数字图像处理
Digital Image Processing

刘利刚
lgliu@ustc.edu.cn
http://staff.ustc.edu.cn/~lgliu

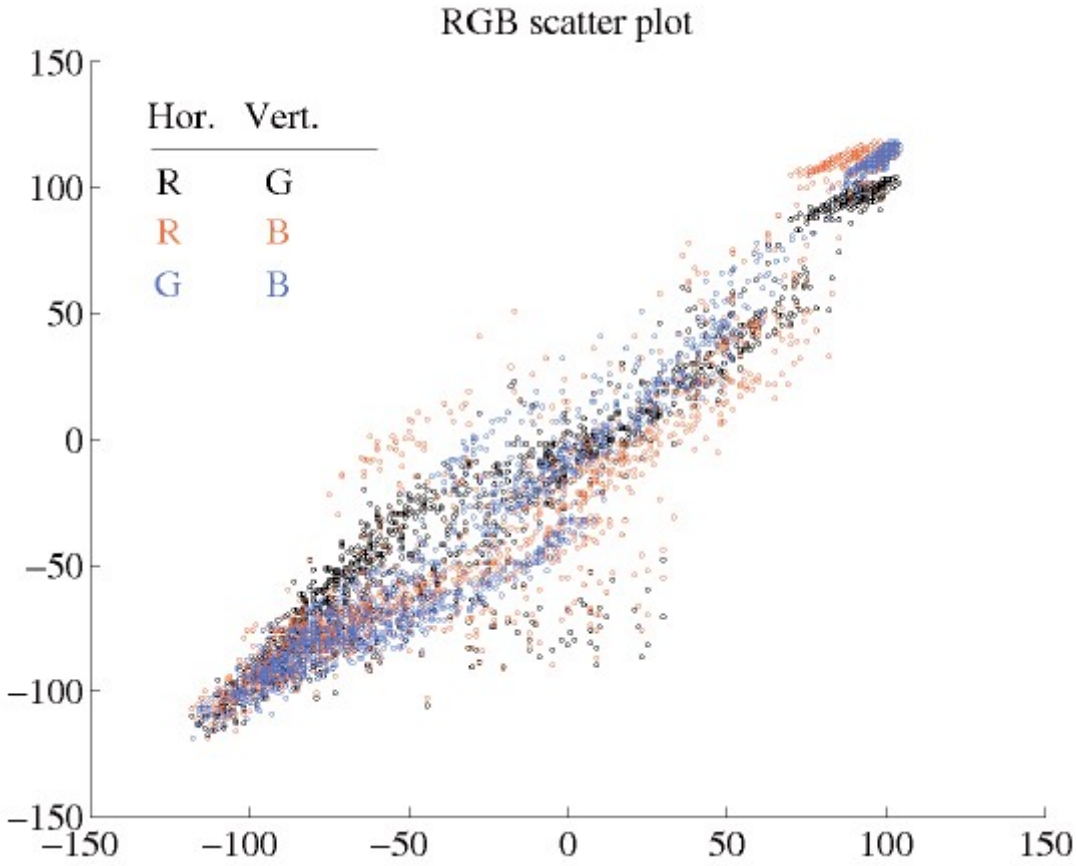
Image as Points (2)

