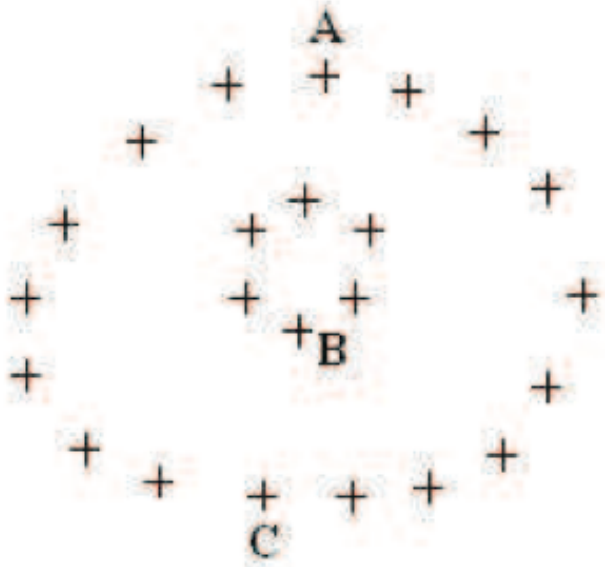
Diffusion Map
&
Stochastic Neighbor Embedding

姚 遠

2017

Data Distances



- We look for distance function such that
  - dist(A,C) is small
  - dist(A,B) is large
- Geodesic distance is one candidate, but hard to compute and sensitive to noise
- Any other distance with such properties but robust to stochastic noise?

Data Graph

- Given n points x_i, i=1,…,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

$$|\lambda_i| \leq 1$$

- 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

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

- 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*

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

- 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

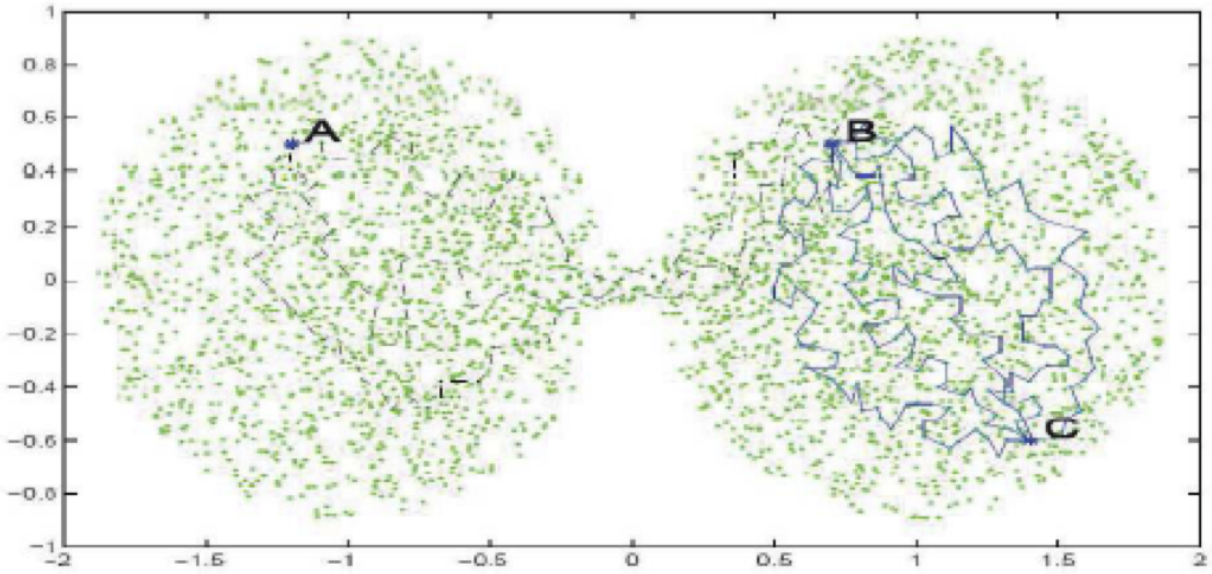


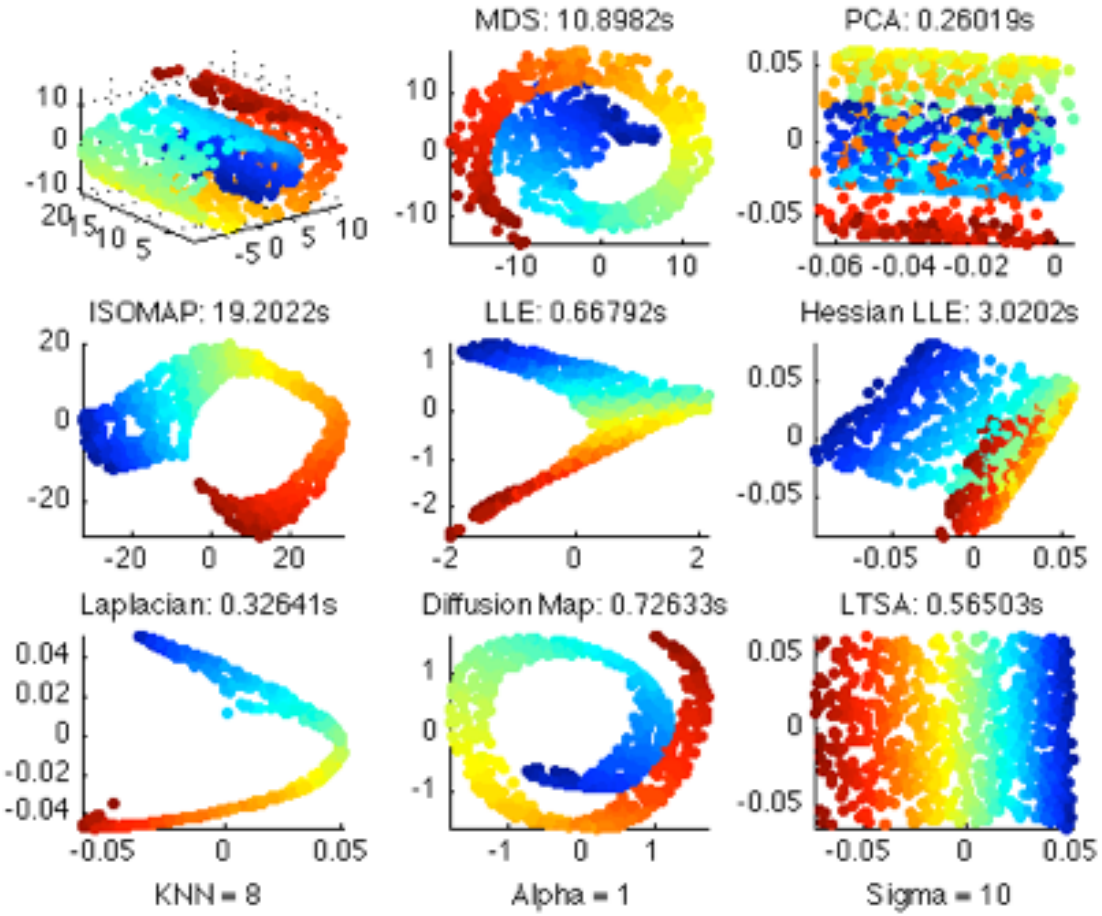
FIGURE 1. Diffusion Distances $d_t(A, B) >> d_t(B, C)$ while graph shortest path $d_{geod}(A, B) \sim d_{geod}(B, C)$.

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



General Diffusion Map

- 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.

