PCA and LDA for OCR Handwritten Digits and Classification using KNN

Sintayehu Zekarias and Fikir Awoke

November 27, 2021

1 Introduction

The aim of this project is to implement Optical Character Recognition (OCR) on handwritten digits (0-9) using the dimensional reduction techniques called Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). Additionally to make a classification using k-Nearest Neighbours (KNN) algorithm. This report presents our implementation of the PCA, LDA, KNN and the approaches we followed to get the final result. Before we get straight to the implementation techniques, we have defined some useful terms.

Optical character recognition (OCR) is the use of technology to distinguish printed or handwritten text characters inside digital images of physical documents, such as a scanned paper document or image file and then converting the text into a machine-readable form to be used for data processing.

Principal component analysis (PCA) is a statistical procedure that is often used as a dimensionality reduction technique. It has been around since 1901 and still used as a predominant dimensionality reduction method in machine learning and statistics. It uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.Sheikh, Patel, and Sinhal (2020)

Linear Discriminant Analysis (LDA) is also a dimensionality reduction technique used in the pre-processing step for pattern-classification and machine learning applications. The goal of LDA is to project the features in higher dimensional space onto a lower-dimensional space with good class-separability in order to avoid curse of dimensionality and also reduce resources and dimensional costs.The original technique was developed in the year 1936 by Ronald A. Fisher.

2 Methodology

2.1 Data collection and Normalization

At first we collected the necessary image data from Kaggle. After getting the data, we extracted the digits from the symbols, then we merged the evaluation data with the training data. In the data Preparation and Prepossessing step, we loaded the raw images from the data set folder and performed the following tasks:

- Read the image file
- Changed the RGB to Gray color(blak and white)
- Cropped the image - Here we removed all the unnecessary spaces all around the image and pulled out what is needed.
- Down-sample the image - The image size and RGB value was 137, 155, 3. so we down-sampled the image to a smaller size which is equal to 28x28.

- Create an intensity percentage - The normal intensity range for RGB value is between 0 and 255. In order to create an input for the algorithm, we simply divided the intensity by 255 and created an intensity percentage.
- We appended the normalized image and the corresponding label to the image and class lists.
- At last we flattened a 2d array into 1d array: Takes a list of images stored as 2D-arrays and returns a matrix in which each row is a fixed length feature vector corresponding to the image. So the image became flatten as a one dimensional array.

NB: We installed openCV to read image files, to change RGB to a Gray scale and to resize our image.

The following image shows the difference between image before and after extraction.

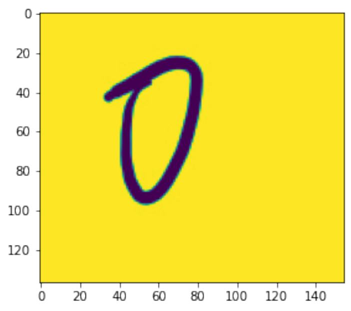
Fig 1: before extraction

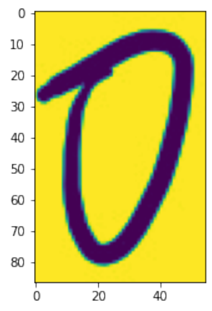
Fig 2: after extraction

The following graphs illustrate the data distribution and data before an application of a dimensionality reduction.

