

# DATA MINING

---

## Dimensionality Reduction

### Manifold Learning

Ashish Pujari

# Lecture Outline

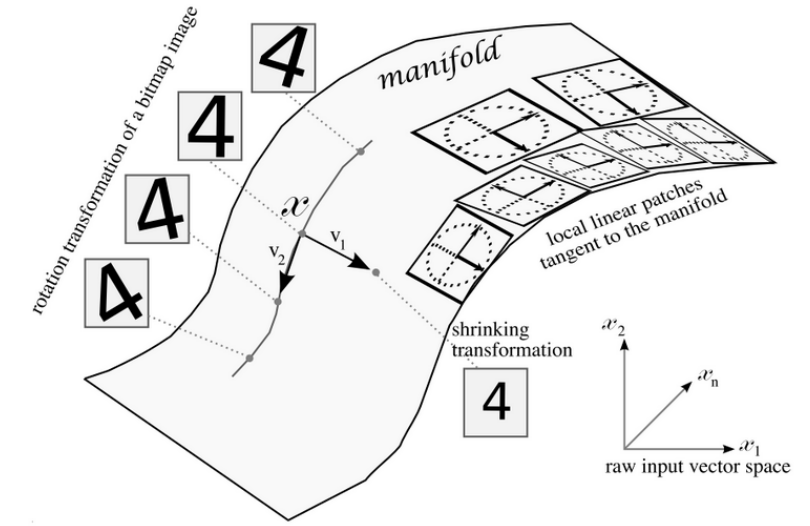
- Manifold Learning
- Algorithms
  - Kernel PCA
  - t-SNE
  - U-MAP

# MANIFOLD LEARNING

---

# Manifold Learning

- What is Manifold Learning ?
  - Approach to non-linear dimensionality reduction
- Manifold Hypothesis
  - Assumption: Observed data lie on a low-dimensional manifold embedded in a higher-dimensional space i.e. the dimensionality of many data sets is only artificially high
- Examples
  - Mostly unsupervised, though supervised variants exist
  - PCA is a mapping algorithm where the manifold is linear
  - Autoencoders learn lower dimensional non-linear representation of high dimensional data



Space of natural images has the differential-geometric structure of a low-dimensional manifold embedded in the high-dimensional pixel space

# Manifold Learning: Steps

- Problem

- Input Data  $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^D$
- We assume that the data points lie on a  $d$ -dimensional manifold embedded into  $\mathbb{R}^D$ , where  $d < D$
- Task is to learn the  $d$ -dimensional manifold structure  $M$ .

- Solution

- Choose a distance measure
- Construct k-Neighborhood Graph, neighborhood radius  $\varepsilon$

$$N_k(x_i) = \{x \in X \mid x \text{ is a } k\text{-nearest neighbour of } x_i\}$$

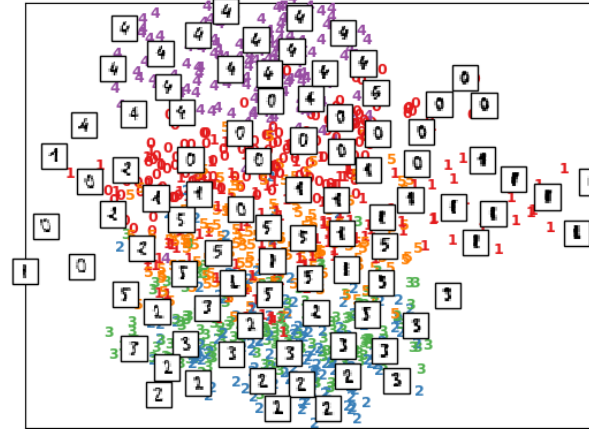
- Construct a  $n \times n$  sparse distance matrix to represent the graph
- Find the diffeomorphisms (differentiable map between manifolds)

$$f_i: N_k(x_i) \rightarrow U_i \subset \mathbb{R}^k$$

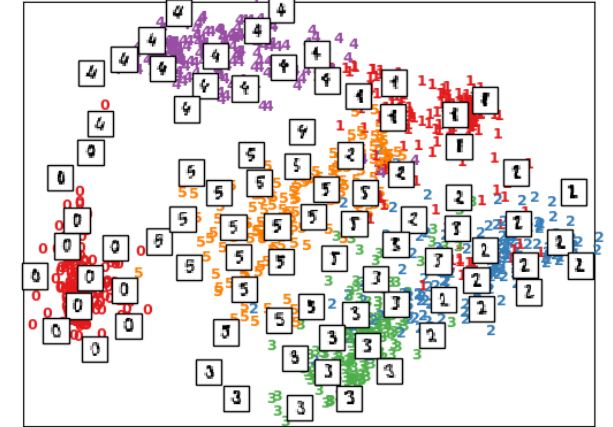
# Manifold Learning: Algorithms

- Algorithms
  - KPCA
  - MDS
  - t-SNE
  - UMAP
  - LLE
  - Isomap
  - Laplacian Eigenmaps
  - Feedforward Autoencoders

Principal Components projection of the digits (time 0.00s)

