PCA: The Algorithm

- Compute the mean of the data

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

- Compute the sample covariance matrix (using the mean subtracted data)

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

- Do the eigenvalue decomposition of the $D \times D$ matrix $\mathbf{S}$
- Take the top K eigenvectors (corresponding to the top K eigenvalues)
- Call these $\mathbf{u}_1, \dots, \mathbf{u}_K$ (s.t. $\lambda_1 \geq \lambda_2 \geq \dots \lambda_{K-1} \geq \lambda_K$)
- $\mathbf{U} = [\mathbf{u}_1 \; \mathbf{u}_2 \; \dots \; \mathbf{u}_K]$ is the projection matrix of size $D \times K$

- Projection of each example $\mathbf{x}_n$ is computed as $\mathbf{z}_n = \mathbf{U}^\top \mathbf{x}_n$
  - $\mathbf{z}_n$ is a $K \times 1$ vector (also called the embedding of $\mathbf{x}_n$)

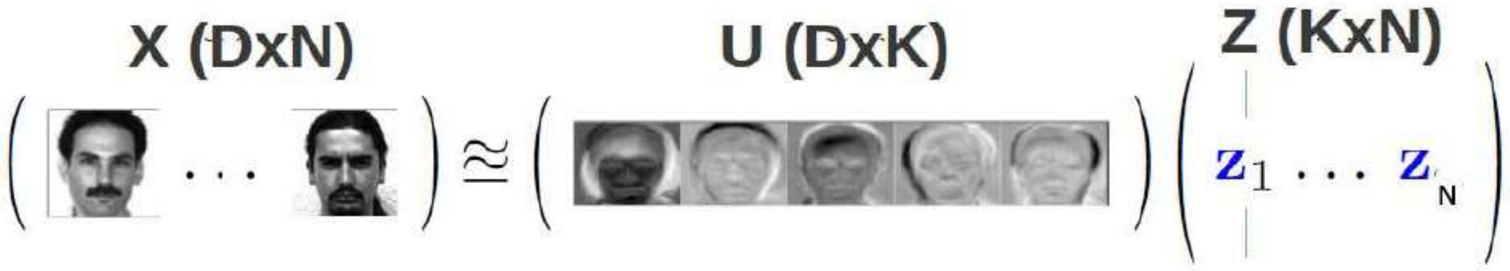
PCA: Approximate Reconstruction

- Given the principal components $\mathbf{u}_1, \dots, \mathbf{u}_K$, the PCA approximation of an example $\mathbf{x}_n$ is:

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{K} (\mathbf{x}_n^\top \mathbf{u}_i)\mathbf{u}_i = \sum_{i=1}^{K} z_{ni}\mathbf{u}_i$$

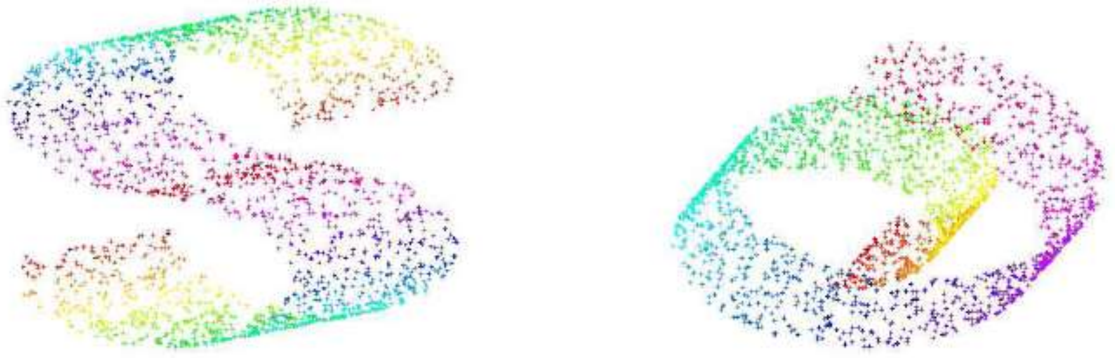
  where $\mathbf{z}_n = [z_{n1}, \dots, z_{nK}]$ is the low-dimensional projection of $\mathbf{x}_n$

- This gives us a way of compressing data
- To compress a dataset $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$, all we need is the set of $K \ll D$ principal components, and the projections $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_N]$ of each example

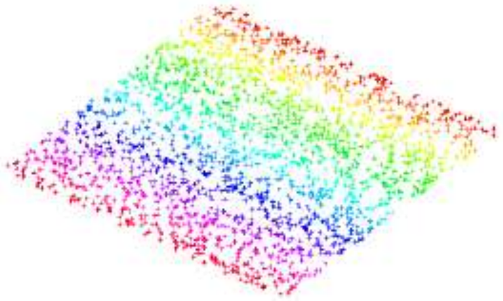


Nonlinear Extensions

- Given: Low-dim. surface embedded **nonlinearly** in high-dim. space
  - Such a structure is called a **Manifold**

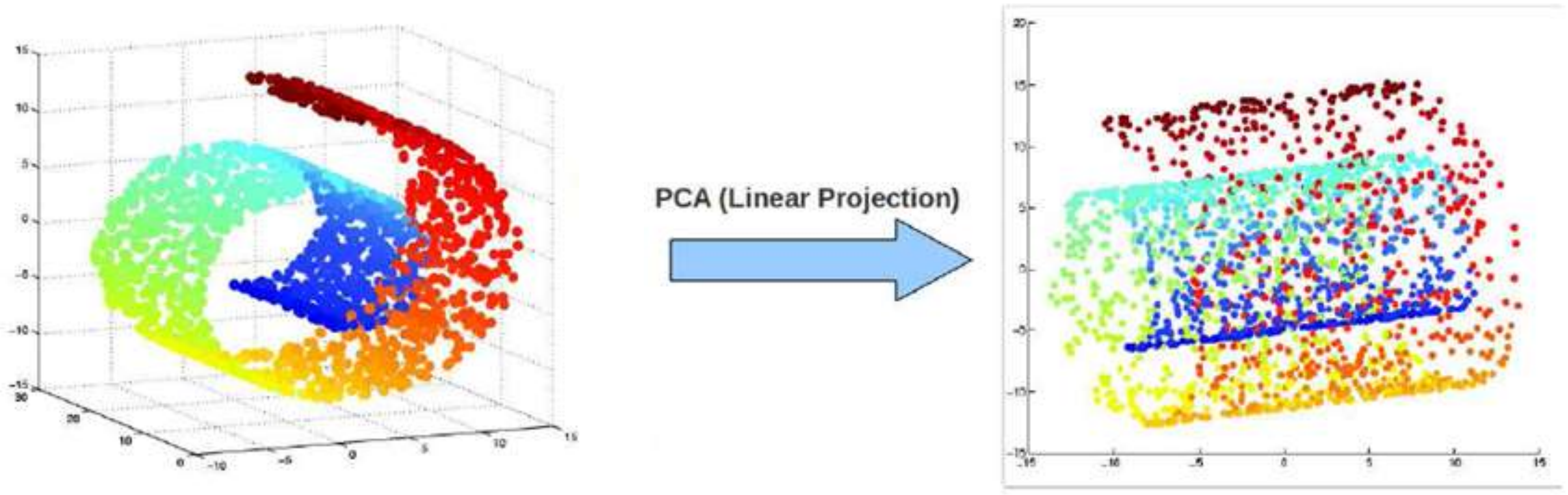


- Goal: Recover the low-dimensional surface



Nonlinear Extensions

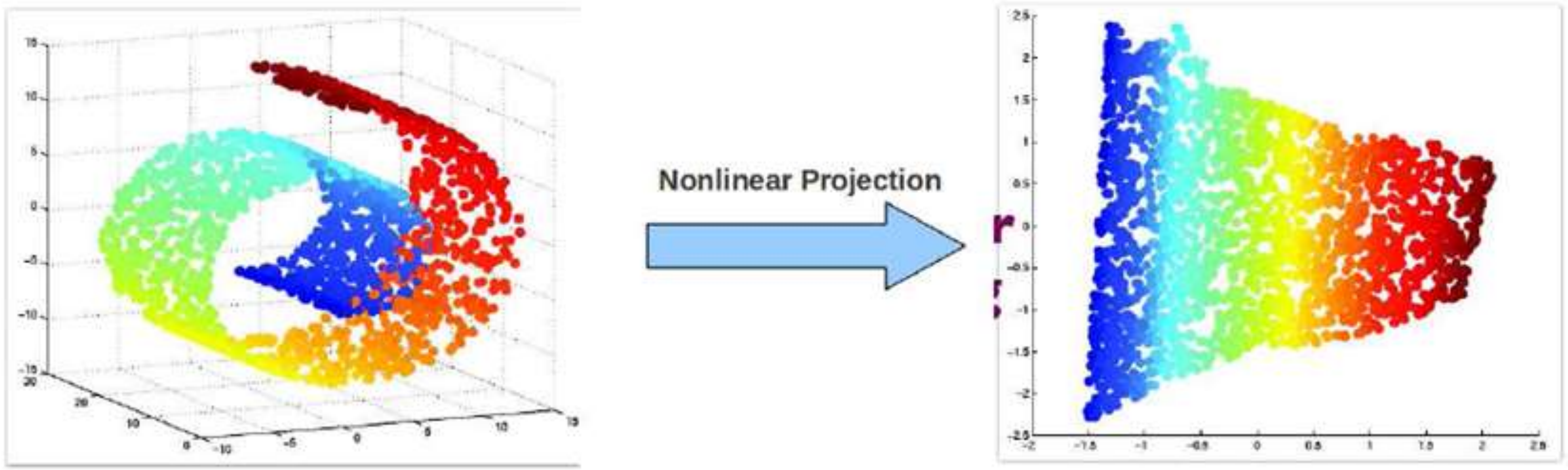
- Consider the swiss-roll dataset (points lying close to a manifold)



- Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities

Nonlinear Extensions

- We want to do nonlinear projections
- Different criteria could be used for such projections
- Most nonlinear methods try to preserve the neighborhood information
  - Locally linear structures (locally linear $\Rightarrow$ globally nonlinear)
  - Pairwise distances (along the nonlinear manifold)
- Roughly translates to "unrolling" the manifold



Nonlinear Extensions

Two ways of doing it:

- Nonlinearize a linear dimensionality reduction method. E.g.:
    - **Kernel PCA (nonlinear PCA)**
- Using manifold based methods. E.g.:
    - **Locally Linear Embedding (LLE)**
    - **Isomap**
    - Maximum Variance Unfolding
    - Laplacian Eigenmaps
    - And several others (Hessian LLE, Hessian Eigenmaps, etc.)

Kernel PCA

- Given N observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\forall \mathbf{x}_n \in \mathbb{R}^D$, define the $D \times D$ covariance matrix (assuming centered data $\sum_n \mathbf{x}_n = \mathbf{0}$)

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

- Linear PCA: Compute eigenvectors $\mathbf{u}_i$ satisfying: $\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i \ \forall i = 1, \ldots, D$
- Consider a nonlinear transformation $\phi(\mathbf{x})$ of $\mathbf{x}$ into an M dimensional space
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\mathbf{x}_n) = \mathbf{0}$)

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

- Kernel PCA: Compute eigenvectors $\mathbf{v}_i$ satisfying: $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i \ \forall i = 1, \ldots, M$
- Ideally, we would like to do this without having to compute the $\phi(\mathbf{x}_n)$'s

Kernel PCA

- Kernel PCA: Compute eigenvectors $\mathbf{v}_i$ satisfying: $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- Plugging in the expression for $\mathbf{C}$, we have the eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^\top \mathbf{v}_i\} = \lambda_i \mathbf{v}_i$$

- Using the above, we can write $\mathbf{v}_i$ as: $\boxed{\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)}$
- Plugging this back in the eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^\top \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)$$

- Pre-multiplying both sides by $\phi(\mathbf{x}_l)^\top$ and re-arranging

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_l)^\top\phi(\mathbf{x}_n)\sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_l)^\top\phi(\mathbf{x}_n)$$

Kernel PCA

- Using $\phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$, the eigenvector equation becomes:

$$\frac{1}{N}\sum_{n=1}^{N} k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^{N} a_{im} k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in} k(\mathbf{x}_l, \mathbf{x}_n)$$

- Define $\mathbf{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$
  - $\mathbf{K}$ is the similarity of two examples $\mathbf{x}_n$ and $\mathbf{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\mathbf{a}_i$ as the $N \times 1$ vector with elements a_{in}
- Using $\mathbf{K}$ and $\mathbf{a}_i$, the eigenvector equation becomes:

$$\mathbf{K}^2\mathbf{a}_i = \lambda_i N \mathbf{K}\mathbf{a}_i \quad \Rightarrow \quad \boxed{\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i}$$

- This corresponds to the original Kernel PCA eigenvalue problem $\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$
- For a projection to $K < D$ dimensions, top K eigenvectors of $\mathbf{K}$ are used

$$\boxed{\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)}$$

Kernel PCA: Centering Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\mathbf{x}_n) = \phi(\mathbf{x}_n) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)$$

- How does it affect the kernel matrix $\mathbf{K}$ which is eigen-decomposed?

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\mathbf{x}_n)^\top\tilde{\phi}(\mathbf{x}_m) \\
&= \phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_n)^\top\phi(\mathbf{x}_l) - \frac{1}{N}\sum_{l=1}^{N}\phi(\mathbf{x}_l)^\top\phi(\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\phi(\mathbf{x}_j)^\top\phi(\mathbf{x}_l) \\
&= k(\mathbf{x}_n,\mathbf{x}_m) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_n,\mathbf{x}_l) - \frac{1}{N}\sum_{l=1}^{N}k(\mathbf{x}_l,\mathbf{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(\mathbf{x}_l,\mathbf{x}_l)
\end{aligned}$$

- In matrix notation, the centered $\boxed{\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N}$
- $\mathbf{1}_N$ is the $N \times N$ matrix with every element $= 1/N$
- Eigen-decomposition is then done for the centered kernel matrix $\tilde{\mathbf{K}}$

Kernel PCA: The Projection

- Suppose $\{\mathbf{a}_1, \ldots, \mathbf{a}_K\}$ are the top K eigenvectors of kernel matrix $\tilde{\mathbf{K}}$

- The K-dimensional KPCA projection $\mathbf{z} = [z_1, \ldots, z_K]$ of a point $\mathbf{x}$:

$$z_i = \phi(\mathbf{x})^\top \mathbf{v}_i$$

- Recall the definition of $\mathbf{v}_i$

$$\boxed{\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n)}$$

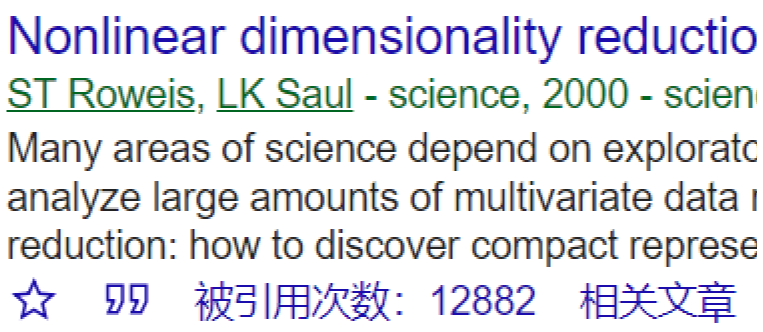
- Thus

$$\boxed{z_i = \phi(\mathbf{x})^\top \mathbf{v}_i = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n)}$$

Manifold Learning

- **Locally Linear Embedding (LLE)**
- **Isomap**
- Maximum Variance Unfolding
- Laplacian Eigenmaps
- And several others (Hessian LLE, Hessian Eigenmaps, etc.)

Locally Linear Embedding (LLE)

