Adv. ML
Live Session 4
Manifold Learning
Find meaningful low-dimensional structures hidden in high-dimensional Data

Curse of Dimensionality:

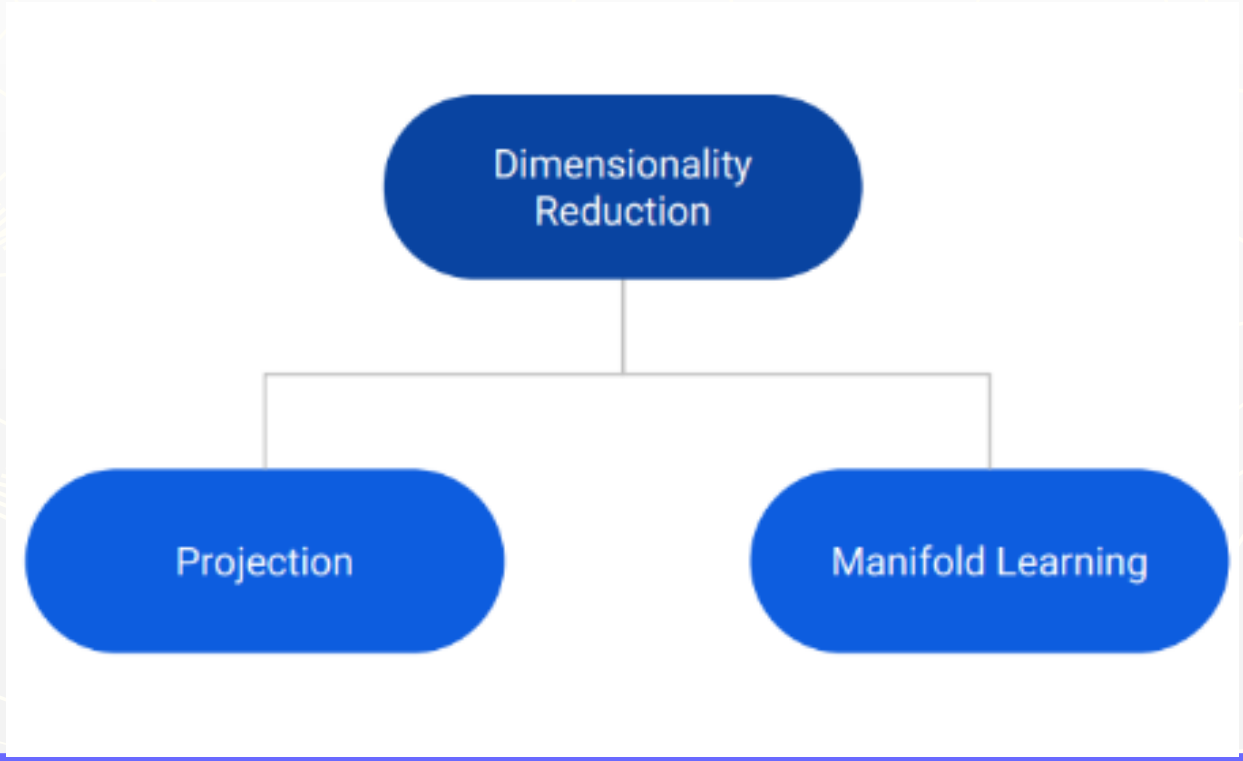
- Fortunately, in real-world problems, it is often possible to reduce the number of features considerably, turning an intractable problem into a tractable one.

- Apart from speeding up training, dimensionality reduction is also extremely useful for data visualization.

- Reducing the number of dimensions down to two (or three) makes it possible to plot a high-dimensional training set on a graph and often gain some important insights by visually detecting patterns, such as clusters.

Dimensionality Reduction:

Dimensionality reduction aims to map the data from original dimension (high dimension) to lower dimension space while minimizing information loss



PCA:

- **Two approaches to perform dim. reduction $\Re^N \rightarrow \Re^M$ (M<N)**
  - **Feature selection**: choosing a subset of all the features

$$[x_1\ x_2 \ldots x_N] \xrightarrow{\text{feature selection}} [x_{i_1}\ x_{i_2} \ldots x_{i_M}]$$

  - **Feature extraction**: creating new features by combining existing ones

$$[x_1\ x_2 \ldots x_N] \xrightarrow{\text{feature extraction}} [y_1\ y_2 \ldots y_M] = f([x_{i_1}\ x_{i_2} \ldots x_{i_M}])$$

    - In either case, the goal is to find a low-dimensional representation of the data that preserves (most of) the information or structure in the data

- **Linear feature extraction**
  - The "optimal" mapping y=f(x) is, in general, a non-linear function whose form is problem-dependent
    - Hence, feature extraction is commonly limited to linear projections **y=Wx**

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{\text{linear feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \\ y_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & & w_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Issue with PCA:

PCA cannot represent a **complex (nonlinear) relationship** between features.

PCA is only focused to place dissimilar data points far apart in a lower dimension representation.

Manifold Learning:

Manifold learning is an approach to non-linear dimensionality reduction.

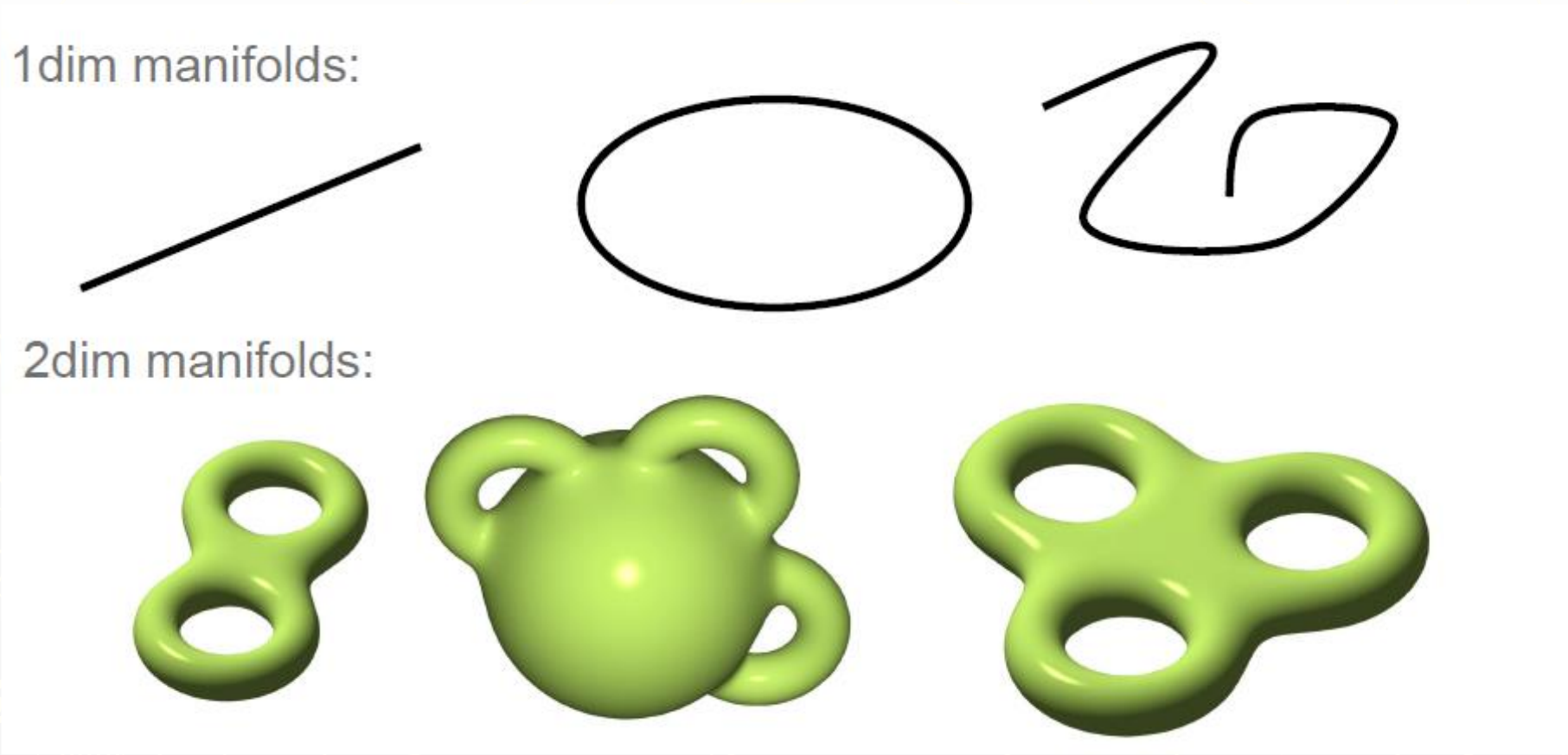
Algorithms for this task are based on the idea that the dimensionality of many data sets is only artificially high.