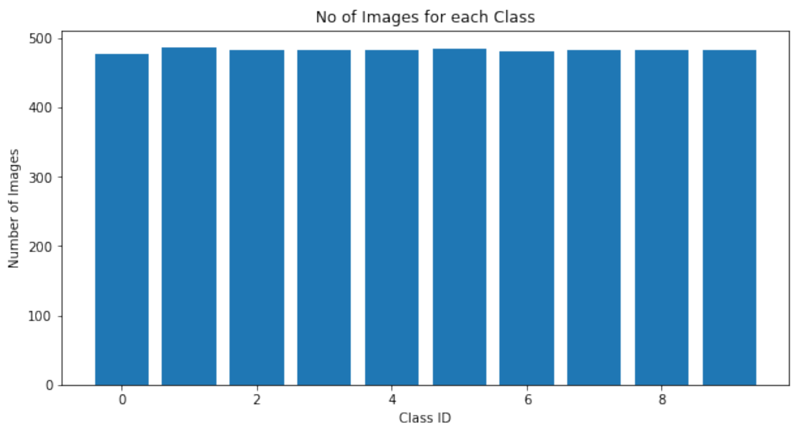

Fig 3: data distribution

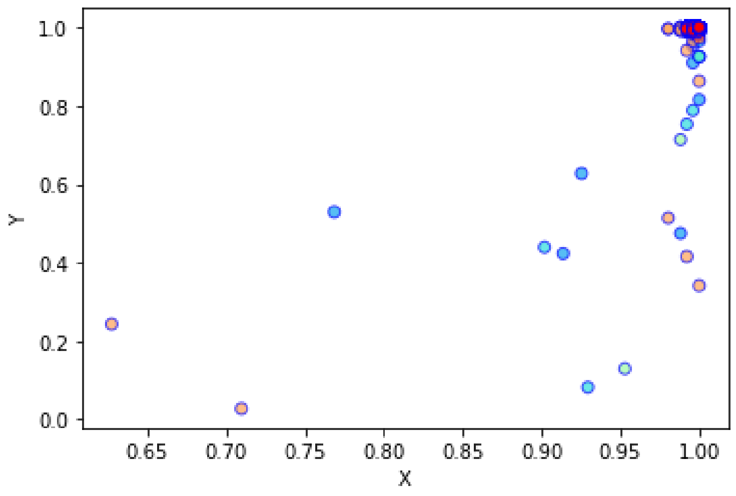

Fig 4: data before reduction

2.2 Dimension reduction using PCA

We followed this steps to implement PCA (Otiko, Odey, and Inyang):

- **Step 1: Standardize the dataset** - our datasets are totally 4833 and have 784 features. We standardized the features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as:

$$z = \frac{(x-u)}{s},$$

where u is the mean of the training samples and s is the standard deviation of the training samples.

- **Step 2: Calculate the covariance matrix** - variance reports variation of a single random variable, and covariance reports how much two random variables vary. On the diagonal of the covariance matrix we have variances, and other elements are the covariances. Covariance matrix has larger diagonal values (variance) and smaller non diagonal values (covariance). The formula to calculate the covariance matrix is:

$$Cov(x,y) = \sum_{i=1}^{n} \frac{(x_i-\overline{x})(y_i-\overline{y})}{n-1}.$$

- **Step 3: Calculate the eigenvalues and eigenvectors for the covariance matrix** - Eigendecomposition is a process that decomposes a square matrix into eigenvectors and eigenvalues. Eigenvectors are simple unit vectors, and eigenvalues are coefficients which give the magnitude to the eigenvectors. Eigenvectors of symmetric matrices are orthogonal. For PCA we have the first principal component which explains most of the variance, orthogonal to that is the second principal component, which explains most of the remaining variance.
- **Step 4: Pick K eigenvalues and form eigenvectors** - we specified the number of K values to select number of eigenvectors.
- **Step 5: Transform the original matrix**
- **Step 6: Visualize the processed data** - The following figure shows the 2D and 3D visualizations of the image data after an application of PCA.

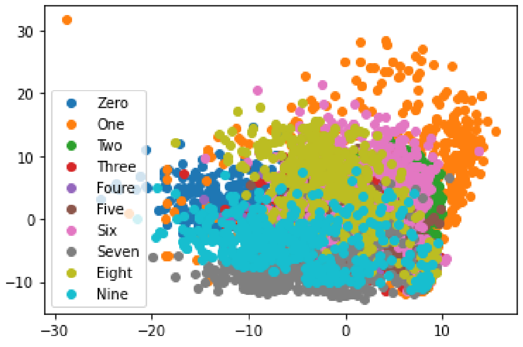

Fig 5: 2D

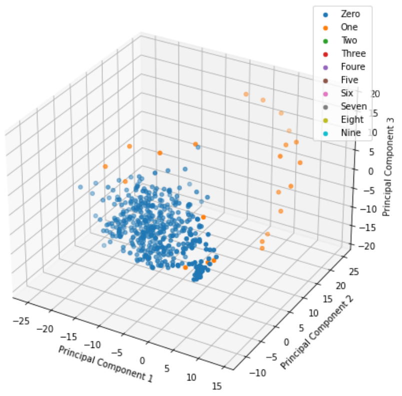

Fig 6: 3D

2.3 Dimension reduction using LDA

We followed this steps to implement LDA:

- **Step 1: Calculate the within-class and between-class scatter matrices** - The formula to calculate within-class scatter matrix is:

$$S_w = \sum_{1=1}^{c} S_i$$

where, c = total number of distinct classes and

$$S_i = \sum_{x \in D_i}^{n} (x - m_i)(x - m_i)^T$$

$$m_i = \frac{1}{n_i} \sum_{x \in D_i}^{n} x_k$$

where, x = a sample (i.e. a row). n = total number of samples within a given class.

