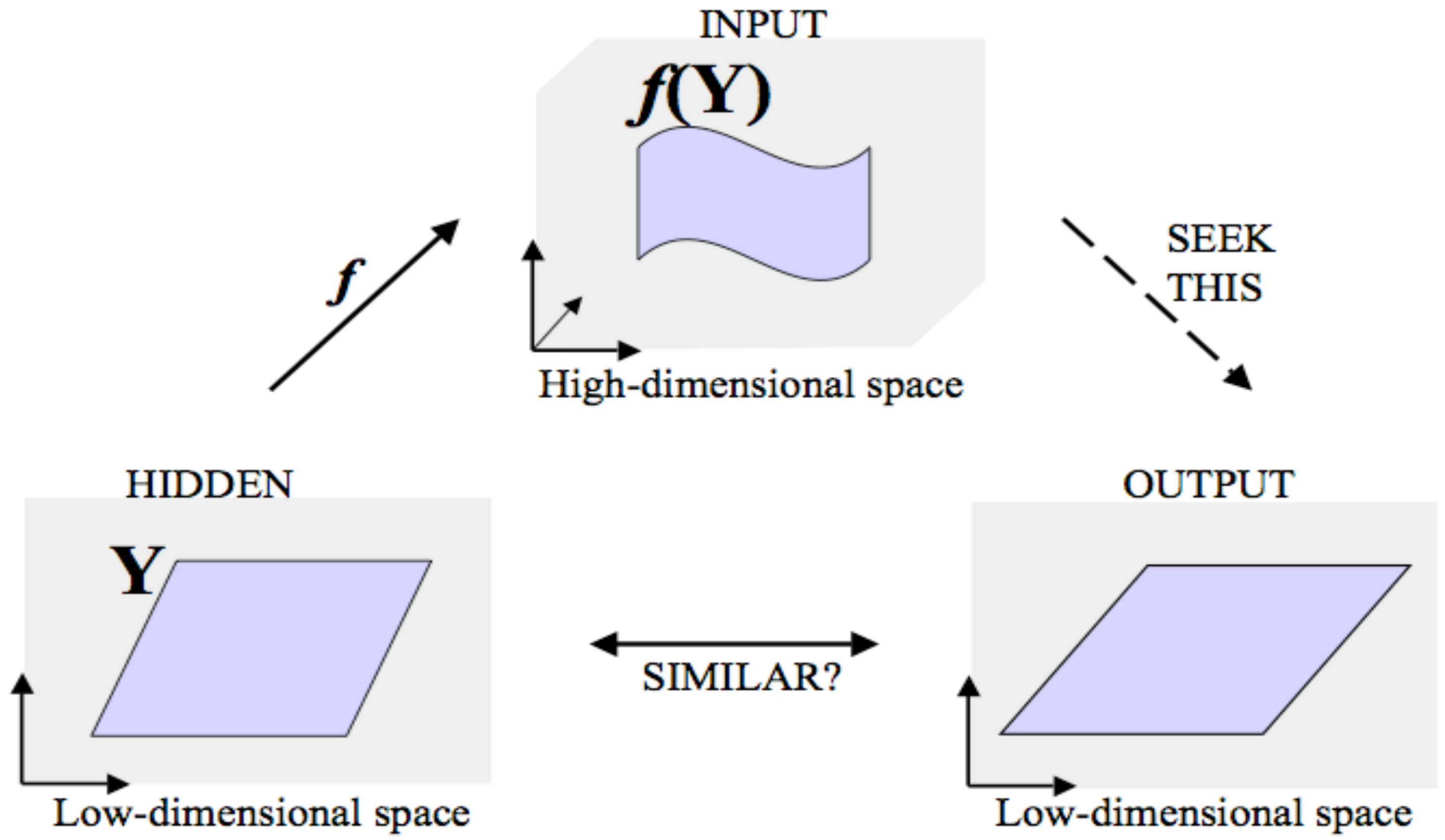
Manifold Learning II: Extended Locally Linear Embedding

姚 遠

2017

Generative Models in Manifold Learning



Spectral Geometric Embedding

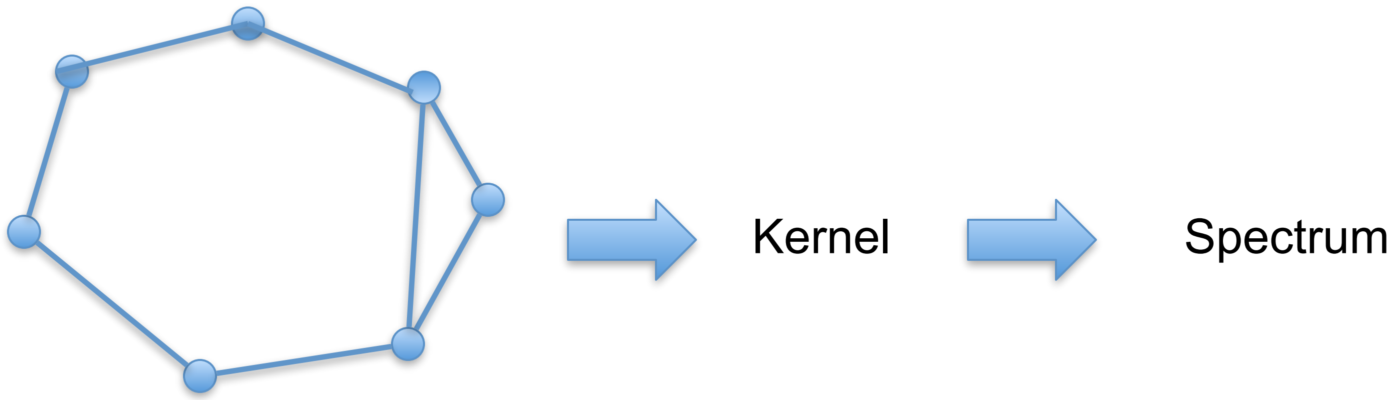
Given $x_1, \ldots, x_n \in \mathcal{M} \subset \mathbb{R}^N$,
Find $y_1, \ldots, y_n \in \mathbb{R}^d$ where $d << N$

