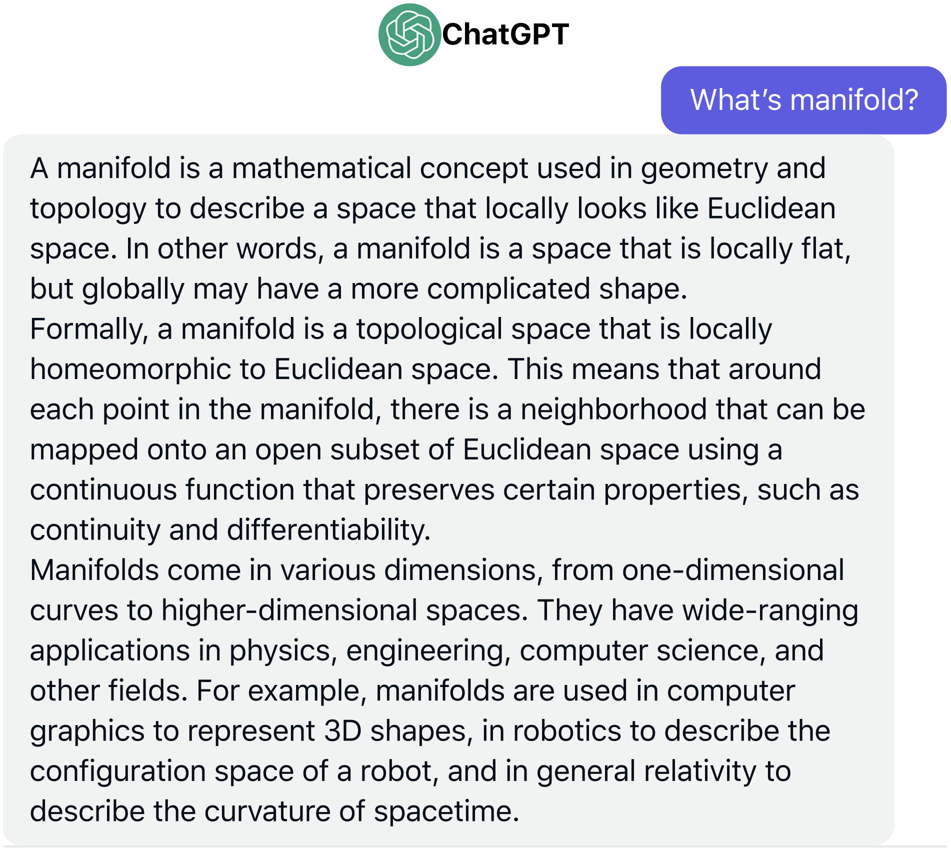
Manifold Learning II: Laplacian, Diffusion, and Stochastic Neighbor Embedding

姚 遠

2023



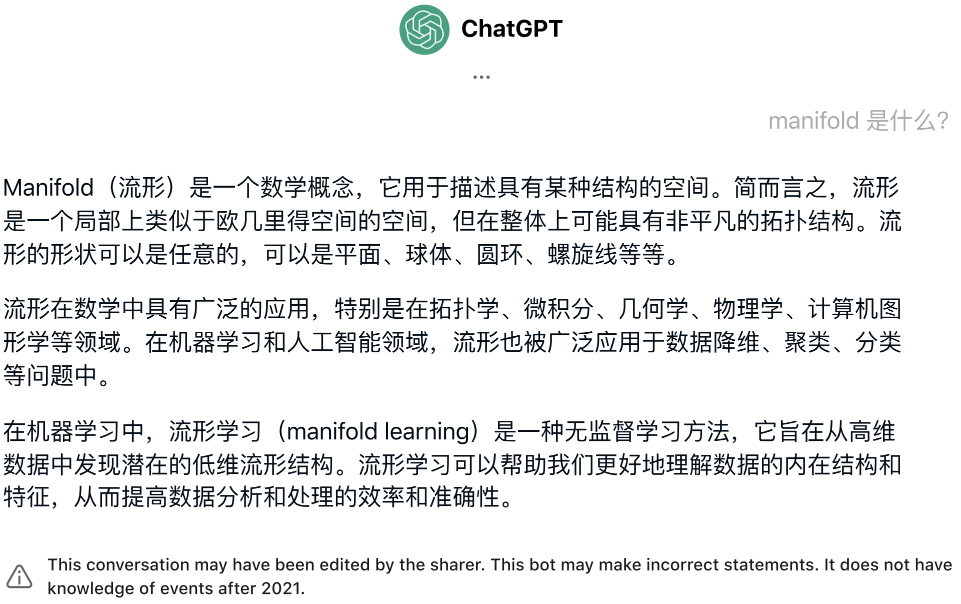
ChatGPT

What’s manifold?

A manifold is a mathematical concept used in geometry and topology to describe a space that locally looks like Euclidean space. In other words, a manifold is a space that is locally flat, but globally may have a more complicated shape.

Formally, a manifold is a topological space that is locally homeomorphic to Euclidean space. This means that around each point in the manifold, there is a neighborhood that can be mapped onto an open subset of Euclidean space using a continuous function that preserves certain properties, such as continuity and differentiability.

Manifolds come in various dimensions, from one-dimensional curves to higher-dimensional spaces. They have wide-ranging applications in physics, engineering, computer science, and other fields. For example, manifolds are used in computer graphics to represent 3D shapes, in robotics to describe the configuration space of a robot, and in general relativity to describe the curvature of spacetime.



ChatGPT

...

manifold 是什么?

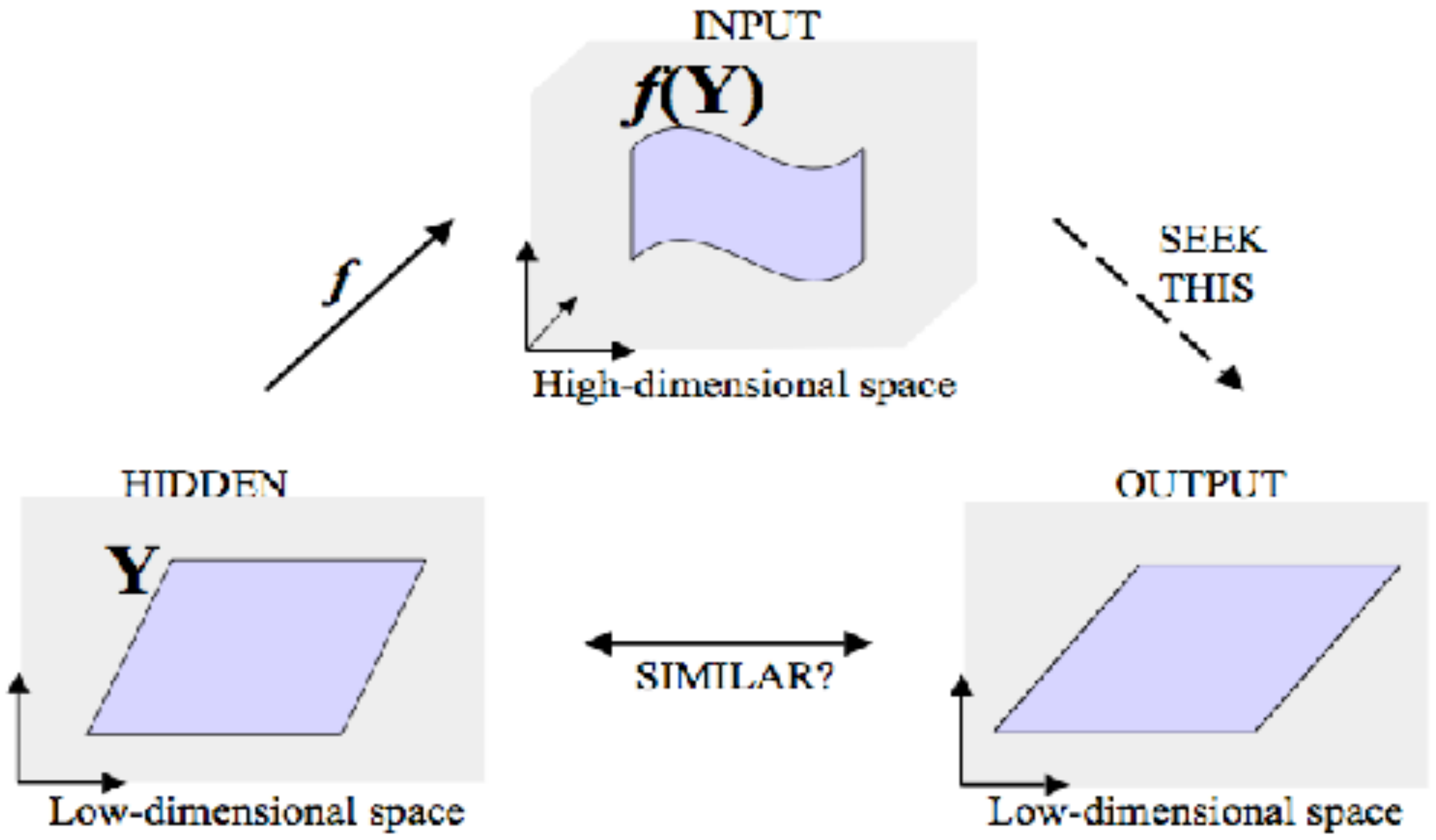
Manifold（流形）是一个数学概念，它用于描述具有某种结构的空间。简而言之，流形是一个局部上类似于欧几里得空间的空间，但在整体上可能具有非平凡的拓扑结构。流形的形状可以是任意的，可以是平面、球体、圆环、螺旋线等等。

流形在数学中具有广泛的应用，特别是在拓扑学、微积分、几何学、物理学、计算机图形学等领域。在机器学习和人工智能领域，流形也被广泛应用于数据降维、聚类、分类等问题中。

在机器学习中，流形学习（manifold learning）是一种无监督学习方法，它旨在从高维数据中发现潜在的低维流形结构。流形学习可以帮助我们更好地理解数据的内在结构和特征，从而提高数据分析和处理的效率和准确性。

⚠ This conversation may have been edited by the sharer. This bot may make incorrect statements. It does not have knowledge of events after 2021.

Generative Models in Manifold Learning



Spectral Geometric Embedding

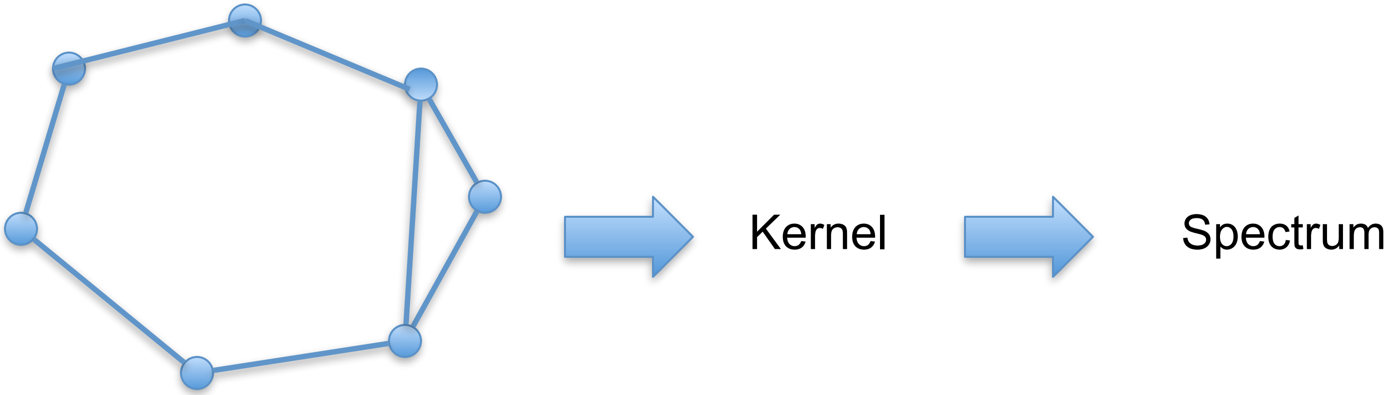
Given $x_1, \ldots, x_n \in \mathcal{M} \subset \mathbb{R}^N$,
Find $y_1, \ldots, y_n \in \mathbb{R}^d$ where $d << N$

- ISOMAP (Tenenbaum, et al, 00)
- LLE (Roweis, Saul, 00)
- Laplacian Eigenmaps (Belkin, Niyogi, 01)
- Local Tangent Space Alignment (Zhang, Zha, 02)
- Hessian Eigenmaps (Donoho, Grimes, 02)
- Diffusion Maps (Coifman, Lafon, et al, 04)

Related: Kernel PCA (Schoelkopf, et al, 98)

Meta-Algorithm

- Construct a neighborhood graph
- Construct a positive semi-definite kernel
- Find the spectrum decomposition



Kernel

Spectrum

Recall: ISOMAP

- Construct a neighborhood graph
- Find shortest path (geodesic) distance between every pair of nodes (points)
- Embed using classical Multidimensional Scaling

Recall: LLE

- Construct a neighborhood Graph G=(V,E)
- Solve weights

$$\min_{\sum_{j\in\mathbb{N}_i} w_{ij}=1} \|x_i - \sum_{j\in\mathcal{N}_i} w_{ij}x_j\|^2,$$

