Computing for Data Sciences

Lecture 9

(Principal Component Analysis, Image Compression and Frobenius Norm)

Principal component analysis (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables or the original variables in the data into a set of values of linearly uncorrelated new variables called **principal components**.

$$\underset{a_1, a_2, a_3, \ldots, a_n}{\text{Original attributes}} \xrightarrow{\text{PCA}} \underset{A_1, A_2, A_3, \ldots, A_n}{\text{Morphed set of new attributes}}$$

This process is basically a dimension reduction technique. The number of principal components which we try to get is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are uncorrelated orthogonal basis vectors.

$$A_i = L_1^{(i)} a_1 + L_1^{(i)} a_2 + \ldots\ldots\ldots + L_n^{(i)} a_n$$

The principal components are orthogonal because they are the eigenvectors of the covariance matrix, which is symmetric. PCA is sensitive to the relative scaling of the original variables.

The main task is to identify the component with maximum variance. We do that by observing the loadings of the PCA. Loadings are the factors which show what amount of load is shared by a particular feature. So, the component with maximum loading value can be considered as the most influential feature. The loadings are arranged in such a manner that the higher significance factors are placed first, followed by subsequent factors. On the basis of our tolerance level or precision level required, we select the number of components to be considered.
One important thing for PCA to get started is to get the same type of data sets, and therefore, we normalize the original data. In order to capture the variances, we need to identify the distribution of data, and therefore we transform the data in to normal distribution. Variance is a nice measure of letting us know the kind of distribution. If we can maximize the variance, then our information is maximized. PCA can be done on any distribution, but as variance measure is best parametric measure for normal distribution and also, datasets generally tends to follow normal distribution or can be transformed easily to be made to follow normal distribution, so we normally take normal distribution variance analysis in PCA.

To capture the variance of each feature with respect to other feature we try to get the variance - covariance matrix of the features, followed by finding out the eigenvalues of the matrix and then finding the eigenvectors of the matrix of dataset which gives the various principal components.

Applications of PCA:

1. **PCA application in gene technology** - we have lots of genes types as dataset, so if we need to compare it to some specific gene type to know about the genes, we will use PCA to compare some specific components rather than comparing the whole data set.
2. **PCA for image compression** - we have to compress images for proper storage or upload. Here, we can use initial highest weighted components to compare the photos and we can use these principal components to compress the image so that photos can be stored in less number of pixels.

Various other fields where PCA is used are:

- Medicine development, manufacturing and quality assistance as chemometrics in pharmaceutics
- Ecological areas - evaluation of aquatic habitat suitability, its regionalization, analysis of fish abundance
- Principal Component Analysis to the Detection and Visualization of Computer Network Attacks

Limitations of PCA

- Directions with the largest variance is assumed as the basis
- We only consider orthogonal transformations (rotations) of the original variables. (Kernel PCA is an extension of PCA that allows non-linear mappings)
- It is based only on the mean vector and the covariance matrix of the data. Some distributions might not be completely characterized by it.
- Dimension reduction can only be achieved if the original variables were correlated. If the original variables were uncorrelated, PCA does nothing, except for ordering them according to their variance.
- PCA is not scale invariant

E.g., suppose we've the measurements in kg and meters. And we want to express PCA in terms of milligrams and decameters. Then we have 2 ways: Either by multiplying kg measurements by 10^6, and multiplying meters by 10^{-1} and then applying PCA or by applying PCA on both the measurements and then rescaling to the appropriate units.
The result for these 2 different ways will generate different result. The reason for the above is because eigenvectors are not scale invariant.

Image Compression

An Image can be considered as a matrix or a collection of vectors. A matrix which contains values between 0 and 255 has no structure. But a grey scale image, on the other hand, has a

structure, like contrast points. The structure of an image cannot be determined by just its column or row vectors. Each vector has different values at different points. If we consider the row vectors to be attributes, then each column becomes an observation and each pixel is a value of that observation. Thus, we have a vector space. In the vector space, it has a dimension. For the example, it is 800 x 547 picture and we do not which is the dimension.

A histogram of a picture is never a flat plot. It always shows peaks and hence, variance is observed. A picture with a flat plot has no contrast and thus no information. Therefore, variance can be used to determine the structure of the image. To utilize this, we can use PCA to determine the direction in which most of the information about the picture is stored.

Example: A 800 x 547 image, where the number of attributes is 547. It is decomposed into an 800 x n/547 image, where the compression ratio is given by (800 x n/547)/800 x 547. A screen plot function gives us the variance covered by the principal components.

In R, we calculate the PCA of the image using the *princomp* function. Applying the *screenplot* function to the new image obtained tells us the variance captured by each of the components.

We can also use SVD to get the principal components. When we call SVD in R, it returns the U, V and Σ matrices. Even though Σ matrix is a diagonal matrix, R returns only the diagonal elements of the matrix rather than the whole matrix. We make it into a diagonal matrix by using the *diag* function and multiply U, V and Σ to reconstruct the original image.

In the above process, the image was decomposed to three different matrices and then combined back together. So, the final output is the original image without any loss in information.

To find N principal components, N number of rows from U, N elements from Σ and N columns of the V matrix are taken and combined. An image with 547 components is now converted into one with N components. These N components may be a combination of some or all the original components of the original image. The compression ratio is given by N/547. Thus, we have N/547 from each U,V and Σ matrix. The image is stored with dimensions 800x 3N/547. The image viewer will convert this into a format that is visible, which here is 800x547.

The reduction is dimension cannot be done simultaneously in both the directions. Either we can reduce it to N from 800 or from 547. Reducing both the number of attributes and the number of observations gives us less information and thus, any further decomposition may not give the whole image as such.

We observe that as N increases, the clarity of the reconstructed image increases. This principle can be used in a webpage where the image keeps becoming clearer with time. This

is done by super-imposing the different principal components over each other to show a change in the image clarity.

The PCA image does not preserve all the properties of the image like borders, etc. It, however, preserves the position of eyes, nose, etc.

PCA is usually used for classification purposes. So, a high contrast image can easily be compressed using PCA. On the other hand, say we have an image with red and blue components super-imposed over each other, with the same intensity, PCA cannot be used. PCA can be used for segmentation too, but with some constraints.

Frobenius Norm

In order to find the difference between the original image and the compressed image, we can use a concept called Frobenius norm. The Frobenius norm can be defined as a matrix norm of matrix $A_{m \times n}$ defined as the square root of the sum of the absolute squares of its elements

$$||A||_F = \sqrt{\sum_i \sum_j a_{ij}^2} = \sqrt{trace\ (A * A)}$$

$$||A - A'_k||_F = \sqrt{\sum_i \sum_j (a_{ij} - a_{ij}^k)^2}$$

Frobenius norm is equal to sum of L_2 norm squared for each row.

$L_2(r_1)^2$
$L_2(r_2)^2$
⋮

Example: The Frobenius norm for the following matrix will be

$$\begin{pmatrix} 2 & -1 \\ -4 & -2 \end{pmatrix}$$

$$||\boldsymbol{A}||_F = [(2)^2 + (-1)^2 + (-4)^2 + (-2)^2]^{1/2} = 5$$

Advantage of Frobenius norm is that it is often easier to compute and is invariant under rotations, that is,

$||\boldsymbol{A}||_F^2 = ||\boldsymbol{AR}||_F^2 = ||\boldsymbol{RA}||_F^2$,for any rotation matrix **R**.