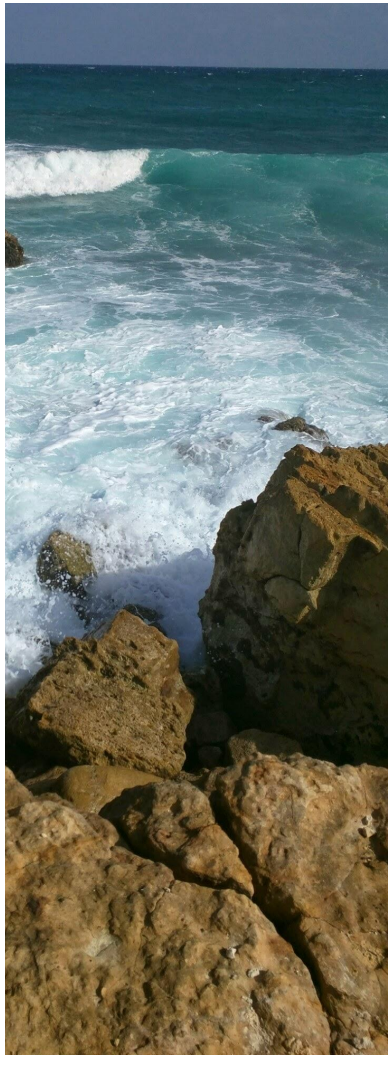


Embedding techniques and dimension reduction algorithms for data

2022

Outline:

Embedding principles
Embedding methods

linear methods:
PCA

non-linear methods
tSNE, diffusion maps
UMAP algorithms
Node2Vec

1. Big data visualisation (always plot your data), but what if your data is multidimensional?
Example of simple dimension reduction method: 3D data of Earth -> 2D data of coordinates

2. Idea on visualisation when we have data with N-dimensions, N>3:
Minimize sum of squares of distances on the line.
We can use two main ideas for embedding (or dimension reduction):
A. use linear methods from linear algebra: eigenvalues decomposition.
PCA idea
S = U A U ^-1
A - matrix from eigenvalues, U are eigenvectors, which are transforming the space.
KNN idea

B. use non-linear methods:
tSNE, stochastic embedding which optimise loss-functions

C. (Chakresh part)... Umap and other techniques

3. Hands-on: letting them to visualise their dataset or our dataset (for their choice)
Comparison of embeddings:
We compare embeddings using rainbow plot, for which we estimate the value of the loss function for the triplets series [Wang et al.]
For each triplet of nodes (i,j,k) the loss function is estimated as
$Loss(i,j,k) = f(d_{ij}, d_{ik})$
Comparing these rainbow plots for various embeddings we can find the most suited embedding for each dataset type.

Embedding types:

sklearn.decomposition.PCA - Principal component analysis that is a linear dimensionality reduction method

sklearn.decomposition.KernelPCA - Non-linear dimensionality reduction using kernels and PCA

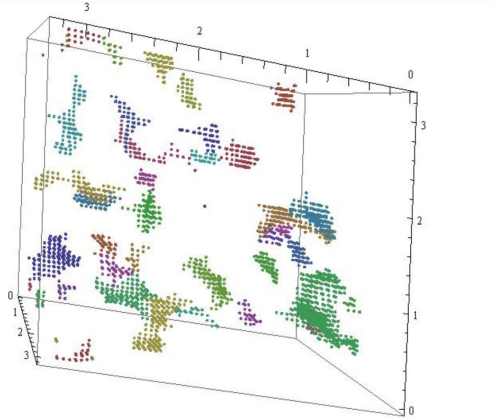
MDS Manifold learning using multidimensional scaling

Isomap Manifold learning based on Isometric Mapping

LocallyLinearEmbedding Manifold learning using Locally Linear Embedding

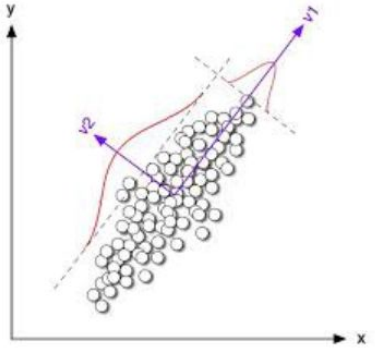
SpectralEmbedding Spectral embedding for non-linear dimensionality

Why embedding?



1. reduce the number of variables, but aren't able to identify variables to remove:
   names of dimensions: height, size, number of days etc.
2. ensure your variables are independent of one another
   not all variables are independent and how to interpret the independence...
3. visualise high-dimensional dataset, learn new hidden patterns

Simple embedding: PCA



Idea PCA: find the direction of v1 variable where the variance is maximum. So in order to do that, rotate the x variable axis on the plane. Here v1 has maximum variance and v2 have minimum variance so v1 has more information about the dataset(see notebook with MNIST)

MNIST dataset consist of 60k handwritten images of numbers from 0-9 and is commonly used for training various image processing systems. Data can be downloaded from here.