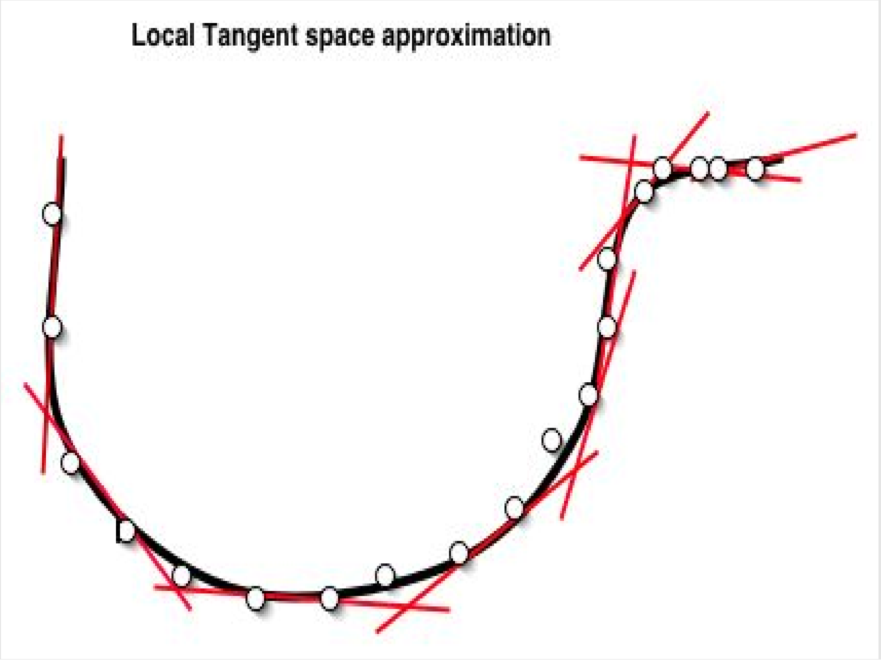
- Compute Embedding

$$\min_{Y} \sum_{i=1}^{n} \|Y_i - \sum_{j=1}^{n} W_{ij}Y_j\|^2 = \mathrm{trace}((I-W)Y^TY(I-W)^T).$$

$$W_{ij}^{n\times n} = \begin{cases} w_{ij} & j \in \mathcal{N}(i), \\ 0 & \text{other's.} \end{cases}$$

This is equivalent to find smallest eigenvectors of $K = (I-W)^T(I-W)$.

Local Tangent Space Alignment

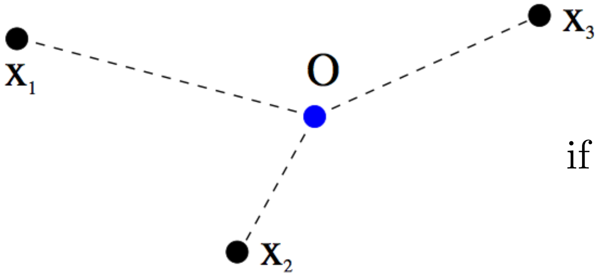


Find a good approximation of tangent space of curve with discrete points
by minimizing the projections on normal spaces.
— Principal curve/manifold (Hastie-Stuetzle’89, Zha-Zhang’02)

Recall LTSA (Zha-Zhang'02)

```
Algorithm 6: LTSA Algorithm
Input: A weighted undirected graph $G = (V, E)$ such that
$V = \{x_i \in \mathbb{R}^p : i = 1, \dots, n\}$
$E = \{(i, j) :$ if $j$ is a neighbor of $i$, i.e. $j \in \mathcal{N}_i\}$, e.g. $k$-nearest neighbors
Output: Euclidean $d$-dimensional coordinates $Y = [y_i] \in \mathbb{R}^{k \times n}$ of data.
Step 1 (local PCA): Compute local SVD on neighborhood of $x_i$, $x_{i_j} \in \mathcal{N}(x_i)$,
    $$\tilde{X}^{(i)} = [x_{i_1} - \mu_i, \dots, x_{i_k} - \mu_i]^{p \times k} = \tilde{U}^{(i)} \tilde{\Sigma} (\tilde{V}^{(i)})^T,$$
    where $\mu_i = \sum_{j=1}^{k} x_{i_j}$. Define
    $$G_i = [1/\sqrt{k}, \tilde{V_1}^{(i)}, \dots, \tilde{V_d}^{(i)}]^{k \times (d+1)};$$
Step 2 (tangent space alignment): Alignment (kernel) matrix
    $$K^{n \times n} = \sum_{i=1}^{n} S_i W_i W_i^T S_i^T, \quad W_i^{k \times k} = I - G_i G_i^T,$$
    where selection matrix $S_i^{n \times k} : [x_{i_1}, \dots, x_{i_k}] = [x_1, \dots, x_n] S_i^{n \times k}$;
Step 3: Find smallest $d + 1$ eigenvectors of $K$ and drop the smallest eigenvector,
    the remaining $d$ eigenvectors will give rise to a $d$-embedding.
```

Recall: Hessian LLE



In LLE, one chooses the weights w_{ij} to minimize the following energy

$$\min_{\sum_{j\in\mathbb{N}_i} w_{ij}=1} \| \sum_{j\in\mathcal{N}_i} w_{ij}(x_j - x_i)\|^2.$$

if the points $\tilde{x}_j = x_j - x_i$ are linearly dependent

$$0 = \sum_{j\in\mathcal{N}_i} w_{ij}\tilde{x}_j, \quad \text{and} \quad 1 = \sum_{j\in\mathcal{N}_i} w_{ij}.$$

For any smooth function $y(x)$, consider its Taylor expansion up to the second order

$$y(x) = y(0) + x^T \nabla y(0) + \frac{1}{2} x^T (\mathcal{H}y)(0) x + o(\|x\|^2).$$

$$\begin{aligned}
(I - W)y(0) \quad &:= \quad y(0) - \sum_{j\in\mathcal{N}_i} w_{ij} y(\tilde{x}_j) \\
&\approx \quad y(0) - \sum_{j\in\mathcal{N}_i} w_{ij} y(0) - \sum_{j\in\mathcal{N}_i} w_{ij} \tilde{x}_j^T \nabla y(0) - \frac{1}{2} \sum_{j\in\mathcal{N}_i} \tilde{x}_j^T (\mathcal{H}y)(0) \tilde{x}_j \\
&= \quad -\frac{1}{2} \sum_{j\in\mathcal{N}_i} \tilde{x}_j^T (\mathcal{H}y)(0) \tilde{x}_j.
\end{aligned}$$

Recall: Hessian LLE Alg. (I)

Algorithm 7: Hessian LLE Algorithm

```
Input: A weighted undirected graph $G = (V, E, d)$ such that
$V = \{x_i \in \mathbb{R}^p : i = 1, \ldots, n\}$
$E = \{(i, j) :$ if $j$ is a neighbor of $i$, i.e. $j \in \mathcal{N}_i\}$, e.g. $k$-nearest neighbors
Output: Euclidean $d$-dimensional coordinates $Y = [y_i] \in \mathbb{R}^{d \times n}$ of data.
Step 1: Compute local PCA on neighborhood of $x_i$, for,
        $\tilde{X}^{(i)} = [x_{i_1} - \mu_i, \ldots, x_{i_k} - \mu_i]^{p \times k} = \tilde{U}^{(i)} \tilde{\Sigma} (\tilde{V}^{(i)})^T, \quad x_{i_j} \in \mathcal{N}(x_i),$
    where $\mu_i = \sum_{j=1}^{k} x_{i_j} = \frac{1}{k} X_i \mathbf{1}$;
```

- Left top singular vectors $\{\tilde{U}_1^{(i)}, \ldots, \tilde{U}_d^{(i)}\}$ give an orthonormal basis of the approximate tangent space at x_i,
- Right top singular vectors $[\tilde{V}_1^{(i)}, \ldots, \tilde{V}_d^{(i)}]$ are representation coordinates in the tangent space of local sample points around x_i.

Continued...

Hessian LLE Algorithm (II)

Step 2: Null Hessian estimation: define

$$M = [1, \tilde{V}_1, ..., \tilde{V}_d, \tilde{V}_1^2, \tilde{V}_1 \odot \tilde{V}_2, ..., \tilde{V}_{d-1} \odot \tilde{V}_d, \tilde{V}_d^2] \in \mathbb{R}^{k \times (1+d+\binom{d+1}{2})}$$

where $\tilde{V}_i \odot \tilde{V}_j = [\tilde{V}_{ik}\tilde{V}_{jk}]^T \in \mathbb{R}^k$ denotes the elementwise product (Hadamard product) between vector $\tilde{V}_i$ and $\tilde{V}_j$. Now we perform a Gram-Schmidt Orthogonalization procedure on M, get

$$\tilde{M} = [1, \hat{v}_1, ..., \hat{v}_d, \hat{w}_1, \hat{w}_2, ..., \hat{w}_{\binom{d+1}{2}}] \in \mathbb{R}^{k \times (1+d+\binom{d+1}{2})}$$

Define

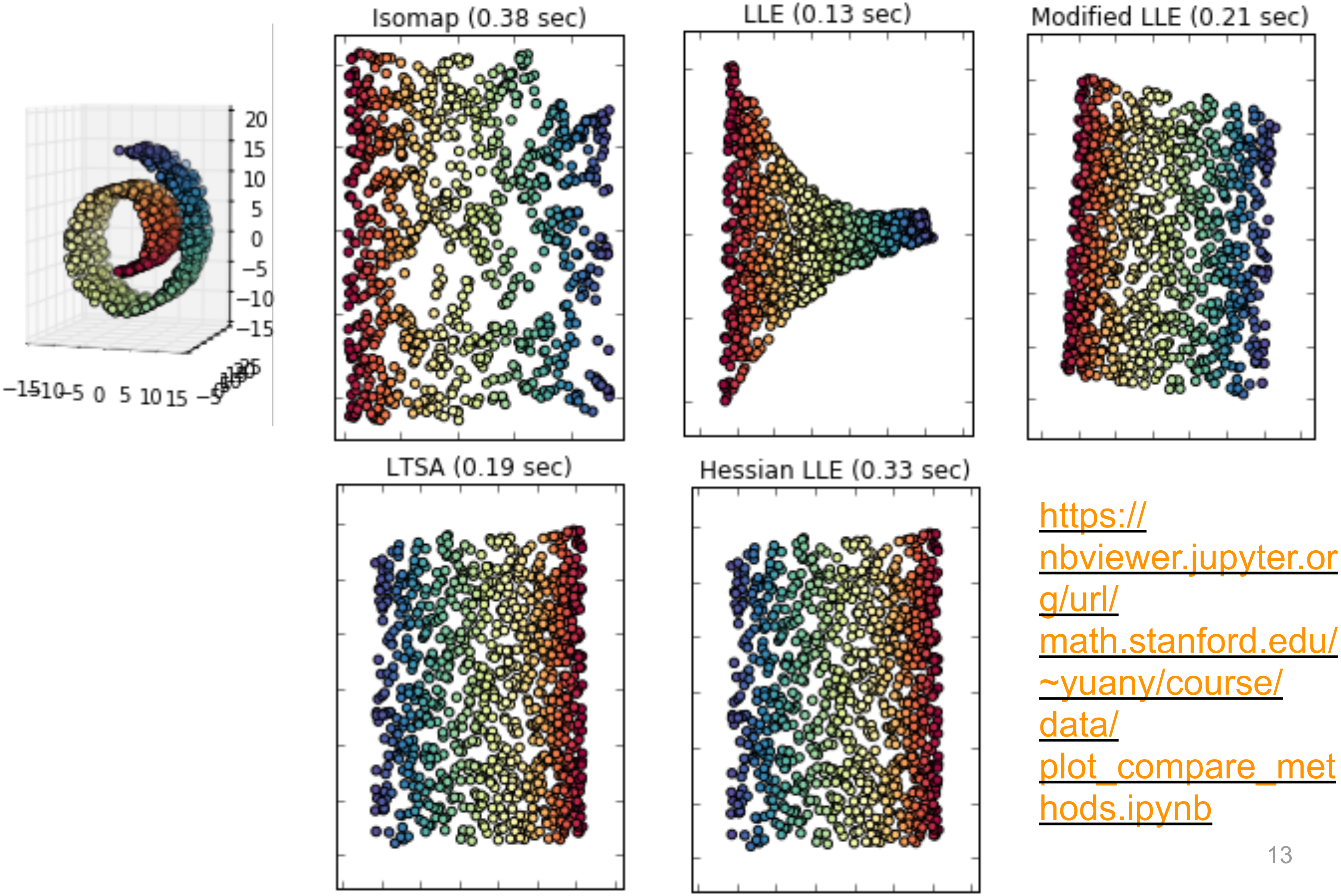
$$[H^{(i)}]^T = [last \quad \binom{d+1}{2} \quad columns \quad of \quad \tilde{M}]_{k \times \binom{d+1}{2}}.$$

Step 3: Define

$$K = \sum_{i=1}^{n} S^{(i)} H^{(i)T} H^{(i)} S^{(i)T} \in \mathbb{R}^{n \times n}, \quad [x_1, .., x_n] S^{(i)} = [x_{i_1}, ..., x_{i_k}],$$

find smallest $d+1$ eigenvectors of K and drop the smallest eigenvector, and the remaining d eigenvectors will give rise to a d-embedding.