It uncovers hidden patterns or relationships in data.

It helps to represent the data in an accurate and meaningful way?

What is a Manifold (Laymen):

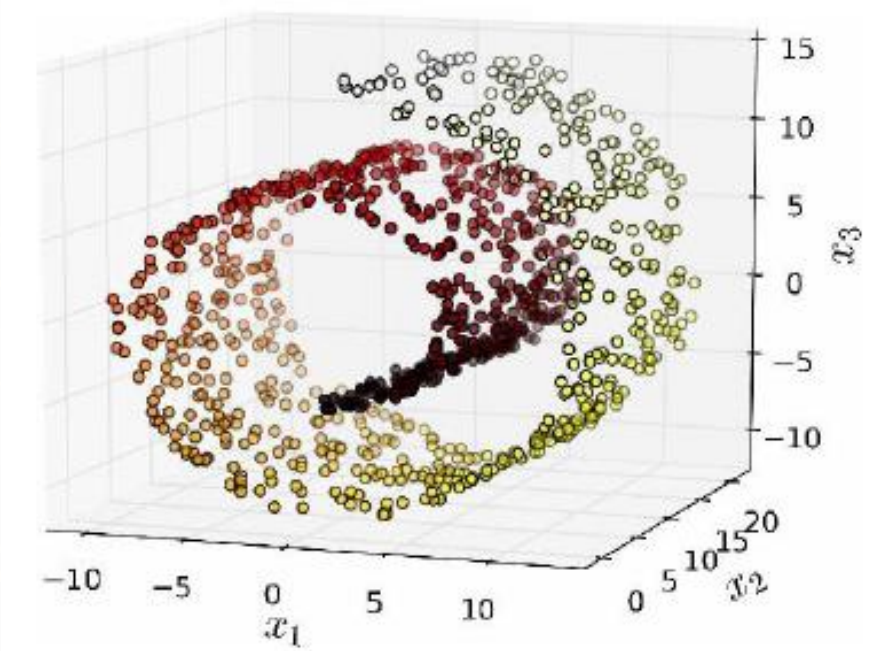
Manifold = any object which is nearly "flat" on small scales.

