CSEN 140L - Lab 9

Nolan Anderson

Lab 9 High-Level Objectives

- Use PCA on the MNIST dataset
- Complete the tasks on the lab assignment PDF a single Jupyter notebook (.ipynb) file
- Use only the **NumPy** and **Matplotlib** libraries for your implementation
    - No machine learning libraries (except for the data retrieval part given by professor)
- Demo to me and submit your notebook to Camino before leaving the lab section

Review: The Data Explained

- The MNIST dataset consists of 28 x 28 grayscale images of handwritten digits (0 – 9).
  - Since the images are 28 x 28, each image has 784 pixels.
    - Each pixel counts as a feature.
  - Because the dataset is quite large, we will just be using the first 2000 samples.
  - As such, your subset of the data should be (2000 x 784).

Principal Component Analysis (PCA)

- Principal Component Analysis is dimensionality reduction method
- The goal is of PCA is preserve the maximum amount of variance by choosing the appropriate axes (i.e. the principal components)
- There are *at most* as many principal components (PCs) as there are features
  - Usually, we want to use enough PCs to at least cover 50% of the variance
  - However, less coverage is okay when plotting since we can only easily interpret plots of 2 or 3 dimensions

The Algorithm (1)

- Prepare the data
  - Load the image data X from the MNIST dataset
    - This should be the first 2000 images
  - Center the data X by subtracting off the mean
- Compute the covariance matrix for all the features
  - Ensure the matrix is (784, 784), transposing if needed
- Find the eigenvectors and eigenvalues of the covariance matrix with *np.linalg.eigh()*
  - You should then transpose your eigenvector result since the columns are the actual eigenvectors

The Algorithm (2)

- Sort the eigenvalues and eigenvectors based on the eigenvalues
  - Should be largest to smallest (descending order)
  - You can find the sorted indices with *np.argsort()*, but you'll need to reverse the order somehow
  - Once sorted, the eigenvectors are the principal components
- Compute how how much variance of the data is explained by having 1, 2, and 3 principal components
  - Your result should between 0 and 1 for each
  - See tips for additional details

The Algorithm (3)

- To actually perform dimensionality reduction, compute the dot product of principal components with the image data
  - For the 2D plot, use the first two PCs for the dot product, result = (784, 2)
  - For the 3D plot, use the first three PCs for the dot product, result = (784, 3)
- Use Matplotlib to construct 2D and 3D scatter plots of the dimensionality-reduced data
  - Use the MNIST target values to have 10 distinct colors
  - Provide a legend to reference which color is which digit
  - See tips for additional details

Tips

- Computing Explained Variance
  - When computing the explained variance, first compute the total sum of all the eigenvalues
  - Then, you can divide the cumulative sum of the eigenvalues by the total to get an array of explained variance
    - You can also use a sum variable and a for loop to do a similar computation, your choice
  - Since we divided a partial sum by the total, we get values between 0 and 1.

Tips

- Plotting in Color
  - To get the target labels (the digits), use mnist[“target”].
    - Then, I suggest you construct a numpy array with those labels with *dtype=int* since scatter expects an array of integers, not strings
  - To actually implement the different colors, use the *c* field of *scatter()* and set that equal to your array for the digits. Then use the *cmap* field to specify a color map string (e.g. “gist_rainbow”)
    - For more details, consult the following links:
      - pyplot.scatter | colormap_reference

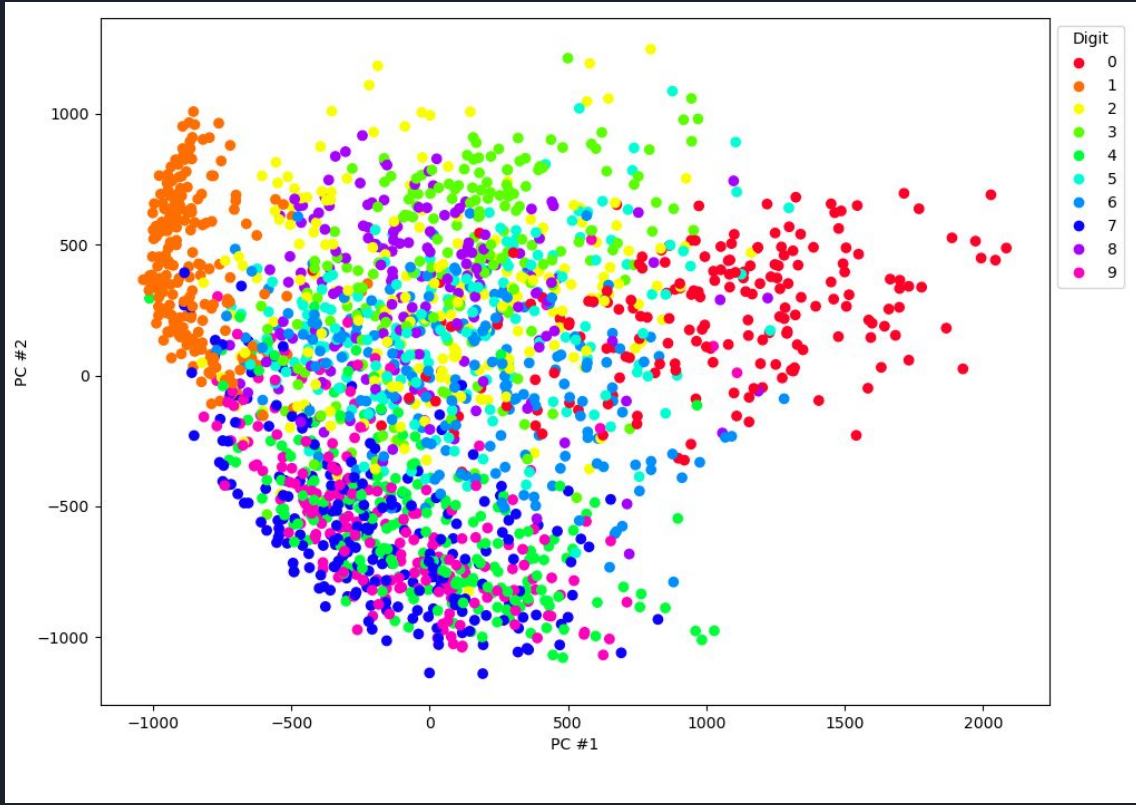
Lab Outcomes | What I Need To See

- Print out the amount of variance covered by having 1, 2, and 3 principal components (three values)
- Display two plots: a 2D plot and 3D plot
  - Your plots must have legends and different colors for different digits

Helpful NumPy Functions

- np.linalg.eigh()
  - Compute the eigenvalues and eigenvectors
- np.argsort()
  - Get sorted indices in ascending order
- np.cumsum()
  - Cumulative sums of an array
- np.mean()
- np.sum()
- np.dot()

Sample 2D Plot



Sample 3D Plot