- **Dimensionality reduction**
  - For 2-D & 3-D arrays we can imagine data by Scatter Plots.
  - For 4-D & 5-D arrays we can imagine data by Pair Plots.
  - For that of higher dimensions we should reduce dimensionality and reduce it to understandable or visualizable dimensions 2-D Or 3-D.
  - By reducing dimensionality, we transform the features into a new set of features which are less in number. The aim lies in preserving the variance which ensures least possible loss of information.
- **Row & Column Vector:**
  - These are one dimensional vector. By default, a 1D vector is column vector.
  - Column vector has dimension of 1 x n. Row vector has n x 1.
  - X € $\mathbb{R}^d$ where $\mathbb{R}$ is real number and d is its dimension.

$$\text{Column Vector: } \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}$$

$$\text{Row Vector: } \begin{pmatrix} 2 & 4 & 1 \end{pmatrix}$$

- **Dataset Representation**
  - In a dataset D = {x, y}, 'x' represents the input vector and 'y' represents the output label/value. and by default, X will be column vector.
  - For a single value, X will be a column vector and y will be a single value or scalar . But for an entire dataset X will be a matrix and Y will be a column vector
  - $\begin{array}{c|ccc} & f1 & f2 & Y \\ 1 & & & 1 \\ 2 & & & 0 \end{array}$ this is a way to represent input data in matrix form with having features as column.

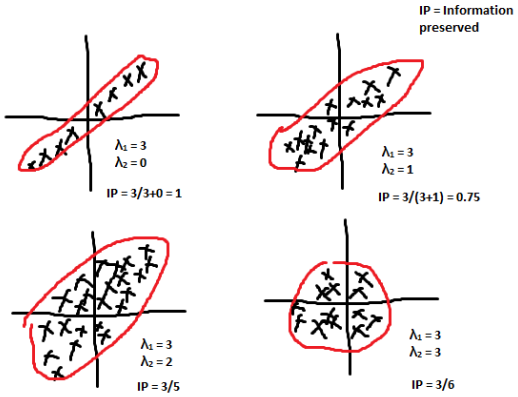
    While as each row will represent datapoint
- **Feature Normalisation**
  - **NORMALIZAION:** Data normalization is the process of rescaling one or more features to the range of 0 to 1. This means that the largest value for each feature is 1 and the smallest value is 0.
  - Normalization is a good technique to use when we do not know the distribution of our data or when we know the distribution is not Gaussian (a bell curve).
  - Normalization is useful when our data has varying scales and the algorithm that we are using does not make assumptions about the distribution of your data
  - Suppose we have dataset with 2 features i.e., Height & Weight

Height	Weight
157	59
167	65
172	70
153	50
181	80

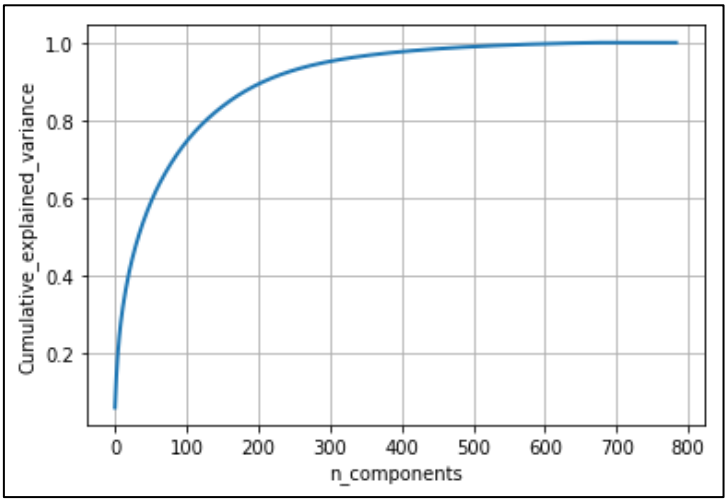
  - For comparison we normalise both these columns such that all values will lie between 0 & 1
  - for normalisation we use formula $a_i' = \frac{a_i - a_{min}}{a_{max} - a_{min}}$ where a is a feature
- **Feature Standardisation**
  - **Standardization:** Data standardization is the process of rescaling one or more features so that they have a mean value of 0 and a standard deviation of 1. Standardization assumes that your data has a Gaussian (bell curve) distribution.
  - This does not strictly have to be true, but the technique is more effective if our feature distribution is Gaussian.
  - Standardization is useful when our data has varying scales and the algorithm you are using does make assumptions about our data having a Gaussian distribution