- 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



Recall: ISOMAP

1. Construct Neighborhood Graph.
2. Find shortest path (geodesic) distances.

D_{ij} is $n \times n$

3. Embed using Multidimensional Scaling.

Recall: LLE

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

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

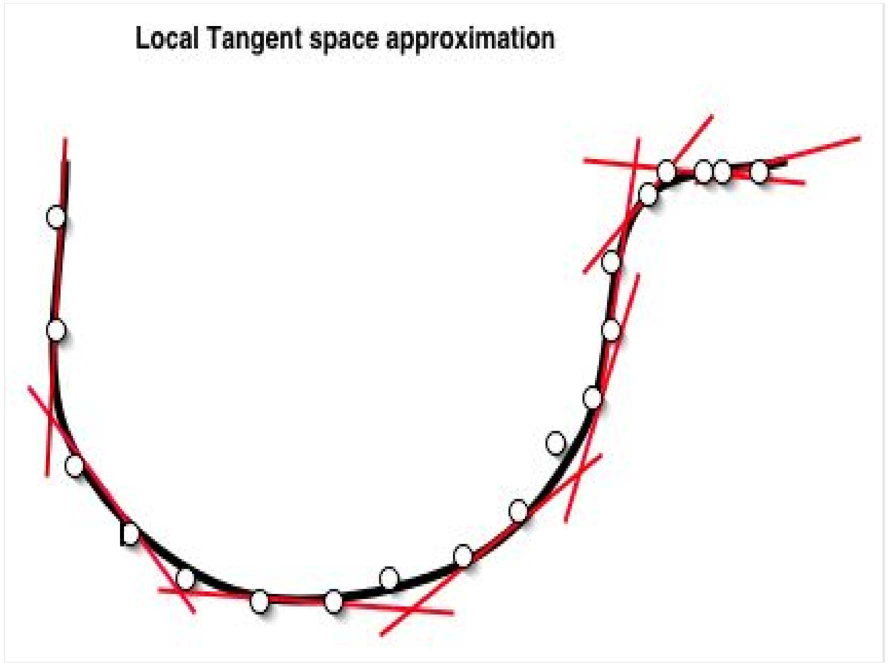
- Compute Embedding

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

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

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

Local Tangent Space Alignment



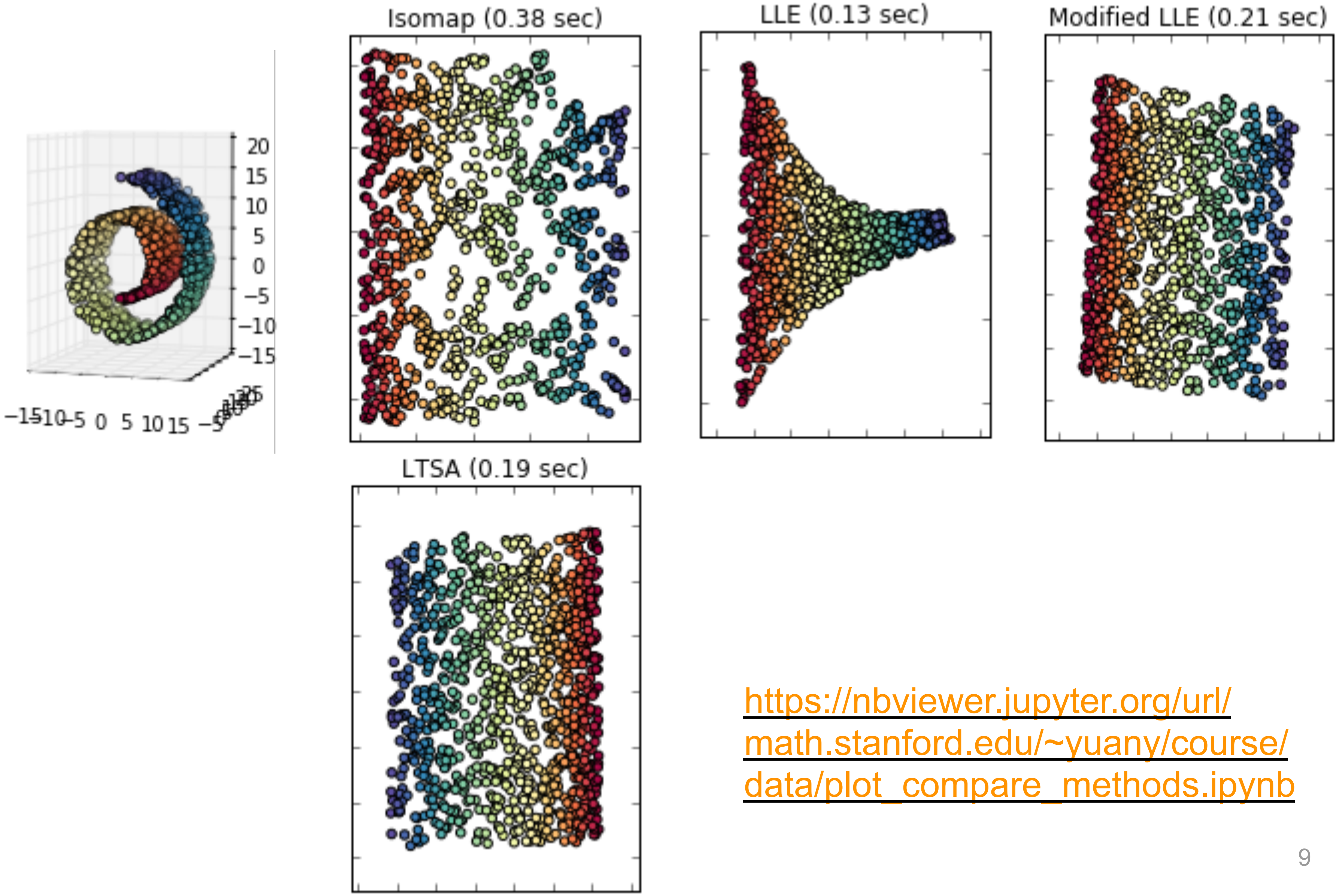
Find a good approximation of tangent space of curve using discrete samples.
— Principal curve/manifold (Hastie-Stuetzle’89, Zha-Zhang’02)

Recall LTSA (Zha-Zhang'02)

Algorithm 6: LTSA Algorithm

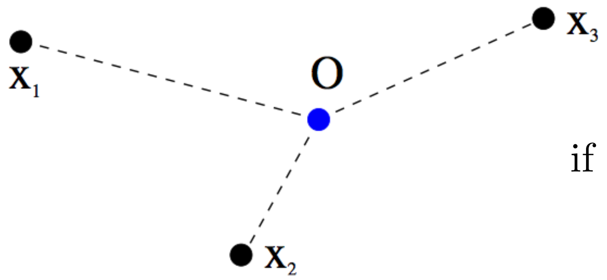
```
  Input: A weighted undirected graph $G = (V, E)$ such that
1 $V = \{x_i \in \mathbb{R}^p : i = 1, \dots, n\}$
2 $E = \{(i, j) :$ if $j$ is a neighbor of $i$, i.e. $j \in \mathcal{N}_i\}$, e.g. $k$-nearest neighbors
  Output: Euclidean $d$-dimensional coordinates $Y = [y_i] \in \mathbb{R}^{k \times n}$ of data.
3 Step 1 (local PCA): Compute local SVD on neighborhood of $x_i$, $x_{i_j} \in \mathcal{N}(x_i)$,
      $$\tilde{X}^{(i)} = [x_{i_1} - \mu_i, ..., x_{i_k} - \mu_i]^{p \times k} = \tilde{U}^{(i)} \tilde{\Sigma} (\tilde{V}^{(i)})^T,$$
    where $\mu_i = \sum_{j=1}^k x_{i_j}$. Define
      $$G_i = [1/\sqrt{k}, \tilde{V_1}^{(i)}, ..., \tilde{V_d}^{(i)}]^{k \times (d+1)};$$
4 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 $S_i^{n \times k} : [x_{i_1}, ..., x_{i_k}] = [x_1, ..., x_n] S_i^{n \times k}$;
5 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.
```

Comparisons on Swiss Roll



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

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

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

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

Hessian Null

The Hessian matrix

$$(\mathcal{H}y)(0) := \left[\frac{\partial^2 y(x)}{\partial x(i)\partial x(j)}\right]_{x=0} = 0,$$

if function $y(x)$ is a linear transform of the coordinates $x \in \mathbb{R}^p$ in the tangent space at x_i. In this case $(I - W)y(0) = 0$ and y reaches a minimizer.

In other words, the kernel of $(\mathcal{H}y)$ has dimension $d+1$, consisting the constant function and d linearly independent coordinates. Inspired by such an observation, Donoho and Grimes [**DG03b**] proposed Hessian LLE (Eigenmap) in search of

$$\min_{y \perp \mathbf{1}} \int \|\mathcal{H}y\|^2, \quad \|y\| = 1.$$

Hessian LLE Algorithm (I)

```
Algorithm 7: Hessian LLE Algorithm
  Input: A weighted undirected graph $G = (V, E, d)$ such that
1 $V = \{x_i \in \mathbb{R}^p : i = 1, \ldots, n\}$
2 $E = \{(i, j) :$ 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.
3 Step 1: Compute local PCA on neighborhood of $x_i$, for,
      $\tilde{X}^{(i)} = [x_{i_1} - \mu_i, ..., x_{i_k} - \mu_i]^{p\times k} = \tilde{U}^{(i)}\tilde{\Sigma}(\tilde{V}^{(i)})^T, \quad x_{i_j} \in \mathcal{N}(x_i),$
   where $\mu_i = \sum_{j=1}^{k} x_{i_j} = \frac{1}{k} X_i \mathbf{1}$;
```

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

Continued...

Hessian LLE Algorithm (II)

4 ***Step 2***: Null Hessian estimation: define

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

where $\tilde{V}_i\tilde{V}_j = [\tilde{V}_{ik}\tilde{V}_{jk}]^T \in \mathbb{R}^k$ 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

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

Define

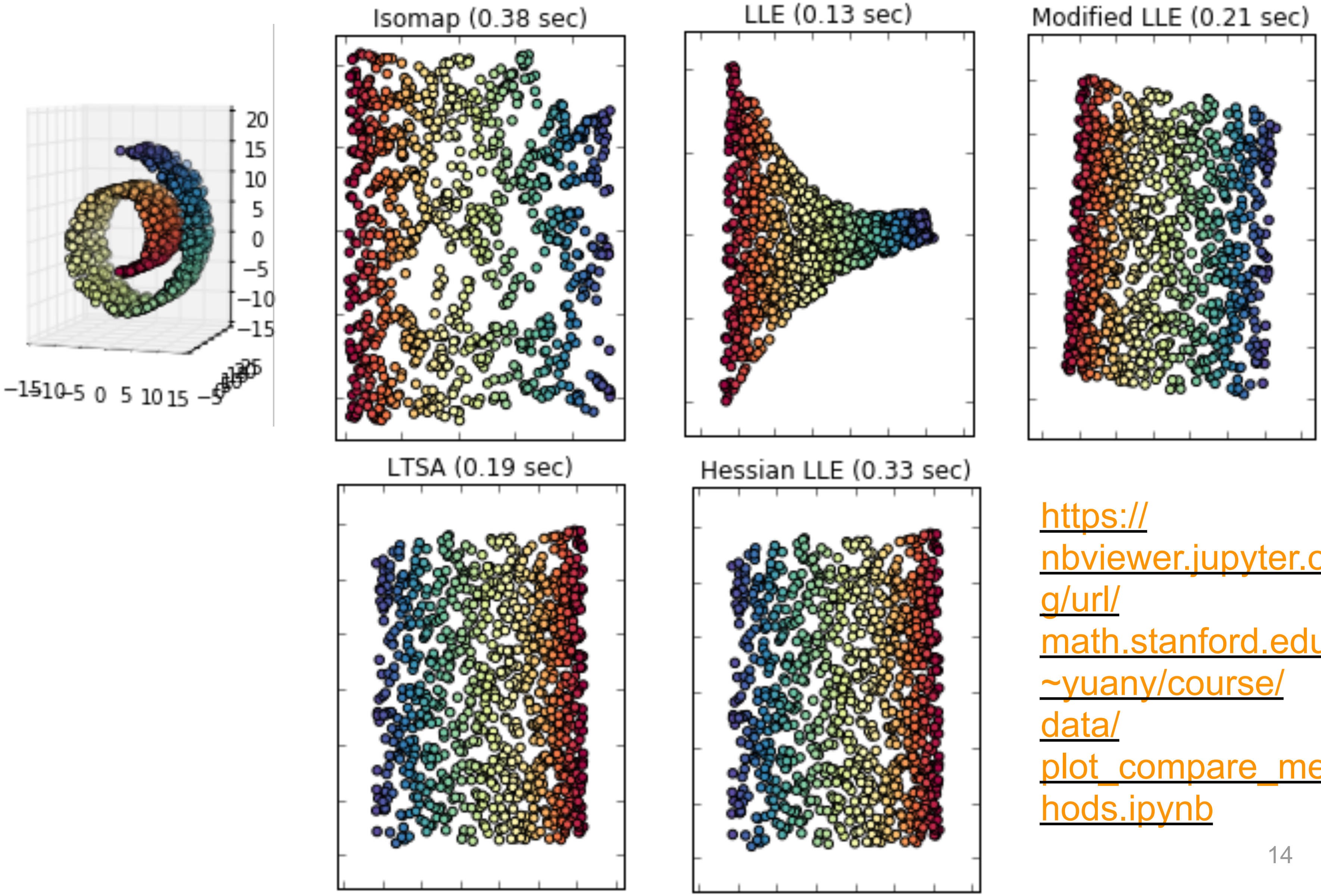
$$[H^{(i)}]^T = [last \quad \binom{d}{2} \quad columns \quad of \quad \tilde{M}]_{k\times\binom{d}{2}}.$$

