UNIT-4

Syllabus:- Principal components analysis (PCA), Locally Linear Embedding (LLE), Factor Analysis

Principal Component Analysis

*Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.

*It is one of the popular tools that is used for exploratory data analysis and predictive modelling.

*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.

*Some real-world applications of PCA are ***image processing, movie recommendation system, optimizing the power allocation in various communication channels.***

Principal Component Analysis is a technique used to:

• *Reduce the dimensionality of the data set*

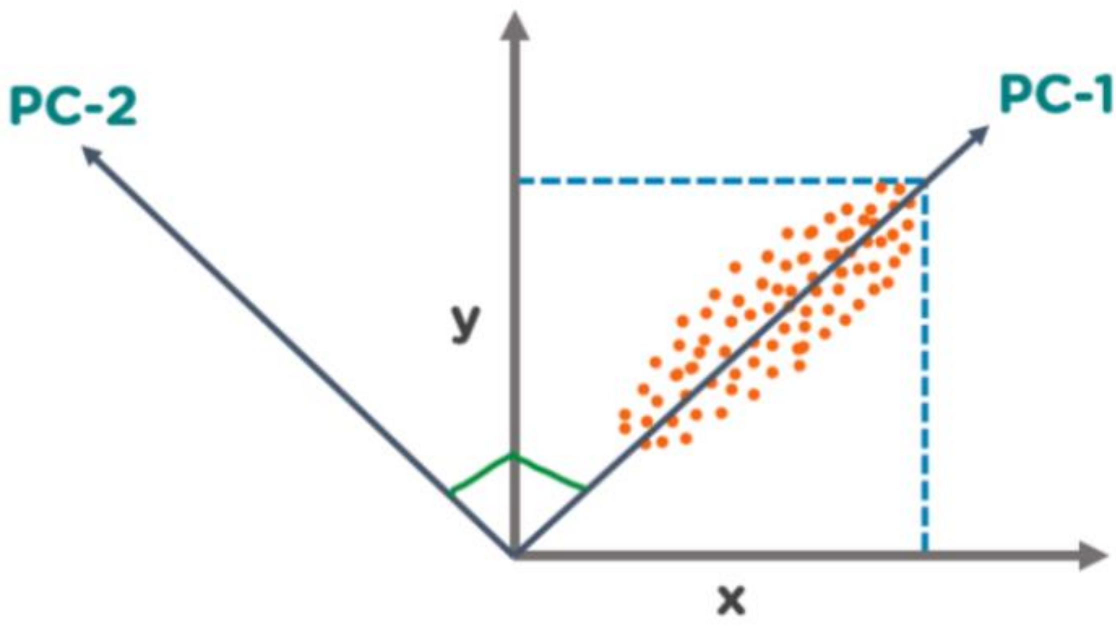
• *Identify new meaningful underlying variables*

• *Loose minimum information*

by finding the directions in which a cloud of data

points is stretched most.

*



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

Steps for PCA algorithm

1.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.

2. **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.

3. **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.

4. **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

5. **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.

6. **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*.

7. **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.