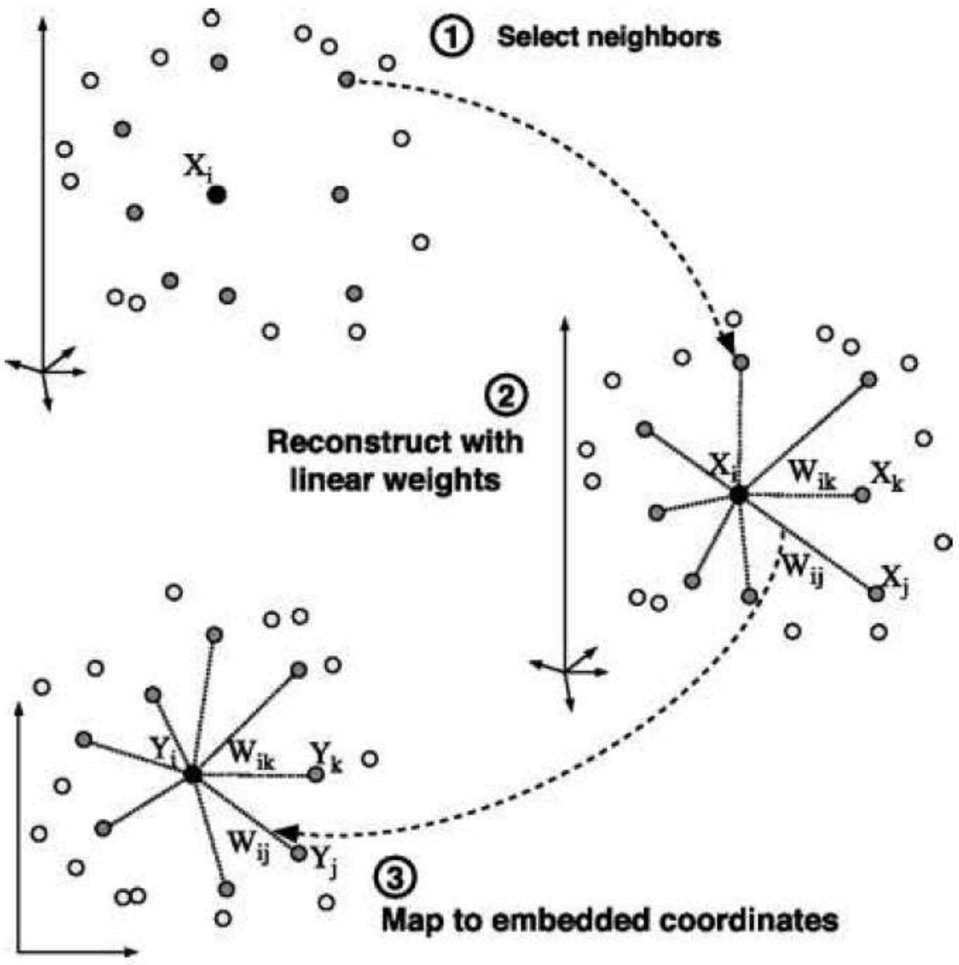

Nonlinear dimensionality reductio

ST Roweis, LK Saul - science, 2000 - scien

Many areas of science depend on exploratc
analyze large amounts of multivariate data I
reduction: how to discover compact represe

☆ 99 被引用次数：12882 相关文章

- Based on a simple geometric intuitic
- Assume each example and its neighl patch of the manifold
- LLE assumption: Projection should
  - Projected point should have the s



Locally Linear Embedding (LLE)

- Given D dim. data $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, compute K dim. projections $\{\mathbf{z}_1, \dots, \mathbf{z}_N\}$
- For each example $\mathbf{x}_i$, find its L nearest neighbors
- Assume $\mathbf{x}_i$ to be a weighted linear combination of the L nearest neighbors

$$\mathbf{x}_i \approx \sum_{j \in \mathcal{N}} W_{ij} \mathbf{x}_j \qquad \text{(so the data is assumed locally linear)}$$

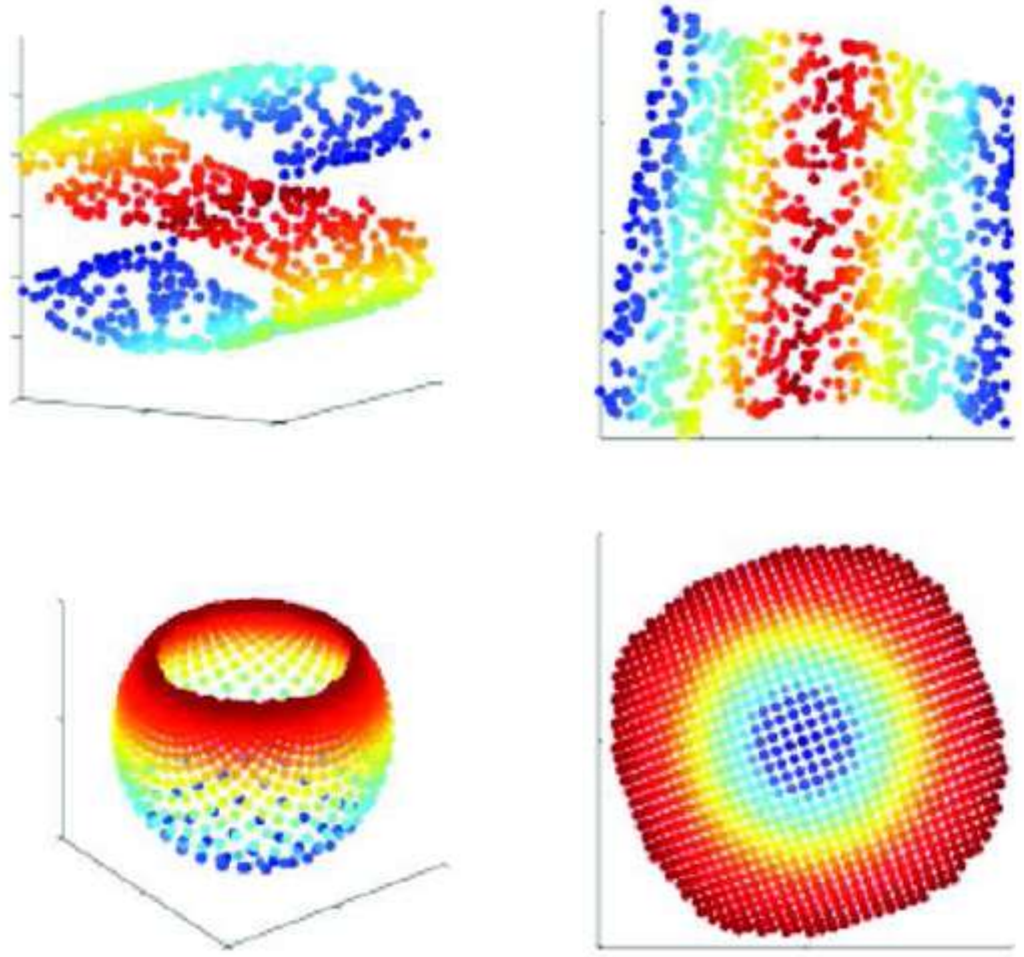
- Find the weights by solving the following least-squares problem:

$$\boxed{W = \arg\min_W \sum_{i=1}^{N} ||\mathbf{x}_i - \sum_{j \in \mathcal{N}_i} W_{ij} \mathbf{x}_j||^2} \qquad s.t. \forall i \quad \sum_j W_{ij} = 1$$

- $\mathcal{N}_i$ are the L nearest neighbors of $\mathbf{x}_i$ (note: should choose $L \geq K + 1$)
- Use W to compute low dim. projections $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ by solving:

$$\boxed{\mathbf{Z} = \arg\min_{\mathbf{Z}} \sum_{i=1}^{N} ||\mathbf{z}_i - \sum_{j \in \mathcal{N}} W_{ij} \mathbf{z}_j||^2} \qquad s.t. \forall i \quad \sum_{i=1}^{N} \mathbf{z}_i = 0, \quad \frac{1}{N} \mathbf{Z}\mathbf{Z}^\top = \mathbf{I}$$