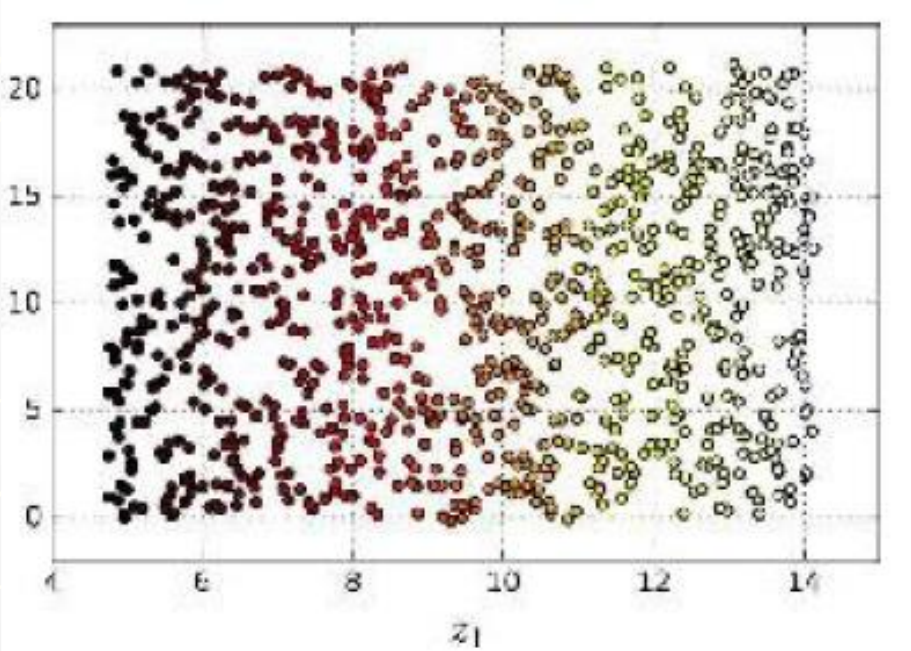


What is a Manifold: Example

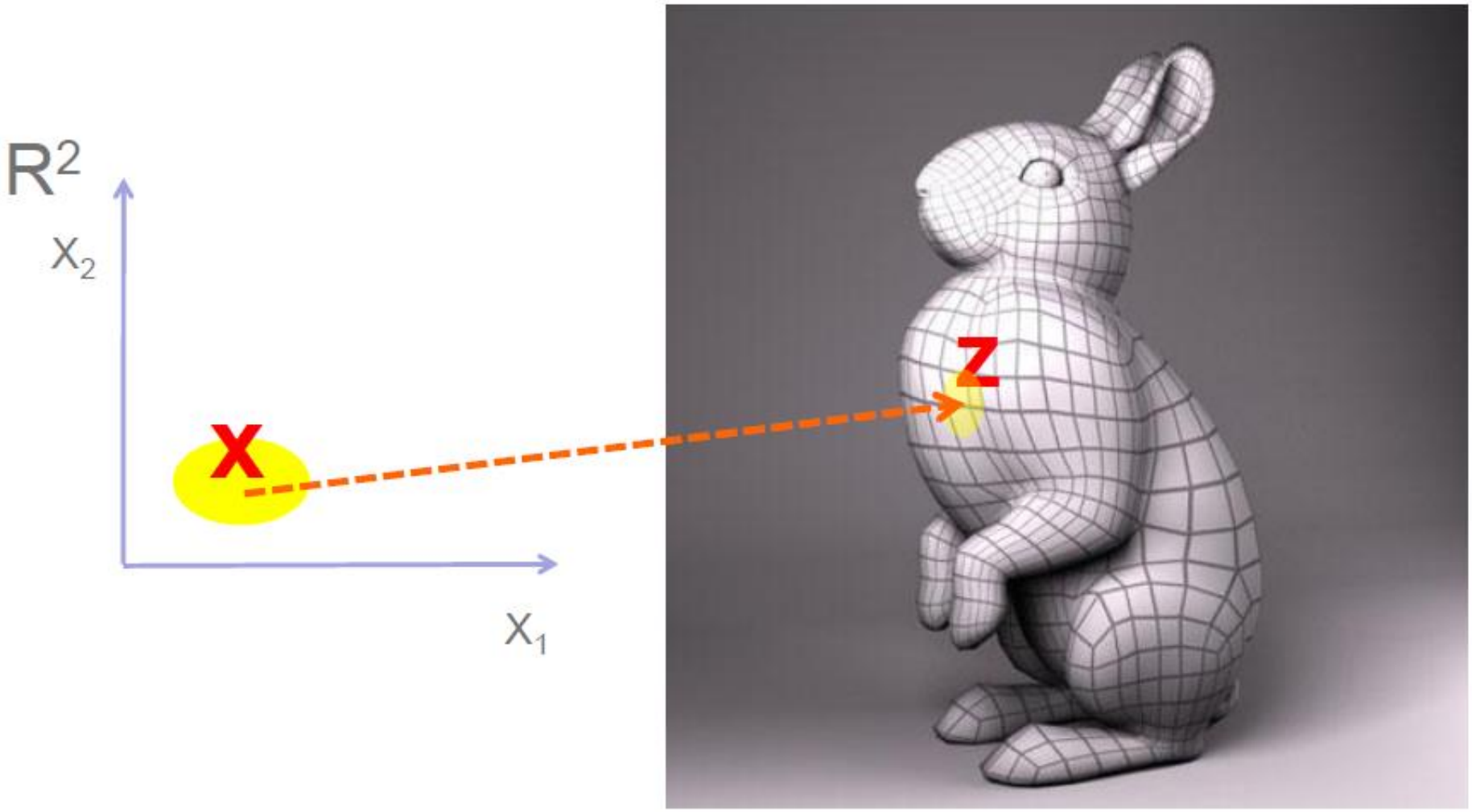
More generally, a *d*-dimensional manifold is a part of an *n*-dimensional space (where $d < n$) that locally resembles a *d*-dimensional hyperplane.

Example: Swiss roll, $d = 2$ and $n = 3$ (it locally resembles a 2D plane, but it is rolled in the third dimension)





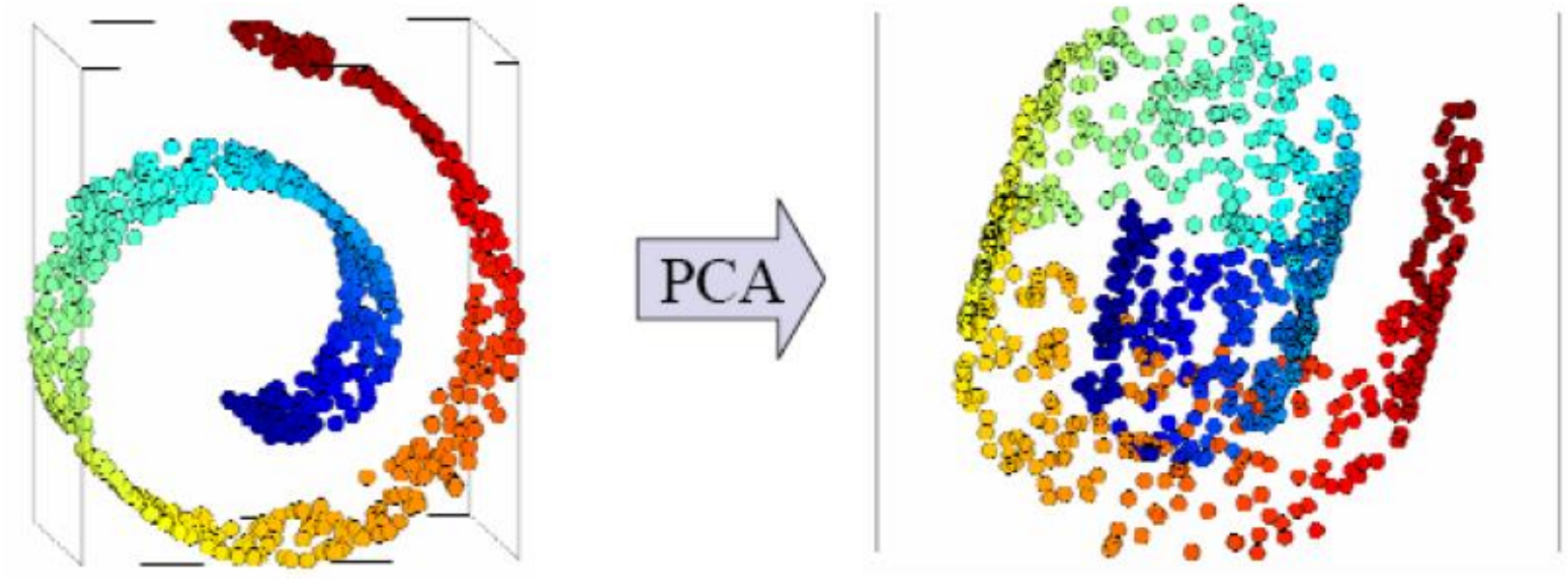
Manifold Learning



Why not PCA?

PCA is a linear method:

it fails to find the nonlinear structure in the data



Multi-Dimensional Scaling:

Multidimensional scaling (MDS) is a method used in data sciences to visualize and compare similarities & dissimilarities of high dimensional data.

PCA minimizes dimensions, preserving covariance of data. MDS minimizes dimensions, preserving distance between data points.

They are same, if covariance in data = Euclidean distance between data points in high dimension.