8. **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

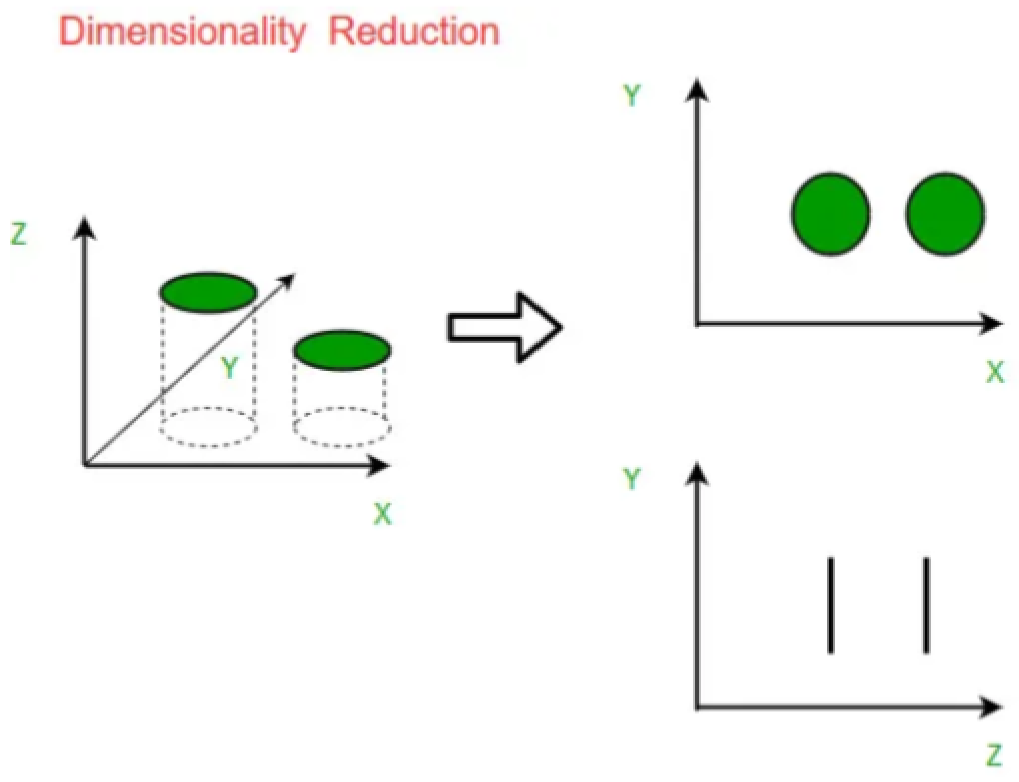
- PCA is mainly used as the dimensionality reduction technique in various AI applications such **as computer vision, image compression, etc.**
- It can also be used for finding hidden patterns if data has high dimensions. Some fields where PCA is used are Finance, data mining, Psychology, etc.

Locally Linear Embedding (LLE)

Locally Linear Embedding (LLE) is a method of Non Linear Dimensionality reduction proposed by Sam T. Roweis and Lawrence K. Saul in 2000 in their paper titled “Nonlinear Dimensionality Reduction by Locally Linear Embedding”. This article is based on multiple sources mentioned in the references section. The project by Jennifer Chu helped me understand LLE better.

* Machine Learning algorithms use the features they are trained on to predict the output. For example, in the case of a house price prediction problem, there might be a number of features like the size of the house, number of bedrooms, number of bathrooms, etc.

* One major problem many machine learning algorithms face while doing this is that of overfitting, where the model fits the training data so well that it is unable to predict the real life test data accurately. This is a problem since it makes the algorithm very effective.



Locally Linear Embedding (LLE)

Data sets can often be represented in a n-Dimensional feature space, with each dimension used for a specific feature.

The LLE algorithm is an unsupervised method for dimensionality reduction. It tries to reduce these n-Dimensions while trying to preserve the geometric features of the original non-linear feature structure. For example, in the below illustration, we cast the structure of the swiss roll into a lower dimensional plane, while maintaining its geometric structure.

In short, if we have D dimensions for data X1, we try to reduce X1 to X2 in a feature space with d dimensions.

$$X1 \epsilon D \rightarrow X2 \epsilon d$$

How does Locally Linear Embedding works.