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.

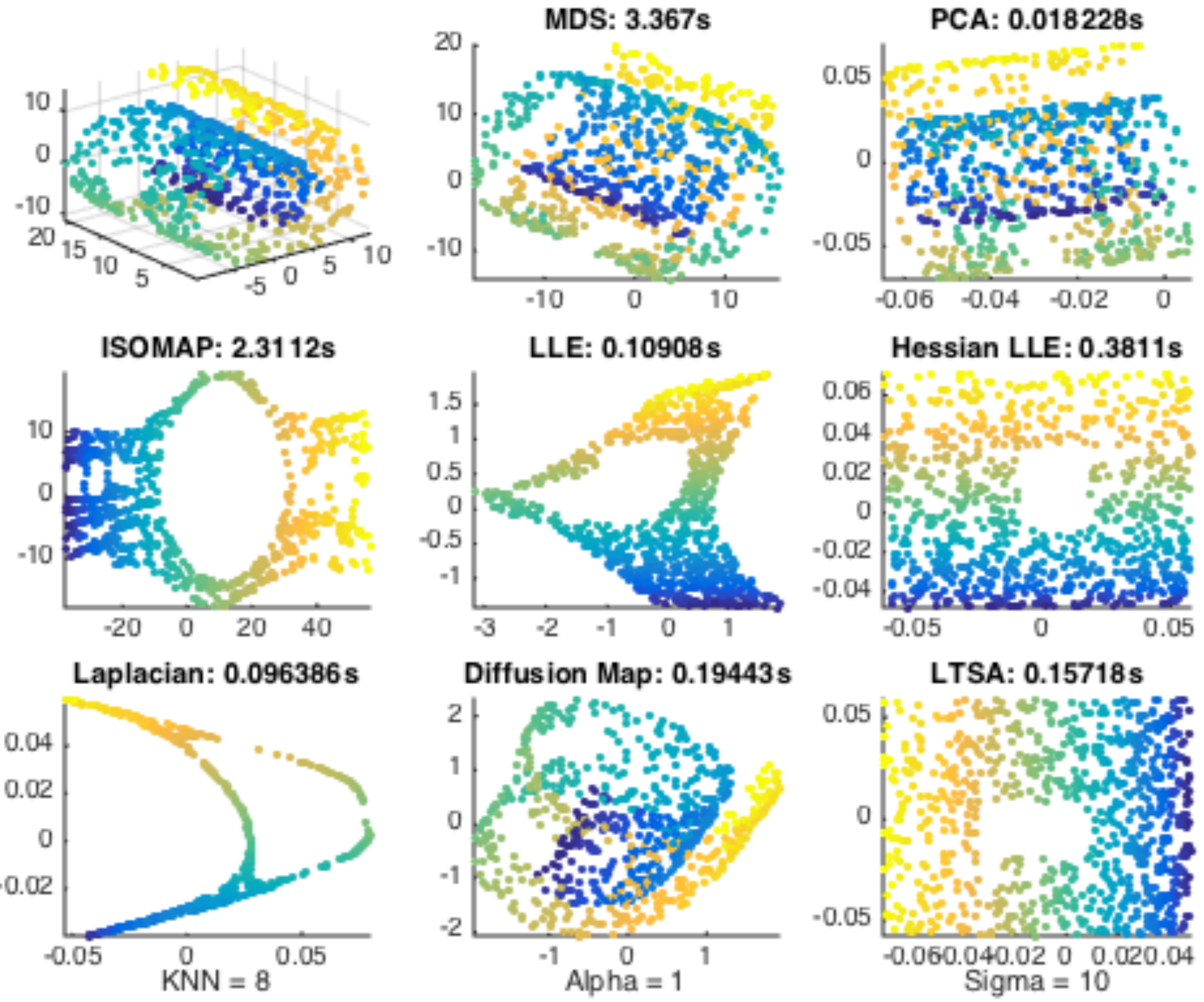
Comparisons on Swiss Roll



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

Comparisons on Swiss Roll with a Hole

- mani.m



Two Assumptions on ISOMAP

(ISO1) *Isometry.* The mapping ψ preserves geodesic distances. That is, define a distance between two points m and m' on the manifold according to the distance travelled by a bug walking along the manifold M according to the shortest path between m and m'. Then the isometry assumption says that

$$G(m, m') = |\theta - \theta'|, \qquad \forall m \leftrightarrow \theta, m' \leftrightarrow \theta',$$

where $|\cdot|$ denotes Euclidean distance in $\mathbb{R}^d$.

(ISO2) *Convexity.* The parameter space Θ is a convex subset of $\mathbb{R}^d$. That is, if θ, θ' is a pair of points in Θ, then the entire line segment $\{(1-t)\theta + t\theta' : t \in (0,1)\}$ lies in Θ.

Convexity is hard to meet: consider two balls in an image which never intersect, whose center coordinate space (x_1,y_1,x_2,y_2) must have a hole.

Relaxations (Donoho-Grimes'2003)

(LocISO1) *Local Isometry.* In a small enough neighborhood of each point m, geodesic distances to nearby points m' in M are identical to Euclidean distances between the corresponding parameter points θ and θ'.

(LocISO2) *Connectedness.* The parameter space Θ is a open connected subset of $\mathbb{R}^d$.

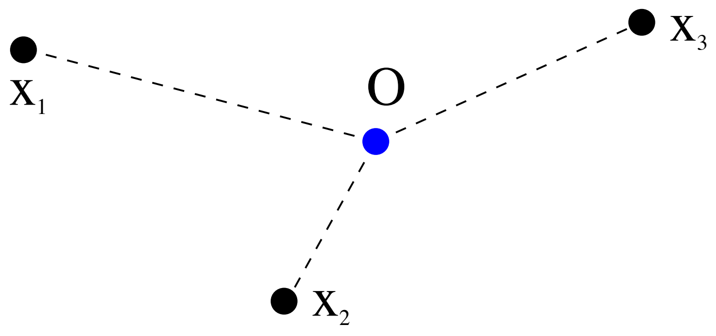
Convergence of Hessian LLE (Donoho-Grimes)

Theorem 1 *Suppose $M = \psi(\Theta)$ where Θ is an open connected subset of $\mathbb{R}^d$, and ψ is a locally isometric embedding of Θ into $\mathbb{R}^n$. Then $\mathcal{H}(f)$ has a $d+1$ dimensional nullspace, consisting of the constant function and a d-dimensional space of functions spanned by the original isometric coordinates.*

We give the proof in Appendix A.

Corollary 2 *Under the same assumptions as Theorem 1, the original isometric coordinates θ can be recovered, up to a rigid motion, by identifying a suitable basis for the null space of $\mathcal{H}(f)$.*

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…

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

But $L = D - W, \quad D = \mathrm{diag}(\sum_{j \in \mathcal{N}_i} w_{ij})$

$$\frac{1}{2}\sum_{i,j}(y_i - y_j)^2 W_{ij} = \mathbf{y}^T L \mathbf{y}$$

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

$$= \sum_i y_i^2 D_{ii} + \sum_j y_j^2 D_{jj} - 2\sum_{i,j} y_i y_j W_{ij}$$

$$= 2\mathbf{y}^T L \mathbf{y}$$

Embedding of Unnormalized Laplacian Eigenmap

$$\lambda = 0 \rightarrow \mathbf{y} = \mathbf{1}$$

Uniform sampling: $$\min_{\mathbf{y}^T\mathbf{1}=0} \mathbf{y}^T L \mathbf{y}$$

Let $Y = [\mathbf{y}_1 \mathbf{y}_2 \ldots \mathbf{y}_m]$

$$\sum_{i,j} ||Y_i - Y_j||^2 W_{ij} = \text{trace}(Y^T L Y)$$

subject to $Y^T Y = I$.

