proving the statements you make. You are allowed to refer to results shown in lecture (or that are in the textbook) as long as you state them precisely, meaning that you should say exactly which hypothesis are needed in the result you use.

This quiz is open book, but you are not allowed to share or discuss this quiz with any person (including not posting regarding this quiz on Ed discussion or elsewhere). If you have any questions or find errors please email me at vak2116@columbia.edu.

If you need to impose extra conditions on a problem to make it easier (or consider specific cases of the question, like taking n to be 2, e.g.), state explicitly that you have done so. Solutions where extra conditions were assumed, or where only special cases where treated, will also be graded (probably scored as a partial answer).

Note that it may take some time to upload the quiz on Gradescope. Please plan accordingly. I suggest uploading your quiz at least 30 minutes before the deadline. Make sure your answers to each problem are clearly stated in the submitted PDF. All code (Jupyter Notebooks and other source files) used to compute your answers should also be uploaded to Gradescope, along with a readme file telling the graders how to run your code if they need to, and if you are using any special packages. Include a .zip file with your online homework submission containing all of your source code.

(1) *Denoising by projections and nearest neighbor classification:* In this exercise you will use the code in the `mnist` folder of `quiz1.zip`. The dataset in `mnist_all.mat` contains images of handwritten digits labeled with their associated numeric values.
   (a) Complete the *denoise* function in `denoising.py` which denoises the given image by orthogonally projecting it onto the subspace of the given sample images. Include the generated images in your submitted homework.
   (b) The nearest-neighbor algorithm (which we defined in Lecture 11) classifies elements of a test set by finding the closest element in the training set. Complete the *compute_nearest_neighbors* function in `nearest_neighbors.py` that finds the image in the training data that is closest to a given test image. Include the generated images in your submitted homework.
   (c) Create a file called `pc_nearest_neighbors.py`. In it you must write code to complete the following tasks:
      (i) Generate a plot of k vs. σ_k, where σ_k is the kth largest singular value of the data (e.g., σ_1 is the largest singular value). Look at `denoising.py` and `nearest_neighbors.py` to see how to use the function *load_train_data* in `mnist_tools.py`. Include the plot in your submitted homework document.
      (ii) Plot (using plot image grid in `plot_tools.py`) the singular vectors corresponding to the top 10 singular values of the data. Your singular vectors should be elements of $\mathbb{R}^{784}$ (i.e., they should represent images). Include the plot in your submitted homework document.

Date: October 17, 2022.

(iii) Use the singular value plot to determine a relatively small number k of right singular vectors (*principal components*) that explains the training data reasonably well. Project the training data and the test data (obtained using *load_test_data* in `mnist_tools.py`) onto the first k principal components, and run nearest neighbors for each test image in this lower dimensional space. Include your choice for k, and the plots of your nearest neighbor results in your submitted homework document. You should use the code from `nearestneighbors.py` to generate your image plots.

(iv) Give a potential reason why the principal component-based nearest-neighbor approach used in the previous part could be more accurate than using the full training set.

Some notes to keep in mind: (a) For $A = USV^T$, the function *numpy.linalg.svd* returns V^T and not V as its third output. (b) The data points in the training and test data are given as rows.

(2) *Curse of dimensionality:* Nearest-neighbor classification tends to be more successful when the test point has training points in close proximity. In this problem we investigate whether this is feasible for high dimensional datasets. Suppose our training points $S = \{x_1, x_2, ..., x_n\}$ all have the same class label, and are chosen independently and uniformly from the unit ball B defined by:

$$B = \{x \in \mathbb{R}^d : \|x\|_2 \leqslant 1\}$$

(a) Suppose our test point is located at the origin, and that the nearest-neighbor classifier is accurate if there is a training sample in S within 0.1 of the origin (measured in Euclidean distance). In terms of n and d, what is the probability the nearest-neighbor estimator is accurate?

(b) What does your answer to the previous part say as d becomes large? I.e., roughly how much data is necessary for the probability of accurate classification to be larger than some constant.

(3) *Randomized projections and nearest neighbor classification:* In this exercise you will use the code in the `randomnn` folder of `quiz1.zip`. Complete the function `random_nearest_neighbors` in `random_nearest_neighbors.py` to implement the nearest neighbor algorithm after Gaussian random projections. The specifications of that function are described in the comments. Include the 3 output plots from the program in your submission.

(4) *Randomized SVD:* In this exercise you will use the code in the `movie` folder of `quiz1.zip`. For the data loading code to work properly, make sure you have the *moviepy* Python package installed on your system. In this problem we will be using the randomized SVD algorithm to analyze the movie file given in `JohnOliverClip1Gray.mkv`. If you are unable to watch the movie on your system, consider installing the latest version of VLC (`https://www.videolan.org/vlc/index.html`). Implement the random svd function in `random_svd_movies.py`. Here you will implement the randomized SVD algorithm discussed in class. As described in the comments, you must handle both Gaussian random projections, and random column subsampling. For more information, review the class notes and the textbook on the randomized SVD algorithms. In your submission, include the output from the plots generated by `random_svd_movies.py`.