Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi

Group Members
Mohsin Zia 2020244
Muhammad Abdullah 2020256
Irtaza Haider Zaidi 2020474

Course Title
ES-304 Linear Algebra ll

Project Title
Principal Component Analysis (CEP)

Instructor Name
Professor Muti Ur Rehman

Dated

13/12/2022

CEP – Principal Component Analysis

I. Introduction

Principal Component Analysis (PCA) is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. Each of the principal components is chosen in such a way so that it would describe most of them still available variance and all these principal components are orthogonal to each other. In all principal components, first principal component has maximum variance. PCA is a linear dimensionality reduction technique/algorithm that transform a set of correlated variables (p) into a smaller k (k<p) number of uncorrelated variables called principal components while keeping as much of the variability in the original data as possible and minimizing information loss. PCA can also be used for image compression. PCA minimizes the size in bytes of an image while keeping as much of the quality of the image as possible.

II. Uses Of PCA

[1] It is used to find inter-relation between variables in the data.

[2] It is used to interpret and visualize data.

[3] The number of variables is decreasing so it makes further analysis simpler.

[4] It's often used to visualize genetic distance and relatedness between populations.

These are basically performed on a square symmetric matrix. It can be a pure sum of squares and cross-products matrix or Covariance matrix or Correlation matrix. A correlation matrix is used if the individual variance differs much.

III. Objectives

[1] It is basically a non-dependent procedure in which it reduces attribute space from a large number of variables to a smaller number of factors.

[2] PCA is basically a dimension reduction process but there is no guarantee that the dimension is interpretable.

[3] The main task in this PCA is to select a subset of variables from a larger set, based on which original variables have the highest correlation with the principal amount.

IV. Steps

PCA can be broken down into the following 5 steps.

[1] Standardization:

We standardize the data to let each feature have an equal contribution to the analysis. Imagine a regression line being plotted for data where x-axis values differ by small quantities like 0.3 to 0.9 (small variation in data), and y-axis values differ by amounts ranging in thousands (large variation in data). Such a line will be very skewed towards the y axis and will not be able to follow variance in data on the x-axis. Thus, we need to bring all the features to a comparable scale so that none of them is dominating. In standardization, we make the feature values to have mean 0 and standard deviation as 1.

[2] Covariance: We make a covariance matrix of the given data to find if there is any relationship between features. Because features may be highly correlated suggesting redundant information. Thus, we try to remove such features. So, to get this insight of highly correlating features we build a covariance matrix.

[3] Eigenvalues and Eigenvectors: This step helps us to determine the principal components of the data. This is done in such a way that the new feature values are uncorrelated and most of the information is made available in the first component. Then remaining in the next and so on. This way we can remove the less important principal components to reduce the dimensionality of the data. Geometrically, eigenvectors (characteristic vectors) are those vectors that do not change their direction when applied to a linear transformation and eigenvalues represent the amount by which they are scaled. So once we find the eigenvectors of the covariance matrix, we get the directions of the data that explain the maximum amount of variance depending on the eigenvalues (large eigenvalue means more variance in that eigenvector's direction). And since the covariance matrix is symmetric, the eigenvectors will be perpendicular. How convenient! so we can now use them as the axes to explain the dataset. Thus, we sort the eigenvectors based on their eigenvalues in descending order to get the important principal components first.

[4] Feature vectors: These are those vectors that make the cut. That is, the selected principal components that we think are sufficient to represent the data without the loss of much information.

[5] Projection to Principal Components: So far, we have just found the directions that best explain the data. Now we reorient the data from the original axes to the ones described by the principal components. This will give us the data with reduced dimensions.

Let's look at the math that happens behind the Principal Component Analysis by analyzing the 5 steps of PCA in Depth.

V. Mathematical Analysis Of PCA

Let's analyze all the steps to perform PCA using the mathematical and statistical concept on a small two-dimensional toy dataset and reduce its dimension into one. After completing all these mathematically step by step, we will do the same thing programmatically in python and validate our result.

Section [1] – Mathematics

[1.1] Matrix: Matrix is an array of numbers. Matrix may be square or rectangular shape.

[1.2] Determinant: It represents a number which can be calculated from a square matrix. |A| represents determinant of a matrix A.

[1.3] Eigenvectors: Eigenvector is a non-zero vector (say, x) which when multiplied with another square matrix (say, A)produces another matrix (also, a vector) which is an scalar multiple of the original vector x.

$$\mathbf{Ax = \lambda x}$$

[1.4] Eigenvalue: In the above equation lambda (**λ**) is called the eigenvalue of the eigenvector x.

