Nonlinear Dimensionality Reduction and Euclidean Embedding

Outline

- Euclidean Embedding
- Dimensionality Reduction vs Euclidean Embedding
- Multidimensional Scaling
- Kernel PCA
- ISOMAP
- Laplacian Eigenmaps

Euclidean Embedding

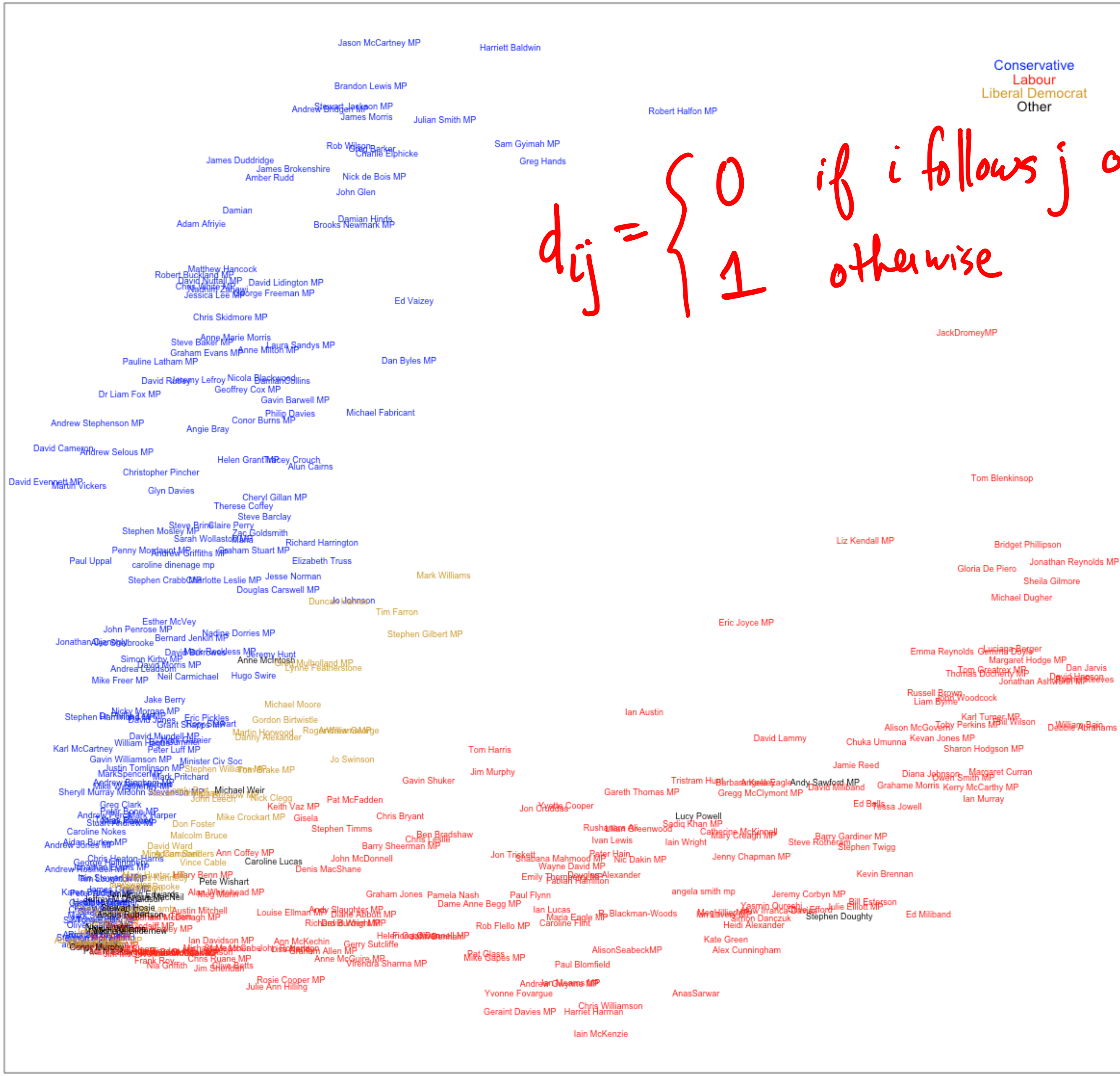
- A dissimilarity matrix is a square, $n \times n$ matrix $\boldsymbol{D} = [d_{ij}]$ such that
  - $d_{ij} \geq 0$
  - $d_{ij} = d_{ji}$
  - $d_{ii} = 0$
- Given: n objects (not necessarily Euclidean vectors) with dissimilarity matrix $\boldsymbol{D}$, find Euclidean vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \in \mathbb{R}^p$ such that

$$d_{ij} \approx \| y_i - y_j \|_2$$

- Uses?