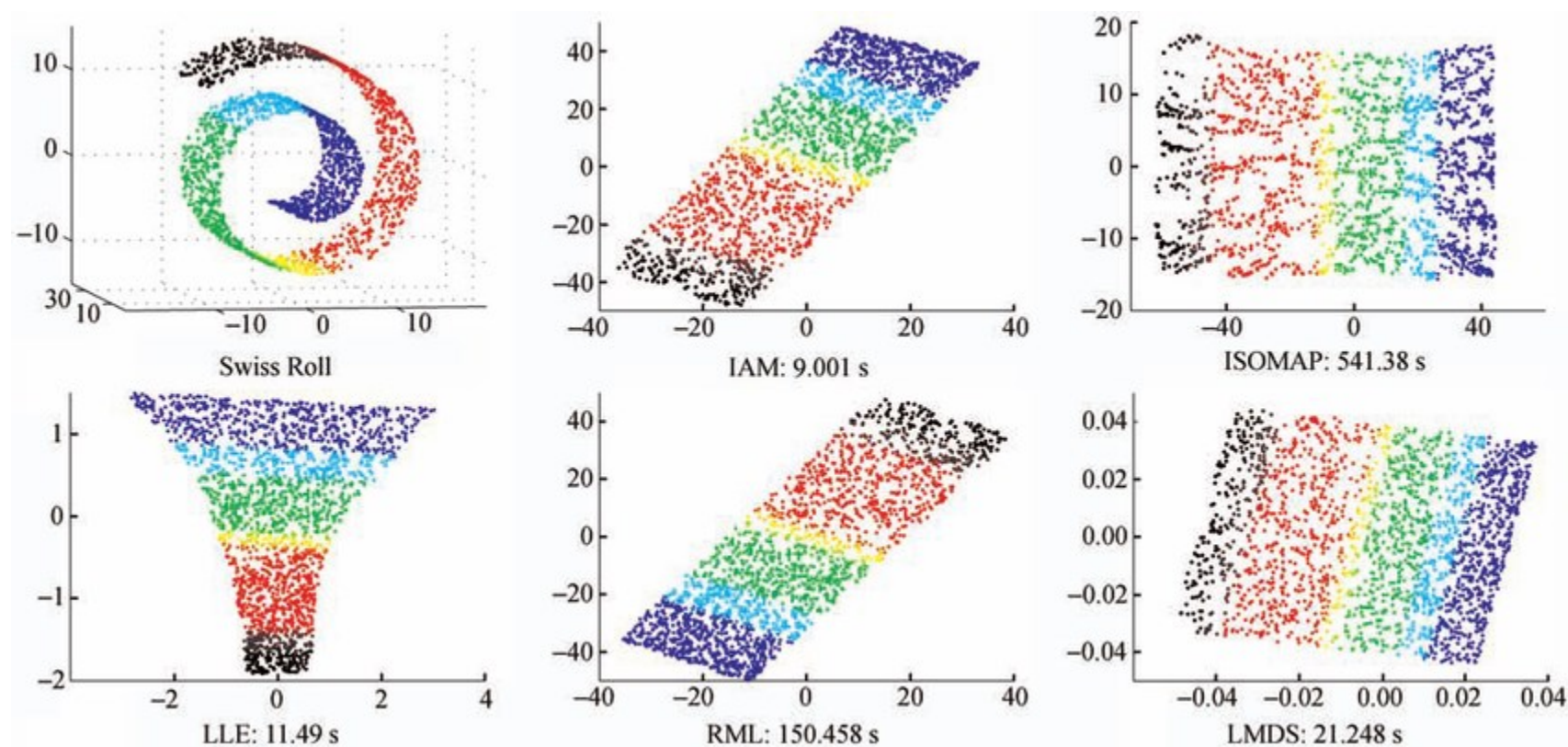


Isomap projection of the digits (time 0.97s)



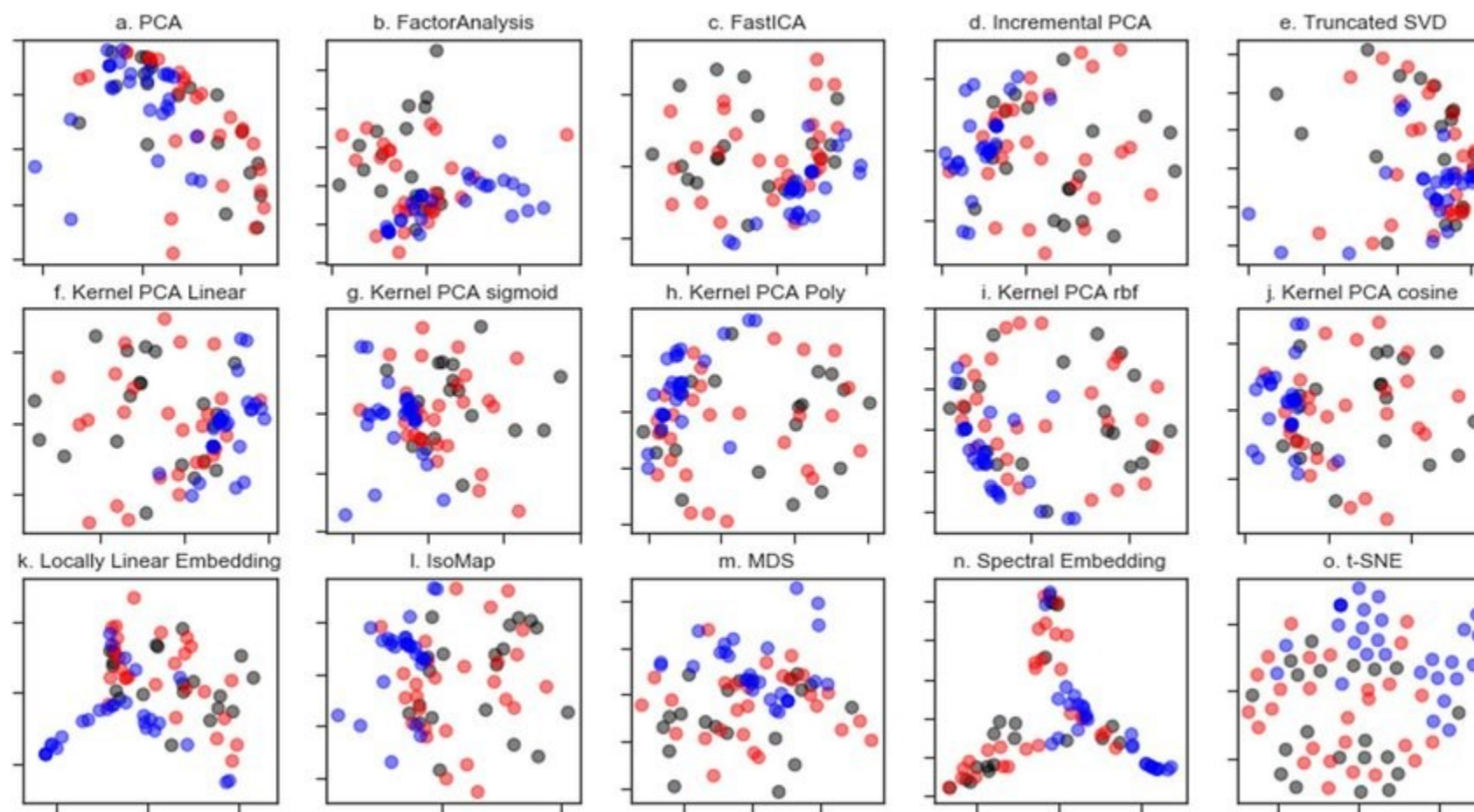
# Manifold Learning: Swissroll

- Preserving Topology vs Intrinsic Geometry





# Manifold Learning : FTIR



Dimensionality reduction methods applied to the FTIR dataset.



# Distance Measure

- Distance (or similarity) measure is a function that quantifies the similarity between two objects.
- Examples:
  - Euclidean distance, Manhattan distance, Chebyshev distance, Minkowski distance
  - Geodesic distance, Hamming Distance
- Conditions that a distance metric in real-valued vector spaces must satisfy:
  - Non-negativity:  $d(x, y) \geq 0$
  - Identity:  $d(x, y) = 0$  if and only if  $x == y$
  - Symmetry:  $d(x, y) = d(y, x)$
  - Triangle Inequality:  $d(x, y) + d(y, z) \geq d(x, z)$

# Euclidean Distance

- Euclidean Distance

$$d_{i,j} = \sqrt{\sum_{k=1}^K (x_{k,i} - x_{k,j})^2} \text{ for } i, j = 1, \dots, n$$

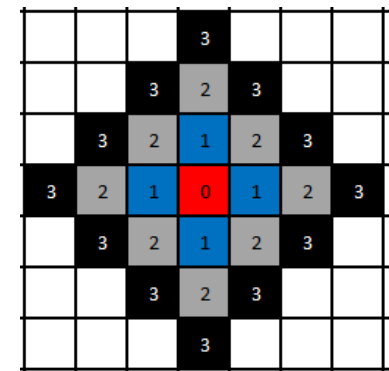
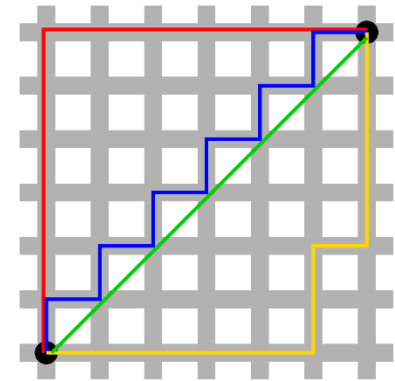
- Where  $d_{i,j}$  is the Euclidean distance between entities  $i$  and  $j$  for the  $K$  variables being considered.