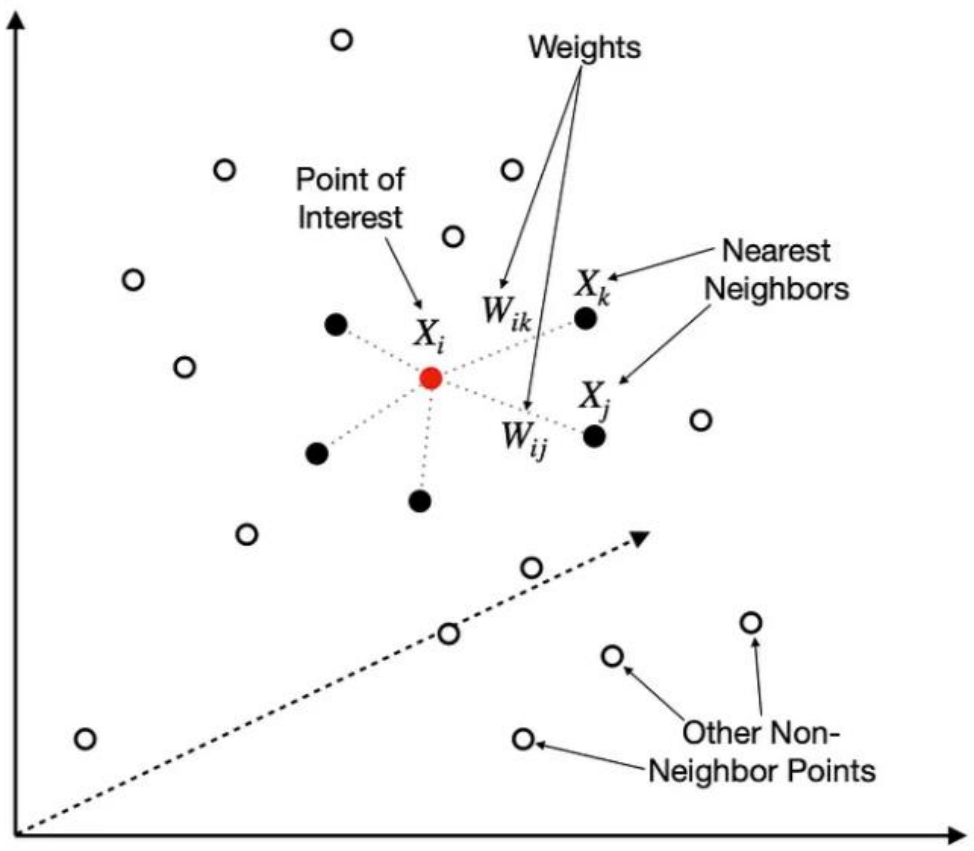
The high-level steps

Similar to Isomap, LLE combines several steps to produce the lower-dimensional embedding. These are:

1. Use a KNN approach to **find the k nearest neighbors** of every data point. Here, “k” is an arbitrary number of neighbors that you can specify within model hyperparameters.

2. **Construct a weight matrix** where every point has its weights determined by minimizing the error of the cost function shown below. Note that every point is a linear combination of its neighbors, which means that **weights for non-neighbors are 0**.