Use eigenvectors of L to embed.

Nonuniform (GEV): $$\underset{\substack{\mathbf{y} \\ \mathbf{y}^T D \mathbf{y}=1}}{\text{argmin}}\, \mathbf{y}^T L \mathbf{y}.$$

How to find weights? Heat kernels…

- $H_t(x,y) = \sum_i e^{-\lambda_i t} \phi_i(x)\phi_i(y)$
- 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

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

good for small *t*

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

Laplacian Eigenmaps (I) [Belkin-Niyogi]

Algorithm 8: Laplacian Eigenmap

```
   Input: An adjacency graph $G = (V, E, d)$ such that
1  $V = \{x_i : i = 1, \ldots, n\}$
2  $E = \{(i, j) :$ if $j$ is a neighbor of $i$, i.e. $j \in \mathcal{N}_i\}$, e.g. $k$-nearest neighbors,
    $\epsilon$-neighbors
3  $d_{ij} = d(x_i, x_j)$, 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.
4  Step 1: Choose weights
5  (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.
6  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.

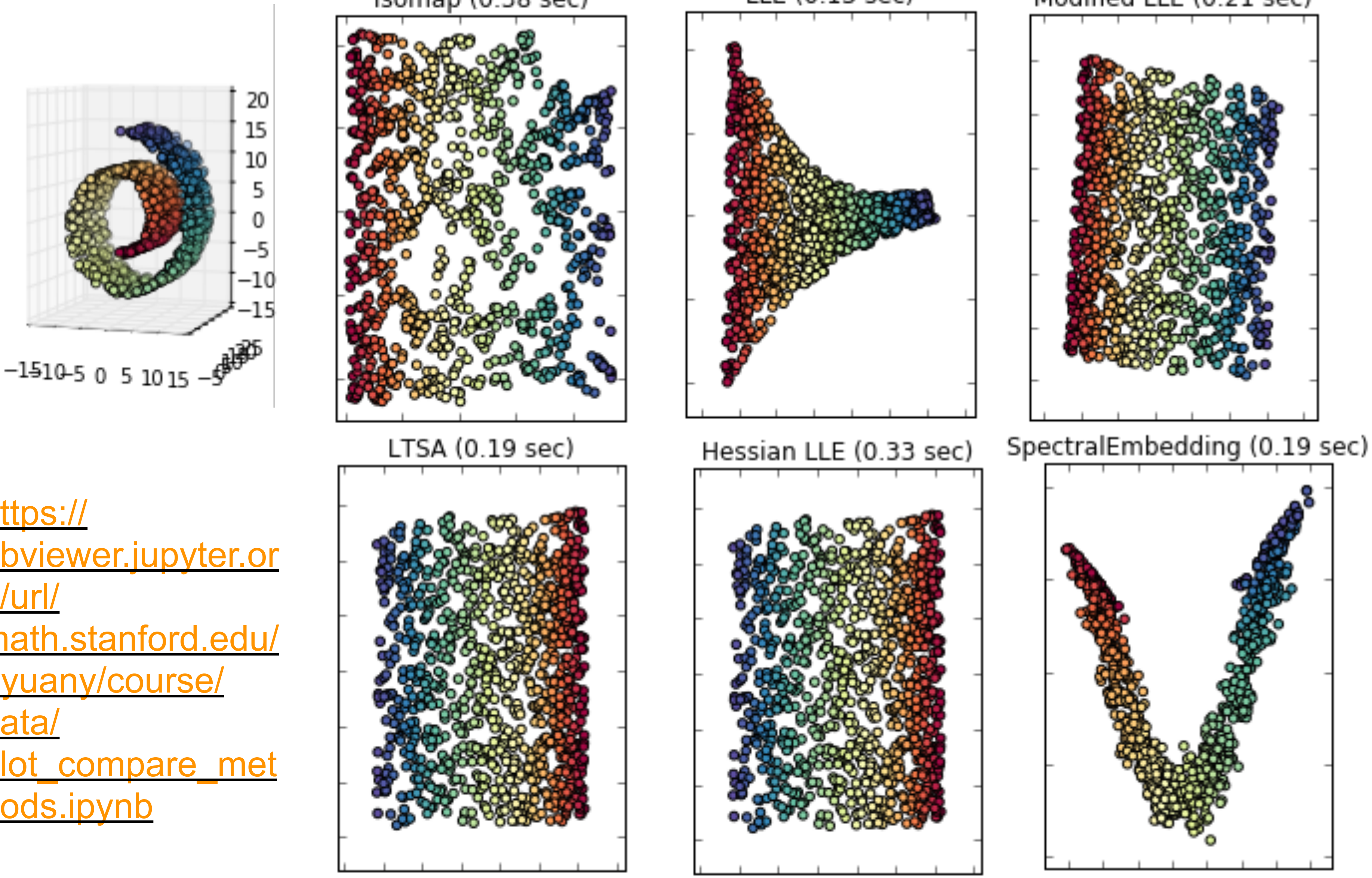
- Hessian LLE

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

- 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

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 = 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 $w_{ij} = w_{ji} \Rightarrow \pi_i p_{ij} = \pi_j p_{ji}$

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