1) visualization / exploratory data analysis $(p = 2 \text{ or } 3)$
2) apply an algorithm that works on Euclidean data

Two dimensional clustering of UK Members of Parliament



$$d_{ij} = \begin{cases} 0 & \text{if } i \text{ follows } j \text{ or } j \text{ follows } i \\ 1 & \text{otherwise} \end{cases}$$

Based on Twitter follows by UK Members of Parliament, Aug 2013. http://andrewwhitby.com/2013/08/21/clustering-of-uk-mps/

- Two dimensional Euclidean embedding of UK members of Parliament
- Dissimilarity based on Twitter follows
- Algorithm is classical MDS
- Color indicates party affiliation (not used by algorithm)

Euclidean Embedding vs Dimensionality Reduction

- Differences:
  - DR starts with a high dimensional data matrix $X \in \mathbb{R}^{d \times n}$
  - EE: Starts with a dissimilarity matrix
  - EE methods apply to non Euclidean data

- Similarities:

  Same goal: low dimensional representation of data

- An EE method can be used for DR, but not vice versa

Euclidean Distance Matrices

- An $n \times n$ dissimilarity matrix $\boldsymbol{D}$ is called a *Euclidean distance matrix* if there exists p and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \in \mathbb{R}^p$ such that

$$d_{ij} = \|y_i - y_j\|_2$$

for all i, j.

- **Theorem (Part 1):** Let $\boldsymbol{D}$ be an $n \times n$ dissimilarity matrix. Set $\boldsymbol{B} = \boldsymbol{H}\boldsymbol{A}\boldsymbol{H}$ where

$$\boldsymbol{A} = [a_{ij}], \quad a_{ij} = -d_{ij}^2, \quad \boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

Then $\boldsymbol{D}$ is a Euclidean distance matrix iff B is PSD.

Euclidean Distance Matrices

- **Theorem (Part 2):** If $\boldsymbol{B}$ is PSD with positive eigenvalues $\lambda_1 > \cdots > \lambda_p$ and corresponding eigenvectors

$$\boldsymbol{u}_1 = \begin{bmatrix} u_{11} \\ \vdots \\ u_{1n} \end{bmatrix}, \ldots, \boldsymbol{u}_p = \begin{bmatrix} u_{p1} \\ \vdots \\ u_{pn} \end{bmatrix}$$

normalized such that $u_k^T u_k = \lambda_k$

then the vectors $y_i = (u_{1i}, \ldots, u_{pi})^T \in \mathbb{R}^p$

satisfy

$$d_{ij} = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|$$

- Proof: Mardia, Kent, and Bibby, *Multivariate Analysis*, 1979.

Classical MDS

- Even if $\boldsymbol{D}$ is not a Euclidean distance matrix, the previous result suggests an algorithm for MDS:
- Input: $\boldsymbol{D}$, p
    - Form $\boldsymbol{B}$ as in the theorem
    - Compute the eigenvalue decompositon

$$\boldsymbol{B} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T, \quad (n \times n)$$

    where $\boldsymbol{V} = [\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n]$, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$
    - Set $\boldsymbol{u}_k = \sqrt{\lambda_k}\boldsymbol{v}_k$

  Return $\boldsymbol{y}_i = i^{\text{th}}$ row of

$$\boldsymbol{U}_p = [\boldsymbol{u}_1 \ \cdots \ \boldsymbol{u}_p] \quad (n \times p).$$

- True (A) or False (B): The above procedure is valid for any dissimilarity matrix $\boldsymbol{D}$