Multi-Dimensional Scaling: An Example

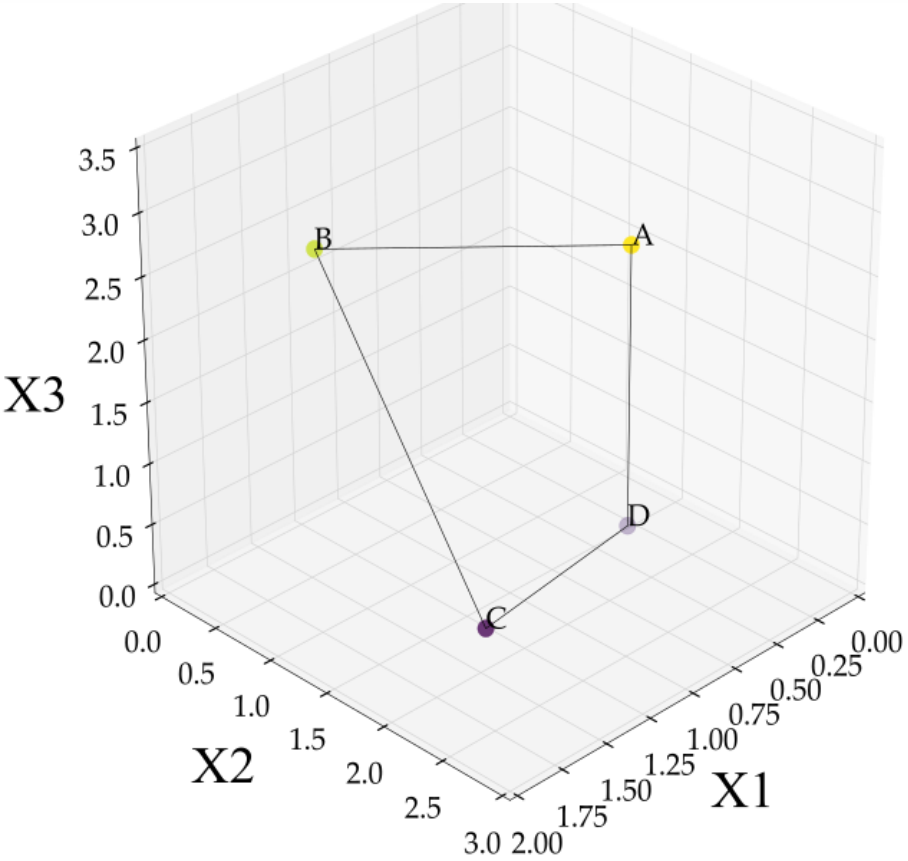
An optimization algorithm is used to embed the entities in a lower dimension space with pairwise distances as close as possible to the input distance matrix.

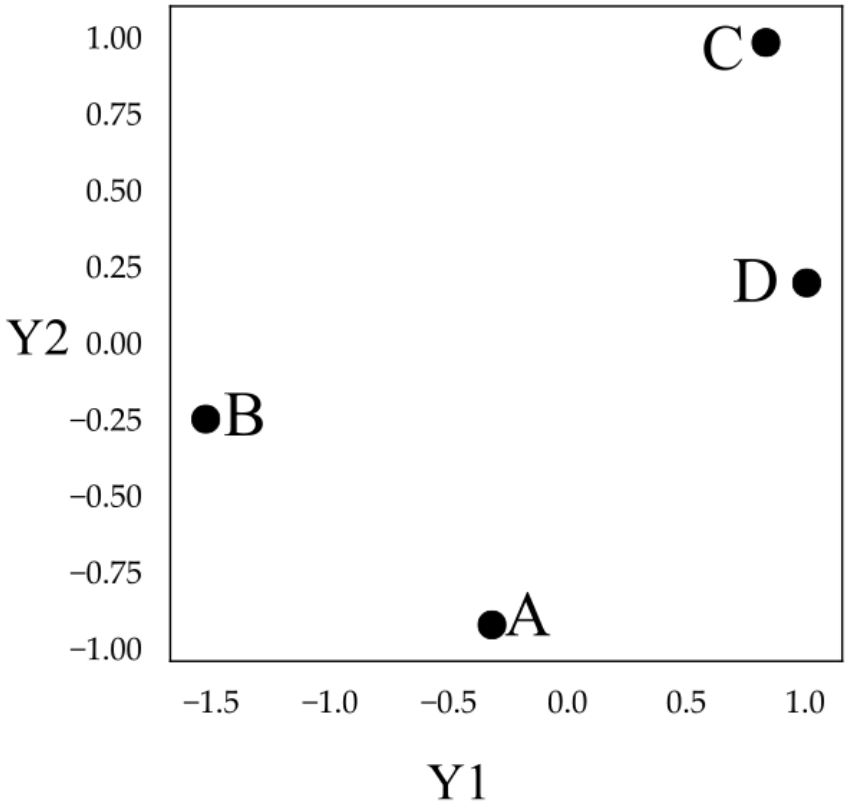
The distances between entities in the lower dimensions are not preserved exactly. MDS finds the best-fit configuration.

The distortion in distances between the lower dimension and higher order space is called stress, which is minimized by the MDS implementation.

Mancell and Deutsch

Multi-Dimensional Scaling: An Example





Mancell and Deutsch

Multi-Dimensional Scaling: An Example

Euclidean Distance in Original Space (3-dimensions)				
A	B	C	D	Entity
	1.69	2.53	2.20	A
		2.66	2.61	B
			0.82	C
				D

Euclidean Distance in Lower Dimension (2-dimensions)				
A	B	C	D	Entity
	1.71	2.53	2.20	A
		2.67	2.59	B
			0.82	C
				D

