



# SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Coimbatore – 35.



## DEPARTMENT OF BIOMEDICAL ENGINEERING

### UNIT – 3

#### Principal Component Analysis

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning. It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. It is one of the popular tools that is used for exploratory data analysis and predictive modeling. It is a technique to draw strong patterns from the given dataset by reducing the variances.

PCA generally tries to find the lower-dimensional surface to project the high-dimensional data.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation system, optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

The PCA algorithm is based on some mathematical concepts such as:

#### **Variance and Covariance**

#### **Eigenvalues and Eigen factors**

Some common terms used in PCA algorithm:

**Dimensionality:** It is the number of features or variables present in the given dataset. More easily, it is the number of columns present in the dataset.

**Correlation:** It signifies that how strongly two variables are related to each other. Such as if one changes, the other variable also gets changed. The correlation value ranges from -1 to +1. Here, -1 occurs if variables are inversely proportional to each other, and +1 indicates that variables are directly proportional to each other.

**Orthogonal:** It defines that variables are not correlated to each other, and hence the correlation between the pair of variables is zero.

**Eigenvectors:** If there is a square matrix  $M$ , and a non-zero vector  $v$  is given. Then  $v$  will be eigenvector if  $Av$  is the scalar multiple of  $v$ .

**Covariance Matrix:** A matrix containing the covariance between the pair of variables is called the Covariance Matrix.

Principal Components PCA:

As described above, the transformed new features or the output of PCA are the Principal Components. The number of these PCs are either equal to or less than the original features present in the dataset. Some properties of these principal components are given below:

The principal component must be the linear combination of the original features. These components are orthogonal, i.e., the correlation between a pair of variables is zero. The importance of each component decreases when going from 1 to  $n$ , it means the 1st PC has the most importance, and  $n$ th PC will have the least importance.

### **Steps for PCA algorithm**

#### **Getting the dataset**

Firstly, we need to take the input dataset and divide it into two subparts  $X$  and  $Y$ , where  $X$  is the training set, and  $Y$  is the validation set.

#### **Representing data into a structure**

Now we will represent our dataset into a structure. Such as we will represent the two-dimensional matrix of independent variable  $X$ . Here each row corresponds to the data items, and the column corresponds to the Features. The number of columns is the dimensions of the dataset.

#### **Standardizing the data**

In this step, we will standardize our dataset. Such as in a particular column, the features with high variance are more important compared to the features with lower variance.

If the importance of features is independent of the variance of the feature, then we will divide each data item in a column with the standard deviation of the column. Here we will name the matrix as  $Z$ .

#### **Calculating the Covariance of $Z$**

To calculate the covariance of  $Z$ , we will take the matrix  $Z$ , and will transpose it. After transpose, we will multiply it by  $Z$ . The output matrix will be the Covariance matrix of  $Z$ .

#### **Calculating the Eigen Values and Eigen Vectors**

Now we need to calculate the eigenvalues and eigenvectors for the resultant covariance matrix  $Z$ . Eigenvectors or the covariance matrix are the directions of the axes with high information. And the coefficients of these eigenvectors are defined as the eigenvalues.

#### **Sorting the Eigen Vectors**

In this step, we will take all the eigenvalues and will sort them in decreasing order, which means from largest to smallest. And simultaneously sort the eigenvectors accordingly in matrix  $P$  of eigenvalues. The resultant matrix will be named as  $P^*$ .

## Calculating the new features Or Principal Components

Here we will calculate the new features. To do this, we will multiply the  $P^*$  matrix to the  $Z$ . In the resultant matrix  $Z^*$ , each observation is the linear combination of original features. Each column of the  $Z^*$  matrix is independent of each other.

Remove less or unimportant features from the new dataset.

The new feature set has occurred, so we will decide here what to keep and what to remove. It means, we will only keep the relevant or important features in the new dataset, and unimportant features will be removed out.

## Applications of Principal Component Analysis

PCA is mainly used as the dimensionality reduction technique in various AI applications such as computer vision, image compression, etc.

It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

The purpose of this post is to provide a complete and simplified explanation of principal component analysis (PCA). We'll cover how it works step by step, so everyone can understand it and make use of it, even those without a strong mathematical background.

PCA is a widely covered machine learning method on the web, and there are some great articles about it, but many spend too much time in the weeds on the topic, when most of us just want to know how it works in a simplified way.

Principal component analysis can be broken down into five steps. I'll go through each step, providing logical explanations of what PCA is doing and simplifying mathematical concepts such as standardization, covariance, eigenvectors and eigenvalues without focusing on how to compute them.

## HOW DO YOU DO A PRINCIPAL COMPONENT ANALYSIS?

1. Standardize the range of continuous initial variables
2. Compute the covariance matrix to identify correlations
3. Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components
4. Create a feature vector to decide which principal components to keep

## 5. Recast the data along the principal components axes

### What Is Principal Component Analysis?

Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data points much easier and faster for machine learning algorithms without extraneous variables to process.

So, to sum up, the idea of PCA is simple — **reduce the number of variables of a data set, while preserving as much information as possible.**

### Step-by-Step Explanation of PCA

#### STEP 1: STANDARDIZATION

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (for example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{\text{value} - \text{mean}}{\text{standard deviation}}$$

Once the standardization is done, all the variables will be transformed to the same scale.

