Distance and similarity (I)

Hongxin Zhang
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU

2015-03-24

In the last lesson

- Data driven decomposition:
  - Data driven curve fitting
  - Singular value decomposition (the power of orthogonal basis)
  - A sort of spectral analysis
- PCA and its related techniques are very useful

In the last lesson

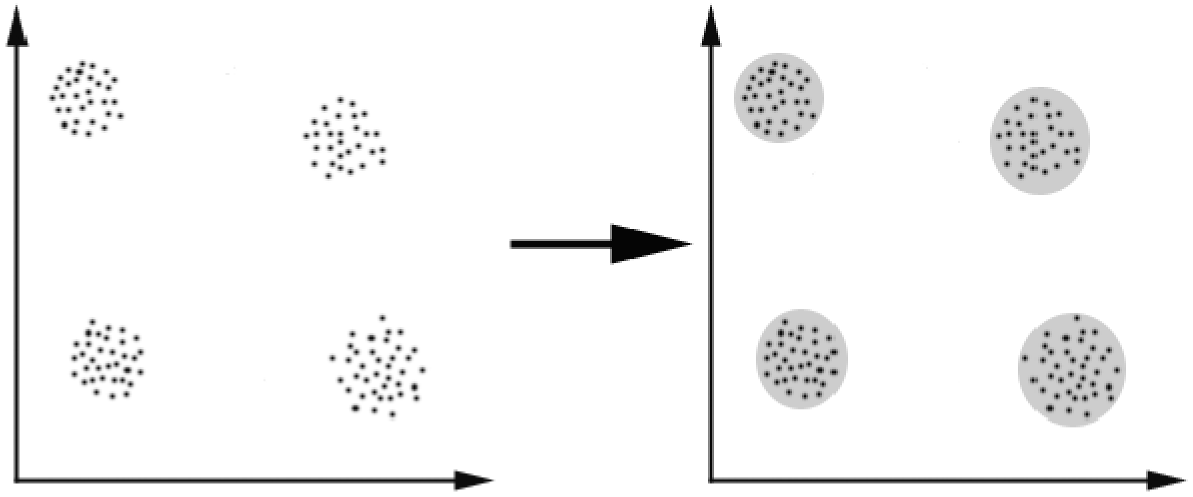
- Mathematical concepts and techniques
  - Least squares (LSQ)
  - Curve fitting
  - Norm (范数) and inner product (内积)
  - Singular value decomposition
  - Eigen vectors and eigen-values
  - Low rank matrix approximation and decomposition

Today's Talk

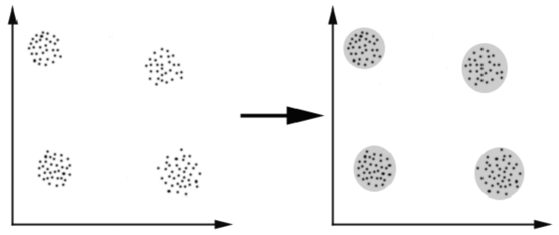
- Pre-processing: Distance / metric learning
  - ISO-map
  - LLE
- What is similarity? How to Clustering
  - Spectral based
  - E-M based
    - MOG and K-means
  - Mean shift

Clustering

- Given set of data points, group them, find the overall structure
- Unsupervised learning
- Learn the similarity. Which patient are similar? (or customers, faces, earthquakes, …)



Distance



- Given n-dimensional vector x, y
- Euclidian (L^2 distance)

$$dist(\mathbf{x},\mathbf{y};2)=\left(\sum_{i=1}^{n}(x_i-y_i)^2\right)^{1/2}$$

- L^1 distance

$$dist(\mathbf{x},\mathbf{y};1)=\sum_{i=1}^{n}\left|x_i-y_i\right|$$

- L^p distance (Minkowsky)

$$dist(\mathbf{x},\mathbf{y};p)=\left(\sum_{i=1}^{n}(x_i-y_i)^p\right)^{1/p}$$

$$dist(\mathbf{x},\mathbf{y};\infty)=\max_i\left|x_i-y_i\right|$$

切比雪夫

Distance, Norm and inner product

- Distance

$$dist(\mathbf{x}, \mathbf{y}; 2) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}$$

- Norm

$$norm(\mathbf{x}; 2) := \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$$

- Inner product

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

$$norm(\mathbf{x} - \mathbf{y}; 2) = dist(\mathbf{x}, \mathbf{y}; 2)$$

$$dist(x, y; 2) = \left(\mathbf{x} \cdot \mathbf{x} + \mathbf{y} \cdot \mathbf{y} - 2\mathbf{x} \cdot \mathbf{y} \right)^{1/2}$$

$$= \left(\|\mathbf{x}\|_2^2 + \|\mathbf{y}\|_2^2 - 2\mathbf{x} \cdot \mathbf{y} \right)^{1/2}$$

Dimensional aware distances

- Along dimension *j*:

$$R_j = \max_i x_{i,j} - \min_i x_{i,j}$$

$$\bar{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j} \qquad S_j = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_{i,j} - \bar{x}_j)^2\right)^{1/2}$$

- Normalized data:

$$x'_{i,j} = \frac{x_{i,j} - \bar{x}_j}{R_j} \qquad x'_{i,j} = \frac{x_{i,j} - \bar{x}_j}{S_j}$$

M-distance

- Consider the dependency of different dimensions

$$dist(\mathbf{x},\mathbf{y};\mathbf{M}) = (\mathbf{x}-\mathbf{y})^{\mathrm{T}}\mathbf{M}^{-1}(\mathbf{x}-\mathbf{y})$$

- M is the covariance matrix of data
- Transform invariance

Data whitening

设有均值为零的随机信号向量 x，其自相关矩阵为

$$R_x = E[xx^T] \neq I$$

很明显，R_x 是对称矩阵，且是非负定的（所有特征值都大于或等于0）。

现在，寻找一个线性变换 B 对 x 进行变换，即 $y = Bx$，使得

$$R_y = BE[xx^T]B^T = I$$

上式的含义是：y的各分量是不相关的，即 $E[y_i y_j] = \delta_{ij}$。通常将这个过程称为“空间解相关”、“空间白化”或“球化”。B 称为空间解相关矩阵（空间白化矩阵、球化矩阵）。

由 R_x 的性质可知，其存在特征值分解：

$$R_x = Q\Sigma Q^T$$

Q 是正交矩阵，Σ 是对角矩阵，其对角元素是 R_x 特征值。

令

$$B = \Sigma^{-1/2}Q^T \qquad (1)$$

$$R_y = BE[xx^T]B^T = I$$

D

上式的含义是：y的各分量是不相关的，即 $E[y_i y_j] = \delta_{ij}$ 。通常将这个过程称为"空间解相关"、"空间白化"或"球化"。 B 称为空间解相关矩阵（空间白化矩阵、球化矩阵）。

