t-SNE

Karmanjyot Singh (2020101062)
Shubh Karman Singh Bhullar (2020101009)
Gaurav Singh (2020111014)
Venika (2020101072)

Introduction

Data visualization provides an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analyzing massive amounts of information and making data-driven decisions.

Do you think giving you the data of lets say 1 Million points in a table/Database file and asking you to provide your inferences by just seeing the data on that table is feasible?

This is when we make use of Data visualization, wherein all the data will be transformed into some form of **plots and analyzed** further from that. As being a human, we are more used to grasp a lot of info from diagrammatic representation than the counterparts.

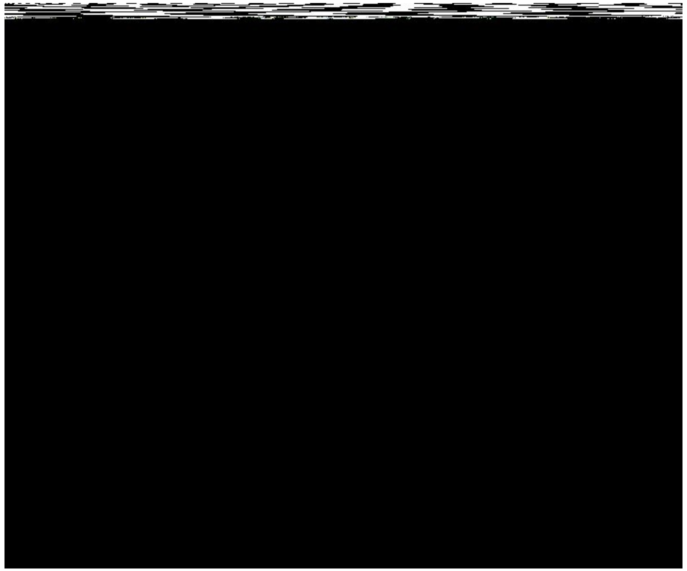
Dimensionality Reduction

To visualize high dimensional data, we need to reduce the dimensionality of the data to plot it in a 2D/3D graph.

Dimensionality reduction methods convert the **high-dimensional** data set X = {x1, x2,..., xn} into **two or three-dimensional (lower)** data Y = {y1, y2,..., yn} that can be displayed in a scatterplot.

Dimensionality reduction does not automatically reduce the existing features. This method will first summarize all features in a dataset into several components according to the algorithm we use. These could further be divided into :

1. **Linear** (example : Principal Component Analysis)
2. **Non-Linear** (example : t-SNE).



Linear vs Non-Linear Techniques

Linear transformations

They can be thought of as shifting and stretching data.

Non-Linear transformations

Whereas non-linear transformation will make more dramatic changes to data, such as turning data "inside out."

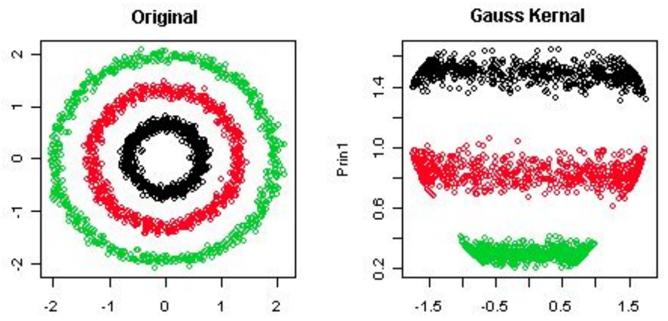


Fig : Non-Linear Transformation

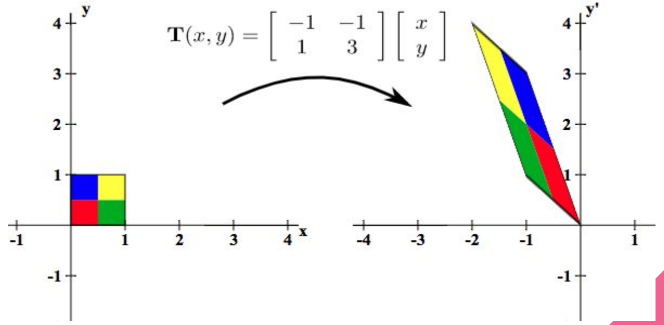

$\mathbf{T}(x,y) = \begin{bmatrix} -1 & -1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
x
y
y'

Fig : Linear Transformation

SNE : Stochastic Neighbour Embedding

1. Stochastic Neighbor Embedding (SNE) starts by converting the high−dimensional Euclidean distances between data points into conditional probabilities that represent similarities.

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}, \qquad q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}.$$

2. The similarity of datapoint x_j to datapoint x_i is the conditional probability, p j|i , that xi would pick x j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at xi .
3. Similarly, q j|i represents the probability in the lower dimensional space.
4. SNE minimizes the sum of Kullback−Leibler divergences over all data points using a gradient descent method. The cost function and gradient is given by：

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \qquad \frac{\delta C}{\delta y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

5. SNE performs a binary search for the value of σ i that produces a P_i with a fixed perplexity that is specified by the user.

SNE : Stochastic Neighbour Embedding

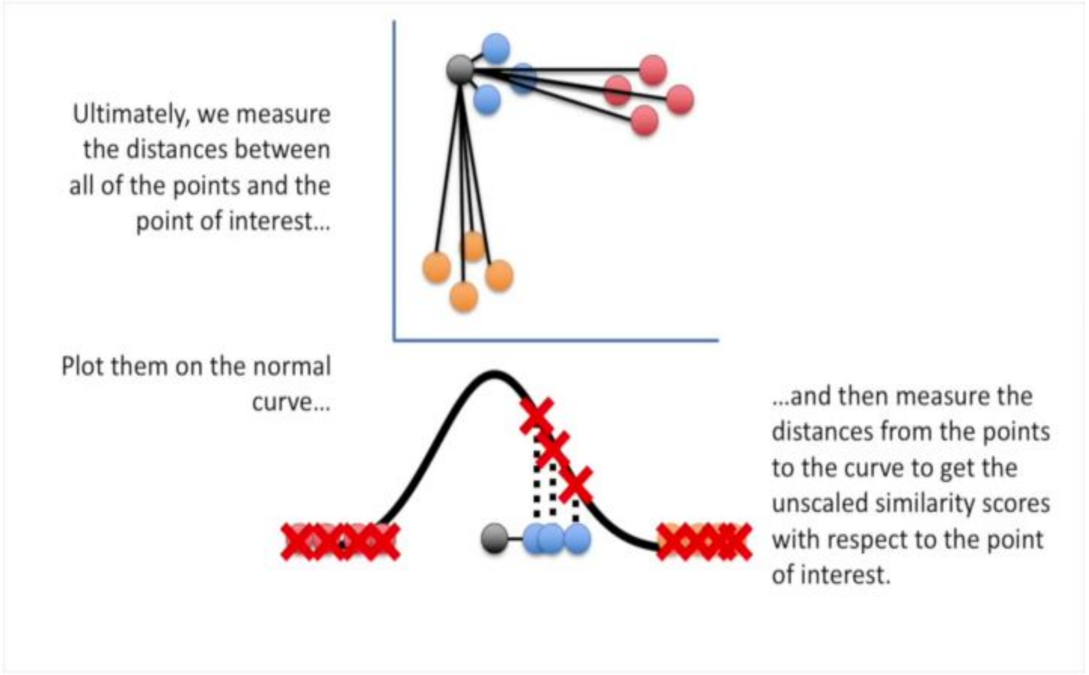

Ultimately, we measure the distances between all of the points and the point of interest...
Plot them on the normal curve...
...and then measure the distances from the points to the curve to get the unscaled similarity scores with respect to the point of interest.

