T-SNE (t-distb stochastic Neighborhood Embedding)

15.1 what is t-SNE

→ state of the art/best dim-red ⟶ Visualization.

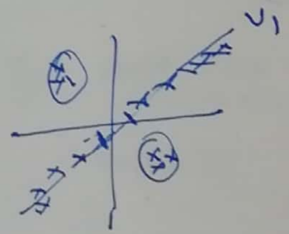
→ PCA basic; old

→ multidimensional scaling, Sammon mapping, Graph based technique

→ T-SNE : 2008 → 11 years old technique.

$$\text{d-dim} \xrightarrow{\text{t-SNE}} \text{2d}$$

→ fundamental diff b/w t-SNE & PCA

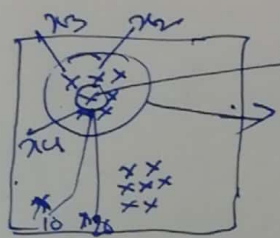


PCA : Try to preserve global shape/structure of data
doesn't care about local structure

TSNE : Preserve local structure

1.6.2 Neighborhood of a point, Embedding

d-dim : (high)



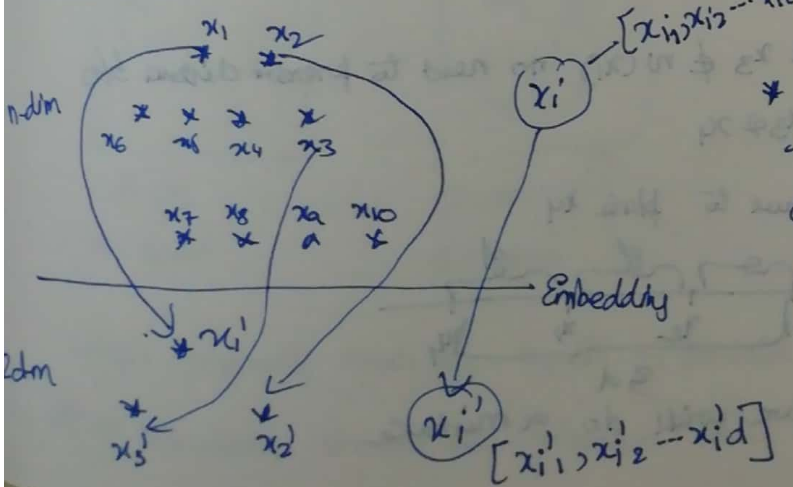
→ neighborhood of this points are points are close to it

$N(x_i) = \{x_j$, such that x_i & x_j are geometrically close$\}$

$\|x_i - x_j\|^2 = dist^2$

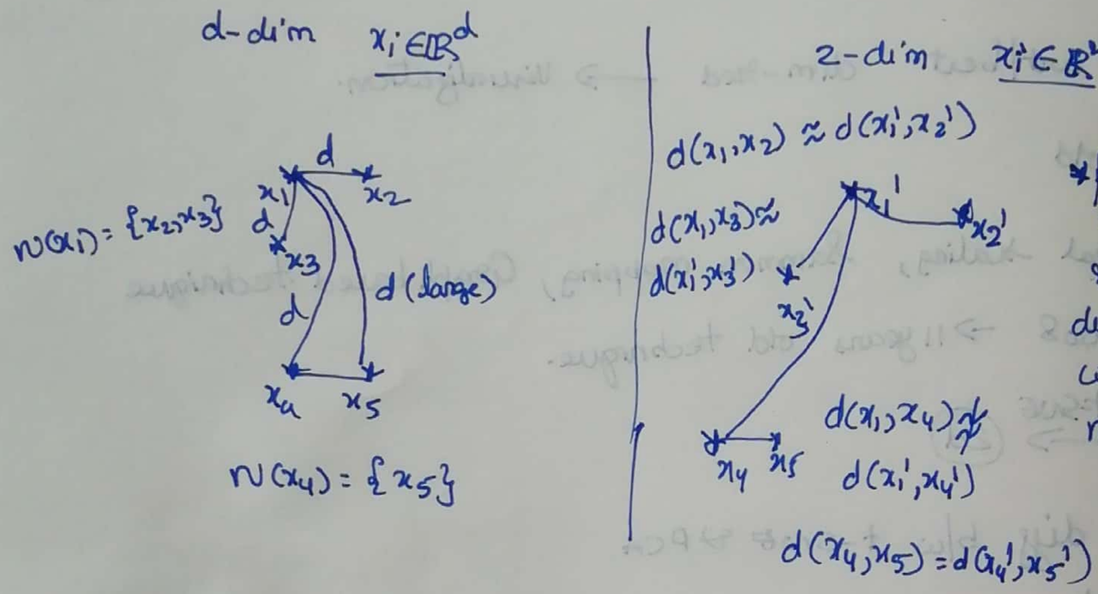
$N(x_1) = \{x_2, x_3, x_4\}$; does not contain x_{10} & x_{20}.