由 R_x 的性质可知，其存在特征值分解：

$$R_x = Q\Sigma Q^T$$

Q 是正交矩阵， Σ 是对角矩阵，其对角元素是 R_x 特征值。

令

$$B = \Sigma^{-1/2}Q^T \qquad (1)$$

则有

$$R_y = (\Sigma^{-1/2}Q^T)Q\Sigma Q^T(\Sigma^{-1/2}Q^T)^T = I$$

因此，通过矩阵 B 线性变换后， y 的各个分量变得不相关了。

对于 R_x 来说，特征值分解和奇异值分解是等价的，而**奇异值分解的数值算法比特征值分解的数值算法具有更好的稳定性**，因此一般都用奇异值分解来构造空间解相关矩阵 B 。

应该注意到，"空间解相关"不能保证各分量信号之间的"独立性"，但它能够简化盲分离算法或改善分离算法的性能。

More complex method for distance computing

- PCA …
- structure aware
  - Main idea:
    - Find a suitable mapping
    - Compute distance in mapped space
  - Available techniques
    - MDS + global geodesic distance: ISO-MAP
    - Local distance approximation: LLE
    - …

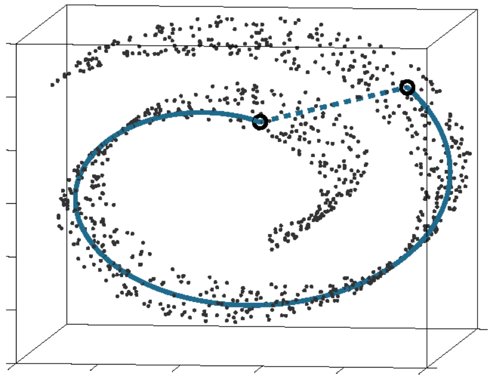
Classical Multi-dimensional Scaling

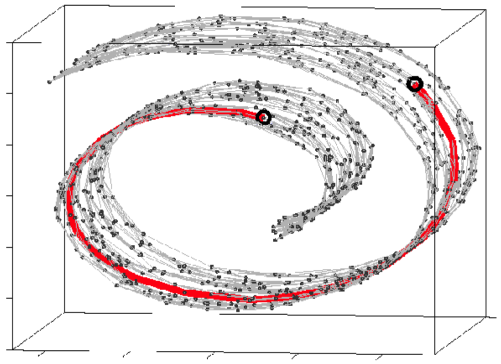
- MDS: 多维标度法
- Main idea:
    - Compute (match) distance between samples
    - Use SVD to find similarity

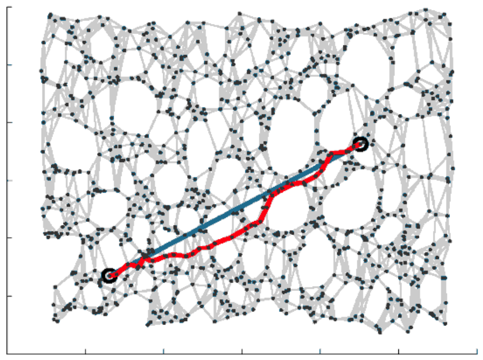
Isomap: (Science 2001) Isometric feature mapping

- Preserve the **intrinsic geometry** of the data.
- Use the **geodesic distances** on manifold between all pairs.

Three steps algorithm

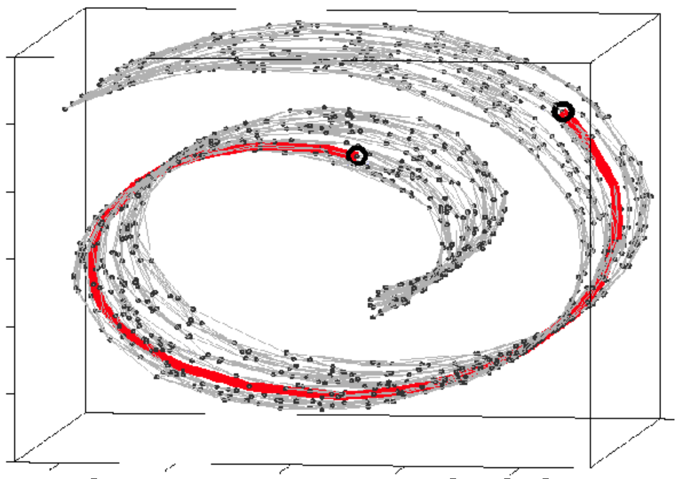






Isomap: Construct Neighborhood Graph

- Determine which points are neighbors, based on the distances d(i,j) .
  - K nearest neighbors
  - ε-radius



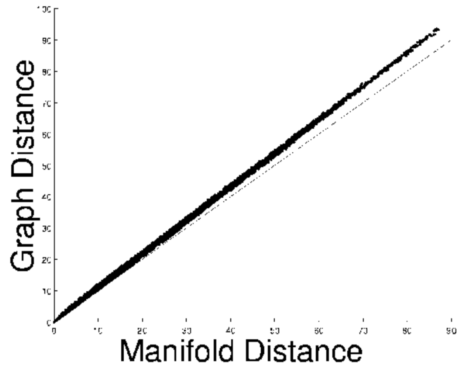
- Create a graph G, with edges between neighbors and distance weights.

Isomap: Compute Shortest Paths

- Estimate the geodesic distances.
- Compute all-pairs shortest paths in G.
- Can be done using Floyd's algorithm, $O(N^2 \ln N)$.

$$d_G(i,j) = d(i,j) \quad \text{neighborin g i, j}$$

$$d_G(i,j) = \infty \qquad \text{othewise}$$



$$\textit{for } \mathrm{k} = 1,2,\ldots,\mathrm{N}$$

$$d_G(i,j) = \min\{\, d_G(i,j),\ d_G(i,k) + d_G(k,j)\}$$

Isomap:
Construct d-dimensional Embedding

Classical MDS with $d_G(i,j)$,
minimize the cost function:

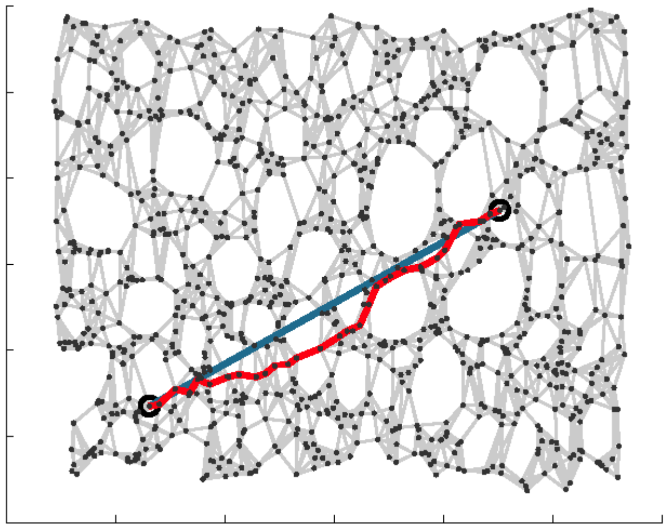
$$E = \left\| \tau(D_G) - \tau(D_Y) \right\|_{L^2}$$

$$where\ D_Y(i,j) = \left\| y_i - y_j \right\|$$

$$D_G(i,j) = d_G(i,j)$$

and

$$\tau(D) = \tfrac{-1}{2}(I - \tfrac{1}{N})D^{.2}(I - \tfrac{1}{N})$$



Solution: take top d eigenvectors of the matrix $\tau(D_G)$

Isomap: Classical Multi-dimensional Scaling

$$\mathbf{X}'\mathbf{X} = -\frac{1}{2}\mathbf{J}\mathbf{E}\mathbf{J}$$

E: Euclidian distance matrix

$$\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{M}\mathbf{J}$$

M: Manifold distance matrix

$$L(\hat{\mathbf{X}}) = \left\| -\frac{1}{2}\mathbf{J}\left(\mathbf{E}-\mathbf{M}\right)\mathbf{J} \right\| = \left\| \hat{\mathbf{X}}\hat{\mathbf{X}}' - \mathbf{B} \right\|.$$

$$\underbrace{\mathbf{B} = \mathbf{Q}\Lambda\mathbf{Q}' \qquad \hat{\mathbf{X}} = \mathbf{Q}_{+}\Lambda_{+}^{\frac{1}{2}}}$$

Eigen-structure analysis again

$$c_i = \sum_{a=1}^{m} x_{ia}^2$$

$$d_{ij}^2 = \sum_{a=1}^{m} (x_{ia} - x_{ja})^2$$

$$\mathbf{E} = \mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2\mathbf{X}\mathbf{X}'$$

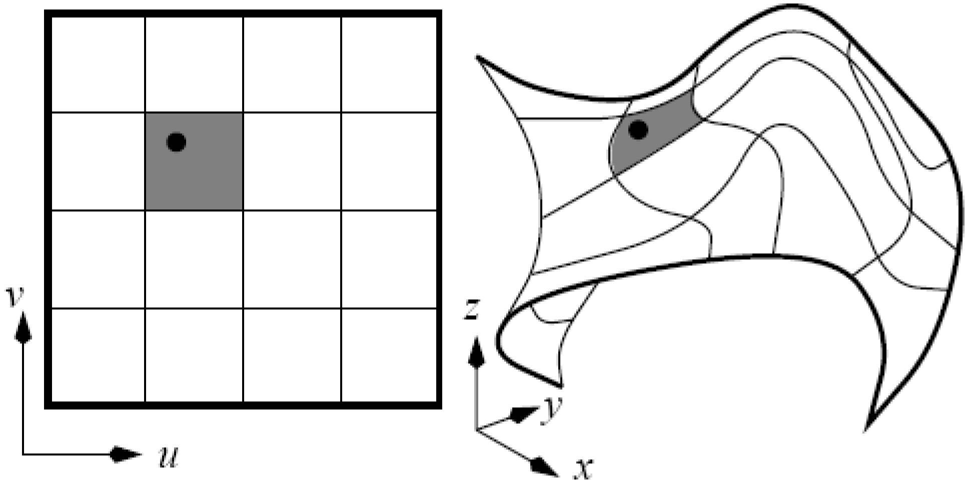
$$\mathbf{J} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}'$$

$$\begin{aligned} \mathbf{B} &= -\frac{1}{2}\mathbf{J}(\mathbf{c}\mathbf{1}' + \mathbf{1}\mathbf{c}' - 2\mathbf{X}\mathbf{X}')\mathbf{J} \\ &= -\frac{1}{2}\mathbf{J}\mathbf{c}\mathbf{0}' - \frac{1}{2}\mathbf{0}\mathbf{c}'\mathbf{J} + \mathbf{J}\mathbf{X}\mathbf{X}'\mathbf{J} \\ &= \mathbf{X}\mathbf{X}'. \end{aligned}$$

Isomap:
Classical Multi-dimensional Scaling (2D)

$$\begin{aligned}
\mathbf{J} &= \mathrm{eye}(n) - \mathrm{ones}(n)./n; \\
\mathbf{B} &= -0.5 * \mathbf{J} * \mathbf{M} * \mathbf{J}; \\
& \text{\% Find largest eigenvalues+their eigenvectors:} \\
[\mathbf{Q}, \mathbf{L}] &= \mathrm{eigs}(\mathbf{B}, 2, \text{'LM'}); \\
& \text{\% Extract the coordinates:} \\
\mathrm{newy} &= \mathrm{sqrt}(\mathbf{L}(1,1)). * \mathbf{Q}(:,1); \\
\mathrm{newx} &= \mathrm{sqrt}(\mathbf{L}(2,2)). * \mathbf{Q}(:,2);
\end{aligned}$$