Fig : Creating Projections in SNE

Symmetric SNE

Symmetric SNE has the property that p_ij = p_ji and q_ij = q_ji for ∀ i, j

1. **Symmetric Probabilities** : The SNE way of computing the pair wise affinities for points in the higher dimensional space, causes problems when the point x_i is an **outlier**, (i.e., all pairwise distances ||xi − xj||^2 are large for xi , and thus this dominates the cost function (the location of its low-dimensional map point y i has very little effect). So we simply tend to circumvent this problem by defining :

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

as joint probability for the matrix P, and thus ensuring that each data point xi makes contribution to the cost function.

2. **Gradient Descent :** The gradient of symmetric SNE is simple and faster to compute than its counterpart.

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j).$$

t-SNE

Although SNE constructs reasonably good visualizations, it is hampered by :

1) Cost function that is difficult to optimize.
2) Crowding problem.

This paper presents a new technique called “t-Distributed Stochastic Neighbor Embedding” or “t-SNE” that aims to alleviate these problems. The cost function used by t-SNE differs from the one used by SNE in two ways:

1) It uses a **symmetrized** version of the SNE cost function with simpler gradients.
2) It uses a **Student-t distribution** rather than a Gaussian to compute the similarity between two points in the low-dimensional space.

Crowding Problem

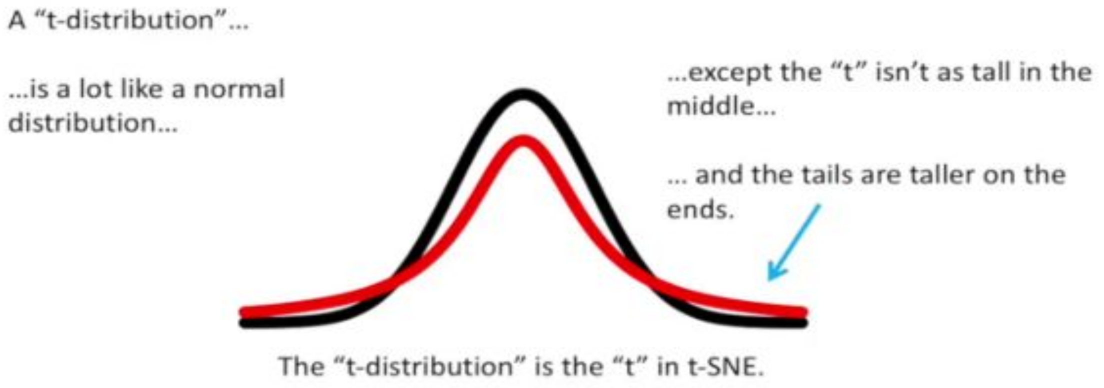
Abstract : The area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby datapoints

There are a number of reasons why pairwise distances in a two-dimensional map cannot accurately model distances between points on a ten-dimensional manifold. In ten dimensions, for example, it is possible to have 11 data points that are mutually equidistant, which is impossible to model faithfully in a two-dimensional map.

Hence, if we want to model the small distances accurately in the map, most of the points that are at a moderate distance from datapoint i will have to be placed much too far away in the two-dimensional map. In SNE, the spring connecting datapoint i to each of these too-distant map points will thus exert a very small attractive force. Although these attractive forces are very small, the very large number of such forces **crushes** together the points in the center of the map, which prevents gaps from forming between the **natural clusters**

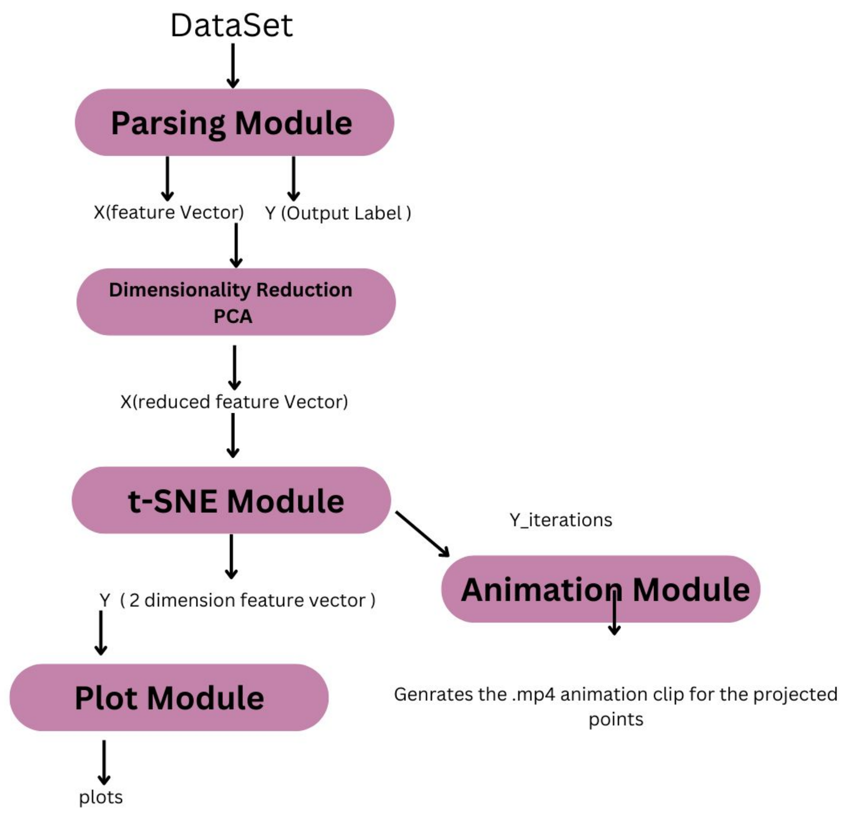
t-SNE

Using the heavy tail-distribution for the lower dimensional space allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance in the map and, as a result, it eliminates the unwanted attractive forces between map points that represent moderately dissimilar data points.


A "t-distribution"...
...is a lot like a normal distribution...
...except the "t" isn't as tall in the middle...
... and the tails are taller on the ends.
The "t-distribution" is the "t" in t-SNE.