Stress-Based MDS

- A second approach to MDS is to minimize the *stress* objective function

$$\min_{y_1,\dots,y_n \in \mathbb{R}^p} \sum_{1 \le i < j \le n} w_{ij} \left(d_{ij} - \| y_i - y_j \| \right)^2$$

  where $w_{ij} \geq 0$ are weights reflecting the similarity of the data points

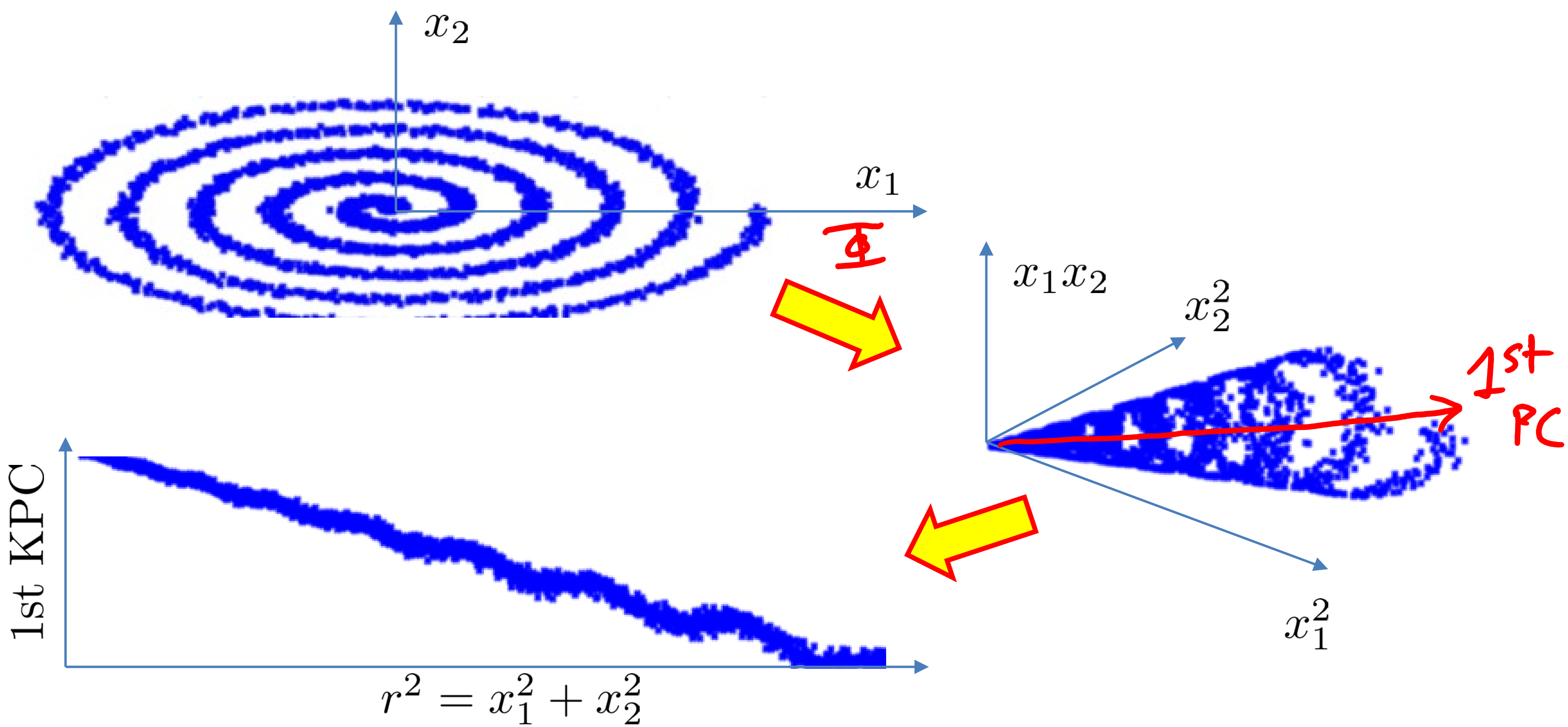
- Examples: $w_{ij} = 1$ or $w_{ij} = d_{ij}^{-\alpha}$, $\alpha > 0$.

- This problem is nonconvex, but a local minimizer can be obtained efficiently with a majorize-minimize algorithm.