Structure Preserving Point Processing

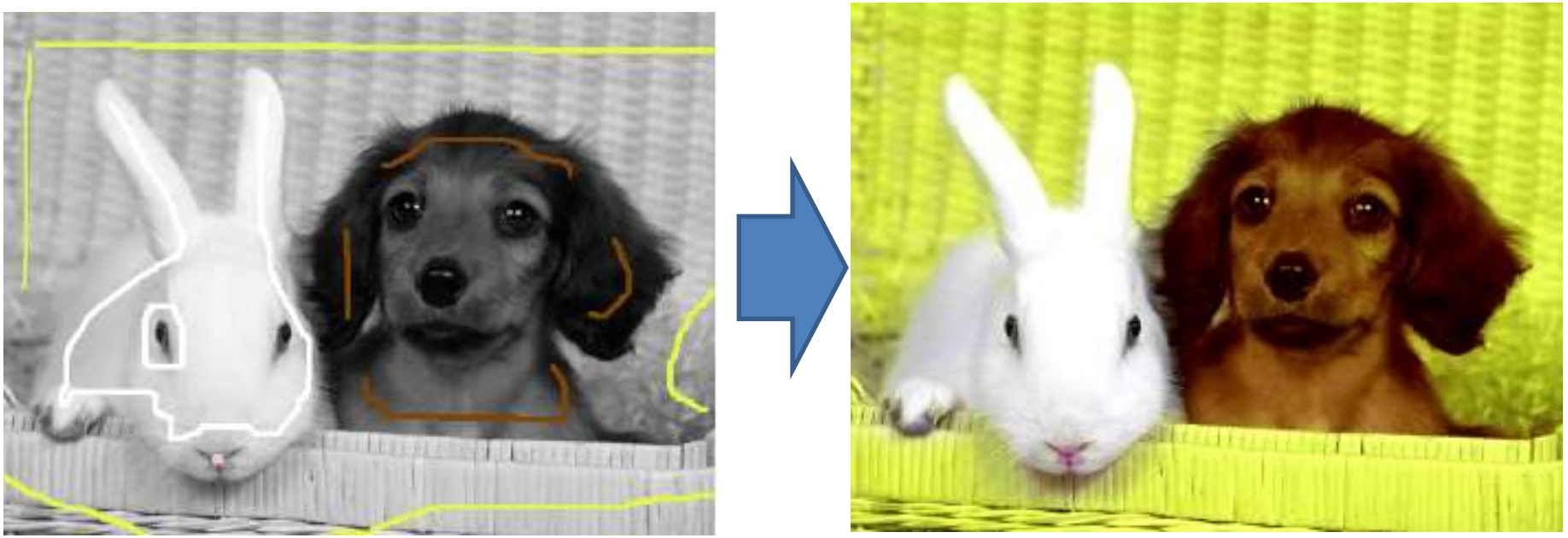
Point Processing of Images

- In a digital image, point = pixel.
- Point processing transforms a pixel's value as function of its value alone;
- it does not depend on the values of the pixel's neighbors.