Implementation Pipeline

The Implementation Pipeline



The main tsne pipeline could be broken into the following modules :

1. Parsing
2. Dimensionality Reduction
3. t-SNE Module
4. Plot and animation Module

The Implementation Pipeline

The main t-SNE algorithm could be divided into the following components :

Parsing Module

- Reading and parsing the dataset, and separating the data into the features, and labels, and returning the feature vector **X** and the label vector **y**
- This could be found in the parser folder, which returns the desired dataset feature vectors and their corresponding labels.

Dimensionality Reduction : PCA

- Principal component analysis (PCA)'s main goal is to minimize the number of connected variables in a data set while preserving as much of the data set's inherent variance as possible. A new set of variables called principal components (PCs), which are uncorrelated and are sorted so that the first few keep the majority of the variance included in all of the original variables, is used to achieve this.
- The reference paper tsne suggests reducing the dimensions of the data to 30 or so components, using the standard linear dimensionality reduction method **PCA**.
- This module takes input as the feature vector **X (with features > 30)** and returns the feature vector X, with reduced features (= 30)
- We apply the main t-sne algorithm on this vector with the reduced dimensions.

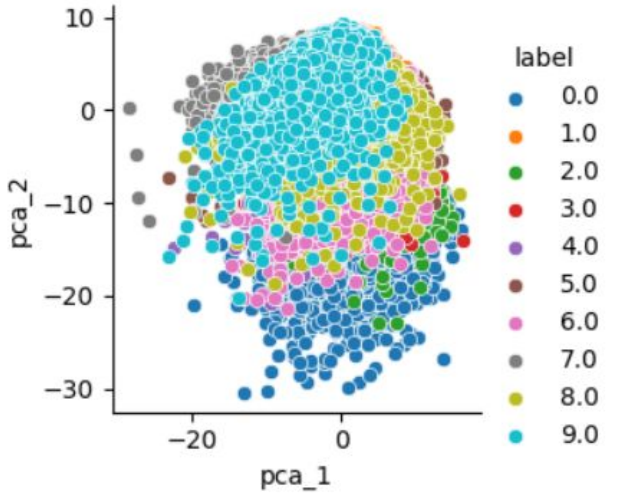


Fig : 2-PC plot for the mnist data set

The Implementation Pipeline

Core t-SNE

1. Measure the similarities between points in high dimensional space by using the euclidean distance between the points and forming a gaussian distribution centred around that point. This gives us a set of probabilities for all points. These probabilities are proportional to the similarities and represent local similarities in this high-dimensional space. The Gaussian distribution or circle can be manipulated using what's called perplexity, which influences the variance of the distribution (circle size) and essentially the number of nearest neighbors. Binary search is used to find the optimal sigmas that produces fixed perplexity mentioned by user:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}, \qquad Perp(P_i) = 2^{H(P_i)}, \qquad H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

2. In second step we do similar thing as we did in first step, but instead of using a gaussian distribution we use t-Distribution with one degree of freedom, which are also called cauchy distribution. This gives us a second probability, which is mathematically given by:

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}.$$

The Implementation Pipeline

Where qij's are the low dimensional counterparts of pij's. It have more heavier tails then normal distribution, so it allows for better modelling of far apart distances.

3. The third step is that we want these probabilities of low dimensional space(qij) to reflect those of high dimensional space(pij) as best as possible. We want the two map structures to be similar. We measure the difference between the probability distributions of the two-dimensional spaces using Kullback-Liebler divergence (KL). We then minimize this KL cost function using gradient descent. Mathematically KL cost function is given by:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}.$$

The Implementation Pipeline : Animation

Animation and Plotting: Displaying the Results

- This module takes in the reduced dimension points Y and generates figures and plots for the same in the lower dimensional space, providing useful insights and visualizations of the data set points, with similar or related points clustered together and placed nearby in the low dimensional feature space.
- We implemented the animation of the tsne algorithm, wherein we present the variations the points undergo as they projected in the lower dimensional space as the gradient descent step is followed.
- This shows the step by step change in the projections, and clustering of similar points in the lower dimensional space.

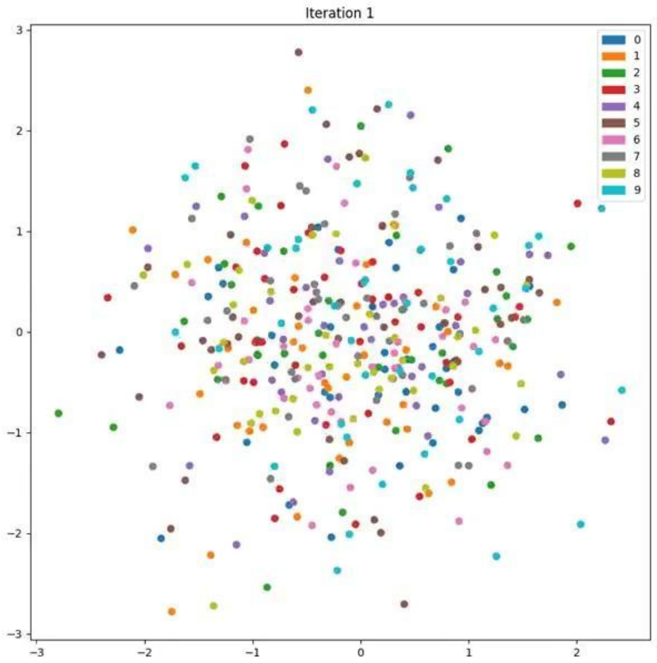


fig : projection update during gradient descent

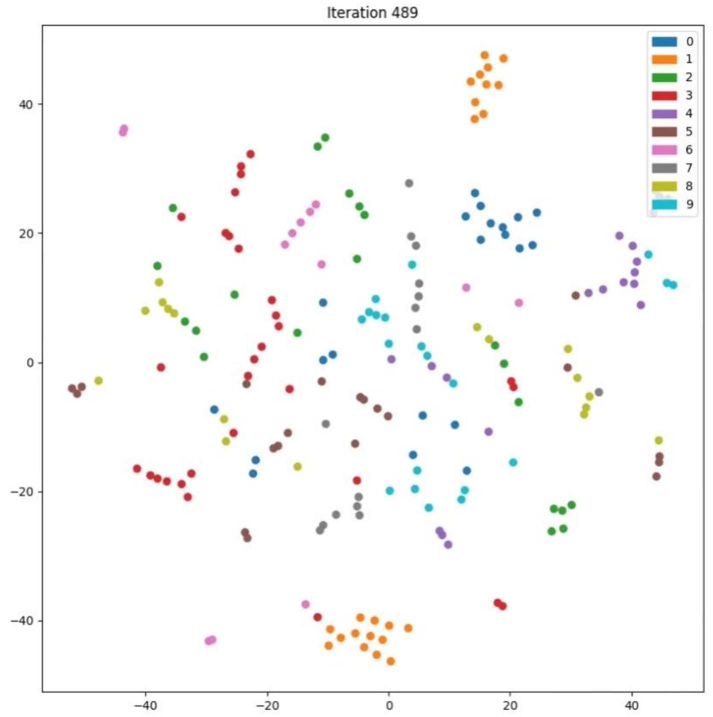
Results And Analysis

Results And Analysis : Plots

To compare the results of our t-SNE implementation, we compare the projections generated by t-SNE with those of :