Original High-Dimensional Space



We know the position of the Point of Interest

We know the positions of all the Nearest Neighbors

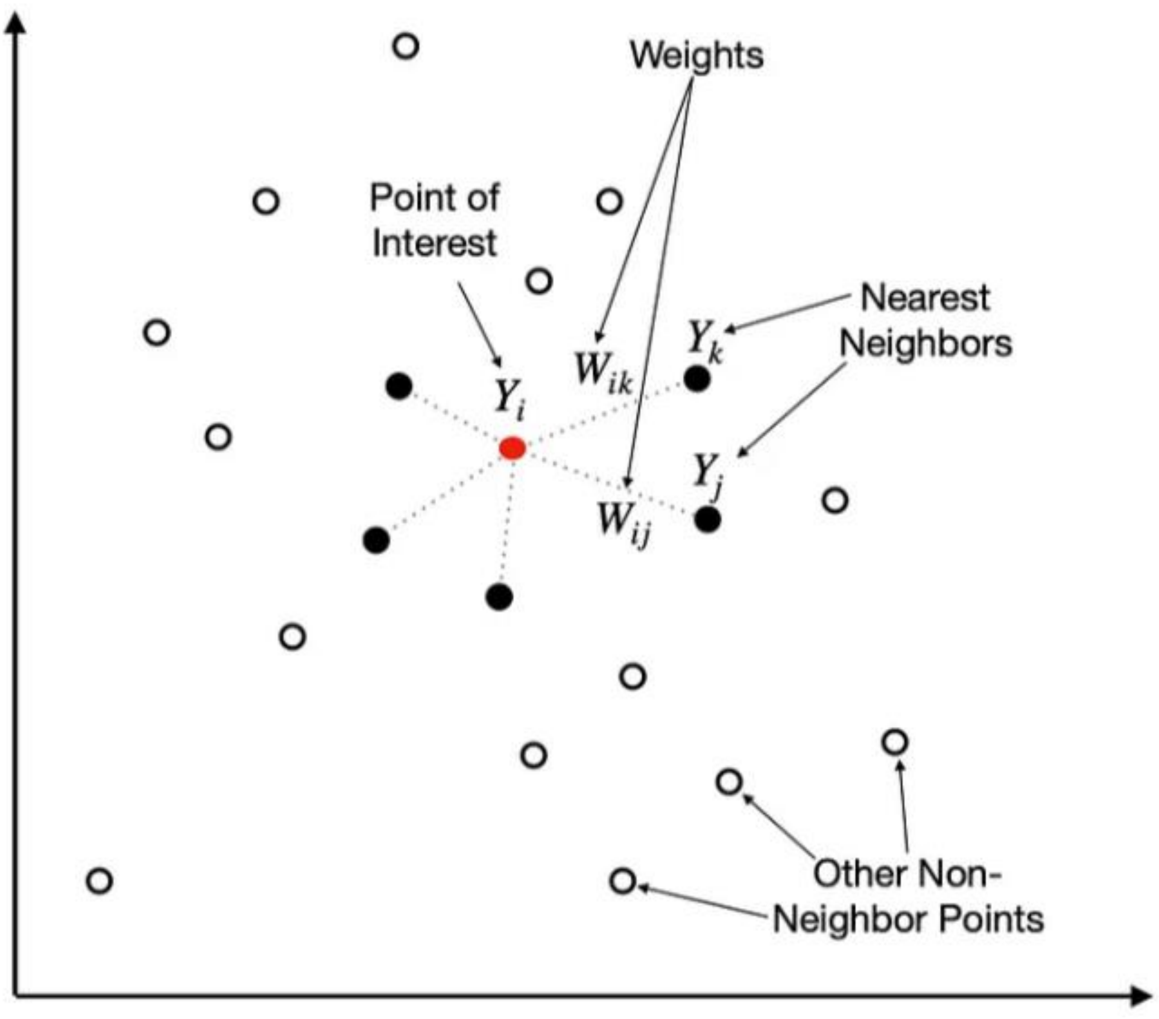
$$E(W) = \sum_i |X_i - \sum_j W_{ij} X_j|^2$$

The cost function is solved to find the weights, where the sum of weights for each X_i is set to equal to 1

$$\sum_j W_{ij} = 1$$

3. **Find the positions** of all the points in the **new lower-dimensional embedding** by minimizing the cost function shown below. Note, here we use weights (W) from step two and solve for Y. The actual solving is performed using Partial Eigenvalue Decomposition.

New Lower-Dimensional Space



We know the weights from the previous step

$$C(Y) = \sum_i |Y_i - \sum_j W_{ij}Y_j|^2$$

The cost function is solved to find the positions of Y_i and its neighbors in the new lower-dimensional space using weights from the previous step.

With the above steps completed, we get a lower-dimensional representation of the data, which we can typically visualize using standard scatterplots, provided we reduce the dimensionality to 3D or less.