Isomap: application texture mapping



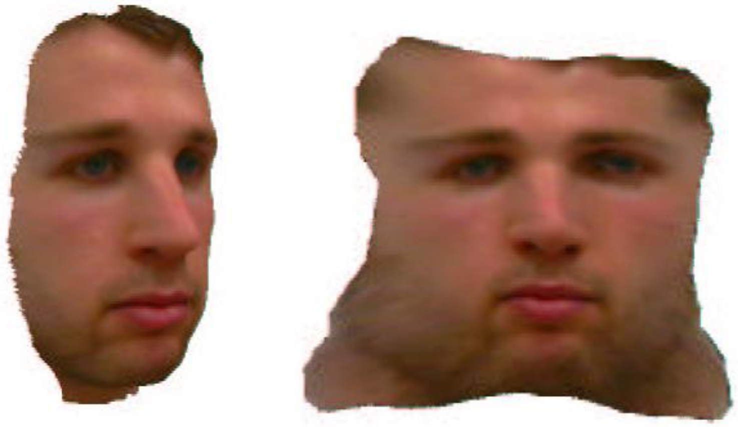
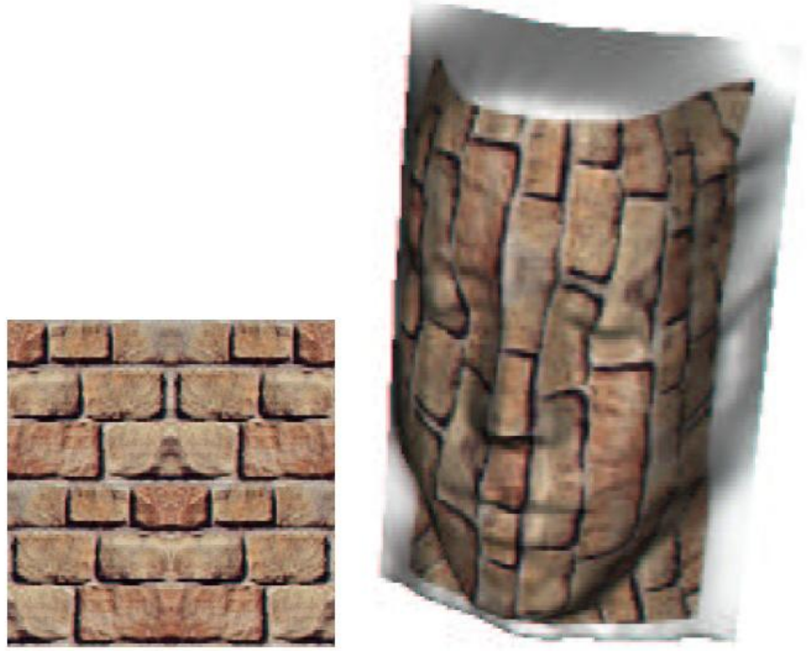
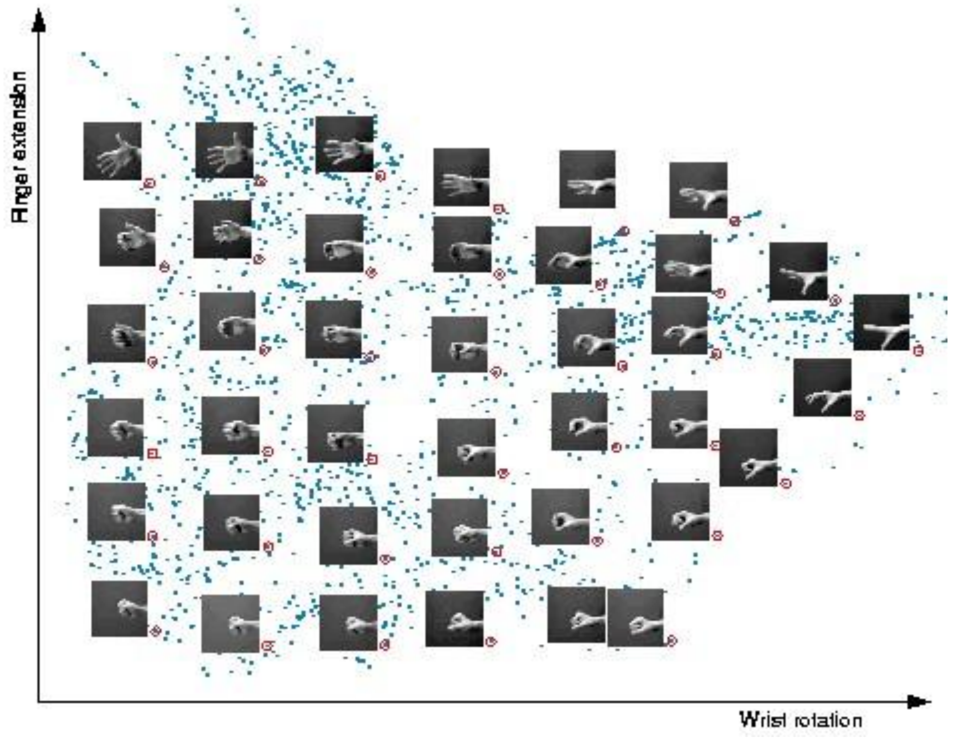


Fig. 3. An example of a face flattening. (a) A 3D reconstruction of a face. (b) The flattened texture image of the face.



Isomap: Examples

- *N*=2000 images 64x64 pixels K=6

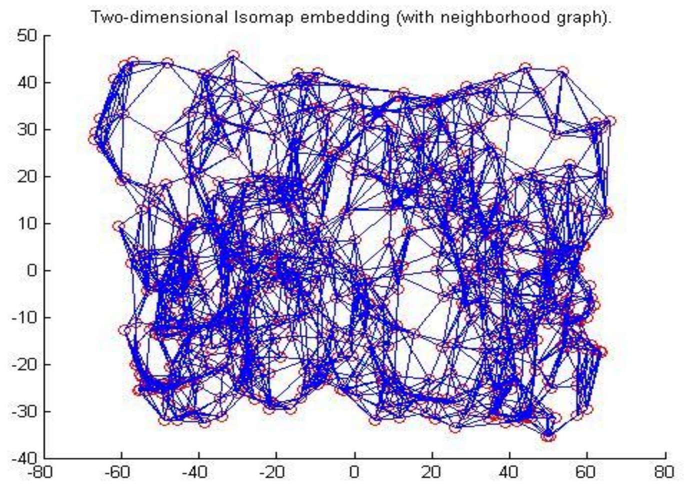


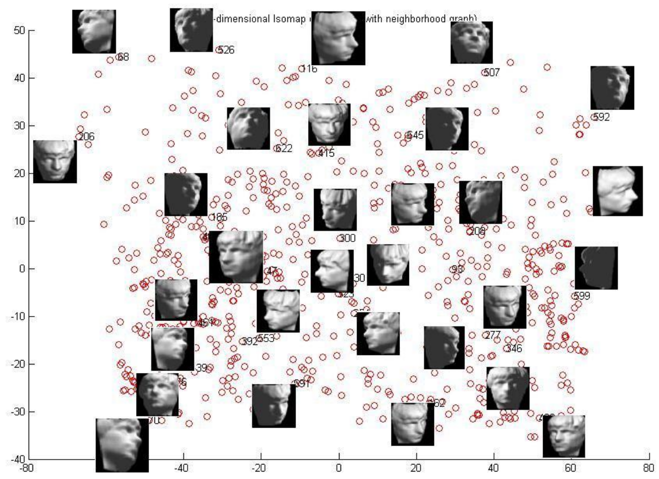
Isomap:
More Results

Input: 698
images of 64x64

K=7, d=2

Outputs:

