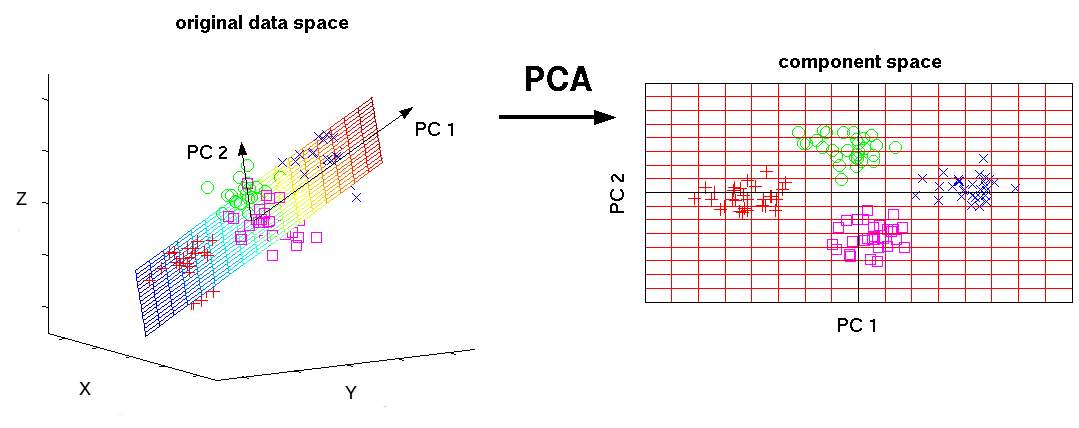
**Principal Component Analysis(PCA):**

* PCA is a method of extracting important variables (in form of components) from a large set of variables available in a data set.
* Extracting low dim set of features from high dim, with high information as possible
* Visualization becomes easy with fewer variables

e.g, Lets take an example of data with 500 \* 100. Scatter plots plotted are 100\*(99)/2.

Selecting subset of predictors with maximum information.



A principal component is a normalized linear combination of the original predictors in a data set. In image above, PC1 and PC2 are the principal components. Let’s say we have a set of predictors as X1,X2…..Xn

The principal component can be written as:

Z1 = k11X1 + k21X2 + k31X3 +…………………kn1Xn

Z1 is the first principal component

K is the loading vector

The loadings are constrained to a sum of square equals to 1. This is because large magnitude of loadings may lead to large variance. It also defines the direction of the principal component (Z¹) along which data varies the most. It results in a line in n dimensional space which is closest to the no. observations. Closeness is measured using average squared euclidean distance.

X1,…Xp are normalized predictors. Normalized predictors have mean equals to zero and standard deviation equals to one.

**First principal component** is a linear combination of original predictor variables which captures the maximum variance in the data set. It determines the direction of highest variability in the data. Larger the variability captured in first component, larger the information captured by component. No other component can have variability higher than first principal component.

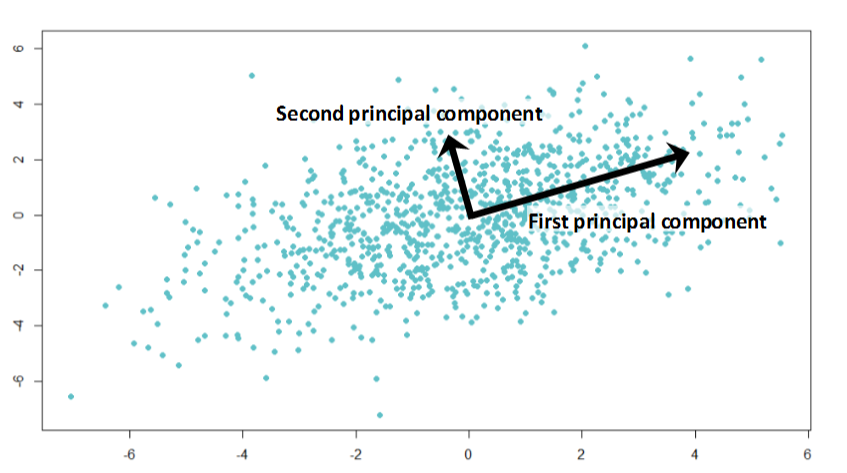
The first principal component results in a line which is closest to the data i.e. it minimizes the sum of squared distance between a data point and the line.