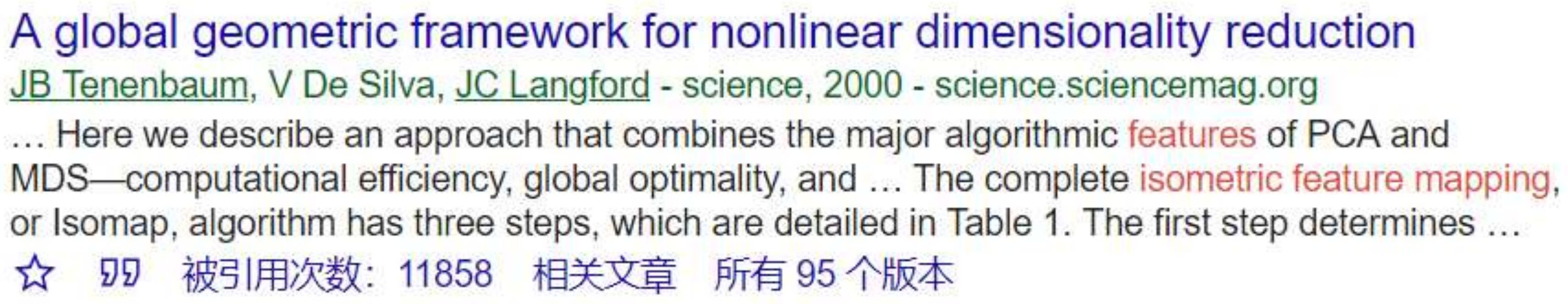
LLE: Examples

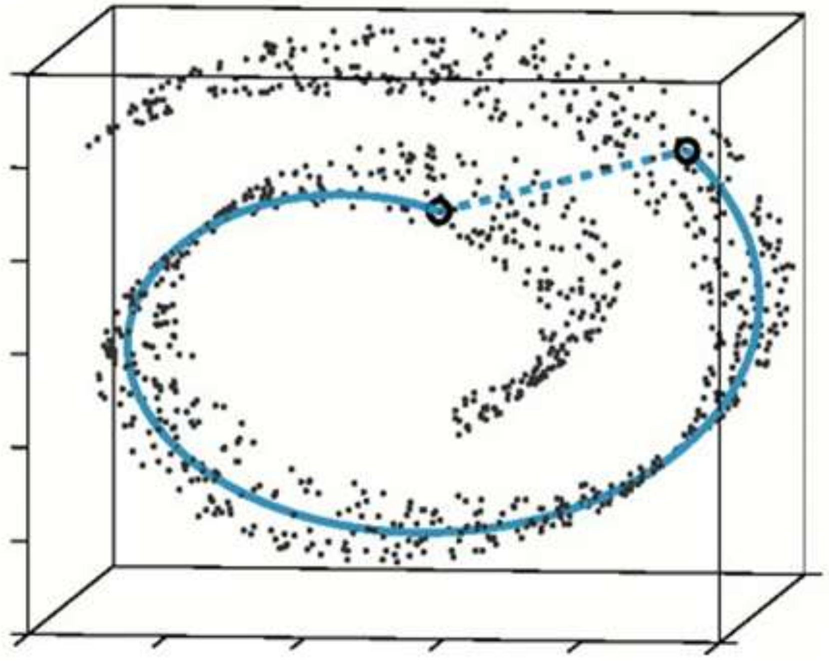


✓ ***Nonlinear dimension reduction***

✓ ***Out-of-sample problem***

Isometric Feature Mapping (ISOMAP)

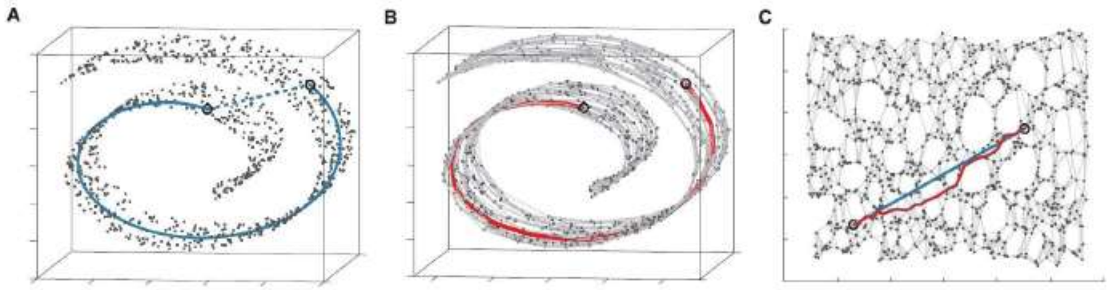

A global geometric framework for nonlinear dimensionality reduction
JB Tenenbaum, V De Silva, JC Langford - science, 2000 - science.sciencemag.org
… Here we describe an approach that combines the major algorithmic features of PCA and MDS—computational efficiency, global optimality, and … The complete isometric feature mapping, or Isomap, algorithm has three steps, which are detailed in Table 1. The first step determines …
☆ 99 被引用次数：11858 相关文章 所有 95 个版本



Isometric Feature Mapping (ISOMAP)

A graph based algorithm based on constructing a matrix of geodesic distances

- Identify the L nearest neighbors for each data point (just like LLE)
- Connect each point to all its neighbors (an edge for each neighbor)
- Assign weight to each edge based on the Euclidean distance
- Estimate the geodesic distance d_{ij} between any two data points i and j
  - Approximated by the sum of arc lengths along the shortest path between i and j in the graph (can be computed using Djikstras algorithm)
- Construct the $N \times N$ distance matrix $\mathbf{D} = \{d_{ij}^2\}$



Isometric Feature Mapping (ISOMAP)

- Use the distance matrix $\mathbf{D}$ to construct the Gram Matrix

$$\boxed{\mathbf{G} = -\frac{1}{2}\mathbf{HDH}}$$

(Refer to MDS Algorithm)

where $\mathbf{G}$ is $N \times N$ and

$$\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$$

$\mathbf{I}$ is $N \times N$ identity matrix, $\mathbf{1}$ is $N \times 1$ vector of 1s

- Do an eigen decomposition of $\mathbf{G}$
- Let the eigenvectors be $\{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ with eigenvalues $\{\lambda_1, \ldots, \lambda_N\}$
    - Each eigenvector $\mathbf{v}_i$ is N-dimensional: $\mathbf{v}_i = [v_{1i}, v_{2i}, \ldots, v_{Ni}]$
- Take the top K eigenvalue/eigenvectors
- The K dimensional embedding $\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iK}]$ of a point $\mathbf{x}_i$:

$$\boxed{z_{ik} = \sqrt{\lambda_k} v_{ki}}$$

Out-of-sample problem

Multi-Dimensional Scaling (MDS)

- Basic Idea
  - 把样本之间距离关系或不相似度关系在低维空间里生成对样本的表示
  - 把样本之间的距离关系或不相似关系在二维或三维空间里表示出来

Multi-Dimensional Scaling (MDS)

- 推导：
  - 有 n 个 d 维样本$x_i \in R^d, i = 1, \cdots, n, \boldsymbol{x}_i = \left[x_i^1, \cdots, x_i^d\right]^T$，所有样本组成 $n \times d$维矩阵$\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]^T$，样本两两内积矩阵$\boldsymbol{B} = \mathbf{X}\mathbf{X}^T$。样本$x_i$与$x_j$之间的欧式距离为

$$d_{ij}^2 = \left|\boldsymbol{x}_i - \boldsymbol{x}_j\right|^2 = |\boldsymbol{x}_i|^2 + \left|\boldsymbol{x}_j\right|^2 - 2\boldsymbol{x}_i\boldsymbol{x}_j^T$$

所有两两点之间欧式距离组成矩阵：

$$\boldsymbol{D}^{(2)} = \left\{d_{ij}^2\right\}_{n \times n} = \boldsymbol{c}\mathbf{1}^T + \mathbf{1}\boldsymbol{c}^T - 2\boldsymbol{B}$$

其中，$\boldsymbol{c}$是矩阵$\boldsymbol{B}$的对角线元素组成的向量，$\mathbf{1}$为单位列向量

$$c1^T=\begin{bmatrix}|\boldsymbol{x}_1|^2 & \cdots & |\boldsymbol{x}_1|^2\\ \vdots & \ddots & \vdots\\ |\boldsymbol{x}_n|^2 & \cdots & |\boldsymbol{x}_n|^2\end{bmatrix}$$

现已知矩阵$\boldsymbol{D}^{(2)}$，求 X

对坐标的平移不影响样本间距离，因此可假设所有样本的质心为坐标原点

$$\sum_{i=1}^{n}\boldsymbol{x}_i=0$$

定义中心化矩阵

$$\boldsymbol{J}=\boldsymbol{I}-\frac{1}{n}\mathbf{1}\mathbf{1}^T$$

其中，$\boldsymbol{I}$是单位对角阵。

显然

$$(\boldsymbol{c}\mathbf{1}^T)\boldsymbol{J} = \mathbf{0}$$

$$\boldsymbol{J}(\mathbf{1}\boldsymbol{c}^T) = \mathbf{0}$$

且有

$$\boldsymbol{J}\boldsymbol{X} = \boldsymbol{X}$$

$$\boldsymbol{J}\boldsymbol{B}\boldsymbol{J} = \boldsymbol{B}$$

对$\boldsymbol{D}^{(2)}$两边乘以中心化矩阵，得

$$\boldsymbol{J}\boldsymbol{D}^{(2)}\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{c}\mathbf{1}^T + 1\boldsymbol{c}^T - 2\boldsymbol{B})\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{c}\mathbf{1}^T)\boldsymbol{J} + \boldsymbol{J}(\mathbf{1}\boldsymbol{c}^T)\boldsymbol{J} - 2\boldsymbol{J}\boldsymbol{B}\boldsymbol{J}$$
$$= -2\boldsymbol{J}\boldsymbol{B}\boldsymbol{J} = -2\boldsymbol{B}$$

可得样本内积矩阵

$$\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^T = -\frac{1}{2}\boldsymbol{J}\boldsymbol{D}\boldsymbol{J}$$