# Manhattan Distance

- Manhattan Distance (City block distance or Taxicab geometry) is used if we need to calculate the distance between two data points in a grid-like path
- Example
  - In a plane with  $p_1$  at  $(x_1, y_1)$  and  $p_2$  at  $(x_2, y_2)$ , it is

$$|x_1 - x_2| + |y_1 - y_2|.$$

- Application
  - Preferred over the Euclidean distance metric as the dimension of the data increases



# Chebyshev Distance

- Chebyshev distance (or maximum metric) measures distance between two points as the maximum difference over any of their axis values
- Example
  - In a 2D grid, if we have two points  $(x_1, y_1)$ , and  $(x_2, y_2)$ , the Chebyshev distance between is  $\max(y_2 - y_1, x_2 - x_1)$
- Application
  - Warehouse logistics, as it effectively measures the time an overhead crane takes to move an object (as the crane can move on the x and y axes at the same time but at the same speed along each axis).

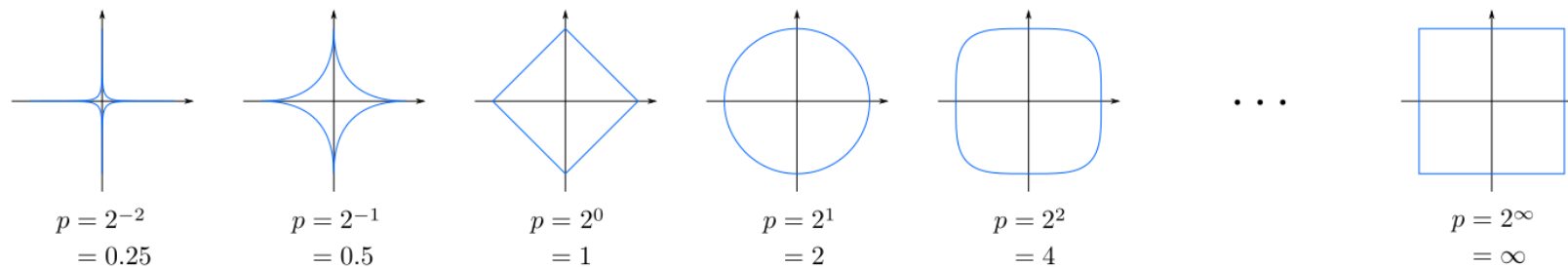
(0, 3)	(3, 1)	(3, 2)	(3, 3)
(0, 2)	(2, 1)	(2, 2)	(2, 3)
(0, 1)	(1, 1)	(1, 2)	(1, 3)
(0, 0)	(0, 1)	(0, 2)	(0, 3)

# Minkowski Distance

- Distance metric intended for real-valued vector spaces
- Minkowski distance of order  $p$  (integer) between two points:  $X = \{x_1, x_2, \dots, x_n, \in \mathbb{R}^n\}$  and  $Y = \{y_1, y_2, \dots, y_n, \in \mathbb{R}^n\}$

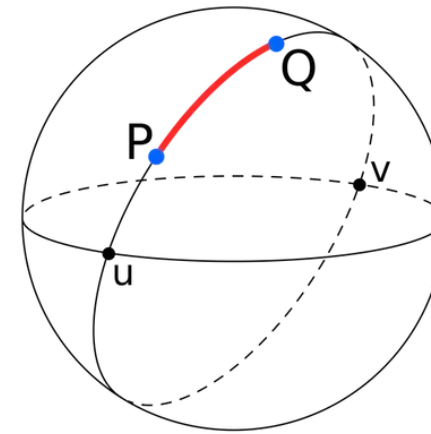
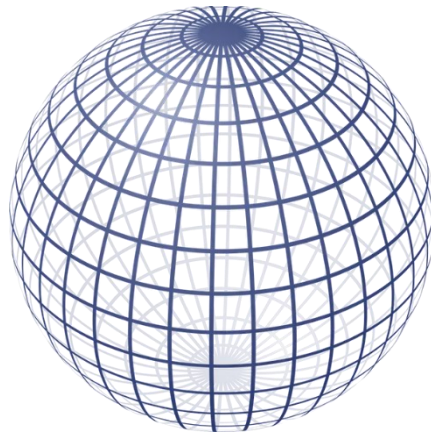
$$D(X, Y) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- Minkowski distance is a generalization of the Manhattan distance ( $p = 1$ ), Euclidean distance ( $p = 2$ ), and Chebyshev distance ( $p \rightarrow \infty$ )



# Geodesic Distance

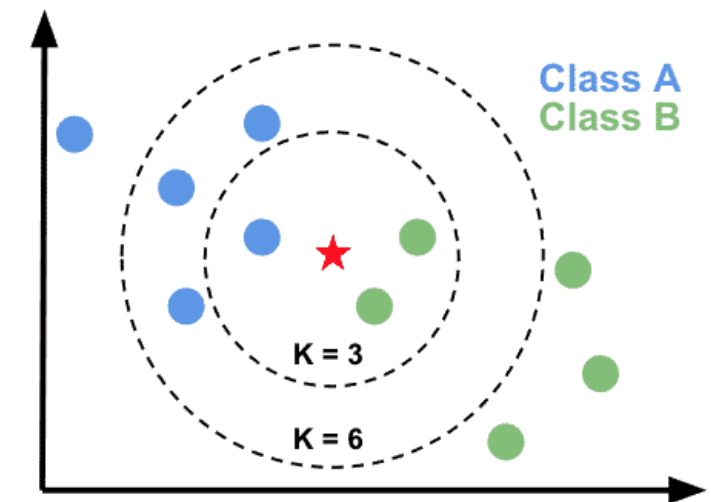
- A geodesic is the shortest path between two points in space in a curved manifold



Geodesic distance is calculated in a 3D spherical space as the distance across the curved surface of the world.

# KNN (k-Nearest Neighbor)

- Supervised learning algorithm to predict the label of data points by looking what is the majority in its closest neighbors
- Pros
  - Simple to implement and robust to noisy training data.
- Cons
  - Need to define which  $k$  value to use
  - High cost of computation compared to other algorithms.
  - Memory based data storage, so less efficient.