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

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

Spectrum of P

- Eigenvalues of S and P are the same, so

$$|\lambda_i| \le 1$$

- 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

- For primitive *P*

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

- Diffusion map of x_i is defined via right eigenvectors

$$\Phi_t(x_i) = \begin{pmatrix} \lambda_1^t \phi_1(i) \\ \lambda_2^t \phi_2(i) \\ \vdots \\ \lambda_n^t \phi_n(i) \end{pmatrix} \in R^n$$

- Laplacian LLE is the special case with *t*=0 and top d+1 eigenvectors

Dimensionality Reduction

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

$$\left|\lambda_i^t\right| \geq 1-\delta, \qquad i = 1,\ldots,m\,,$$

$$\left|\lambda_k^t\right| < 1-\delta, \qquad k > m$$

- Define

$$\Phi_t^\delta\left(x_i\right) = \begin{pmatrix} \lambda_2^t \phi_2(i) \\ \lambda_3^t \phi_3(i) \\ \mathrm{M} \\ \lambda_m^t \phi_m(i) \end{pmatrix} \in R^{m-1}$$

- Varying *t or δ* leads to a multiscale analysis

Diffusion Distance

- Define the diffusion distance between points at scale *t*

$$D_t(x_i,x_j) := \left\|\Phi_t\left(x_i\right)-\Phi_t\left(x_j\right)\right\|_{l^2} \cong \sum_{k=2}^{m} \lambda_k^t(\phi_k(x_i)-\phi_k(x_j))^2$$

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

$$D_t(x_i,x_j) := \left\|P_{i*}^t - P_{j*}^t\right\|_{l^2(1/d)} = \sum_{k=2}^{m} \frac{(P_{ik}^t - P_{jk}^t)^2}{d_k}$$