SECTION [2] – STATISTICS

[2.1] Mean:

$$\text{Let } X = (X_1, X_2, \ldots\ldots, X_n) \text{ be a list of numbers.}$$

$$\bar{X} \text{ is the mean of the numbers, and } \bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

[2.2] Standard Deviation: (S) is the measure of dispersion of data from the mean in a list of numbers.

$$s = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}$$

[2.3] Variance: It's the square of Standard Deviation (S).

[2.4] Covariance: As variance depicts spread of data in one column, covariance is the joint form (related to two columns in a dataset) of the same thing.

$$\text{var}(X) = \text{cov}(X, X) = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X})}{n-1}$$

[2.5] Z-Score: We can standardize or scale a group of observed values using z-score for which the mean is deducted from each observed value and then divided by the standard deviation. The mean of the new observed data set becomes 0 after applying Z-score.

$$Z = \frac{X_i - \bar{X}}{S_x}$$

X_i = Observed value

$\bar{X}$ = Mean of the group

S_x = Standard Deviation of the group

SECTION [3] – MATHEMATICAL EXAMPLE

Let us consider a 2-D dataset with 2 columns (features) and 6 rows. Columns names are X and Y. Here (2,3), (4,5), (6,5), (6,7), (7,8), (5,8) are called feature vectors.

X	Y
2	3
4	5
6	5
6	7
7	8
5	8

Step [3.1] Standardize:

We can standardize or scale the data using z-score. Let's calculate corresponding mean and standard deviation for each column.

$$\text{Mean (X)} = \bar{X} = (2+4+6+6+7+5)/6 = 5$$
$$\text{Mean (Y)} = \bar{Y} = (3+5+5+7+8+8)/6 = 6$$

$$S_x = \sqrt{((2-5)^2 + (4-5)^2 + (6-5)^2 + (6-5)^2 + (7-5)^2 + (5-5)^2)/6}$$
$$= \sqrt{(9+1+1+1+4+0)/6}$$
$$= \sqrt{16/6} = 1.633$$

$$S_y = \sqrt{((3-6)^2 + (5-6)^2 + (5-6)^2 + (7-6)^2 + (8-6)^2 + (8-6)^2)/6}$$
$$= \sqrt{(9+1+1+1+4+4)/6}$$
$$= \sqrt{20/6} = 1.826$$

S_x and S_y are the standard deviations of coulmn X and Y respectively.

Original data

X	Y
2	3
4	5
6	5
6	7
7	8
5	8

Z-Score calculation

X	Y
(2-5)/1.633	(3-6)/1.826
(4-5)/1.633	(5-6)/ 1.826
(6-5)/1.633	(5-6)/ 1.826
(6-5)/1.633	(7-6)/ 1.826
(7-5)/1.633	(8-6)/ 1.826
(5-5)/1.633	(8-6)/ 1.826

=

Scaled data

X	Y
-1.837	-1.643
-0.612	-0.548
0.612	-0.548
0.612	0.548
1.225	1.095
0	1.095

Step [3.2] Covariance Matrix:

$$\text{Cov(X,X)} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{n-1} = \frac{\sum_{i=1}^{n} X_i^2}{n-1}$$
$$= ((-1.837)^2 + (-0.612)^2 + (0.612)^2 + (0.612)^2 + (1.225)^2 + 0)/5$$
$$= 1.2$$

$$\text{Cov(Y,Y)} = \frac{\sum_{i=1}^{n}(Y_i - \bar{Y})(Y_i - \bar{Y})}{n-1} = \frac{\sum_{i=1}^{n} Y_i^2}{n-1}$$
$$= ((-1.643)^2 + (-0.548)^2 + (-0.548)^2 + (0.548)^2 + (1.095)^2 + (1.095)^2)/5$$
$$= 1.2$$

$$\text{Cov(X,Y)} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1} = \frac{\sum_{i=1}^{n} X_i Y_i}{n-1}$$
$$= ((-1.837)(-1.643) + (-0.612)(-0.548) + (0.612)(-0.548) + (0.612)(0.548) + (1.225)(1.095) + 0)/5$$
$$= 0.939$$

We know, Cov(Y,X) = Cov(X,Y)

We know covariance matrix of X, Y is

$$\text{Covariance Matrix} = \begin{bmatrix} Cov(X,X) & Cov(X,Y) \\ Cov(Y,X) & Cov(Y,Y) \end{bmatrix}$$
$$= \begin{bmatrix} 1.2 & 0.939 \\ 0.939 & 1.2 \end{bmatrix}$$

