## PRINCIPAL COMPONENTS ANALYSIS (PCA)

In real world data analysis tasks, we analyze complex data i.e. multi-dimensional data. We plot the data and find various patterns in it or use it to train some machine learning models. One way to think about dimensions is that suppose you have a data point **x,** if we consider this data point as a physical object then dimensions are merely a basis of view, like where is the data located when it is observed from horizontal axis or vertical axis.

As the dimensions of data increases, the difficulty to visualize it and perform computations on it also increases. So, how to reduce the dimensions of a data-

* Remove the redundant dimensions
* Only keep the most important dimensions

PCA is considered a technique for reducing the number of variables without loss of information and for identifying new variables with greater meaning. PCA reduces data by geometrically projecting them onto lower dimensions, called principal components. The first principal component is chosen to minimize the total distance between the data and their projection onto the principal component. The second (and subsequent) principal components are selected similarly, with the additional requirement that they are uncorrelated with all previous principal components. This requirement of no correlation means that the maximum number of principal components possible is either the number of samples or the number of features, whichever is smaller. The principal components selection process has the effect of maximizing the correlation between data and their projection. There are different ways of solving PCA. The most efficient algorithm uses Singular Value Decomposition (SVD).

*Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components.*

First try to understand some terms –



**Variance:**It is a measure of the variability or it simply measures how spread the data set is. Mathematically, it is the average squared deviation from the mean score. We use the following formula to compute variance var(x).

**Covariance:**Itis a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction. Formula is shown above denoted by cov(x,y) as the covariance of x and y.  
Here, xi is the value of x in ith dimension.x bar and y bar denote the corresponding mean values.

Since we try to find the patterns among the data sets so we want the data to be spread out across each dimension. Also, we want the dimensions to be independent. Such that if data has high covariance when represented in some n number of dimensions then we replace those dimensions with linear combination of those n dimensions. Now that data will only be dependent on linear combination of those related n dimensions. (related = have high covariance).

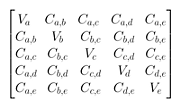
So, PCA finds a new set of dimensions (or a set of basis of views) such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them. It means more important principle  
axis occurs first. (more important = more variance/more spread out data)

How does PCA work -

1. Calculate the covariance matrix *X* of data points.
2. Calculate Eigen vectors and corresponding Eigen values.
3. Sort the Eigen vectors according to their Eigen values in decreasing order.
4. Choose first k Eigen vectors and that will be the new k dimensions.
5. Transform the original n dimensional data points into k dimensions.

D:\ML & AI Learning\1.png

Assuming we have the knowledge of variance and covariance, Let’s look into what a **Covariance matrix** is.



A covariance matrix of some data set in 4 dimensions a,b,c,d.  
Va : variance along dimension a  
Ca,b : Covariance along dimension a and b

If we have a matrix X of m\*n dimension such that it holds n data points of m dimensions, then covariance matrix can be calculated as

D:\ML & AI Learning\3.png

It is important to note that the covariance matrix contains –

* variance of dimensions as the main diagonal elements.
* covariance of dimensions as the off diagonal elements

Also, covariance matrix is symmetric. (observe from the image above)  
As, we discussed earlier we want the data to be spread out i.e. it should have high variance along dimensions. Also we want to remove correlated dimensions i.e. covariance among the dimensions should be zero (they should be linearly independent). Therefore, our covariance matrix should have -

* large numbers as the main diagonal elements.
* zero values as the off diagonal elements.

We call it a diagonal matrix.

So, we have to transform the original data points such that their covariance is a diagonal matrix. The process of transforming a matrix to diagonal matrix is called diagonalization.

**Always normalize your data before doing PCA because if we use data (features here) of different scales, we get misleading components. We can also simply use correlation matrix instead of using covariance matrix if features are of different scales.**

This defines the goal of PCA -

1. Find linearly independent dimensions (or basis of views) which can losslessly represent the data points.
2. Those newly found dimensions should allow us to predict/reconstruct the original dimensions. The reconstruction/projection error should be minimized.

Let’s try to understand what I mean by projection error. Suppose we have to transform a 2 dimensional representation of data points to a one dimensional representation. So we will basically try to find a straight line and project data points on them. (A straight line is one dimensional). There are many possibilities to select the straight line. Let’s see two such possibilities -

|  |  |
| --- | --- |
| **D:\ML & AI Learning\1_4HxMtrguXVYqCnQP711nxQ.png** | **D:\ML & AI Learning\5.png** |

Say magenta line will be our new dimension. If you see the red lines (connecting the projection of blue points on magenta line) i.e. the perpendicular distance of each data point from the straight line is the projection error. Sum of the error of all data points will be the total projection error.  
Our new data points will be the projections (red points) of those original blue data points. As we can see we have transformed 2 dimensional data points to one dimensional data points by projection them on 1 dimensional space i.e. a straight line. That magenta straight line is called *principal axis.*Since we are projecting to a single dimension, we have only one principal axis.