$$a_i' = \frac{a_i - \bar{a}}{\sigma}$$

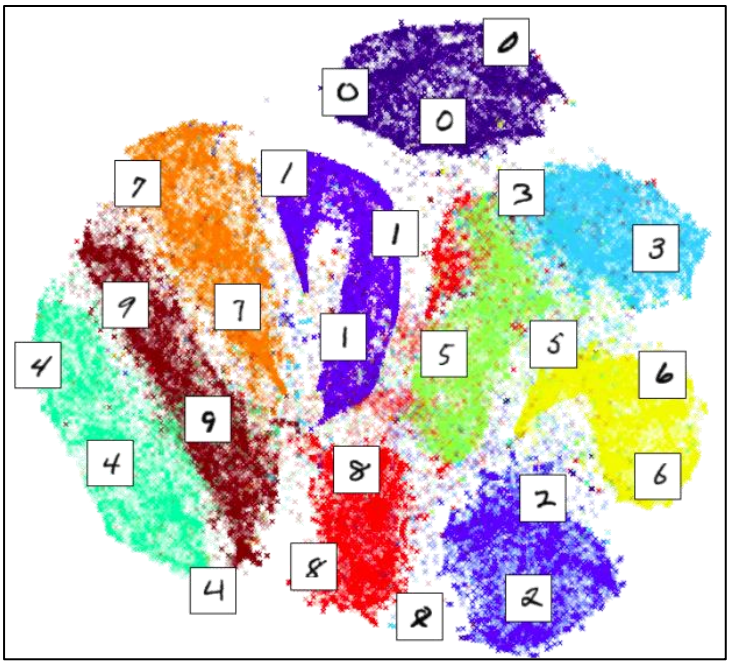
- **Principal Component Analysis (PCA)**
  - Principal Component Analysis (PCA) is a tool that has two main purposes:
    - To find variability in a data set.
    - To reduce the dimensions of the data set.
  - Reducing dimensions means that redundancy in the data is eliminated; This can make patterns in the data set clearer. Therefore, Principal Component Analysis is a good tool to use if we suspect that there is redundancy in data set.
  - Before going for PCA, we should do column standardization. CS can help us bring the features under uniform metric and we can observe the variance of each feature easily.
  - From Geometrical interpretation, if we are unable to eliminate one feature from scattered plot by analysing Variance, we can rotate the axis of features by some angle and align the rotated line on to the data points. This way we can analyse variance more lucidly and eliminate one feature.
    1. Speaking broadly the whole idea of PCA (dimension reduction technique) is to convert LARGER-DIMENSION to human analysable way i.e., 3d,2d or 1d.
    2. In a meanwhile, we have gone through many complex mathematical derivations which has lots of significance on its own. Let me explain them one by one.
    3. As we took example of 2-d model having features f1 on x-axis and f2 on y-axis. Since our main goal is to convert 2D to 1D, we had column standardized the values because its' mean' lies at origin and ' variance' =1. The main idea behind column standardization is to obtain some kind of linearity so that we can analyse the data in more precise way removing outliers.
    4. After performing column standardization we move to our main gain goal i.e converting 2d to 1d .Here we shifted our axis by some angle theta such that our axis can lie on the given dataset (points) forming new axes are f1dash and f2dash.
    5. Since we converted 2d to 1d by just shifting the axis the next step is to find the direction of points with maximum variance. Here you may ask what is the use of finding direction of points with maximum variance. The answer is in order to analyse the data better, variance should more. Let me explain you in a broad way, assume there are cars with different colours let's say (green, blue, red, etc). We can easily distinguish the cars based on its colour because we can differentiate colour which has lots of variations(variance). In the same way if have bunch of cars with same colour but with different shades. So, which is best example to differentiate cars obviously cars which have different colours.
    6. So that reason why we came across calculating variance/co variance (since mean is 0 both can be used interchangeably). After this our aim is to find the direction of max variance. So, for that we have unit vector concept to determine the direction.
    7. So, this the reason for using concept of unit vector. Since we don't know where is the xi's (data points) lie in space so we took the concept of projection of xis on unit vector(u1). This projection in fact determines the distance between our data points (xi's) and unit vector(u1).
    8. That's the reason we calculated minimum distance and not the maximum, because our unit vector should be close to data points(xi's).
    9. Now in this video we came across something called eigen values and eigen vectors (probably learnt in ug). The reason we are calculating eigen values and e vectors is to find unit vector for every element in matrix (data points) such that for obtaining max unit vector(u1) having large variance. From eigen values we can calculate corresponding eigen vector. Simply go through this link(https://math.stackexchange.com/questions/1425754/finding-eigenvectors-given-eigenvalues).
- **Variance Maximization-**
  - Variance is a measure of the "variability" of the data we have. Potentially the number of components is infinite (actually, after numeration it is at most equal to the rank of the matrix, as @jazibjamil pointed out), so you want to "squeeze" the most information in each component of the finite set you build.
    If, to exaggerate, you were to select a single principal component, you would want it to account for the most variability possible: hence the search for maximum variance, so that the one component collects the most "uniqueness" from the data set.
- **Eigenvectors and eigenvalues -**
  - The eigenvectors and eigenvalues of a covariance (or correlation) matrix represent the "core" of a PCA:
  - The eigenvectors (principal components) determine the directions of the new feature space, and the eigenvalues determine their magnitude.