Step [3.3] Eigenvalues:

Now, the Eigen value of our covariance matrix will be

$$A - \lambda I = \begin{bmatrix} 1.2 & 0.939 \\ 0.939 & 1.2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.2 & 0.939 \\ 0.939 & 1.2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$
$$= \begin{bmatrix} 1.2-\lambda & 0.939 \\ 0.939 & 1.2-\lambda \end{bmatrix}$$

$$\text{So, } \begin{vmatrix} 1.2-\lambda & 0.939 \\ 0.939 & 1.2-\lambda \end{vmatrix} = 0$$

$$\Rightarrow (1.2-\lambda)(1.2-\lambda) - (0.939) \times (0.939) = 0$$

$$\Rightarrow 1.44 - (2 \times 1.2)\lambda + \lambda^2 - 0.8817 = 0$$

$$\Rightarrow \lambda^2 - 2.4\lambda + 0.5583 = 0$$

Solving for λ we get $\lambda = 2.139$ and $\lambda = 0.261$

So, two Eigen values are 2.139 and 0.261

Step [3.4] Eigenvectors:

$$Eigen\ Vector = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}$$

Step [3.5] Projection of data one 1D:

As our main aim to perform PCA is to project the data from higher dimension to lower dimension, let's continue with our example and project the scaled data to one dimension using the Eigen vector found above.

(Transpose of Eigen vector) X (Feature vector)

Scaled data

X	Y
-1.837	-1.643
-0.612	-0.548
0.612	-0.548
0.612	0.548
1.225	1.095
0	1.095

=

Projected Data
-2.461
-0.820
0.046
0.820
1.640
0.774

From PCA we come up with the above projected data in one dimension.

SECTION [4] – PCA USING PYTHON

Step [4.1] Prepare Date:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

data = np.array([[2,3],[4,5],[6,5],[6,7],[7,8],[5,8]])
df = pd.DataFrame(data=data, columns=['X','Y'])
print("Our toy dataset:")
df
```

Our toy dataset:

	X	Y
0	2	3
1	4	5
2	6	5
3	6	7
4	7	8
5	5	8

Step [4.2] Standardize Data:

```
from scipy.stats import zscore
df_scaled = df.apply(zscore)
print("Scaled Data:")
df_scaled
```

Scaled Data:

	X	Y
0	-1.837117	-1.643168
1	-0.612372	-0.547723
2	0.612372	-0.547723
3	0.612372	0.547723
4	1.224745	1.095445
5	0.000000	1.095445

Step [4.3] Covariance:

```
cov = np.cov(df_scaled, rowvar=False)
print("Covariance Matrix:")
print(cov)
```

```
Covariance Matrix:
[[1.2        0.93914855]
 [0.93914855 1.2       ]]
```

Step [4.4] Fit 2D Data and Eigen Values:

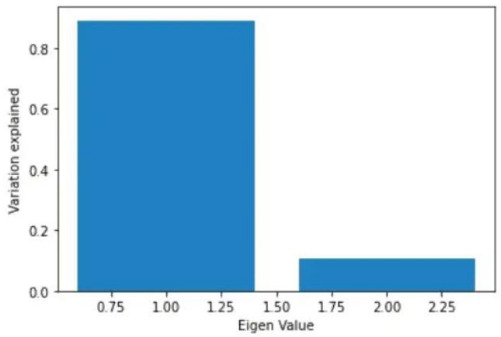
```
pca = PCA(n_components=2)
pca.fit(df_scaled)

print("Eigen values:")
print(pca.explained_variance_)
```

```
Eigen values:
[2.13914855 0.26085145]
```

Step [4.5] Plot of Variance Ratio:

```
plt.bar(list(range(1,3)),pca.explained_variance_ratio_)
plt.ylabel('Variation explained')
plt.xlabel('Eigen Value')
plt.show()
```



Step [4.6] Project Data into new single dimension:

```
pca3 = PCA(n_components=1)
pca3.fit(df_scaled)
Xpca3 = pca3.transform(df_scaled)
print("Projected data: ")
print(Xpca3)
```

Scaled Data:

	X	Y
0	-1.837117	-1.643168
1	-0.612372	-0.547723
2	0.612372	-0.547723
3	0.612372	0.547723
4	1.224745	1.095445
5	0.000000	1.095445

=