Lumpability of Markov Chains

- Let P be the transition matrix of a Markov chain defined on n states S={1,…,n}.
- $\Gamma=\{S_1,\ldots,S_k\}$ is a partition of S into k macrostates.
- Sequences $\{x_0,\ldots,x_t,\ldots\}$ generated by P, i.e.

$$\text{Prob}(x_t=j\ ;\ x_{t-1}=i)= P_{ij}$$

- Induced dynamics: relabel x_t by y_t from corresponding states in partition Γ
- [Kemeny-Snell'76] P is called *lumpable* if

$$\text{Prob}(y_t=k_0;\ y_{t-1}=k_1,\ \ldots,y_{t-m}=k_m) = \text{Prob}(y_t=k_0;\ y_{t-1}=k_1)$$

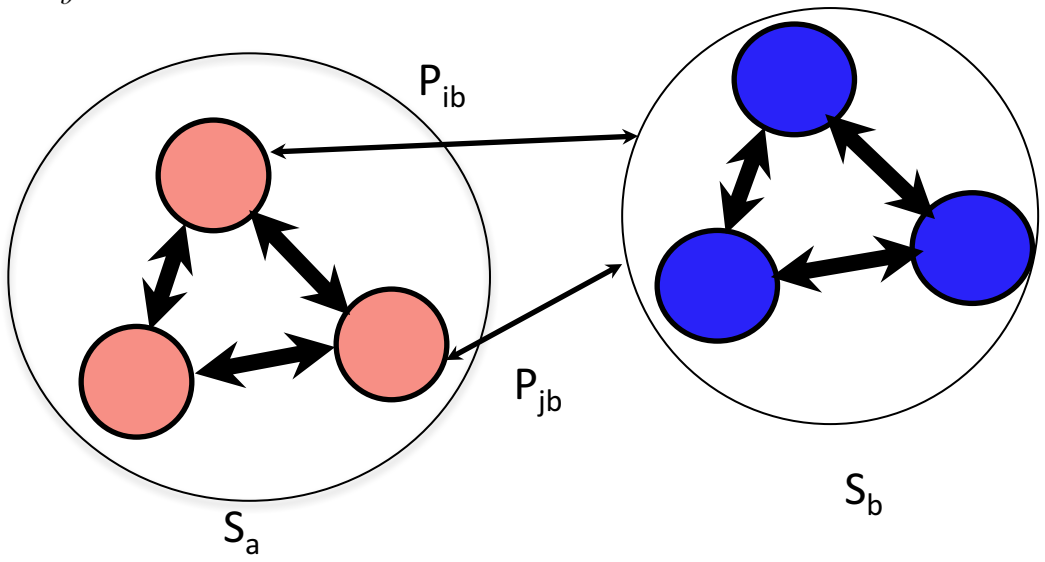
i.e. the induced dynamics is Markovian.

A Necessary and Sufficient Condition for Lumpability

- [Kemeny-Snell'76] P is *lumpable* w.r.t. partition $\Gamma=\{S_1,\ldots,S_k\}$ iff for any s, t chosen from P, and for any i, j lying in S_a, the following holds

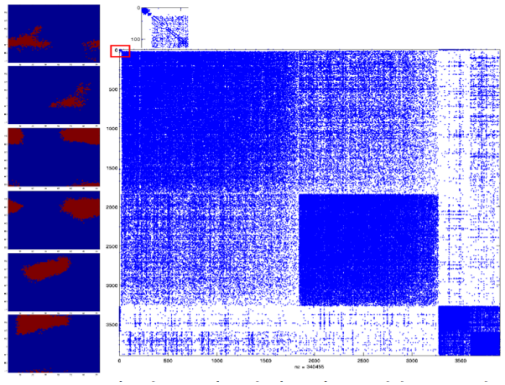
$$P_{ib} = P_{jb}$$

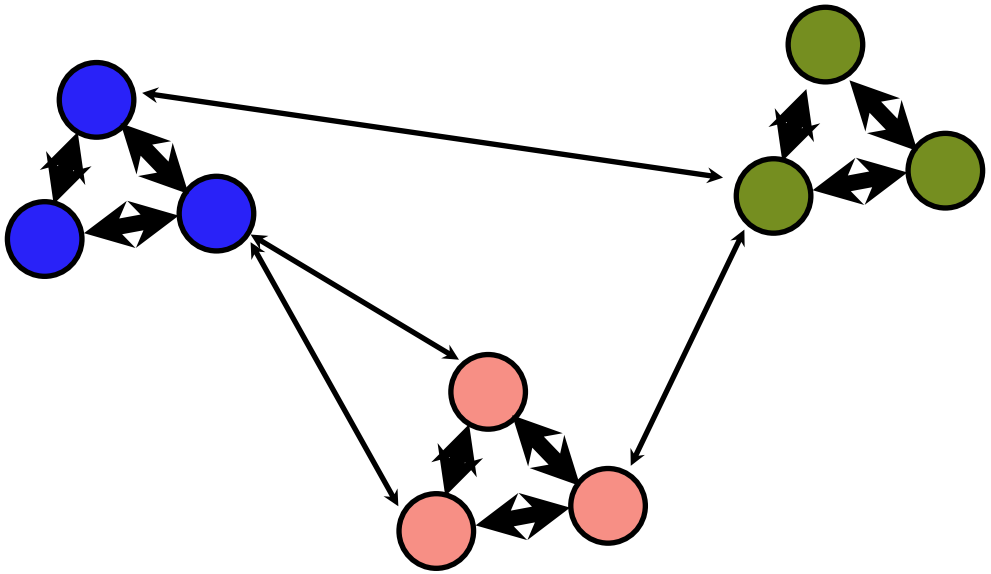
where $P_{ib} = \sum_{k \in S_b} P_{ik}$



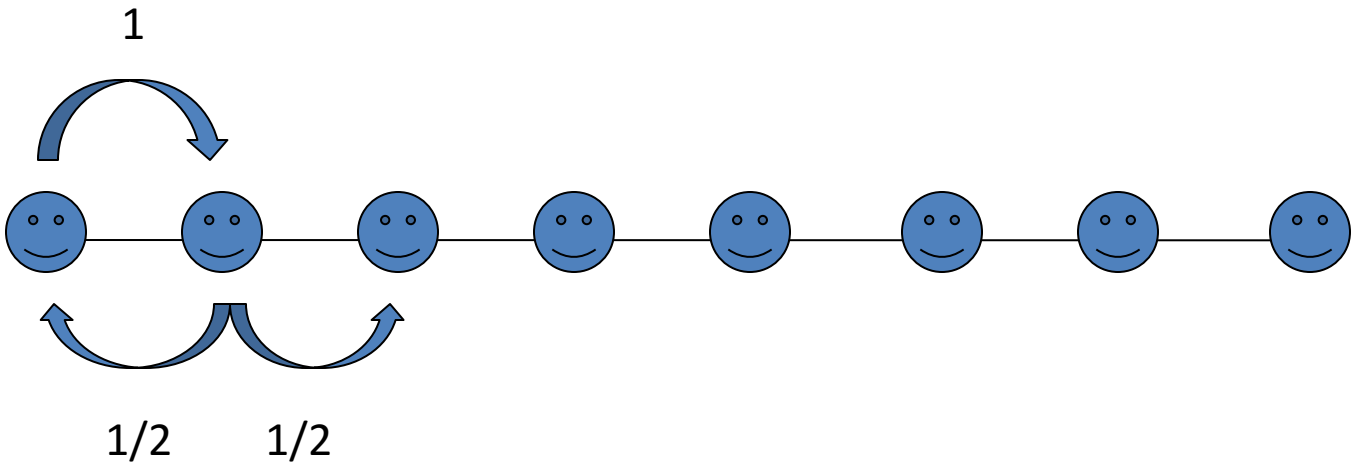
Spectral Theory of Lumpability

- [Meila-Shi 2001] P is *lumpable w.r.t. P* iff P has k independent piece-wise constant right eigenvectors in the span of characteristic functions of $\Gamma=\{S_1,\ldots,S_k\}$.
- Special case: If P is block diagonal, i.e. uncoupled Markov chain, then P is lumpable with piece-wise constant right eigenvectors associated with multiple eigenvalue 1.
- [e.g. Belkin-Shi-Yu 2007] If P is nearly block diagonal, then there are top-k eigenvectors which fix signs within the block.
- So Diffusion Map will map lumpable clusters to a simplex