图像：3D线性空间的点集



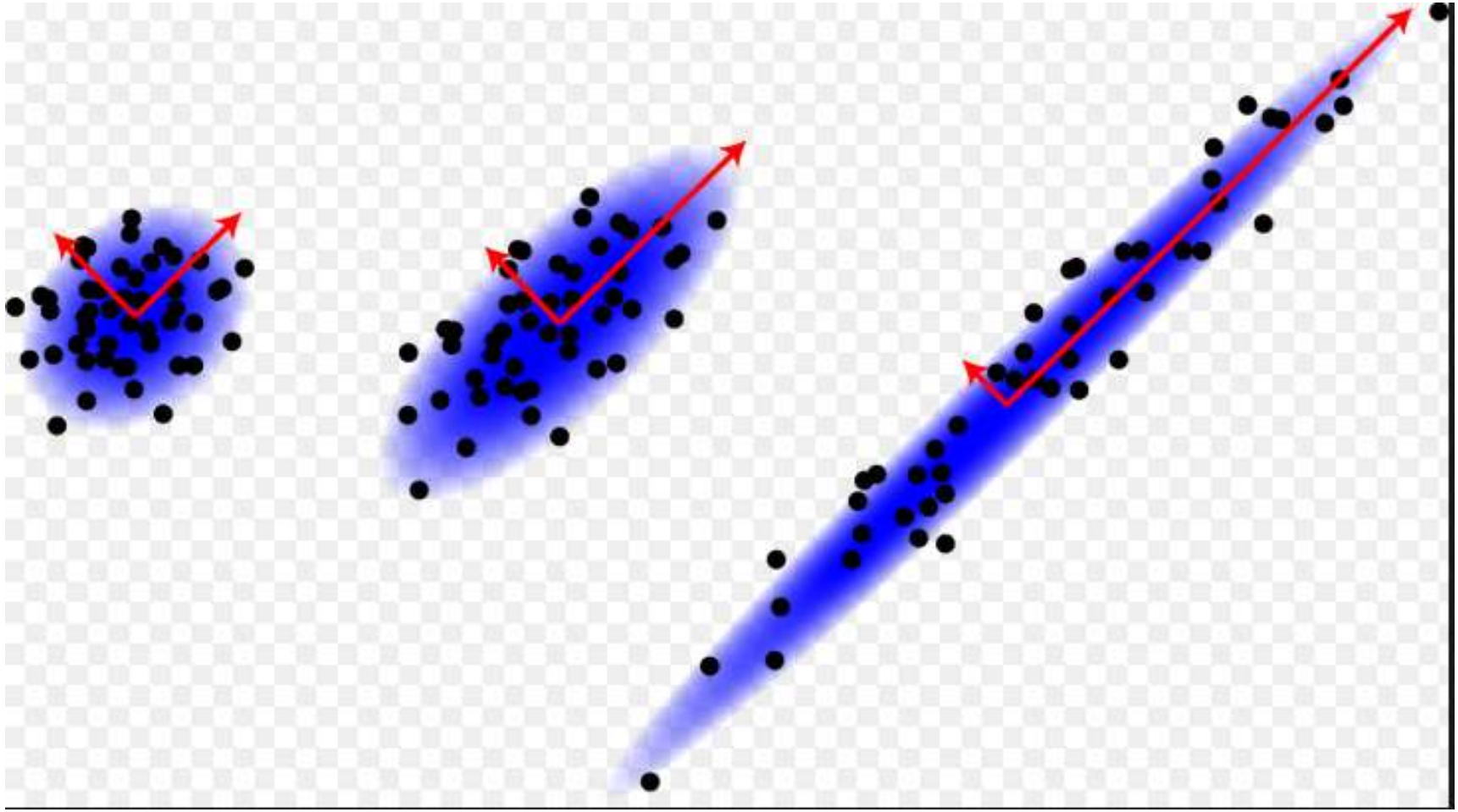
Problem: Editing Propagation



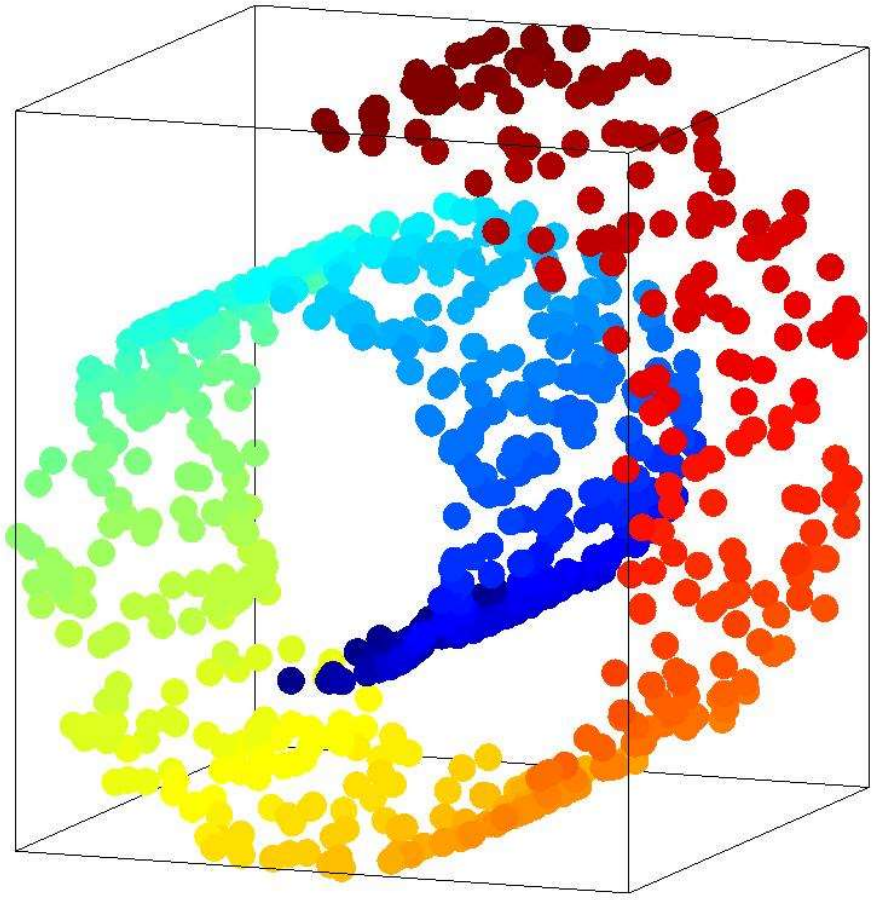
Goal: Preserving gradients, structure…?

Structure as a Manifold

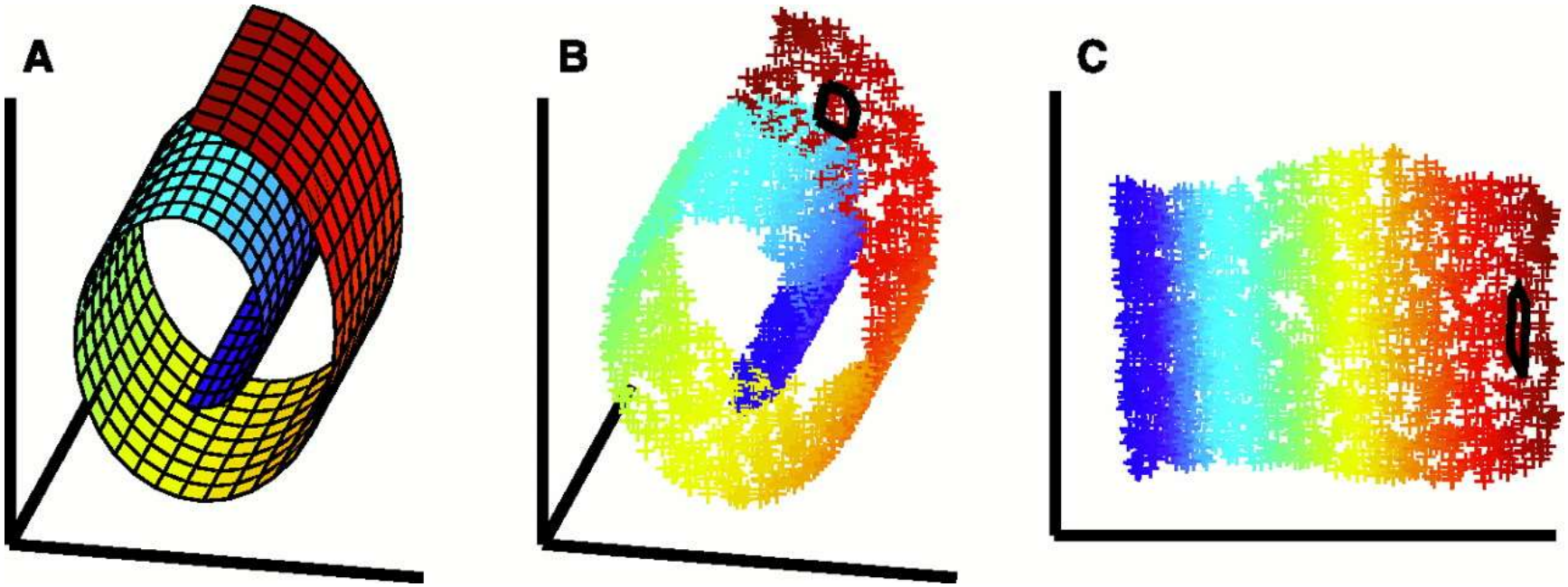
Ambient points always has low-dimensional structure!