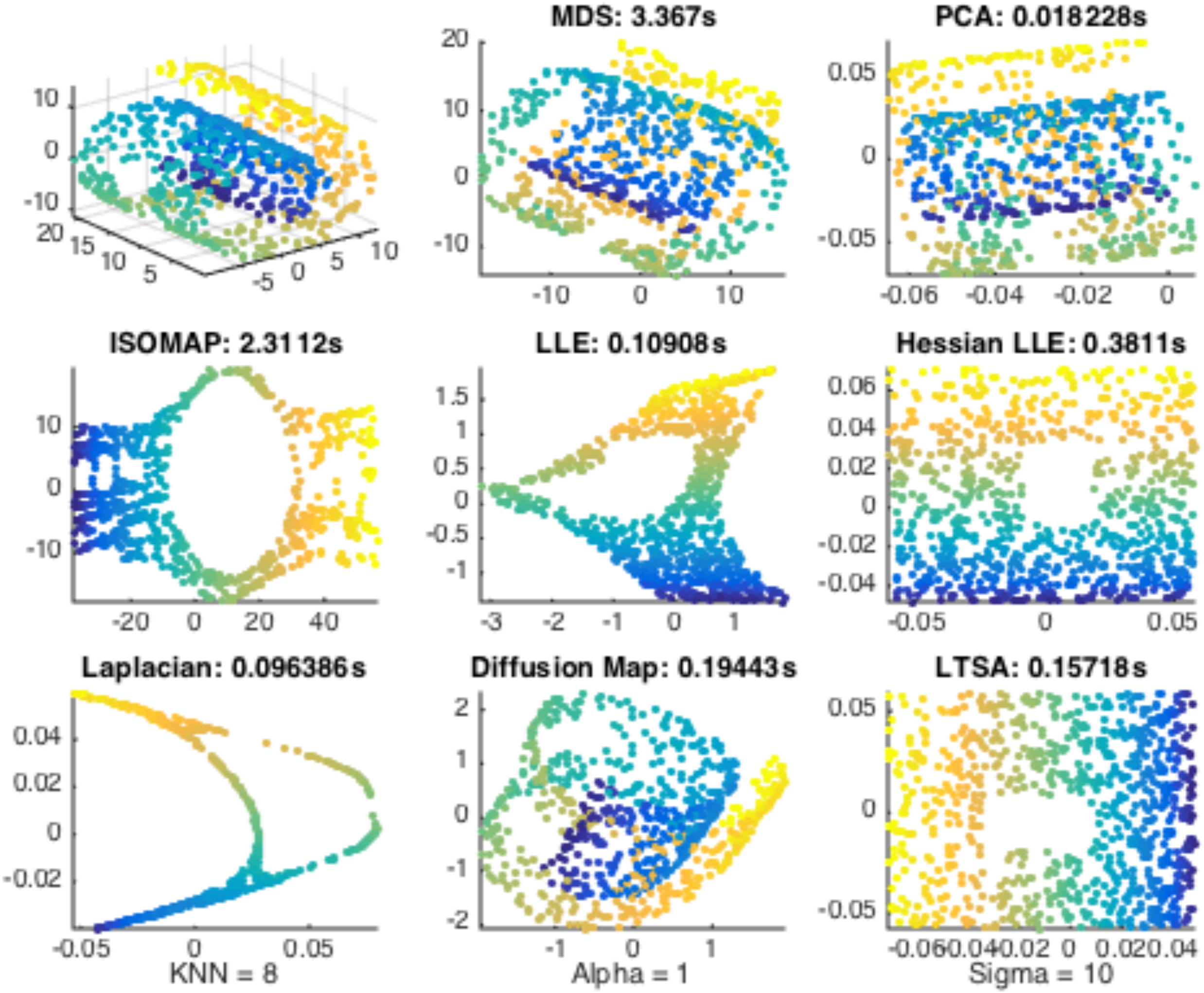
Comparisons on Swiss Roll



https://nbviewer.jupyter.org/url/math.stanford.edu/~yuany/course/data/plot_compare_methods.ipynb

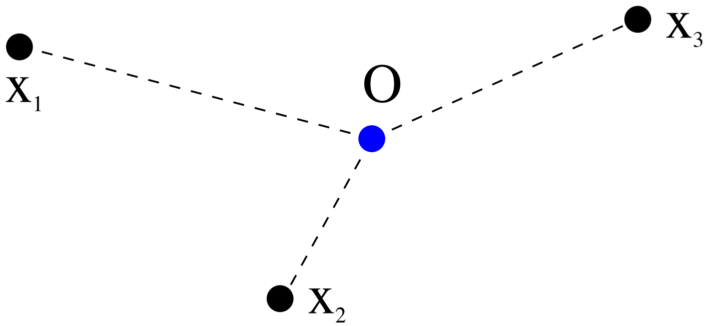
Comparisons on Swiss Roll with a Hole

- mani.m



Laplacian LLE (Eigenmap)

Laplacian and LLE



$$\sum w_i x_i = 0$$

$$\sum w_i = 1$$

Hessian H. Taylor expansion :

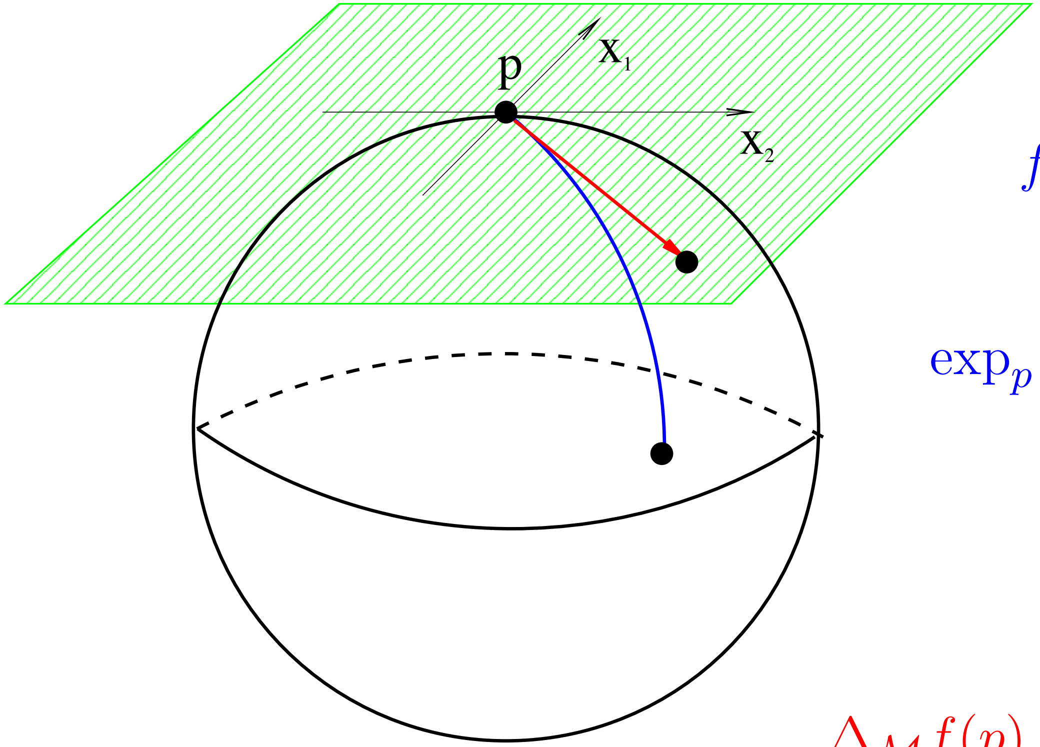
$$f(x_i) = f(0) + x_i^t \nabla f + \frac{1}{2} x_i^t H x_i + o(\|x_i\|^2)$$

$$(I-W)f(0) = f(0) - \sum w_i f(x_i) \approx f(0) - \sum w_i f(0) - \sum_i w_i x_i^t \nabla f - \frac{1}{2} \sum_i x_i^t H x_i =$$

$$= -\frac{1}{2} \sum_i x_i^t H x_i \approx -trH = \Delta f$$

when x_i becomes an orthonormal basis…

Laplacian-Beltrami Operator on Manifold



$f : \mathcal{M}^k \to \mathbb{R}$

$\exp_p : T_p\mathcal{M}^k \to \mathcal{M}^k$

$$\Delta_{\mathcal{M}} f(p) \equiv \sum_i \frac{\partial^2 f(\exp_p(x))}{\partial x_i^2}$$

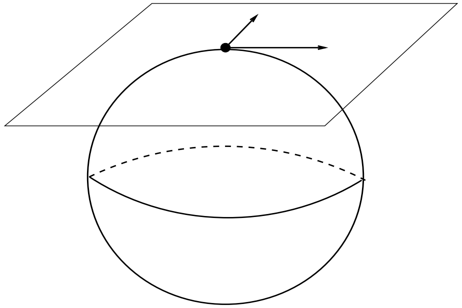
Orthonormal coordinate system.

Manifold Laplacian

Recall ordinary Laplacian in $\mathbb{R}^k$
This maps

$$f(x_1, \ldots, x_k) \to \left(-\sum_{i=1}^{k} \frac{\partial^2 f}{\partial x_i^2} \right)$$

Manifold Laplacian is the same on the tangent space.



Discrete Approximation

smooth map $f : \mathcal{M} \to R$

$$\int_{\mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 \approx \sum_{i \sim j} W_{ij}(f_i - f_j)^2$$

Recall standard gradient in $\mathbb{R}^k$ of $f(z_1, \ldots, z_k)$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial z_1} \\ \frac{\partial f}{\partial z_2} \\ \cdot \\ \\ \cdot \\ \\ \frac{\partial f}{\partial z_k} \end{bmatrix}$$

Stokes Theorem

A Basic Fact

$$\int_{\mathcal{M}} \|\nabla_{\mathcal{M}} f\|^2 = \int f \cdot \Delta_{\mathcal{M}} f$$

This is like

$$\sum_{i,j} W_{ij}(f_i - f_j)^2 = \mathbf{f}^T \mathbf{L} \mathbf{f}$$

where

$\Delta_{\mathcal{M}} f$ is the manifold Laplacian

Discrete Laplacian

Find $y_1, \ldots, y_n \in R$

$$\min \sum_{i,j} (y_i - y_j)^2 W_{ij}$$

Tries to preserve locality

A Fundamental Identity

- Discrete Laplacian

  $$L = D - W$$

  where $D = \mathbf{diag}(D_{ii})$ with $D_{ii} := \sum_j W_{ij}$.

- Fundamental identity:

  $$\sum_{i,j} W_{ij}(y_i - y_j)^2 = \mathbf{y}^T L \mathbf{y}$$

$$\begin{aligned}
\sum_{i,j} W_{ij}(y_i - y_j)^2 &= \sum_{i,j} W_{ij}(y_i^2 + y_j^2 - 2y_i y_j) \\
&= \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2 \sum_{i,j} W_{ij} y_i y_j \\
&= \mathbf{y}^T L \mathbf{y}
\end{aligned}$$

Laplacian Eigenmap: uniform sampling

- $L\mathbf{1} = \mathbf{0}$, so $(0, \mathbf{1})$ is an eigenvalue-eigenvector pair.
- For uniform samples, define $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{d \times n}$

$$\min_{\mathbf{Y}\mathbf{1}=\mathbf{0}} \quad \sum_{i,j} W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 = \operatorname{tr} \mathbf{Y} L \mathbf{Y}^T$$

$$\text{subject to} \quad \mathbf{Y}\mathbf{Y}^T = I_d$$

- Eigenvectors of L gives the embedding.

Laplacian Eigenmap: non-uniform sampling

- For nonuniform samples, solves

$$\begin{aligned} \min_{\mathbf{y}^T\mathbf{1}=0} \quad & \mathbf{y}^T L \mathbf{y} \\ \text{subject to} \quad & \mathbf{y}^T D \mathbf{y} = 1 \end{aligned}$$

- Generalized Eigenvectors of $L\mathbf{y} = \lambda D\mathbf{y}$, or eigenvectors of normalized Laplacian $L_n = D^{-1}L$, give the embedding.
- This is particularly a case of Diffusion Map.

Laplacian Eigenmaps (I) [Belkin-Niyogi 2002]

Algorithm 8: Laplacian Eigenmap

```
Input: An adjacency graph $G = (V, E, d)$ such that
$V = \{x_i : i = 1, \ldots, n\}$
$E = \{(i, j) :$ if $j$ is a neighbor of $i$, i.e. $j \in \mathcal{N}_i\}$, e.g. $k$-nearest neighbors, $\epsilon$-neighbors
$d_{ij} = d(x_i, x_j)$, e.g. Euclidean distance for $x_i \sim x_j$ are in neighbor
Output: Euclidean $d$-dimensional coordinates $Y = [y_i] \in \mathbb{R}^{k \times n}$ of data.
Step 1: Choose weights
(a) Heat kernel weights (parameter $t$):
    $$W_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{t}}, & i \sim j, \\ 0, & \text{otherwise.} \end{cases}$$
    (b) Simple-minded ($t \to \infty$), $W_{ij} = 1$ if $i$ and $j$ are connected by an edge and $W_{ij} = 0$ otherwise.
Step 2 (Eigenmap): Let $D = \mathrm{diag}(\sum_j W_{ij})$ and $L = D - W$. Compute smallest $d + 1$ generalized eigenvectors
    $$L y_l = \lambda_l D y_l, \quad l = 0, 1, \ldots, d,$$
such that $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_d$. Drop the zero eigenvalue $\lambda_0$ and constant eigenvector $y_0$, and construct $Y_d = [y_1, \ldots, y_d] \in \mathbb{R}^{n \times d}$.
```

Hessian vs. Laplacian

- Laplacian LLE

$$f^T L f = \sum_{i \geq j} w_{ij}(f_i - f_j)^2 \geq 0 \sim \int \|\nabla_M f\|^2 = \int (\mathrm{trace}(f^T \mathcal{H} f))^2$$

where $\mathcal{H} = [\partial^2/\partial_i \partial_j] \in \mathbb{R}^{d \times d}$ is the Hessian matrix.