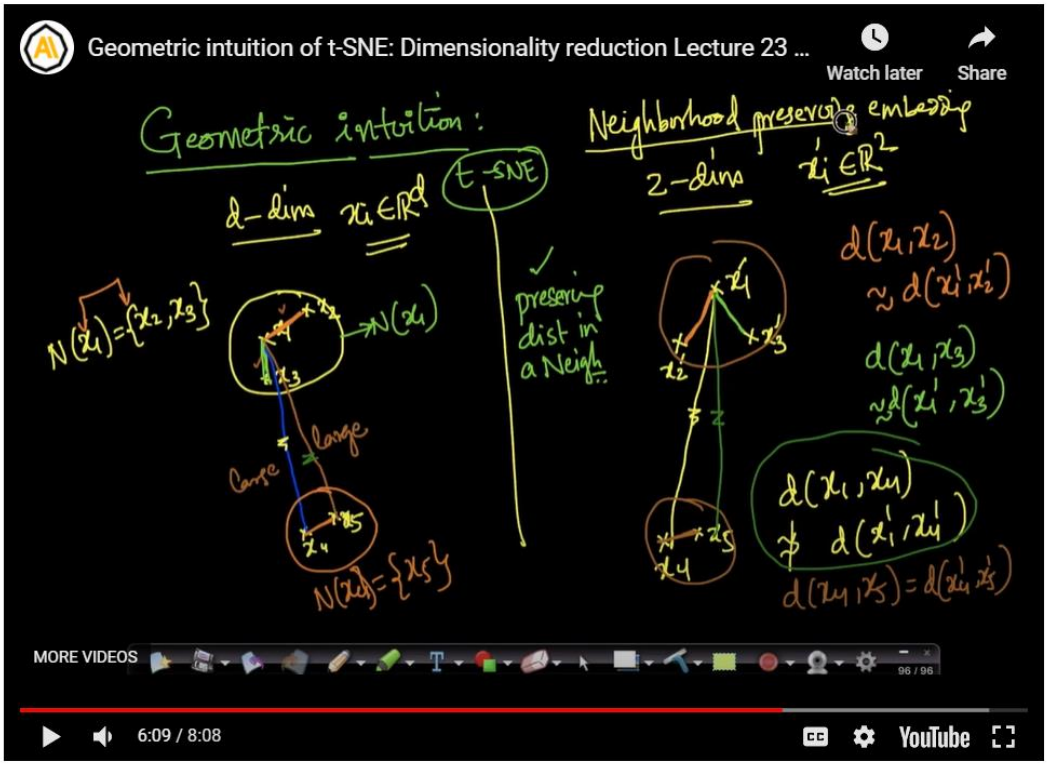


- Eigen vectors are perpendicular to each other. This implies, that dot product of pairs of Eigen vectors is zero. $V_i^T V_j = 0$
- No of Eigen values & No of vectors will be same as dimension of vector
- First eigen value will be maximum i.e., λ_1 and it will specify on which axis maximum values will be lying.
- Eigen value can also tell us that how much data is retained after dimensionality reduction by $\frac{\lambda_i}{\sum_1^d \lambda}$
