# Dimensionality Reduction (Contd)

CS771: Introduction to Machine Learning

Piyush Rai

# Plan

- A fast method for computing eigenvectors (power method)
- Supervised dimensionality reduction
- Nonlinear dimensionality reduction
  - Kernel PCA
  - Manifold Learning (LLE and SNE/tSNE)

# Power Method for Computing Eigenvectors

- Eigen-decomposition is expensive in general – $O(D^3)$ for a $D \times D$ matrix
- For naïve methods, even to get one eigenvector, we need to perform full eigen-decom.
- If we want $K < D$ eigenvectors, there are some more efficient methods
- Power Method (a.k.a. Power Iteration) is one such iterative approach
  - Sequentially finds the top $K$ eigenvectors of a cov matrix $\mathbf{S} = \sum_{k=1}^{D} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top$

$O(KD^2)$ cost to find $K$ top eigenvectors

- Based on the fact that any vector $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \sum_{k=1}^{D} z_k \boldsymbol{u}_k$, and thus

$$\mathbf{S}\boldsymbol{x} = \sum_{k=1}^{D} z_k \lambda_k \boldsymbol{u}_k \quad \text{and} \quad \underbrace{(\mathbf{SS}\ldots,\mathbf{S})}_{M \text{ times}} \boldsymbol{x} = \sum_{k=1}^{D} z_k \lambda_k^M \boldsymbol{u}_k$$

- Assuming $\lambda_1 > \lambda_2 \geq \lambda_3 \ldots \geq \lambda_D$ then for large $M$

$$\underbrace{(\mathbf{SS}\ldots,\mathbf{S})}_{M \text{ times}} \boldsymbol{x} \approx z_1 \lambda_1^M \boldsymbol{u}_1$$

# Power Method for Computing Eigenvectors

- So we had the following:

$$\underbrace{(\mathbf{SS}\ldots,\mathbf{S})}_{M \text{ times}}\boldsymbol{x} \approx z_1\lambda_1^M\boldsymbol{u}_1$$

- This gives us a simple algorithm to get the top eigenvector
  - Initialize $\boldsymbol{x}_0 \sim \mathcal{N}(0, \mathbf{I}_D)$
  - For $m = 1, \ldots, M$, compute $\boldsymbol{x}_m$ as $\boldsymbol{x}_m = \mathbf{S}\boldsymbol{x}_{m-1}$ and normalize it as $\boldsymbol{x}_m = \boldsymbol{x}_m/||\boldsymbol{x}_m||_2$
  - After convergence, $\boldsymbol{x}_M$ is the largest eigenvector and $||\mathbf{S}\boldsymbol{x}_M||$ is the largest eigenvalue

Since eigenvectors should have unit norm

Using the fact $\boldsymbol{Sx} = \lambda\boldsymbol{x}$ and that $\boldsymbol{x}$ has unit norm

- The main dominant cost is computing $\boldsymbol{Sx_{m-1}}$ whose cost is $\boldsymbol{O}(D^2)$
- Can use this technique to also obtain the remain eigenvectors sequentially using a "peeling" technique

# Power Method with Peeling Technique

- Can use Power Method with a "peeling" technique to get all the top $K$ eigenvectors