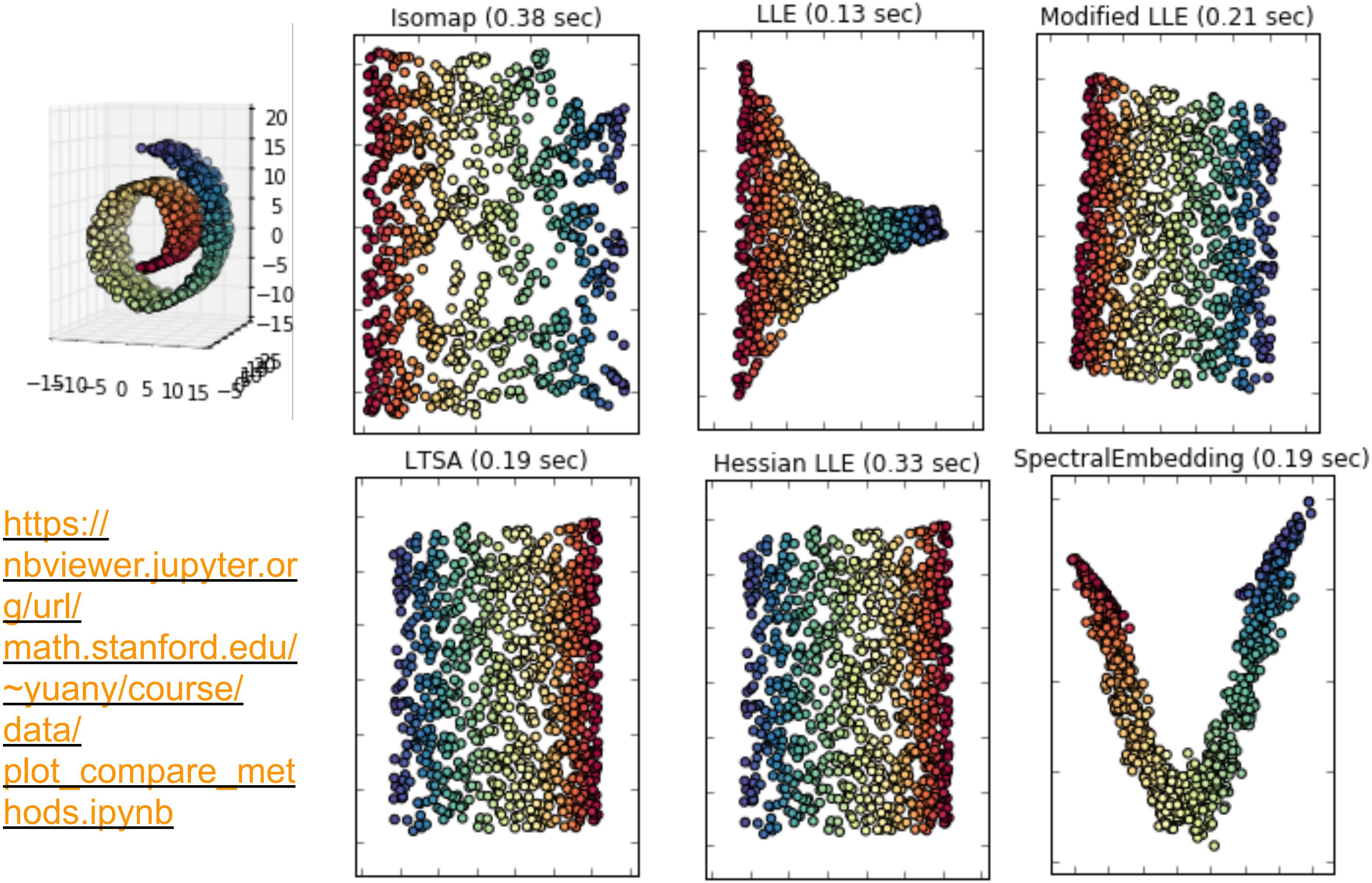
- Hessian LLE

$$\min \int \|\mathcal{H} f\|^2, \quad \|f\| = 1$$

- Laplacian kernel: const + linear + bilinear
- Hessian kernel: const + linear functions

Note that: $\Delta(f) = trace\big(H(f)\big)$

Comparisons on Swiss Roll



https://nbviewer.jupyter.org/url/math.stanford.edu/~yuany/course/data/plot_compare_methods.ipynb

Convergence of Laplacian Eigenmaps

Manifold Laplacian Eigenvectors

Eigensystem

$$\Delta_{\mathcal{M}} f = \lambda_i \phi_i$$

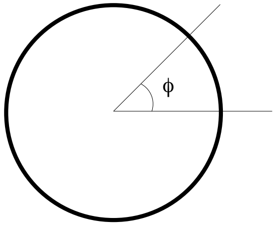
$\lambda_i \geq 0$ and $\lambda_i \to \infty$

$\{\phi_i\}$ form an orthonormal basis for $L^2(\mathcal{M})$

$$\int \|\nabla_{\mathcal{M}} \phi_i\|^2 = \lambda_i$$

Manifold Laplacian is non-compact!

Example: Circle



$$-\frac{d^2u}{dt^2} = \lambda u \text{ where } u(0) = u(2\pi)$$

Eigenvalues are

$$\lambda_n = n^2$$

Eigenfunctions are

$$\sin(nt), \cos(nt)$$

Spherical Harmonics in high-D sphere!

Spectral Growth

$$\lambda_1 \leq \lambda_2 \ldots \leq \lambda_j \leq \ldots$$

Then

$$A + \frac{2}{d}\log(j) \leq \log(\lambda_j) \leq B + \frac{2}{d}\log(j+1)$$

Example: on S^1

$$\lambda_j = j^2 \implies \log(\lambda_j) = \frac{2}{1}\log(j)$$

(Li and Yau; Weyl's asymptotics)

Solution of Heat Equations

Heat equation in $\mathbb{R}^n$:

$u(x,t)$ – heat distribution at time t.
$u(x,0) = f(x)$ – initial distribution. $x \in \mathbb{R}^n, t \in \mathbb{R}$.

$$\Delta_{\mathbb{R}^n} u(x,t) = \frac{du}{dt}(x,t)$$

Solution – convolution with the heat kernel:

$$u(x,t) = (4\pi t)^{-\frac{n}{2}} \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} dy$$

Discretization of Heat Eq.

Functional approximation:

Taking limit as $t \to 0$ and writing the derivative:

$$\Delta_{\mathbb{R}^n} f(x) = \frac{d}{dt}\left[(4\pi t)^{-\frac{n}{2}} \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} dy\right]_0$$

$$\Delta_{\mathbb{R}^n} f(x) \approx -\frac{1}{t}(4\pi t)^{-\frac{n}{2}} \left(f(x) - \int_{\mathbb{R}^n} f(y) e^{-\frac{\|x-y\|^2}{4t}} dy\right)$$

Empirical approximation:

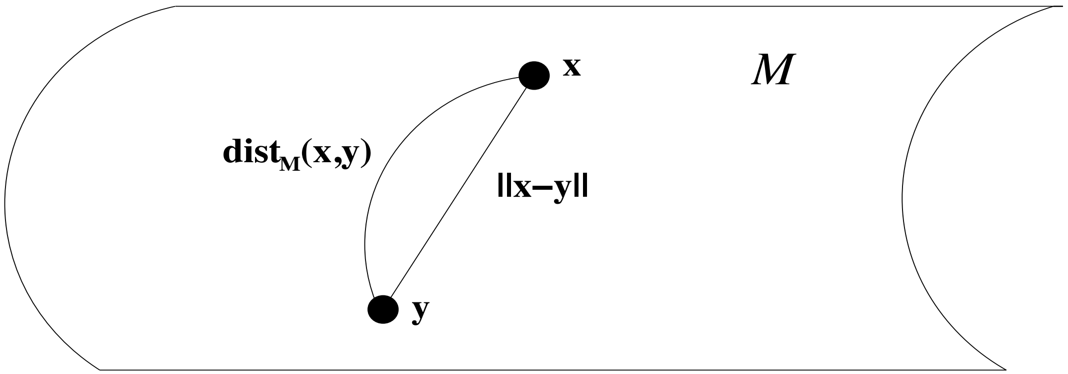
Integral can be estimated from empirical data.

$$\Delta_{\mathbb{R}^n} f(x) \approx -\frac{1}{t}(4\pi t)^{-\frac{n}{2}} \left(f(x) - \sum_{x_i} f(x_i) e^{-\frac{\|x-x_i\|^2}{4t}}\right)$$

Some Difficulties for Manifolds

Some difficulties arise for manifolds:

- Do not know distances.
- Do not know the heat kernel.



Careful analysis needed.

The Heat Kernel Approximation

- $H_t(x,y) = \sum_i e^{-\lambda_i t}\phi_i(x)\phi_i(y)$
- in $\mathbb{R}^d$, closed form expression

$$H_t(x,y) = \frac{1}{(4\pi t)^{d/2}} e^{-\frac{\|x-y\|^2}{4t}}$$

- Goodness of approximation depends on the gap

$$\left| H_t(x,y) - \frac{1}{(4\pi t)^{d/2}} e^{-\frac{\|x-y\|^2}{4t}} \right|$$

- H_t is a Mercer kernel intrinsically defined on manifold. Leads to SVMs on manifolds.

Pointwise Convergence

$$f : \mathcal{M} \to \mathbb{R} \quad x \in \mathcal{M} \quad x_1, \ldots, x_n \in \mathcal{M}$$

Graph Laplacian:

$$L_n^t(f)(x) = f(x) \sum_j e^{-\frac{\|x-x_j\|^2}{t}} - \sum_j f(x_j) e^{-\frac{\|x-x_j\|^2}{t}}$$

Theorem [pointwise convergence] $t_n = n^{-\frac{1}{k+2+\alpha}}$

$$\lim_{n\to\infty} \frac{(4\pi t_n)^{-\frac{k+2}{2}}}{n} L_n^{t_n} f(x) = \Delta_{\mathcal{M}} f(x)$$

Belkin 03, Lafon Coifman 04, Belkin Niyogi 05, Hein et al 05

Convergence of Eigenfunctions

Theorem [convergence of eigenfunctions]

$$\lim_{t \to 0, n \to \infty} Eig[L_n^{t_n}] \to Eig[\Delta_{\mathcal{M}}]$$

Belkin Niyogi 06

Diffusion Map

Connection to Markov Chain

- $L = D - W$
- $P = I - D^{-1}L = D^{-1}W$ is a markov matrix
- v is generalized eigenvector of L: $L v = \lambda D v$
- v is also a right eigenvector of P with eigenvalue $1-\lambda$
- P is lumpable iff v is piece-wise constant
- So Laplacian eigenmaps have Markov Chain interpretations (Diffusion Map), with more connection to topology …

Data Graph

- Given n points x_i, $i=1,\ldots,n$, as vertices in V
- Similarity weight between x_i and x_j is $w_{ij}=w_{ji}$, e.g.

$$W_{ij} = \begin{cases} e^{-\frac{\|x_i - x_j\|^2}{t}}, & i \sim j, \\ 0, & \text{otherwise.} \end{cases}$$

- Undirected weighted graph $G(V,E,W)$

Random Walk on Graphs

- Degree $d_i = \Sigma_k w_{ik}$, $D = \mathrm{diag}(d_i)$
- Random walk on $G(V,E,W)$
  - Transition probability $P = D^{-1} W$ where $p_{ij} = w_{ij}/d_i$
  - Stationary distribution $\pi_i \sim d_i$
  - primitive (G is connected with a finite diameter)
  - Reversible $w_{ij} = w_{ji}$ → $\pi_i p_{ij} = \pi_j p_{ji}$

Symmetric Kernel

- P = $D^{-1}W$ is similar to S = $D^{-1/2}WD^{-1/2}$, as P = $D^{-1/2}SD^{1/2}$
- S is real symmetric, whence eigen-decomposition

$$S = V\Lambda V^T, \qquad \Lambda = diag(\lambda_i \in R)$$

⇨ $$P = D^{-1/2} V \Lambda V^T D^{1/2} = \Phi \Lambda \Psi^T, \qquad \Phi = D^{-1/2} V, \qquad \Psi = D^{1/2} V$$

Spectrum of P

- Eigenvalues of S and P are the same, so

$$\left|\lambda_i\right| \le 1$$

- Φ and Ψ are right and left eigenvector matrix of P, respectively, $\Phi^T\Psi = V^TV = I$
- In particular, P 1 = 1, whence

$$\phi_1(i) = 1, \qquad \psi_1(i) = \frac{d_i}{\sum_i d_i} = \pi_i$$

Diffusion Map

- If P is *primitive* (any two points can be connected by path of length no more than the diameter),

$$1 = \lambda_0 \geq \lambda_1 \geq \lambda_2 \ldots \geq \lambda_{n-1} > -1.$$

- **Diffusion map** embedding at scale τ by dropping the constant eigenvector ϕ_0:

$$\Phi_\tau(x_i) = [\lambda_1^\tau \phi_1(i), \cdots, \lambda_{n-1}^\tau \phi_{n-1}(i)] \in \mathbb{R}^{n-1}, \ \tau \geq 0.$$

