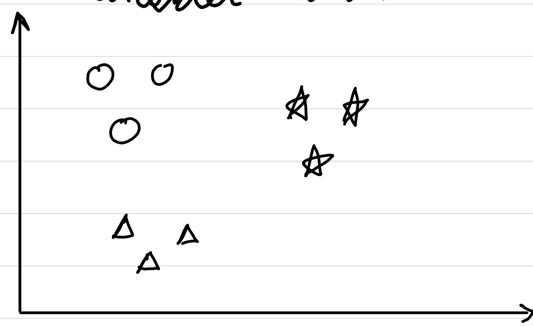
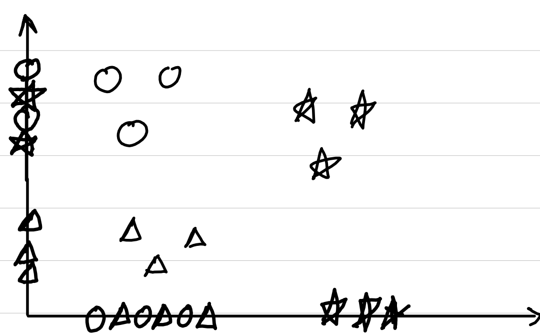
t-SNE

t-SNE preserves the original clustering of the data.

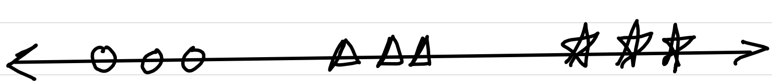
Consider data →



Projecting on either x axis or y axis will not give the correct clustering



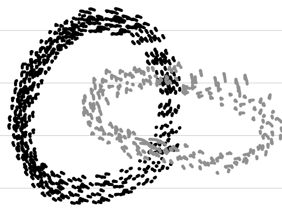
We want a projection that preserves the data like this



t- Stochastic Neighbourhood Embedding

PCA seperates data using a straight line
However not all data can be seperated so.
hence it fails to interpret complex datasets.

t-SNE performs nonlinear dimentionality reduction & overcomes this limitation



Linearly Non seperable data.

t-SNE is a form of Manifold learning

It is mainly used for data visualization in lower dimentions (XD to 2D)

It can also be used for dimentionality reduction

Useful for revealing clusters in data

Preserves local structure of data - points that are close together in higher dimentions remain close in the lower dimentions as well

How t-SNE operates

t-SNE works on minimizing the divergence between the two distributions - One in the higher dimentions and one in the lower dimentions.

It is a probabilistic model that uses gradient descent. This may give different results when run different times.

How t-SNE operates →

1. Measure Similarity between points in a Matrix
2. Map random to lower dimention & calculate similarity
3. Adjust the points to have same distribution of similarity as original high dimentional matrix
4. Visualize final point positions.

Unlike PCA, t-SNE is based on probability distributions with random walk on neighborhood graphs to find the structure within the data

t-SNE has a non-convex objective function that is minimized using gradient descent. It has random initiation which means that different runs give different solutions

t-SNE produces well-seperated clusters, providing clear insights into the grouping of similar datapoints.

Unlike LDA, t-SNE does not assume that the covariances are equal

Applications →

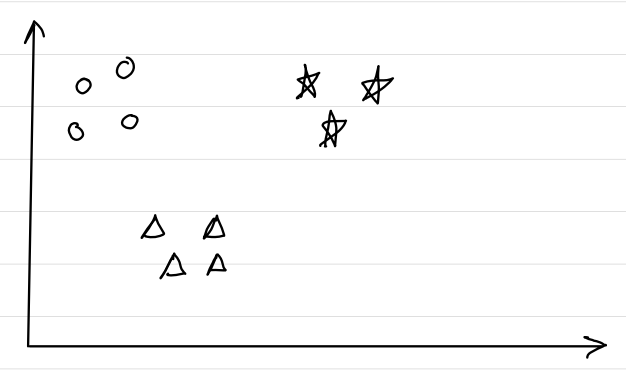
Data Exploration - Understand structure & relationship in high dimentional data

NLP → Visualize word embeddings.

Market segmentation, Social Network analysis, recommender systems.

Anamoly detection → By visualizing the data in low dimentional space, t-SNE can help identify clusters of datapoints that are different from rest of data. Useful for potential fraud, anamoly detection, outliers.

How t-SNE Works



There are 3 distinct classes that we recognize

First step is to find the similarity between points

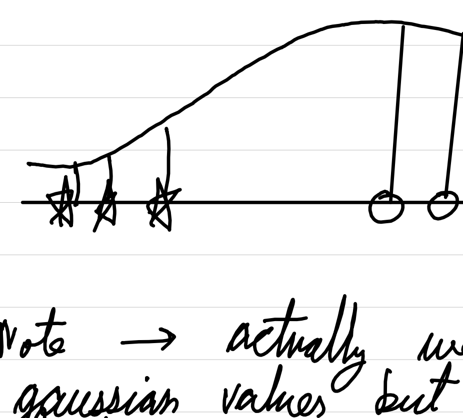
Similarity of a datapoint x_j to datapoint x_i is the datapoint $P_{j|i}$ that x_i would pick up x_j as its neighbour. in a random walk under a gaussian centered at x_i

Hence for nearby datapoints $P_{j|i}$ is relatively high, whereas for widely spaced points $P_{j|i}$ is low and reduces to infinitisimal

Let the variance of the gaussian centered around x_i be σ_i. σ_i will be calculated later.