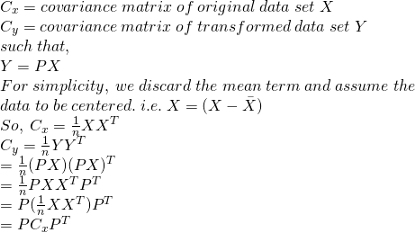
Clearly, second choice of straight line is better because -

1. The projection error is less than that in the first case.
2. Newly projected red points are more widely spread out than the first case. i.e. more variance.

Steps we have performed so far –

* We have calculated the covariance matrix of original data set matrix **X.**

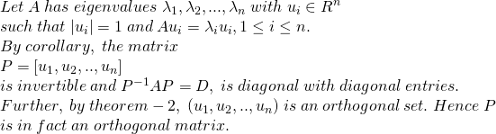
Nowwe want to transform the original data points such that the covariance matrix of transformed data points is a diagonal matrix. How to do that?

****

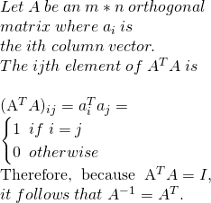
Here’s the trick- If we find the matrix of eigen vectors of Cx and use that as P (P is used for transforming X to Y, see the image above) , then Cy (covariance of transformed points) will be a diagonal matrix. Hence Y will be the set of new/transformed data points.  
Now, if we want to transform points to k dimensions then we will select first k eigen vectors of the matrix Cx (sorted decreasingly according to eigen values) and form a matrix with them and use them as P.

So, if we have m dimensional original n data points then  
X : m\*n  
P : k\*m  
Y = PX : (k\*m)(m\*n) = (k\*n)  
Hence, our new transformed matrix has n data points having k dimensions.

Theorem 1:

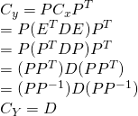
++

Theorem 2:



Having these theorems, we can say that

A symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors. Orthonormal vectors are just normalized orthogonal vectors.



It is evident that the choice of P diagonalizes Cy. This was the goal for PCA. We can summarize the results of PCA in the matrices P and Cy.

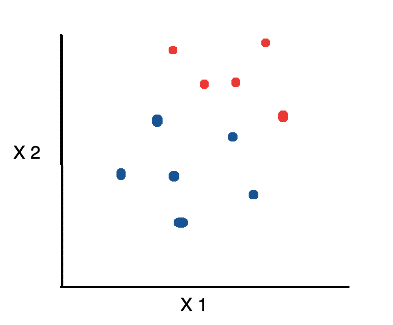
* The principal components of X are the eigenvectors of Cx.
* The i th diagonal value of Cy is the variance of X along pi

D:\ML & AI Learning\11.png

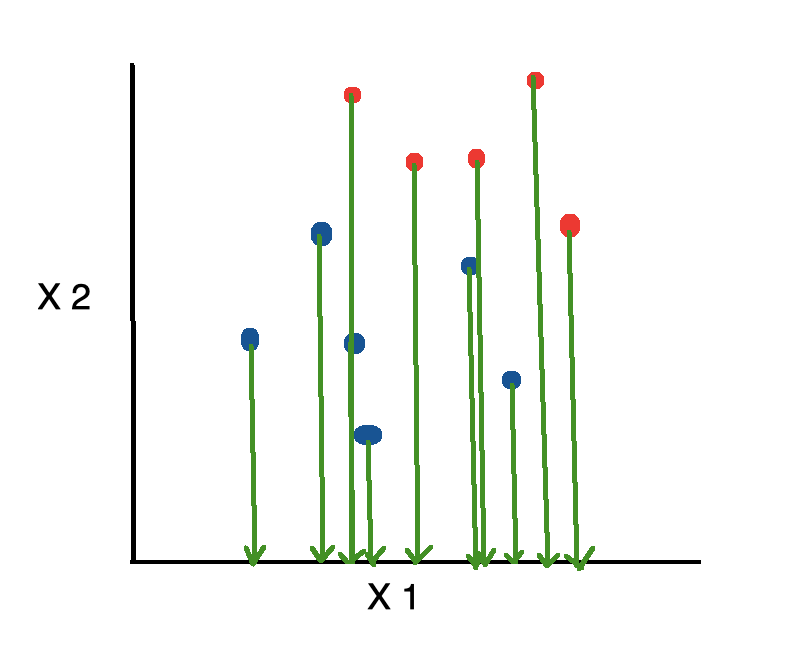
***Note: PCA is an analysis approach. You can do PCA using SVD, or you can do PCA doing the eigen-decomposition (like we did here), or you can do PCA using many other methods. SVD is just another numerical method. So, don’t confuse the terms PCA and SVD. However, there are some performance factors of sometimes choosing SVD over eigen-decomposition or the other way around(not that you should care much about).***

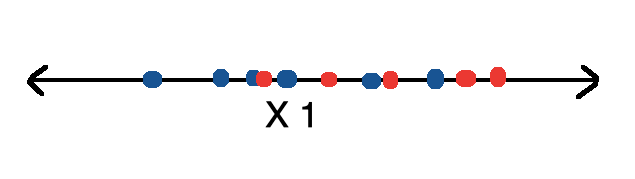
## LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique. As the name implies dimensionality reduction techniques reduce the number of dimensions (i.e. variables) in a dataset while retaining as much information as possible. For instance, suppose that we plotted the relationship between two variables where each color represents a different class.