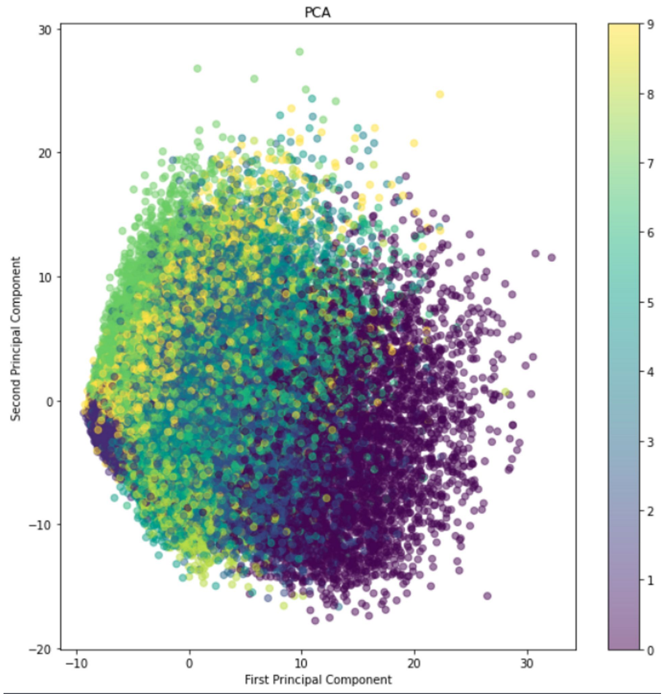
1. **Isomap** and,
2. **PCA**

for our analysis. The following slides have detailed comparison for the plots generated.

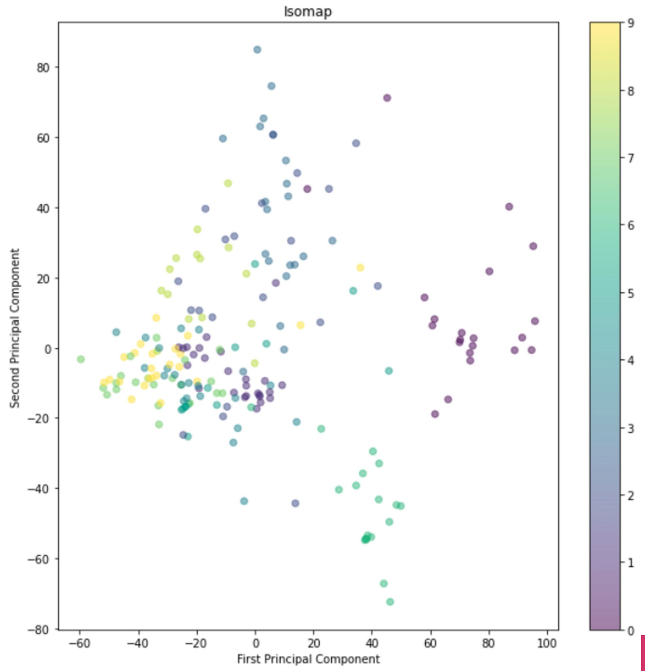


MNIST Dataset

MNIST



Plot : PCA

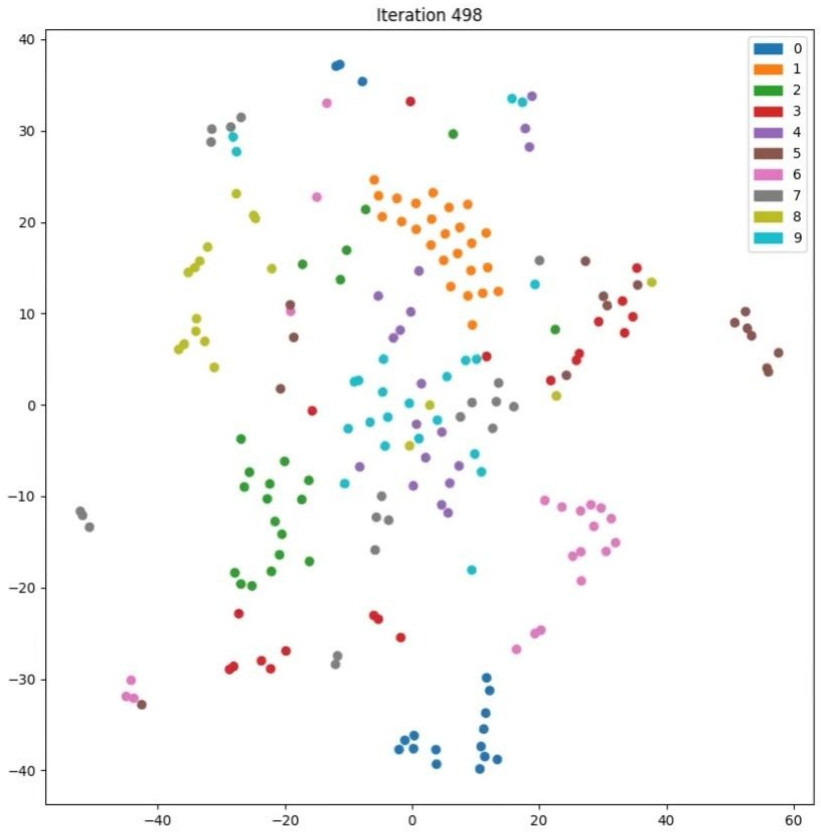


Plot : IsoMap

MNIST

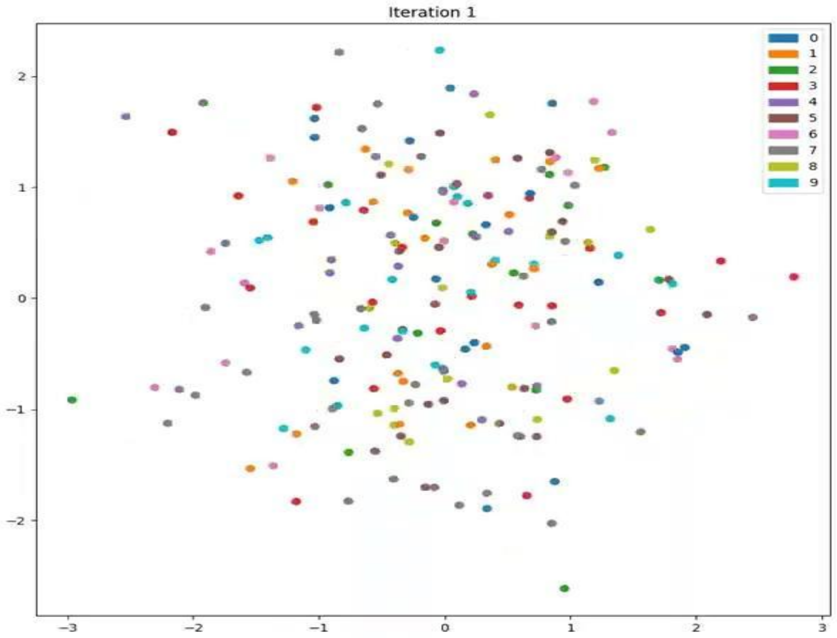
Seeing from our previous plots of PCA and Isomaps, we could clearly infer that the tsne technique produces better visualisation for the mnist data plots with retaining the **local structure** of the data samples.