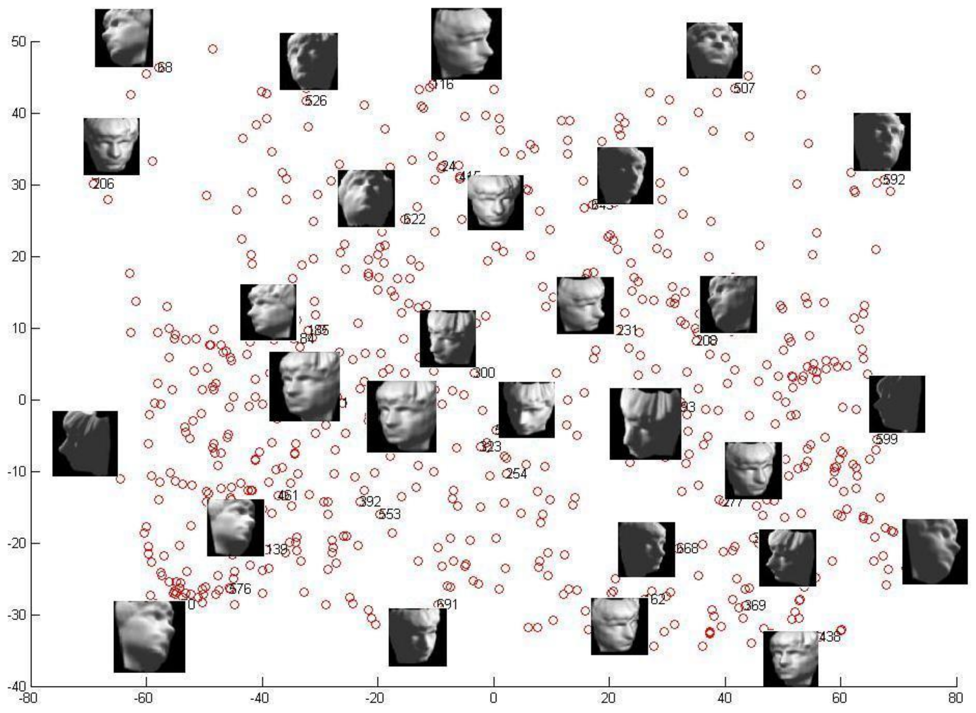




Isomap:
More Results

- Same inputs, but this time with d=3

698 images of 64x64 K=7



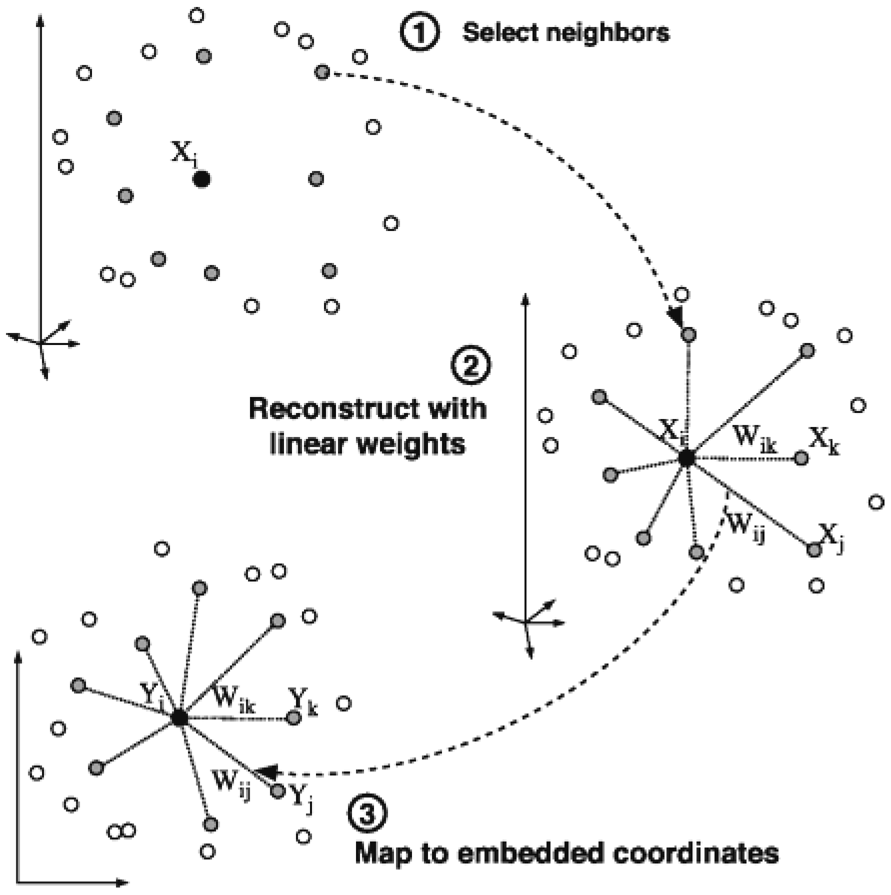
Locally Linear Embedding (LLE)

- Recovers global nonlinear structure from locally linear fits.
- Each data point and it's neighbors is expected to lie on or close to a locally linear patch.
- Each data point is constructed by it's neighbors:

$$\vec{\hat{X}}_i = \sum_j W_{ij} \vec{X}_j$$

$$W_{ij} = 0 \text{ if } \vec{X}_j \text{ is not a neighbor of } \vec{X}_i$$

LLE – main idea



LLE: Getting the Reconstruction Weights

- We want to minimize the error function:

$$\varepsilon(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

- With the constrains:

$$W_{ij} = 0 \quad \text{if } \vec{X}_j \text{ is not a neighbor of } \vec{X}_i$$

$$\sum_j W_{ij} = 1$$

- Solution (using Lagrange multipliers):

$$W_j = \sum_k C_{jk}^{-1} (\vec{X}\vec{\eta}_k + \lambda)$$

$$\lambda = 1 - \sum_{jk} C_{jk}^{-1} (\vec{X}\vec{\eta}_k) \Big/ \sum_{jk} C_{jk}^{-1}$$

LLE: Find Embedded Coordinates

- Choose d-dimensional coordinates, Y, to minimize: $$\phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2$$

Under: $\sum_i \vec{Y}_i = \vec{0}, \quad \frac{1}{N} \sum_i \vec{Y}\vec{Y}^T = I$

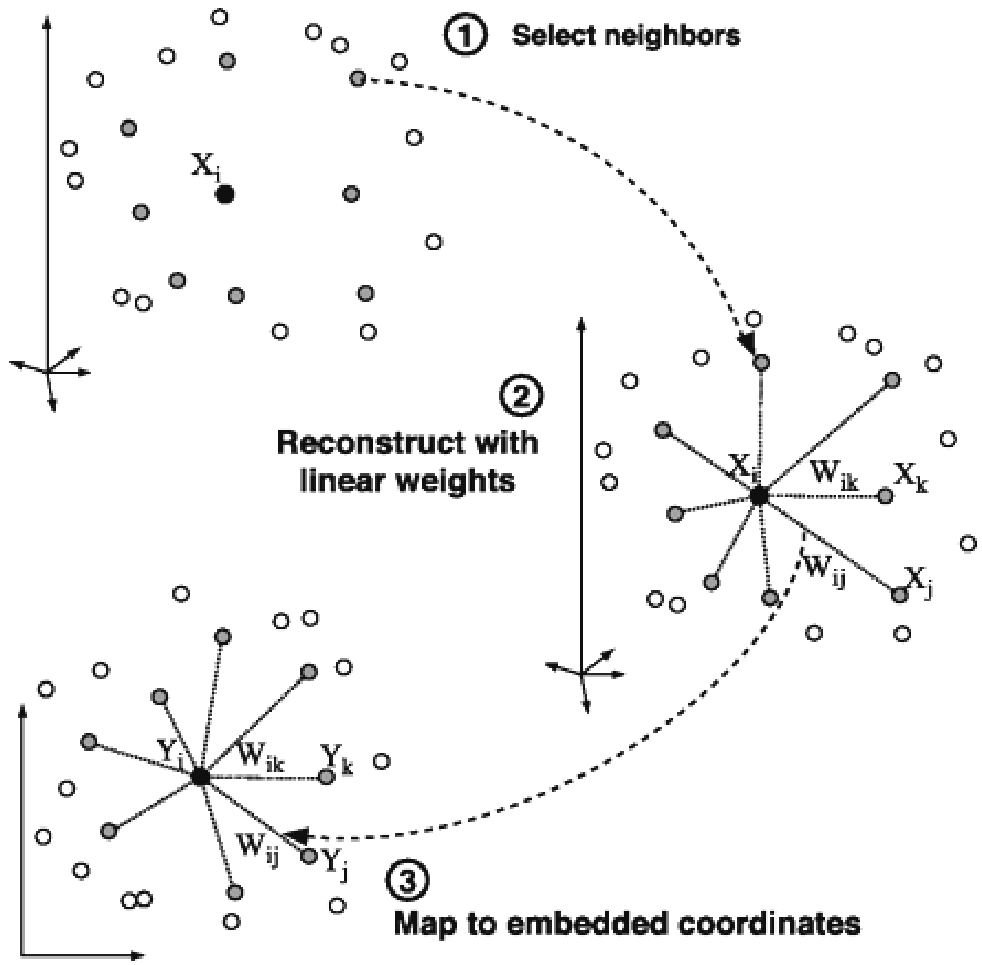
Quadratic form: $$\phi(Y) = \sum_{ij} M_{ij} (\vec{Y}_i \vec{Y}_j)$$