这种做法又称作双中心化（double centering）

如果$\boldsymbol{D}^{(2)}$由欧式距离组成，则$\boldsymbol{B}$为对称矩阵，可用奇异值分解来求解$\boldsymbol{X}$

$$\boldsymbol{B} = \boldsymbol{U\Lambda}\mathbf{U}^{\mathrm{T}}$$

其中，U是由矩阵B的本征向量组成的矩阵，Λ是以B的本征值为对角元素的对角阵

$$\boldsymbol{X} = \boldsymbol{U\Lambda}^{1/2}$$

如果样本不是中心化的，则只要知道样本的均值向量$\bar{\boldsymbol{x}}$就知道各个样本原始坐标

$$\widetilde{\boldsymbol{x}_i} = \boldsymbol{x}_i + \bar{\boldsymbol{x}}, \qquad i = 1,\cdots,n$$

如果要用$k < d$维空间来表示这些样本，则可按照本征值从大到小排序

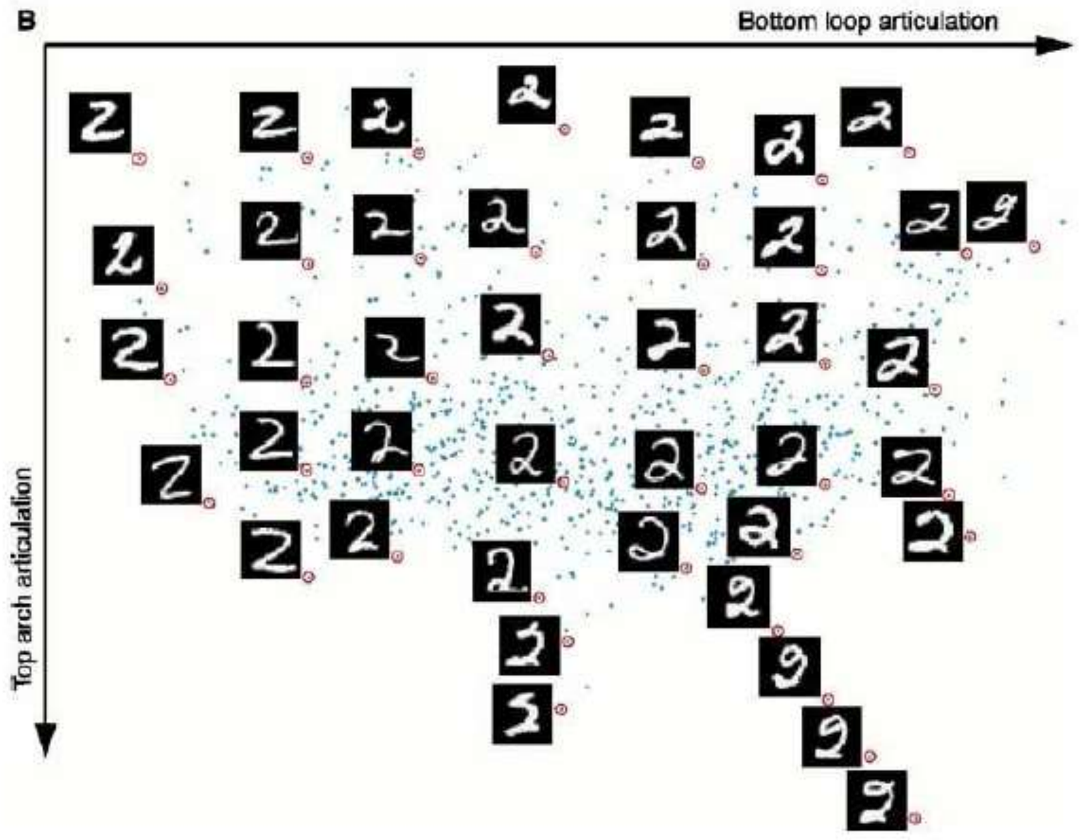
$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq \cdots \geq \lambda_d$$

用$\lambda_1 \sim \lambda_k$组成$\boldsymbol{\Lambda}$，只用这些本征值对应的本征向量组成$\boldsymbol{U}$。

易证，如果已知样本集，从中计算出$\boldsymbol{D}^{(2)}$，再用古典尺度法得到$\boldsymbol{X}$的低维表示，结果与主成分分析相同。Why?

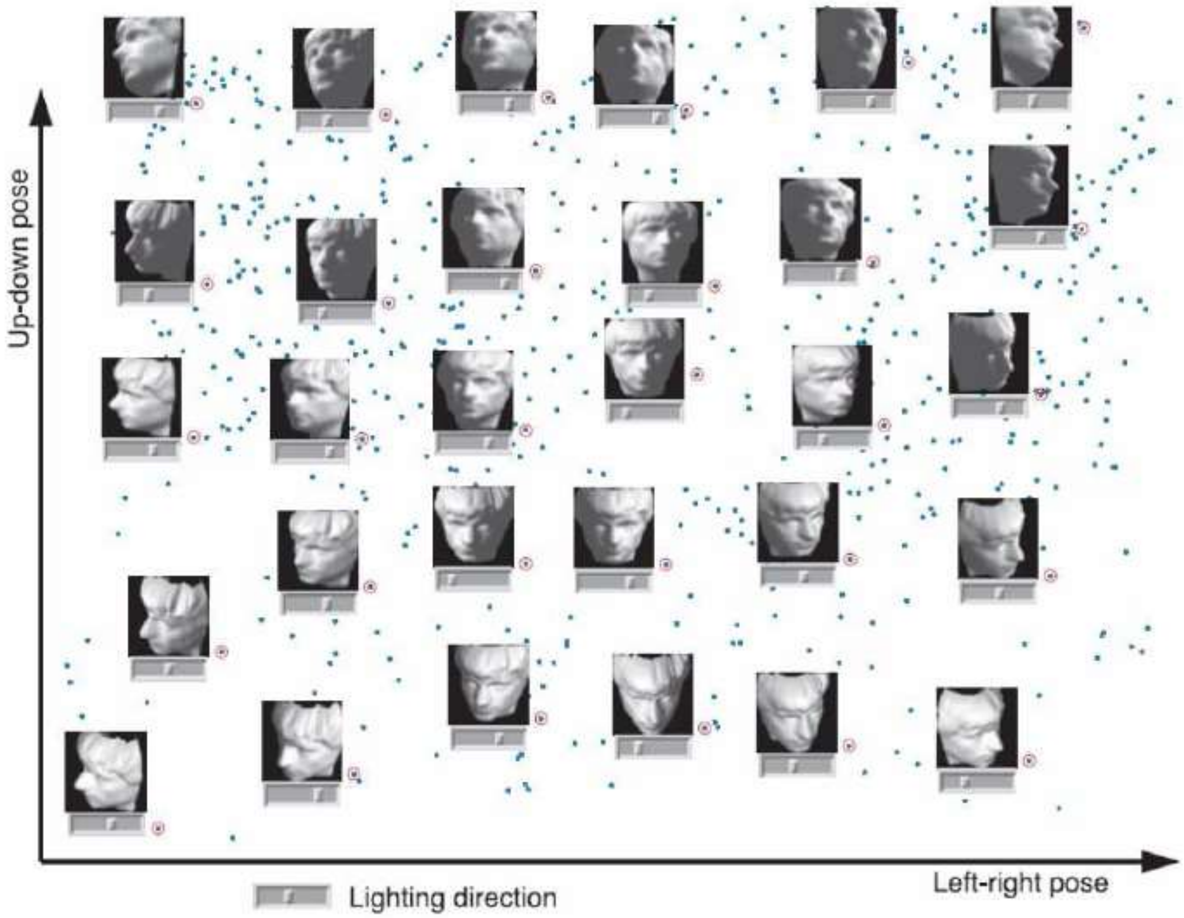
ISOMAP: Example

Digit images projected down to 2 dimensions



ISOMAP: Example

Face images with varying poses



Why manifold learning?

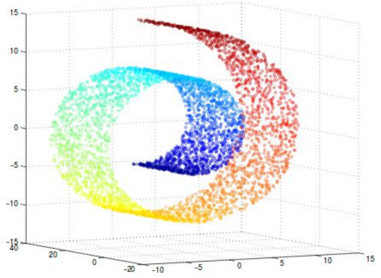
Why PCA fails to properly reduce dimensions of MNIST?

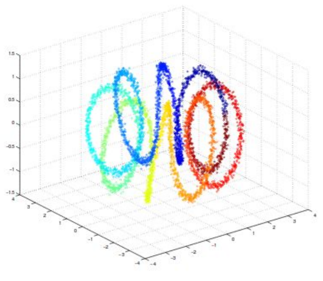
- PCA is good, but it is a linear algorithm, meaning that it cannot represent complex relationship between features

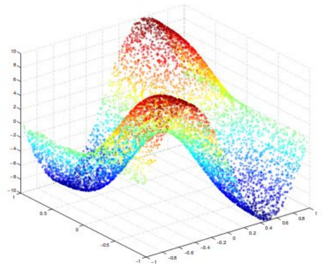
t-SNE is non-linear dimensionality reduction technique that has better performance. It is designed for visualization purposes.