How about these points?



2D manifold in 3D



数据的观察维数

- 任何数据都是某个高维空间的一个数（向量）

地震波、蛋白质结构
基因序列…

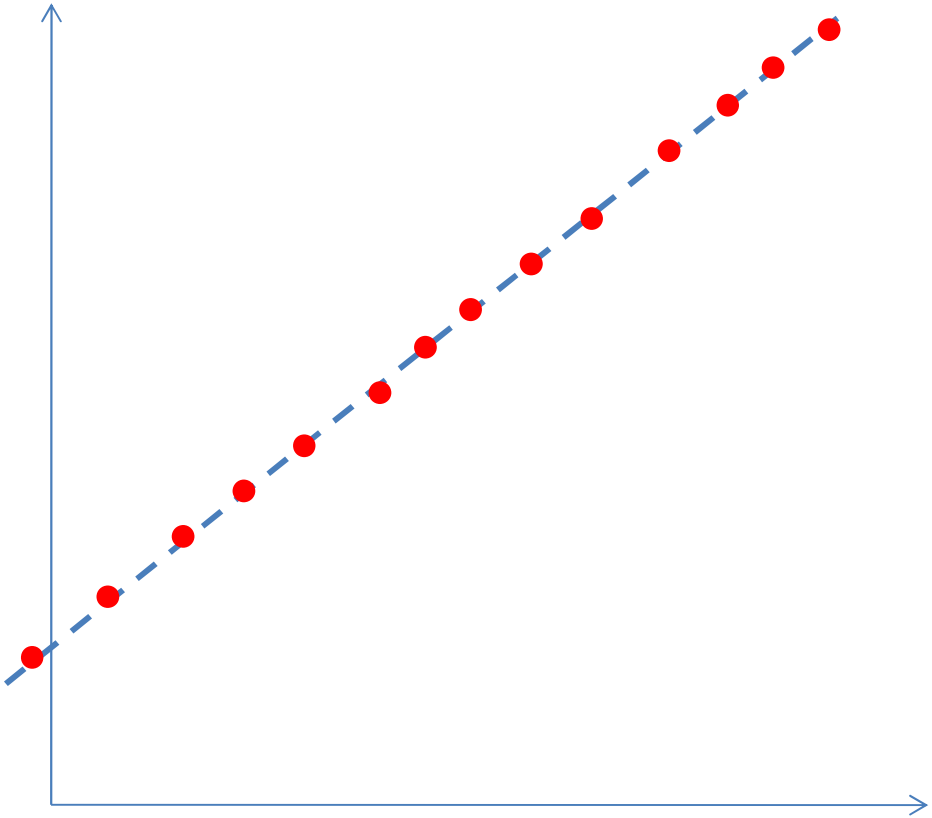
“特征”

