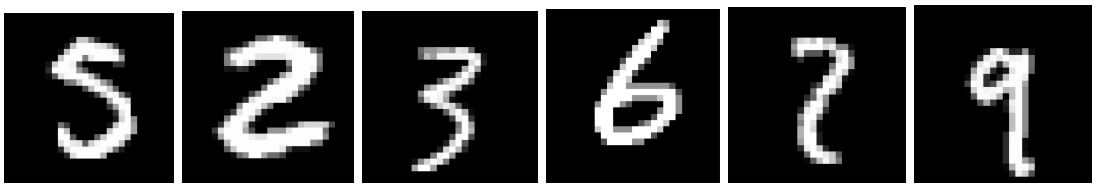
CSE 5544 Introduction to Data Visualization

Lab 4: Visualization and Analysis of the MNIST data with t-SNE

Assign: 3/6
Due: 3/20
15 points

In this lab, you will analyze the MNIST handwritten digits dataset using Python and D3. You will process the MNIST dataset using Python packages (e.g., numpy, Pandas, Matplotlib), and project the high dimensional images into 2D with the TSNE/PCA function in the scikit-learn package (see the provided sample code). D3 is your weapon to visualize and interact with individual data instances (i.e., images of digits) of the data.

MNIST (http://yann.lecun.com/exdb/mnist/) is a collection of gray-scale images for handwritten digits. The dataset contains 70,000 images for 10 digits/classes (i.e., digit 0-9). Each class has ~7,000 images of a digit with different styles (e.g., bold, italic). All images share the same resolution of 28x28 pixels. Some instances of the dataset are shown below as examples (from left to right are images of digit 5, 2, 3, 6, 2, and 9).

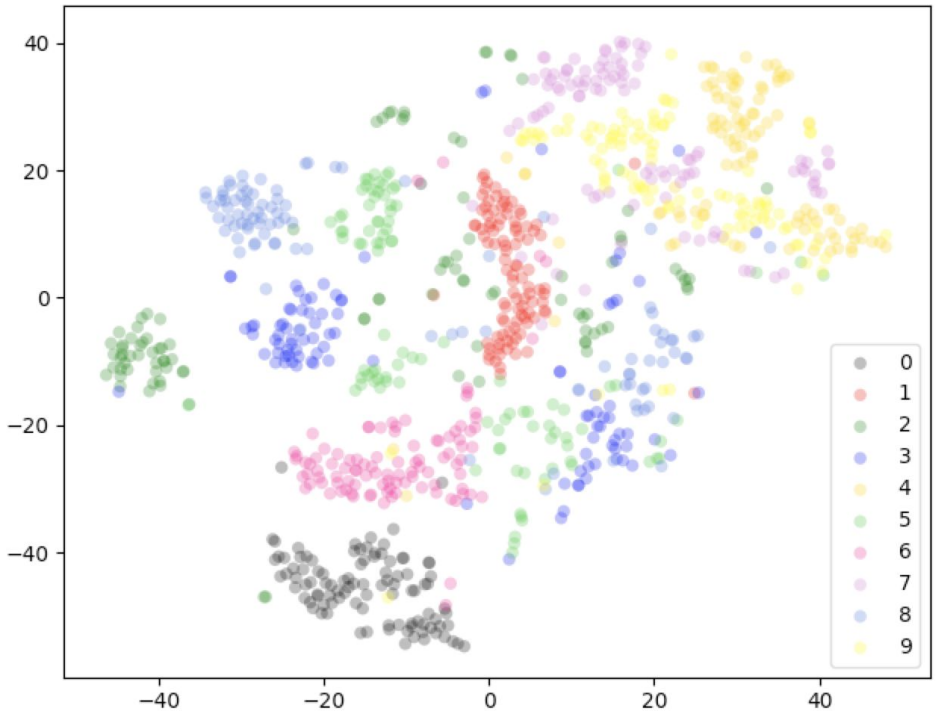


Please use the following Python code to download and verify the MNIST data:

1. **download_mnist.py**: run the code with "python download_mnist.py", and the script will (1) download the original MNIST data (70,000 images and their labels) into the current directory as "./mldata/mnist-original.mat"; (2) extract 100 images for each digit and save them as "./mnist_X.npy" and the corresponding labels as "./mnist_y.npy". In other words, "./mnist_X.npy" contains 1,000 images, the first 100 are images of digit 0, the second 100 are images of digit 1, … This lab will use "./mnist_X.npy" and "./mnist_y.npy" instead of the original MNIST data (too large for us). Therefore, you are free to delete "./mldata/mnist-original.mat" after executing the script.
2. **vis_mnist.py**: run the code with "python vis_mnist.py" (make sure it is in the same directory with your data). The script will randomly select one image from the 1,000 images in "./mnist_X.npy" and visualize it as a gray scale image (use this script to verify your data and have a glance of different handwritten digits).

Each 2D image can be considered as a high-dimensional vector (i.e., linelize the 28x28 pixels of one handwritten digit image into a 786 dimensional vector), and our MNIST data set ("./mnist_X.npy") has 1,000 such vectors. These vectors can be considered as 1,000 data points in a 786 dimensional space (far beyond what our eyes can perceive). Your goal in this lab is to transfer/project these 1,000 points from the 786 dimensional space to two dimensional space (so that we can see them).

One fundamental rule to follow is that data points that are close to each other in the 786 dimensional space should still be close in the two dimensional space. For example, different images of digit 0 in MNIST are similar to each other (i.e. close to each other in the 786 dimensional space) when compared with images of other digits (e.g. digit 2, 3). Therefore, these images, when projected to 2D space, should also be close to each other. Many existing dimensionality reduction algorithms can help us do this work, such as Principal Component Analysis (PCA), Multidimensional Scaling (MDS) and t-Distributed Stochastic Neighbor Embedding (t-SNE, our focus in this lab). The following image shows the result of projecting the 1,000 images in our MNIST data into 2D space. It can be seen that images of different digits show very clear cluster structures. For example, the bottom cluster of black circles represent all images of digit 0. One circle in this 2D plane represents one image in the original 786 dimensional space. You can reproduce the image by running the **tsne_mnist.py** with "python tsne_mnist.py". Notice that you may not see exactly the same result (as t-SNE does not guarantee the global position of each data point), but the cluster structure should be similar, i.e., points representing the same digit should roughly be in the same cluster.



Your tasks in this lab:

1. Read and understand the code in **tsne_mnist.py**. Extract the 2D coordinates for each original image and save them into a file (say, mnist_2d.csv or mnist_2d.json)
2. Use D3 to visualize the 2D locations of each image as a circle. Use different colors for different digits (something similar to the above figure).
3. When hovering over different circles, pop up the original handwritten image and the label for the image (to do this, you need to understand **vis_mnist.py,** and save the pixel value of each image and its corresponding labels as a json file).
4. Modeify **tsne_mnist.py** to change the dimensionality reduction algorithm from t-SNE to PCA, and repeat task 2 and 3. Here is the reference for PCA in Python: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
5. Compare your results from task 3 and task 4, and write a brief summary of your findings.

Hint:
For task 3, you have two alternative ways to draw images on the screen:

1. You can learn how to draw images on a canvas object
    - See w3Schools Canvas tutorial
    - The benefit is that you don't need to generate image files for digits. Canvas can directly render images from the gray scale data of pixels.
2. You can also generate one image file for each digit by Python, and insert an image file as an "image" element on SVG
    - See the description of MDN SVG <image>
    - *x* and *y* attributes control the position of the image element
    - *width* and *height* attributes control the size of the image element
    - *xlink:href* attribute indicates URL for the image file

The provided python code has been tested with Python **2.7**. Email the TA if you have any questions about the code.