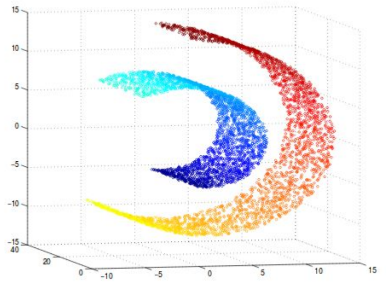
Why not use Neural Networks?

- There is a dimensionality reduction technique based on Neural Network called Autoencoder!


(a) Swiss roll dataset.


(b) Helix dataset.


(c) Twinpeaks dataset.


(d) Broken Swiss roll dataset.

Good visualization

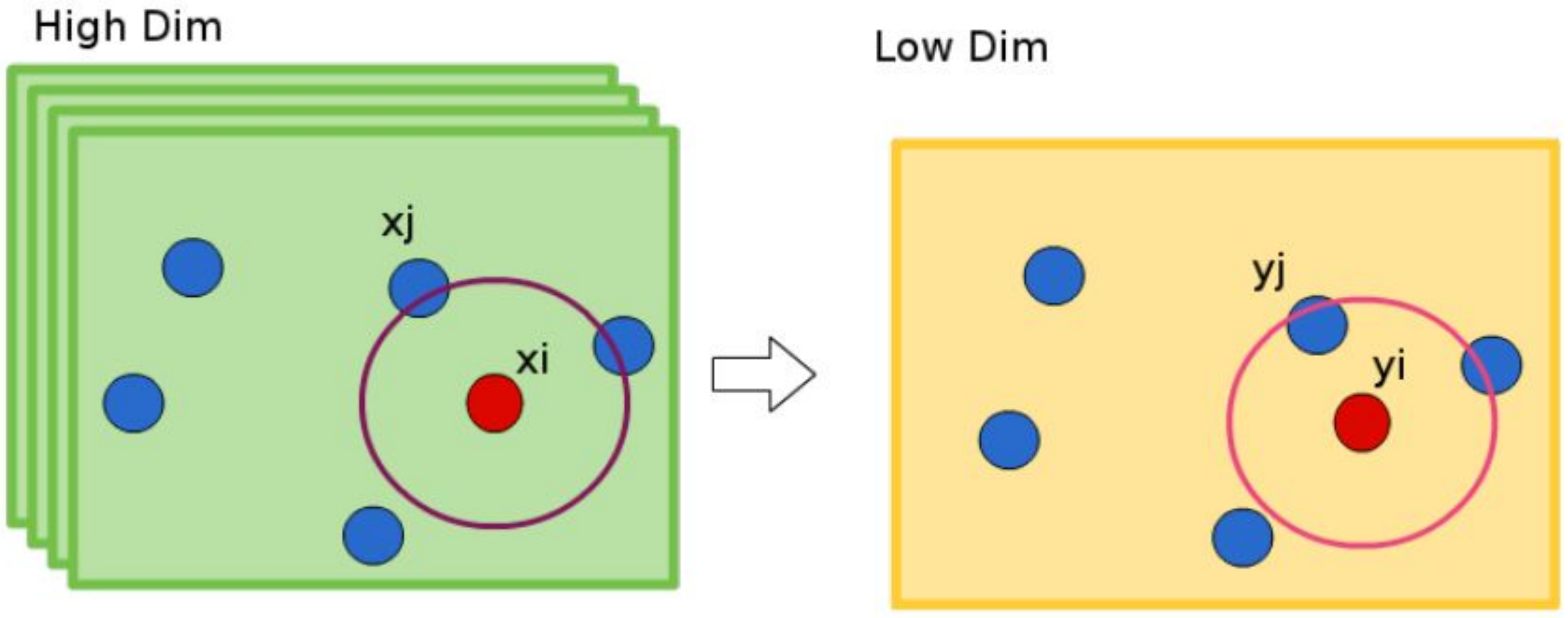
Patterns

- Discover natural clusters
- Linear relationships
- Visualize embeddings

Technical Requirements

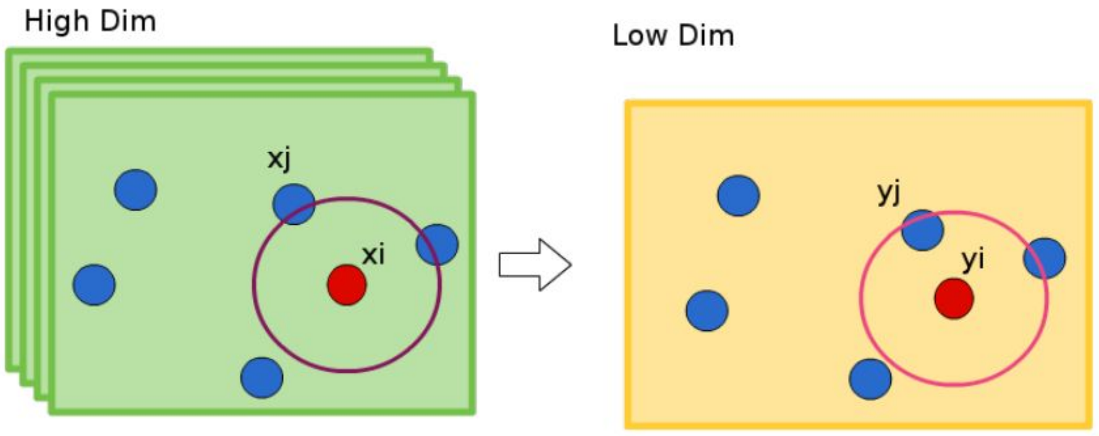
- Each high dimensional object is represented by a low-dimensional object
- Preserve the neighborhood
- Distant points correspond to dissimilar objects
- Scalability: large, high-dimensional data sets

Underlying idea of t-SNE



Stochastic Neighbor Embedding

Measure pairwise similarities between high-dimensional and low-dimensonal objects



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

Stochastic Neighbor Embedding

Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities

- Similarity of datapoints in High Dimension

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

- Similarity of datapoints in Low Dimension

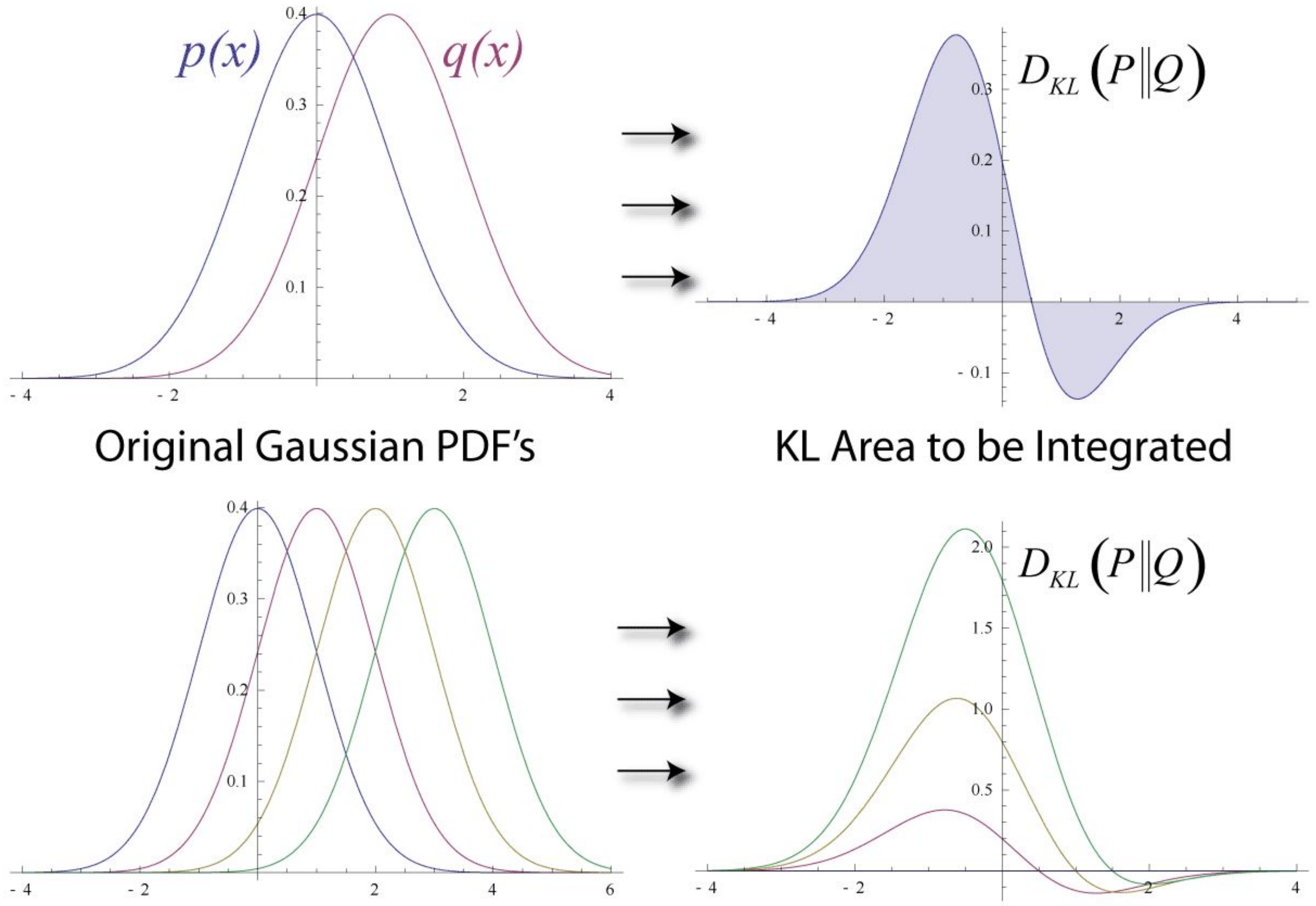
$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