- Laplacian LLE (eigenmap) is just the special case $\tau = 0$ with top $d+1$ eigenvectors

Dimensionality Reduction

- $\lambda_0 = 1$ and $\phi_0 = \mathbf{1}$, so it does not distinguish points
- Threshold by δ, for those

$$|\lambda_i^\tau| \geq 1 - \delta, \quad i = 1, \ldots, d,$$

$$|\lambda_j^\tau| \leq 1 - \delta, \quad j \geq d + 1,$$

- Diffusion map embedding with dimensionality reduction:

$$\Phi_\tau(x_i) = [\lambda_1^\tau \phi_1(i), \cdots, \lambda_{n-1}^\tau \phi_d(i)] \in \mathbb{R}^d, \ \tau \geq 0.$$

- Varying τ or δ leads to a multiscale analysis

Diffusion Distance

- Define the diffusion distance between points at scale t

$$D^t(x_i, x_j) = \|\Phi_t(x_i) - \Phi_t(x_j)\|_{\ell^2} := \left(\sum_k \lambda_k^{2t} (\phi_k(i) - \phi_k(j))^2 \right)^{1/2},$$

- This is exactly the weighted 2-distance between diffusion profiles

$$D^t(x_i, x_j) = \|P_{i*}^t - P_{j*}^t\|_{\ell^2(1/d)} := \left(\sum_{k=1}^{n} \frac{(P(i,k) - P(j,k))^2}{d_k} \right)^{1/2}.$$

Diffusion Distance Example

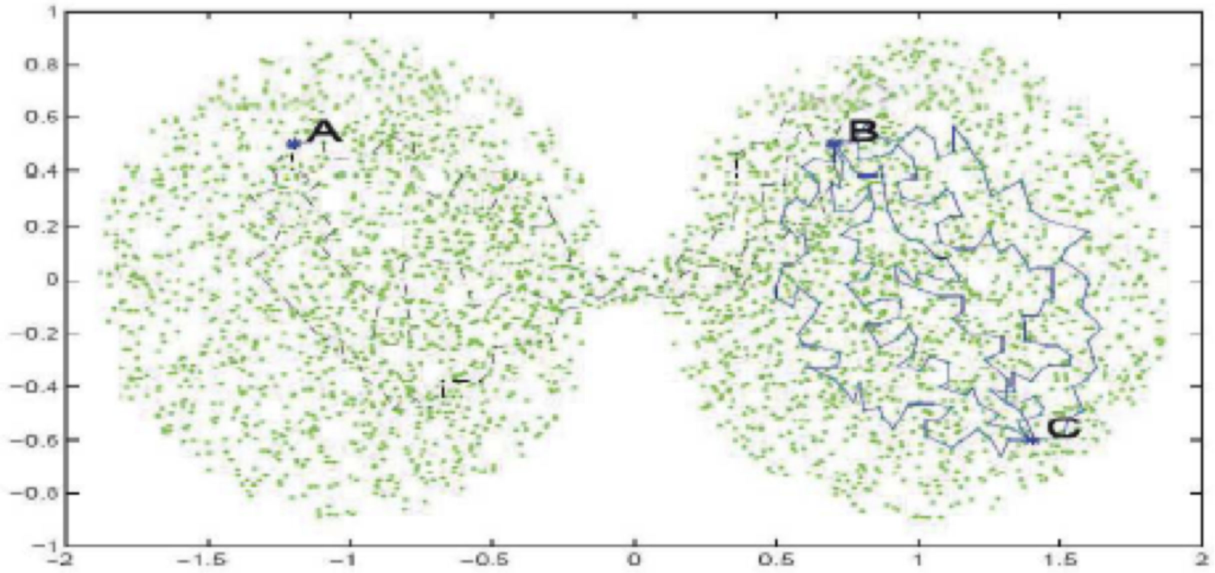


FIGURE 1. Diffusion Distances $d_t(A, B) >> d_t(B, C)$ while graph shortest path $d_{geod}(A, B) \sim d_{geod}(B, C)$.

General Diffusion Map

- Let $k_t(x,y) = \exp(-\|x-y\|^2/t)$
- Define
$$q_t(x) = \int_{\mathcal{M}} k_t(x,y)q(y)dy$$
and form the new kernel
$$k_t^{(\alpha)}(x,y) = \frac{k_t(x,y)}{q_t^\alpha(x)q_t^\alpha(y)}.$$
- Let
$$d_t^{(\alpha)}(x) = \int_{\mathcal{M}} k_t^{(\alpha)}(x,y)q(y)dy$$
and define the transition kernel of a Markov chain by
$$p_{t,\alpha}(x,y) = \frac{k_t^{(\alpha)}(x,y)}{d_t^{(\alpha)}(x)}.$$

General Diffusion Map

- Define the Markov chain

$$P_{t,\alpha} f(x) = \int_{\mathcal{M}} p_{t,\alpha}(x,y) f(y) q(y) dy.$$

- Define the Laplacian

$$L_{t,\alpha} = \frac{I - P_{t,\alpha}}{t}.$$

- The bottom eigenvectors of $L_{t,\alpha}$ give the embedding.

Convergence of General Laplacian

Theorem (Coifman-Lafon (2006))

Let $\mathcal{M} \in \mathbb{R}^p$ be a compact smooth submanifold, $q(x)$ be a probability density on $\mathcal{M}$, and $\Delta_{\mathcal{M}}$ be the Laplacian-Beltrami operator on $\mathcal{M}$.

$$\lim_{t \to 0} L_{t,\alpha} = \frac{\Delta_{\mathcal{M}}(f q^{1-\alpha})}{q^{1-\alpha}} - \frac{\Delta_{\mathcal{M}}(q^{1-\alpha}))}{q^{1-\alpha}}.$$

This suggests that

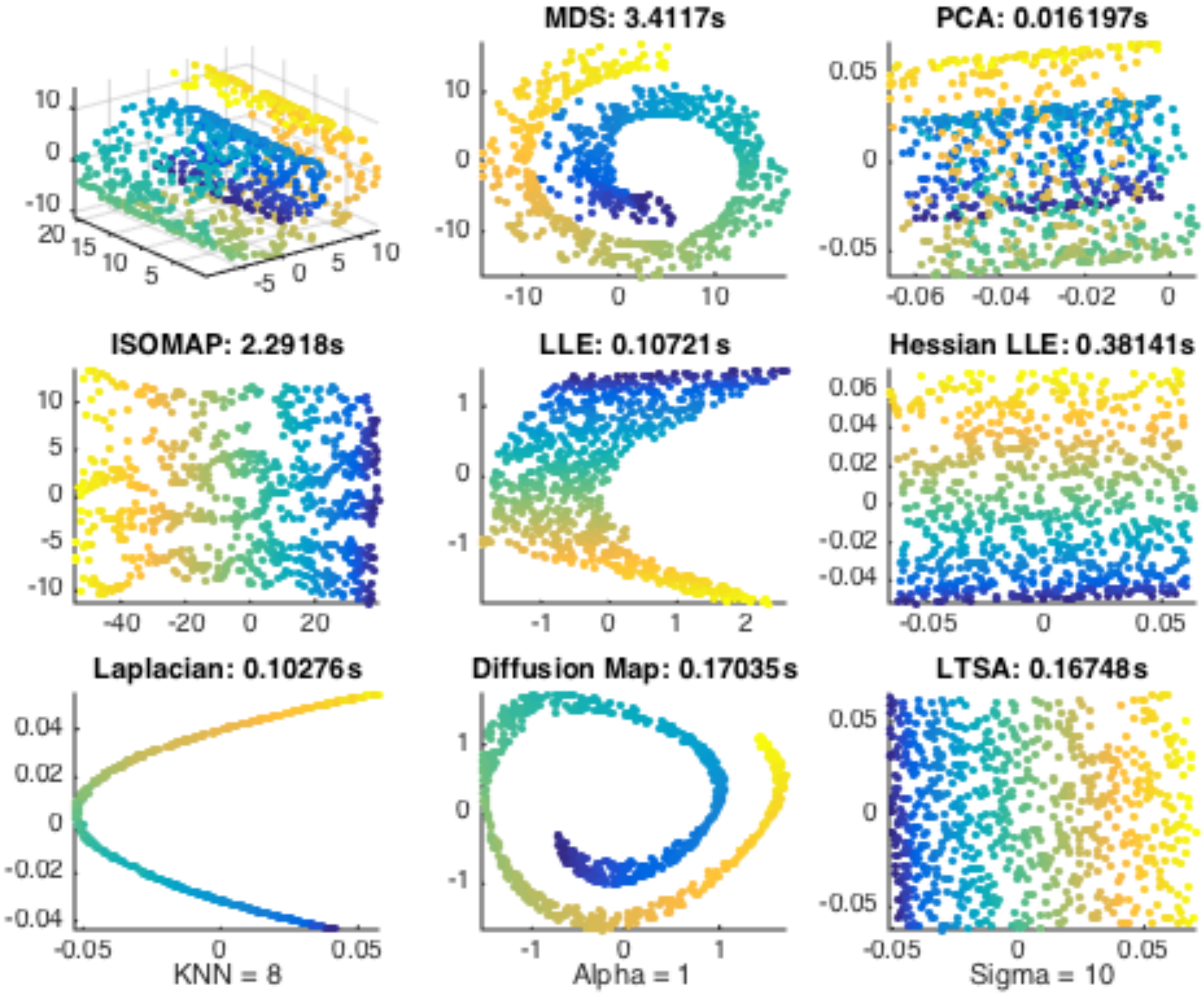
- for $\alpha = 1$, it converges to the Laplacian-Beltrami operator $\lim_{t\to 0} L_{t,1} = \Delta_{\mathcal{M}}$;
- for $\alpha = 1/2$, it converges to a Schrödinger operator whose conjugation leads to a forward Fokker-Planck equation;
- for $\alpha = 0$, it is the normalized graph Laplacian.

Comparisons of Manifold Learning Techniques

- MDS
- PCA
- ISOMAP
- LLE
- Hessian LLE
- Laplacian LLE
- Diffusion Map
- Local Tangent Space Alignment
- Matlab codes: mani.m

Courtesy of Todd Wittman

Comparisons on Swiss Roll



Diffussion Map vs. Stochastic Neighbor Embedding

- In Diffusion Map, it looks for MDS embedding which preserves diffusion distances

$$D_t(x_i, x_j) := \|P^t_{i*} - P^t_{j*}\|_{\ell^2(1/d)} = \sum_{k=1}^{m} \frac{(P^t_{ik} - P^t_{jk})^2}{d_k}$$

- SNE considers to find a low-dimensional Euclidean embedding Y which preserves the distribution P_{i*}

Stochastic Neighbor Embedding

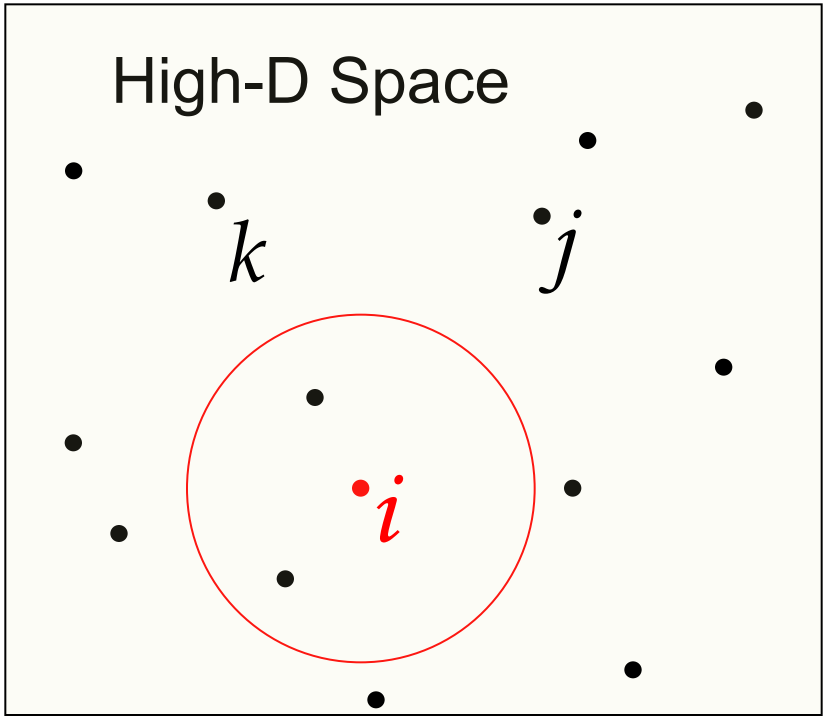
- Like diffusion map, consider the conditional probability that one data point will pick the other data point as its neighbor $p_{j|i}$

- However, to reconstruct the probability rather than clusters in embedding:
  - Use the pairwise distances in the low-dimensional map to define the probability that a map point will pick another map point as its neighbor.
  - Compute the Kullback-Leibler divergence between the probabilities in the high-dimensional and low-dimensional spaces.

A probabilistic local method

- Each point in high-D has a conditional probability of picking each other point as its neighbor.