# Entropy

- Entropy is used to measure the uncertainty of a system.
- In Information theory, it quantifies the number of bits required to encode and transmit an event from a probability distribution.

$$H(P) = H(p_1, p_2, \dots, p_n) = - \sum_x P(x) \log P(x)$$

- When there are  $n$  symbols, all equally probable, the probability of any of them is  $1/n$ .

$$h = - \sum \frac{1}{n} (\log 1/n) = \log(n)$$

- Examples:
  - Fair Coin :  $(\frac{1}{2} * \log(1/2) + \frac{1}{2} * \log(1/2)) = \log(2) = 1$
  - Fair Dice :  $(6 * (\frac{1}{6} * \log(1/6))) = \log(6) = 2.58$

# KL Divergence

- KL divergence measures how one probability distribution differs from a second distribution.
- Discrete Probability Distributions

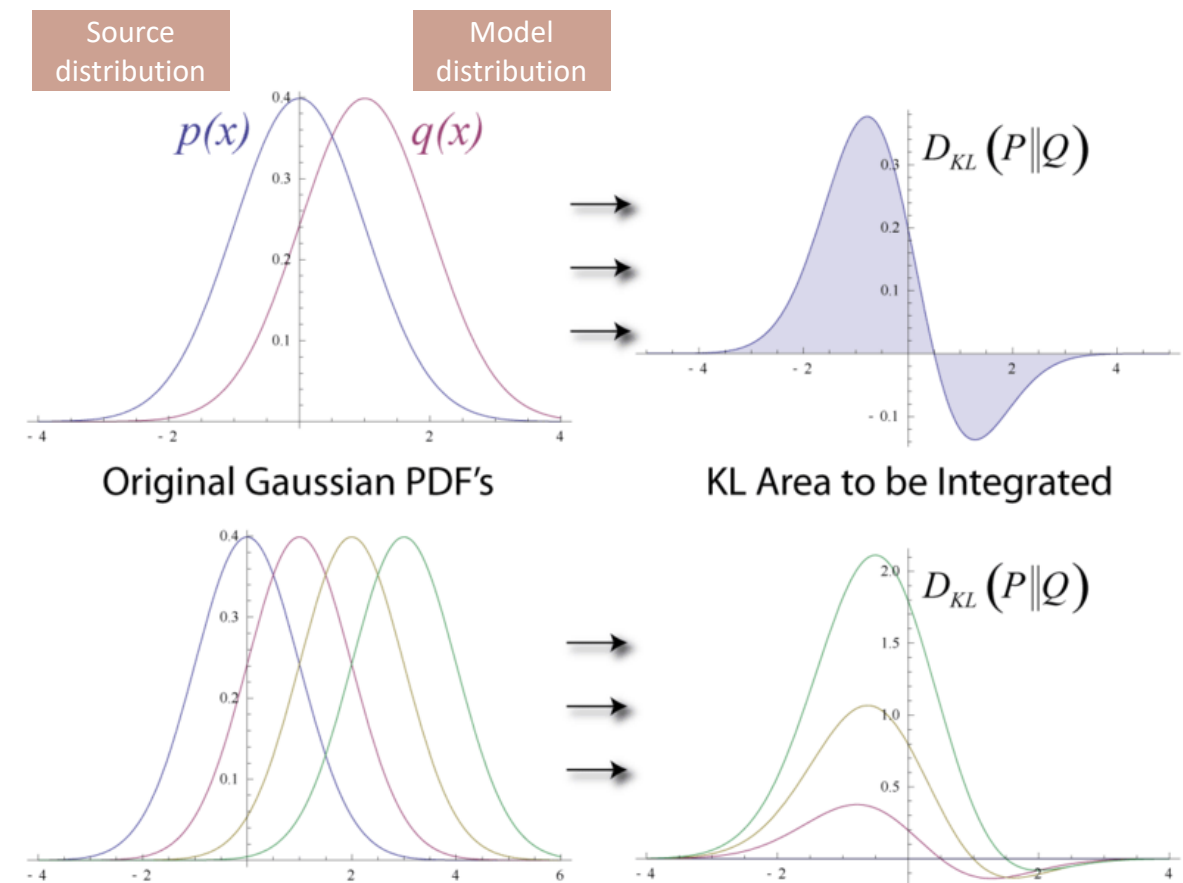
$$KL(P \parallel Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

- Continuous Probability Distributions

$$KL(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} d(x)$$

where  $p$  and  $q$  denote densities of  $P$  and  $Q$

- KL divergence is non symmetric and non-negative



# Cross Entropy

- Cross entropy is the average number of bits used to transfer the information and always greater than or equal to the entropy

$$H(q, p) = H(p) + D_{KL}(p, q) = - \sum_i p_i \log(q_i)$$

$$\text{Cross Entropy} = \text{Entropy} + \text{KL Divergence}$$

- Minimizing cross entropy is the same as minimizing the KL divergence
- Used as a loss function when optimizing logistic regression and neural networks

# MANIFOLD LEARNING ALGORITHMS

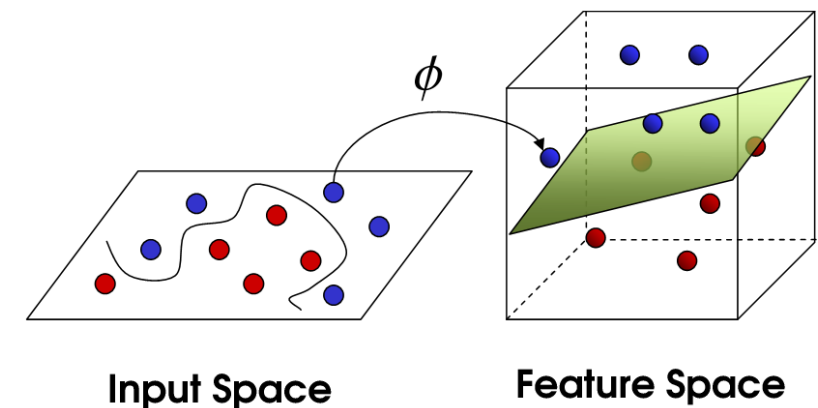
---

Kernel PCA, MDS, t-SNE, UMAP

# Kernel Function

- Kernel Function
  - A kernel expresses a measure of similarity between vectors
  - A kernel function takes as its inputs vectors in the original space and returns the dot product of the vectors in the feature space.
- Example
  - If we have data  $x, y \in X$  and a map  $\phi: X \rightarrow \mathbb{R}^N$  then below is a kernel function