- Cost function

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

KL Divergence

Measures the similarity between two probability distributions & it is asymmetric



$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\,\log\frac{P(i)}{Q(i)}.$$

Stochastic Neighbor Embedding

Gradient has a surprisingly simple form

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

The gradient update with momentum term is given by

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial y_i} + \beta(t)(Y^{(t-1)} - Y^{(t-2)})$$

Derivation of Gradient is given in paper [1]

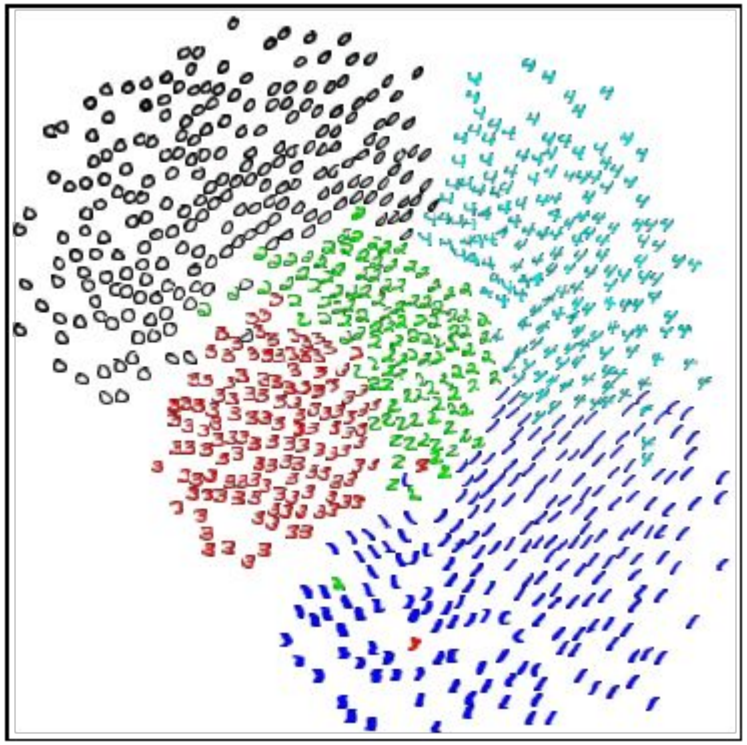
Stochastic Neighbor Embedding

The result of running the SNE algorithm on 3000 256-dimensional grayscale images of handwritten digits.

Pictures of the original data vectors xi (scans of handwritten digit) are shown at the location corresponding to their low-dimensional images yi as found by SNE.

The classes are quite well separated even though SNE had no information about class labels. Furthermore, within each class, properties like orientation, skew and strokethickness tend to vary smoothly across the space.

Not all points are shown: to produce this display, digits are chosen in random order and are only displayed if a 16 x 16 region of the display centered on the 2-D location of the digit in the embedding does not overlap any of the 16 x16 regions for digits that have already been displayed.



Symmetric SNE

- Minimize the sum of the KL divergences between the conditional probabilities

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

- Minimize a single KL divergence between a joint probability distribution

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

- The obvious way to redefine the pairwise similarities is

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

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

Symmetric SNE

Such that $p_{ij} = p_{ji}, q_{ij} = q_{ji}$, the main advantage is simplifing the gradient

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

However, in practice we symmetrize (or average) the conditionals

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

Set the bandwidth σ_i such that the conditional has a fixed perplexity (effective number of neighbors) $Perp(P_i) = 2^{H(P_i)}$, typical value is about 5 to 50

t-Distribution

Use heavier tail distribution than Gaussian in low-dim space, we choose

$$q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$$

Then the gradient could be

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

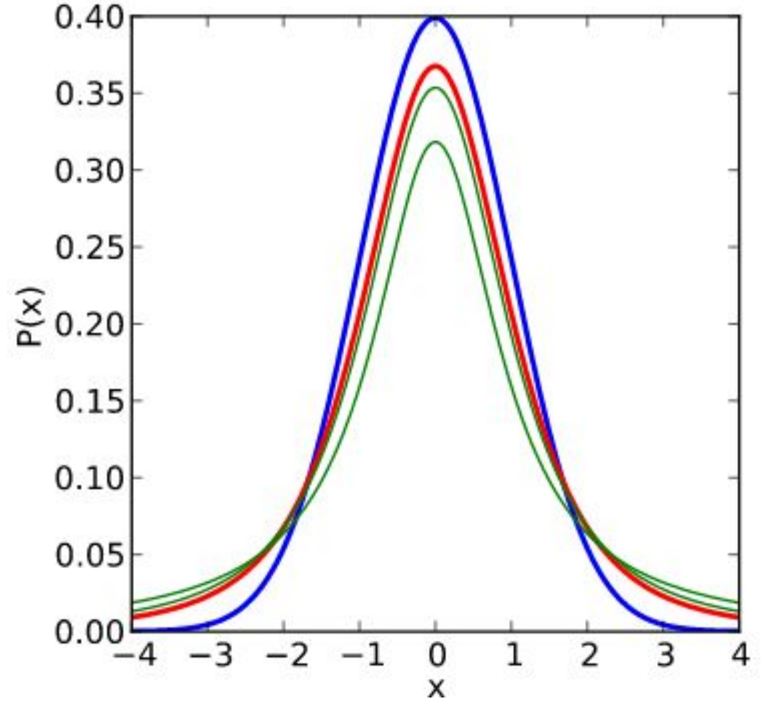
Why Student-t Distribution?

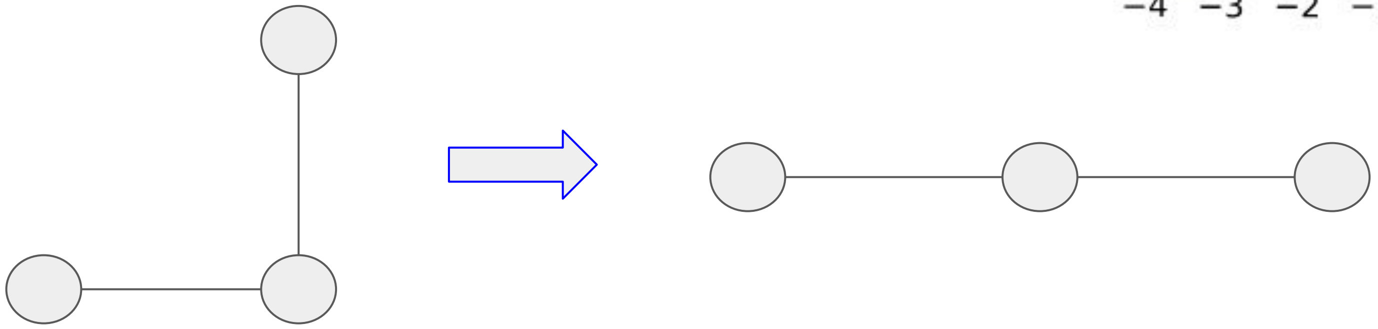
Why do we define map similarities as $q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$

Suppose data is intrinsically high dimensional

We try to model the local structure of this data in the map

Result: Dissimilar points have to be modeled as too far apart in the map!