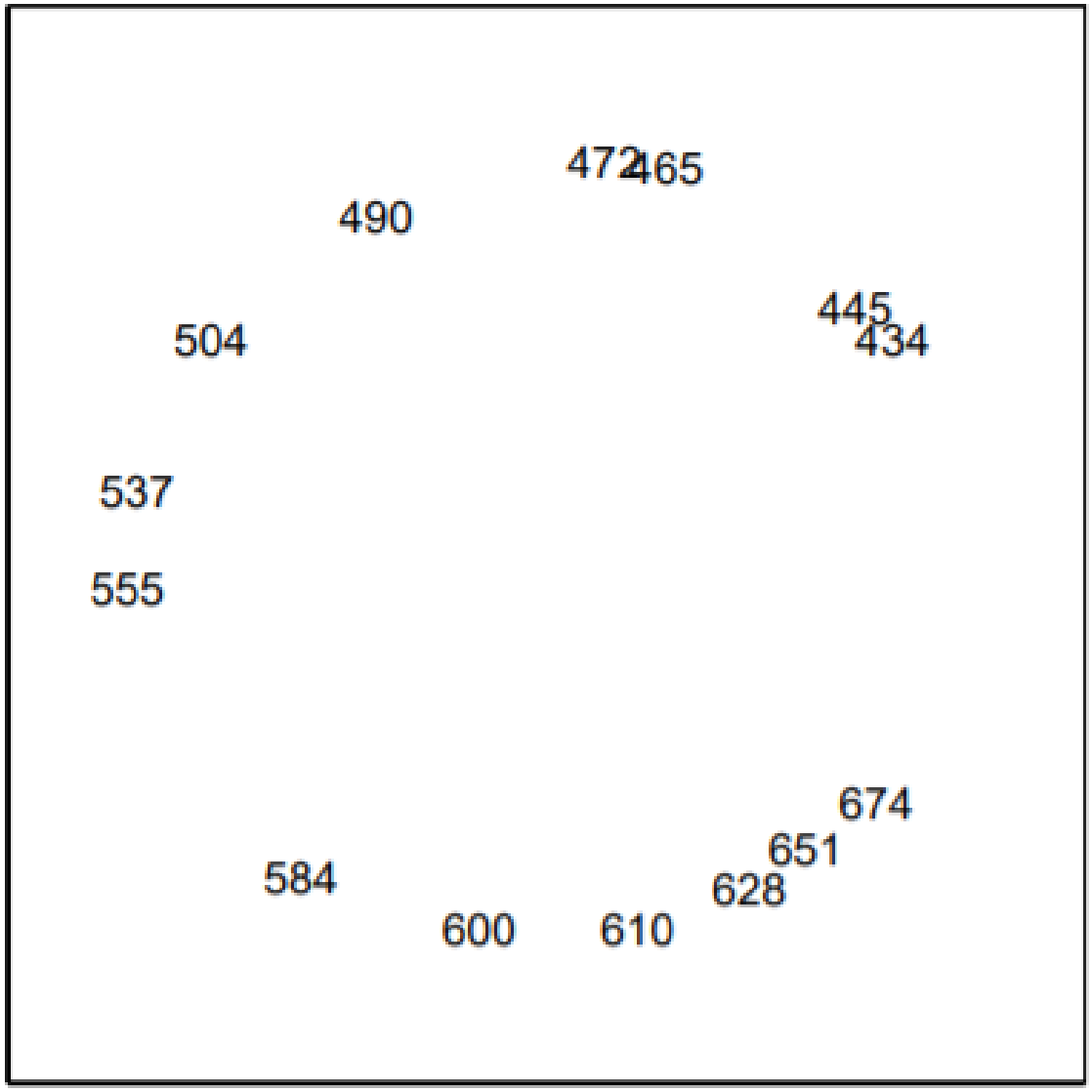
Mancell and Deutsch

Multi-Dimensional Scaling: Example-II

nm	434	445	465	472	490	504	537	555	584	600	610	628	651	674
434	–													
445	.14	–												
465	.58	.50	–											
472	.58	.56	.19	–										
490	.82	.78	.53	.46	–									
504	.94	.91	.83	.75	.39	–								
537	.93	.93	.90	.90	.69	.38	–							
555	.96	.93	.92	.91	.74	.55	.27	–						
584	.98	.98	.98	.98	.93	.86	.78	.67	–					
600	.93	.96	.99	.99	.98	.92	.86	.81	.42	–				
610	.91	.93	.98	1.00	.98	.98	.95	.96	.63	.26	–			
628	.88	.89	.99	.99	.99	.98	.98	.97	.73	.50	.24	–		
651	.87	.87	.95	.98	.98	.98	.98	.98	.80	.59	.38	.15	–	
674	.84	.86	.97	.96	1.00	.99	1.00	.98	.77	.72	.45	.32	.24	–

Source: Ekman

Multi-Dimensional Scaling: Example-II



MDS:

Class

sklearn.manifold.**MDS**(n_componenets=2, n_init=4, max_iter=100, eps=.001, dissimilarity= 'Euclidean')

N_components	Number of dimensions in which to immerse the disimmilariy
N_init	Number of times the optimization algorithm will run with different initializations. The final output will be the best result (smallest stress)
eps	Convergence threshold for the stress
dissimilarity	Dissimilarity measure to use. Other option is precomputed..

Isomap:

In non-linear manifolds, the Euclidean metric for distance holds good if and only if neighborhood structure can be approximated as linear. If neighborhood contains holes, then Euclidean distances can be highly misleading.

In isometric mapping (Isomap), we try to preserve the geodesic distances in the lower dimension.

Isomap starts by calculating an approximate geodesic distance between all pairs of points.

Journey so far:

Covariance-→ Euclidean ---→ Geodesic

Isomap:

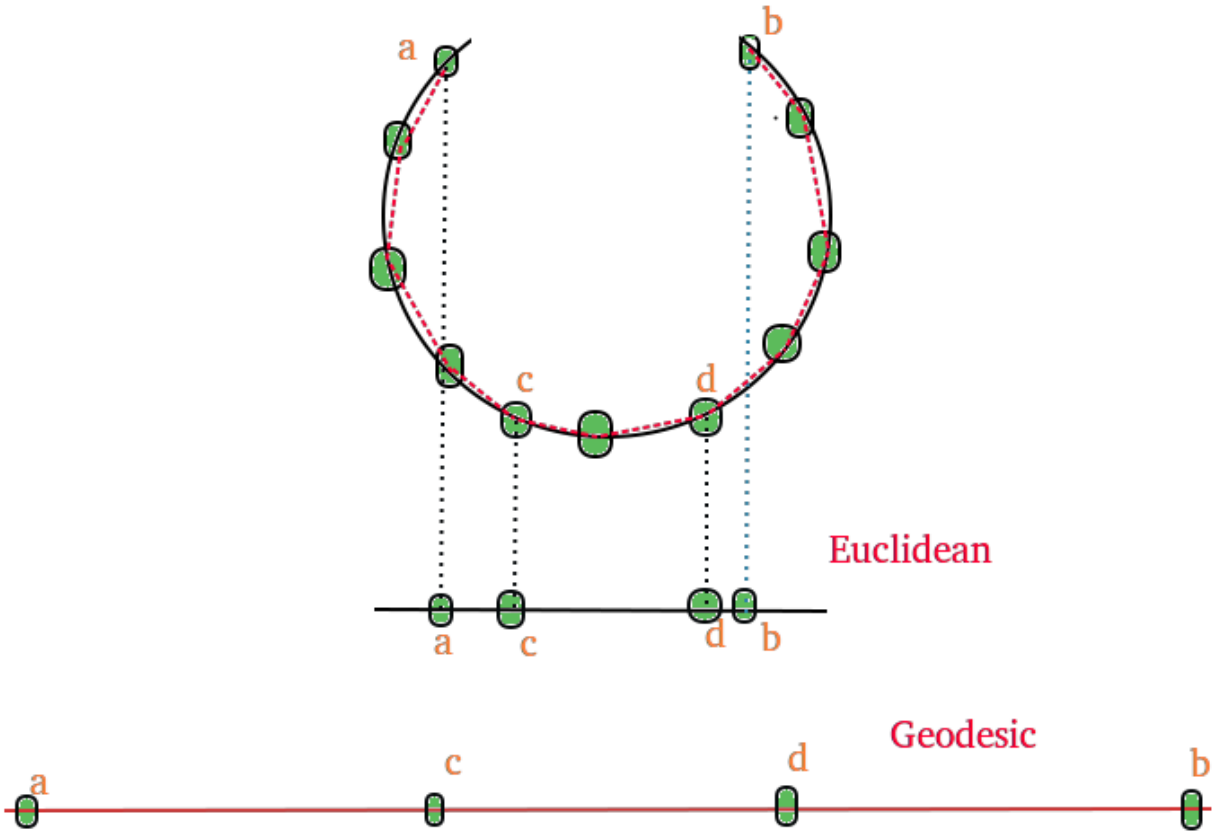
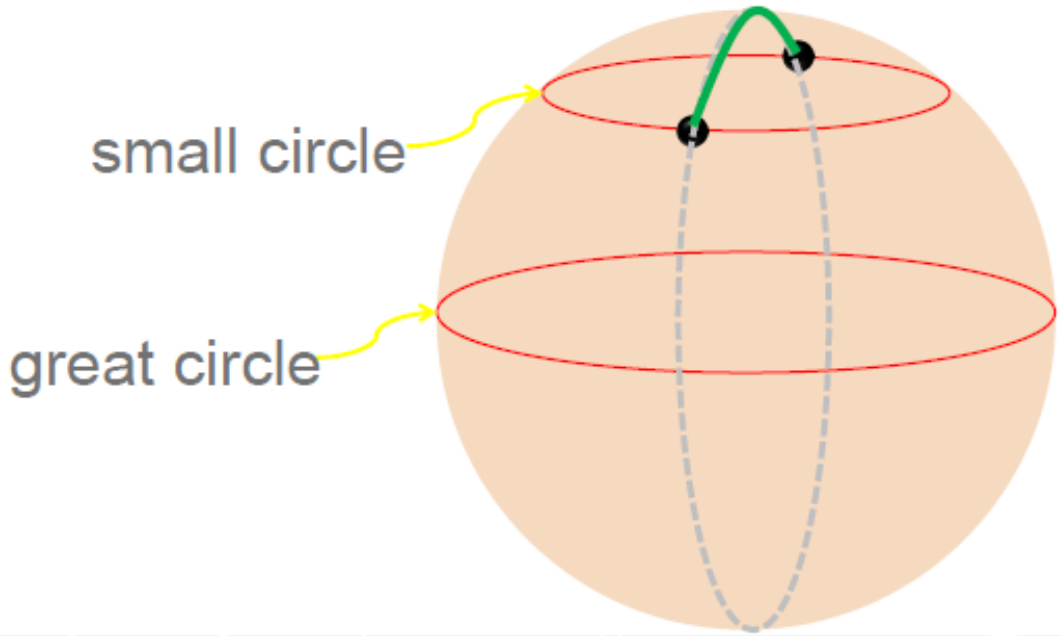