- The distribution over neighbors is based on the high-D pairwise distances.
  - If we do not have coordinates for the datapoints we can use a matrix of dissimilarities instead of pairwise distances.

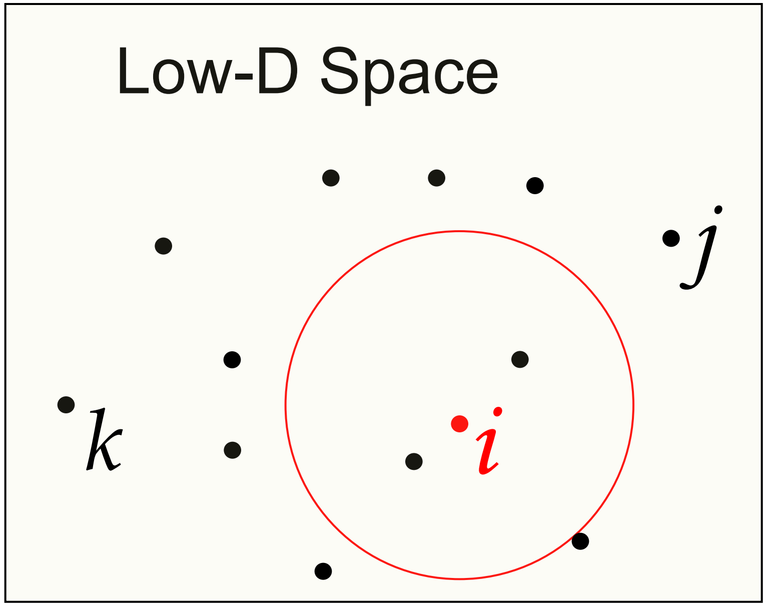


$$p_{j|i} = \frac{e^{-d_{ij}^2 \big/ 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 \big/ 2\sigma_i^2}}$$

probability of picking j given that you start at i

Evaluating an arrangement of the data in a low-dimensional space **Y**

- Give each data point a location in the low- dimensional space **Y.**
  - Evaluate this representation by seeing how well the low-D probabilities model the high-D ones.



$$q_{j|i} = \frac{e^{-d_{ij}^2}}{\sum_k e^{-d_{ik}^2}}$$

probability of picking j given that you start at i

The cost function for a low-dimensional representation

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

- For points where p_{ij} is large and q_{ij} is small we lose a lot.
  - Nearby points in high-D really want to be nearby in low-D
- For points where q_{ij} is large and p_{ij} is small we lose a little because we waste some of the probability mass in the Q_i distribution.
  - Widely separated points in high-D have a mild preference for being widely separated in low-D.

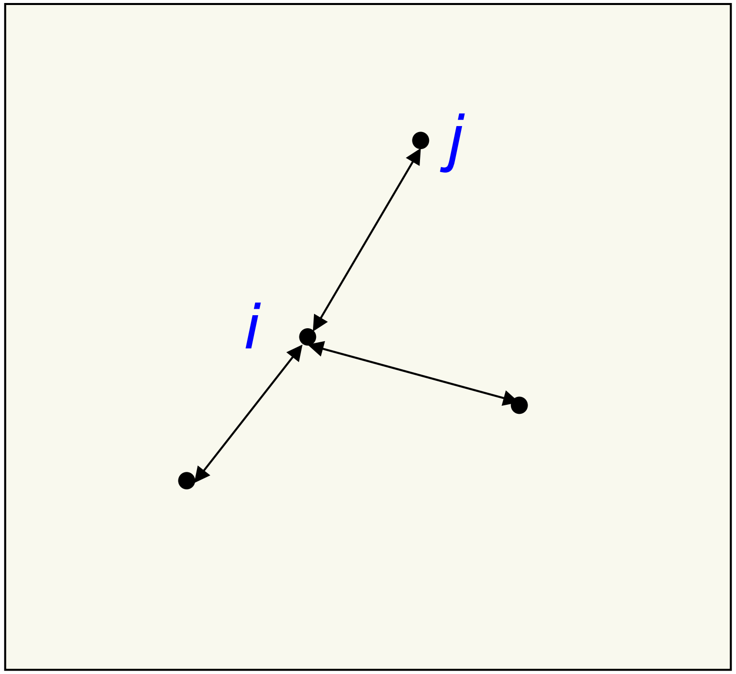
Gradient Descent

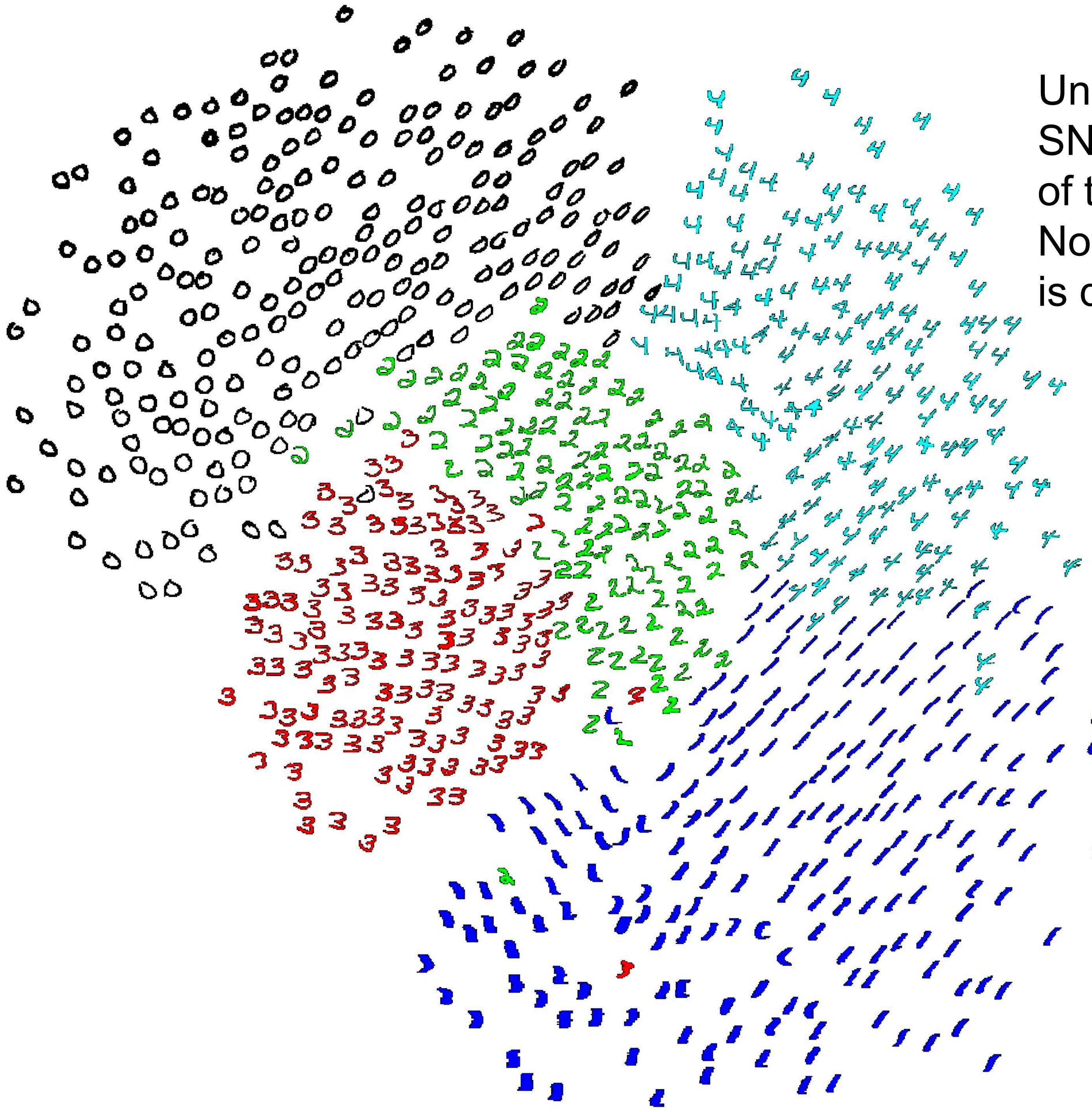
$$\frac{\partial Cost}{\partial \mathbf{y}_i} = 2\sum_j (\mathbf{y}_j - \mathbf{y}_i)\,(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$$

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$$

$$\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$$

- Points are pulled towards each other if the p's are bigger than the q's and repelled if the q's are bigger than the p's





Unsupervised SNE embedding of the digits 0-4. Not all the data is displayed

Picking the radius of the gaussian that is used to compute the p's

- We need to use different radii in different parts of the space so that we keep the effective number of neighbors about constant.
- A big radius leads to a high entropy for the distribution over neighbors of i.
- A small radius leads to a low entropy.
- So decide what entropy you want and then find the radius that produces that entropy.
- Its easier to specify 2^entropy
  - This is called the perplexity
  - It is the effective number of neighbors.

$$Perp(P_i) = 2^{H(P_i)},$$

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

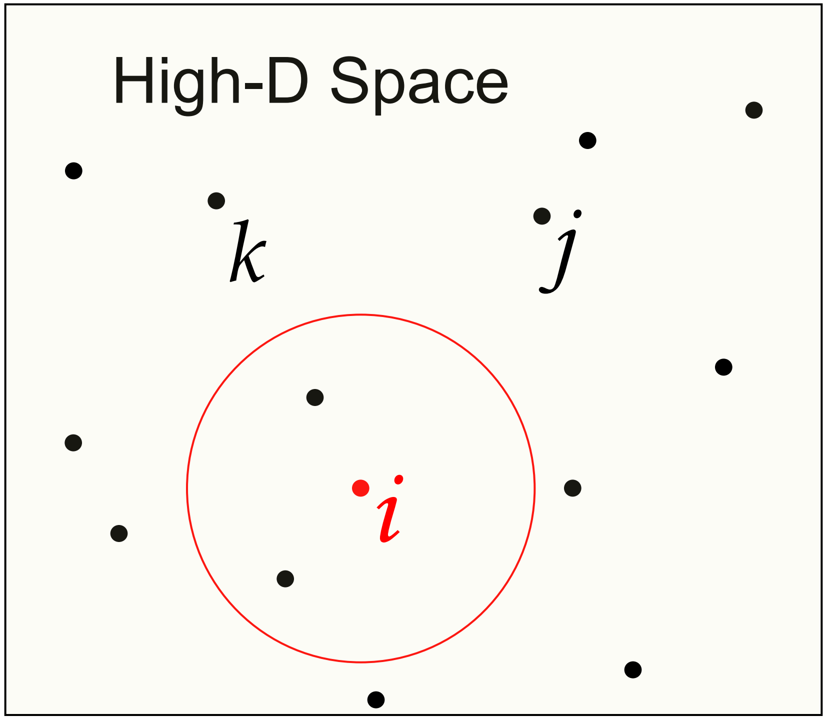
Symmetric SNE

- There is a simpler version of SNE which seems to work about equally well.
- Symmetric SNE works best if we use different procedures for computing the p's and the q's
  - This destroys the nice property that if we embed in a space of the same dimension as the data, the data itself is the optimal solution.

Computing the p’s for symmetric SNE

- Each high dimensional point, i, has a conditional probability of picking each other point, j, as its neighbor.

- The conditional distribution over neighbors is based on the high-dimensional pairwise distances.



$$p_{j|i} = \frac{e^{-d_{ij}^2 / 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 / 2\sigma_i^2}}$$

probability of picking j
given that you start at i

Turning conditional probabilities into pairwise probabilities

To get a symmetric probability between i and j we sum the two conditional probabilities and divide by the number of points (points are not allowed to choose themselves).

joint probability of picking the pair i,j →

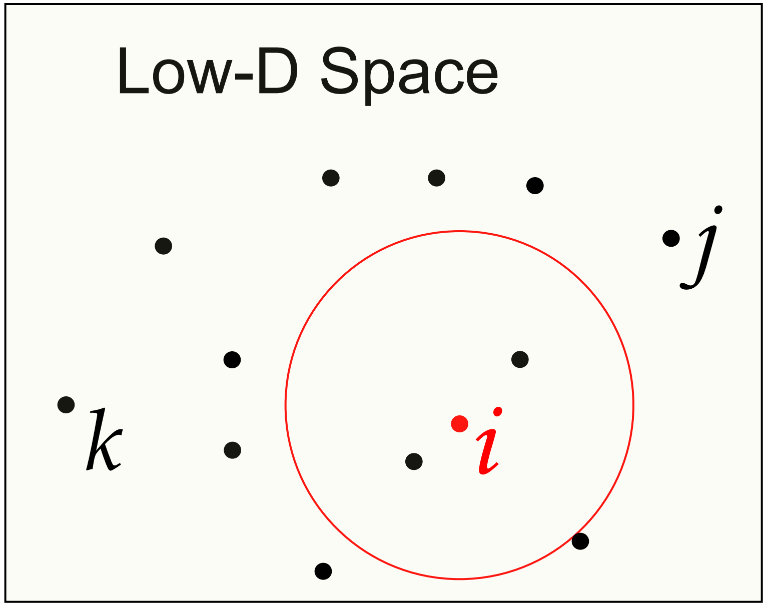
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

This ensures that all the pairwise probabilities sum to 1 so they can be treated as probabilities.