## STEP 2: COVARIANCE MATRIX COMPUTATION

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

The covariance matrix is a  $p \times p$  symmetric matrix (where  $p$  is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables  $x$ ,  $y$ , and  $z$ , the covariance matrix is a  $3 \times 3$  data matrix of this form:

$$\begin{bmatrix} Cov(x, x) & Cov(x, y) & Cov(x, z) \\ Cov(y, x) & Cov(y, y) & Cov(y, z) \\ Cov(z, x) & Cov(z, y) & Cov(z, z) \end{bmatrix}$$

### Covariance Matrix for 3-Dimensional Data.

Since the covariance of a variable with itself is its variance ( $Cov(a,a)=Var(a)$ ), in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. And since the covariance is commutative ( $Cov(a,b)=Cov(b,a)$ ), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

**What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?**

It's actually the sign of the covariance that matters:

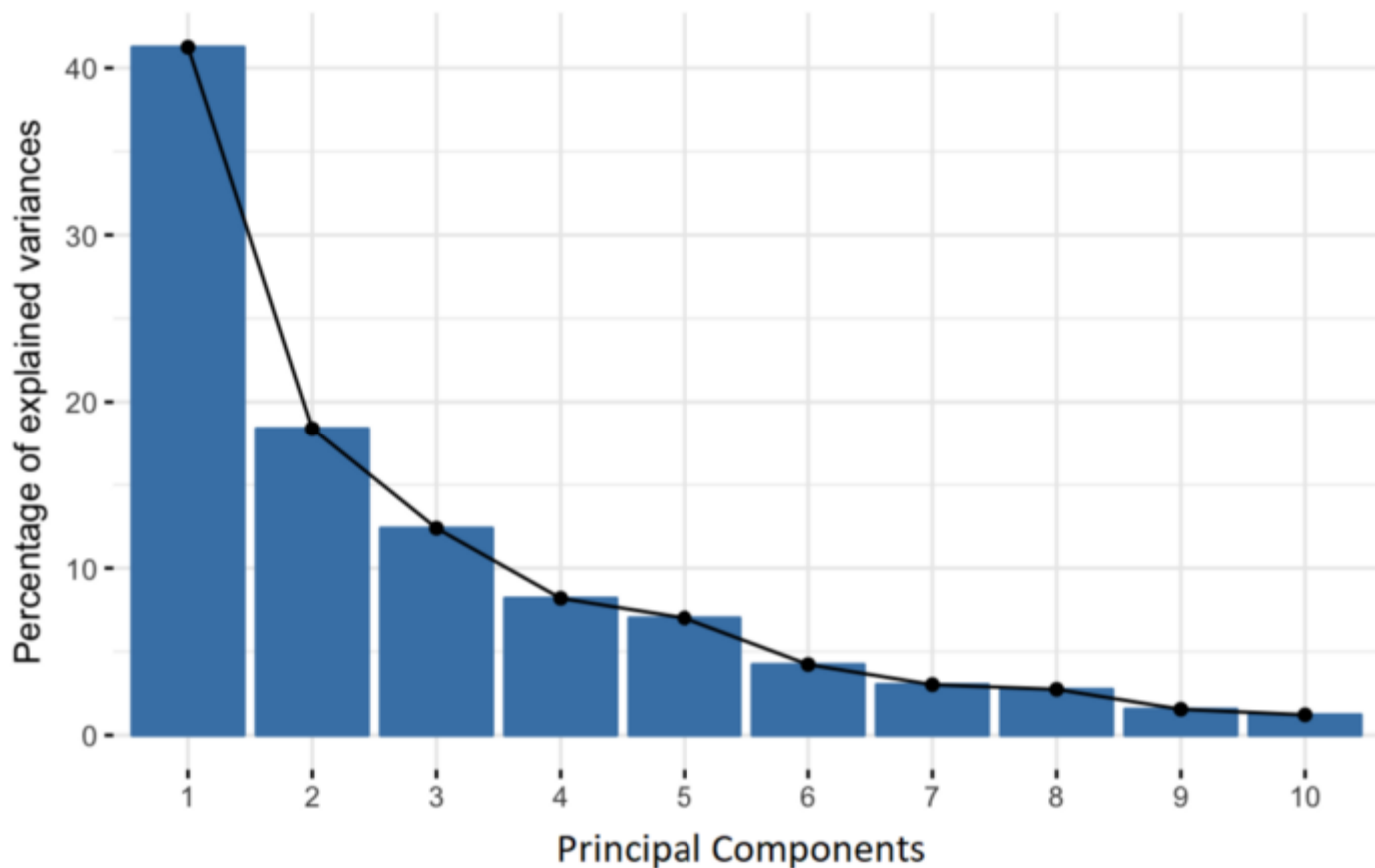
- If positive then: the two variables increase or decrease together (correlated)
- If negative then: one increases when the other decreases (Inversely correlated)

Now that we know that the covariance matrix is not more than a table that summarizes the correlations between all the possible pairs of variables, let's move to the next step.

### **STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS**

Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the *principal components* of the data. Before getting to the explanation of these concepts, let's first understand what do we mean by principal components.

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the scree plot below.



Percentage of Variance (Information) for each by PC.

Organizing information in principal components this way, will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

An important thing to realize here is that the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.

Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has. To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.



#### **STEP 4: FEATURE VECTOR**

As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. In this step, what we do is, to choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

So, the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only  $p$  eigenvectors (components) out of  $n$ , the final data set will have only  $p$  dimensions.

#### **STEP 5: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES**

In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Reference:

1. <https://www.javatpoint.com/principal-component-analysis>