Demo

Kernel PCA

- Intuitively, you can think of this like any other kernel method: First apply a nonlinear feature map $\boldsymbol{\Phi}$ (associated to a SPD kernel), and then do PCA in the new feature space.
- KPCA is a good general purpose method for dimensionality reduction



KPCA Algorithm

Input: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, dimension p , kernel k

$[K]_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle$

- Form centered kernel matrix

$$\tilde{K} = K - OK - KO + OKO$$

$[\tilde{K}]_{ij} = \langle \tilde{\Phi}(x_i), \tilde{\Phi}(x_j) \rangle$

where O is $n \times n$ and has entries equal to $\frac{1}{n}$

where

$\tilde{\Phi}(x) = \Phi(x) - \frac{1}{n}\sum_{i=1}^{n} \Phi(x_i)$

- Compute the eigenvalue decomposition

$$\tilde{K} = U \Lambda U^T$$
$$U = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n \end{bmatrix}$$
$$\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_m, 0, \cdots, 0)$$

- Set $\boldsymbol{\alpha}_j = (\alpha_{j1}, \ldots, \alpha_{jn})^T := \frac{1}{\sqrt{\lambda_j}} \boldsymbol{u}_j \in \mathbb{R}^n, \quad 1 \leq j \leq p$

Output: Dimensionality reduction mapping

$$\boldsymbol{x} \mapsto \boldsymbol{y} = (y_1, \ldots, y_p)^T \in \mathbb{R}^p$$

where

$\tilde{k}(x, x') = \langle \tilde{\Phi}(x), \tilde{\Phi}(x') \rangle$

$$y_j = \sum_{i=1}^{n} \alpha_{ji} \tilde{k}(\boldsymbol{x}, \boldsymbol{x}_i)$$

KPCA Algorithm

Input: $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, dimension p

- Form centered kernel matrix

$$\tilde{K} = K - OK - KO + OKO$$

where O is $n \times n$ and has entries equal to $\frac{1}{n}$

- Compute the eigenvalue decomposition

$$\tilde{K} = U \Lambda U^T$$
$$U = \begin{bmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n \end{bmatrix}$$
$$\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_m, 0, \cdots, 0)$$

- Set $\boldsymbol{\alpha}_j = (\alpha_{j1}, \ldots, \alpha_{jn})^T := \frac{1}{\sqrt{\lambda_j}} \boldsymbol{u}_j \in \mathbb{R}^n, \quad 1 \le j \le p$

Output: Dimensionality reduction mapping

$$\boldsymbol{x} \mapsto \boldsymbol{y} = (y_1, \ldots, y_p)^T \in \mathbb{R}^p$$

where

$$y_j = \sum_{i=1}^{n} \alpha_{ji} \tilde{k}(\boldsymbol{x}, \boldsymbol{x}_i)$$

True (A) or False (B): This procedure is valid as long as p <= m, the number of nonzero eigenvalues of the centered kernel matrix.

Isomap

- KPCA does not try to capture the *intrinsic dimension* of the data
- A method that does is Isomap (isometric mapping)
- The algorithm is simple:
  - Input $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$
  - Construct a similarity graph such as a k-nearest neighbor graph
  - Form a dissimilarity matrix $\boldsymbol{D} = [d_{ij}]$ where d_{ij} = length of shortest path connecting $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$
  - Apply MDS to $\boldsymbol{D}$ to get an embedding $\boldsymbol{y}, \ldots, \boldsymbol{y}_n$
- Tenenbaum, de Silva and Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, 290, 2319-2323 (2000)