Simple embedding: KNN

Idea KNN: find k-nearest neighbors around each point
k is predefined constant, new data point is classified by assigning the label which is most frequent among the *k* training samples nearest to that new data point.

Issues with the method:

Simple embedding: KNN

Idea KNN:
find k-nearest neighbors around each point
k is predefined constant, new data point is classified by assi
among the *k* training samples nearest to that new data point.

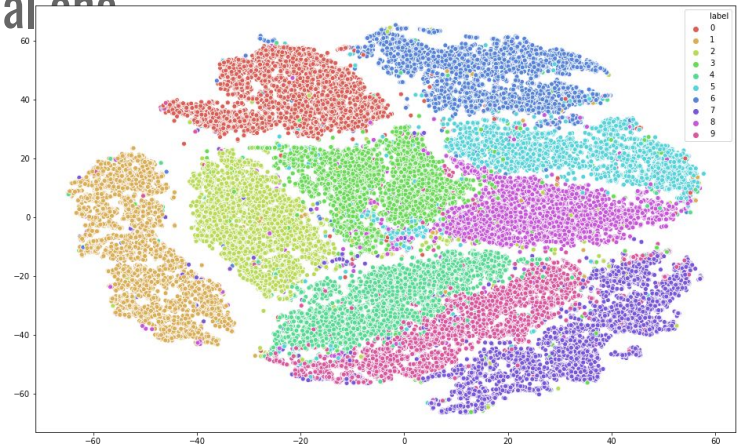
Issues with the method:
presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance, metric dependence

Simple classification for data: KNN algorithm

See notebook classroom

Issues with embeddings

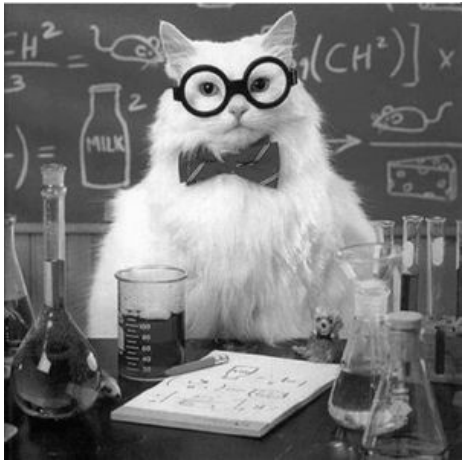
The curse of dimensionality:
issues when analyzing and organizing data in high-dimensional spaces that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience:
crowded problem.
Preserve the local structure of data while keeping global one

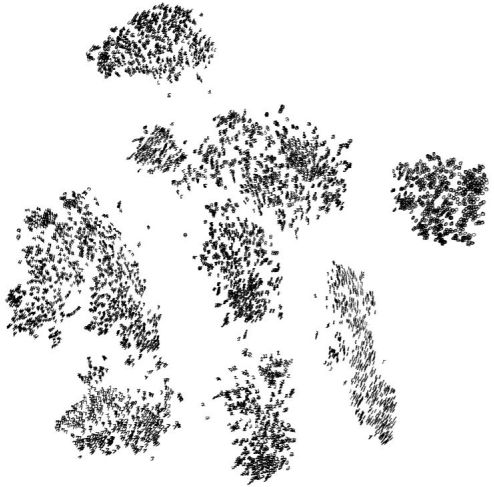


Non-linear embedding tSNE: stochastic embedding

The main idea of the stochastic embedding:
Measure **similarity** between points in high-dimensions.

Brilliant idea:
Cost function - minimisation of distances between distributions of points P(initial) and Q (projected) spaces.



