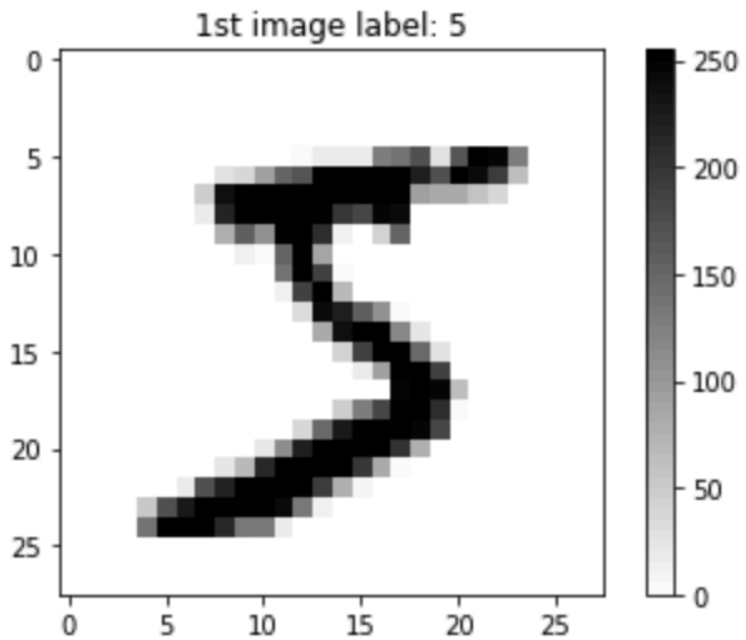
[Intro to AI] HW 3

Due on Apr. 24 at 1:00 pm

Download MNIST dataset by running *hw3_template.ipynb* from the piazza. The *train_dataset* contains a set of images and its target labels of handwritten digits. Each matrix contains a 28x28 pixel grayscale image. Here, it is assumed that you installed [PyTorch](), but you can download MNIST from some other deep learning platforms like [Tensorflow]().

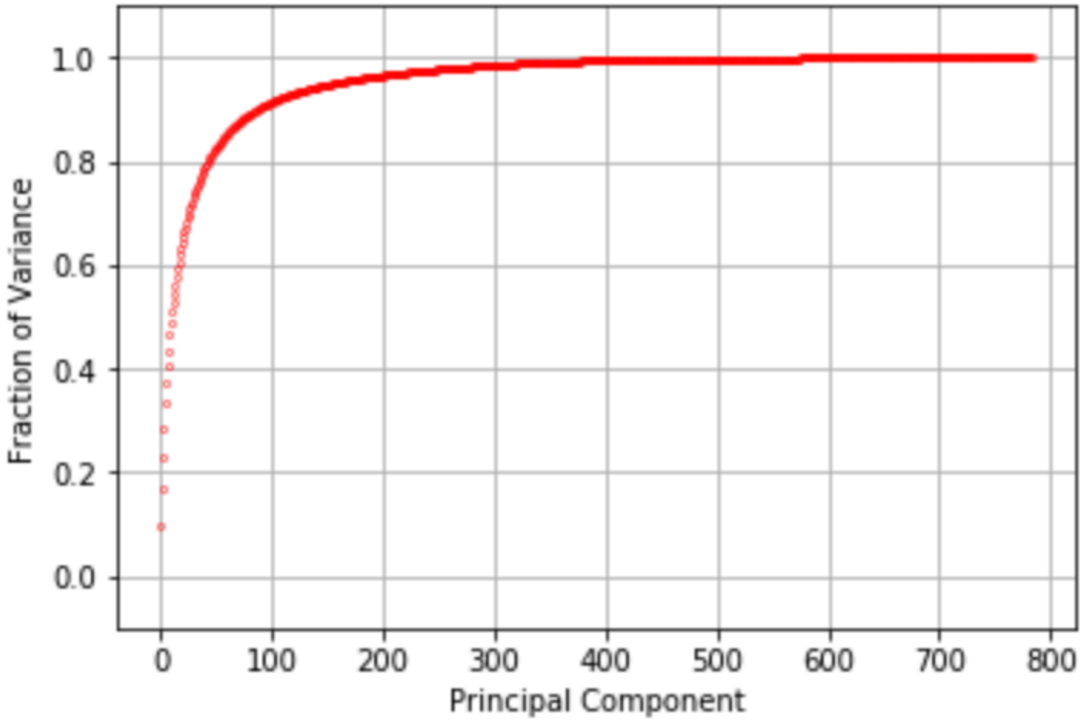
0. **Data type conversion** This step has been implemented in *hw3_template.ipynb* and just execute the cell. Step 0 helps us work with the numpy array X (training data) and y (target labels) from now on, which is converted from torch.Tensor data type.
1. **Data visuallization** We'd like to visualize the handwritten digits. Use `matplotlib.pyplot.imshow` to display the first image of X. Try displaying the same image in a vector form by using `numpy.reshape`. The images below are based on the following colormap: `imshow(..., cmap='gray_r')`



2. **Principal Component Analysis (PCA)**
   - Vectorize all images X by using `numpy.reshape`. The shape of X is now (# of samples) by (784).