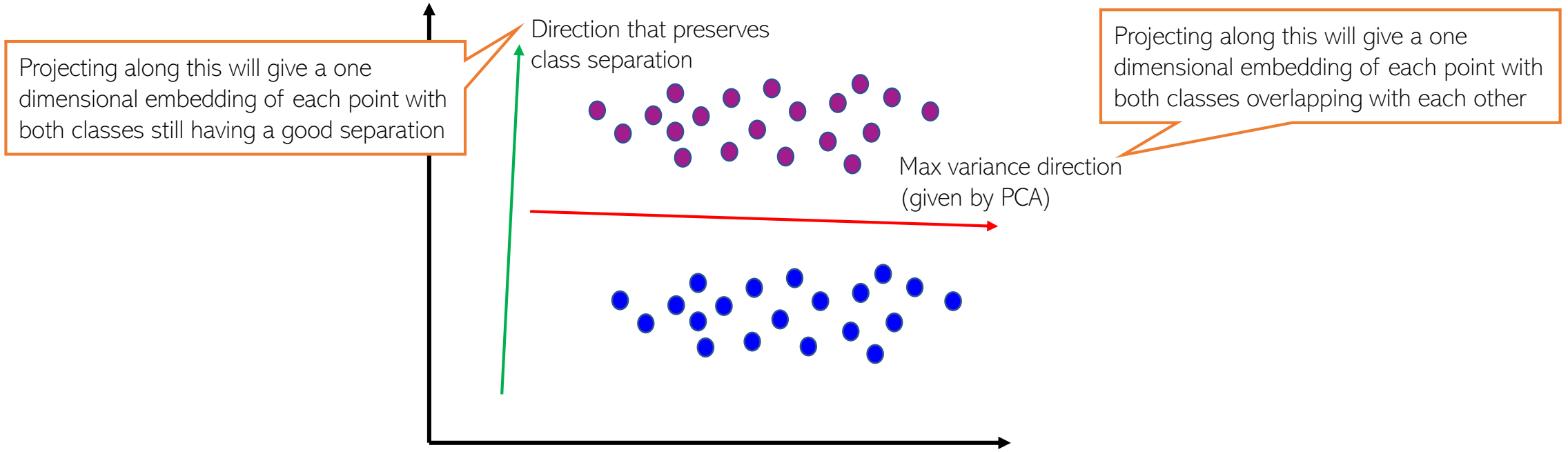
  - The basic procedure would be
    - Initialize $\mathbf{S}^{(0)} = \mathbf{S}$ ($\sum_{k=1}^{D} \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top$)
    - For $k = 1, \ldots, K$

$$\{\boldsymbol{u}_k, \lambda_k\} = \text{POWER-METHOD}(\mathbf{S}^{(k-1)})$$
$$\mathbf{S}^{(k)} = \mathbf{S}^{(k-1)} - \lambda_k \boldsymbol{u}_k \boldsymbol{u}_k^\top \quad (\text{"Peeling" the covariance matrix})$$

- Each power iteration is $O(D^2)$, overall cost for getting $K$ eigenvectors is $O(KD^2)$

# Supervised Dimensionality Reduction

- Maximum variance directions may not be aligned with class separation directions




- Be careful when using PCA for supervised learning problems
- A better option would be to project such that
  - Points within the same class are close (low intra-class variance)
  - Points from different classes are well separated (the class means are far apart)

# Supervised Dimensionality Reduction

- Many techniques. A simple yet popular one is Fisher Discriminant Analysis, also known as Linear Discriminant Analysis (FDA or LDA)

This LDA should not be confused with another very popular ML technique for finding topics in text data (Latent Dirichlet Allocation)

- For simplicity, assume two classes (can be generalized for more than 2 classes too)
- Suppose a projection direction $\boldsymbol{u}$. After projection the means of the two classes are

$$\mu_1 = \frac{1}{N_1}\sum_{n:y_n=1} \boldsymbol{u}^\top \boldsymbol{x}_n, \quad \mu_2 = \frac{1}{N_2}\sum_{n:y_n=2} \boldsymbol{u}^\top \boldsymbol{x}_n$$

Here we considered projection to one dimension but can be generalized to projection to $K$ dim

- Total variance of the points after projection will be $s_1^2 + s_2^2$ where

$$s_1^2 = \frac{1}{N_1}\sum_{n:y_n=1} (\boldsymbol{u}^\top \boldsymbol{x}_n - \mu_1)^2, \quad s_2^2 = \frac{1}{N_2}\sum_{n:y_n=2} (\boldsymbol{u}^\top \boldsymbol{x}_n - \mu_2)^2$$

- Fisher discriminant analysis finds the optimal projection direction by solving

$$\arg\max_{\boldsymbol{u}} \frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2}$$

Push the means far apart

Make each class tightly packed after projection (small variance)

The solution to this problem involves solving an eigendecomposition problem that involves within class covariance matrices and between class covariance matrices

# Dimensionality Reduction given Pairwise Distances between points

# Dim. Reduction by Preserving Pairwise Distances

- PCA/SVD etc assume we are given points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ as vectors (e.g., in $D$ dim)
- Often the data is given in form of distances $d_{ij}$ between points $(i, j = 1,2, \ldots, N)$
- Would like to project data such that pairwise distances between points are preserved