Diffusion Distance Example

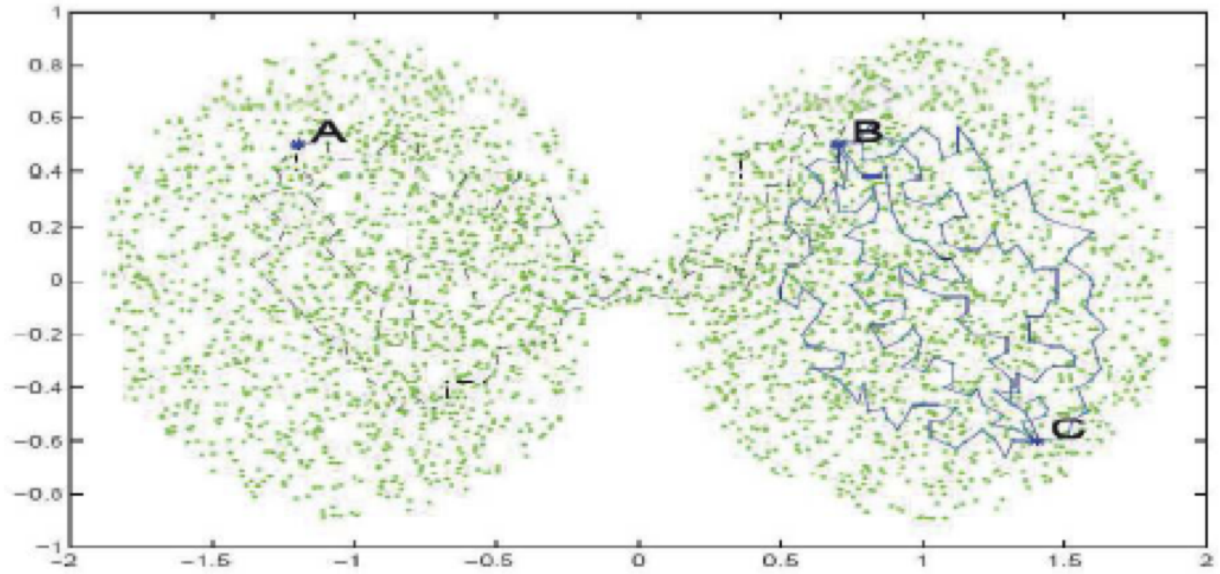


FIGURE 1. Diffusion Distances $d_t(A, B) >> d_t(B, C)$ while graph shortest path $d_{geod}(A, B) \sim d_{geod}(B, C)$.

General Diffusion Map

This image cannot currently be displayed.

- Gaussian kernel

$$K_\varepsilon(x,y) = \exp\left(-\frac{\|x-y\|^2}{\varepsilon^2}\right)$$

- Normalize kernel

$$K^{(\alpha)}(x,y) = \frac{K_\varepsilon(x,y)}{p^\alpha(x)p^\alpha(y)} \quad where \quad p(x) = \int K_\varepsilon(x,y)d\mu(y)$$

- Renormalized kernel

$$A_\varepsilon(x,y) = \frac{K^{(\alpha)}(x,y)}{\sqrt{d^{(\alpha)}(x)}\sqrt{d^{(\alpha)}(y)}} \quad where \quad d^{(\alpha)}(x) = \int K^{(\alpha)}(x,y)d\mu(y)$$

  - α=1, Laplacian-Beltrami operator, separate geometry from density
  - α=0, classical normalized graph Laplacian
  - α=1/2, backward Fokkar-Planck operator

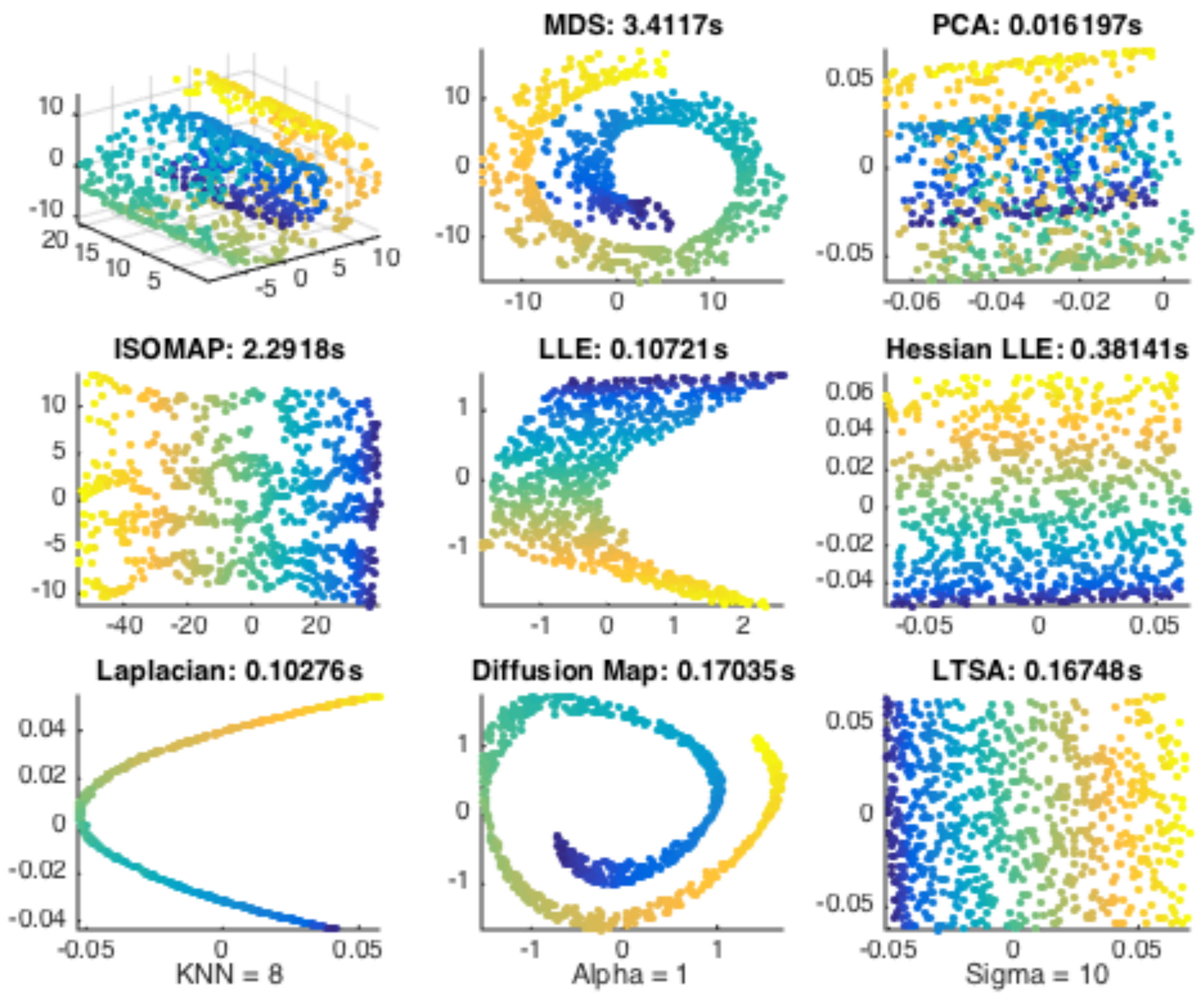
Coifman-Lafon 2006. Diffusion Maps.

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



Diffusion Map vs. Stochastic Neighbor Embedding

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

$$D_t(x_i, x_j) := \left\| P_{i*}^t - P_{j*}^t \right\|_{l^2(1/d)} = \sum_{k=2}^{m} \frac{(P_{ik}^t - P_{jk}^t)^2}{d_k}$$

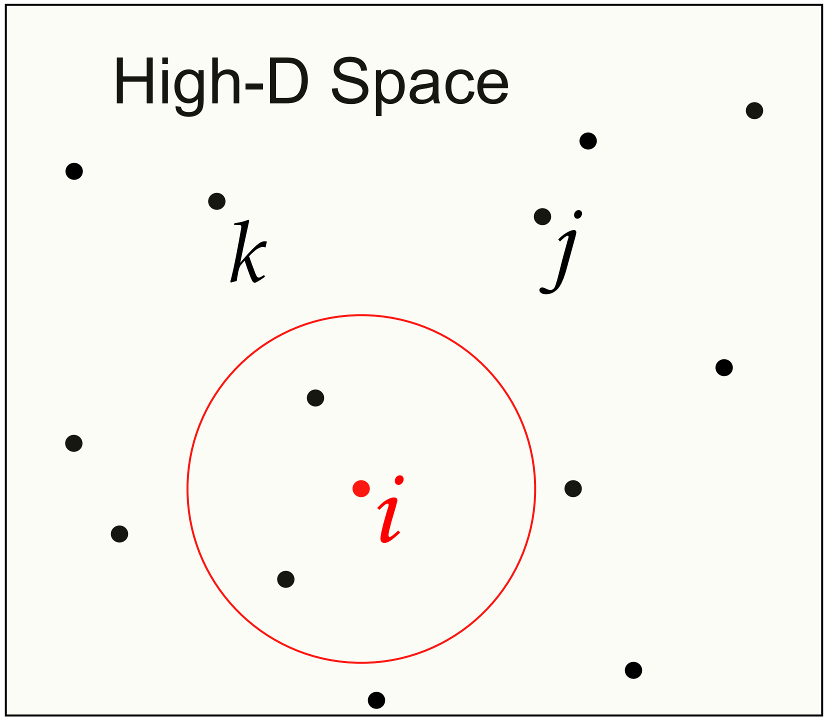
- SNE considers to find a low-dimensional Euclidean embedding *Y* which preserves the distribution P_{i*}