where: $$M = (I - W)^T (I - W)$$

- Solution: compute bottom d+1 eigenvectors of M. (discard the last one)

LLE: Summary

- Input: N data items in D dimension (X).
- Output: d < D dimensional embedding coordinates (Y) for the input points.



LLE:
Algorithm Pseudocode (I)

```
Find neighbors in X space
For i=1:N
    compute the distance from Xi to every other point Xj
    find the K smallest distances
    assign the corresponding points to be neighbors of Xi
end
```

http://www.cs.toronto.edu/~roweis/lle/algorithm.html

LLE: Algorithm Pseudocode (II)

```
Solve for reconstruction weights W.
for i=1:N
    create matrix Z consisting of all neighbors of Xi
    subtract Xi from every column of Z
    compute the local covariance C=Z'*Z
    solve linear system C*w = 1 for w
    set Wij=0 if j is not a neighbor of I
    set the remaining elements in the ith row of W equal to
    w/sum(w);
end
```

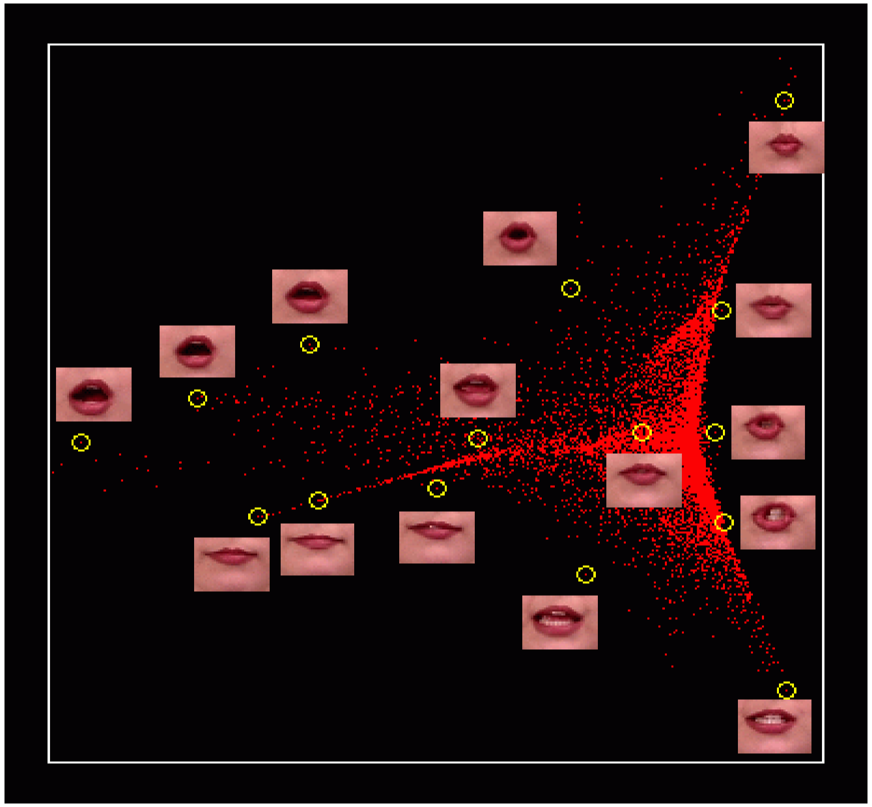
LLE: Algorithm Pseudocode (III)

Compute embedding coordinates Y using weights W.

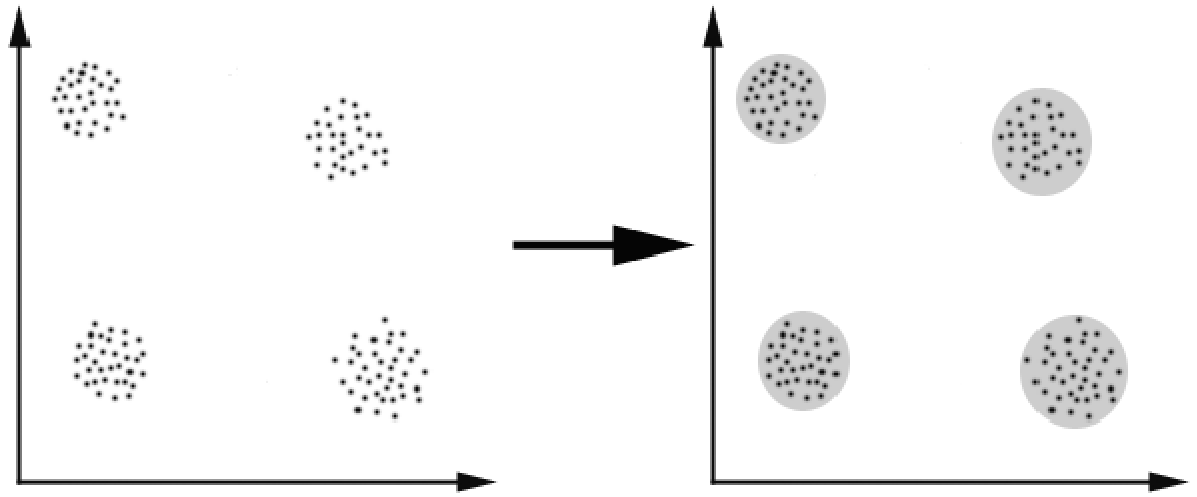
```
    create sparse matrix M = (I-W)'*(I-W)

    find bottom d+1 eigenvectors of M (corresponding to
    the d+1 smallest eigenvalues)

    set the $q$-th ROW of Y to be the $q$+1 smallest
    eigenvector (discard the bottom eigenvector [1,1,1,1...]
    with eigenvalue zero)
```

