Indian Institute of Technology, Kharagpur

# DIMENSIONALITY REDUCTION AND t-SNE

## DIMENSIONALITY REDUCTION

In simpler words, dimensionality reduction is nothing but the reduction of n dimension data to n' dimension data, where n > n'.

### WHY USE DIMENSIONALITY REDUCTION?

- ☐ Saves computational resources
- ☐ Visualisation of high-dimensional data
- ☐ Mitigate the problem of overfitting

## PRINCIPAL COMPONENT ANALYSIS

The idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

1. Standardise the Data(the mean of the data is made 0).

$$x_{new} = \frac{x - \mu}{\sigma}$$

2. Computation of covariance matrix by the formula given below:

   $\mathbf{X^TX/m}$

   Where X is the input dataset matrix of the order m×n

   Where m is the number of datapoints and n is the number of dimensions.

3. Computation of the eigenvalues and eigenvectors of the covariance matrix to identify the principle components

# **t**-DISTRIBUTED **S**TOCHASTIC **N**EIGHBOUR **E**MBEDDING

t-Distributed Stochastic Neighbour Embedding (t-SNE) is an **unsupervised, non-linear dimensionality reduction technique primarily used for data exploration and visualising high-dimensional data**.

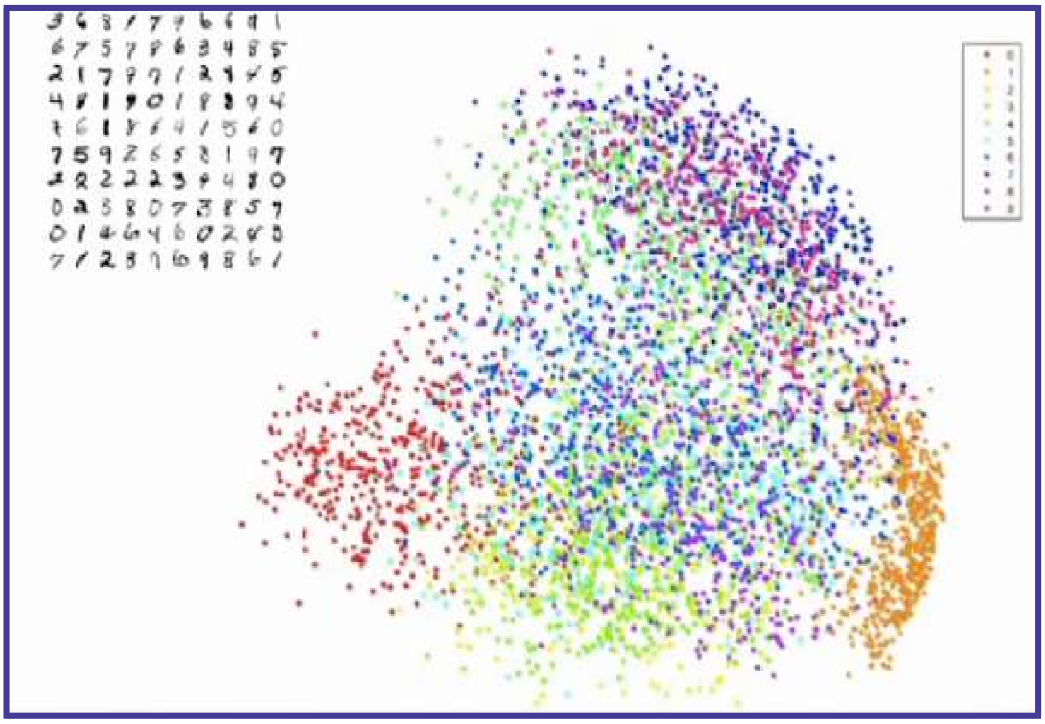
## Why is t-SNE better than PCA?

t-SNE can handle outliers whereas PCA is highly influenced by outliers present in the data. Outliers are the points that deviate highly from the overall pattern of the dataset.

t-SNE does a better job (it tries to preserve topology neighbourhood structure) as compared to PCA when it comes to visualising the different patterns of the clusters. t-SNE preserves the local structure whereas PCA focuses more on explaining the variance in the data.

