Report

Princy Gautam(B20BB051)

Task 1: PCA (API)

Method

The MNIST dataset is a common dataset used for image classification. tensorflow allow us to import and download the MNIST dataset directly from their keras API.

The MNIST database contains 60,000 training images and 10,000 testing images. Each row represents a number that means there are 70,000 different images, each of size 28x28 pixels flattened into 784 pixels. These images are encoded as NumPy arrays, and the label is array of digits, ranging from 0 to 9.

Separate the dataset into train and test. X_train and x_test part contains RGB codes (from 0 to 255) and y_train and y_test contain labels from 0 to 9 representing number they are. We will visualize these numbers with the help of matplotlib.

Enter the image index. You can select a number between 0 to 60,000. Use matplotlib to view the number at the specified index.

Reshape the x_train array to 28x28

Convert the values in x_train and x_test array to float to get decimal points after division. Also, normalize the RGB codes by dividing it to max RGB value.

Reshape the x_train to 2D array of size (60,000, 784) here, x_train_2d.

Standardize the x_train_2d array to convert mean=0 and standard deviation=1 for each variable.

Import PCA from scikit library. The PCA class depends only upon the feature set and not the label set.

Create an object named pca and initialize the PCA class by passing the number of components to the constructor.

Pass the feature set to fit and then transform method. The transform method returns the specified number of principal components.

The PCA class contains explained_variance_ratio_ which returns variance by each of the principal components. Its datatype is float.

The PCA class contains n_components_ which returns the number of components, by passing the variance to be retained.

Results & observations

It can be seen that first principal component is responsible for 0.097% variance. Similarly, the second causes 0.070% variance and so on in the dataset. These values are quite low since it has too many data. From this, we can say that 0.167% of classification information contained in the feature set is captured by the first two principal components.

The MNIST data already has images with 28x28 pixel size. On reshaping, it would not affect either way. Although on a different size image, we can reshape it to 28x28 pixel size.

Passing the amount of variance to be retained, we obtain the corresponding principal components required. As in our case, we required five principal components.

Task 2: PCA from scratch

Method

Import NumPy, pandas libraries for calculation purpose and matplotlib for plotting.

Import MNIST dataset from tensorflow. Separate the dataset into training and testing sets.

Obtain a 2D array of x_train of size (60000,784) by reshaping it.

Define a function PCA with x_train_2d and number of components as the parameters. Now in the function perform as follows:

Take the mean of the x_train_2d (2D array of x_train)

Subtract the mean of each variable from every row of the respective column to center the dataset around the origin. This is the mean – centered data. This helps in calculating the covariance matrix.

We can calculate the covariance matrix using *np.cov ()* method. To get covariance matrix in the required dimensions set rowvar to false.

NumPy.linalg.eigh returns the eigenvectors and eigenvalues of a symmetric matrix. By default, it returns eigenvalues in ascending order so we need to sort it in descending order with corresponding eigenvectors. It will also arrange the principal components in descending order of their variability. *np.argsort* provides us with an array of indices of same shape.

Now, select a subset from the rearranged eigenvalue matrix as per specified (for e.g., take number of components =50). This means we selected first fifty principal components. *Num_components=50* means our final data should be reduced to 50 variables.

At last, reduce the data to lower dimensions by computing dot product between the transpose of eigenvector subset and the transpose of the mean – centered data. And further transposing the outcome of the dot product we will get reduced dimension data.

A plot with number of components =10 has been tried.

Next, define a function reconstruct_data to reconstruct an image for different values of number of components mentioned. This function takes four parameters the score matrix, eigenvector matrix, vectors corresponding to data mean and the number of components to include. Apply matrix multiplication of score matrix and transpose of eigenvectors matrix and add the mean of data to the outcome of multiplication.

Call the function for the different values of component numbers mentioned and plot using matplotlib.

Now, compare the images reconstructed above with the original image.

Subtract the array of residual image from the original image, this gives us the residual image. Plot the residual image for each value of K (number of components). Calculate the reconstruction error.

Results & observations

The shape of covariance matrix is 784x784. Clearly, it is a square matrix denoting the covariance of the elements with each other. Also note that the covariance of an element with itself is nothing but just its variance. That is why the diagonal elements of a covariance matrix are just the variance of the elements.

The eigenvectors of the covariance matrix we get are orthogonal to each other and represents a principal axis.

Higher eigenvalue corresponds to higher variability. Thus, a principal axis with higher eigenvalue will be an axis with higher variability in the data.

After transforming the data, we get (60000,50) as the reduced shape of our data. Originally, data was of higher dimension (60000,784).