Similarly, we can compute the second principal component also.

**Second principal component** (Z2) is also a linear combination of original predictors which captures the remaining variance in the data set and is uncorrelated with Z1. In other words, the correlation between first and second component should is zero. It can be represented as:

Z2 = k12X1 + k22X2 +……+ kn2Xn

If the two components are uncorrelated, their directions should be orthogonal (image below). This image is based on a simulated data with 2 predictors. Notice the direction of the components, as expected they are orthogonal. This suggests the correlation b/w these components in zero.

All succeeding principal component follows a similar concept i.e. they capture the remaining variation without being correlated with the previous component. In general, for *n × p* dimensional data, min(*n-1, p)* principal component can be constructed.

The directions of these components are identified in an unsupervised way i.e. the response variable(Y) is not used to determine the component direction. Therefore, it is an unsupervised approach.

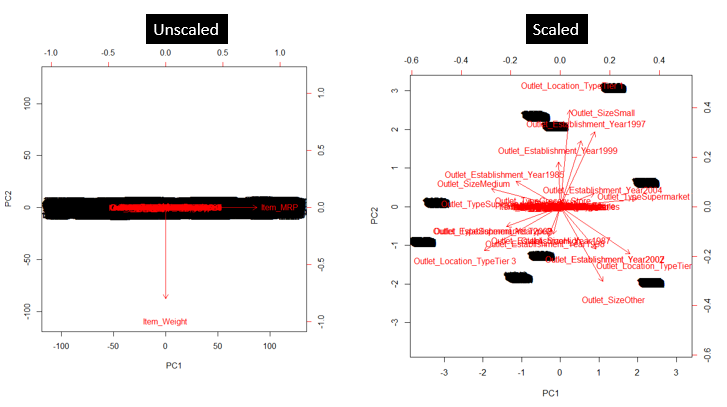
*Note: Partial least square (PLS) is a supervised alternative to PCA. PLS assigns higher weight to variables which are strongly related to response variable to determine principal components.*

**Why is normalization of variables necessary ?**

The principal components are supplied with normalized version of original predictors. This is because, the original predictors may have different scales. For example: Imagine a data set with variables’ measuring units as gallons, kilometers, light years etc. It is definite that the scale of variances in these variables will be large.

Performing PCA on un-normalized variables will lead to insanely large loadings for variables with high variance. In turn, this will lead to dependence of a principal component on the variable with high variance. This is undesirable.

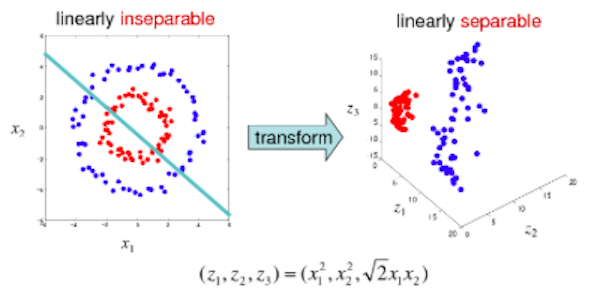
As shown in image below, PCA was run on a data set twice (with unscaled and scaled predictors). This data set has ~40 variables. You can see, first principal component is dominated by a variable Item\_MRP. And, second principal component is dominated by a variable Item\_Weight. This domination prevails due to high value of variance associated with a variable. When the variables are scaled, we get a much better representation of variables in 2D space.