4

This is the output obtained after applying PCA on the MNIST dataset.

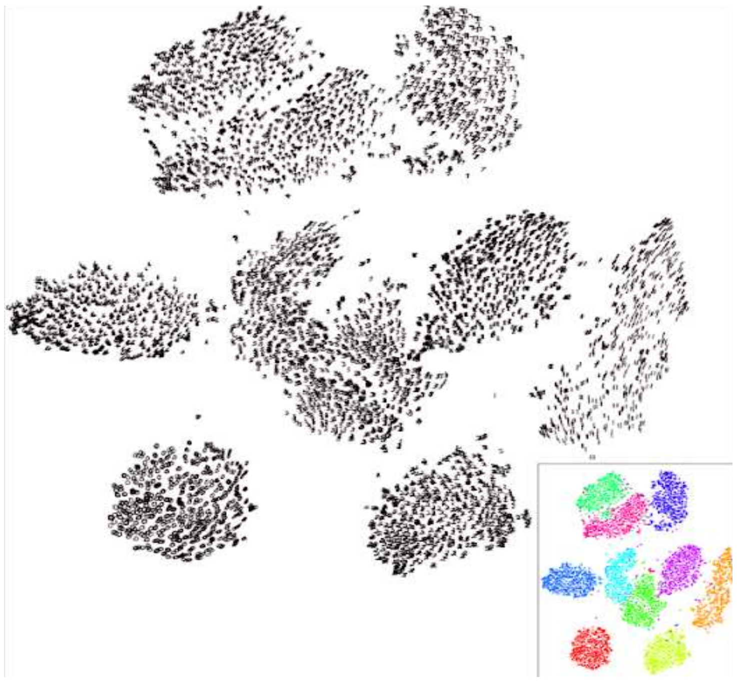


There is a lot of overlapping and less clustering.

PCA, at least here, is able to separate highly dissimilar ones(the digits with very less overlapping features) like 1's and 0's.But it is unable to represent the structure of the digits in between(slightly similar digits like 4,7,9 etc.).

This is the plot obtained after applying t-SNE on MNIST dataset.

The digits are themselves plotted point-wise. You can see that in the zeroes' cluster which is at the bottom left corner, the circular ones are to the left and more oblique and thinner ones are towards the right.



So the local structure is well preserved and the cluster of digits of 1's are far from the cluster of 0's and are located on the right side because 1 and 0 have very less overlapping features.

## STEPS INVOLVED IN t-SNE

1. Calculating **conditional probability distribution** of pairs of points from high dimensional data.
2. Computing **joint probability distribution** by taking the average of conditional probability distribution for a pair of points.
3. Creating a dataset of points in the target dimension and then calculating the **low dimensional joint probability distribution** for them.
4. **Computing Cost Function.**
5. Using Gradient Descent and momentum to **move the points in the low dimensional graph** after each iteration.

### STEP 1&2: CALCULATION OF CONDITIONAL PROBABILITY AND JOINT PROBABILITY DISTRIBUTION

The similarity of datapoint $x_j$ to datapoint $x_i$ is the conditional probability, $p_{j|i}$ , that $x_i$ would pick $x_j$ as its neighbour if neighbours were picked in proportion to their probability density under a Gaussian centred at $x_i$ .

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

On summing $p_{ij}$ for all points $x_j$ other x, then for that point $x_i$, the sum would be equal to 1. From this $\sigma_i$ is obtained.
(n=number of higher dimensions).

## STEP 3: COMPUTING LOW DIMENSIONAL JOINT PROBABILITY DISTRIBUTION

The low dimensional graph is initialised randomly and the low dimensional joint probability distribution is calculated using the given formula.

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

The expression on the RHS is a Student t-distribution expression.

## STEP 4: COMPUTING COST FUNCTION

**Cost function in t-SNE tries to reduce the difference between probability distribution values of the pairs of points in lower and higher dimensions**.