Embedding



* for Every point in high dimensional space finding a point in low dimensional space is called Embedding

15.3 Geometric intuition of t-SNE



* preserving the dist in neighborhood & not preserving the distance b/w the points which are outside neighborhood (x_4, x_5)

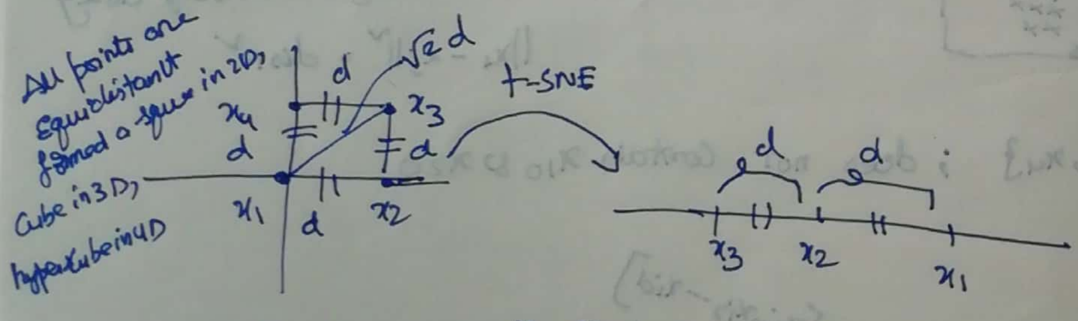
→ neighborhood preserving Embedding.

mathematical formulation

→ fairly advanced (2008)

15.4 Crowding Problem

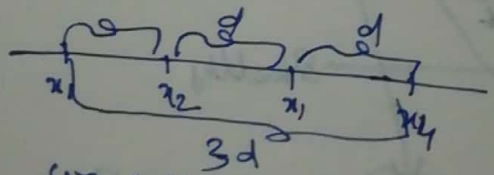
t-SNE :- Tries to preserve dist in a N



2D

* Since $x_3 \notin N(x_1)$ no need to preserve distance b/w x_3 & x_4

* we have to place x_4



* we even we have placed x_4 we will do a mistake

+ Sometimes; It is impossible to preserve dist in all the neighbourhoods (N)

This is called Crowding problem.

15.5 How to apply t-SNE and interpret its output

https://distill.pub/2016/misread-tsne/

d-dim $\longrightarrow$ d' dim

$d = d' = 2$

→ T-SNE is an iterative algorithm. and reach a point where a clusters are no-more moving

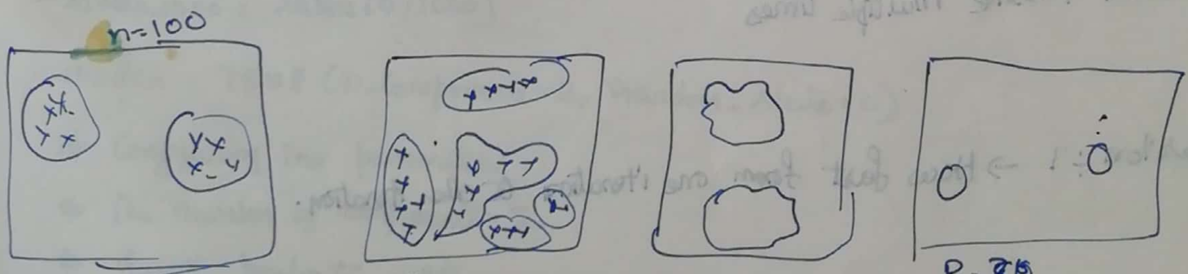
Two most imp parameters

1) Perplexity ② step-size (# of iterations) as high the better.

↓

# of neighbours distance that we are going to preserve.

If my perplexity 2, 5, 30, 50, 100

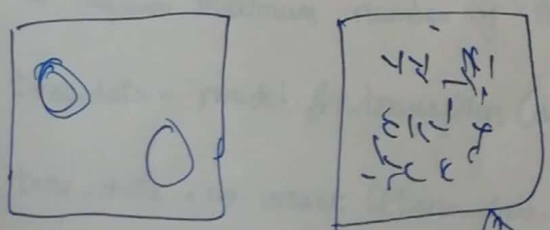


Original

Perplexity=2
step = 5000

P = 5
S = 5000

P = 30
S = 5000



P: 50
S = 5000

P = 100
S = 5,000

when perplexity matches the # of points This is a mess

# of iterations should be increased until the shape is stabilized

t-SNE
↳ Stochastic
↑
Probability

* Run t-SNE on the same dataset with same parameter we will be a slightly different results

Deterministic algo ÷ same results for any run
Stochastic algo ÷ slightly different result every time

* Expands dense clusters
* Shrinks sparse clusters

} Drawbacks

Steps

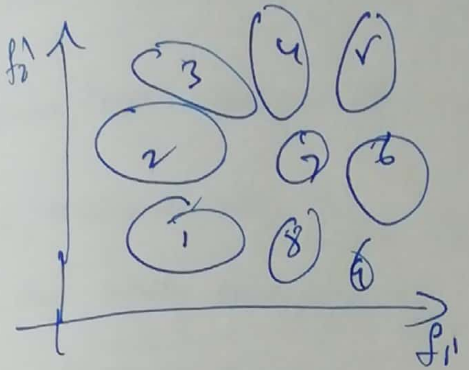
① Run step/iteration till shape stabilizes.
② Perplexity $\boxed{2 \le P \le n}$
③ re-run t-sne multiple times

* Epsilon ÷ 1 → How fast from one iteration to other iteration.

Scanned by CamScanner

15.6 t-SNE on MNIST   colab.github.io/posts/2014-10-~~Visualizing~~ Visualizing-MNIST/

784 dim → 2 dim



* Cannot interpret cluster sizes (or) inter cluster distances

15.7 Code Example of t-SNE

t-SNE using Scikit Learn

```
from sklearn.manifold import TSNE

# Picking the top 1000 points as TSNE takes a lot of time for 15K points

data_1000 = standardized_data[0:1000,:]
labels_1000 = labels[0:1000]

model = TSNE(n_components=2, random_state=0)
# Configuring the parameters
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(data_1000)

tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=('Dim1', 'Dim2', 'label'))

# Plotting the result of tsne
sn.FacetGrid(tsne_df, hue='label', size=6).map(plt.scatter, 'Dim1', 'Dim2')
plt.show()
```

Try

n_Components = 2, perplexity = 50, random-State = 0