- From figure below we can understand out of total 784 dimensions how much data will be retained for each n dimension reduction.



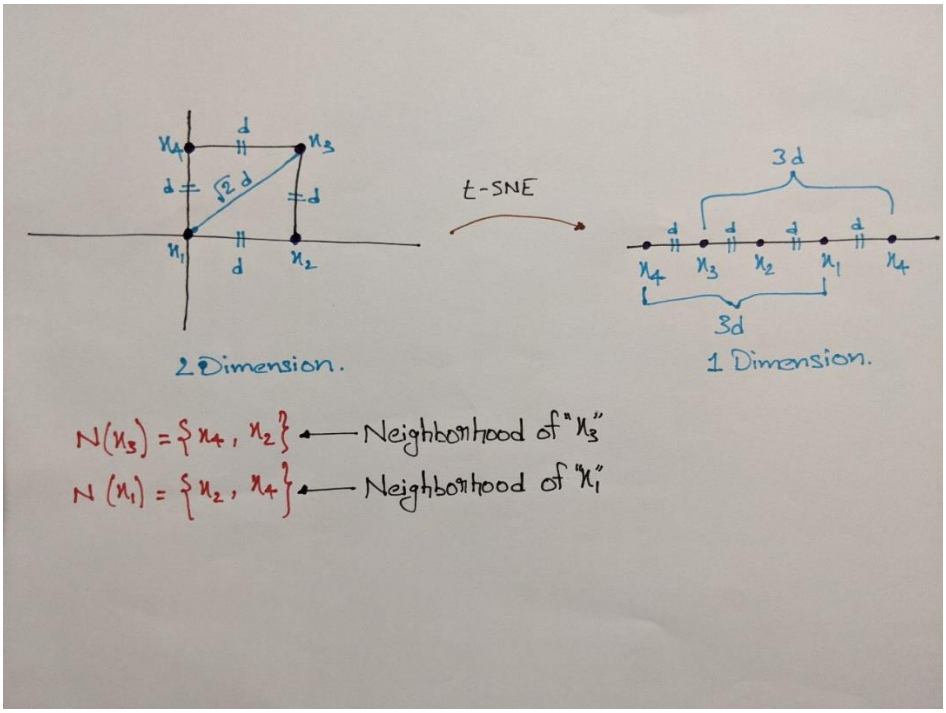
- **Visualize MNIST dataset**
  - https://colah.github.io/posts/2014-10-Visualizing-MNIST/
- **t-distributed stochastic neighbour embedding-**
  - t-SNE is a process of dimensionality reduction with neighbourhood preserving technique. Here neighbourhood means group of points lying near a single point.
  - The process of plotting a 2d point against n dimensional point is called as embedding