$$k(x, y) = \langle \phi(x), \phi(y) \rangle$$



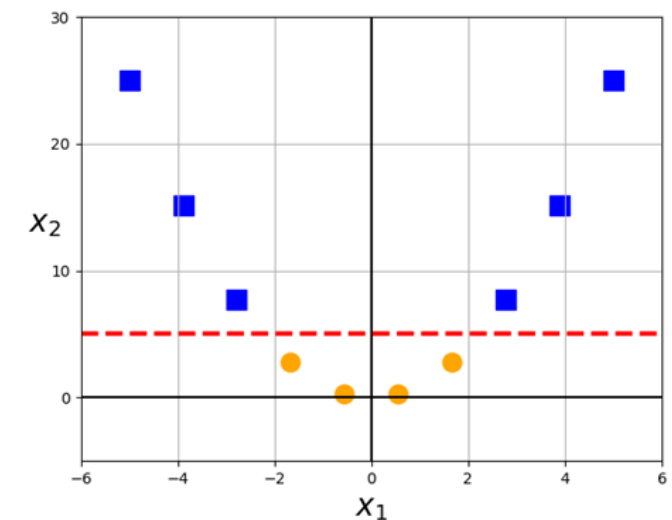
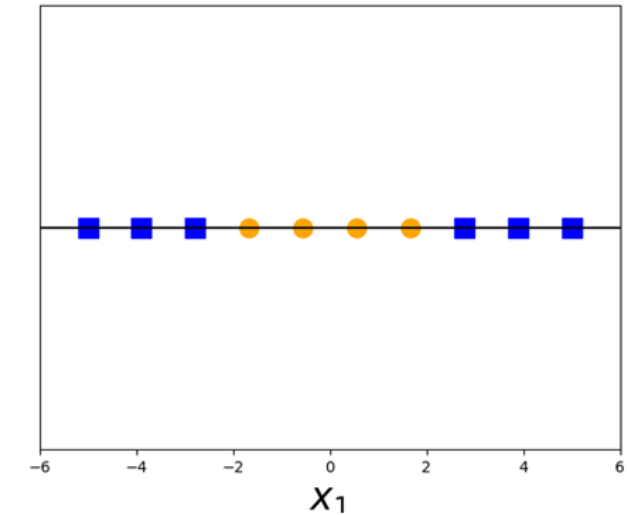
# Polynomial Kernel

- Computes relationships between pairs of observations

$$k(x, y) = (x^\top y + c_0)^d$$

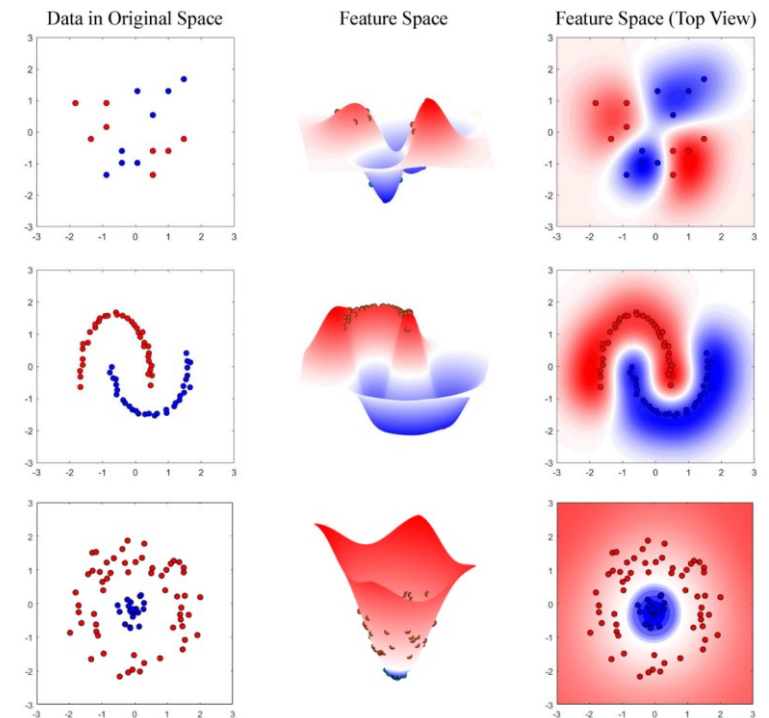
- $x, y$  are the input vectors
  - $c_0$  is the coefficient
  - $d$  is the kernel degree
- Linear Kernel
    - Special case of polynomial kernel where  $d = 1$  and  $c_0 = 0$

$$k(x, y) = x^\top y$$



# Kernel PCA (KPCA)

- KPCA
  - Non-linear dimensionality reduction through the use of kernels
- Approach
  - Extends data to a high dimensional feature space using the kernel trick and extracts principal components in that space using PCA.
  - The result will be non-linear in the original data space.
- Challenges
  - Non-trivial to find a good kernel for a given problem; may not yield good results with standard kernels.
  - The kernel matrix is  $n \times n$ , so kernel PCA can have difficulties for large data sizes.
  - Cannot be used for feature selection as there are no "loadings" KPCA as the kernel principal components are computed directly, bypassing computation of principal axes using Kernel trick

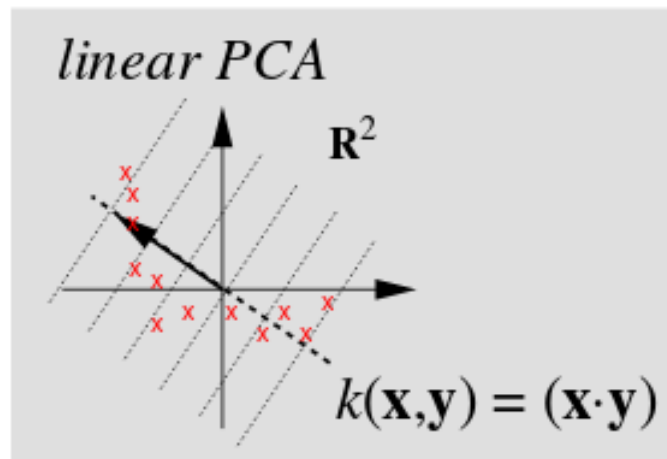




# PCA vs KPCA

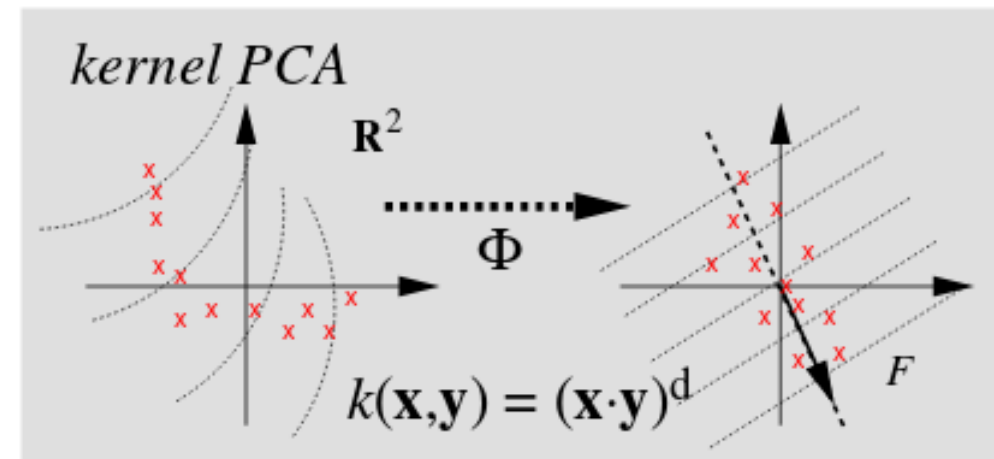
- Standard PCA is equivalent to Kernel PCA with linear kernel.
- PCAs covariance matrix scales with the number of input dimensions  $N$

$$\text{Cov} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T$$



- Unlike standard PCA, the eigenvectors of KPCA are not the principal component axes but the samples projected onto those axes.
- KPCAs kernel matrix scales with the number of data points

$$\text{Cov} = \frac{1}{N} \sum_{i=1}^N \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T$$