Image Source: Paperspaceblog

Geodesic Distance:

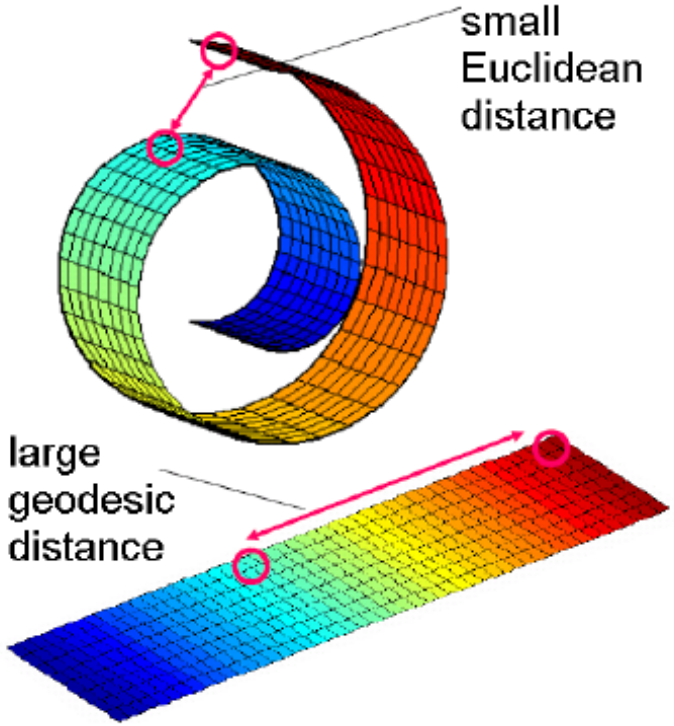
Geodesic: the shortest curve on a manifold that connects two points on the manifold

Example (3D sphere)



Isomap:

Euclidean distance needs not be a good measure between two points on a manifold
Length of geodesic is more appropriate



Isomap-Hyper parameters:

class
sklearn.manifold.Isomap(n_components=2, eigen_solver='auto', tol=0, max_iter=None)

eigen_solver:
'auto' : Attempt to choose the most efficient solver for the given problem.
'arpack' : Use Arnoldi decomposition to find the eigenvalues and eigenvectors.
'dense' : Use a direct solver (i.e. LAPACK) for the eigenvalue decomposition.

tol float, default=0
Convergence tolerance passed to arpack or lobpcg. not used if eigen_solver == 'dense'.

max_iter int, default=None
Maximum number of iterations for the arpack solver. not used if eigen_solver == 'dense'.

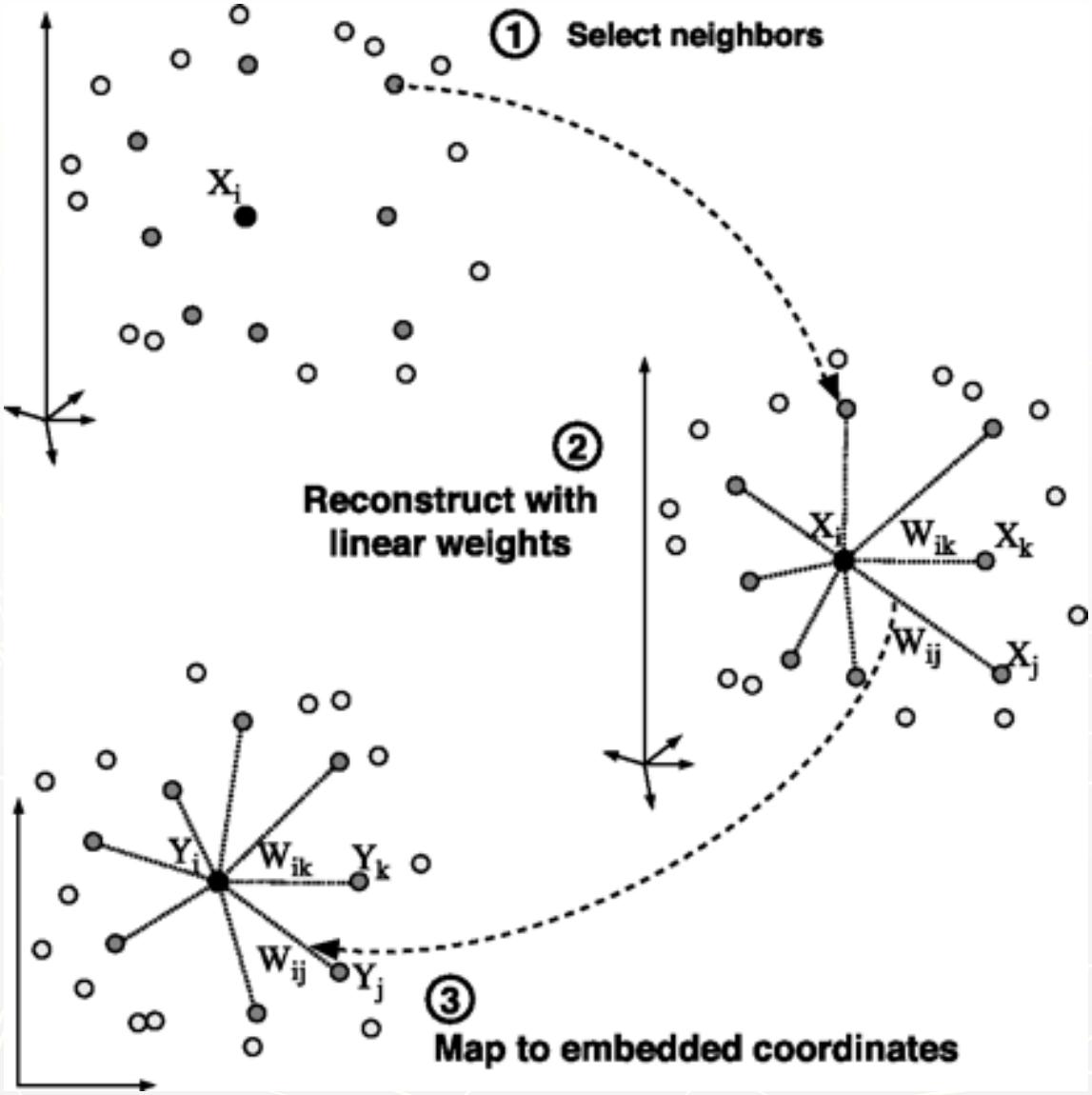
Locally Linear Embedding (LLE)