Isomap

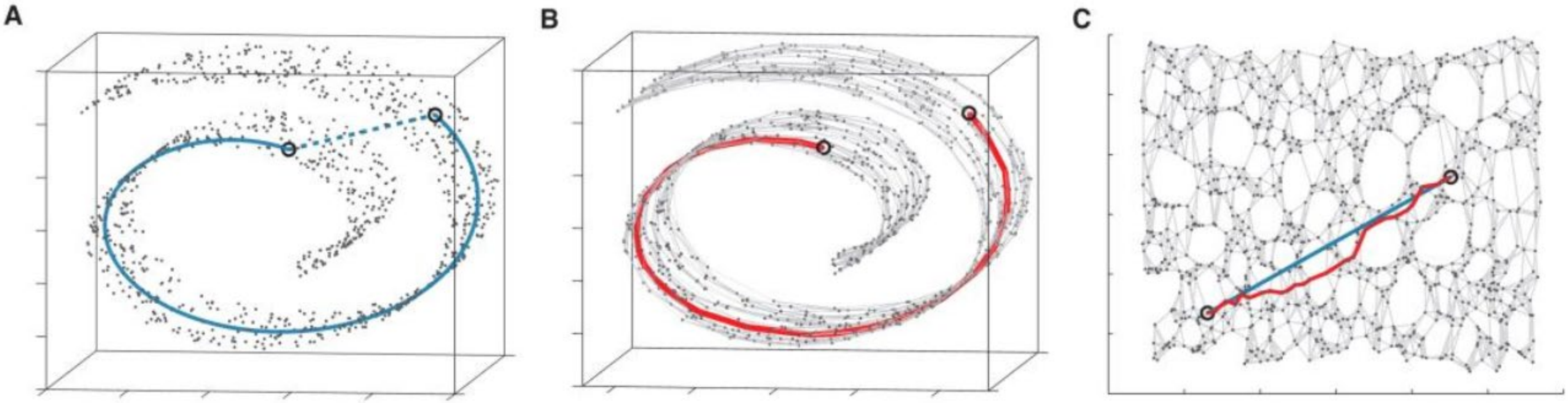


Fig. 3. The "Swiss roll" data set, illustrating how Isomap exploits geodesic paths for nonlinear dimensionality reduction. (**A**) For two arbitrary points (circled) on a nonlinear manifold, their Euclidean distance in the high-dimensional input space (length of dashed line) may not accurately reflect their intrinsic similarity, as measured by geodesic distance along the low-dimensional manifold (length of solid curve). (**B**) The neighborhood graph G constructed in step one of Isomap (with $K = 7$ and $N = 1000$ data points) allows an approximation (red segments) to the true geodesic path to be computed efficiently in step two, as the shortest path in G. (**C**) The two-dimensional embedding recovered by Isomap in step three, which best preserves the shortest path distances in the neighborhood graph (overlaid). Straight lines in the embedding (blue) now represent simpler and cleaner approximations to the true geodesic paths than do the corresponding graph paths (red).