We could clearly see the similar classes cluster together, while different classes tend to have a considerable separation between them even in the lower dimension projection.



Plot : tSNE

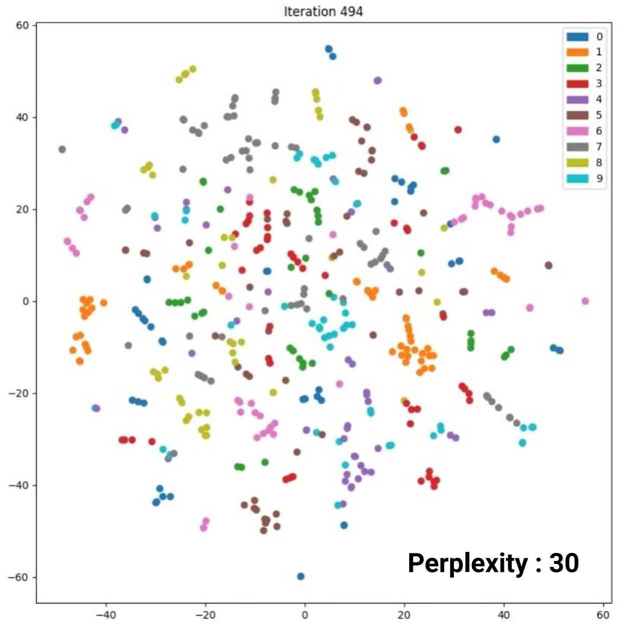
MNIST

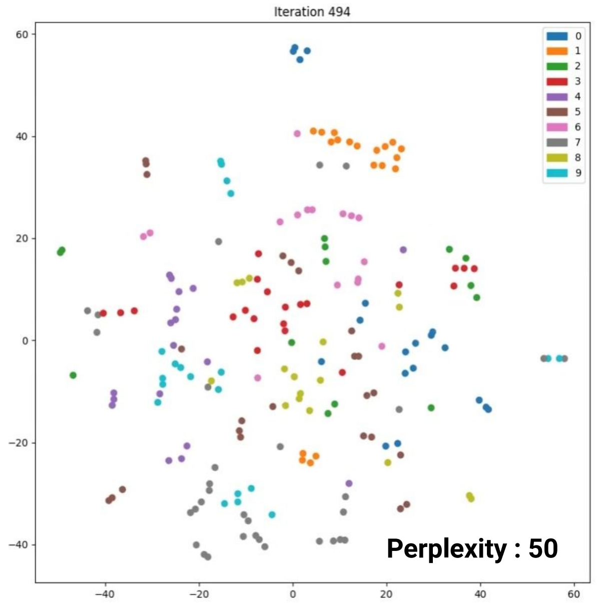


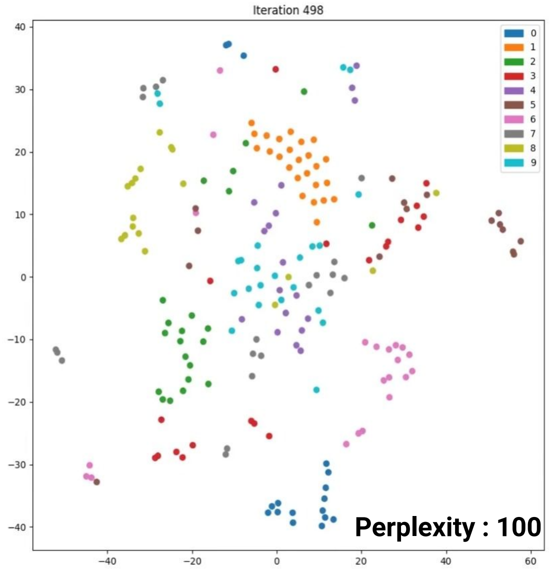
Video : MNIST perplexity : 50

MNIST : Varying Parameters

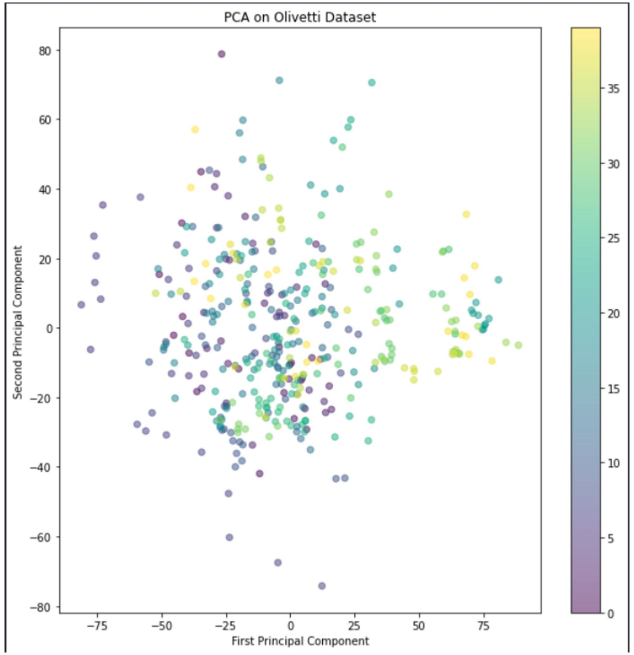
Since, tsne algorithm's performance greatly depends on the choice of parameters and attributes, we compared the effect of the **perplexity** parameter on the visualisations created. We see that for greater perplexity values, clusters tend to become more crowded.