$$X = (x_1, x_2, \cdots, x_N) \in R^N$$

数据的维数真有那么高吗？

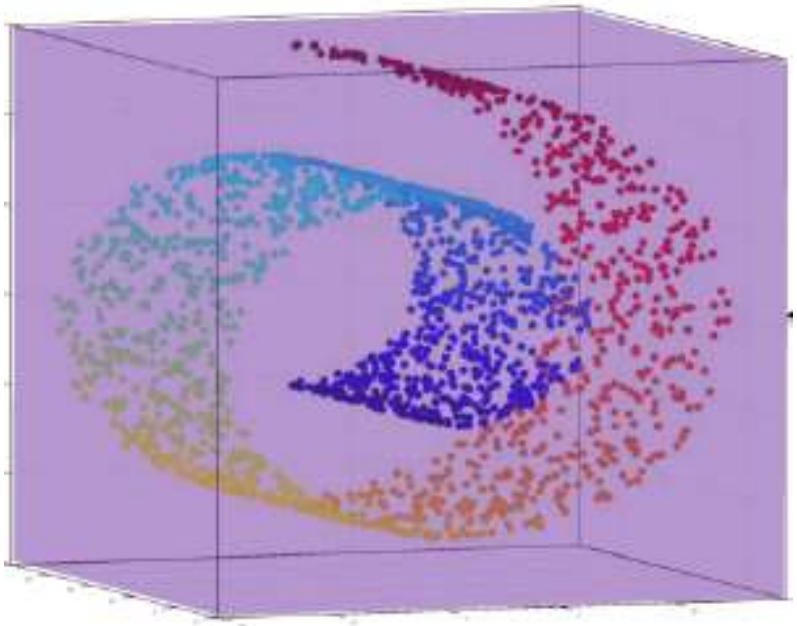
61.562198 21.702432
69.589896 23.116031
69.689998 30.561331
63.219798 29.909931
78.298497 24.213831
77.958698 31.596431
77.822498 39.146431
70.139797 38.301631
64.147498 37.712831
96.651898 26.682431
95.903298 34.240231
86.893197 32.779931
87.502498 25.348631
95.225397 41.707531
86.403297 40.250431
94.292697 56.312032
85.767398 55.206031
86.029197 47.734331
77.722297 46.807731

观察的维数

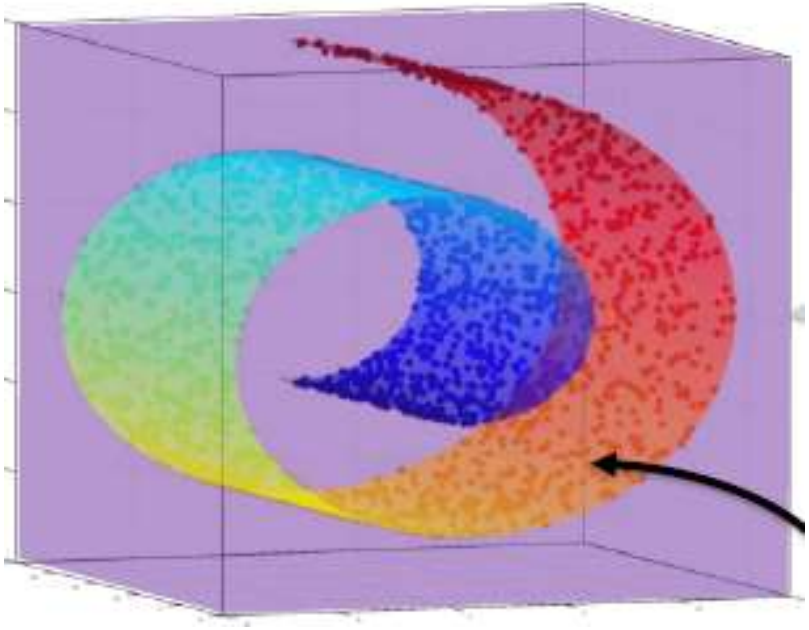


真实的维数

数据的真实维数可能很低



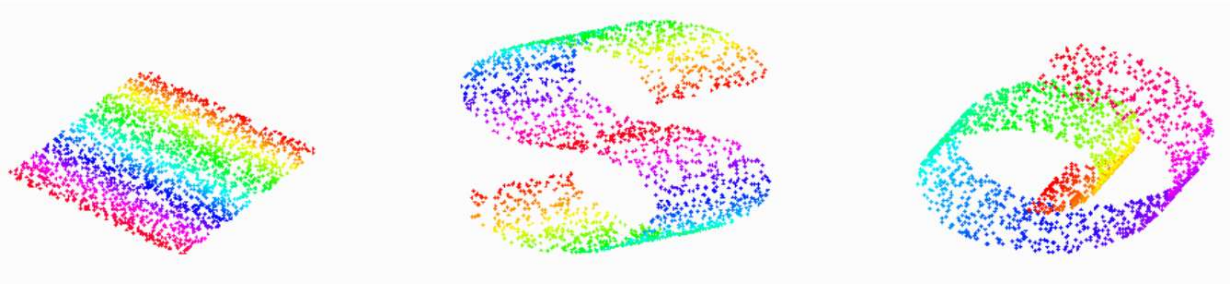
观察的维数



真实的维数

目标：高维数据中的低维结构

- 数据 >> 信息
- 数据 = 信息 + 不相关数据
- 目标：从不完全数据中“抽取”信息（结构）



“数据科学”