- Compute the mean μ of all the images and subtract it from X along the feature dimensions. Set X_0 to the zero-mean data matrix.
- Compute the covariance Σ of X_0. You may use `numpy.cov` or code up the covariance.
- Compute the eigenvalues & eigenvectors of the covariance by using `np.linalg.svd`.

3. **Fraction of variance** We'd like to understand the amount of variance captured by the orthogonal directions we found in step 2.
    - Make a plot of fraction of variance as a function of the number of principal components. You may use `numpy.cumsum`.



    - How many principal components do we need to capture 80% of the total variance at least? Use `numpy.where` to estimate the number, and print out the number as follows:

    `We need at least ## principal components to capture 80% of the total variance.`

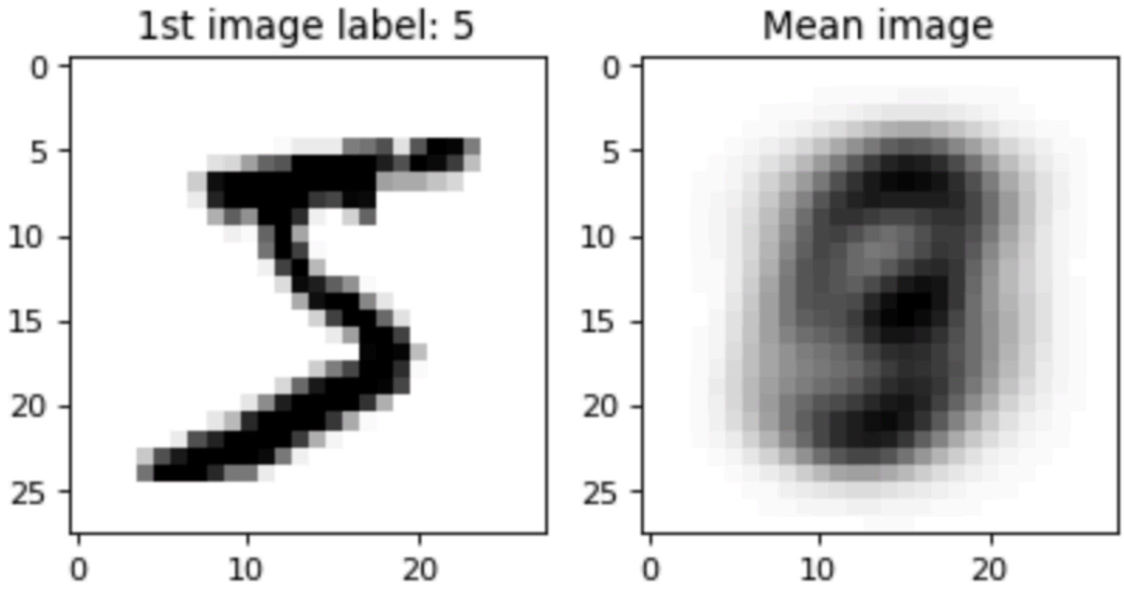
4. **Low-dimensional reconstruction**

    Reconstruct a low-dimensional version of the first image (the first row of the images matrix) from the first 100 eigenvectors. The resulting vector should be $\hat{\mathbf{x}}_i = \overline{\mathbf{x}} + \sum_{k=1}^{K}(\mathbf{x}_i \cdot \mathbf{e}_k)\mathbf{e}_k$, where

    $\mathbf{x}_i$ is the i-th image sample. $\mathbf{e}_k$ is the eigenvector of k-th largest eigenvalue. $\overline{\mathbf{x}}$ is the mean of entire images. $\hat{\mathbf{x}}_i$ is the reconstructed image from K eigenvectors.

5. **Visualize average image**

    Display the first image together with the average image μ.



6. **Visualize the reconstructed image**

   Display the 100 images reconstructed from $\{\mathbf{e}_1\}$, $\{\mathbf{e}_1, \mathbf{e}_2\}$, ... ,$\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{100}\}$. Use `numpy.reshape` and `imshow(..., cmap='gray_r')` as in step 1.