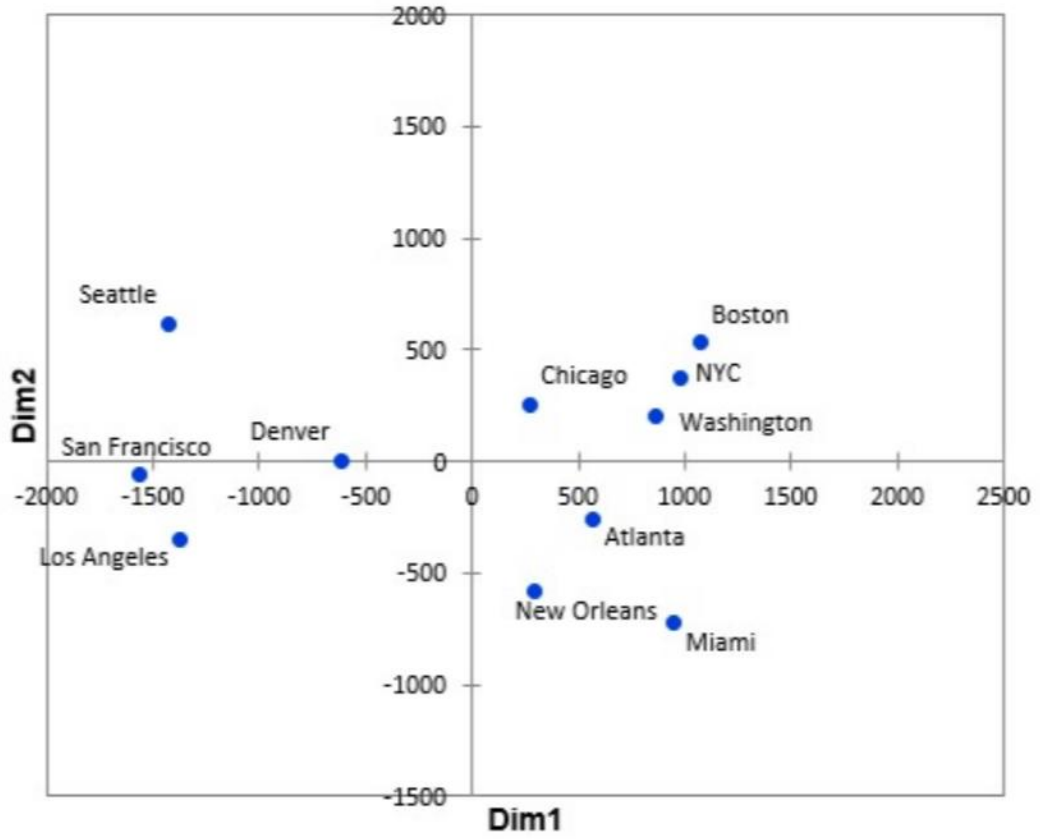
$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \mathcal{L}(\mathbf{Z}) = \arg\min_{\mathbf{Z}} \sum_{i,j=1}^{N} (d_{ij} - ||\boldsymbol{z}_i - \boldsymbol{z}_j||)^2$$

$\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ denote low-dim embeddings/projections of points $\boldsymbol{i}$ and $\boldsymbol{j}$

- Basically, if $\boldsymbol{d_{ij}}$ is large (resp. small), would like $\|\boldsymbol{z}_i - \boldsymbol{z}_j\|$ to be large (resp. small)
- Multi-dimensional Scaling (MDS) is one such algorithm
- Note: If $\boldsymbol{d_{ij}}$ is the Euclidean distance, MDS is equivalent to PCA
- The above approach tries to preserve all pairwise distances
  - Can try to preserve pairwise distances only between close-by points (i.e., b/w nearest neighbors). It helps achieve non-linear dim red. Algos like Isomap and locally linear embedding (LLE) do this

# MDS: An Example

- Result of applying MDS (with $K = 2$) on pairwise distances between some US cities