And the formula to calculate between-class scatter matrix is:

$$S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$$

$$\text{where, } m_i = \frac{1}{n_i} sum_{x \in D_i}^{n} x_k$$

$$\text{and } m = \frac{1}{n} \sum_{i}^{n} x_i$$

- **Step 2: Calculate the eigenvectors and eigenvalues for the scatter matrices** - We sorted the eigenvalues from the highest to the lowest since the eigenvalues with the highest values carry the most information about the distribution of data. Then, we pick the first k eigenvectors. Finally, we will place the eigenvalues in a temporary array to make sure the eigenvalues map to the same eigenvectors after the sorting is done.
- **Step 3:** Sort the eigenvalues and select the top k.
- **Step 4:** Create a new matrix that will contain the eigenvectors mapped to the k eigenvalues.
- **Step 5:** Obtain new features by taking the dot product of the data and the matrix from the previous step.
- **Step 6:** Visualize the processed data - The following figure shows the 2D and 3D visualizations of the image data after an application of LDA.

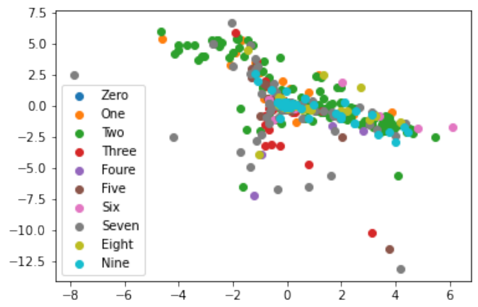

Fig 7: 2D

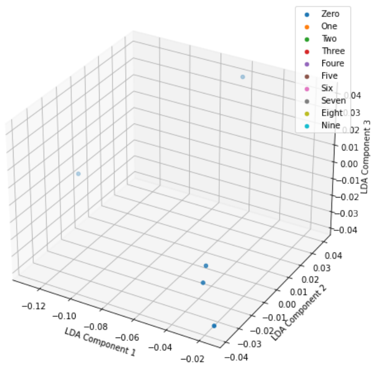

Fig 8: 3D

2.4 Classification using KNN

We have done the dimension reduction of the digits datasets using PCA and LDA, and now our data is ready for AI algorithms. We used k-Nearest Neighbours (KNN) algorithms. KNN is categorized in supervised learning algorithms, it is used for both regression and classification. It tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closet to the test data. In terms of our data, First, we separated the data in to training and testing data, then performed the following data preparation steps.

Data preparation for training and testing

- Prepared a DataFrame that contains both the features and the label of the dataset from the reduced one.
- Shuffled the dataset.
- Stored the labels into a variable y.
- Stored the pixel data in variable X.
- Split the data and the label, 20 percent for testing and 80 percent for training.

Handwritten Digit Classification using KNN- We used KNN to locate the nearby projections made by the training images in the following steps:

- **Step 1: Calculate Euclidean Distance :** We can calculate the straight line distance between two vectors using the Euclidean distance measure. It is calculated as the square root of the sum of the squared differences between the two vectors. The formula for euclidean distance is:
$$distance = \sqrt{\sum_{i=1}^{n}(p1_i - p2_i)^2}$$
- **Step 2:Get Nearest Neighbors :** Neighbors for a new piece of data in the dataset are the k closest instances(k=5). To locate the neighbors for a new piece of data within a dataset, we calculated the distance between each record in the dataset to the new piece of data. After the distance is calculated, we sorted all the records in the training dataset by their distance to the new data. Then we selected the top k to return the most similar neighbors.This is done by keeping track of the distance for each record in the dataset as a tuple, by sorting the list of tuples with the distance (in descending order) and then by retrieving the neighbors.
- **Step 3:Make Predictions :** Prediction can be made based on the similar neighbors collected from the training dataset.In classification, we can return the most represented class among the neighbors by performing the argmax() on the list of output values from the neighbors. Given a list of class values observed in the neighbors, argmax() takes a set of unique class values and calls the count on the list of class values for each class value in the set.
- **Step 4:Measure Accuracy:** The following table shows the accuracy levels with different numbers of eigenvectors for PCA and LDA.