Isomap

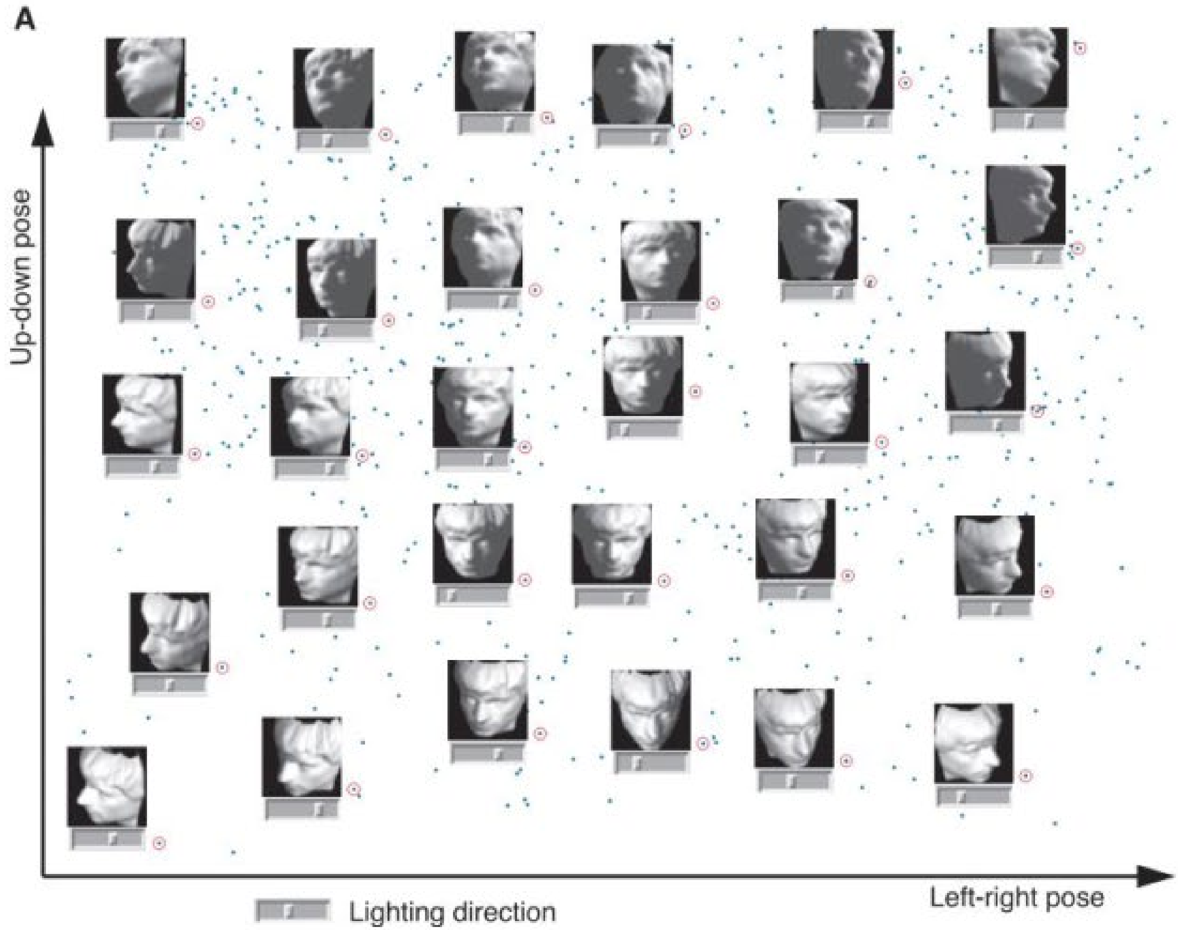


Fig. 1. (**A**) A canonical dimensionality reduction problem from visual perception. The input consists of a sequence of 4096-dimensional vectors, representing the brightness values of 64 pixel by 64 pixel images of a face rendered with different poses and lighting directions. Applied to $N = 698$ raw images, Isomap ($K = 6$) learns a three-dimensional embedding of the data's intrinsic geometric structure. A two-dimensional projection is shown, with a sample of the original input images (red circles) superimposed on all the data points (blue) and horizontal sliders (under the images) representing the third dimension. Each coordinate axis of the embedding correlates highly with one degree of freedom underlying the original data: left-right pose (x axis, $R = 0.99$), up-down pose (y axis, $R = 0.90$), and lighting direction (slider position, $R = 0.92$). The input-space distances $d_X(i,j)$ given to Isomap were Euclidean distances between the 4096-dimensional image vectors.

Laplacian Eigenmaps

- This is the Euclidean embedding method implicit in spectral clustering
- Discussed in spectral clustering notes
- Can be derived independent from spectral clustering, as follows:
- Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be training data, not necessarily Euclidean vectors
- Let $\boldsymbol{W} = [w_{ij}]$ be a similarity graph as in spectral clustering
- Idea: select $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \in \mathbb{R}^p$ to optimize

$$\min_{y_1, \ldots, y_n} \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \| y_i - y_j \|^2 = tr(YLY^T)$$

$$Y = [y_1 \cdots y_n]$$

- To avoid a trivial solution, need to impose an energy constraint. Options:

$$YY^T = I_p \quad , \quad YDY^T = I_p$$

Laplacian Eigenmaps

- The solution to

$$\min_{\boldsymbol{Y}} \ \mathrm{tr}(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^T)$$
$$\text{s.t. } \boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{I}$$

  is

  where $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$ are the p smallest eigenvectors of

- If the graph is connected, then $\boldsymbol{u}_1$ is a multiple of $\mathbf{1}$ and we can equivalently solve

  which determines a dimensional embedding

Laplacian Eigenmaps

- The solution to

$$\min_{\boldsymbol{Y}} \ \mathrm{tr}(\boldsymbol{Y}\boldsymbol{L}\boldsymbol{Y}^T)$$
$$\text{s.t. } \boldsymbol{Y}\boldsymbol{D}\boldsymbol{Y}^T = \boldsymbol{I}$$

  is

  where $\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_p$ are the p smallest eigenvectors of

- If the graph is connected, then $\boldsymbol{u}_1$ is a multiple of $\mathbf{1}$ and we can equivalently solve