- 两个目标：
  - 数据表达（representation）
  - 内在结构（intrinsic structures）
- 两个任务：
  - 聚类（clustering）： unsupervised
  - 分类（classification）： supervised
    - Regression/fitting, deep learning…

数据的表达：特征/编码

- 量化
- 压缩
- 数据变换：从不同的空间来看数据

数据变换的例子：Wavelets

- Feature map

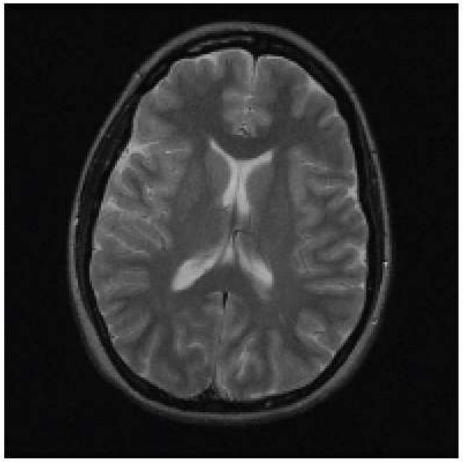
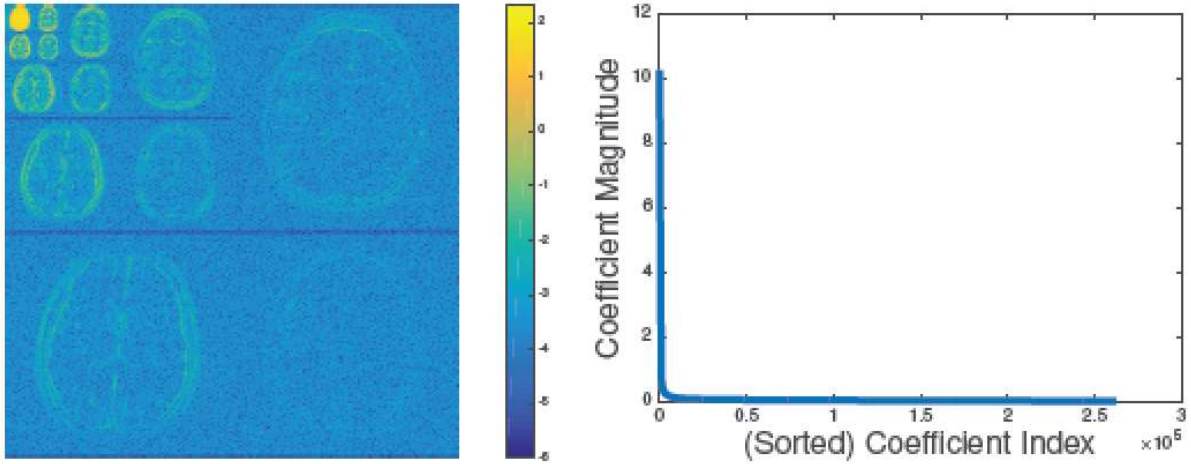