$$\sum_{i,j} p_{ij} = 1$$

Evaluating an arrangement of the points in the low-dimensional space

- Give each data-point a location in the low- dimensional space.
  - Define low-dimensional probabilities symmetrically.
  - Evaluate the representation by seeing how well the low-D probabilities model the high-D affinities.



$$q_{ij} = \frac{e^{-d_{ij}^2}}{\sum_{k<l} e^{-d_{kl}^2}}$$

The cost function for a low-dimensional representation

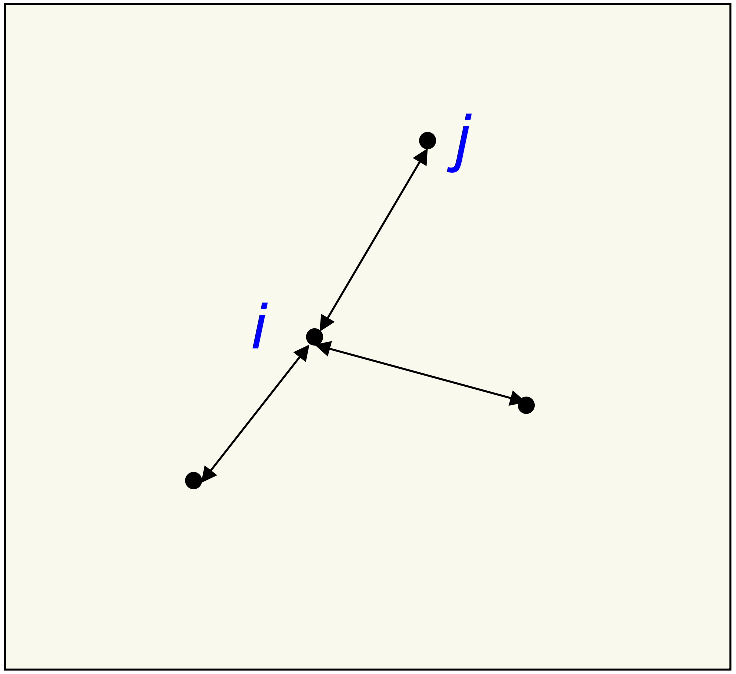
$$Cost \; = KL(P \,\|\, Q) \,| \; = \sum_{i<j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

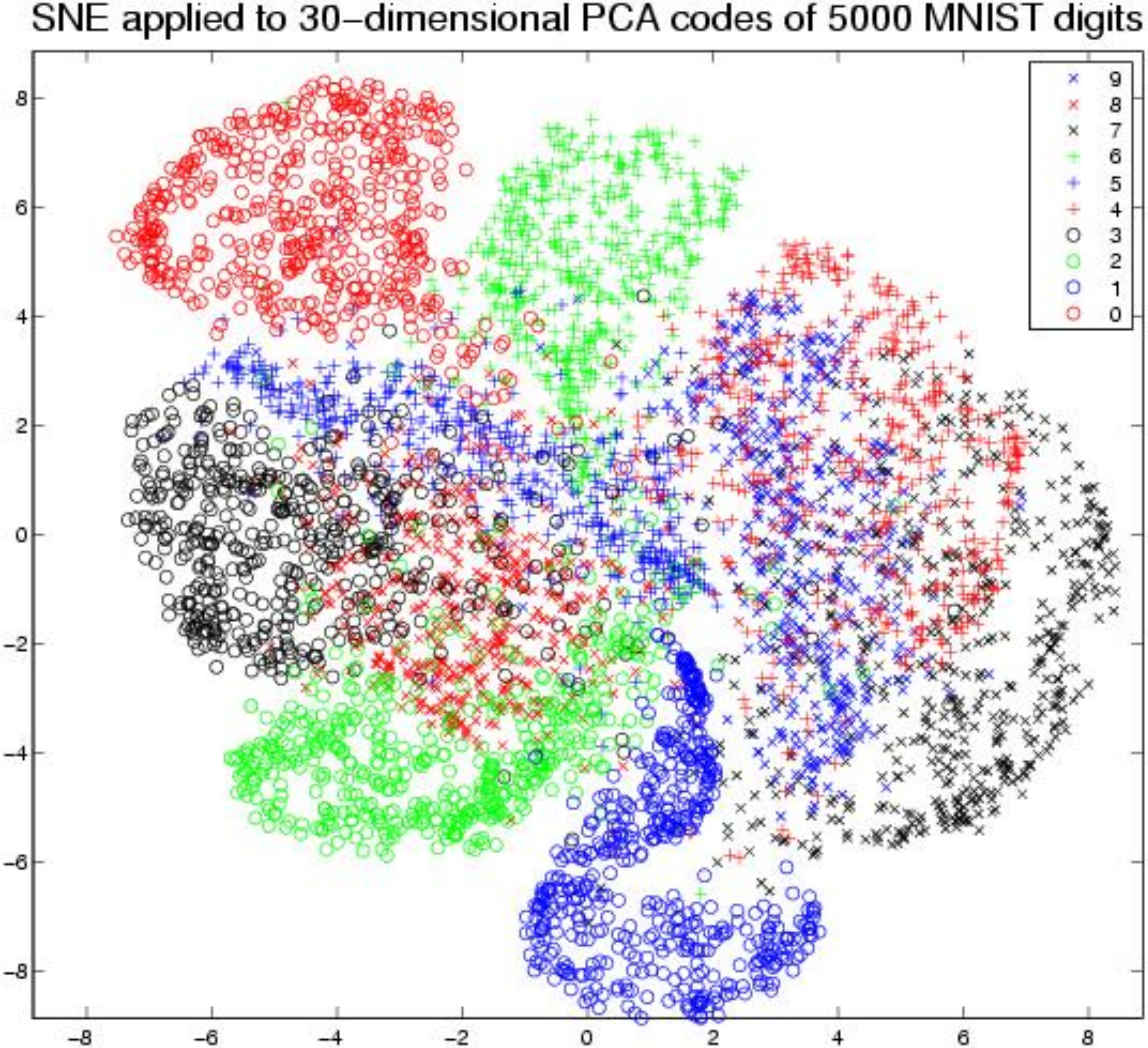
- It's a single KL instead of the sum of one KL for each datapoint.

The forces acting on the low-dimensional points

$$\frac{\partial KL(P \| Q)}{\partial \mathbf{y}_i} = 2\sum_j (\mathbf{y}_i \overset{\substack{\text{extension}\\ \Downarrow}}{-} \mathbf{y}_j)\,(p_{ij} \overset{\substack{\text{stiffness}\\ \Downarrow}}{-} q_{ij})$$

- Points are pulled towards each other if the p’s are bigger than the q’s and repelled if the q’s are bigger than the p’s
    - Its equivalent to having springs whose stiffnesses are set dynamically.




SNE applied to 30−dimensional PCA codes of 5000 MNIST digits
9
8
7
6
5
4
3
2
1
0
8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8

Why SNE does not have gaps between classes

- In the high-dimensional space there are many pairs of points that are moderately close to each other.
  - The low-D space cannot model this. It doesn’t have enough room around the edges.
- So there are many pij’s that are modeled by smaller qij’s.
  - This has the effect of lots of weak springs pulling everything together and crushing different classes together in the middle of the space.
- One solution
  - Use light tail Gaussian kernel for high-D pij but;
  - Heavy tail for low-D qij

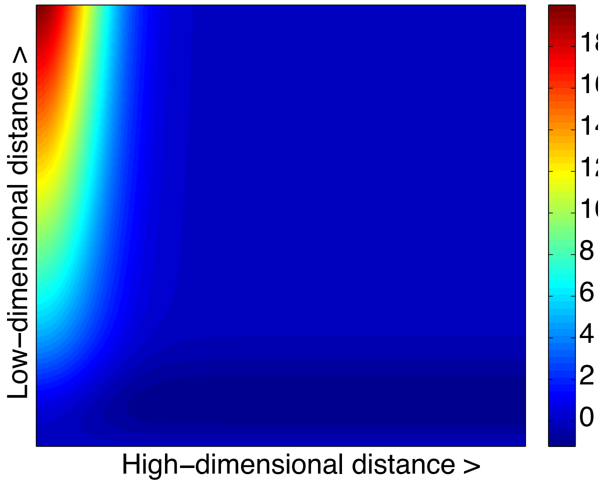
t-SNE

- Use a heavy tailed Student t-distribution (Cauchy) for *q* which allows a moderate distance in high-dimensional space to be faithfully represented by a larger distance (push away) in low-dimensional embedding

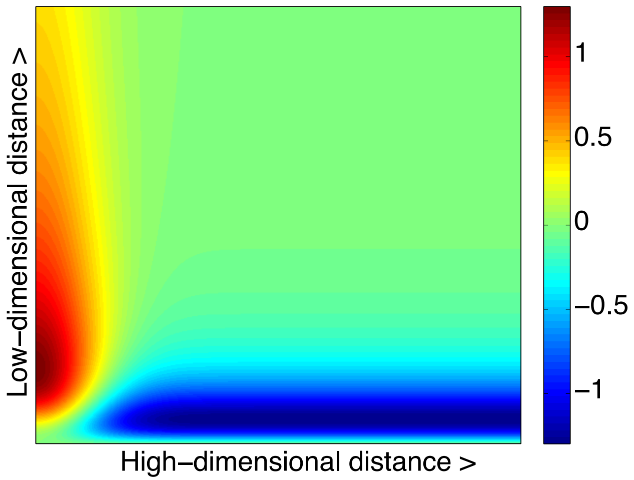
$$q_{ij} \propto \frac{1}{1+d_{ij}^2}$$

Gradient of t-SNE

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$



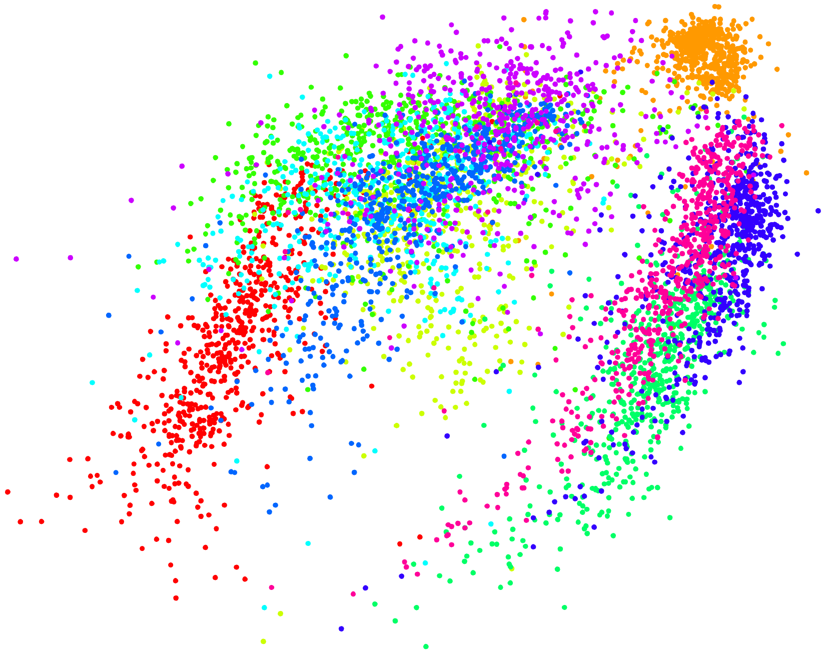
(a) Gradient of SNE.



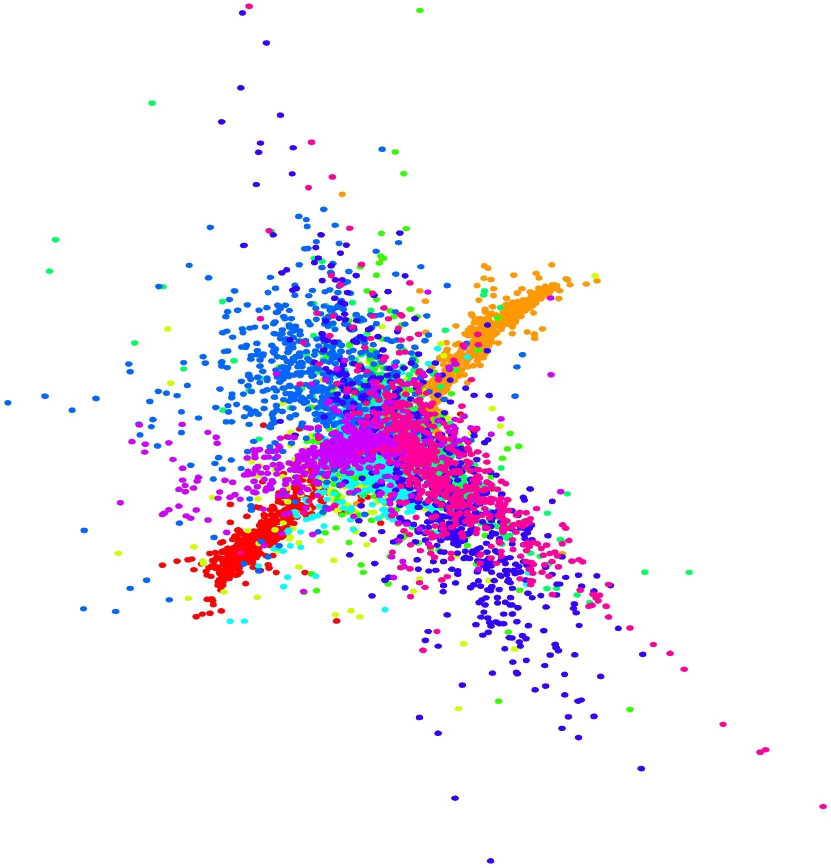
(c) Gradient of t-SNE.