```
Projected data:
[[ 2.46093311]
 [ 0.82031104]
 [-0.04571437]
 [-0.82031104]
 [-1.64062207]
 [-0.77459667]]
```

Hence, we can see that we have obtained the same data (approx.) by doing it mathematically by hand and programmatically. The only difference is the sign. Both are valid as python considered the negative values of the Eigen vectors while we considered the positive values. But both are correct as long as the Eigen vector is normalized to have length 1. Now, you can easily understand what the mathematics behind the PCA algorithm is.

SECTION [5] – CONCLUSION

We have tried to show you PCA in a very small 2-dimensional dataset. In real life scenario, there are large dataset with huge number of dimensions (column or feature, whatever you say) where PCA is useful when you do not know which of the features are not adding value to your final model. After you perform PCA in the large dataset and reduce its dimension (that's why PCA is called, Dimensionality Reduction technique) it will be easier for visualization and will be ready for your further modelling.

As we have understood the concept of PCA by this two-dimensional data set, we can now apply these techniques for larger datasets. Let's apply PCA on Landsat images by using the concepts of PCA we've just understood.

PHASE [1] OF PCA

I. INTRODUCTION

This is the first phase of the report which will focus on applying PCA using library functions. This phase has 2 parts, one in which we compress the main image and the other in which we stretch the image after combining the 7 bands. The project has been coded in the python programming language and the library used to perform the PCA is called scikit learn. In this part we will be using PCA, which is an unsupervised machine learning algorithm, for applying dimensionality reduction on the image of our chosen location.

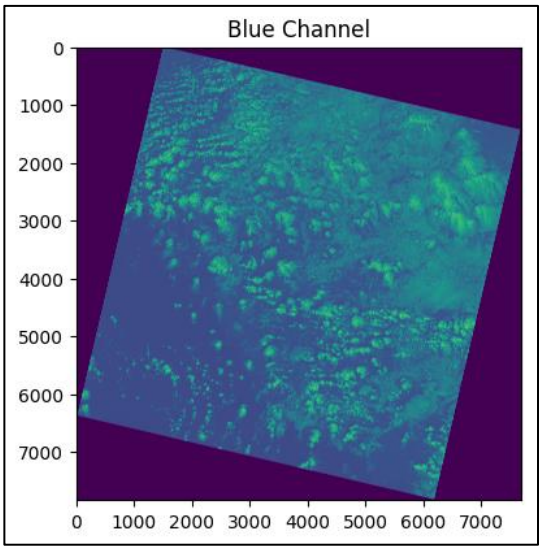
II. SPLITTING

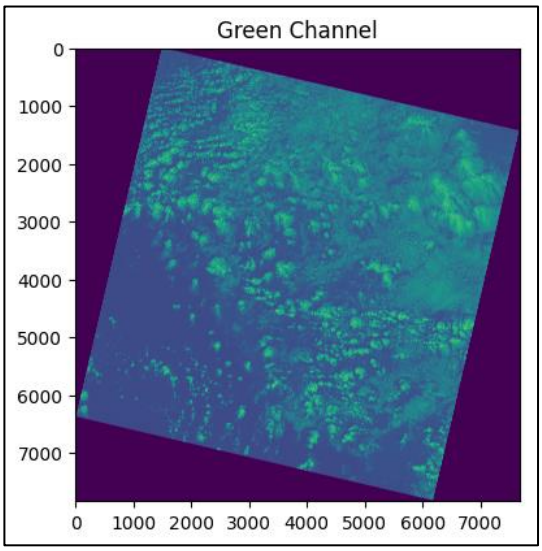
A. *Preprocessing*

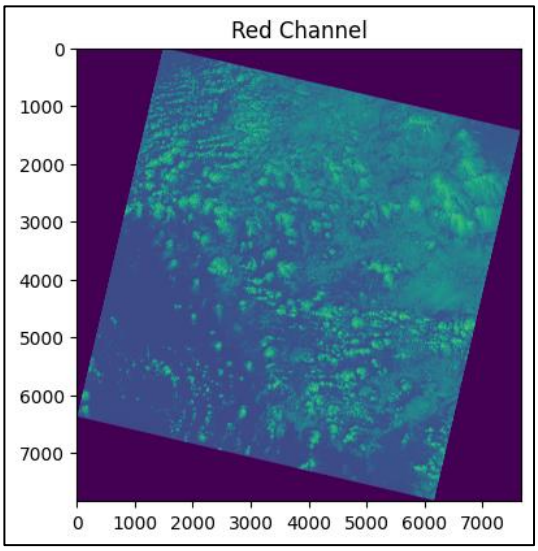
First of all, we import and load all the libraries that are required for assisting and performing PCA on our image. These contain NumPy and Pandas which are used to store data in the form of tables and datasets, OpenCV to open images with python, matplotlib for plotting graphs, and the main scikit learn library which contains the functions that will be used to perform the PCA.