LLE: Example

- N=8588 (RGB) images of lips of size 108x84. D=27216
- Num of neighbors K=16



Clustering



Spectral clustering

- The spectral clustering method we define relies on a random walk representation over the points. We construct this in three steps

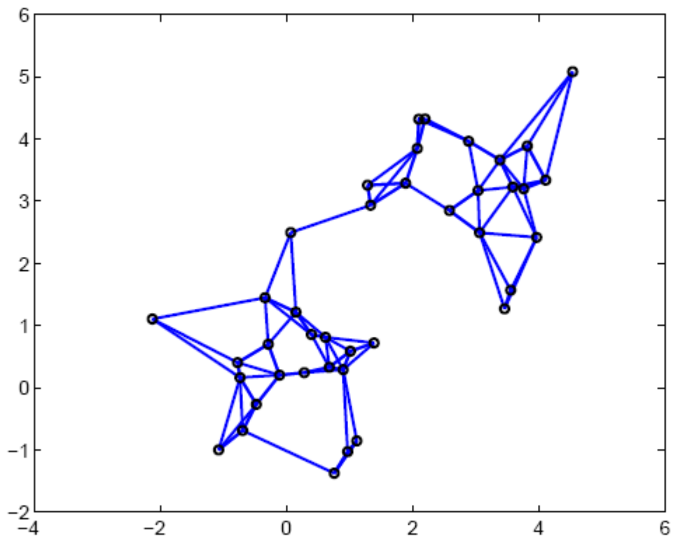
1. a nearest neighbor graph
2. similarity weights on the edges:

$$W_{ij} = \exp\{-\beta \|\mathbf{x}_i - \mathbf{x}_j\|\}$$

   where $W_{ii} = 1$ and the weight is zero for non-edges.

3. transition probability matrix

$$P_{ij} = W_{ij} / \sum_{j'} W_{ij'}$$



Properties of the random walk

- If we start from i_0, the distribution of points i_t that we end up in after t steps is given by

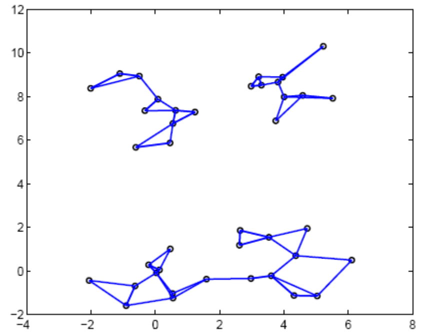
$$i_1 \sim P_{i_0\,i_1}, \qquad P_{ij} = \frac{W_{ij}}{W_{i\cdot}}, \quad \text{where } W_{i\cdot} = \sum_j W_{ij}$$

$$i_2 \sim \sum_{i_1} P_{i_0,i_1} P_{i_1\,i_2} = [P^2]_{i_0\,i_2},$$

$$i_3 \sim \sum_{i_1} \sum_{i_2} P_{i_0,i_1} P_{i_1\,i_2} P_{i_2\,i_3} = [P^3]_{i_0\,i_3},$$

$$\ldots$$

$$i_t \sim [P^t]_{i_0\,i_t}$$

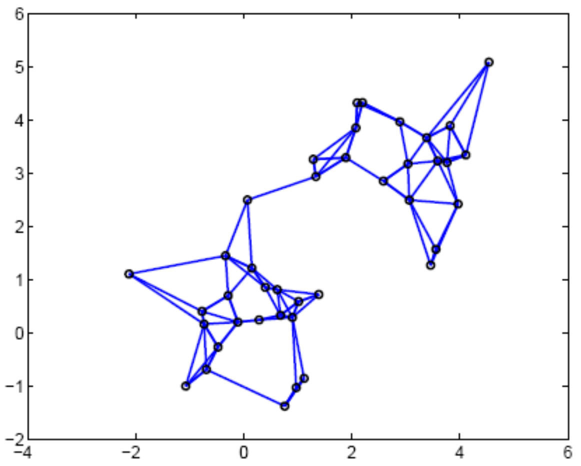


where $P^t = PP\ldots P$ (t matrix products) and $[\cdot]_{ij}$ denotes the i,j component of the matrix.

Random walk and clustering

- The distributions of points we end up in after t steps converge as t increases. If the graph is connected, the resulting distribution is independent of the starting point

  Even for large t, the transition probabilities $[P^t]_{ij}$ have a slightly higher probability of transitioning within "clusters" than across; we want to recover this effect from eigenvalues/vectors



Eigenvalues/vectors and spectral clustering

- Let W be the matrix with components W_{ij} and D a diagonal matrix such that $D_{ii} = \sum_j W_{ij}$. Then

$$P = D^{-1}W$$

- To find out how P^t behaves for large t it is useful to examine the eigen-decomposition of the following symmetric matrix

$$D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = \lambda_1 \mathbf{z}_1 \mathbf{z}_1^T + \lambda_2 \mathbf{z}_2 \mathbf{z}_2^T + \ldots + \lambda_n \mathbf{z}_n \mathbf{z}_n^T$$

where the ordering is such that $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.

Eigenvalues/vectors cont'd

- The symmetric matrix is related to P^t since

$$(D^{-\frac{1}{2}}WD^{-\frac{1}{2}})\cdots(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}) = D^{\frac{1}{2}}(P\cdots P)\,D^{-\frac{1}{2}}$$

This allows us to write the t step transition probability matrix in terms of the eigenvalues/vectors of the symmetric matrix

$$\begin{aligned} P^t &= D^{-\frac{1}{2}}\left(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}\right)^t D^{\frac{1}{2}} \\ &= D^{-\frac{1}{2}}\left(\lambda_1^t \mathbf{z}_1\mathbf{z}_1^T + \lambda_2^t \mathbf{z}_2\mathbf{z}_2^T + \ldots + \lambda_n^t \mathbf{z}_n\mathbf{z}_n^T\right) D^{\frac{1}{2}} \end{aligned}$$

where $\lambda_1 = 1$ and

$$P^\infty = D^{-\frac{1}{2}}\left(\mathbf{z}_1\mathbf{z}_1^T\right) D^{\frac{1}{2}}$$