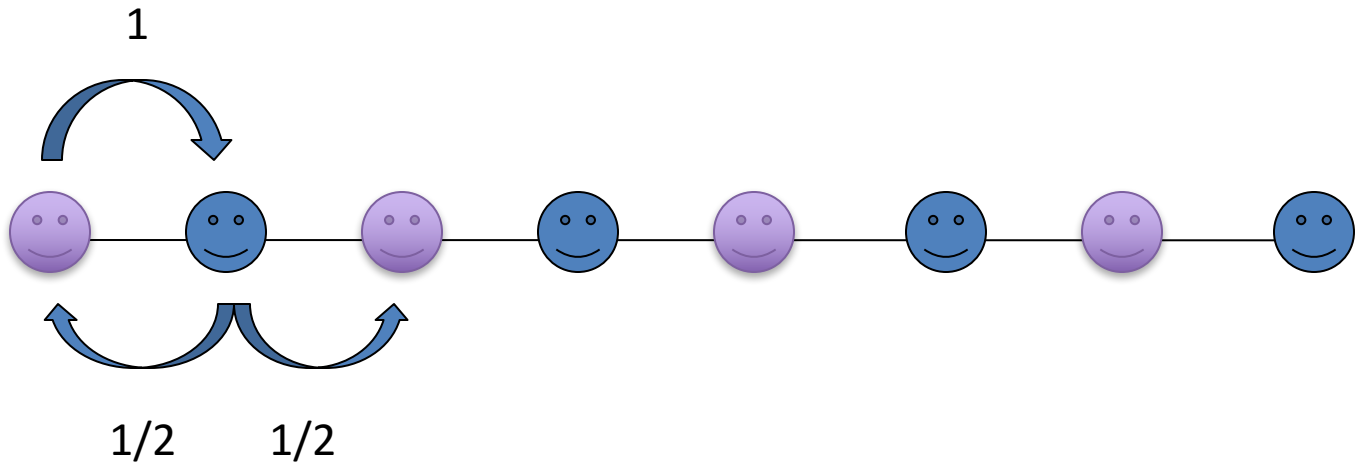


Example I



- Consider 2n nodes on a linear chain
- Markov Chain: a node will jump to its neighbors with equal probability
  - P(i, i-1) = P(i, i+1) = ½, for 2n>i>1
  - P(1,2) = P(2n,2n-1) = 1

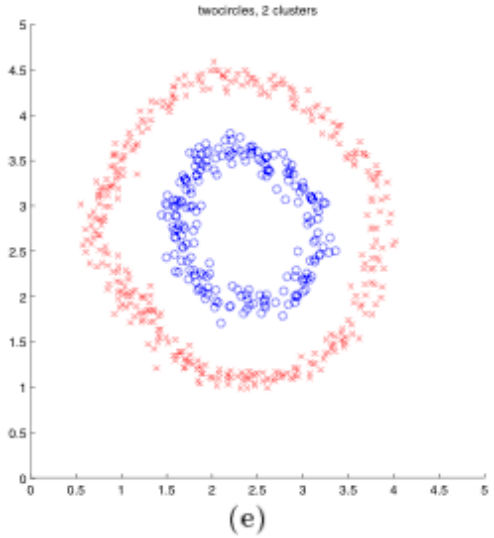
Example I



- P is lumpable w.r.t. $\Gamma^*=(S_{even},S_{odd})$
  - S_{even}: even nodes
  - S_{odd}: odd nodes
- Γ^* corresponds to eigenvector with eigenvalue -1

Spectral Clustering Algorithm

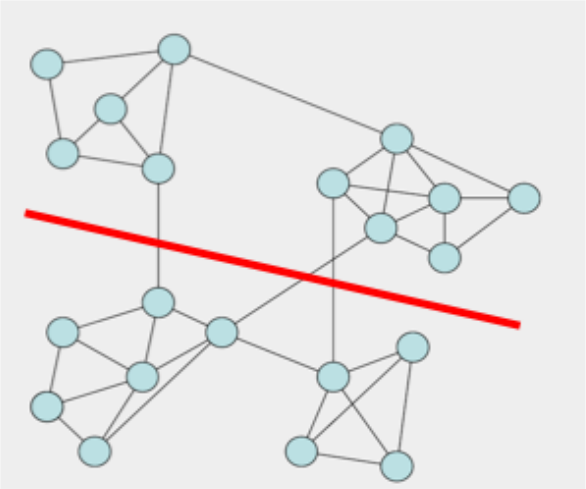
- Typical spectral algorithm to find lumpable states in nearly uncoupled systems [Ng-Jordan-Weiss NIPS'01]:
    1) Find top d+1 right eigenvectors of P where a large spectral gap occurs, $\phi_0,\ldots,\phi_d$
    2) Embed the data into R^d by dropping ϕ_0
    3) Use k-means (or alternatives) to find d clusters in R^d



Graph Partition Problem

- goal: find a cut with the smallest Cheeger ratio (conductance)
  - For $S \subset V$, volume of S: $vol(S) = \sum_{v \in S} d_v$
  - $\partial S = \{(u, v) \in E : u \in S \& v \in S\}$
  - Cheeger ratio of S, $h(S) = \frac{|\partial S|}{\min\{vol(S), vol(G) - vol(S)\}}$

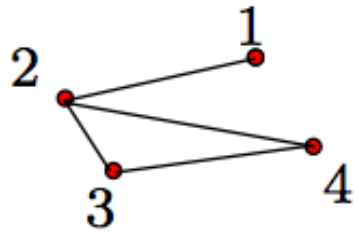
- applications
  - clustering
  - segmentation
  - task partitioning for parallel processing
  - a preprocessing step to divide-and-conquer algorithms



Graph Laplacian Operator

- given an undirected graph G=(V, E),
  - Adjacency matrix A:

$$A(u,v) = \left\{ \begin{array}{ll} 1 & \text{if } u \sim v \\ 0 & \text{o.w.} \end{array} \right\}$$



  - Diagonal degree matrix $D = diag(d_{v_1}, \cdots, d_{v_n})$
  - Graph Laplace Operator $L = D^{-1}(D - A)$
  - Tranistion probability matrix $W = D^{-1}A = I - L$,
  - $Wv = \lambda v$ implies $Lv = (1 - \lambda)v$
  - 1 is the largest eigenvalue for W; 0 is the smallest eigenvalue for L.

Graph Partition Problem

- Rayleigh quotient $R(f) = \frac{\sum_{u \sim v}(f(u)-f(v))^2}{\sum_u f^2(u) d_u}$ for $f \neq 0$
  - find a boolean function f minimizing $R(f)$ $\Leftarrow$ NP-complete
  - RELAXATION: find a real valued function f minimizing $R(f)$
  - $R(f) = \frac{<f,(D-A)f>}{<f,Df>}$
  - $\lambda_1 = \inf_f R(f) \Rightarrow \lambda_1$ and f are the first nonzero eigenvalue and eigenvector of L.

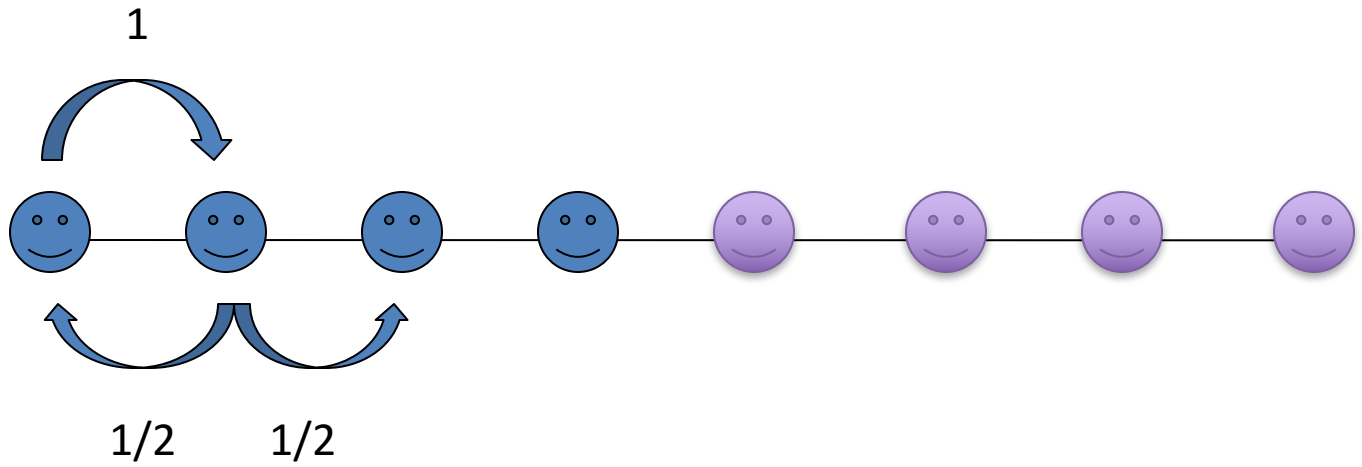
How good is this relaxation? Cheeger inequality