LLE variants

You should be aware of a few LLE variants, which improve upon the original setup. However, note that these improvements come at the cost of efficiency, making the algorithm slower. Here is how scikit-learn describes these variants:

- **Modified LLE (MLLE)** — One well-known issue with LLE is the regularization problem. A way to address it is to use **multiple weight vectors** in each neighborhood. This is the essence of MLLE.
- **Hessian LLE (HLLE)**— Hessian Eigenmapping is another method of solving the regularization problem of LLE. It revolves around a **hessian-based quadratic form** at each neighborhood used to recover the locally linear structure.

While I will not go into details, I recommend you experiment with them to see which variant yields the best results for your data. Personally, I find MLLE to perform well in most scenarios (see an example of this in the next section).

Difference between LLE and Isomap

The two algorithms are similar in the way they approach dimensionality reduction, but they do have their differences.

Similar to LLE, Isomap also uses KNN to find the nearest neighbors in the first step. However, the second step constructs neighborhood graphs instead of describing each point as a linear combination of its neighbors. Then it uses these graphs to compute the shortest path between every pair of points.

Finally, Isomap uses those pairwise distances between all points to construct a lower-dimensional embedding.

Factor Analysis

Factor Analysis in Machine Learning

1.Reduce a large number of variables into fewer numbers of factors.

2.Puts maximum common variance into a common score.

3.Associate multiple observed variables with a latent variable.

4.Has the same number of factor's and variables, where each factor certain amount of overall variance.

Eigenvalue:- A measure of the variance that a factor explains for observed variables. A factor with eigenvalue less than one variance than a single observed value.

Factor Analysis Process :-

1. <u>Principal Analysis Component (PCA)</u>

Extract the hidden factor from the dataset.

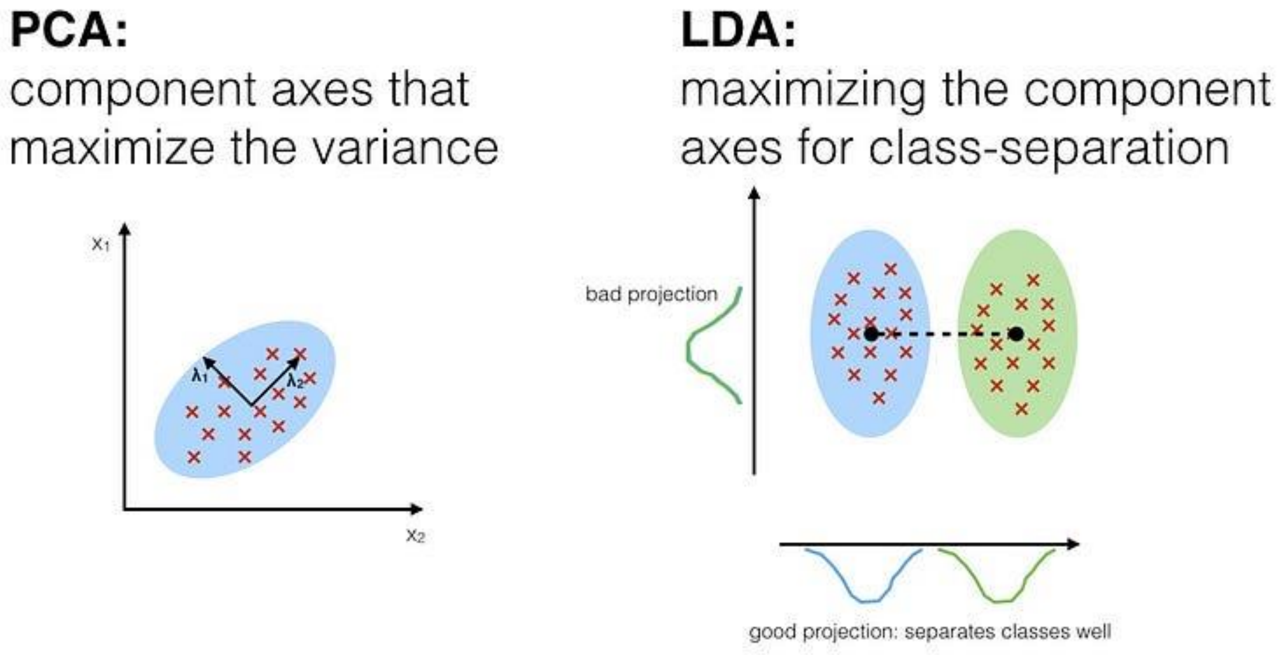
Defines data using less numbers of components,explaining the variance in your data

Reduce the computation complexity .