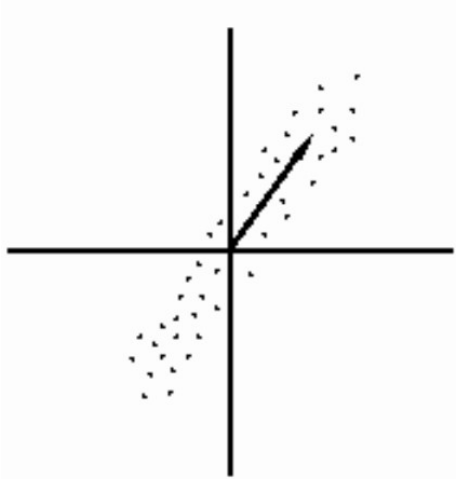
image $I(v)$

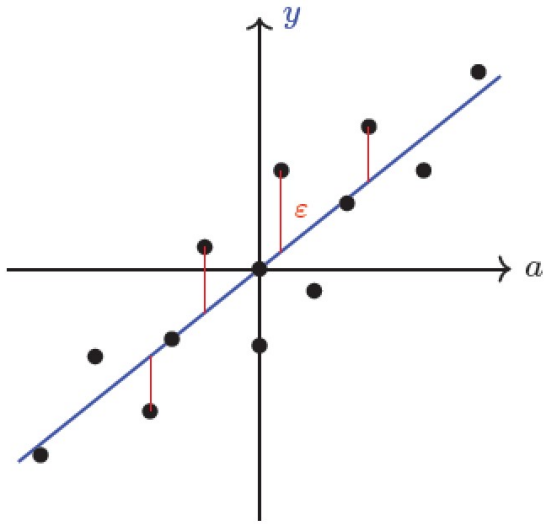


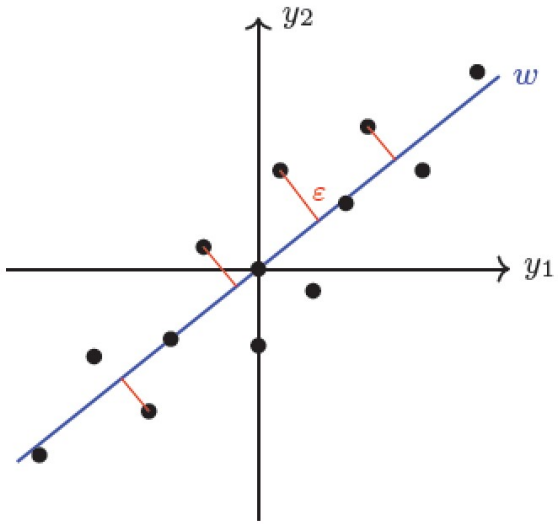
wavelet coefficients x: $I = \Psi[x]$.

Principal Component Analysis (PCA)

- Find linear subspace projection which preserves the data locations (under quadratic error)







内在结构

- 哪些数据是一组的（相似的）？

- 相似性的“度量”？
  - 距离
  - 线性空间（同一组基表达）
  - 非线性空间（流形空间）

内在结构

- 哪些数据是一组的（相似的）？

- 相似性的“度量”？
  - 距离
  - 线性空间（同一组基表达）
  - 非线性空间（流形空间）

Clustering Analysis

TWO Distinct Clustering Approaches

- Two Basic Approaches for Doing Clustering:
  - Hierarchical Clustering
  - Non-Hierarchical Clustering
- **Hierarchical Clustering** – final clusters are built following a distinct set of sequential steps
- **Non-Hierarchical Clustering** – Clusters are built in such a way that if M clusters are built there is no guarantee that putting together two of the clusters would give rise to the same (M-1) clusters built separately by the method.

TWO Distinct Clustering Approaches

- Two Basic Approaches for Doing Clustering:
  - Hierarchical Clustering
  - Non-Hierarchical Clustering
- **Hierarchical Clustering** – final clusters are built following a distinct set of sequential steps
- **Non-Hierarchical Clustering** – Clusters are built in such a way that if M clusters are built there is no guarantee that putting together two of the clusters would give rise to the same (M-1) clusters built separately by the method.

K-Means Clustering (Non-Hierarchical Clustering)

- K-Means Clustering is a **non-hierarchical method** in the sense that if one has 2 clusters, say, generated by pre-specifying 2 means (centroids) in the K-means algorithm and 3 clusters generated by pre-specifying 3 means in the K-means algorithm, then it may be the case that no combination of any two clusters of the 3 cluster group can give rise to the 2 cluster grouping.

Steps in the K-Means Clustering Approach

- Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \cdots . \mathbf{x}_\mathbf{n})$ where each observation is a d-dimensional real vector, then K-means clustering aims to partition the n observations into K sets (K < n), $\boldsymbol{S} = \{S_1, S_2, \cdots . S_K\}$ so as to minimize the within-cluster sum of squares (WCSS):

$$arg \min_{\boldsymbol{S}} \sum_{i=1}^{K} \sum_{\boldsymbol{x_j} \in S_i} \left\| \boldsymbol{x_j} - \boldsymbol{\mu_i} \right\|^2 \qquad (1)$$

where $\boldsymbol{\mu_i}$ is the mean of the points in S_i . Now minimizing (1) can, in theory, be done by the **integer programming method** but this can be extremely time-consuming. Instead the **Lloyd algorithm** is more often used.

Lloyd Algorithm

- The steps of the **Lloyd algorithm** are as follows. Given the initial set of K-means $\boldsymbol{m}_1^{(1)}, \cdots, \boldsymbol{m}_K^{(1)}$ which can be specified randomly or by some heuristic, the algorithm proceeds by alternating between two steps:
- Assignment Step: Assign each observation to the cluster with the closest mean

$$S_i^{(t)} = \left\{ \boldsymbol{x_j} : \left\| \boldsymbol{x_j} - \boldsymbol{m_i^{(t)}} \right\| \leq \left\| \boldsymbol{x_j} - \boldsymbol{m_{i^*}^{(t)}} \right\| \right\} \text{ for all } i^* = 1,2,\cdots,K. \qquad (2)$$

- Update Step: Calculate the **new means** to be the centroids of the observations in the clusters, i.e.

$$\boldsymbol{m_i^{(t+1)}} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\boldsymbol{x_j} \in S_i^{(t)}} \boldsymbol{x_j} \text{ for } i = 1,2,\cdots,K. \qquad (3)$$