Cheeger Inequality

$$2h_G \geq \lambda_1 \geq \frac{h_f^2}{2} \geq \frac{h_G^2}{2}.$$

- f is the eigenvector of L corresponding to λ_1
- h_G is the smallest conductance (Cheeger ratio) of graph G
- h_f: the minimum Cheeger ratio determinded by a sweep of f
  - order the vertices: $f(v_1) \geq f(v_2) \geq \cdots \geq f(v_n)$.
  - $S_i = \{v_1, \cdots, v_i\}$
  - $h_f = \min_i h_{S_i}$
- find a partition whose conductance is within $2\sqrt{h_G}$

Example I



- One graph min-cut given by second largest right eigenvector of T
- n=8,
  - v_2=[0.4714 0.4247 0.2939 0.1049 -0.1049 -0.2939 -0.4247 -0.4714]
  - Eigenvalue is 0.9010

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*}

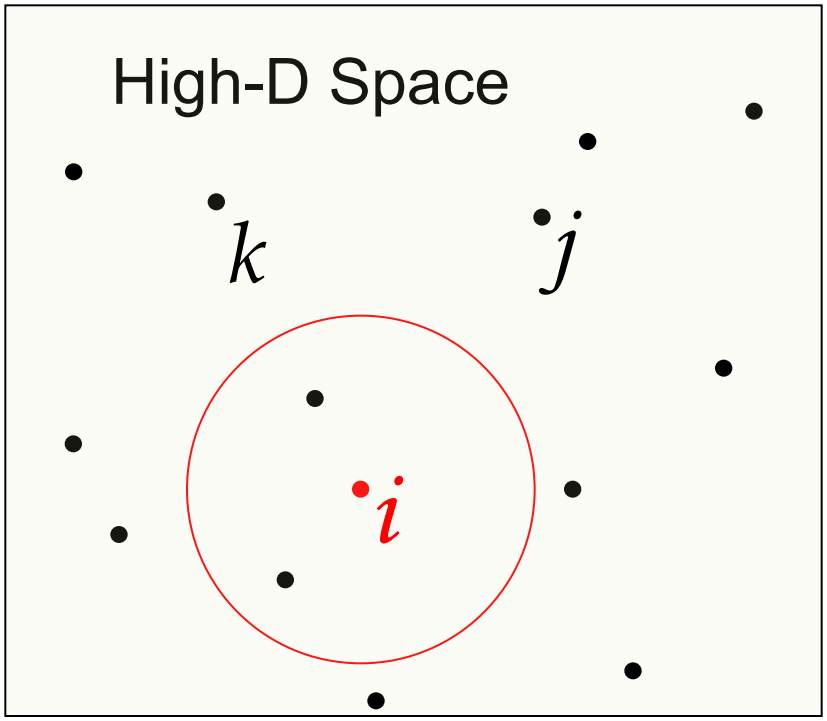
Stochastic Neighbor Embedding

- 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.

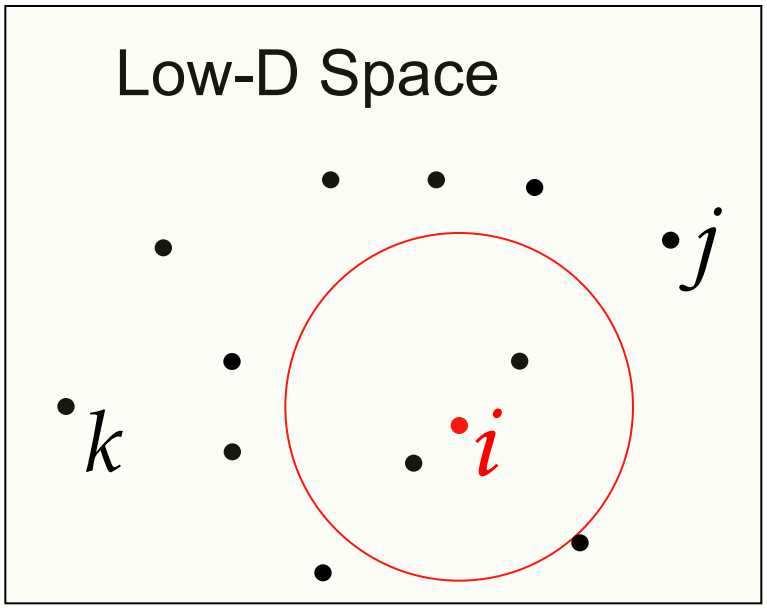


$$p_{j|i} = \frac{e^{-d_{ij}^2 / 2\sigma_i^2}}{\sum_k e^{-d_{ik}^2 / 2\sigma_i^2}}$$

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.