B. *Splitting the image into channels*

We will then split the into red, green and blue channels (r, g, b) and then perform PCA separately on these datasets. We will then merge these datasets to recreate the compressed image. We are splitting the image into red, green and blue channels because the variance is better using the splitting method.







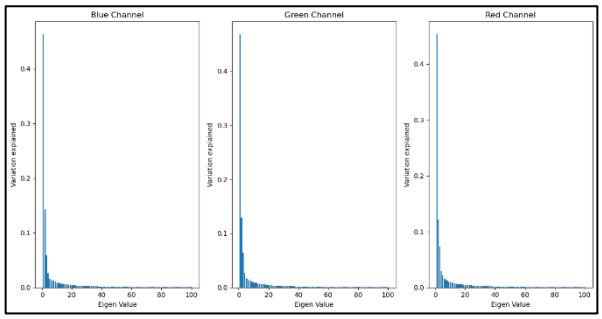
C. *Normalizing*

After splitting and storing the different channels into their respective datasets, we normalize the data by dividing it by 256 so that it is scaled between 0 and 1.

III. PERFORMING PCA

Now for performing PCA we will only consider 100 dimensions from our original 7831x7681 dimension image, then we perform PCA on each of our 3 channel data sets. Choosing only 100 dimensions allows us to keep around 94 percent of variance in our data. Choosing even less dimensions reduces the variance resulting in more dimension reduction of our image. Reducing large dimensions results in a more blurred image.

After performing the PCA we can see the variance explained across the 3 rgb bands and their eigenvalues.



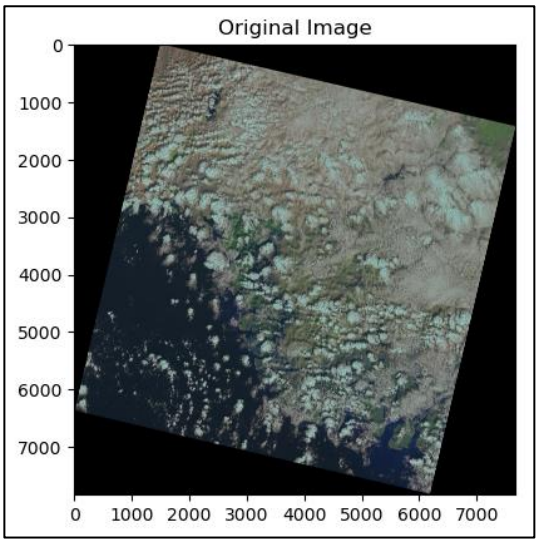
IV. RECREATING THE IMAGE

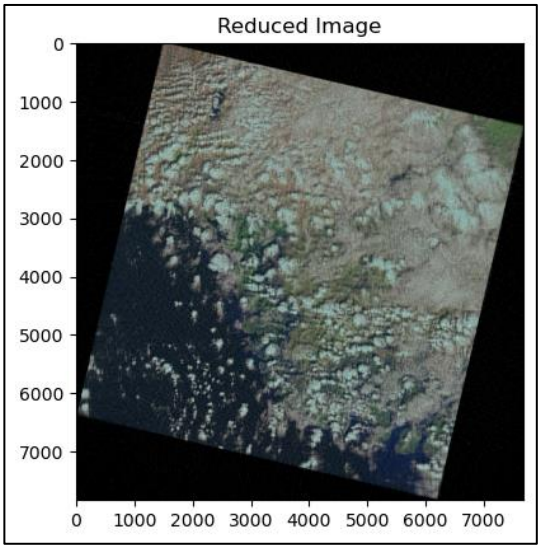
A. *Reverse transform the image*

Before merging all the channels data together, we must retransform them into their original dimensions which were 7831x7681. For this we reverse transform the datasets which are now compressed by PCA and then merge them all together resulting in a compressed image.

B. *Visualizing*

Now we can visualize and compare the images by keeping the original and the compressed image side by side.





This shows that after compressing the images we can still recognize the compressed image even after reducing the dimension of each channel to 100 from 7831.

V. ERROR ANALYSIS AND CONCLUSION

Initially we had a dimension of 7831x7681 before applying PCA. After performing PCA with 100 components the dimensions reduced to 7831×100. Earlier, to store 3 images which were split into RGB, we required a 3×7831×7681 matrix. We need to store 3×100 matrix for reduced image and 3×7831×100 matrix for storing principal components. As the image is gray scale, let's suppose it requires 2 bits to store each pixel of an image. Therefore, after compressing the image, we will be able to save:

= [(3×7831×7681) × 2)] – [(3×100) + (3×7831×100)]×2
= 356200266 bits