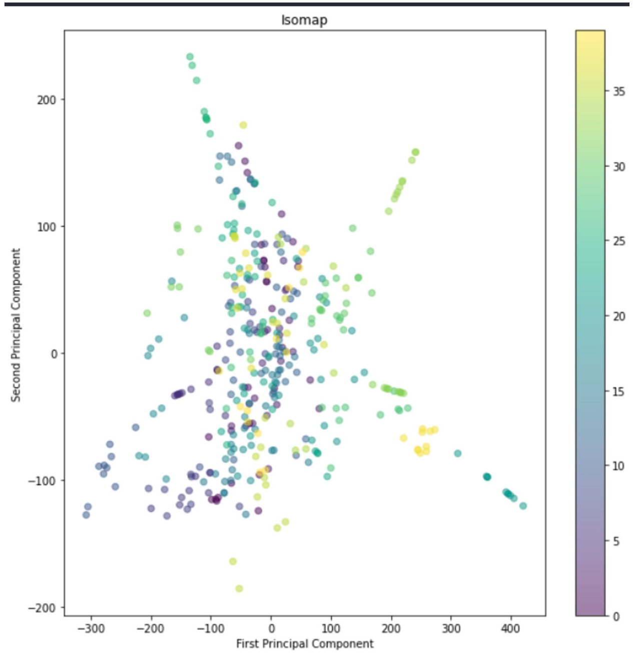






Oliverfetti





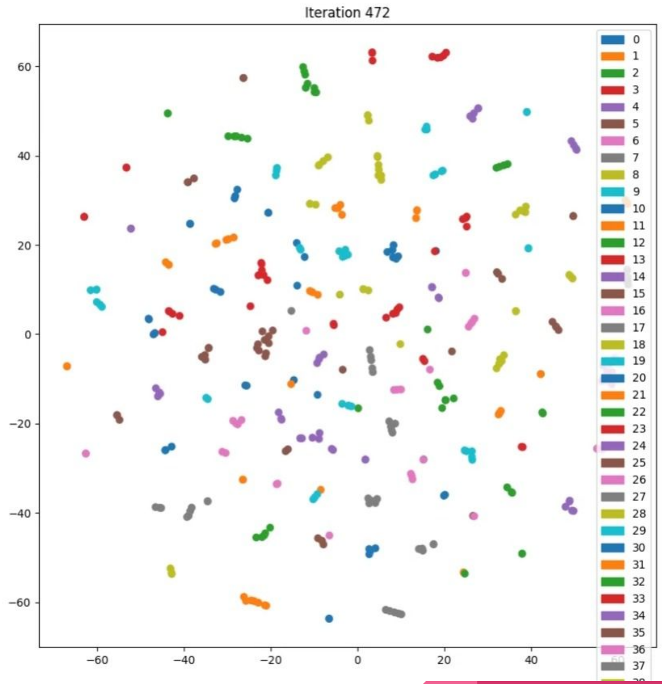
Plot : PCA

Plot : IsoMap

Oliverfetti

Seeing from our previous plots of PCA and Isomaps, we could clearly infer that the tsne technique produces better visualisation for the mnist data plots with retaining the **local structure** of the data samples.

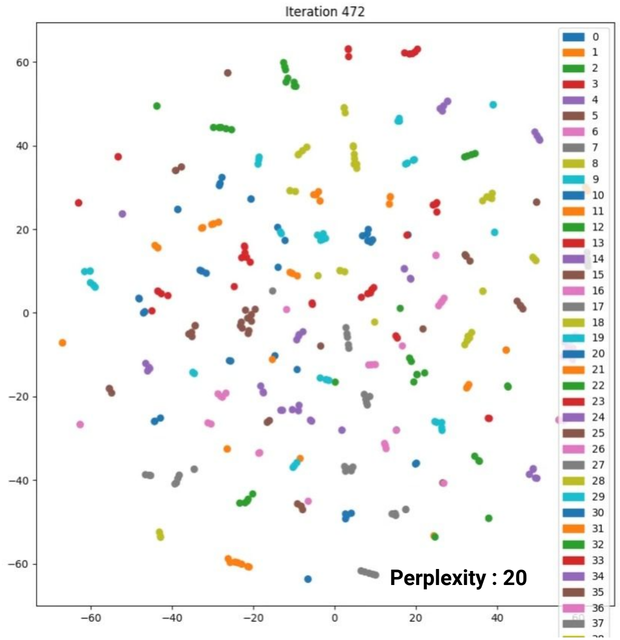
And as we see that none of the classes become clearly separable in either pca or isomap tSNE does a better job at doing this

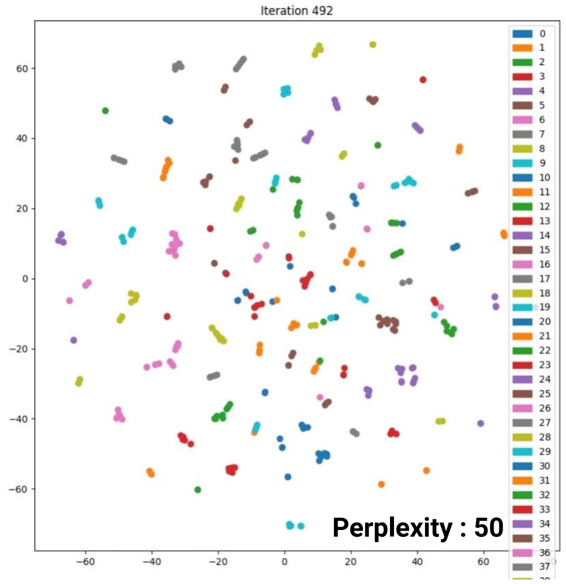


Plot : tSNE

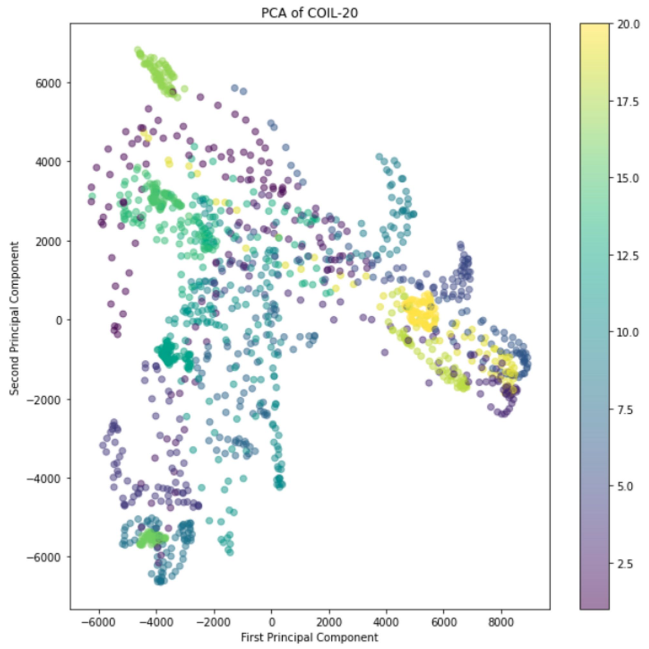
Oliverfetti : Varying Parameters

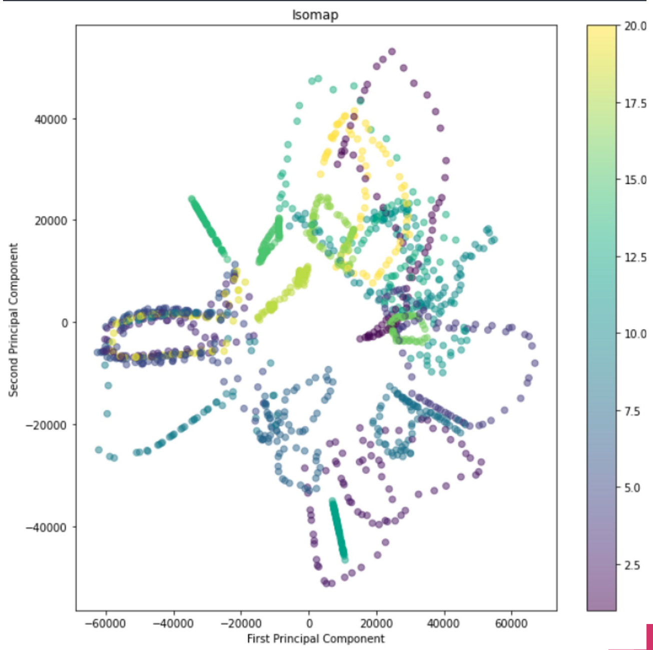
Since, tsne algorithm's performance greatly depends on the choice of parameters and attributes, we compared the effect of the **perplexity** parameter on the visualisations created. We see that for greater perplexity values, clusters tend to become more crowded.