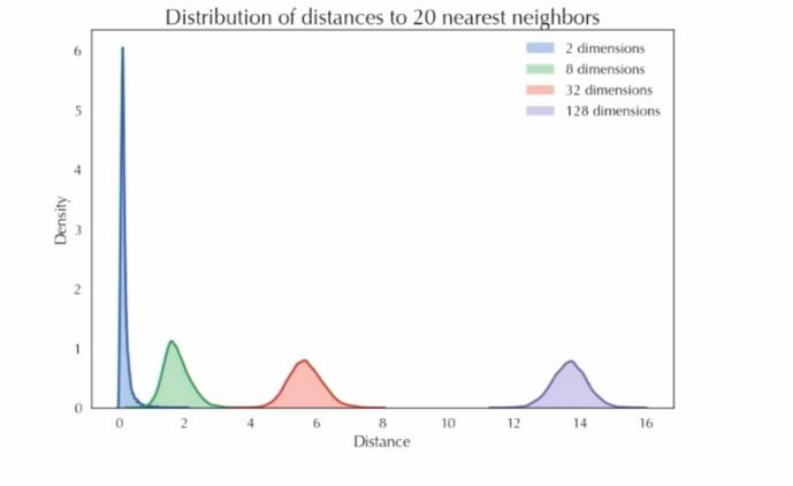


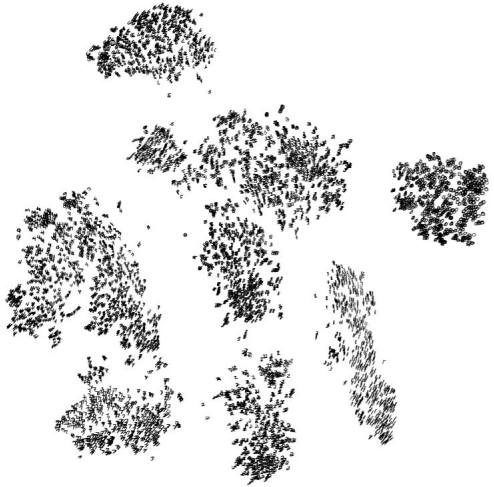
Non-linear embedding tSNE: stochastic embedding

The main idea of the stochastic embedding:
Measure **similarity** between points in high-dimensions.

Brilliant idea:
Cost function - minimisation of distances between distributions of points P(initial) and Q (projected) spaces.

BUT: cost of calculating distances distribution between all points is high...





Non-linear embedding tSNE: stochastic embedding

The main idea of the stochastic embedding:
Measure **similarity** between points in high-dimensions.

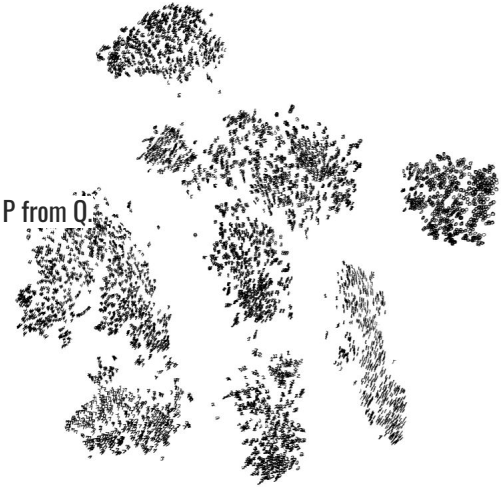
We pick up points to be close to each other **with some probability (!)** and not with certainty (as for PCA), which are close to each other and not the points,
we denote distance between them by by p_ij. Then we follow:

step 1, we compute the similarity between two data points using a conditional probability p. For example, the conditional probability of j given i represents that *x_j* would be picked by *x_i* as its neighbor assuming neighbors are picked in proportion to their probability density under a **Gaussian** distribution centered at *x_i* .

step 2, we let *y_i* and *y_j* to be the low dimensional counterparts of *x_i* and *x_j*.
Then we consider q to be a similar conditional probability for *y_j* being picked by *y_i*
We employ a **student t-distribution** in the low dimension map.

Cost function - minimisation of distances between distributions of points P(initial) and Q (projected)
The locations of the low dimensional data points are determined by minimizing the **Kullback–Leibler divergence** of probability distribution P from Q

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



Embedding: tSNE

Methods parameters:
Gaussian variance for estimating distribution between points
Perplexity level - hyper-parameter analogy with nearest neighbors - global parameter.
Number of simulations (iterations)

Initial implementation
https://lvdmaaten.github.io/tsne/

Paper http://www.jmlr.org/papers/volume9/vandermaaten08a/vand

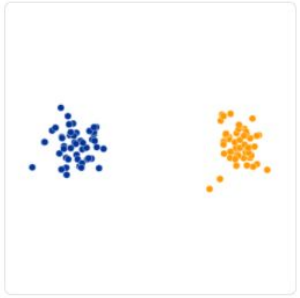
How to use tSNE effectively?

Parameters choice

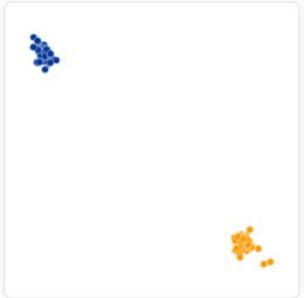
https://distill.pub/2016/misread-tsne/

How to use tSNE effectively?