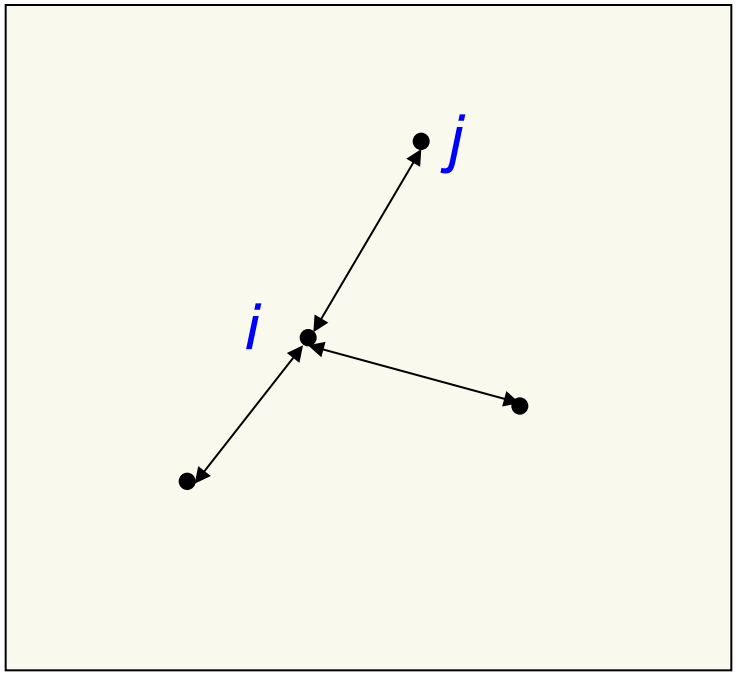
Gradient Descent

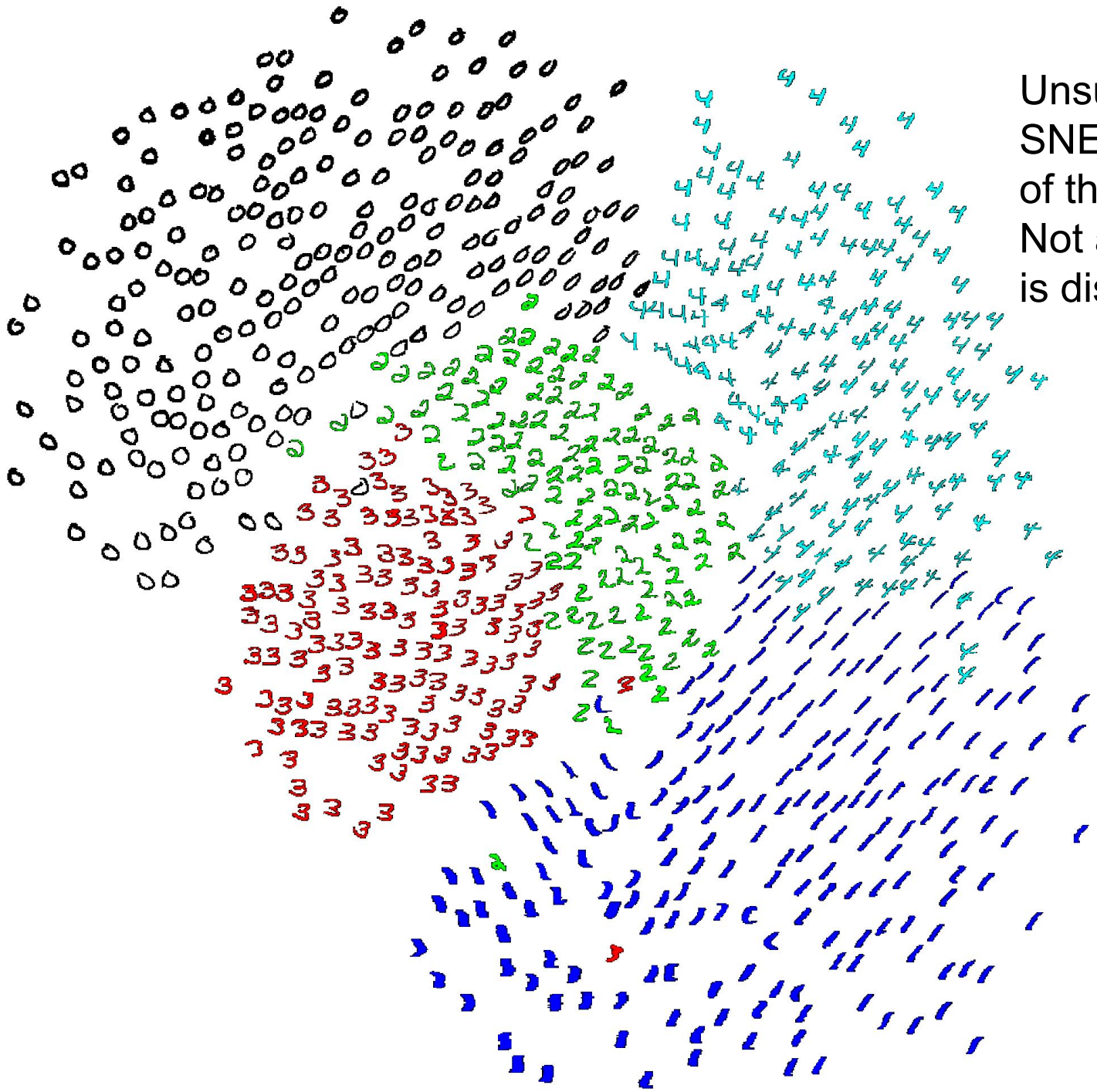
$$\frac{\partial Cost}{\partial \mathbf{y}_i} = 2\sum_j (\mathbf{y}_j - \mathbf{y}_i)\left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)$$

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

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

- 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.

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

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

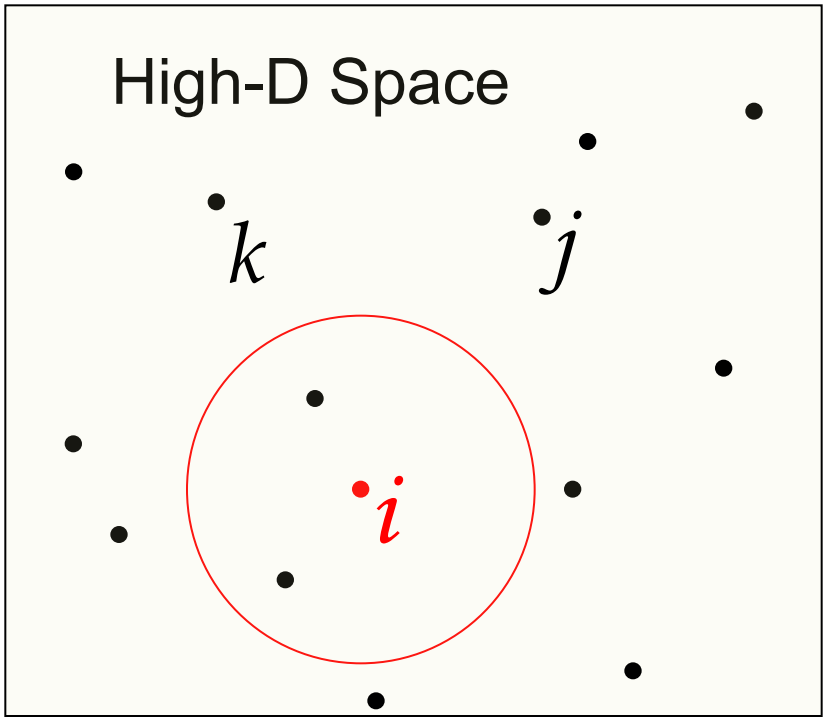
Symmetric SNE

- 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

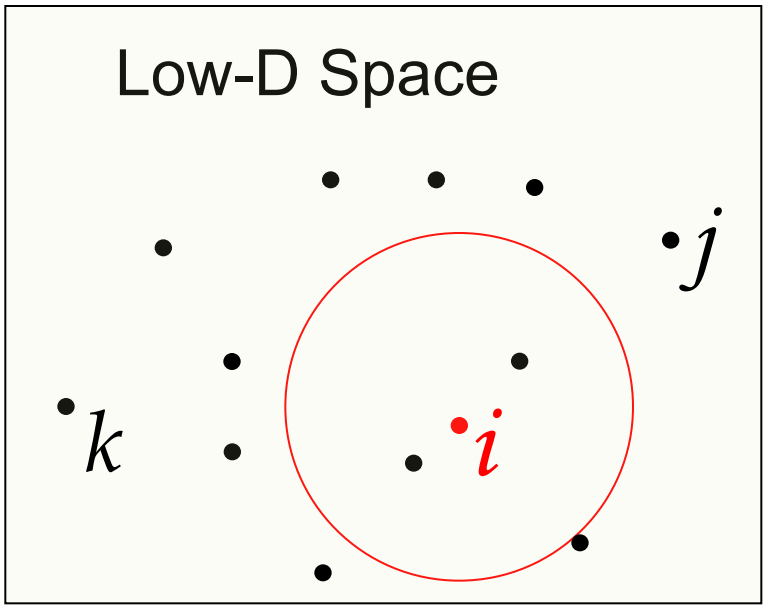
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

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

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

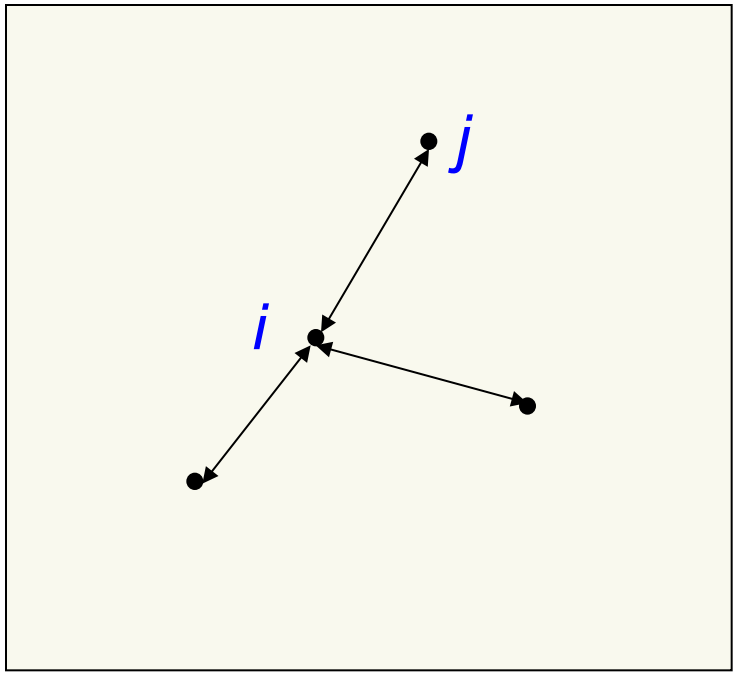
- It's a single KL instead of the sum of one KL for each datapoint.