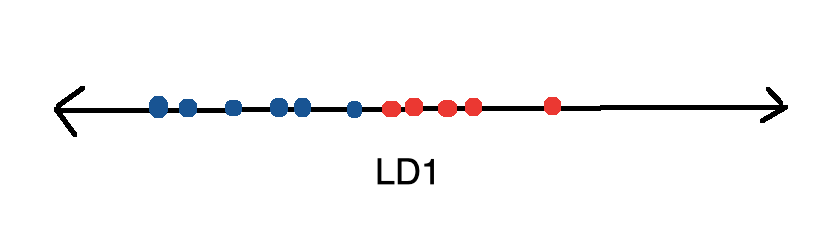
If we’d like to reduce the number of dimensions down to 1, one approach would be to project everything on to the x-axis.





This is bad because it disregards any useful information provided by the second feature. On the other hand, Linear Discriminant Analysis, or LDA, uses the information from both features to create a new axis and projects the data on to the new axis in such a way as to minimizes the variance and maximizes the distance between the means of the two classes.

|  |  |
| --- | --- |
| https://miro.medium.com/max/787/1*Fz3JQ80No5Nnbap28EGRTg.png | https://miro.medium.com/max/786/1*5lhckC2RQzq28zNL7WtU5A.png |

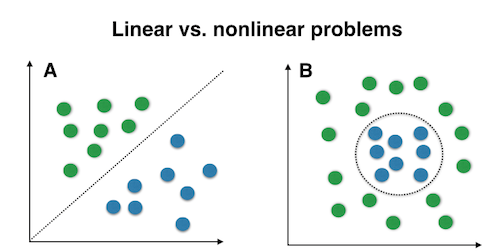


## ****KERNEL PCA****

PCA is a linear method. That is it can only be applied to datasets which are linearly separable. It does an excellent job for datasets, which are linearly separable. But, if we use it to non-linear datasets, we might get a result which may not be the optimal dimensionality reduction. Kernel PCA uses a kernel function to project dataset into a higher dimensional feature space, where it is linearly separable. It is similar to the idea of Support Vector Machines.

### Nonlinear Dimensionality Reduction

The “classic” PCA approach described above is a linear projection technique that works well if the data is linearly separable. However, in the case of linearly inseparable data, a nonlinear technique is required if the task is to reduce the dimensionality of a dataset.



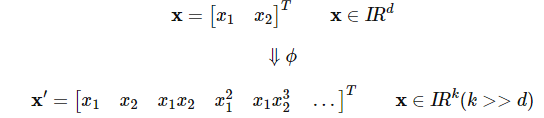
### Kernel Functions and The Kernel Trick

The basic idea to deal with linearly inseparable data is to project it onto a higher dimensional space where it becomes linearly separable. Let us call this nonlinear mapping function ϕ so that the mapping of a sample x can be written as x→ϕ(x), which is called “kernel function.”

Now, the term “kernel” describes a function that calculates the dot product of the images of the samples x under ϕ.



In other words, the function ϕϕ maps the original d-dimensional features into a larger, k-dimensional feature space by creating nononlinear combinations of the original features. For example, if xx consists of 2 features:



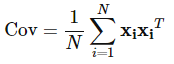
Often, the mathematical definition of the RBF kernel is written and implemented as



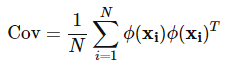
where  γ is a free parameter that is to be optimized.

#### Gaussian radial basis function (RBF) Kernel PCA

In the linear PCA approach, we are interested in the principal components that maximize the variance in the dataset. This is done by extracting the eigenvectors (principle components) that correspond to the largest eigenvalues based on the covariance matrix:



Generalized this approach for data that was mapped onto the higher dimensional space via a kernel function:



However, in practice the the covariance matrix in the higher dimensional space is not calculated explicitly (kernel trick). Therefore, the implementation of RBF kernel PCA does not yield the principal component axes (in contrast to the standard PCA), but the obtained eigenvectors can be understood as projections of the data onto the principal components.

In order to implement the RBF kernel PCA we just need to consider the following two steps.

1. Computation of the kernel (similarity) matrix.

In this first step, we need to calculate



for every pair of points. E.g., if we have a dataset of 100 samples, this step would result in a symmetric 100x100 kernel matrix.

1. Eigen decomposition of the kernel matrix.

Since it is not guaranteed that the kernel matrix is centered, we can apply the following equation to do so:



where 1N is (like the kernel matrix) a N×N matrix with all values equal to 1/N  
Now, we have to obtain the eigenvectors of the centered kernel matrix that correspond to the largest eigenvalues. Those eigenvectors are the data points already projected onto the respective principal components.