= 44525033.25 bytes
= **44.52 MB**

The most suitable number of components in our case were 100 components as they contain approximately 94% of the variance. In 50 components we get about 89% of the variance. In 200 components we get around 96% variance. And in 500 components we get 99% variance. Therefore, the minimum number of components for which we had minimum data loss seems to be 100.

IMAGE STRETCHING USING PCA

I. INTRODUCTION

This is the second part of the first phase. IN this part we will be stretching the image after combining the 7 bands.

The way stretching works is that first we take the whole image's pixel value range and then stretch it to between 0 and 255. Doing so brightens a dark image or lessens the brightness of a very bright image. so that its content become visible.

II. COMBINING

A. *Preprocessing*

First we import all the required libraries that are required to stretch the image. The main libraries we use here are xarray, rioxarray and earthpy. The xarray and rioxarray libraries allow us to store all the useful data in the 7 bands in a single array, we can then plot these bands and apply different functions on them. The earthpy library allows us to plot the combined band data as a normal composite image or even the stretch and plot the image.

B. *Combining the bands*

We first open all the images in python and store them in xarray's and then concatenate these arrays into a single xarray.

III. PLOTTING

We can now plot this composite data xarray as an image using the earthpy library.

We can also stretch and plot the image using the built in stretch option in the earthpy library.

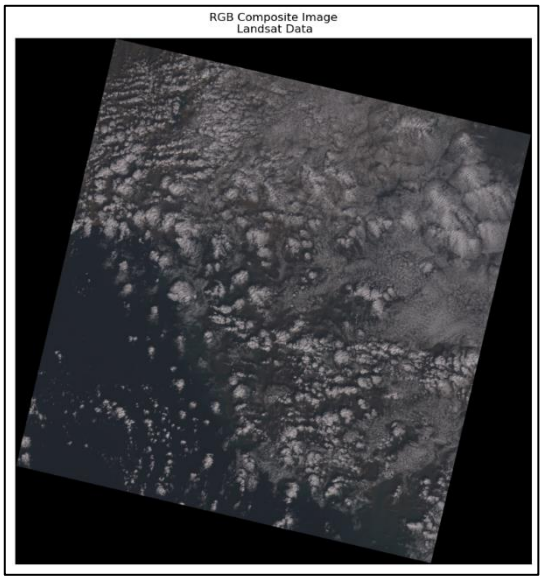


Figure 1 Composite image without stretching

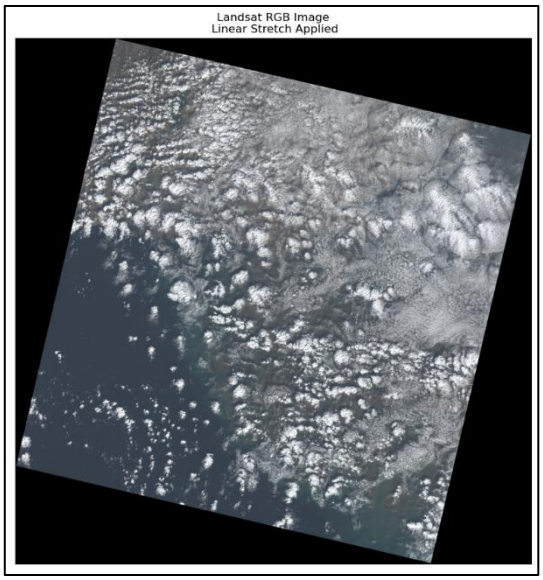


Figure 2 Composite image with stretching applied.

We can see that in figure 2 after applying stretching on the pixels of the composite image, the resulting image is brighter than before.

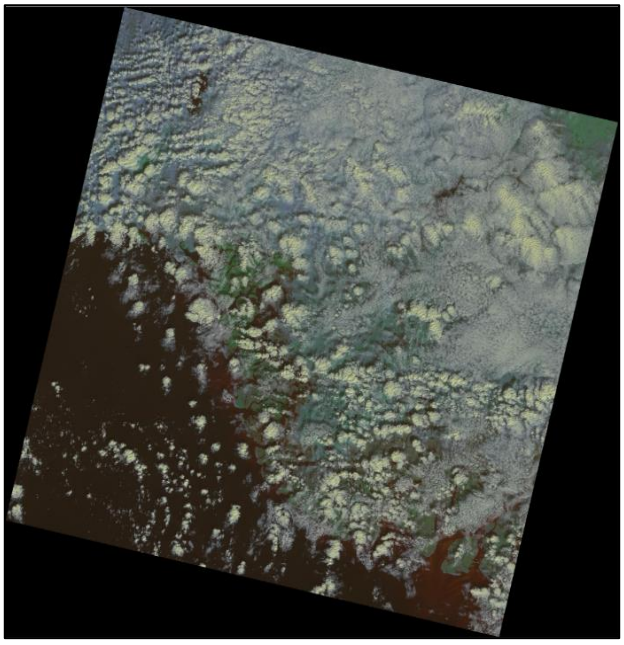
PHASE [2] OF PCA

I. INTRODUCTION

Technically, an image is a matrix of pixels whose brightness represents the reflectance of surface features within that pixel. The reflectance value ranges from 0 to 255 for an 8-bit integer image. So the pixels with zero reflectance would appear as black, pixels with value 255 appear as pure white and pixels with value in-between appear in a gray tone. The transformation