Parameters choice

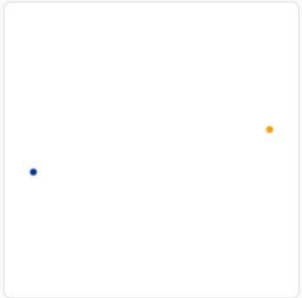


Original

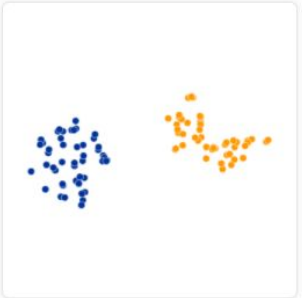


Perplexity: 30
Step: 10

Perplexity: 30
Step: 20

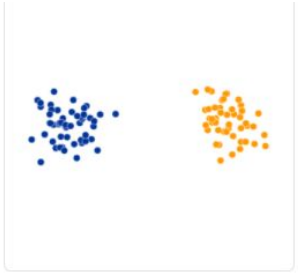


Perplexity: 30
Step: 60

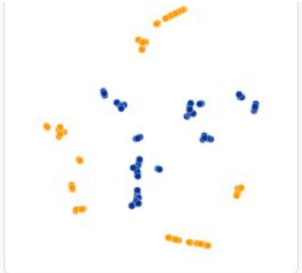


Perplexity: 30
Step: 120

Perplexity: 30
Step: 1,000



Original



Perplexity: 2
Step: 5,000

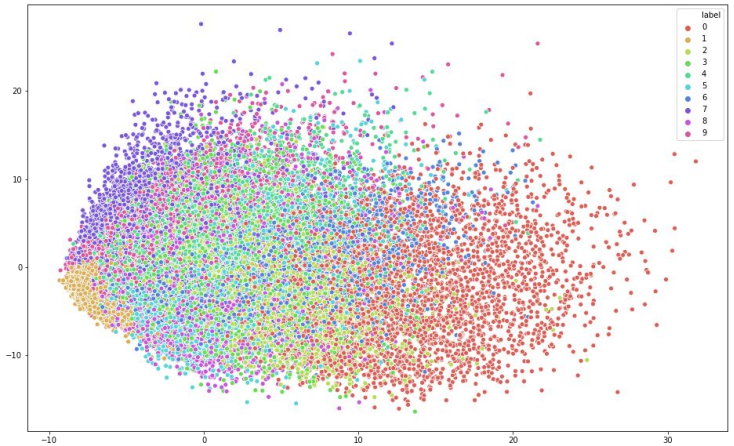
Perplexity: 5
Step: 5,000

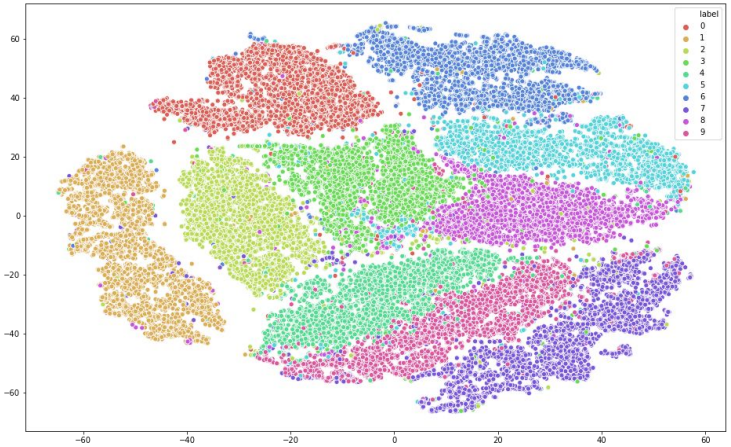
Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

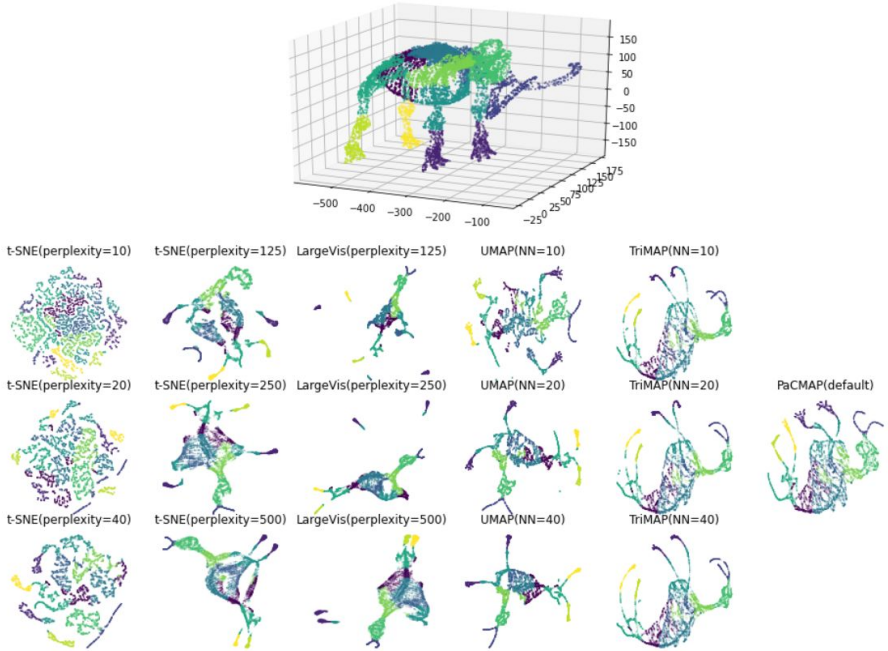
Embeddings comparison
How to compare?