No. of eigenvectors	2	3	5	10	20	50	100	200	300	500	700	780
Accuracy	39.08	50.56	71.56	85.10	87.10	88.62	85.52	81.9	79.11	77.86	77.14	77.66

Table 1: PCA Accuracy

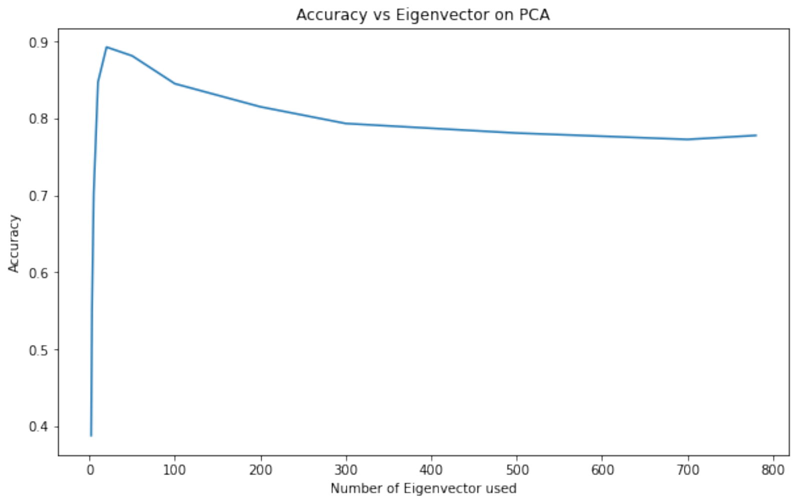

Fig 9: Accuracy in PCA

No. of eigenvectors	2	3	5	10	20	50	100	200	300	500	700	780
Accuracy	21.3	24.09	16.75	21.40	25.12	18.92	15.20	14.37	15.82	16.75	13.85	16.54

Table 2: LDA Accuracy

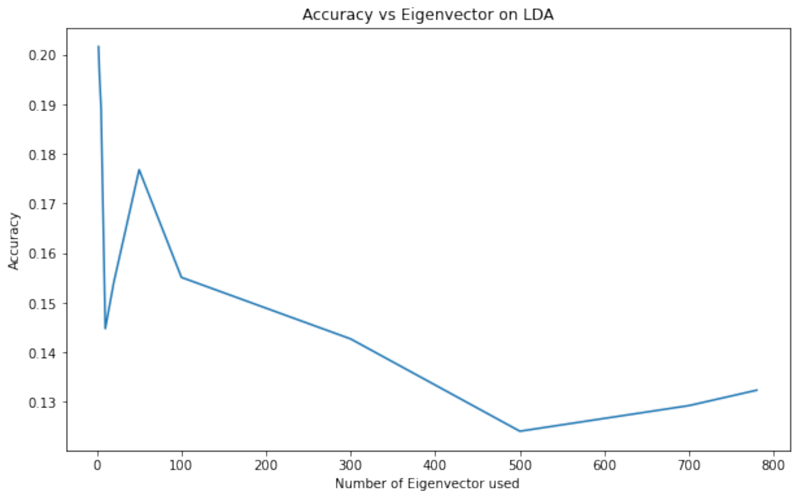

Fig 9: Accuracy in LDA

3 Conclusion

We have implemented the OCR on handwritten digits (0-9) using the dimensional reduction techniques of PCA and LDA from scratch. We have also implemented K-Nearest Neighbors from scratch and applied it to a classification problem.The PCA shows a better result than the LDA on our dataset. During this project we have learned a lot regarding image processing, matrix operation, collecting datasets from online, preparation and preprocessing the image datasets,eigenvalue and eigenvector calculation, deminationality reduction using PCA and LDA, and classifying the data using KNN.

References

A. O. Otiko, J. A. Odey, and G. A. Inyang. Handwritten digit recognition: A performance study of machine learning tools.

R. Sheikh, M. Patel, and A. Sinhal. Recognizing mnist handwritten data set using pca and lda. In *International Conference on Artificial Intelligence: Advances and Applications 2019*, pages 169–177. Springer, Singapore, 2020.