is applied in such a way that linearly correlated variables get transformed into uncorrelated variables. Correlation tells us that there is a redundancy of information and if this redundancy can be reduced, then information can be compressed. For example, if there are two variables in the variable set which are highly correlated, then, we are not gaining any extra information by retaining both the variables because one can be nearly expressed as the linear combination of the other. In such cases, PCA transfers the variance of the second variable onto the first variable by translation and rotation of original axes and projecting data onto new axes. The direction of projection is determined using eigenvalues and eigenvectors.

II. LOADING MODULES AND IMAGE DATA

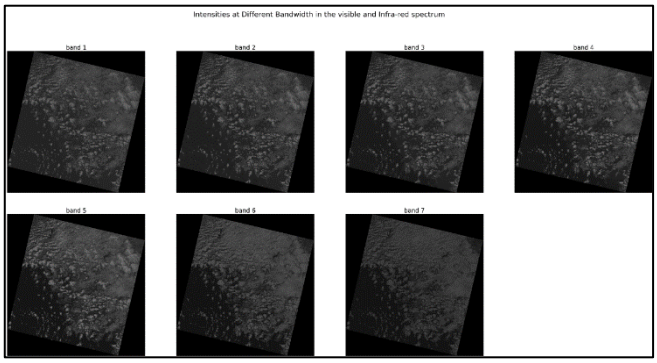
The first step is to import the required libraries and load data. To make accessibility and processing easier, the band images are stacked in a 3d NumPy array of sizes 7831 x 7681 x 7 (height x width x no of bands). The color image shown below is a composite of Red, Green, and Blue (RGB) band images, reproducing the same view as it would have appeared to us. Get a glimpse of the scene.



III. DATA EXPLORATION

If we observe the images, all bands have captured one or more surface features and each feature is captured well in multiple bands. There exists redundancy of information between the bands which means reflectance's are somewhat correlated

across bands. This gives us the right opportunity to test PCA on them.



IV. Data Standardization

Before applying PCA, we must bring our data to a common format through standardization. The purpose of doing this is to make sure that variables are internally consistent with each other regardless of their type. For example, if a dataset has two variables temperature measured in degrees Celsius and rainfall measured in cm. Since the variables range and units are different, we shouldn't use dissimilar variables as they are, otherwise, variables differing in order of magnitude may introduce a model bias towards some variables. Standardization is done by centering the variable by finding z-score. Since the variables (band images) we are dealing with are similar and have the same range, standardization is not necessary but still, it is a good practice to apply.

V. Eigen Values and Vectors Computation