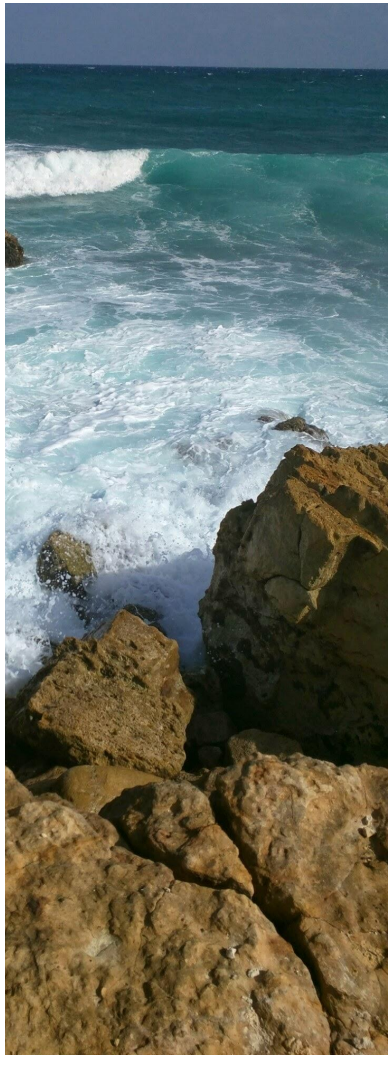




Embeddings comparison:

Choose the dataset we all know well





tSNE algorithm theory

Definition of tSNE - stochastic embedding

The main idea of the stochastic embedding:
we pick up points to be close to each other **with some probability (!)** and not with certainty (as for PCA), which are close to each other and not the points: this is set by p_ij

1. Consider two points i and j. Define the conditional probability $p_{j|i} = \frac{e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2}}$, where σ_i is computed by applying a binary search to solve the equation:

$$\text{Perplexity} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}.$$

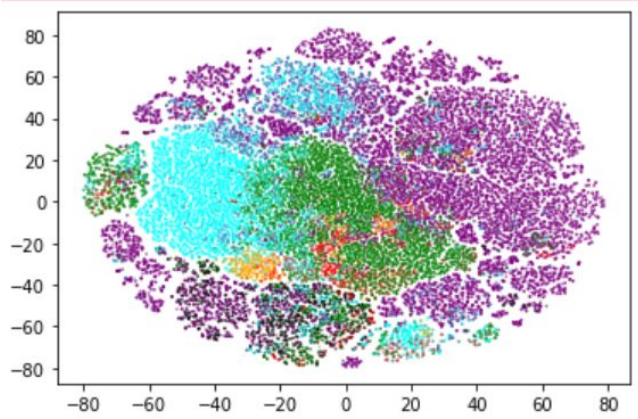
   The user-defined parameter Perplexity is monotonically increasing in σ_i. Intuitively, as Perplexity becomes larger, the probabilities become uniform. When Perplexity becomes smaller, the probabilities concentrate heavily on the nearest points.

2. Define the symmetric probability $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$. Observe that this probability does not depend on the decision variables $\mathbf{Y}$ and is derived from $\mathbf{X}$.

3. Define the probability $q_{ij} = \frac{(1+\|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1+\|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$. Observe that this probability is a function of the decision variables $\mathbf{Y}$.

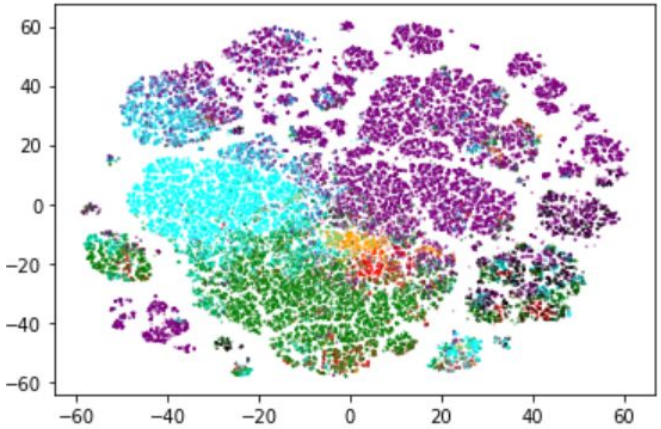
4. Define the loss function as $\sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$.

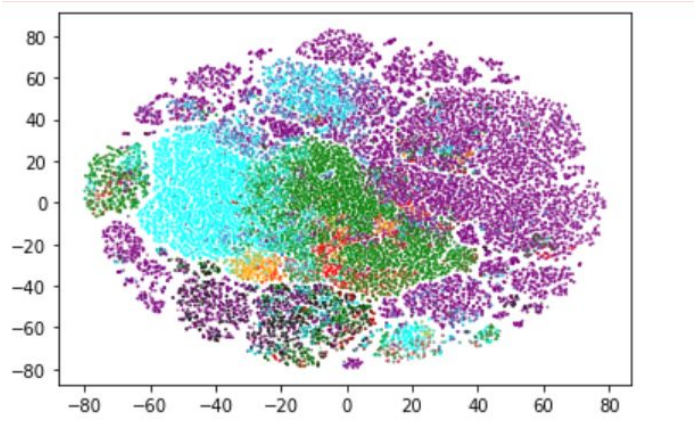
5. Initialization: initialize $\mathbf{Y}$ using the multivariate Normal distribution $\mathcal{N}(0, 10^{-4}I)$, with I denoting the two-dimensional identity matrix.