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.

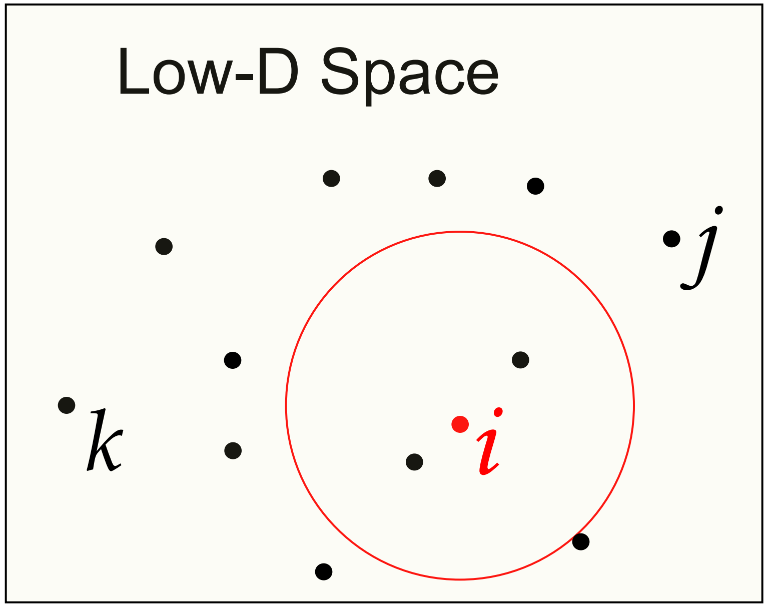


$$p_{j|i} = \frac{e^{-d_{ij}^2 / 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 / 2\sigma_i^2}}$$

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.

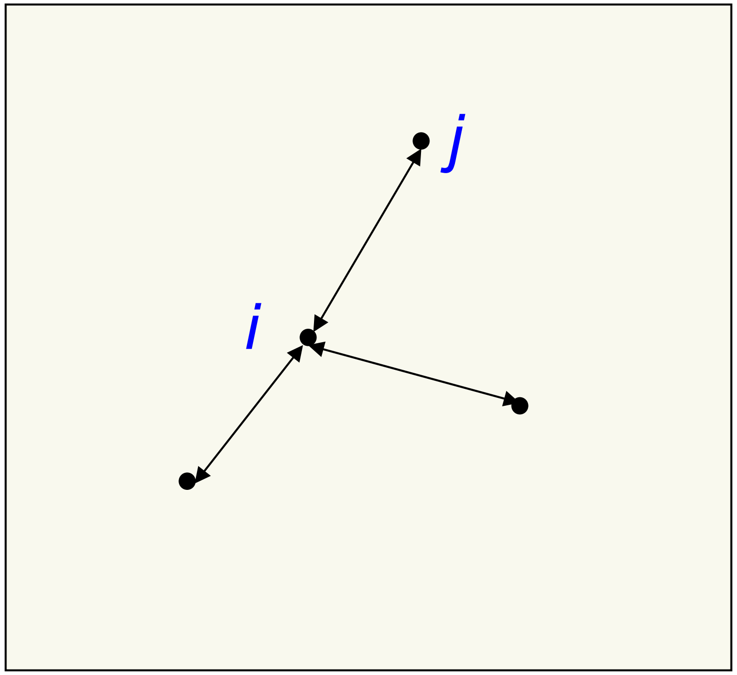
Gradient Descent

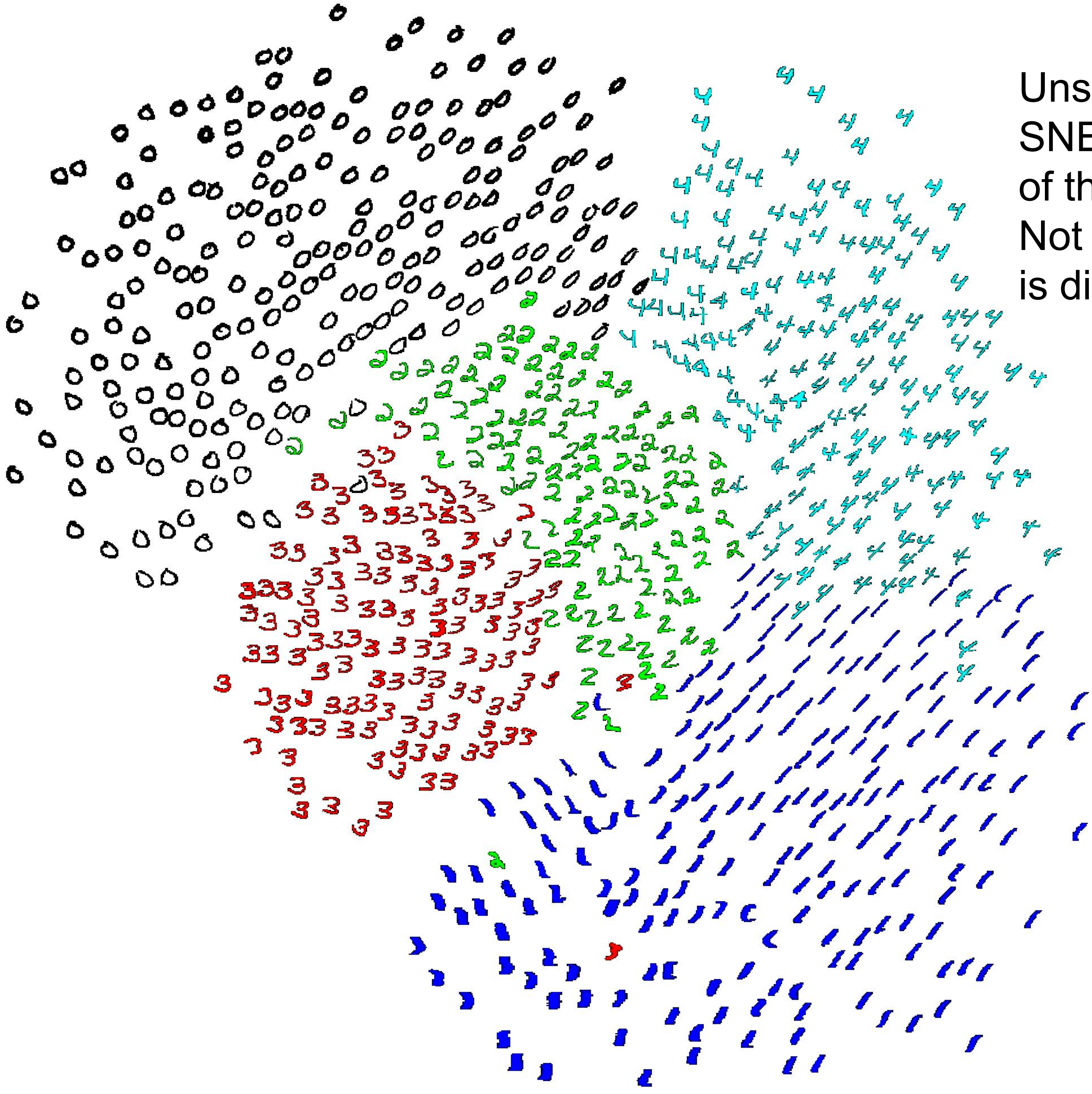
$$\frac{\partial Cost}{\partial \mathbf{y}_i} = 2\sum_j (\mathbf{y}_j - \mathbf{y}_i)(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$$

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

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

- 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.

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

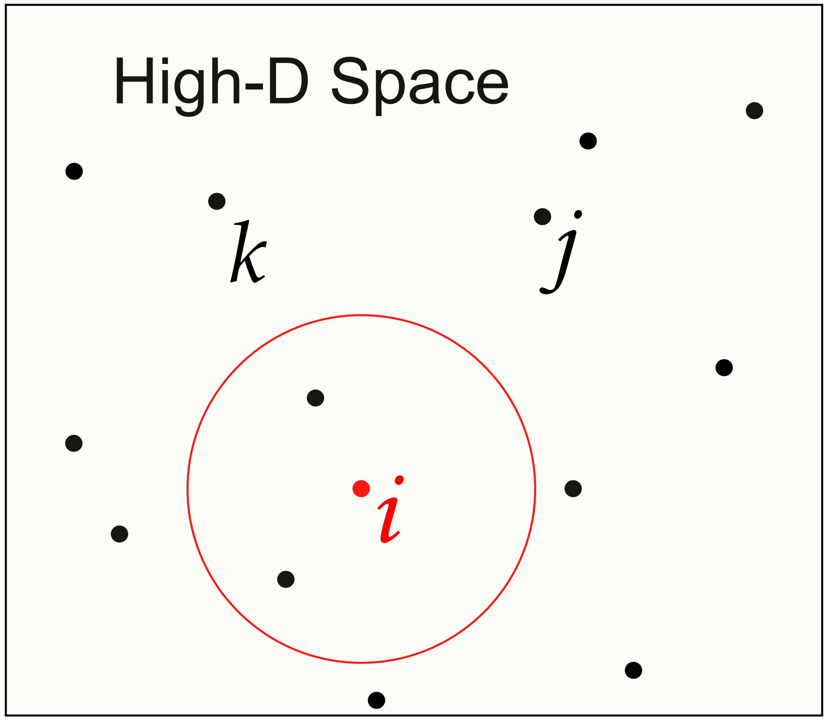
$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

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).

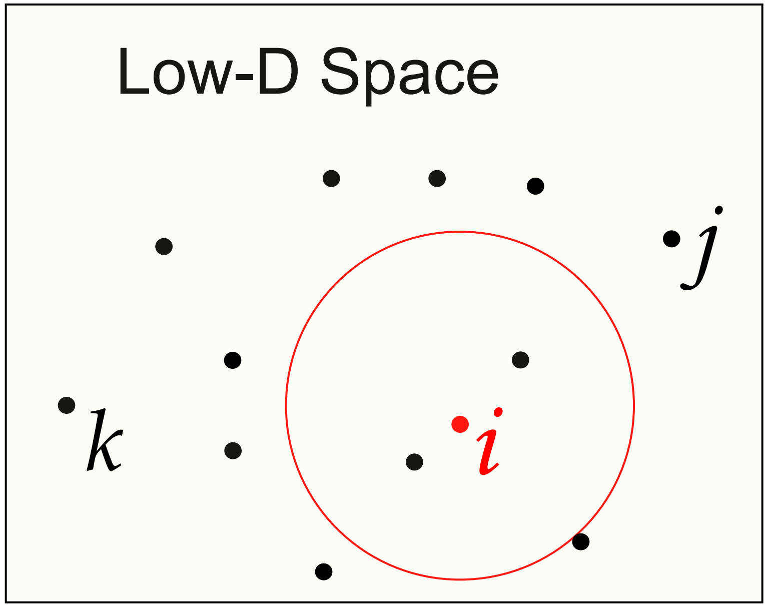
joint probability of picking the pair i,j $\longrightarrow$ $p_{ij} = \dfrac{p_{j|i} + p_{i|j}}{2n}$

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

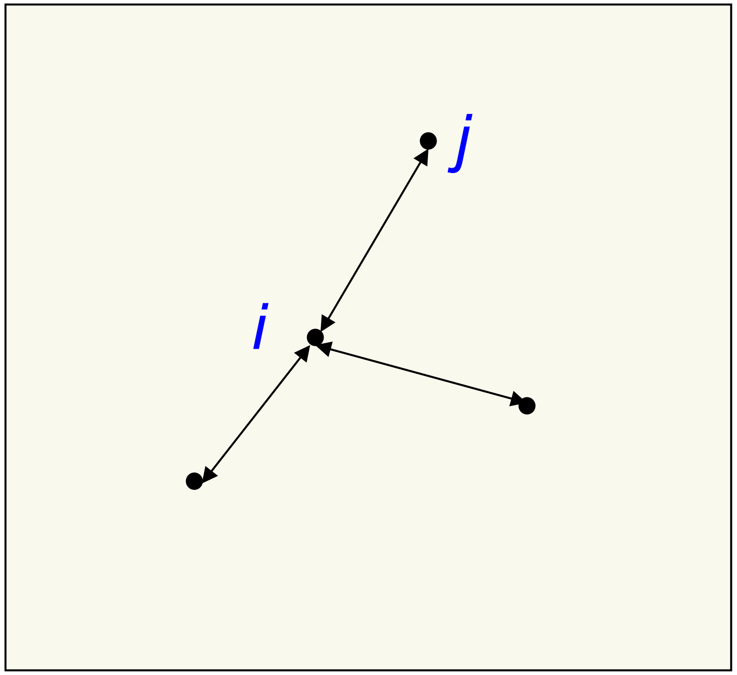
$$Cost \ = KL(P \,\|\, Q) \mid \ = \sum_{i<j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

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

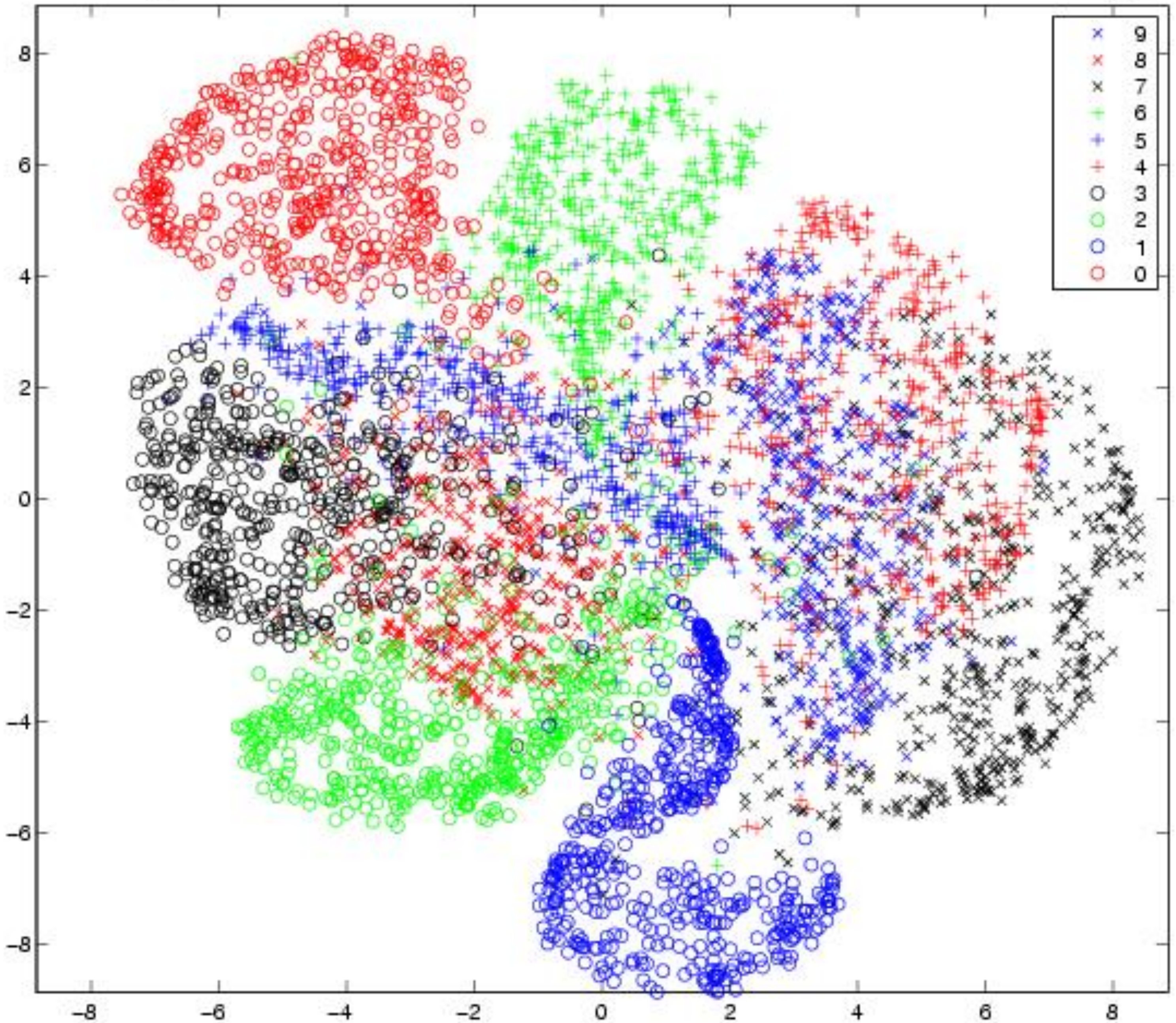
The forces acting on the low-dimensional points