Determine that the new data is the part of the group of data points from the training set.

2. Linear Dimensions Analysis(LDA) :-

    Reduces dimensions.
    Search the linear combination of variables that best separates two class.
    Reduce degree of overfitting.
    Determine how to classify the new observation out of oup of classes.


PCA:
component axes that maximize the variance
x_1
λ_1
λ_2
x_2
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

Direction of Maximum Variance: -

1. PCA seeks the linear combination of variables in order to extract the maximum variance.

2.2. Compute Eigenvector that are principal components of the dataset and collect them in projection matrix.

3.3. Each of the Eigenvector is associate with Eigenvalue, which is magnitudes.

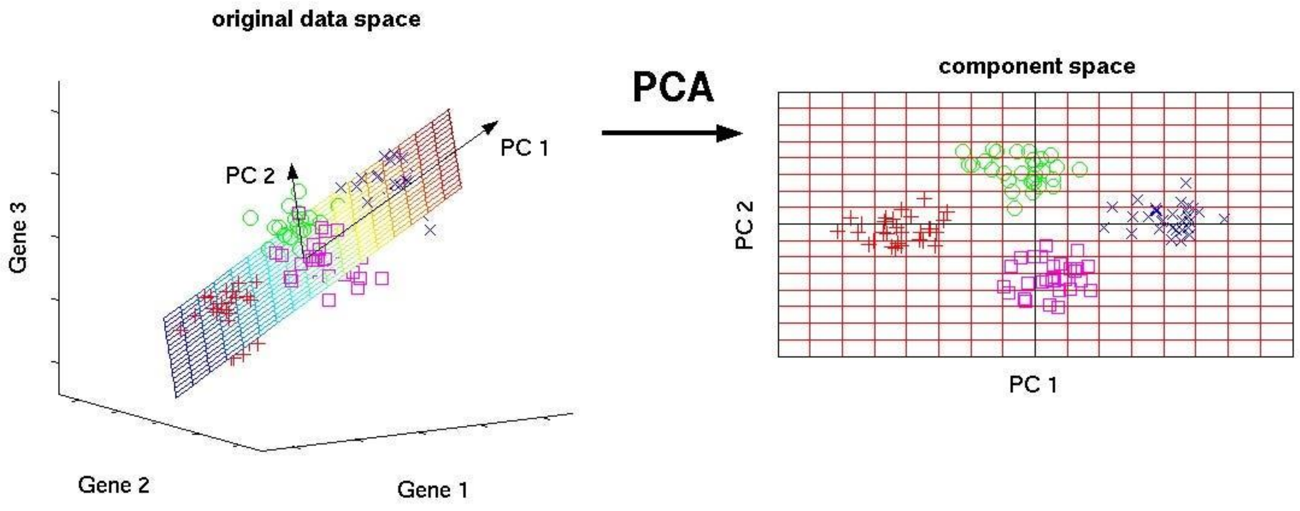
4.4. Reduce the dataset into smaller dimensional subspace by dropping the less informative Eigenpairs.

