# Dimensionality Reduction through the use of Principal Component Analysis and Eigenvalue Decomposition

Brian So, 20477254

Professor Birkett, SYDE 312
April 1, 2018

**Abstract**

Principal Component Analysis is a widely used dimensionality reduction technique which has applications in the field of image compression and reconstruction through the use of linear algebra through eigenvalue decomposition and orthogonal transformations.

## *I. Background and Discussion of application area*

One important area of focus in image processing is the idea of image preprocessing through image compression and dimensionality reduction. A typical image on the internet is represented by a multi dimensional array of the 3 additive primary colours [1] Red, Green and Blue. Each RGB image found online is made up of individual pixels which have exactly 3 values each for the red green and blue channels. By leveraging this realization that an image is simply a multi dimensional array of 3-tuples, it is possible to reduce the amount of information that this image has and only focus on the important components through linear algebra techniques involving dimensionality reduction. Image compression has many desirable benefits such as reducing the time of file transfer between clients through image compression, as well as improving machine learning classification techniques [2] .

## *II. Relevance of linear algebra to the analysis*

In order to compute the values of the compressed image, it is necessary to utilize a technique called Principal Component Analysis. The goal of principal Component Analysis is to extract relevant information from complex and confusing data sets. It does so by reducing the dimension of of a data set [2]. PCA methods aim to identify basis which align with an axis of the data that produces the maximum variance between the data points and the basis. An example of this is shown in figure 3.

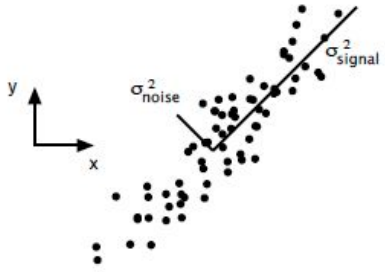


Fig 3. Example of Identified Basis with maximum variance (signal)

It can be seen from figure 3 that the basis which maximizes the signal (or variance) is the component labeled $\sigma^2_{signal}$ . In other words, PCA is finding an orthogonal axes that diagonalizes the covariance matrix C.

$$C' = ACA^T \quad (1)$$

From equation 1, the rows of matrix A will contain the eigenvectors which will act as the basis which maximize the variance,

and C` will contain the eigenvalues as the trace values in sorted order from highest to lowest variance. It is also worth mentioning that the mean must be subtracted from the data points before PCA can be performed [2].

*III. Presentation of a specific case study or problem (real or invented).*

A specific problem that utilizes this method is the problem of image compression as mentioned in part 1. The problem defined is that given an input image of size 350x780 pixels, with each pixel containing 3 values - an integer value of one of Red/Blue/Green channels, compress the image through PCA by 20x then uncompress the image by reconstructing the image to the original dimensionality by utilizing the calculated components and mean from the original image. For this problem, Python is utilized as the language of choice as it provides various built in functions to obtain the eigenvectors/eigenvalues from the covariance matrix of the input data, it also contains functions that will help separate an image into each color channel. This is a problem which was examined in SYDE-522, the source code for which can be found at the github link [3].

*IV . Mathematical solution and selection of appropriate numerical technique(s).*

The first step of this problem would be to separate the image into its 3 colour channels and individually compress each channel to a lower dimensional space. Each channel of the image is extracted and it will be divided into 10x10 pixel patches, each patch will then be flattened into a 100-dimensional array thus there will be 35x78=2730 of these arrays - these vectors, variable named as ‘img_ch’’ will have PCA applied to them. In order to “compress” these vectors by 20x, there must be 5 components found which maximizes the variance across the 2730x100 array.

```
def compress(img_ch, n_components=5):
    patches = patchify(img_ch)
    pca =
PCA(n_components=n_components)
    y = pca.fit_transform(patches)
    components = pca.components_ #
Obtain eigenvectors as components,
obtained through EVD on the covariance
matrix of the input matrix
    mean = pca.mean_ # Obtain the mean
of each datapoint
    img_size = img_ch.shape
    comp_data = {
        'y' : y,
        'aux_data' : [components, mean,
img_size]
    }
    # apply PCA with n=n_components on
the patches
    # return dictionary {'y':
compressed data, 'aux_data': auxiliary
data for reconstruction}
    return comp_data
```

During reconstruction, the mean of each data point will be added to the projection of the data points onto the component vectors.

```
def inverse_transform(y, components,
means):
    decompressed_data = np.dot(y,
components)+means
    return decompressed_data
```

This compression/decompression process is done for each colour channel and the

results are stacked together in python to reproduce the output image.

### *V. Interpretation of the results in terms of the application.*

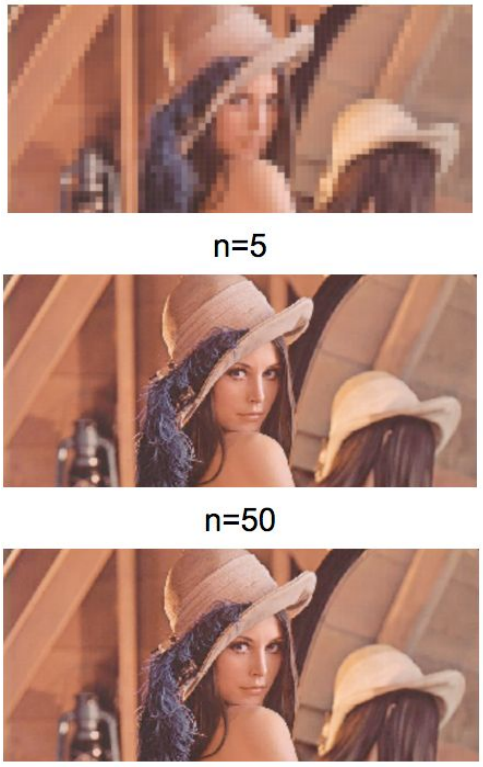


Figure 4. Image compression/decompression results with number of components

The results of image compression and reconstruction is shown in Figure 4. It can be seen that image reconstruction quality increases with more components utilized during image compression. After 50 components, it can be observed that it the reconstructed image begins to very closely resemble the original image (n=100).

### *VI. Conclusions*

PCA Is a very powerful and widely used dimensionality reduction technique which has many applications. One of which that is shown in this paper is the application thereof in image compression and reconstruction. It was shown that image compression is achievable by obtaining the eigenvectors (principal components) corresponding to the covariance matrix of a given matrix.

### *VII . References*

[1] Additive Colour Mixing, pages.cs.wisc.edu/~dyer/ah336/papers/07_additive-color.pdf

[2] A Tutorial on Principal Component Analysis, https://arxiv.org/abs/1404.1100

[3] PCA SYDE-522, https://github.com/bcso/PCA