Coil 20





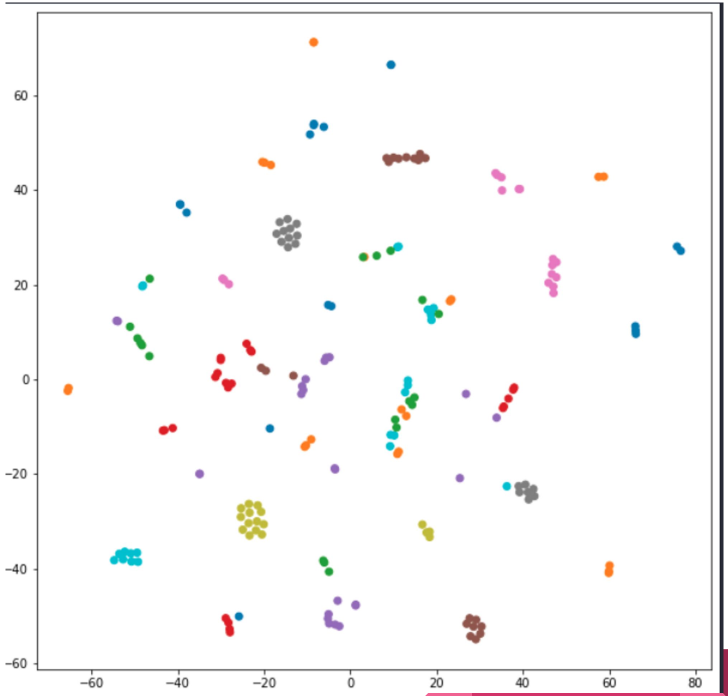
Plot : PCA

Plot : IsoMap

Coil 20

Seeing from our previous plots of PCA and Isomaps, we could clearly infer that the tsne technique produces better visualisation for the mnist data plots with retaining the **local structure** of the data samples.

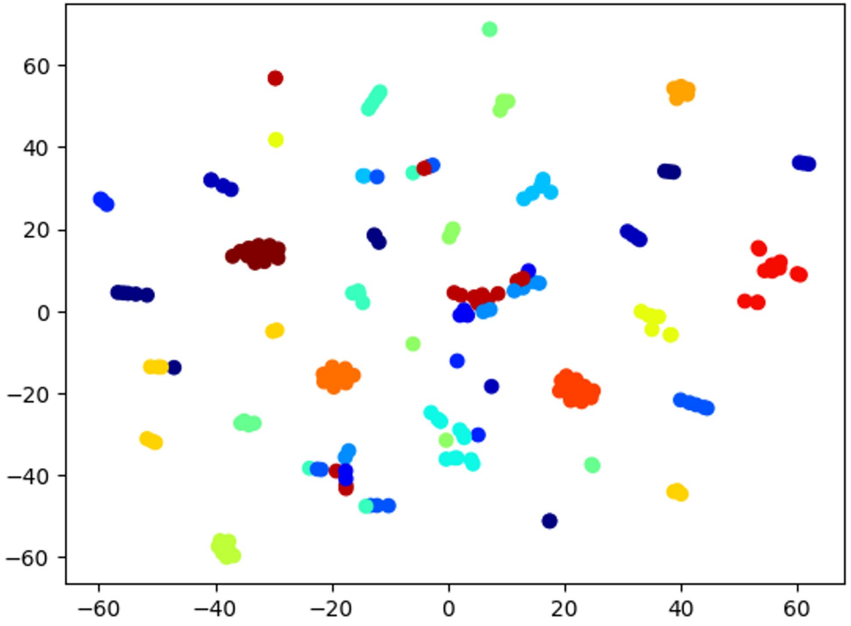
And as we see that none of the classes become clearly separable in either pca or isomap tSNE does a better job at doing this



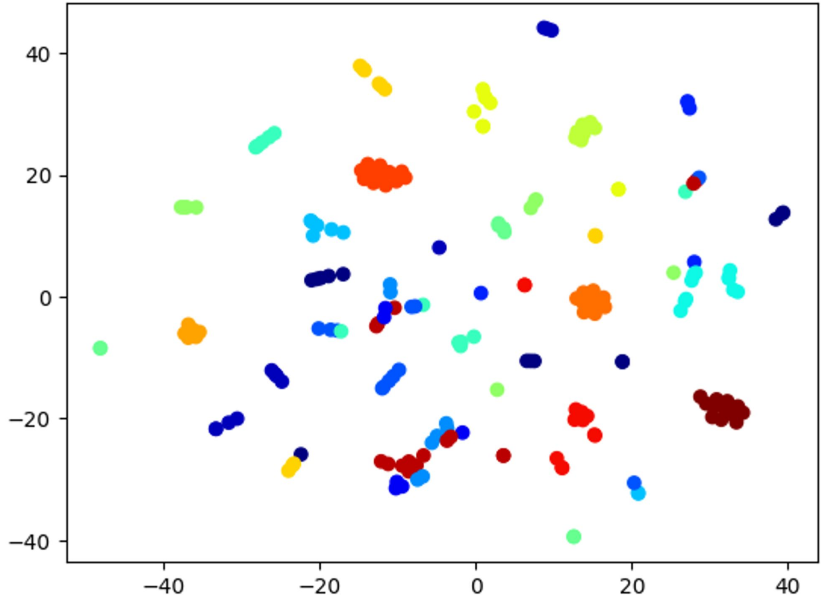
Plot : tSNE

Oliverfetti : Varying Parameters

We varied the learning rate alpha for the gradient descent step, and thereby altering the rate of learning for our dataset i.e. the rate at which the plot stabilizes.