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

- 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



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

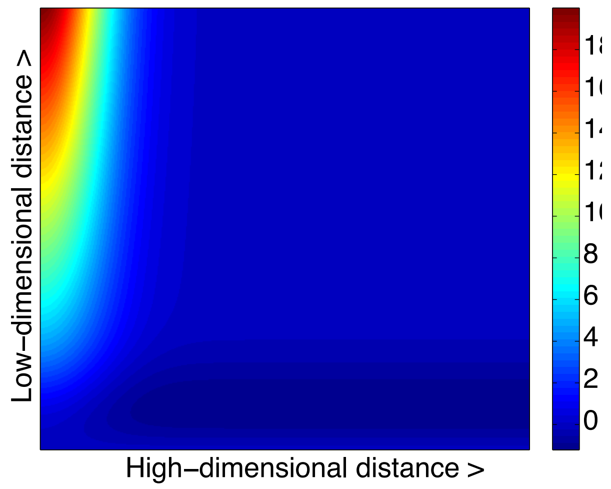
t-SNE

- 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

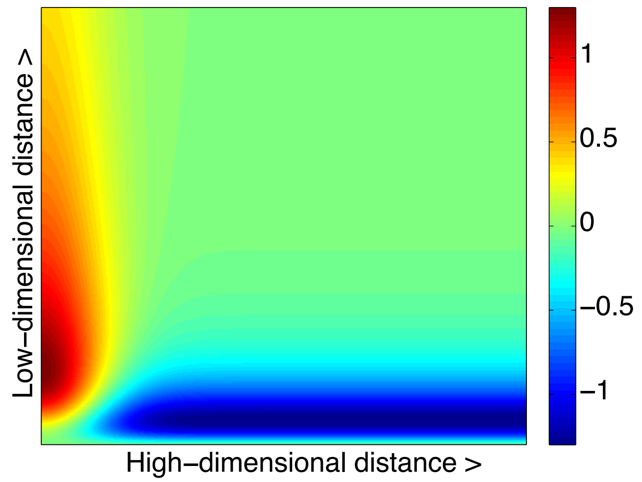
$$q_{ij} \propto \frac{1}{1+d_{ij}^2}$$

Gradient of t-SNE

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



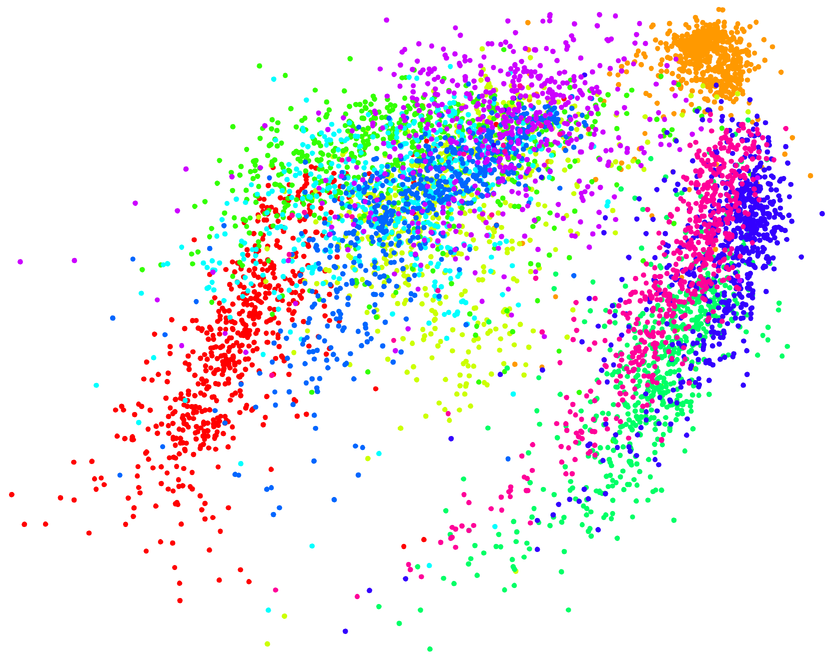
(a) Gradient of SNE.



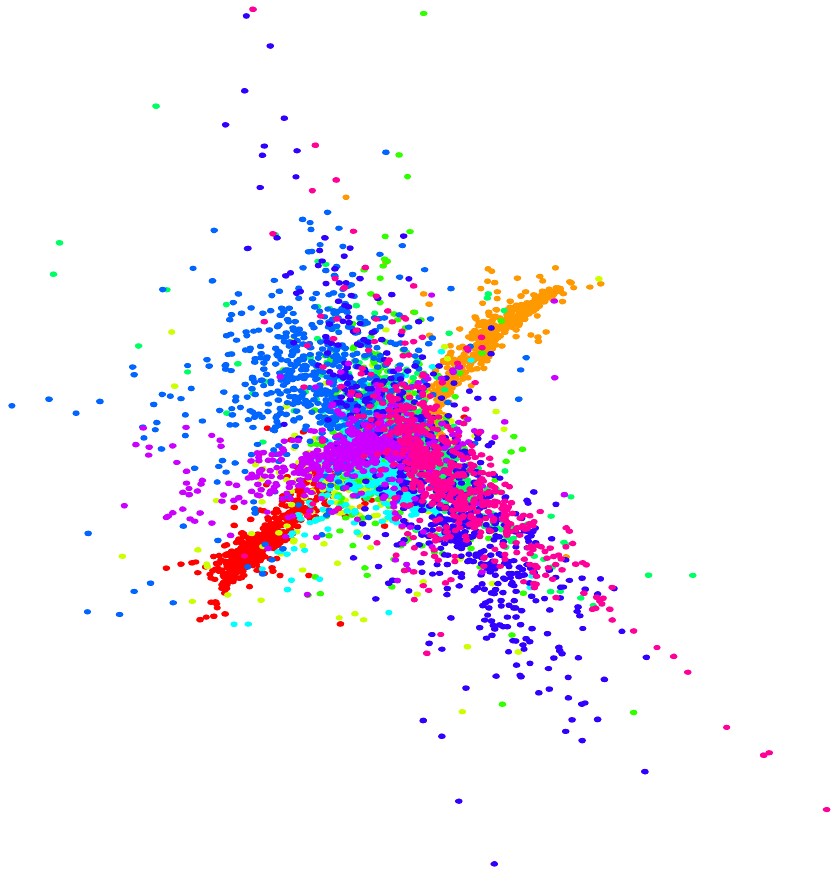
(c) Gradient of t-SNE.

t-SNE allows more points in moderate distance neighbors

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

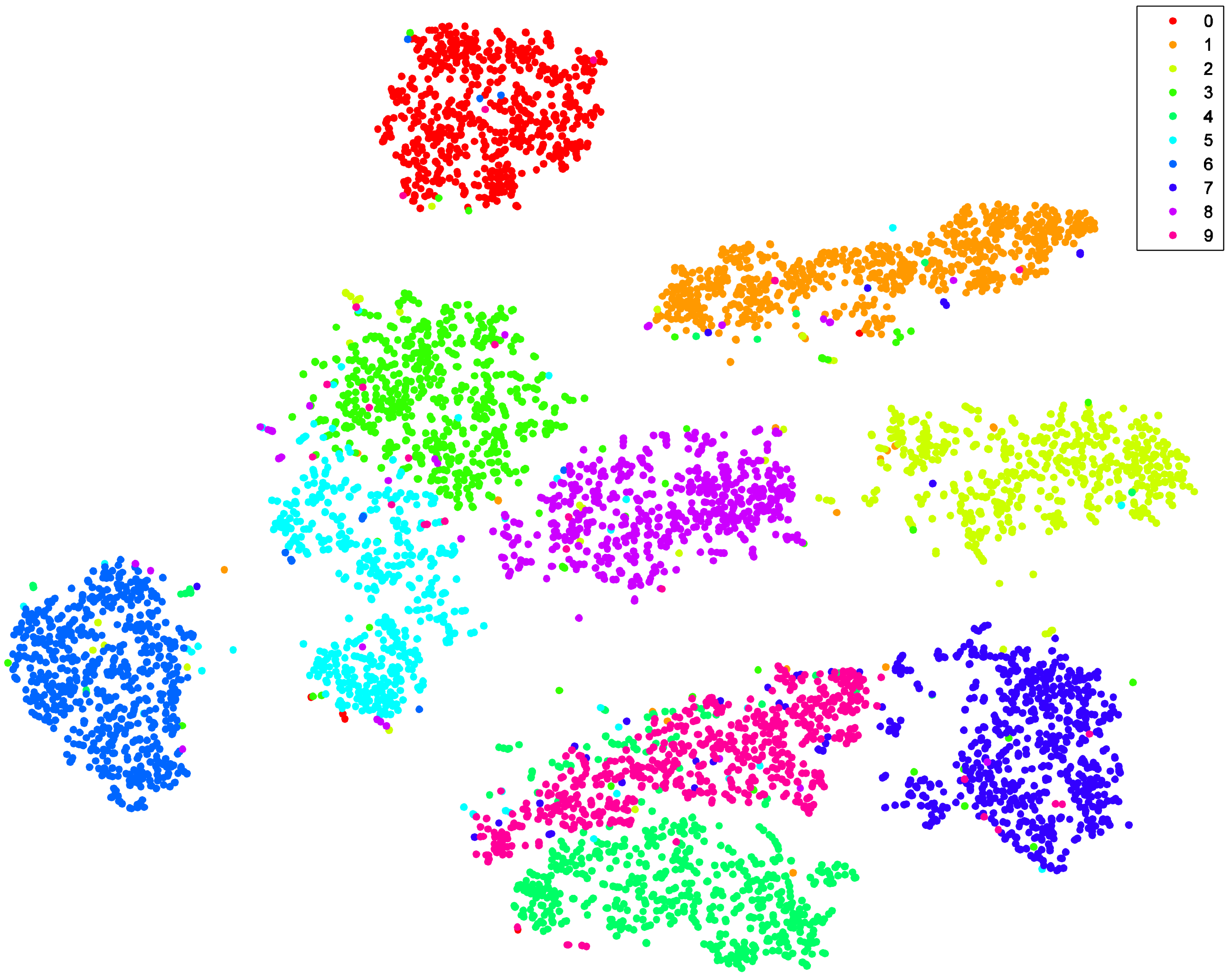


Isomap



Locally Linear Embedding

t-SNE on the 6000 MNIST digits



SNE vs. Laplacian Eigenmap

- 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.

SNE vs. Laplacian LLE -> Elastic Embedding

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

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

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

$$\textstyle\sum_{m=1}^{N} p_{nm} = 1$$

Elastic Embedding (EE):

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

Swiss Roll Example

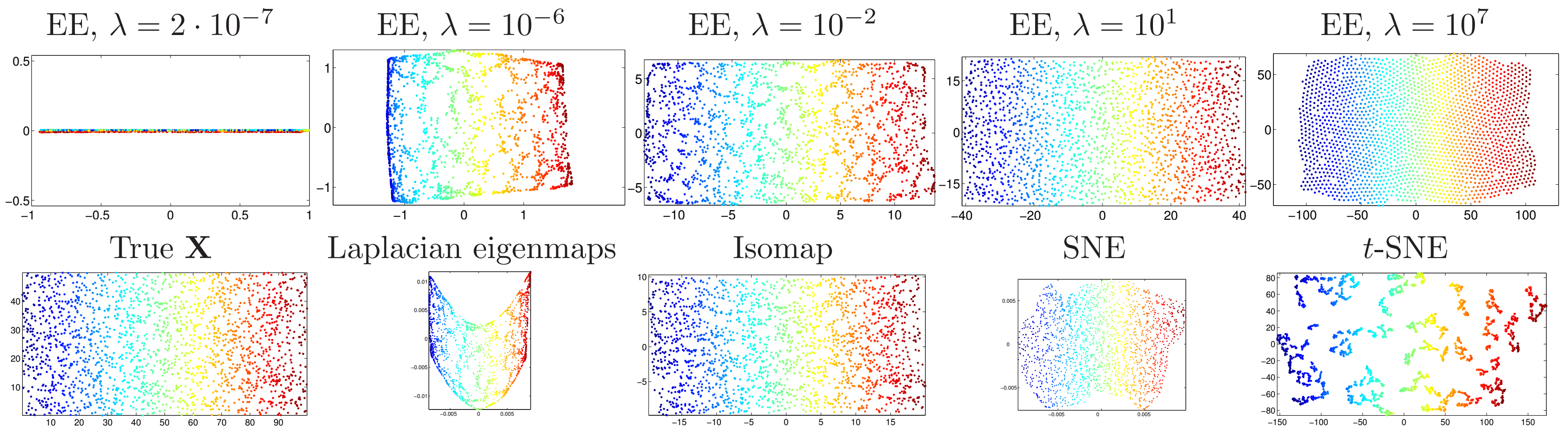


Figure 3. Swiss roll. *Top*: EE with homotopy; we show $\mathbf{X}$ for different λ. *Bottom*: true $\mathbf{X}$ and results with other methods.