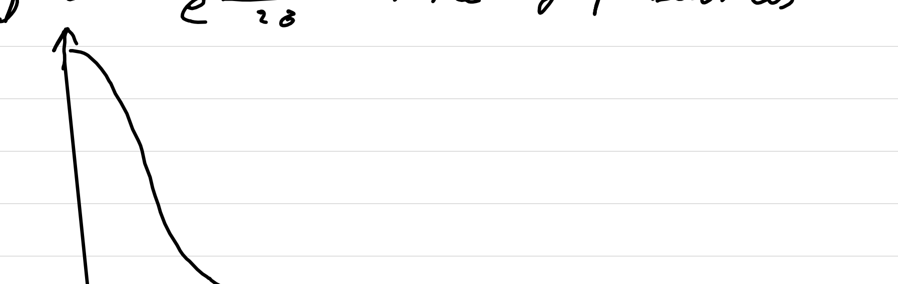
$$P_{j|i} = \frac{e^{\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} e^{\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}}}$$

$P_{j|i}$ is the similarity of point i & j measured from i

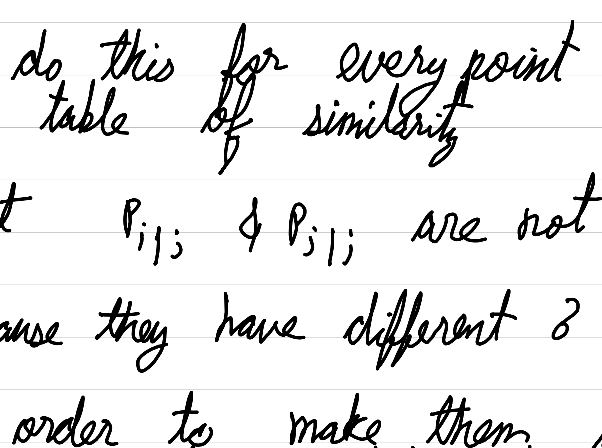


Similarly for every point j we calculate similarity

$\sum_j P_{ij} = 1$



Note $\rightarrow$ actually we are not taking the gaussian values but slightly different function $e^{\frac{-\|x_i - x_j\|^2}{2\sigma}}$ hence graph becomes



We do this for every point & make a table of similarity

But $P_{i|j}$ & $P_{j|i}$ are not same

because they have different σ values.

In order to make them same, average value is taken (symmetric SNE)

$$P'_{i|j} = P'_{j|i} = \frac{P_{i|j} + P_{j|i}}{N}$$ $\leftarrow$ No of dimentions

Our Similarity table $\rightarrow$

	x_1	x_2	x_3	...	x_k
x_1	0	0.3	0.3	...	
x_2	0.3	0	0.4	..	..
x_3	0.2	0.4	0	.	.
⋮	⋮		⋮	⋮	⋮
x_k	..	..		.	0

The aim is that when we reduce the dimentions, the same similarity table must be followed

The next step is to create a low dimentional space with the same number of points y_i

1. Randomly place the points
2. Calculate similarity
3. Move points with gradient descent to have same similarity as the original table

New similarity using gaussian

$$q_{j|i} = \frac{e^{-\|y_i - y_j\|^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|^2}}$$

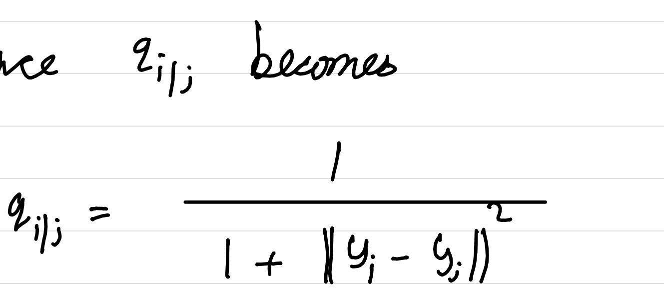
Here $\sigma = 1/\sqrt{2}$

This is the Method for SNE. But for t-SNE the method is different.

Using a gaussian leads to a "Crowding problem".

Clusters all clump up in the middle & are hard to see.

Hence the t distribution is used instead with a single degree of freedom



It is not at tall as gaussian in the middle & has tall tails.

Hence $q_{i|j}$ becomes

$$q_{i|j} = \frac{\frac{1}{1 + \|y_i - y_j\|^2}}{\sum_{k=l} \frac{1}{1 + \|y_i + y_j\|^2}}$$

The objective function is using the "kullback leiber divergence"

to check how similar P & q are

$$C = D_{KL}(P \| Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

$$= \sum_i \sum_j P_{j|i} \log\left(\frac{P_{j|i}}{q_{j|i}}\right)$$

Gradient descent

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

This can be interpreted as springs that attract or repel.

t-SNE gradient strongly repels dissimilar datapoints.
Similar datapoints are attracted

This way, after random shuffling, the rearrangement occurs

Calculation of σ_i

σ_i is found using binary search to match the parameter perplexity.

$$Perp = 2^{-\sum_j P_{j|i} \log_2 P_{j|i}}$$

Perplexity parameter

Perplexity is a knob that can be used to adjust the number of effective nearest neighbours.

The most appropriate value depends on the density of the data.

Larger / denser dataset requires a large perplexity

Typical values of perplexity range between 5 & 50

If the perplexity value is too high then all points become equidistant & a strange ball with uniformly distributed points occur

If the dataset contains very large numbers then the binary search may fail & t-SNE reports a very low but results look bad. Scaling must be performed.

t-SNE	PCA
Visualization of high dimentional dataset	Dimentionality reduction
Non Linear Manifold learning	Linear transformation
Probabilistic algorithm	Algebraic algorithm
Preserves local structure	Maximizes variance
Data Reduced to 2D or 3D Visualization	Lower dimentions, may be depending on % of loss
Less scalable for large datasets	More scalable to large datasets
Robust to Noise	Sensative to noise
Computationally Intensive	Better computation than t-SNE
Depends on random initialization	Same result everytime