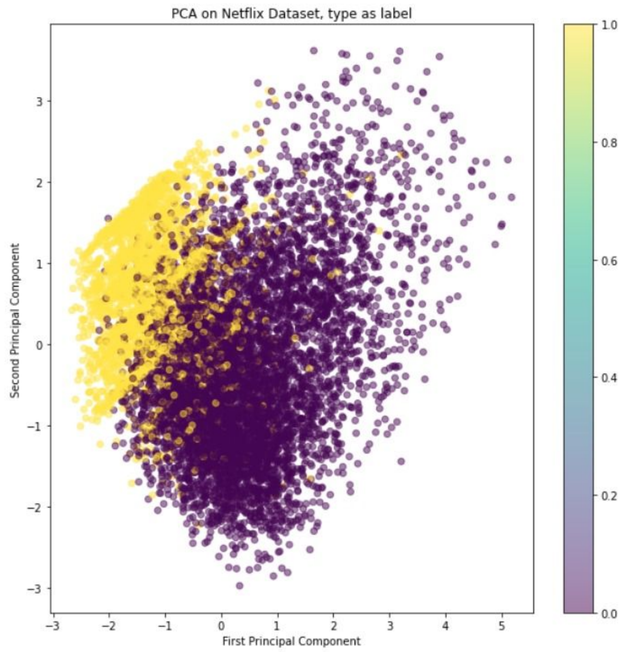


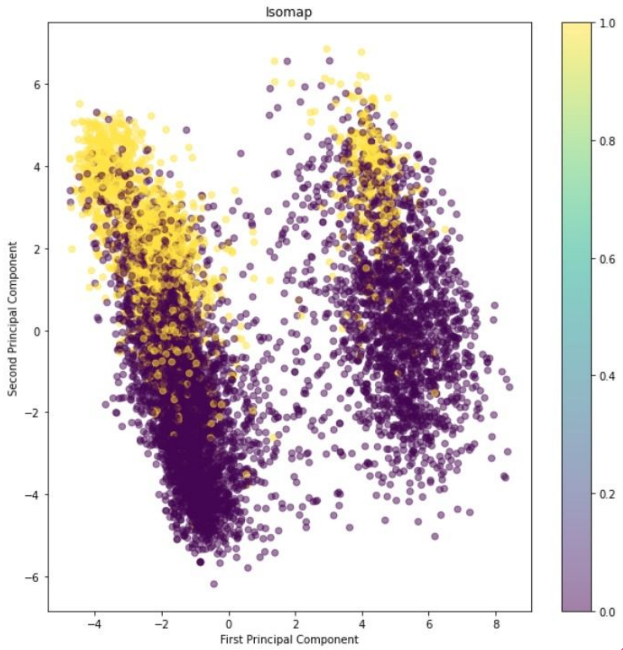
Learning Rate : 200



Learning Rate : 50

Netflix





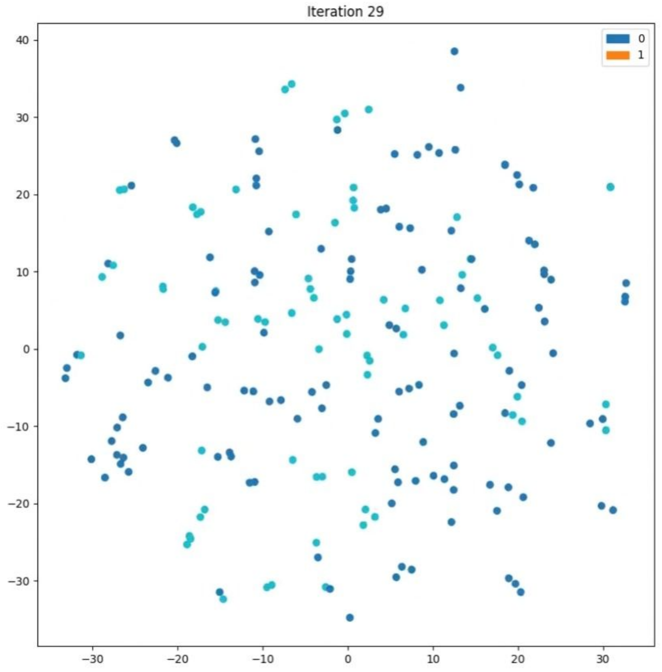
Plot : PCA

Plot : IsoMap

Netflix

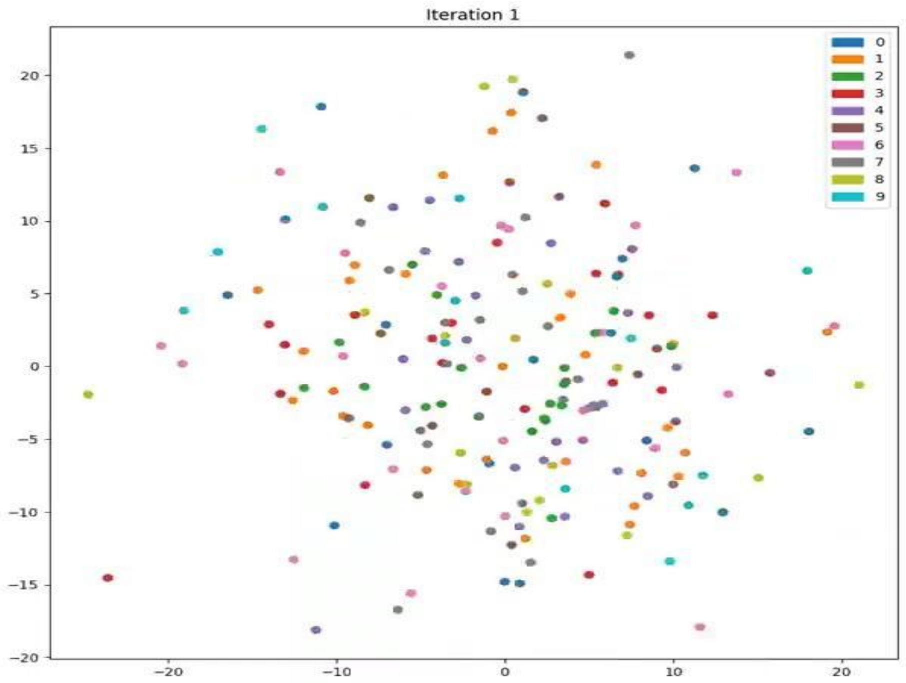
Seeing from our previous plots of PCA and Isomaps, we could clearly infer that the tsne technique produces better visualisation for the mnist data plots with retaining the **local structure** of the data samples.

And as we see that none of the classes become clearly separable in either pca or isomap. tSNE does a better job at doing this



Plot : tSNE

Early Exaggeration : MNIST



Work Distribution

1. Venika Sruthi Annam (2020101072)
    a. Data Set Parsing
    b. Tsne implementation on Datasets
2. Shubh Karman Singh Bhullar (2020101009)
    a. Animation Module
    b. Binary Search
    c. Probability Matrix Computation
3. Karmanjyot Singh (2020101062)
    a. Animation Module
    b. Dimensionality Reduction Module (PCA)
    c. Gradient Descent step
4. Gaurav Singh (2020111014)
    a. Perplexity and Joint probability matrix and Optimal variance calculation
    b. Dimensionality Reduction Module (PCA)
    c. Plots Generating Function