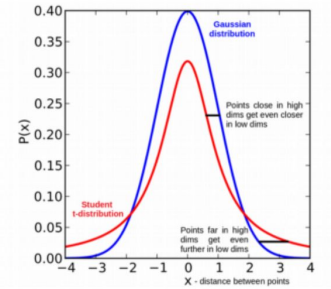


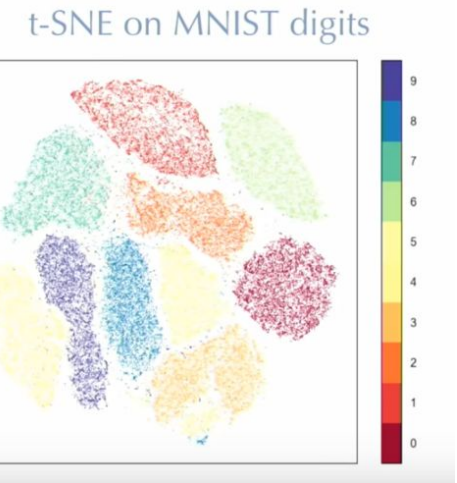
Summary on tSNE from https://distill.pub/2016/misread-tsne/

1. First one needs to define tSNE performs a binary search for the value of σi that produces a Pi with a fixed perplexity that is specified by the user. The perplexity is defined as 2^(H(P)),
as given in Maaten, L.v.d. and Hinton, G., 2008. Journal of Machine Learning Research, Vol 9(Nov), pp. 2579—2605.
2. Since it is known that there may not be one perplexity value for a given dataset,
that will capture distances across all clusters—and sadly perplexity is a global parameter.
Another problem with tsne is that random noise doesn't always look random.
3. Sometimes you can read topological information off a t-SNE plot, but that typically requires views at multiple perplexities.
4. Every time we run tSNE we get different results. We can select one the visualization from many different ones with the lowest value of the objective function as final visualization.









tSNE algorithm: some properties (different from SNE)

About projection to two-dimension space (especially in tSNE, and you probably know it as "crowding problem"):

Consider a set of datapoints that lie on a 2-dimensional curved manifold which is approximately linear on a small scale,

and which is embedded within a higher-dimensional space.

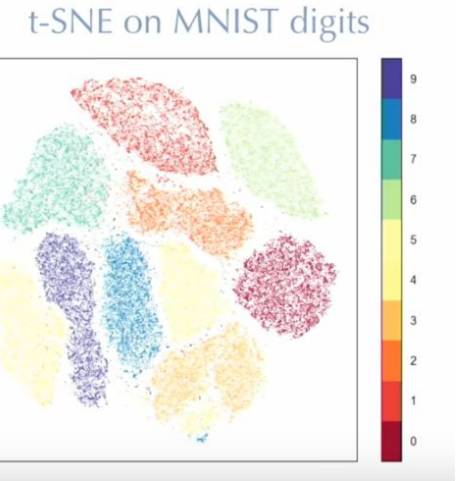
There are obvious problems to model the pairwise distances between datapoints well in a two-dimensional map:

we can take e.g. (10+x) points from 10 dimensional space equally distant and this won't be represented well in 2D (unless bijection is not fulfilled).

SO if you have many problems in your initial 100-dimensional space, which are equal-distant and you want to make projection to 2D, then there is

a "solution" for such problem: points, which are at a moderate distance from datapoint i will have to be placed much too far away in the two-dimensional map.

tSNE algorithm: some properties (different from SNE)



About search of suitable dimension of projection:

since in tSNE non-parametric dimensionality reduction techniques are used, it is not easy to say

what is the optimal dimension could be.

One way is to use Lagrangian multipliers technique and to add dimension as additional parameter maybe?

But the trick here is to look at the distribution of distances in your initial data, how is looks like?