MDS is good choice if the embedding in higher dimensions are made through simple operations like rotations, translations, and scaling.

MDS fails in case when the embedding is nonlinear—that is, when it goes beyond this simple set of operations.

The solution is Locally Linear Embedding (LLE)

Locally Linear Embedding (LLE): Process



class
sklearn.manifold.
LocallyLinearEmbedding(n_neighbors=5, n_components=2, eigen_solver='auto', tol=1e-06, max_iter=100)

eigen_solver:

'auto' : Attempt to choose the most efficient solver for the given problem.
'arpack' : Use Arnoldi decomposition to find the eigenvalues and eigenvectors.
'dense' : Use a direct solver (i.e. LAPACK) for the eigenvalue decomposition.

tol float, default=0
Convergence tolerance passed to arpack or lobpcg. not used if eigen_solver == 'dense'.

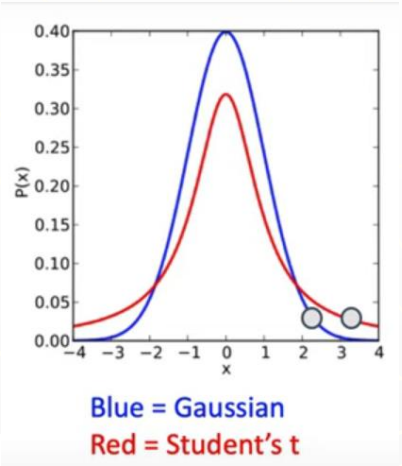
max_iter int, default=None
Maximum number of iterations for the arpack solver. not used if eigen_solver == 'dense'.

t-SNE

t-SNE is implemented by converting the high-dimensional **Euclidean distances** between **data points** into **conditional probabilities** that represent similarities.