# KPCA

- The kernel is related to the transform  $\phi(\mathbf{x})$ , and the resulting  $n \times n$  kernel matrix looks like this:

$$\mathbf{K} = \mathbf{k}(\mathbf{x}, \mathbf{y}) = (\phi(\mathbf{x}), \phi(\mathbf{y})) = \phi(\mathbf{x})^T \phi(\mathbf{y})$$

- Mean of the data in feature space

$$\mu = \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i) = 0$$

- Covariance

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T$$

- Eigenvectors

$$\mathbf{C} \mathbf{v} = \lambda \mathbf{v}$$

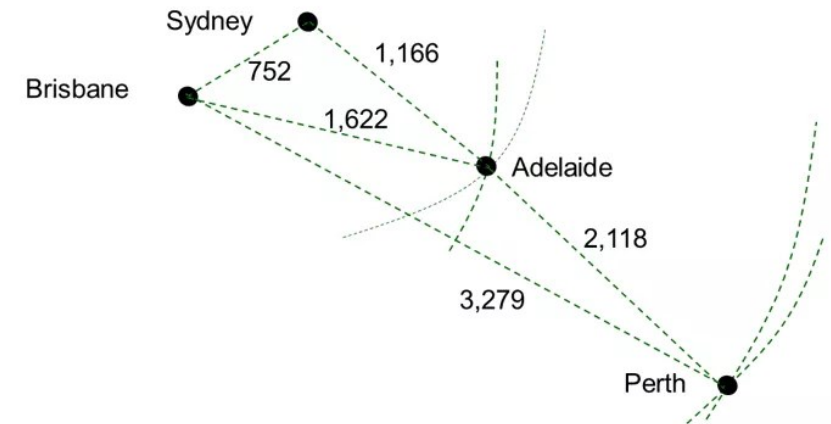
- Eigenvectors can be expressed as a linear combination of features

$$\mathbf{v} = \sum_{i=1}^n \alpha_i \phi(\mathbf{x}_i) = 0$$

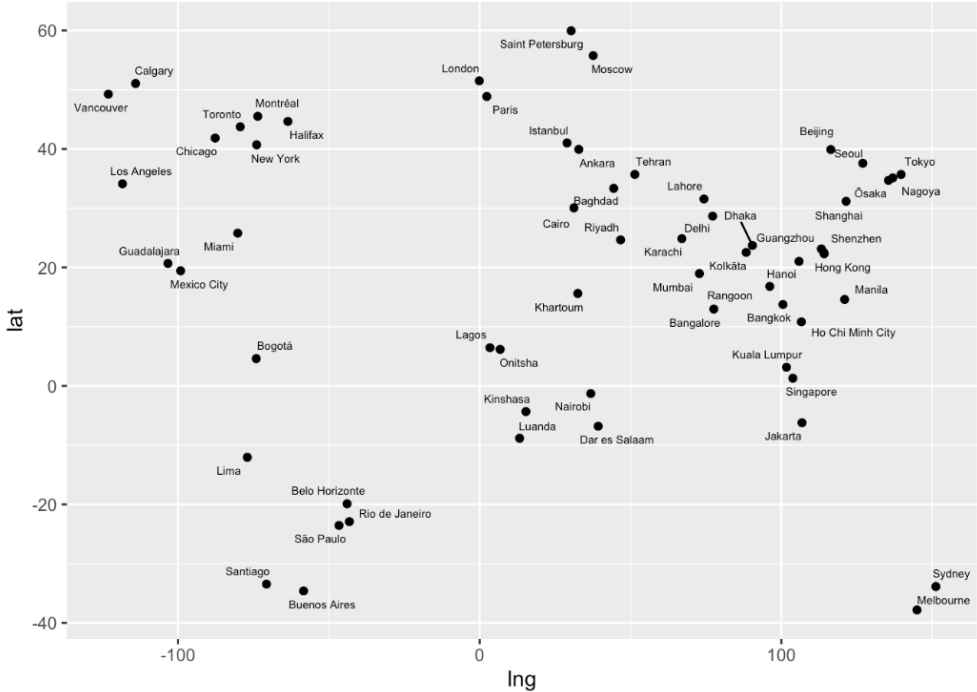
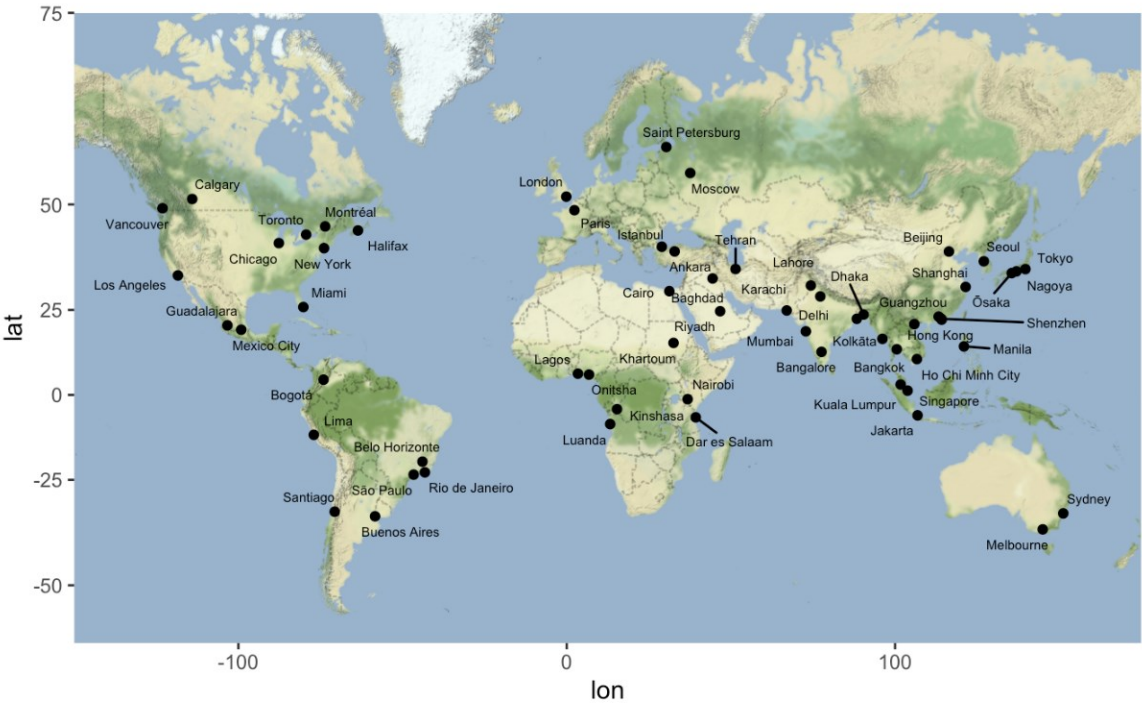
- Finding the eigenvectors is equivalent to finding the coefficients  $\alpha_i$
- KPCA algorithm collects the top  $k$  eigenvectors of the centered kernel matrix based on their corresponding eigenvalues, which are ranked by decreasing magnitude.