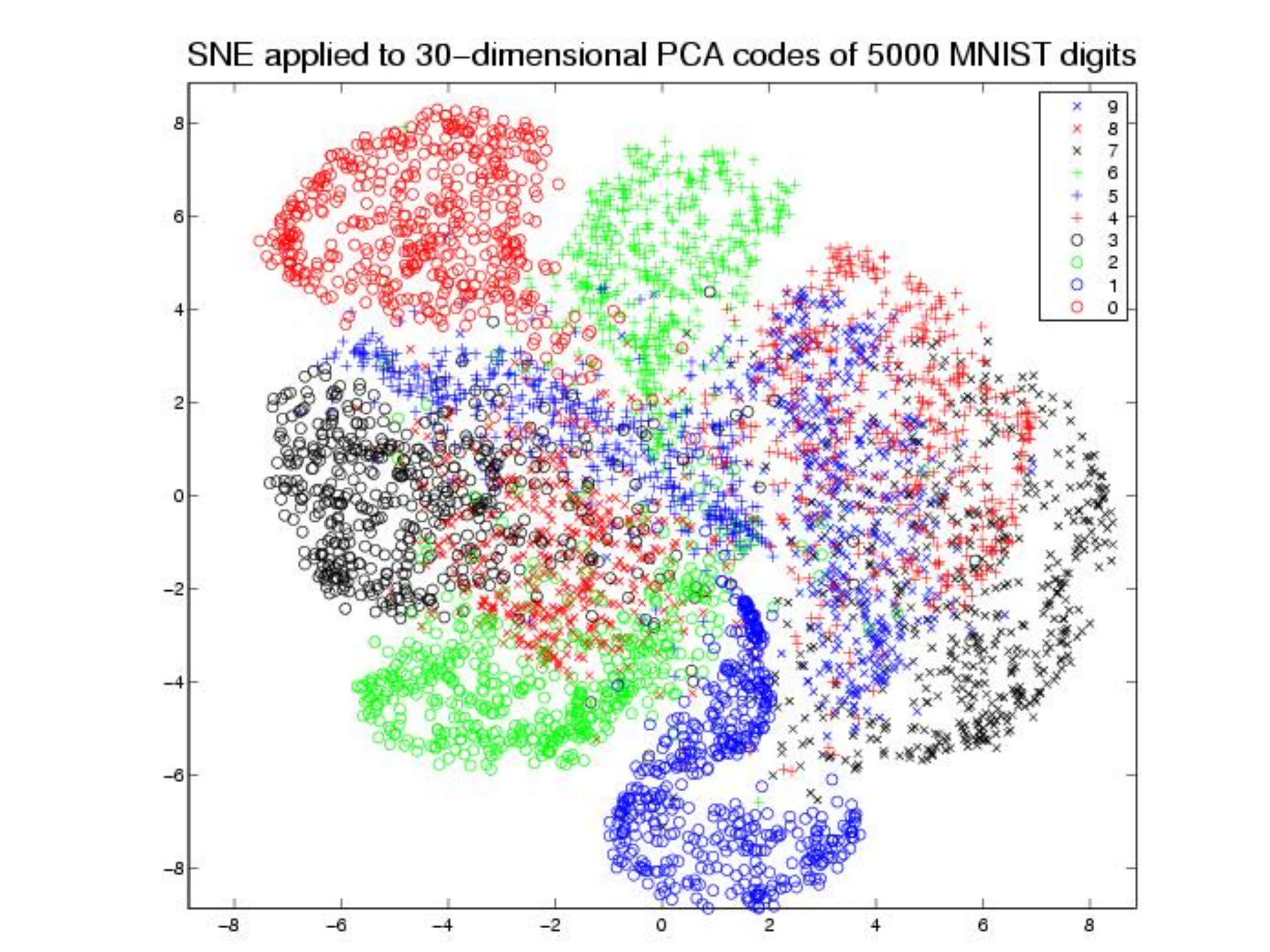
The forces acting on the low-dimensional points

extension stiffness

$$\frac{\partial KL(P \| Q)}{\partial \mathbf{y}_i} = 2\sum_j (\mathbf{y}_i - \mathbf{y}_j)(p_{ij} - q_{ij})$$

- 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

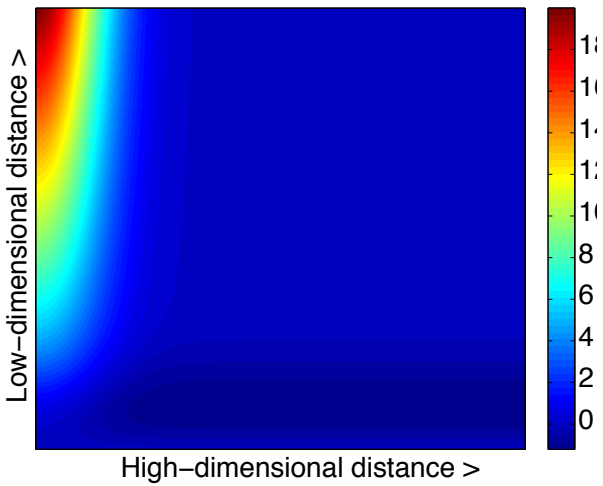
t-SNE

- 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

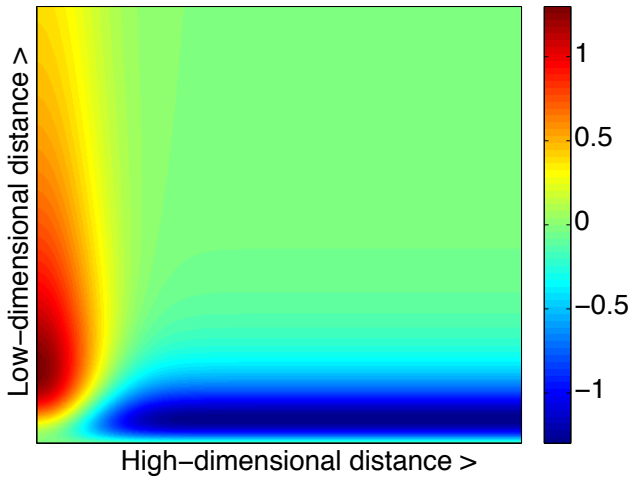
$$q_{ij} \propto \frac{1}{1+d_{ij}^2}$$

Gradient of t-SNE

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



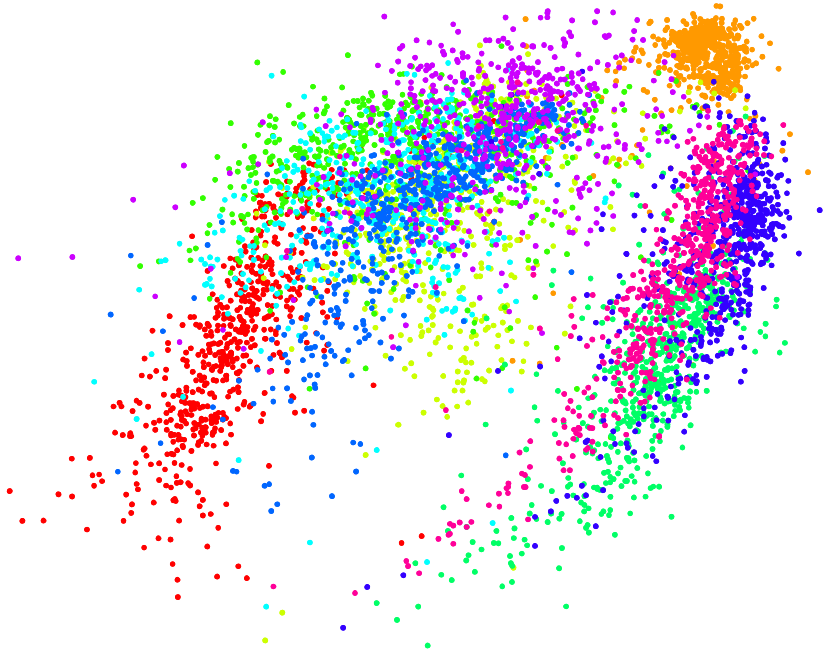
(a) Gradient of SNE.