Cost Function basically calculates the sum of difference of high dimensional and low dimensional joint probability distribution scores of all pairs of points.

$$\begin{aligned} C = KL(P\|Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij}. \end{aligned}$$

## STEP 5: USING GRADIENT DESCENT AND MOMENTUM TO UPDATE LOCATION OF POINTS IN THE LOW DIMENSIONAL GRAPH

Calculation of low dimensional joint probability distribution, gradient descent and the low dimensional data representation set is done in each iteration as after each iteration, position of all points change as they move towards their cluster and away from others.

Gradient Descent

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

Set containing low dimensional representation of data

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta\frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$$

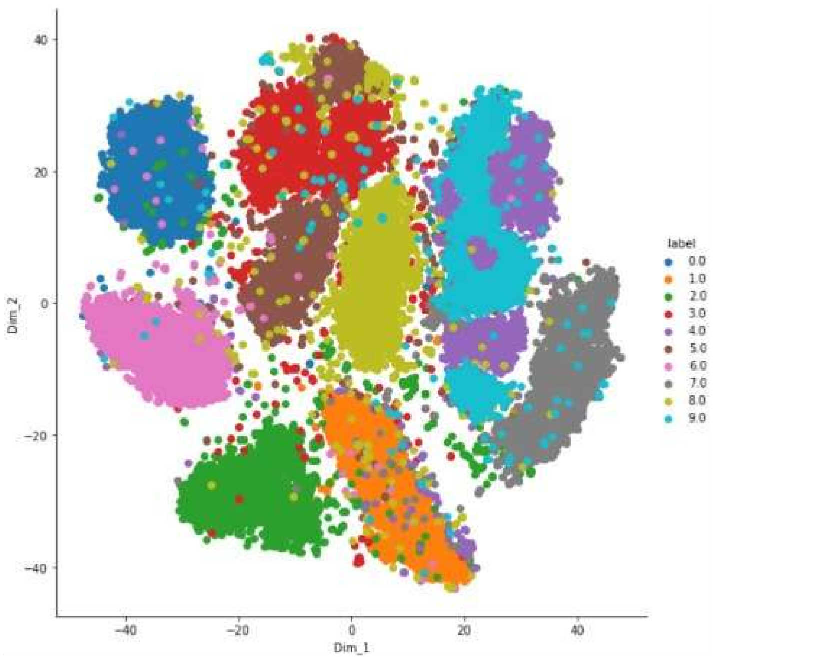
As this set containing low dimensional representation of data gets updated after each iteration, the points move towards each other and eventually we get the clusters in the lower dimension.

## CODE FOR t-SNE

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sn
import scipy as sp
from keras.datasets import mnist
(X_train, y_train), _ = mnist.load_data()
np.shape(X_train)
data = np.reshape(X_train,(len(X_train),784))
labels=y_train
data,labels
from sklearn.preprocessing import StandardScaler
std_data = StandardScaler().fit_transform(data)
```

```
from sklearn.manifold import TSNE
model =
TSNE(n_components=2,perplexity
100,learning_rate=1000,n_iter=
500,random_state=0)
tsne_data =
model.fit_transform(data[:25000])
tsne_data = np.vstack((tsne_data.T,
labels[:25000])).T
tsne_df = pd.DataFrame(data=tsne_data,
columns=("Dim_1", "Dim_2", "label"))
sn.FacetGrid(tsne_df, hue="label",
height=8).map(plt.scatter, 'Dim_1',
'Dim_2').add_legend()
plt.show()
```

## OUTPUT OF t-SNE ON 40% MNIST DATA



## REFERENCES

https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1

https://www.google.com/url?sa=t&rct=j&q&esrc=s&source=web&cd&cad=rja&uact=8&ved=2ahUKEwiJgJuC1f34AhVR-TgGHdnmC1gQFnoECEoQAQ&url=https%3A%2F%2Flvdmaaten.github.io%2Ftsne%2F&usg=AOvVaw2ULZkob00wOk8f_pcqNIZb&hl=en_GB

https://arxiv.org/pdf/1802.03426.pdf

https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

## TEAM MEMBERS

ADITYA KRISHNA DAS

YASASWANI RONGALI

YASH KUMAR