  which again determines a $p-1$ dimensional embedding

Laplacian Eigenmaps

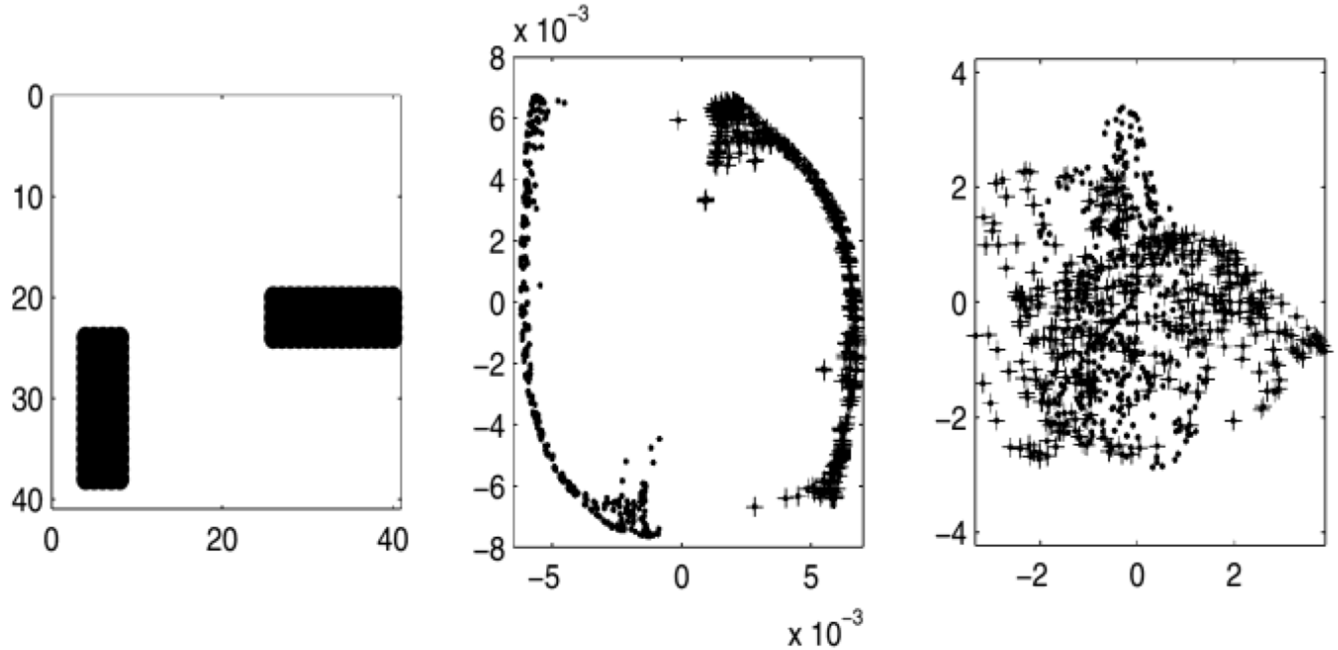


Figure 3: (Left) A horizontal and a vertical bar. (Middle) A two-dimensional representation of the set of all images using the Laplacian eigenmaps. (Right) The result of PCA using the first two principal directions to represent the data. Blue dots correspond to images of vertical bars, and plus signs correspond to images of horizontal bars.

t-SNE

- Given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$
- For all $i \neq j$, compute

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

- Set $p_{ij} = (p_{i|j} + p_{j|i})/2n$. These values are nonnegative, symmetric, and sum to 1 (summing over both i and j) $\longrightarrow$ probability distribution P
- For candidate embedded points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$, define a probability distribution Q analogously but using

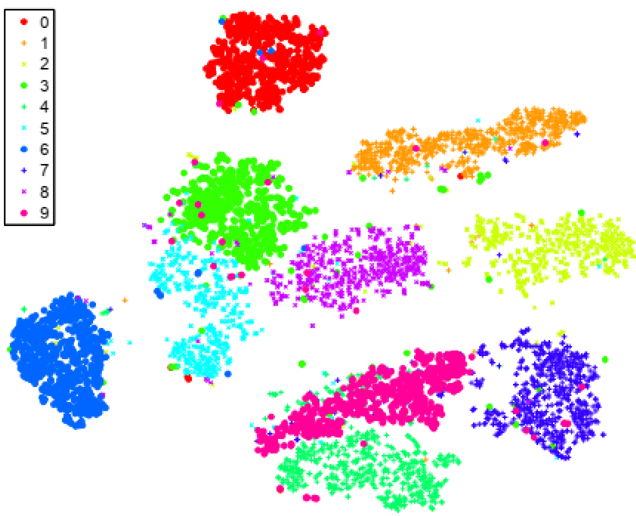
$$q_{i|j} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\boldsymbol{y}_i - \boldsymbol{y}_k\|^2)^{-1}}$$

- Optimize $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ by minimizing the Kullback-Liebler divergence, $KL(P\|Q) = \sum_{ij} p_{ij} \log(p_{ij}/q_{ij})$ using gradient descent
- Remarks: data-dependent bandwidths, name, avoid crowding