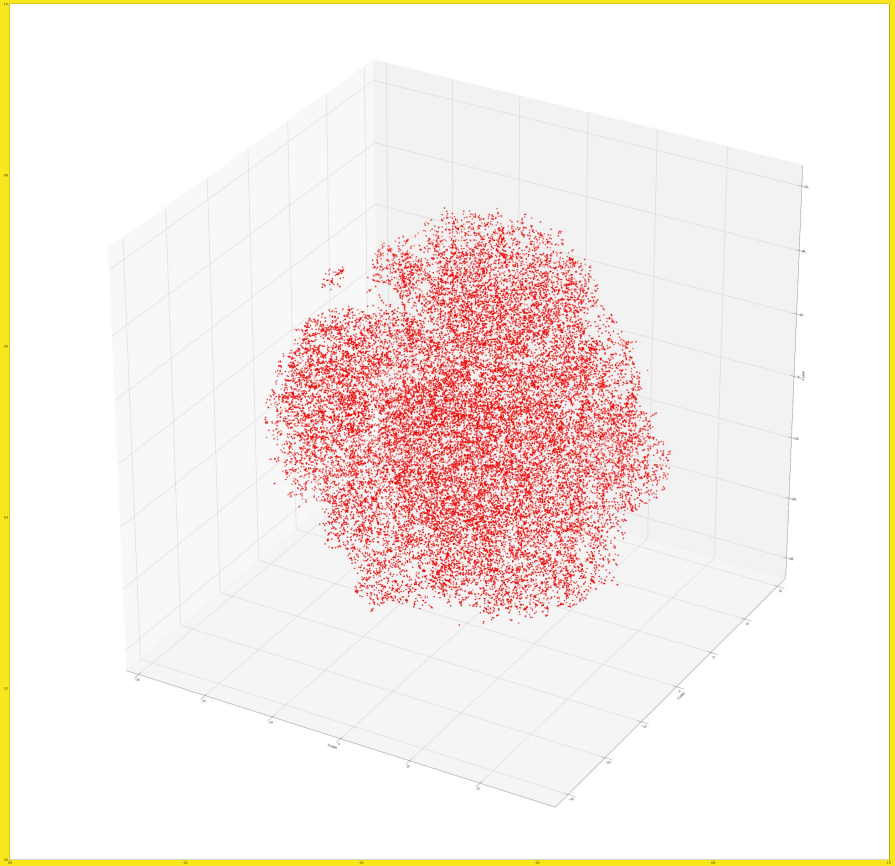
Because depending on this distribution different techniques can be used, such as

p.2587 from tSNE paper

p. from Diffusion maps paper

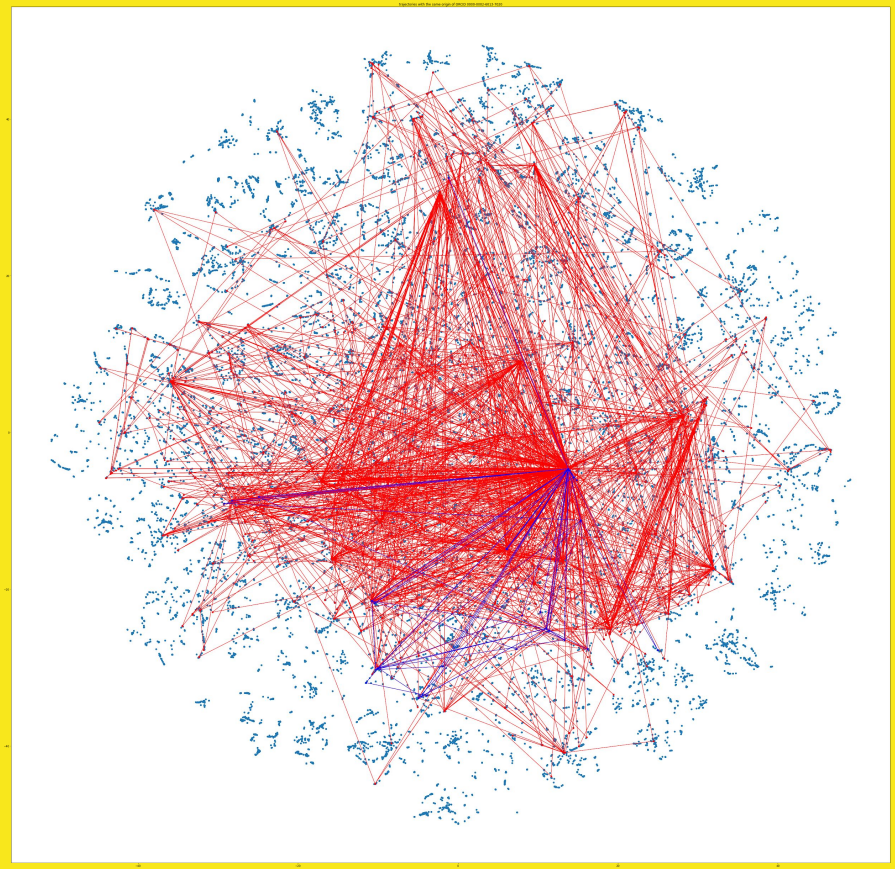
Some results

Applications on arxiv data
With TSNE algorithm



Some results

Applications on arxiv data
With TSNE algorithm



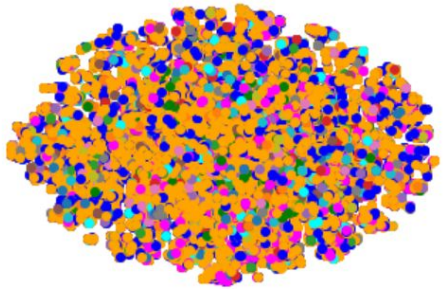
Simple embedding: UMAP

(see notebook with MNIST)