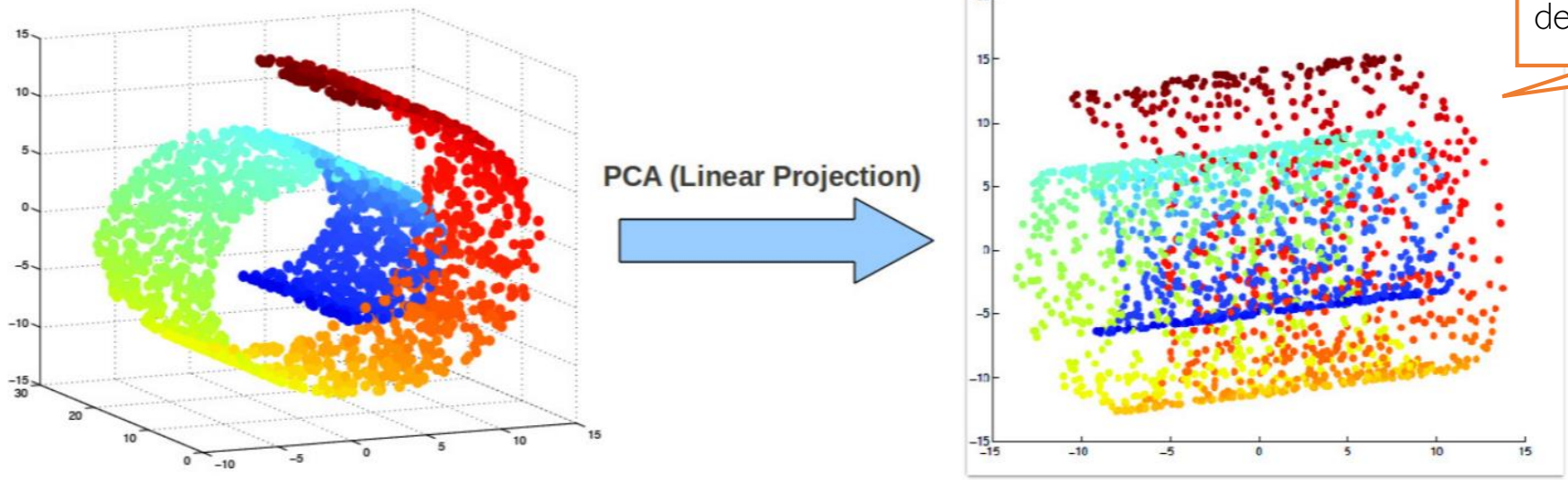



- MDS produces a 2D embedding such that geographically close cities are also close in embedding space

# Nonlinear Dimensionality Reduction

# Beyond Linear Projections

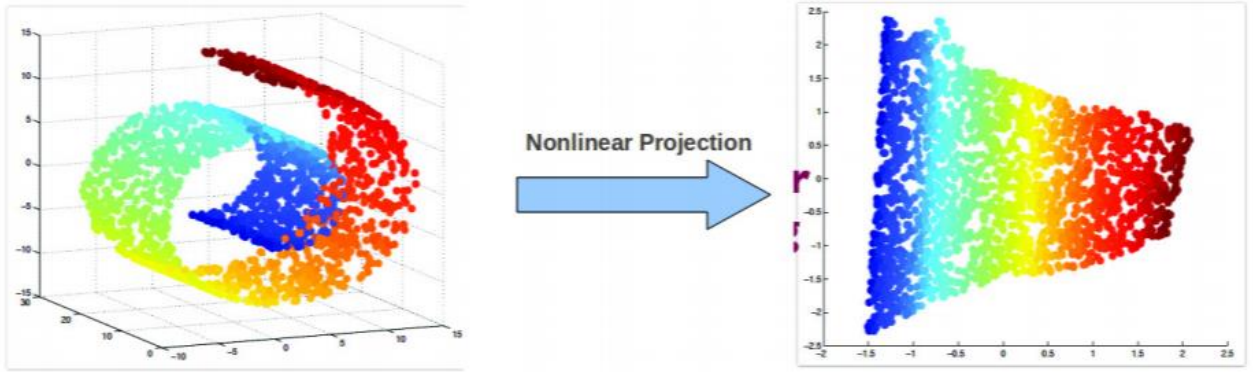
- Consider the swiss-roll dataset (points lying close to a manifold)




- Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities
  - Maximum variance directions may not be the most interesting ones

# Nonlinear Dimensionality Reduction

- We want to a learn nonlinear low-dim projection




Relative positions of points preserved after the projection

- Some ways of doing this
  - Nonlinearize a linear dimensionality reduction method. E.g.:
    - Cluster data and apply linear PCA within each cluster (mixture of PCA)
    - Kernel PCA (nonlinear PCA)
  - Using manifold based methods that intrinsically preserve nonlinear geometry, e.g.,
    - Locally Linear Embedding (LLE), Isomap
    - Maximum Variance Unfolding
    - Laplacian Eigenmap, and others such as SNE/tSNE, etc.
- .. or use unsupervised deep learning techniques (later)

Will look at KPCA, LLE, SNE/tSNE

# Kernel PCA

- Recall PCA: Given $N$ observations $\boldsymbol{x}_n \in \mathbb{R}^D$, $n = 1,2,\ldots,N$,

$D \times D$ cov matrix assuming centered data

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

$D$ eigenvectors of $\mathbf{S}$

$$\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \;\forall i = 1,\ldots,D$$

- Assume a kernel $k$ with associated $M$ dimensional nonlinear map $\boldsymbol{\phi}$

$M \times M$ cov matrix assuming centered data in the kernel-induced feature space

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} \phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top$$

$M$ eigenvectors of $\mathbf{C}$

$$\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \;\forall i = 1,\ldots,M$$

- Would like to do it without computing $\mathbf{C}$ and the mappings $\boldsymbol{\phi}(\boldsymbol{x}_n)'s$ since $M$ can be very large (even infinite, e.g., when using an RBF kernel)

- Boils down to doing eigendecomposition of the $N \times N$ kernel matrix $\mathbf{K}$ (PRML 12.3)
  - Can verify that each $\boldsymbol{v}_i$ above can be written as a lin-comb of the inputs: $\boldsymbol{v}_i = \sum_{n=1}^{N} a_{in}\boldsymbol{\phi}(\boldsymbol{x}_n)$
  - Can show that finding $\boldsymbol{a}_i = [a_{i1}, a_{i2}, \ldots, a_{iN}]$ reduces to solving an eigendecomposition of $\mathbf{K}$
  - Note: Due to req. of centering, we work with a centered kernel matrix $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$

$N \times N$ matrix of all 1s

# Locally Linear Embedding

Several non-lin dim-red algos use this idea

Essentially, neighbourhood preservation, but only local

- Basic idea: If two points are local neighbors in the original space then they should be local neighbors in the projected space too
- Given $N$ observations $\boldsymbol{x}_n \in \mathbb{R}^D$, $n = 1,2,\ldots,N$, LLE is formulated as

Solve this to learn weights $W_{ij}$ such that each point $\boldsymbol{x}_i$ can be written as a weighted combination of its local neighbors in the original feature space

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \sum_{j \in \mathcal{N}(i)} W_{ij}\boldsymbol{x}_j||^2$$

$\mathcal{N}(i)$ denotes the local neighbors (a predefined number, say K, of them) of point $\boldsymbol{x}_i$

- For each point $\boldsymbol{x}_n \in \mathbb{R}^D$, LLE learns $\boldsymbol{z}_n \in \mathbb{R}^K$, $n = 1,2,\ldots,N$ such that the same neighborhood structure exists in low-dim space too

Requires solving an eigenvalue problem

$$\hat{\mathbf{Z}} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{N} ||\boldsymbol{z}_i - \sum_{j \in \mathcal{N}(i)} W_{ij}\boldsymbol{z}_j||^2$$

- Basically, if point $\boldsymbol{x}_i$ can be reconstructed from its neighbors in the original space, the same weights $W_{ij}$ should be able to reconstruct $\boldsymbol{z}_i$ in the new space too

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science 290 (2000)