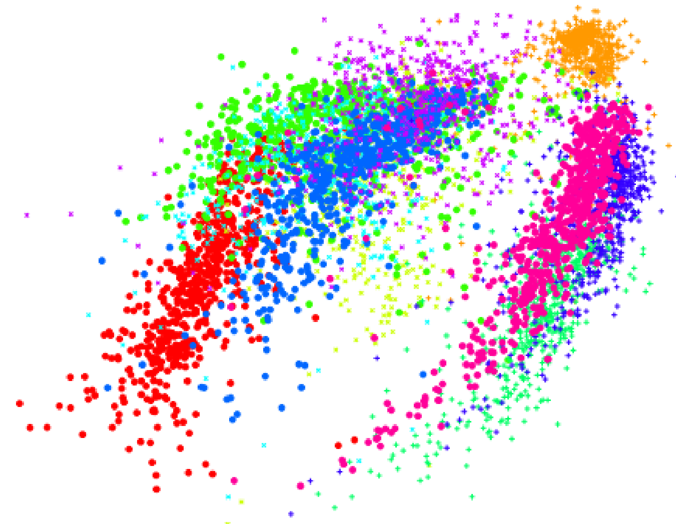
t-SNE avoids crowding

t-SNE Illustration

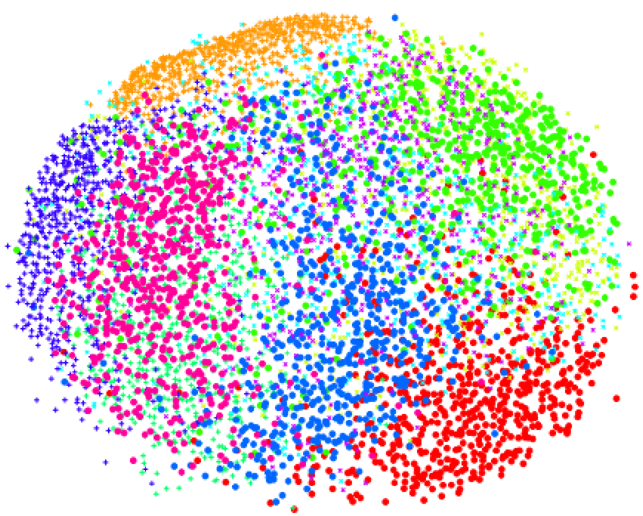
Van der Maaten and Hinton, 2008



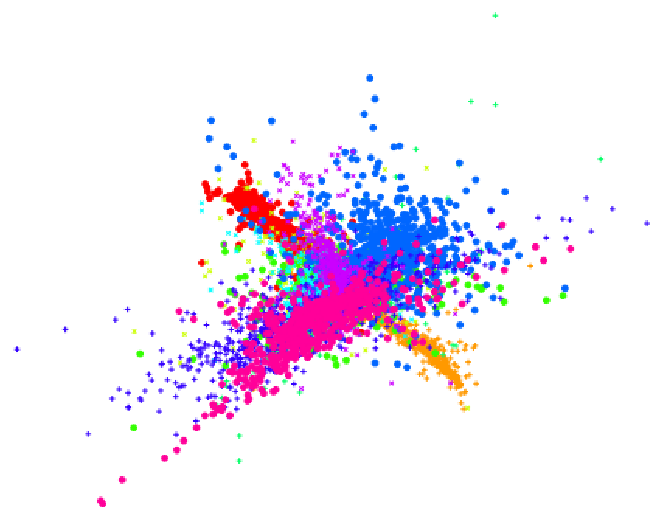
(a) Visualization by t-SNE.



(a) Visualization by Isomap.



(b) Visualization by Sammon mapping.



t-SNE Demo

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

Summary

- DR and EE: Two classes of methods for learning a low-dimensional Euclidean representation of data
- Methods differ in whether they emphasize preservation of local or global distances
- Some methods have natural out-of-sample extensions, others do not
- Some methods aim to capture intrinsic dimension (manifold learning), some do not
- t-SNE and UMAP (not covered) are quite popular. These use gradient descent to iteratively optimize some nonconvex objective. Initialization is important, e.g., by LEM, which globally optimizes its objective.
    - https://umap-learn.readthedocs.io/en/latest/how_umap_works.html
    - https://arxiv.org/abs/1802.03426
    - https://arxiv.org/abs/2009.12981

Autoencoder