The entire subject of statistics is based around the idea that you have this big set of data,

and you want to analyse that set in terms of the relationships between the individual

points in that data set.

The Standard Deviation (SD) of a data set is a measure of how spread out the data is.

How do we calculate it? The English definition of the SD is: “The average distance

from the mean of the data set to a point”. The way to calculate it is to compute the

squares of the distance from each data point to the mean of the set, add them all up,

divide by n/n-1, and take the positive square root. n/n-1 depends whether you are taking the entire population or a sample from the population.

Variance is another measure of the spread of data in a data set. In fact it is almost identical to the standard deviation.

Both of these measures are 1-dimensional.We can calculate the SD/var for each dimension of the data independent of the other dimensions.

In most cases it would be useful to figure out how much the different dimensions differ with each other. This takes us to the measure of covariances- which gives us a measure of two dimensions vary w.r.t to each other.

So if you calculate the co variance between one dimension and itself you get back the variance!.(sum\_over\_all\_data\_points(x-x\_bar)\*(y-y\_bar))(n-1)

If the value is positive: then that indicates that both dimensions increase together, zero implies independence and negative implies increase in one decreases the other.

Oh hang one this one sounds similar to a concept we learnt in school-proportionality(direct/indirect)-rings a bell?

If we have a dataset with more than two dimensions, then it would make sense to put the covariance values in a matrix. Notice that the diagonal values will 1, as this this will represent the variance with itself.

For a n dimensional data you calculate n!/(n-2)!\*2

Matrix

2 3       3            12         3

      x            =        =4 x

2 1       2            8          2

If we consider the matrix (3,2)T , as a vector in two dimensional space going from (0,0) to (3,2). if we are multiplying this with a 2x2 matrix on the left, that acts as transformation. Transforming our line from the original space to a new

space, hence the result is a multiple of the original vector.

This is where eigenvectors come from. Here (3,2) is a eigen vector of our square matrix. Note we will always find unit length eigenvetors, because its direction that we are interested in.

Eigen vectors can be found of square matrices (not all square matrix has eigen vector tho!). For a nxn matrix there would be n eigen vectors.

Another property of eigenvectors is that even if I scale the vector by some amount before I multiply it, I still get the same multiple of it as a result. This

is because if you scale a vector by some amount, all you are doing is making it longer,not changing it’s direction. All eigenvectors of a matrix are orthogonal. Eigenvectors comes in pairs with its corresponding Eigenvalues.in the above example 4 is your eigen value.

so given a square matrix A and v is your eigen vector, then we can write

Av=lambda x v where lambda is your eigen value.

There will be libraries now a days that will do the calulation for you and find the eigenvalues/vectors. If you wanted to do it for the above square matrix, you can do ((2,3;2 1).T-Ixlambda)(v)=0

where v be your eigen vectors and lambda is your eigen value, I is an identity matirx of the same dimensions as our square matrix. Provided v is non zero we can solve for ((2,3;2 1).T-Ixlambda)=0 and find out the eigen values.

Then plugging those in the equation with v and equating it 0 we can find the eigen vectors.

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly,

while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables,

which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order.

So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

Intuitively, Principal Component Analysis can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint.

Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analysing data.

The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, ie. by reducing the number of dimensions, without much loss of information. This technique used in image compression.

First step is to normalize the data that we have so that PCA works properly. This is done by subtracting the respective means from the numbers in the respective column. Then find the covariance matrix and figure out the eigenvectors

and values of this covriance matrix. The eigenvector with the highest eigenvalue is the principle component of the data set. How many principal components we use in our data depends on us, you do loose some information if you discard the

lower valued eigenvalues, but its nothing significant. The main idea is that, the remaining ones should capture/explain the maximum variability in the data. So your input feature vector dimensions is hugely reduced.

As our newDataset= (n significant eigenvectors, where n<D).T X scaledData

of course to get the scaledData back one has to to matrix Inverse operation on the principal component matrix and multiply the newDataset to that.

Maximum Margin Classifier is the simple way to divide your data if your data if your data is linearly (or via a plane in multi-dimensional space) separable. But due to obvious reasons it can’t be applied to the data where no clear linear boundary is possible.

Support Vector Machine or SVM is a further extension to SVC to accommodate non-linear boundaries. Though there is a clear distinction between various definitions but people prefer to call all of them as SVM to avoid any complications.

The goal of SVM is to find the decision boundary linearly separating our classes in out dataset. The boundary will have the equation wTx+b=0. Any thing above this decision boundary will have label 1 or zero otherwise.

Thus, we can figure out if an instance has been classified properly by checking that y(wT x+b)>1 (which will be the case as long as either both y;wT x+b > 0 or else y;wT x + b < 0).

If the data can be perfectly separated by a hyperplane it is likely true that there are multiple potential separating hyperplanes.

We need a method for identifying the optimal separating hyperplane. This is known as the maximal margin hyperplane, which is the separating hyperplane that is farthest from the training observations.

The margin is the smallest possible (perpendicular) distance between a training observation and the separating hyperplane.The maximal margin hyperplane defines the hyperplane that minimizes the marginal distance across all

training observations. The expectation is that a classifier with a large margin for the training observations will also have a large margin for the test observations, leading to accurate classifications.

Kernels are a computationally efficient method for extending the feature space to accomodate a non-linear decision boundary.The idea is that our data, which isn’t linearly separable in our ’n’ dimensional space may be linearly separable in a higher dimensional space.

Support Vector Classifier: SVC is an extension to maximum margin classifier where we allow some misclassification to happen.Support vector classifiers relax the requirement of the maximal margin classifier by allowing the

separating hyperplane to not perfectly separate the observations; instead, it can make some errors. This is reasonable when:

                There exists no perfectly separating hyperplane

                A perfectly separating hyperplane is too sensitive to individual training observations, generating potentially very small margins or overfitting the training set