Eigenvalues/vectors and spectral clustering

- We are interested in the largest correction to the asymptotic limit

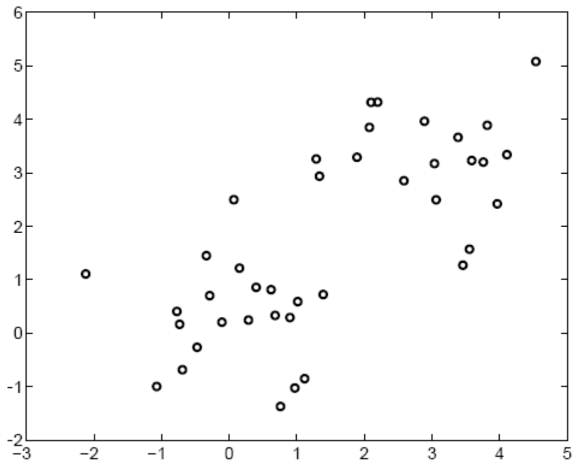
$$P^t \approx P^\infty + D^{-\frac{1}{2}} \left(\lambda_2^t \, \mathbf{z}_2 \mathbf{z}_2^T \right) D^{\frac{1}{2}}$$

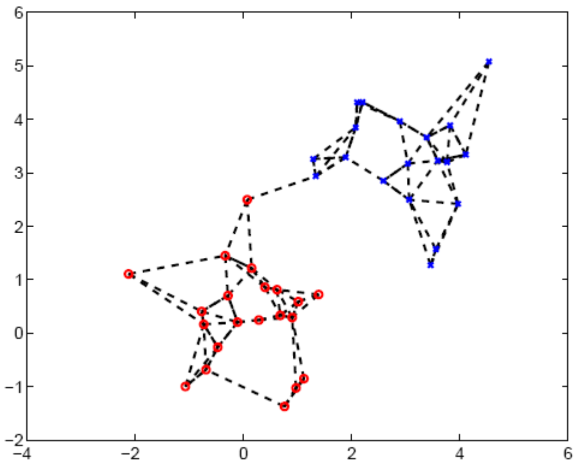
  Note: $[\mathbf{z}_2 \mathbf{z}_2^T]_{ij} = z_{2i} z_{2j}$ and thus the largest correction term increases the probability of transitions between points that share the same sign of z_{2i} and decreases transitions across points with different signs

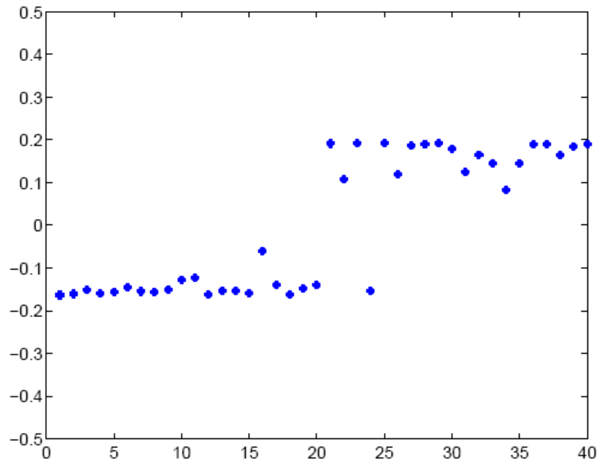
- Binary spectral clustering: we divide the points into clusters based on the sign of the elements of $\mathbf{z}_2$

$$z_{2j} > 0 \Rightarrow \text{ cluster 1, otherwise cluster 0}$$

Spectral clustering: example







Components of the eigenvector corresponding to the second largest eigenvalue

Reference papers of SC

- A. Y. Ng, M. I. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, NIPS, (2001)
- Y. Weiss, *Segmentation using eigenvectors: a unifying view*. ICCV, (1999)
- J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE TPAMI, 22 (2000)

- And more about image segmentations …
  - Graph cut
  - Mean-shift

Classical methods on cluster distance

- A *linkage* method: we have to be able to measure distances between clusters of examples C_k and C_l

a) Single linkage:

Nearest neighbor

$$d_{kl} = \min_{i \in C_k, j \in C_l} d(\mathbf{x}_i, \mathbf{x}_j)$$

b) Average linkage:

$$d_{kl} = \frac{1}{|C_l|\,|C_k|} \sum_{i \in C_k, j \in C_l} d(\mathbf{x}_i, \mathbf{x}_j)$$

c) Centroid linkage:

$$d_{kl} = d(\bar{\mathbf{x}}_k, \bar{\mathbf{x}}_l), \quad \bar{\mathbf{x}}_l = \frac{1}{|C_l|} \sum_{i \in C_l} \mathbf{x}_i$$

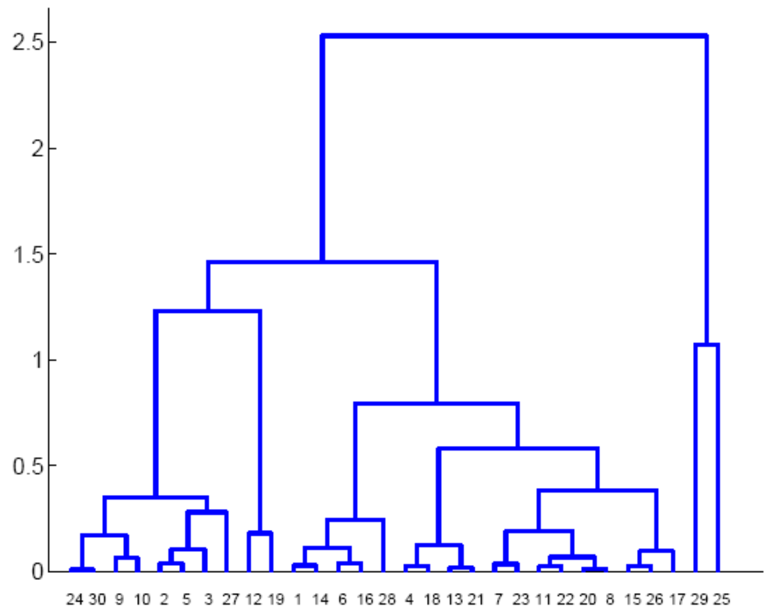
Hierarchical (bottom-up) clustering

- Hierarchical agglomerative clustering: we sequentially merge the pair of "closest" points/clusters

- The procedure
  1. Find two closest points (clusters) and merge them
  2. Proceed until we have a single cluster (all the points)
- Two prerequisites:
  1. distance measure d(xi, xj) between two points
  2. distance measure between clusters (cluster linkage)

Hierarchical (bottom-up) clustering

- A dendrogram representation of hierarchical clustering



The height of each pair represents the distance between the merged clusters; the specific linear ordering of points is chosen for clarity

Clustering vs. Classification

- **Clustering**
  - **Instance:** $\{\mathbf{x}_i\}_{i=1}^{N}$
  - **Learn:** $< \mathbf{x}_i, t_i >$ and/or mapping from $\mathbf{x}$ to $t(\mathbf{x})$

- **Classification/Regression**
  - **Instance:** $< \mathbf{x}_i, t_i >$
  - **Learn:** mapping from $\mathbf{x}$ to $t(\mathbf{x})$

The End

新浪微博：@浙大张宏鑫

微信公众号：