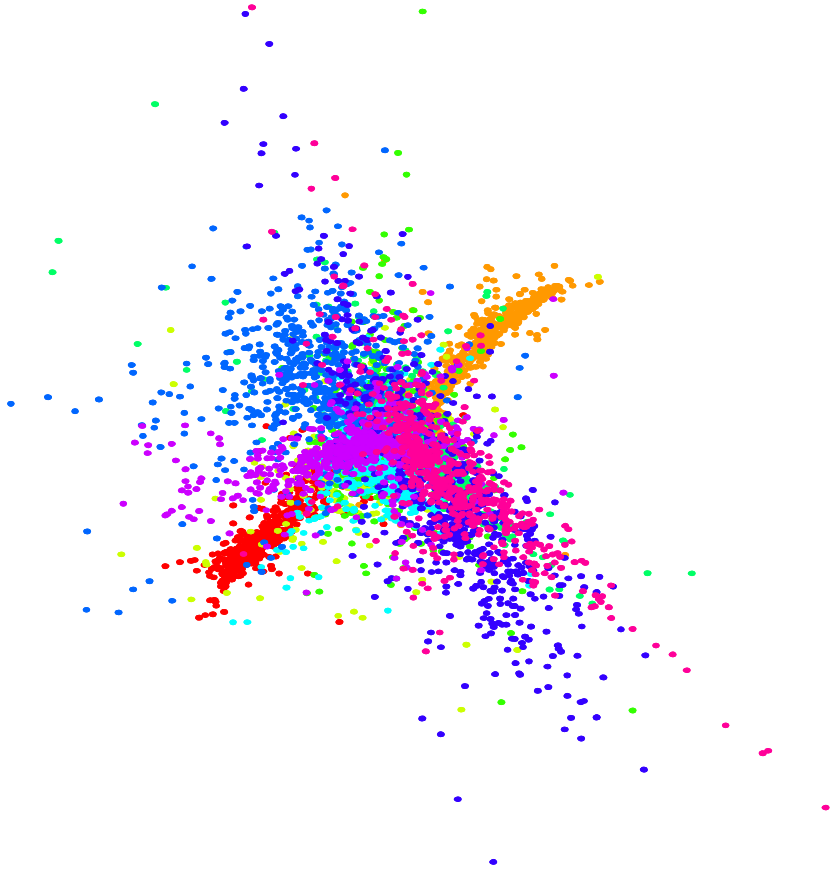
(c) Gradient of t-SNE.