# Multidimensional Scaling (MDS)

- MDS
  - Means of visualizing the level of similarity of individual cases of a dataset
- Approach
  - MDS takes the pairwise distances between the entities and finds best-fit representations of the points in all lower dimensional spaces.
  - Input is a similarity/distance matrix, providing measurements between all pairs of data.
  - The output is typically a two-dimensional scatterplot, where each of the objects is represented as a point.
- Applications
  - Perceptual Mapping
  - Geo-statistics



# MDS: Application



Perceptual Mapping

# MDS: Algorithm

- Given a distance matrix  $D \in \mathbb{R}^{n \times n}$  whose entries are the pairwise distances between the data points, the MDS attempts to find a  $d$ -dimensional representation of the data ( $d < D$ ) such that the pairwise distances are preserved.

$$\min \sum_i \sum_j (D_{ij} - \hat{D}_{ij})^2$$

- If the distances are the Euclidean distances:

$$D_{ij} = \|x_i - x_j\|^2$$

$$\hat{D}_{ij} = \|\hat{x}_i - \hat{x}_j\|^2$$

- MDS on Euclidean distances is equivalent to PCA

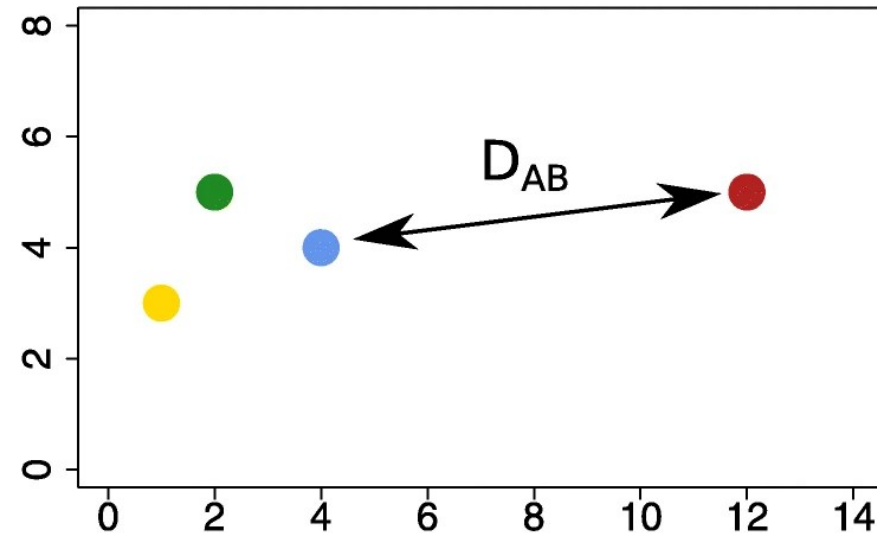
# MDS: Algorithm

- The distances between data points in the lower dimensions are not preserved exactly. The distortion in distances between the lower dimension and higher order space is called stress, which is minimized by the MDS implementation.

A

	A	B	C	D
A	0	$d_{AB}$	$d_{AC}$	$d_{AD}$
B	$d_{AB}$	0	$d_{BC}$	$d_{BD}$
C	$d_{AC}$	$d_{BC}$	0	$d_{CD}$
D	$d_{AD}$	$d_{BD}$	$d_{CD}$	0

B



C

$$Stress = S = \sum (D_{ij} - d_{ij})^2$$

# t-SNE (t-Stochastic Neighbor Embedding )

- t-SNE
  - Reduces dimensionality while preserving local structure of the data
- Approach
  - Converts high-dimensional distances between datapoints into conditional probabilities that represent similarities.
  - Student-t distribution is used to compute the similarity between two points in the low-dimensional space, to address crowding problem
  - Trained using gradient descent and tries to minimize entropy between distributions.
- Applications
  - Primarily for visualization (detangling) high-dimensional data with multiple manifolds
- Demo
  - t-SNE



# t-SNE: Step 1

Find pairwise similarity between nearby points in a high dimensional space.

- Given high-dimensional data  $X = \{x_1, x_2, \dots, x_n\}$  and its low dimensional mapping  $Y = \{y_1, y_2, \dots, y_n\}$
- The similarity of datapoint  $x_j$  to datapoint  $x_i$  is the conditional probability,  $p_{j|i}$

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

- Symmetrize the conditional probabilities in high dimension space to get the final similarities in high dimensional space.

$$p_{ij} = \frac{(p_{i|j} + p_{j|i})}{2N}$$

## t-SNE: Step 2

Map each point in high dimensional space to a low dimensional map

- For low-dimensional counterparts  $y_i$  and  $y_j$  of the high-dimensional datapoints  $x_i$  and  $x_j$ , it is possible to compute a similar conditional probability, which we denote by  $q_{j|i}$

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

- We compute the conditional probability  $q_{j|i}$  similar to  $p_{j|i}$  centered under a Gaussian centered at point  $y_i$  and then symmetrize the probability.

## t-SNE: Step 3

Find a low-dimensional data representation that minimizes KL Divergence

- A natural measure of the faithfulness with which  $q_{j|i}$  models  $p_{j|i}$  is the KL divergence
- SNE minimizes the sum of KL divergences over all datapoints using a gradient descent method

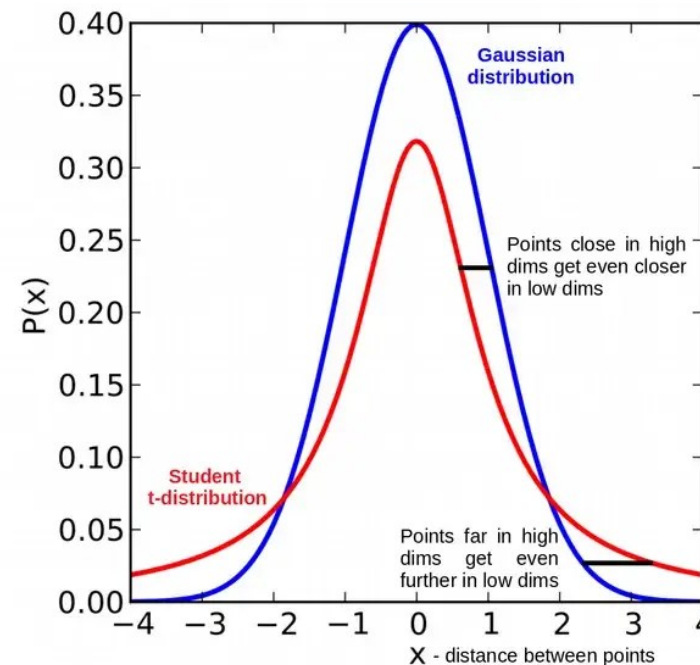
$$C = \sum_i \text{KL}(P_i \parallel Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$
$$\frac{dKL}{dy_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

- SNE cost function focuses on retaining the local structure of the data in the map

## t-SNE: Step 4

Use Student-t distribution to compute the similarity in the low-dimensional space

- The heavy tails of the Student t-distribution with one degree of freedom are here to overcome the Crowding Problem when embedding into low dimensions.