**Kernal PCA:**

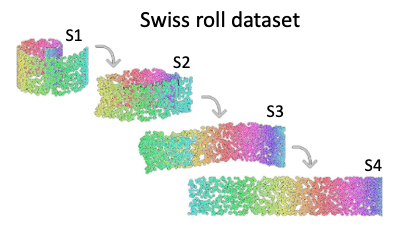
PCA applies linear transformation, which is just its limitation. *Kernel PCA* extends PCA to non-linearity. It first maps the original data to some nonlinear feature space (usually higher dimension), then applies PCA to extract the principal components in that space. This can be understood by Figure (B). The graph in the left shows the blue and red dots can not be separated using any linear transformation. But if all the dots are projected onto a 3D space, the result becomes linearly separable! We then apply PCA to separate the components.

Where does the intuition come from? Why does component separation become easier in a higher dimensional space? This has to go back to the Vapnik-Chervonenkis (VC) theory. It says mapping into a higher dimensional space often provides greater classification power.



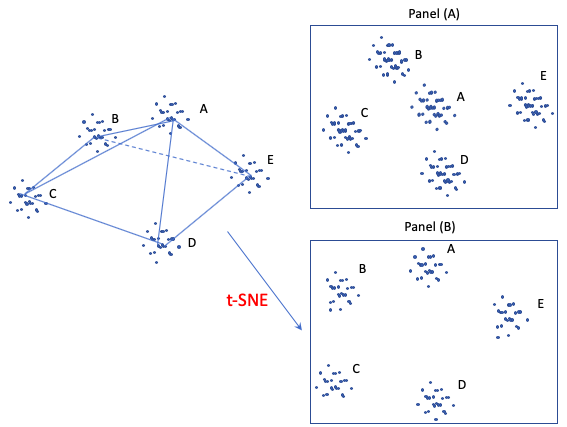
***t*-distributed Stochastic Neighbor Embedding (t-SNE)**

*t*-SNE is developed by [Laurens van der Maaten and Geoggrey Hinton](http://www.cs.toronto.edu/~hinton/absps/tsne.pdf). It is a machine learning algorithm for visualization that presents embedding high-dimensional data in a low-dimensional space of two or three dimensions.

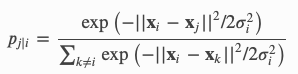


What is the best way to present the above three-dimensional Swiss roll to two-dimensional? Intuitively we want to “unroll” the Swiss roll to a flat cake. In mathematics, it means similar points will become nearby points and dissimilar points will become distance points.

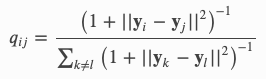
Figure (C) shows another example. It is a 3-dimensional tetrahedron with data points clustering in the vertex corners. If we just collapse the 3-dimensional graph to a 2-dimensional graph like Panel (A) does, it does not work well because group (A) becomes the center cluster. In contrast, Panel (B) is probably a better 2-D exhibit that that preserves the far distances between Cluster (A)-(E) while keeps the local distances of points in each cluster. *t*-SNE, a nonlinear dimension reduction technique, is designed to preserve the local neighborhoods. If a set of points cluster together on a t-SNE plot, we can be fairly certain that these points are close to each other.



*t*-SNE models the similarities among points. How does it define similarities? First, it is defined by the Euclidean distance between point *Xi* and *Xj*. Second, it is defined as the conditional probability that “the similarity of data point *i* to point *j* is the conditional probability *p* that point *i* would pick data *j* as its neighbor if other neighbors were picked according to their probabilities under a Gaussian distribution.” In the following conditional expression, if point *j* is closer to point *i* than other points, it has a higher probability (notice the negative sign) to be chosen.



t-SNE aims to match the above conditional probability *p* between *j* and *i* as well as possible by a low-dimensional space *q* between point *Yi* and *Yj,* as shown below. The probability *q* follows a fat-tailed Student-t distribution, thus the “*t*” in t-SNE comes from.



The next step is to find *Yi* such that the distribution *q* will be as close to the distribution *p* as possible. t-SNE uses the gradient decent technique, an optimization technique, to find the values.