SNE: Gaussian Distribution

t-SNE: Student-t distribution



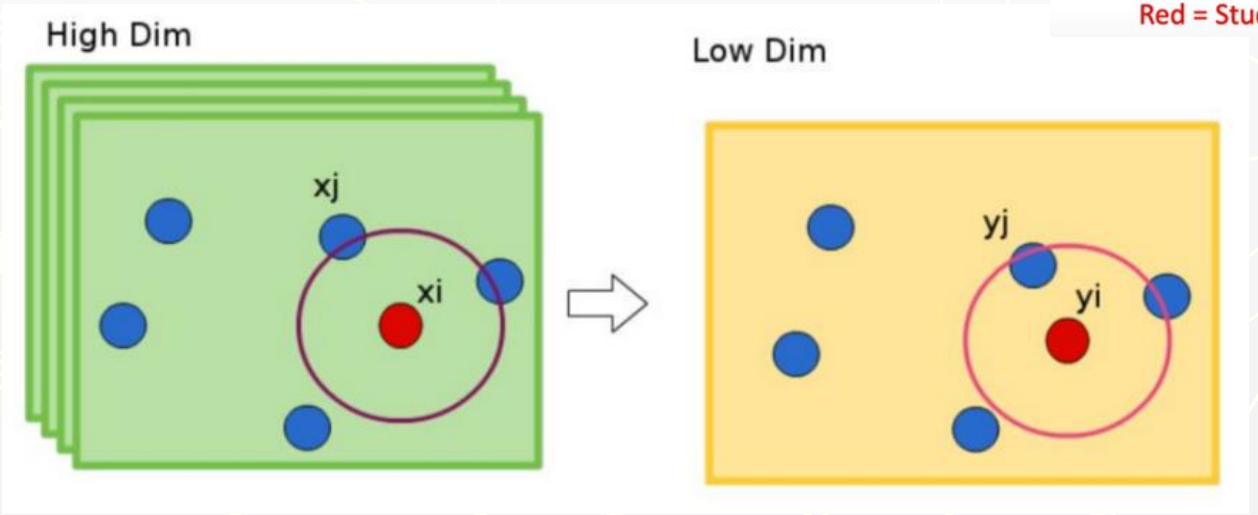


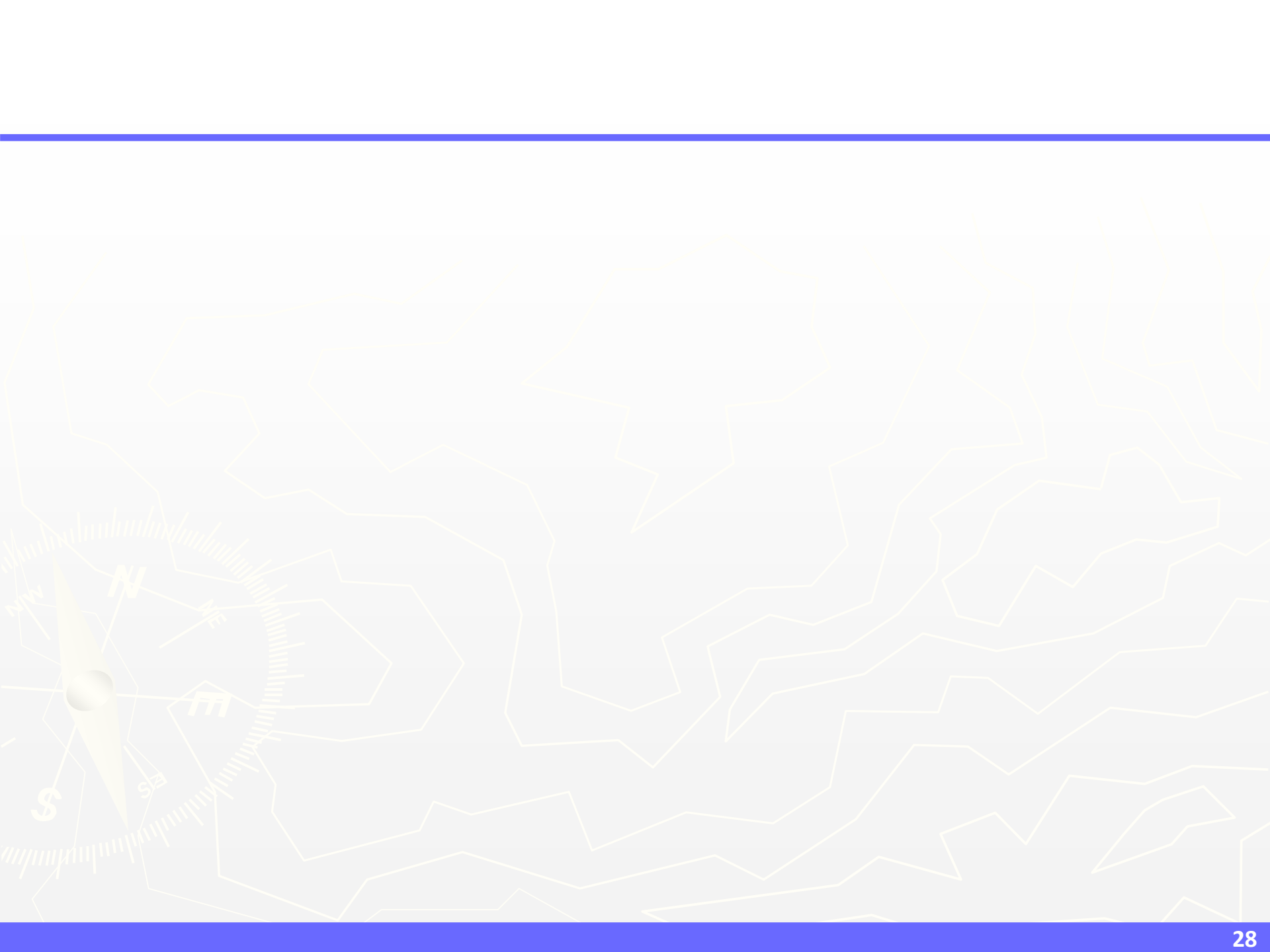
Image Source: Dimensionality reduction with t-SNE, Zaur Fataliyev

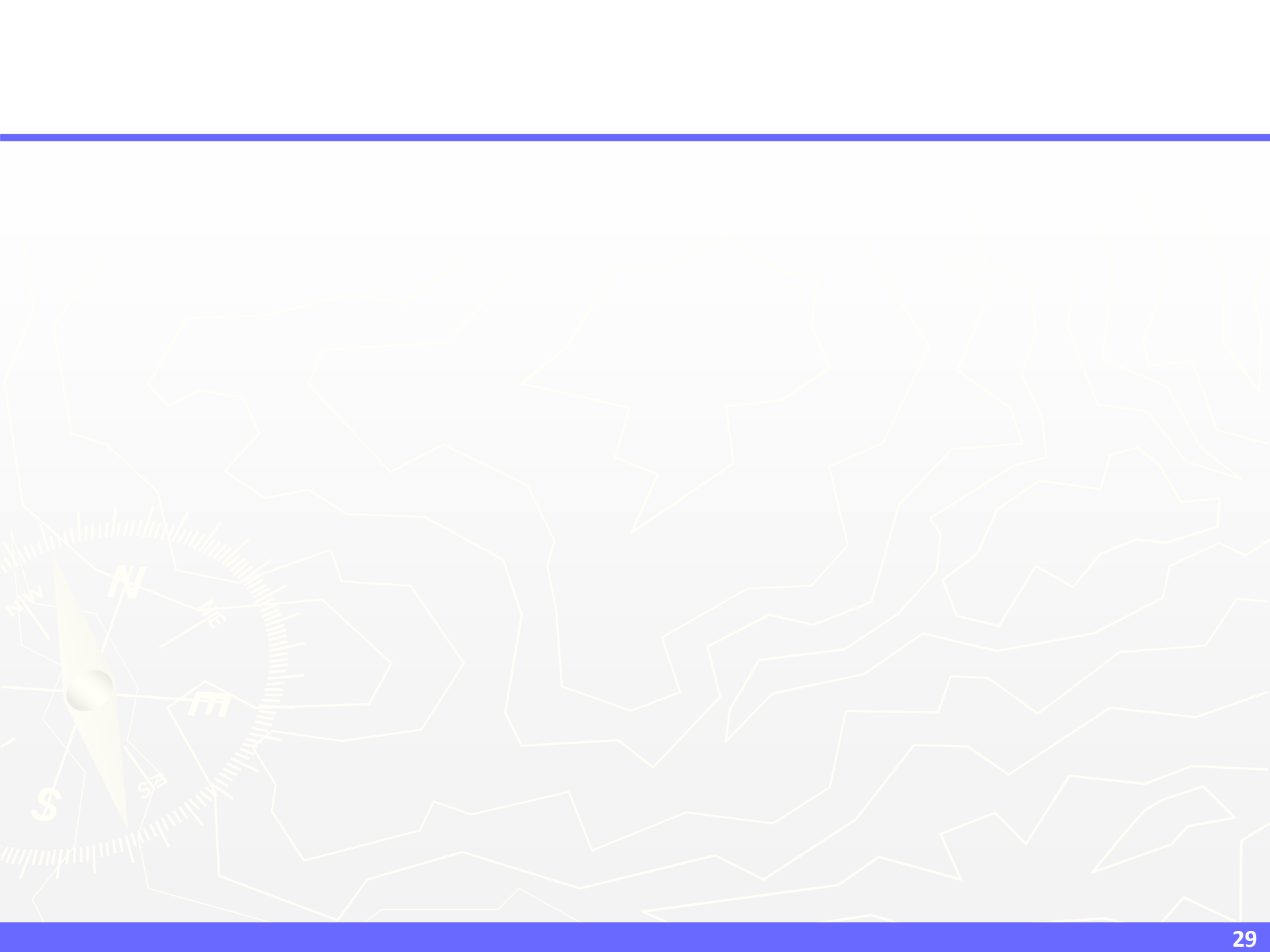
Sklearn t-SNE functionality

class sklearn.manifold.TSNE(*n_components=2*, *perplexity=30.0*)

Perplexity float, default=30.0

The perplexity is related to the number of nearest neighbors that is used in other manifold learning algorithms. Larger datasets usually require a larger perplexity. Consider selecting a value between 5 and 50. Different values can result in significantly different results.





THANKYOU