# SNE and t-SNE

Thus very useful if we want to visualize some high-dim data in two or three dims

- Also nonlin. dim-red methods, especially suited for projecting to 2D or 3D
- SNE stands for Stochastic Neighbor Embedding (Hinton and Roweis, 2002)
- Uses the idea of preserving probabilistically defined neighborhoods
- SNE, for each point $\boldsymbol{x}_i$, defines the probability of a point $\boldsymbol{x}_j$ being its neighbor as

Neighbor probability in the original space

$$p_{j|i} = \frac{\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma^2)}{\sum_{k\neq i}\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_k||^2/2\sigma^2)}$$

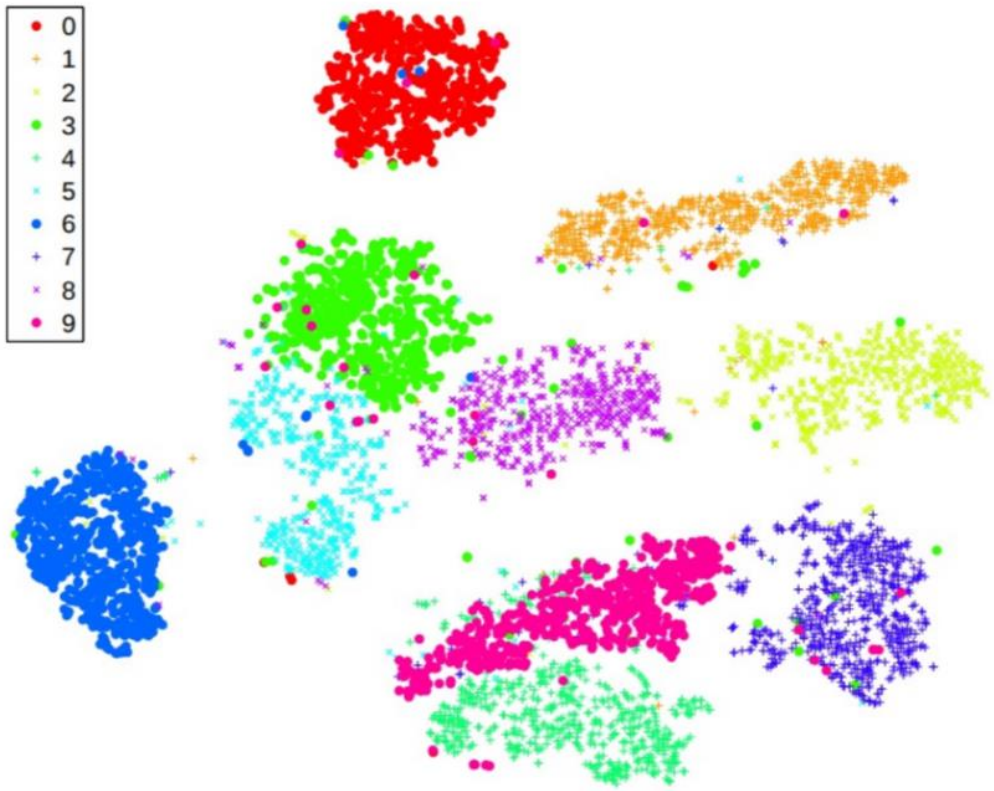
Neighbor probability in the projected/embedding space

$$q_{j|i} = \frac{\exp(-||\boldsymbol{z}_i - \boldsymbol{z}_j||^2/2\sigma^2)}{\sum_{k\neq i}\exp(-||\boldsymbol{z}_i - \boldsymbol{z}_k||^2/2\sigma^2)}$$

- SNE ensures that neighbourhood distributions in both spaces are as close as possible
  - By minimizing their Kullback-Leibler divergence, summed over all points $\sum_{i=1}^{N}\sum_{j=1}^{N} KL(p_{j|i}||q_{j|i})$
- t-SNE (van der Maaten and Hinton, 2008) offers a couple of improvements to SNE
  - Learns $\boldsymbol{z}_i$'s by minimizing symmetric KL divergence
  - Uses Student-t distribution instead of Gaussian for defining $q_{j|i}$

# SNE and t-SNE

- Especially useful for visualizing data by projecting into 2D or 3D




Result of visualizing MNIST digits data in 2D (Figure from van der Maaten and Hinton, 2008)