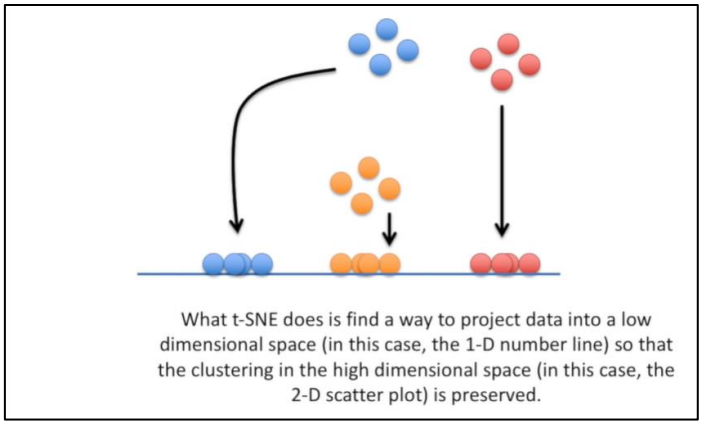


  - In simplest of layman words TSNE does nothing more than maintaining clusters from n-dimension to lower dimensions. So, assuming all our datapoints for each digits 1,2,3 etc are isolated clusters in n-dimension it maintains that geometry by transforming it into 2d isolated circles. Hence better visualization

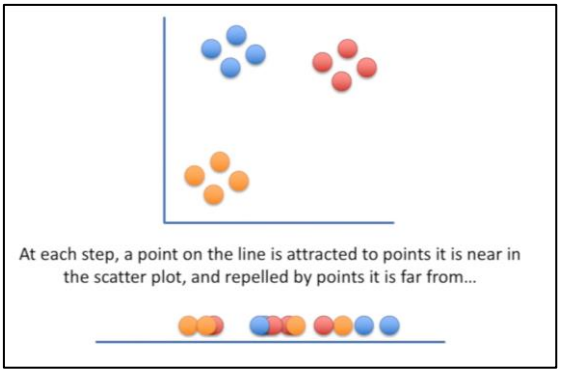
- Crowding Problem-
  - Sometimes, it is impossible to preserve distance in all neighbourhoods.



- **T-SNE**



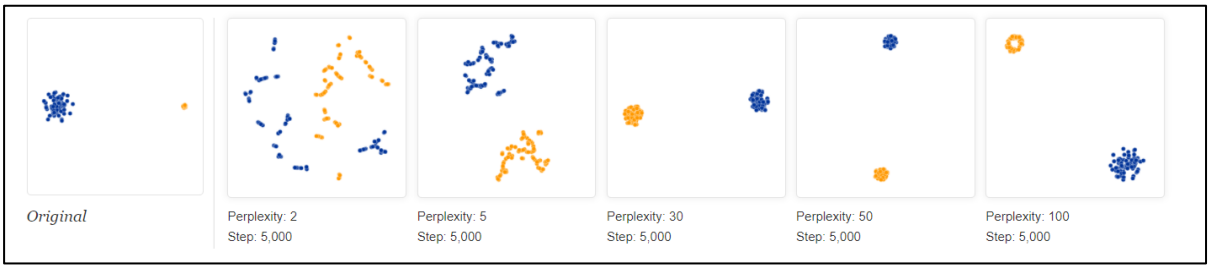
  - **Step –** No of iterations



  - T-SNE tries to process all the data in iterations and eventually it wants to reach a stage where the clusters are no more moving. At every iteration, T-SNE tries to move the points, find and improve the embedding and preserve the neighbourhoods as many as possible. For each iteration, we get a better solution. We should keep iterating till we get a stable configuration.
  - Initially the T-SNE algorithm makes a note of the points that are in the neighbourhood in the original dimensional space. Now as we keep progressing through the iterations, changes are made in the embeddings that were present in the previous iteration and by the end of all the iterations, we get a stable embedding and

the point in the neighbourhood of each point are almost the same as that of the original dataset. The dimensionality gets reduced.

- Perplexity- no of datapoints in the neighbourhood whose distance needs to be preserved while dimensional reduction.
- T-sne will expand dense points and shrinks sparse points



- **Steps**
  - Run steps/iterations till shape stabilizes
  - Keep changing perplexity (between 2 to n)
  - Rerun tsne multiple times