t-SNE allows more points in moderate distance neighbors

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

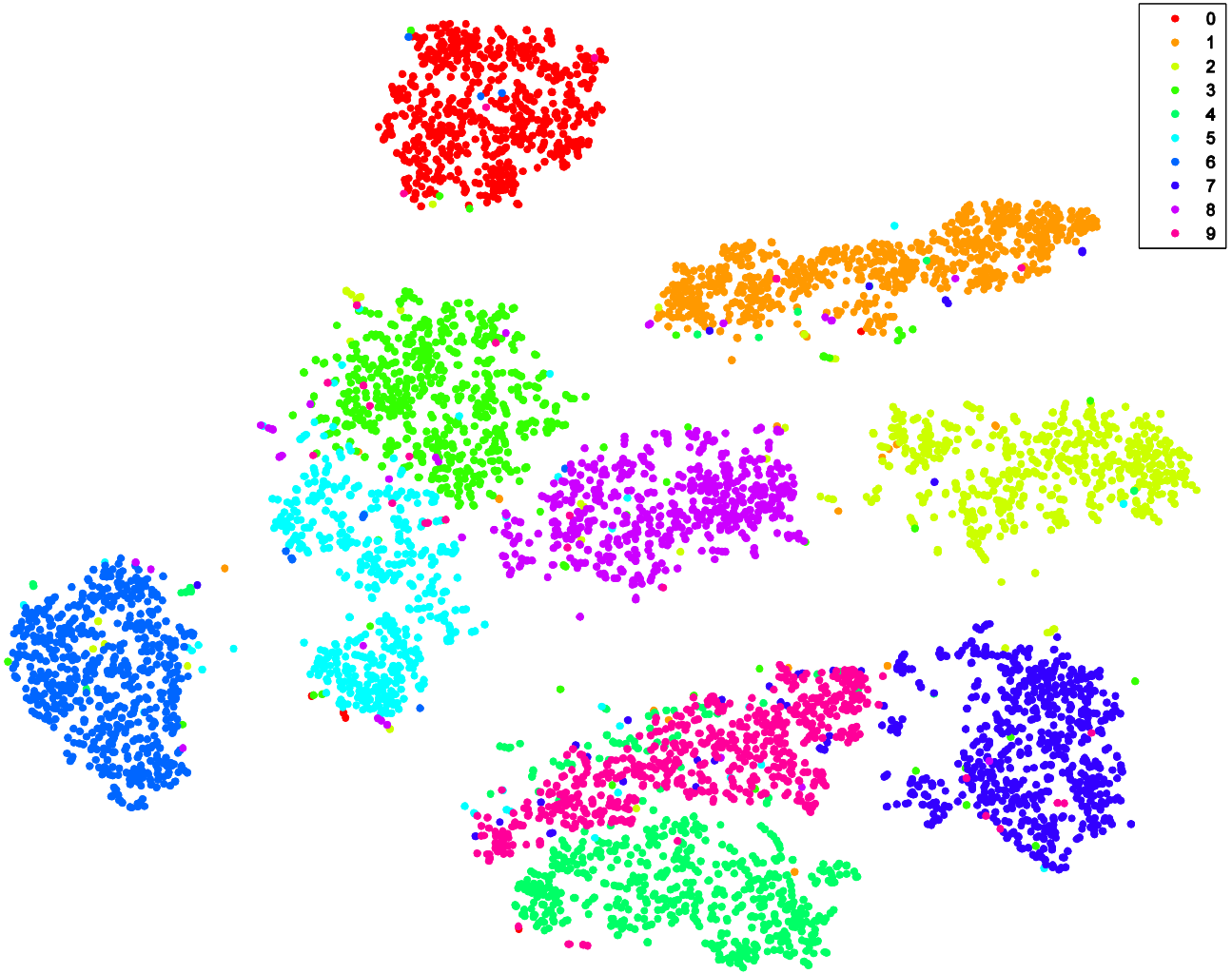


Isomap



Locally Linear Embedding

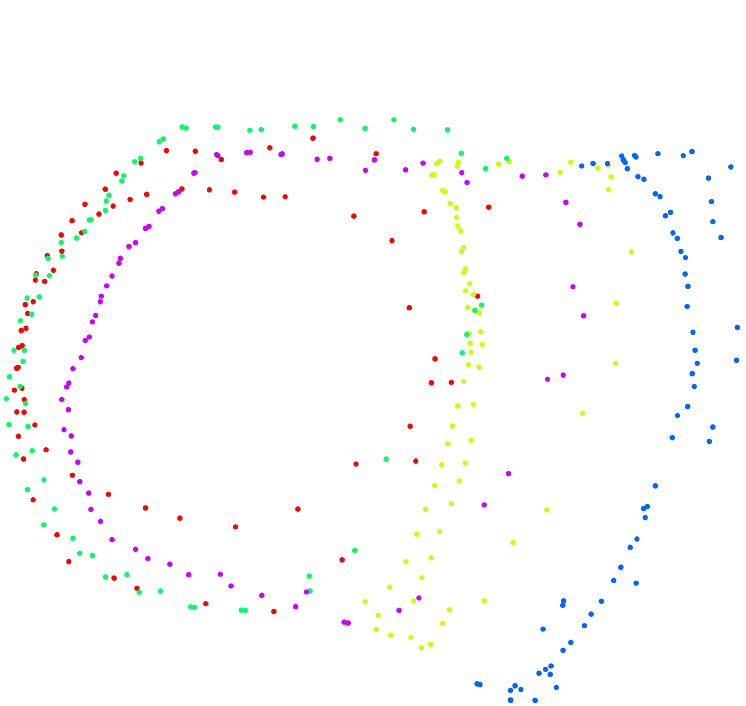
t-SNE on the 6000 MNIST digits



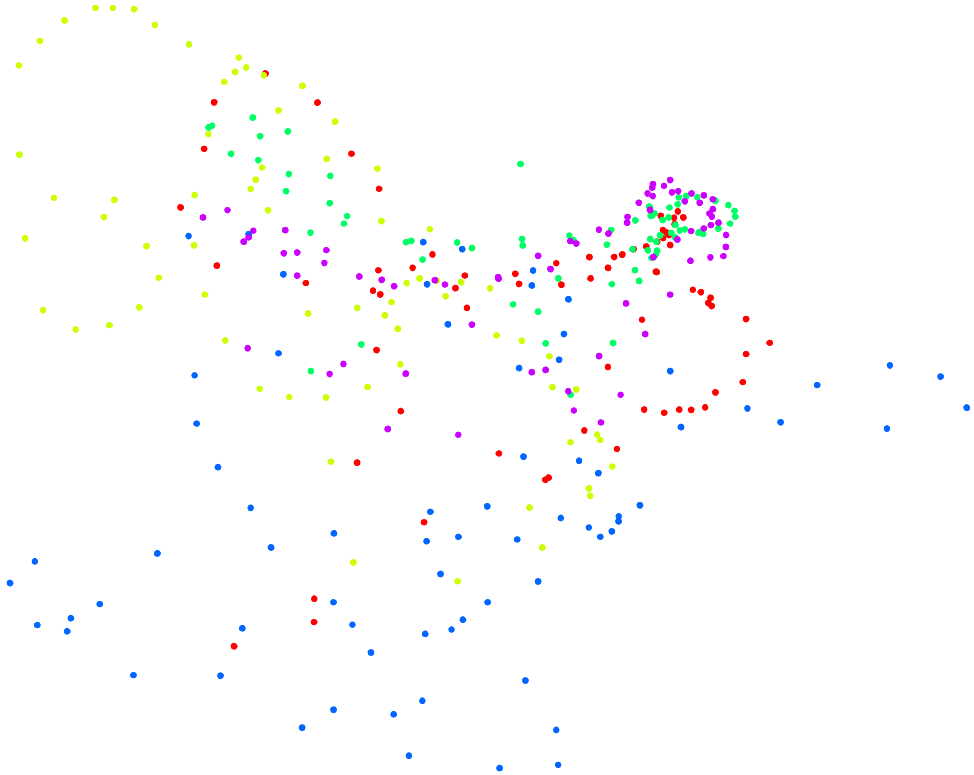
The COIL20 dataset

Each object is rotated about a vertical axis to produce a closed one-dimensional manifold of images.

Isomap & LLE for COIL20 dataset

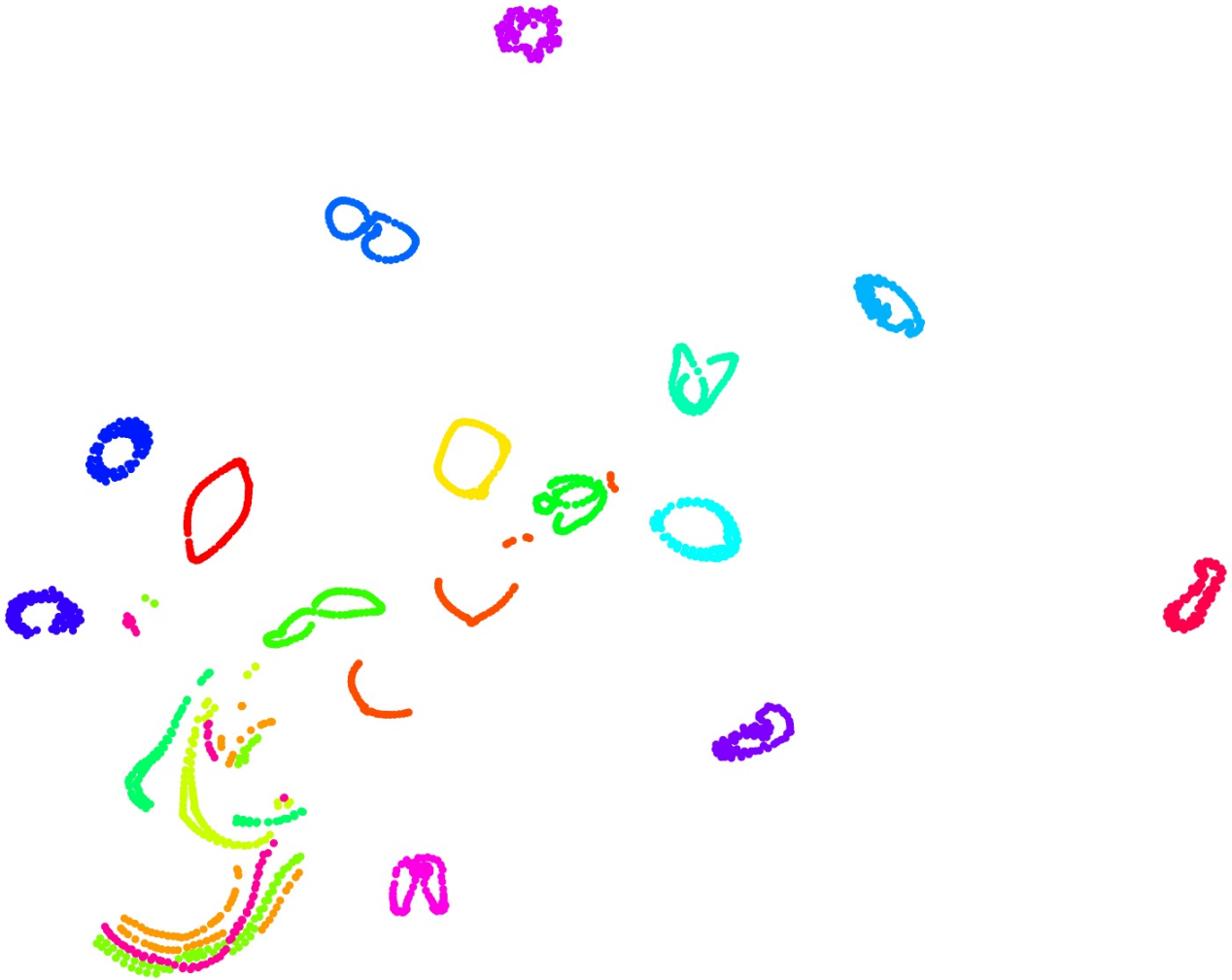


Isomap



Locally Linear Embedding

t-SNE for COIL20 dataset



Show the map of 2000 English words produced by Joseph Turian using t-SNE on the feature vectors learned by Colobert and Weston (ICML 2008)

Using t-SNE to see what you are thinking

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

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

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

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

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

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

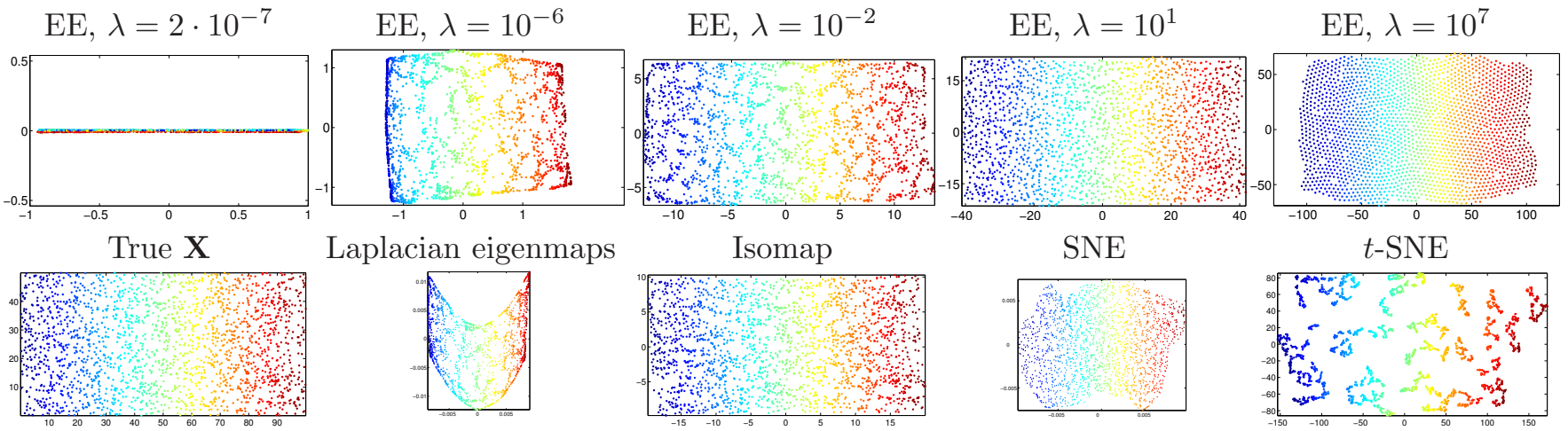


Figure 3. Swiss roll. *Top*: EE with homotopy; we show $\mathbf{X}$ for different λ. *Bottom*: true $\mathbf{X}$ and results with other methods.