t-SNE allows more points in moderate distance neighbors

Two other state-of-the-art dimensionality reduction methods on the 6000 MNIST digits

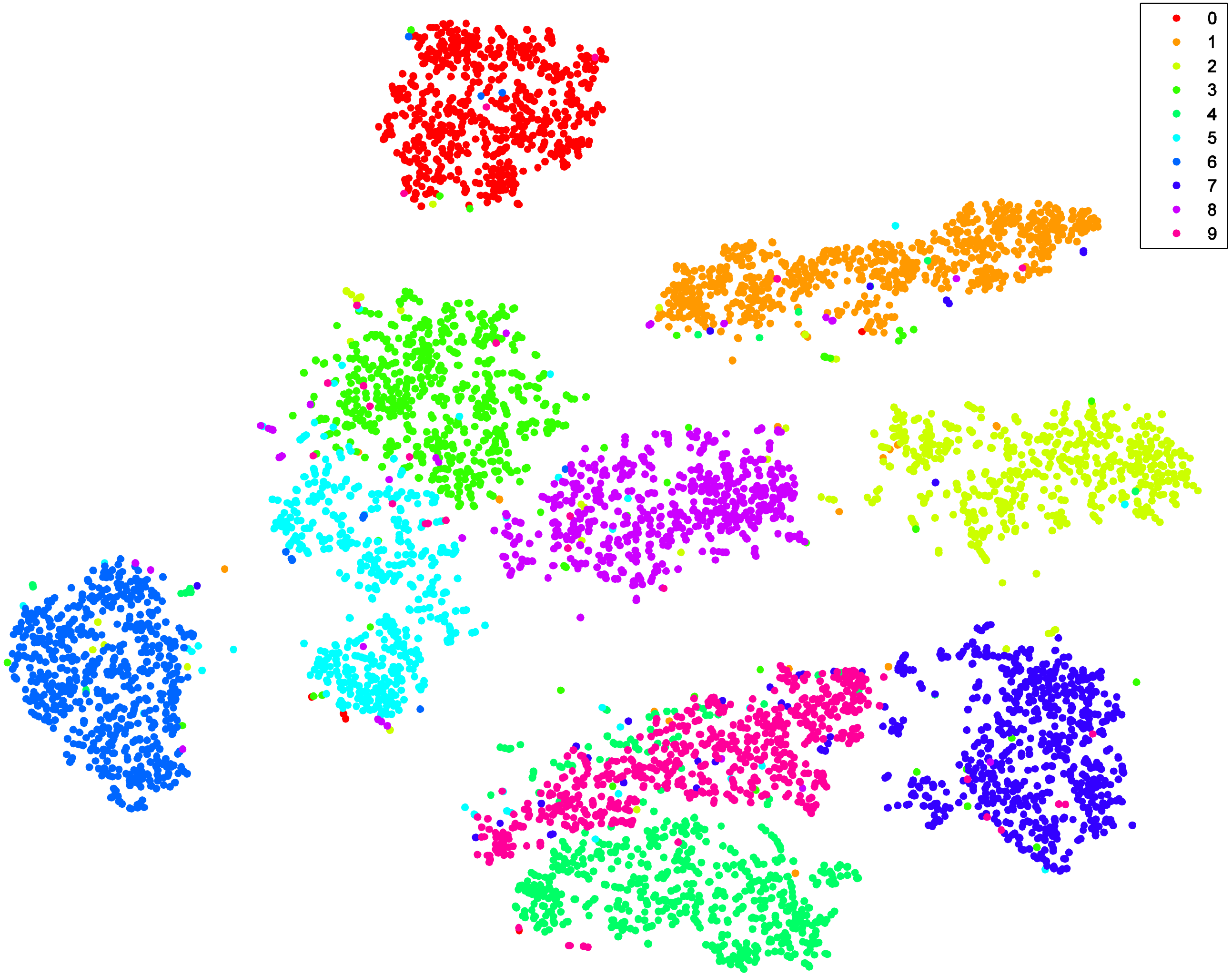


Isomap



Locally Linear Embedding

t-SNE on the 6000 MNIST digits



Stochastic Neighbor Embedding vs. Laplacian Engenmap

- Miguel Carreira-Perpinan (ICML 2010) showed that the original SNE cost function can be rewritten so that it is equivalent to Laplacian Eigenmaps with an extra repulsion term that spreads out the map points.

- This led to a much faster optimization method. The fast code is now on the t-SNE webpage

Stochastic Neighbor Embedding

- The Energy (Loss) function of SNE:

$$E_{\mathrm{SNE}}(\mathbf{X}) = \sum_{n=1}^{N} D_{KL}\left(P_n \| Q_n\right) = \sum_{n,m=1}^{N} p_{nm} \log \frac{p_{nm}}{q_{nm}}$$

where

$$p_{nm} = \frac{\exp\left(-d_{nm}^2\right)}{\sum_{n \neq m'} \exp\left(-d_{nm'}^2\right)}, \qquad p_{nn} = 0$$

$$q_{nm} = \frac{\exp\left(-\left\|\mathbf{y}_n - \mathbf{y}_m\right\|^2\right)}{\sum_{n \neq m'} \exp\left(-\left\|\mathbf{y}_n - \mathbf{y}_{m'}\right\|^2\right)}$$

Stochastic Neighbor Embedding vs. Laplacian Eigenmap

- Expanding the Energy function of SNE and ignoring the terms that do not depend on $\mathbf{y}$, we have

$$E_{\text{SNE}}(\mathbf{Y}) = \sum_{n,m=1}^{N} p_{nm} \left\|\mathbf{y}_n - \mathbf{y}_m\right\|^2 + \sum_{n=1}^{N} \log \sum_{n \neq m} \exp\left(-\left\|\mathbf{y}_n - \mathbf{y}_m\right\|^2\right)$$

- Laplacian Embedding has energy function:

$$E_{\text{LE}}(\mathbf{Y}) = \sum_{n,m=1}^{N} w_{nm} \left\|\mathbf{y}_n - \mathbf{y}_m\right\|^2$$

SNE: attraction and repulsion

- SNE has both the attractive and repulsive terms:
  - The attraction term in both SNE and LE:

$$\sum_{n,m=1}^{N} w_{nm} \|\mathbf{y}_n - \mathbf{y}_m\|^2$$

    pulls points toward each other
  - The repulsion term

$$\sum_{n=1}^{N} \log \sum_{n \neq m} \exp\left(- \|\mathbf{y}_n - \mathbf{y}_m\|^2\right)$$

    push away the points leaving each other

Elastic Embedding

- Define the Energy for EE, by dropping the $\log$ in SNE

$$E_{EE}(\mathbf{Y};\lambda) = \sum_{n,m=1}^{N} w_{nm}^{+} \left\| \mathbf{y}_n - \mathbf{y}_m \right\|^2 + \lambda \sum_{n,m=1}^{N} w_{nm}^{-} \exp\left(- \left\| \mathbf{y}_n - \mathbf{y}_m \right\|^2 \right)$$

  where

  - the attractive weights: $w_{nm}^{+} = \exp\left(-\frac{1}{2} \left\| (\mathbf{x}_n - \mathbf{x}_m)/\sigma \right\|^2\right)$ $(n \neq m)$
  - the repulsive weights: $w_{nm}^{-} = \bar{w}_{nm}^{-} \left\| \mathbf{x}_n - \mathbf{x}_m \right\|^2$ and $\bar{w}_{nm} = 1$ $(n \neq m)$
  - and $w_{nn}^{+} = w_{nn}^{-} = 0$

Gradient

- The gradient of Energy function of EE:

$$\frac{\partial E_{EE}}{\partial \mathbf{y}_n} = 4 \sum_{m \neq n}^{N} w_{nm} \left(\mathbf{y}_n - \mathbf{y}_m \right)$$

- Or in matrix form,

$$\mathbf{G}(\mathbf{Y}; \lambda) = \frac{\partial E_{EE}}{\partial \mathbf{Y}} = 4\mathbf{Y} \left(\mathbf{L}^{+} - \lambda \widetilde{\mathbf{L}}^{-} \right) = 4\mathbf{Y}\mathbf{L}$$

where we define the affinities

$$\begin{aligned} \tilde{w}_{nm}^{-} &= w_{nm}^{-} \exp \left(- \left\| \mathbf{y}_n - \mathbf{y}_m \right\|^2 \right) \\ w_{nm} &= w_{nm}^{+} - \lambda \widetilde{w}_{nm}^{-} \end{aligned}$$

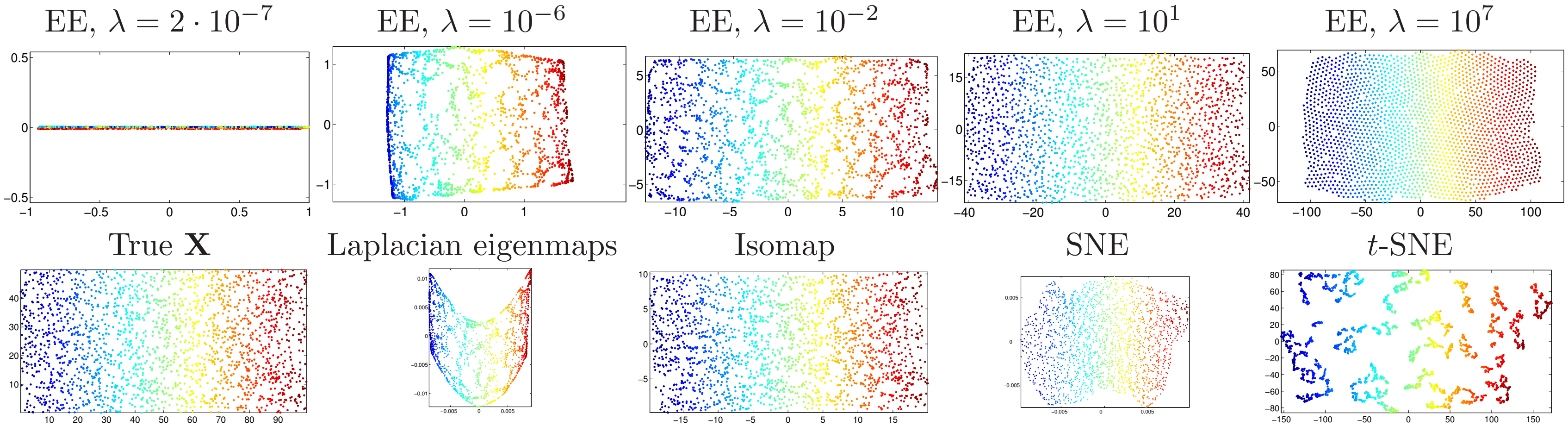
and their graph Laplacians $\tilde{\mathbf{L}} = \widetilde{\mathbf{D}} - \widetilde{\mathbf{W}}, \mathbf{L} = \mathbf{D} - \mathbf{W}$ in the usual way

Remark

- $\mathbf{L}^+$ is the usual (unnormalised) graph Laplacian that appears in Laplacian eigenmaps.

- W can be considered a learned affinity matrix and contains negative weights for $\lambda > 0$.

Swiss Roll Example


EE, $\lambda = 2 \cdot 10^{-7}$

EE, $\lambda = 10^{-6}$

EE, $\lambda = 10^{-2}$

EE, $\lambda = 10^{1}$

EE, $\lambda = 10^{7}$

True $\mathbf{X}$

Laplacian eigenmaps

Isomap

SNE

t-SNE

Figure 3. Swiss roll. *Top*: EE with homotopy; we show $\mathbf{X}$ for different λ. *Bottom*: true $\mathbf{X}$ and results with other methods.

Reference

- Tenenbaum, de Silva, and Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 290:2319-2323, 22 Dec. 2000.
- Roweis and Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 290:2323-2326, 22 Dec. 2000.
- M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation, 15:1373–1396, 2003.
- D. L. Donoho and C. Grimes. Hessian eigenmaps: New locally linear embedding techniques for high-dimensional data. PNAS 100 (10): 5591–5596 2003.
- R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. PNAS 102 (21):7426-7431, 2005.
- M. Hein, J.-Y. Audibert, and U. von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. In P. Auer and R. Meir, editors, Proc. of the 18th Conf. on Learning Theory (COLT), pages 486–500, Berlin, 2005. Springer.
- Ulrike von Luxburg, Mikhail Belkin, and Olivier Bousquet. Consistency of spectral clustering. Ann. Statist. Volume 36, Number 2 (2008), 555-586.

Reference

- M. Belkin and P. Niyogi. Convergence of Laplacian Eigenmaps. 2006. Short version NIPS 2008.
- Singer, Amit. From graph to manifold Laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*. 2006.
- Zhenyue Zhang and Hongyuan Zha, Principal Manifolds and Nonlinear Dimension Reduction via Local Tangent Space Alignment, SIAM Journal of Scientific Computing, 2002
- Singer, Amit and Hau-Tieng Wu, Vector Diffusion Map and the Connection Laplacian. Communications on Pure and Applied Mathematics, 65 (8):1067-1144, 2012. Matlab VDM codes downloaded at
  - https://sites.google.com/site/hautiengwu/home/download

Acknowledgement

- Slides stolen from M. Belkin, R. Coifman, G. Hinton, et al.