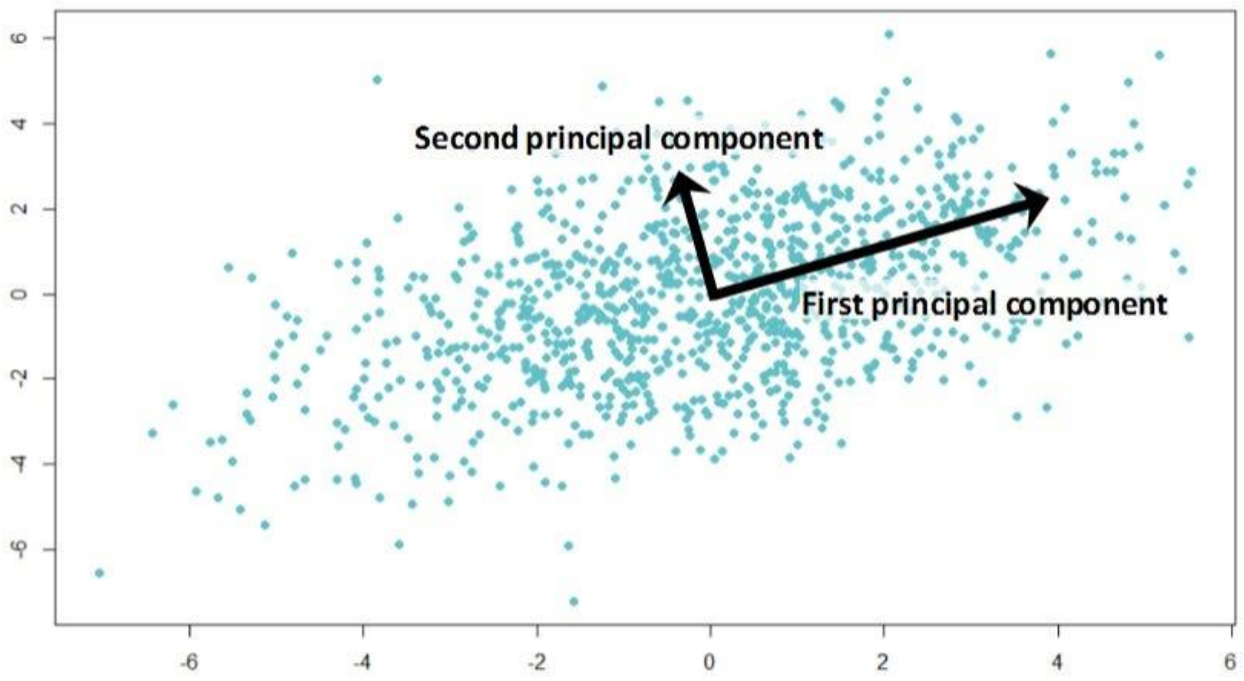
Plafond’s line depending on two criteria:-

1. The variation of values should be maximal along this line.

2. The error should be minimum if you don't reconstruct original two positions of a blue dot from the new position of the red dot.

First Principle Component: -





The first principle component (PCI) is the direction of the maximum variance and is obtained by solving Eigenvector.

<u>Finding PCl :-</u>

PCl (Mathematically) : alxl + a2x2 + a3x3 +.....................................+anxn

Constraint: a1A2 + a2A2 + a3A2 + ...+akA2

Eigen decomposition to solve the equation.

NOTE : - Eigen decomposition is the factorisation of the matrix into a canonical form, where the matrix is represented in terms of Eigenvectors or Eigenvalues.

<u>Eigenvalues and PCA. :-</u>

Eigenvalues are the variances of the principal component arranged in descending order.

Summary of PCA Process :

1. Standardize the data PCA : Requires that the input variables have similar scales of the measurement.
2. Build the correlation matrix : This summarizes how your variables all relate to one another.
3. Obtain the Eigenvalue and Eigenvector from correlation matrix : Break the matrix down in direction and magnitude . Sort Eigenvalues in descending order and choose Eigenvectors that corresponds to the largest Eigenvalue.
4. Construct the projection matrix from selected Eigenvector : Reduce the dataset by dropping less informative Eigenpairs.
5. Transform the original dataset to obtain a kk-dimensional feature sub space : Compress your data into smaller space by excluding less important directions.