t-Distributed Stochastic Neighbor Embedding

- Similarity of datapoints in High Dimension

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

- Similarity of datapoints in Low Dimension

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

t-Distributed Stochastic Neighbor Embedding

- Cost function

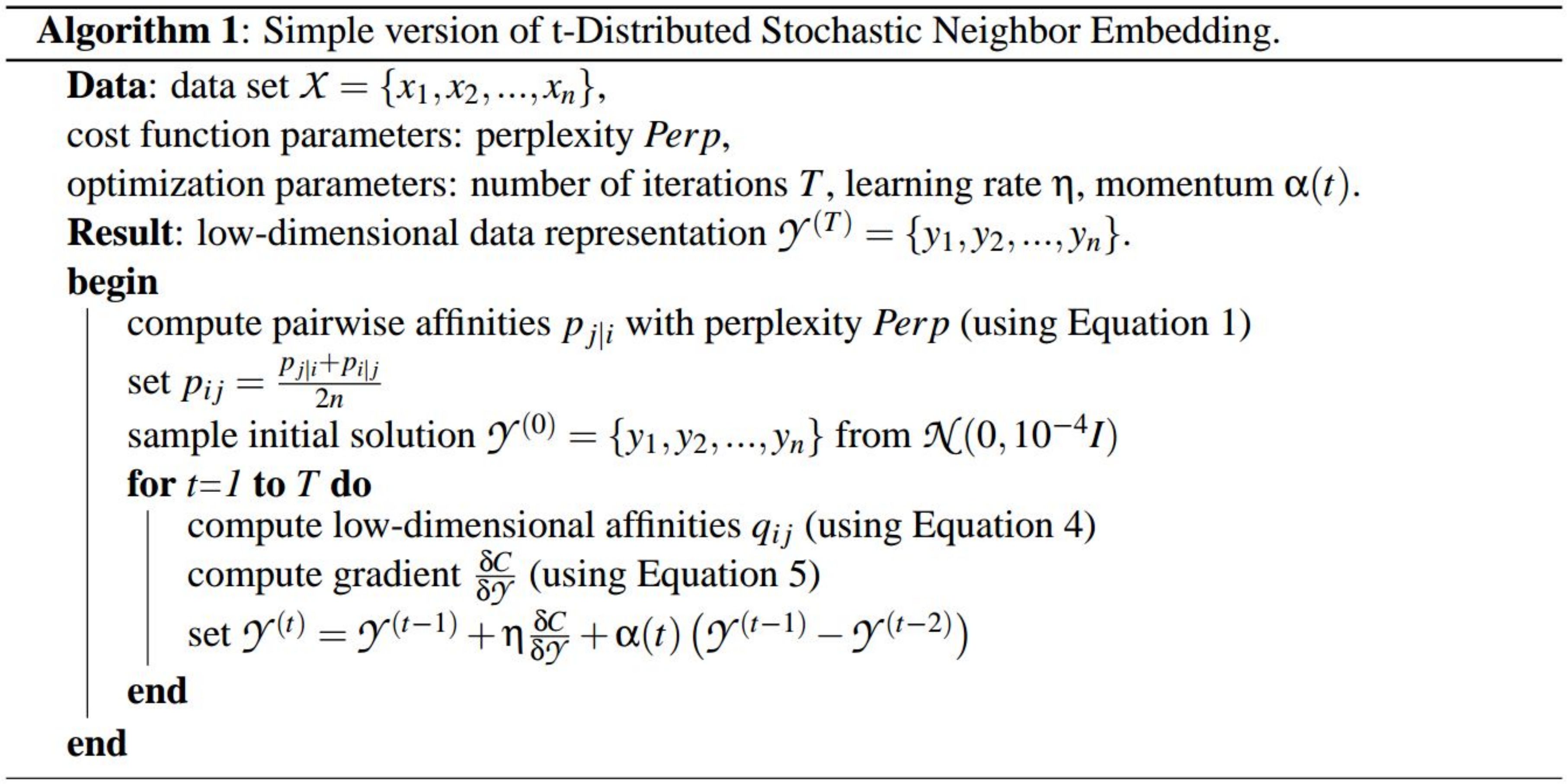
$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

  - Large p_{ij} modeled by small q_{ij}: Large penalty
  - Small p_{ij} modeled by large q_{ij}: Small penalty
  - t-SNE mainly preserves local similarity structure of the data

- Gradient

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

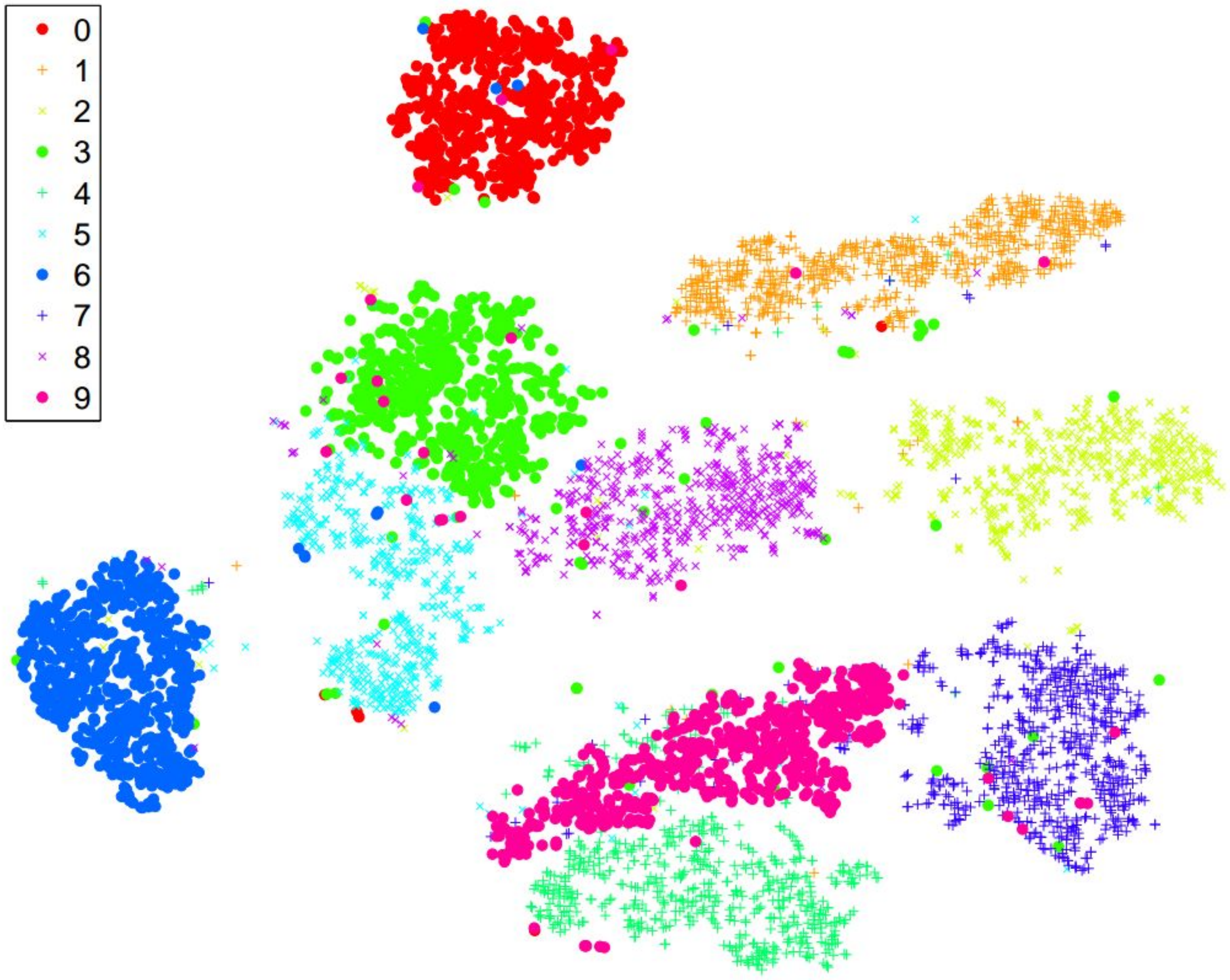
t-SNE Algorithm



Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
    for $t$=1 to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left( \mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end
```

Results: MNIST



References

1. “Visualizing Data using t-SNE”, L. Maaten, et. al.
2. “Stochastic Neighbor Embedding”, G. Hinton, et. al.
3. t-SNE implementations and other resources - https://lvdmaaten.github.io/tsne/
4. “A Tutorial on Principal Component Analysis”, J. Shlens
5. “Dimensionality Reduction: A Comparative Review”, L. Maaten
6. Google Tech Talks: https://youtu.be/RJVL80Gg3lA