Paper http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

https://github.com/deveshSingh06/t-SNE/blob/master/t-SNE%20Implementation.ipynb

UMAP algorithm (UMAP Uniform Manifold Approximation and Projection for Dimension Reduction)

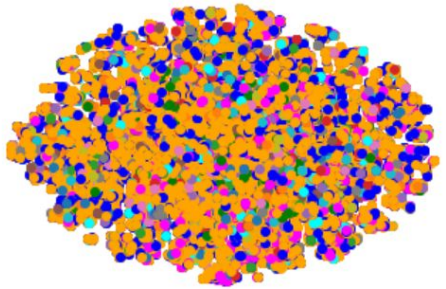


1. Topological data analysis

2. Main idea: almost any manifold can be reconstructed by simplicial complex triangulation.
Triangulation can be represented as a network.

Theorem 1 (Nerve theorem). *Let $\mathcal{U} = \{U_i\}_{i \in I}$ be a cover of a topological space X. If, for all $\sigma \subset I$ $\bigcap_{i \in \sigma} U_i$ is either contractible or empty, then $\mathcal{N}(\mathcal{U})$ is homtopically equivalent to X.*

UMAP algorithm: mathematical properties



In UMAP the attractive force between two vertices i and j at coordinates $\mathbf{y_i}$ and $\mathbf{y_j}$ respectively, is determined by:

$$\frac{-2ab\|\mathbf{y_i} - \mathbf{y_j}\|_2^{2(b-1)}}{1 + \|\mathbf{y_i} - \mathbf{y_j}\|_2^2} w((x_i, x_j))\,(\mathbf{y_i} - \mathbf{y_j})$$

where a and b are hyper-parameters.

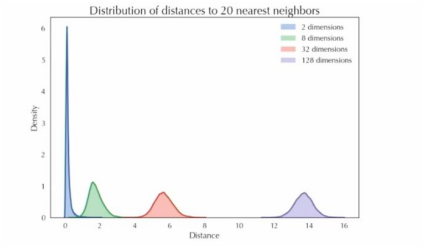
Definition 10. *The cross entropy C of two fuzzy sets (A, μ) and (A, ν) is defined as*

$$C((A,\mu),(A,\nu)) \triangleq \sum_{a \in A} \left(\mu(a) \log\left(\frac{\mu(a)}{\nu(a)}\right) + (1-\mu(a)) \log\left(\frac{1-\mu(a)}{1-\nu(a)}\right)\right).$$

UMAP algorithm

4. The problem is that data are NON **homogeneously** distributed along the manifold

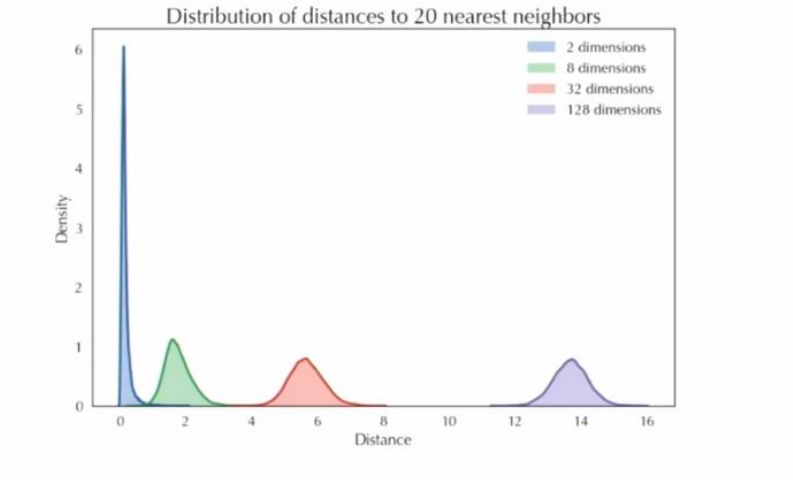
5. As with other k-neighbour graph based algorithms, **UMAP can be described in two phases.**
A. In the first phase a particular weighted k-neighbour graph is constructed.
B. In the second phase a low dimensional layout of this graph is computed.



UMAP algorithm: some properties

UMAP embedding is better than tSNE, since it keeps the form of the distribution but just changes the average of distribution.

Look at distance position (20th nearest neighbours).



UMAP algorithm hyperparameters

1. the number of neighbors to consider when approximating the local metric;
2. d, the target embedding dimension;
3. min-dist, the desired separation between close points in the embedding space; and
4. n-epoches

[“UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction” Mc.Innes et al.]
https://github.com/lmcinnes/umap