In this step, data compression and dimensionality reduction come into the picture. Eigenvalues give us the order of significance of eigenvectors or directions. The next step is to order eigenvectors by their eigenvalue, highest to lowest, to rearrange principal components in order of significance. We need to transform the data in the direction of ordered eigenvectors which in turn results in principal components.

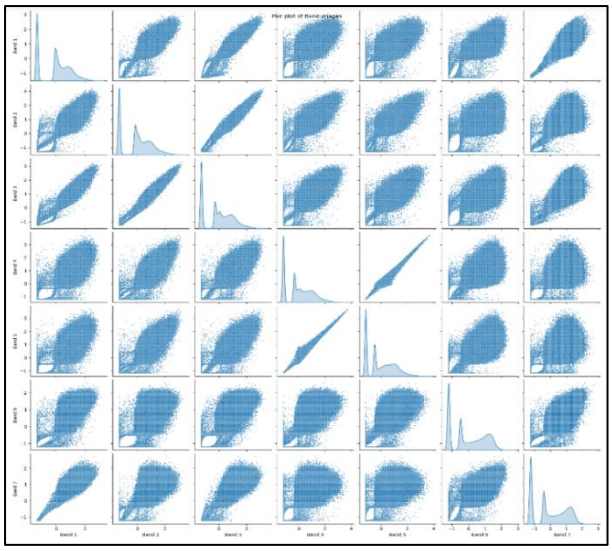
Eigenvalues:

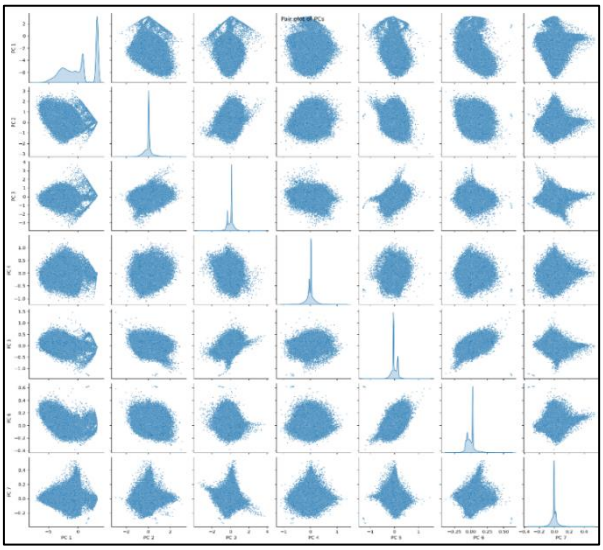
[6.880e+00, 8.987e-02, 2.573e-02, 3.117e-03, 1.163e-03, 1.963e-04, 3.427e-04]

VI. Validation of Principal Components

A. Dependency Check

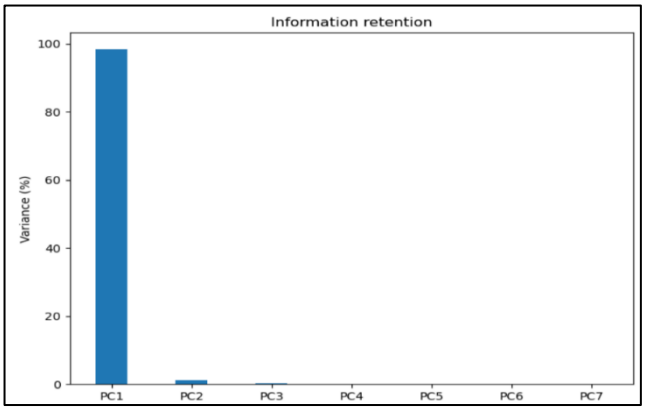
We were able to produce principal components successfully. Now, Let's validate the PCs to check whether they were able to reduce redundancy and also check the extent to which data compression was achieved. we will create scatter plots to visualize the pairwise relationship in the original bands and compare the same with the pairwise relationship of PCs to test for the existence of dependency.





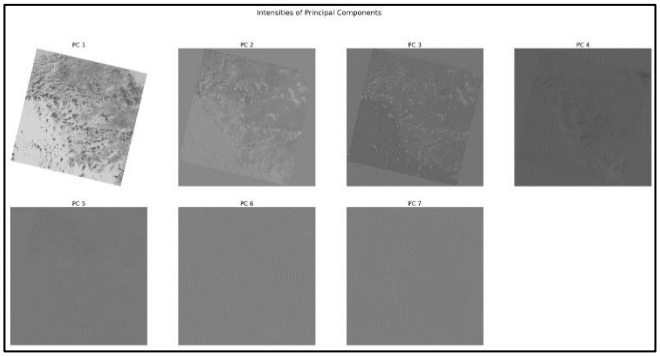
B. Compressibility Check

The Bar plot of Eigenvalues expressed in percentage plotted above gives us the information retained in each PC. Notice that the last PCs eigenvalues are small and less significant, this is where dimensionality reduction comes into play. If we choose to keep the first three relevant components that retain 98% information, then the final data can be reduced from 7 dimensions to 3 without losing much information.



VII. Converting PCs back to Images

It's time to reshape 1-d PCs back to the original image shape and normalize PCs between 0 and 255 which is the same as the original image range to make image visualization possible.



VIII. PC and RGB Image Comparision

Finally, we reproduced the same scene using the first three Principal components. The image on the left appears more colorful than the original image RGB which makes the features in the scene appear clearer and more distinguishable from each other. For example, farmland can be more easily distinguishable from urban areas because of differentiable colors. Some features also appear to be more prominent in PC images that are difficult to identify in the left image. So, it can be concluded that the PCA has done a great job on our Image data in terms of compressibility and information retention.