# t-SNE: Perplexity

- The perplexity can be interpreted as a measure of the effective number of neighbors

$$\text{Perp}(P_i) = 2^{H(P_i)}$$

- where  $H(P_i)$  is the Shannon entropy of  $P_i$  measured in bits

$$H(P_i) = - \sum_j P_{j|i} \log_2 P_{j|i}$$

- Typical values are between 5 and 50. The default value is 30.
- Larger datasets usually require a larger perplexity. Perplexity can have a value between 5 and 50.

# t-SNE: Limitations

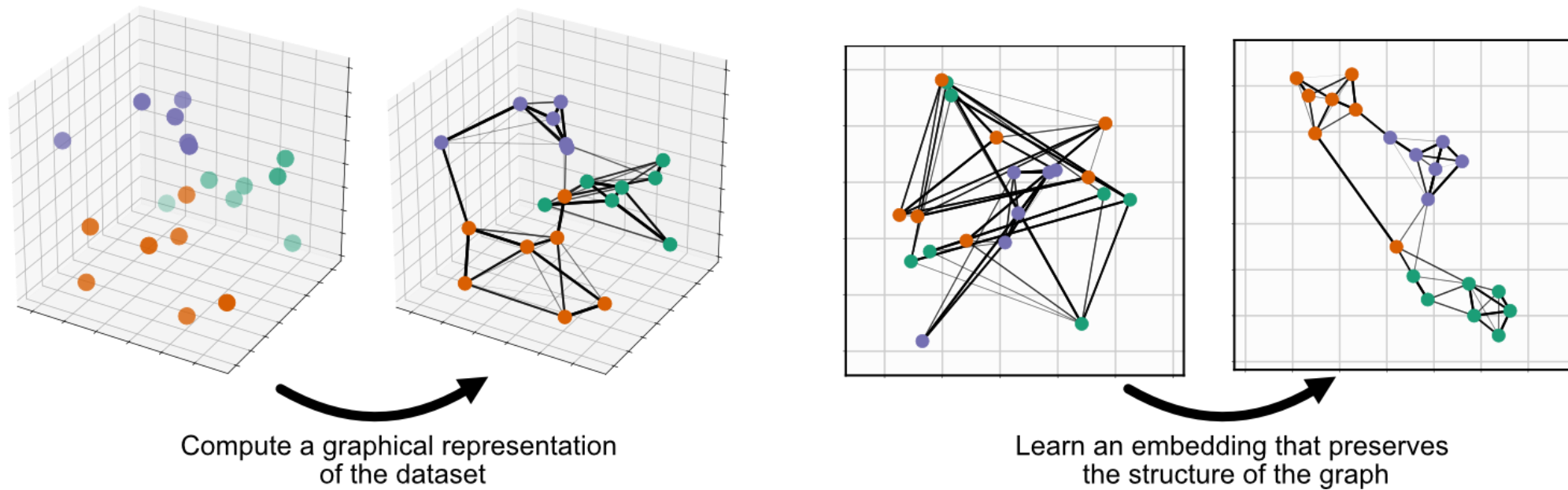
- Does not scale well to large datasets
  - Computational and memory complexity is quadratic in the number of datapoints.
  - Large memory consumption (especially when using large perplexity hyperparameter)
  - Can practically only embed into 2 or 3 dimensions
- Cannot work with high-dimensional data directly
  - Autoencoder/PCA are often used for a pre-dimensionality reduction before tSNE
- Does not preserve global data structure
  - Within cluster distances are meaningful while between cluster similarities are not guaranteed

# U-Map (Uniform Manifold Approximation and Projection)

- UMAP
  - Manifold learning techniques and ideas from topological data analysis
  - Seeks to learn the manifold structure of data and find a low dimensional embedding that preserves the essential topological structure of that manifold
- Approach
  - Builds a “fuzzy simplicial complex” and calculates similarity scores in clustered points in high dimensional space and preserve the clusters in the low dimensional graph
  - Optimization problem of finding the low dimensional representation with the closest fuzzy topological structure
  - Uses binary cross-entropy (CE) as a cost function instead of the KL-divergence like t-SNE does.
- Demo
  - [UMAP – Zoo](#)
  - [Understanding U-Map](#)



# U-Map: Algorithm



# U-Map: Algorithm

$$\sum_{e \in E} w_h(e) \log \left( \frac{w_h(e)}{w_l(e)} \right) + (1 - w_h(e)) \log \left( \frac{1 - w_h(e)}{1 - w_l(e)} \right)$$

Provides an attractive force between the points spans whenever there is a large weight associated to the high dimensional case

Provides a repulsive force between the ends of  $e$  whenever  $w_h(e)$  is small

- On balance this process of pull and push, mediated by the weights on edges of the topological representation of the high dimensional data, will let the low dimensional representation settle into a state that relatively accurately represents the overall topology of the source data.

# Summary

	PCA	t-SNE	UMAP
Linearity	Linear method.	Non-linear method.	Non-linear method.
Structure	Tries to preserve the global structure of the data.	Tries to preserve the local structure(cluster) of data.	Tries to preserve the local structure(cluster) of data while also preserving global structure as best as possible
Hyperparameters	It does not involve Hyperparameters.	It involves Hyperparameters such as perplexity, learning rate and number of steps.	It involves Hyperparameters such as number of neighbors, components, metric
Outliers	It gets highly affected by outliers.	It can handle outliers.	It can handle outliers and can also be used for outlier detection
Algorithm	Deterministic	Non-deterministic/stochastic	Non-deterministic/stochastic
Approach	Works by rotating the vectors for preserving variance.	Works by minimizing the distance between the point in a gaussian.	Works by using graph layout algorithms to arrange data in low-dimensional space
Preserves	global structures and variance by selecting appropriate number of eigen values.	preserve local structures - distance from high dimensional representation using hyperparameters.	preserve the clusters or topological structure of the high dimensional representation
Scale and Speed	Variations can scale well to large datasets and is relatively fast	Does not scale well to large datasets and can be slow	Handles large datasets and high dimensional data, and is faster than TSNE