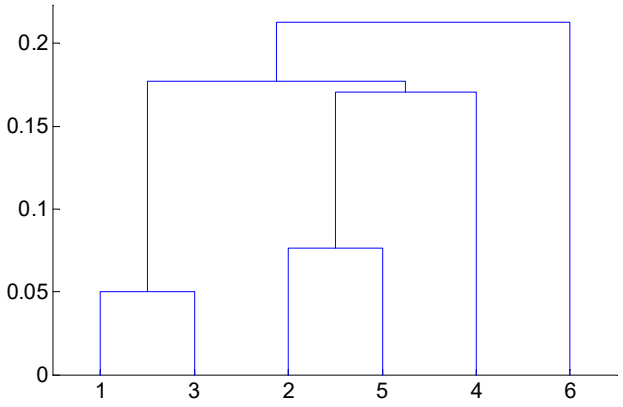
- Repeat the Assignment and Update steps until WCSS (equation (1)) no longer changes. Then the centroids and members of the K clusters are determined.
- **Note:** When using random assignment of the K-means to start the algorithm, one might try several starting point K-means and then choose the "best" starting point to be the random K-means that produces the smallest WCSS among all of the random starting points tried in the K-means procedure.
- Regardless of the clustering technique used, one should strive to choose clusters that are interpretable and make sense given the domain-specific knowledge that we have about the problem at hand.

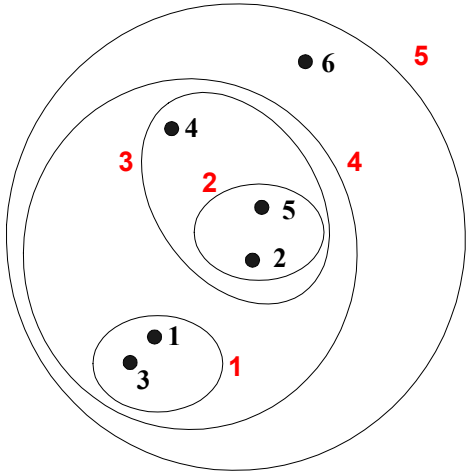
TWO Distinct Clustering Approaches

- Two Basic Approaches for Doing Clustering:
  - Hierarchical Clustering
  - Non-Hierarchical Clustering
- **Hierarchical Clustering** – final clusters are built following a distinct set of sequential steps
- **Non-Hierarchical Clustering** – Clusters are built in such a way that if M clusters are built there is no guarantee that putting together two of the clusters would give rise to the same (M-1) clusters built separately by the method.

Hierarchical Clustering

- Produces a set of ***nested clusters*** organized as a hierarchical tree
- Can be visualized as a **dendrogram**
  - A tree-like diagram that records the sequences of merges or splits





Strengths of Hierarchical Clustering

- No assumptions on the number of clusters
  - Any desired number of clusters can be obtained by ‘cutting’ the dendogram at the proper level

- Hierarchical clusterings may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

Hierarchical Clustering

- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Complexity of hierarchical clustering

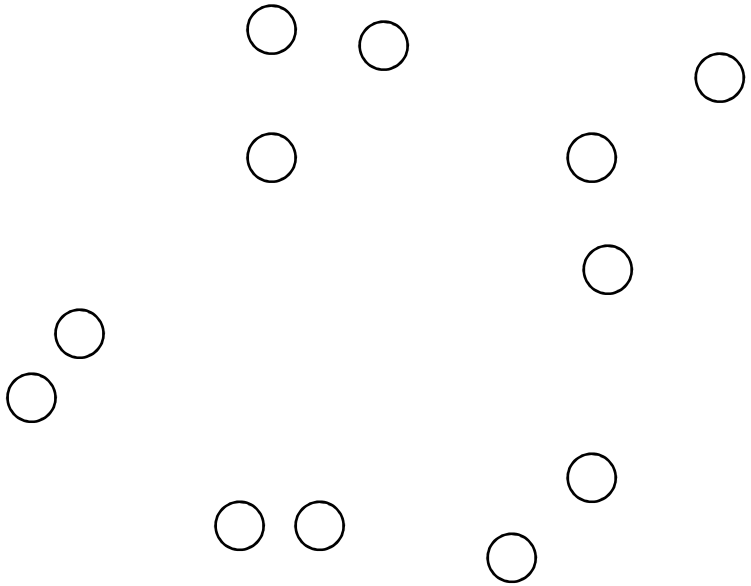
- Distance matrix is used for deciding which clusters to merge/split
- At least quadratic in the number of data points
- Not usable for large datasets

Agglomerative clustering algorithm

- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

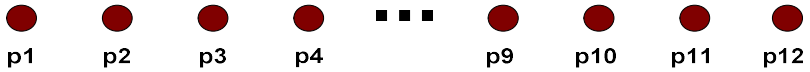
Input/ Initial setting

- Start with clusters of individual points and a distance/proximity matrix



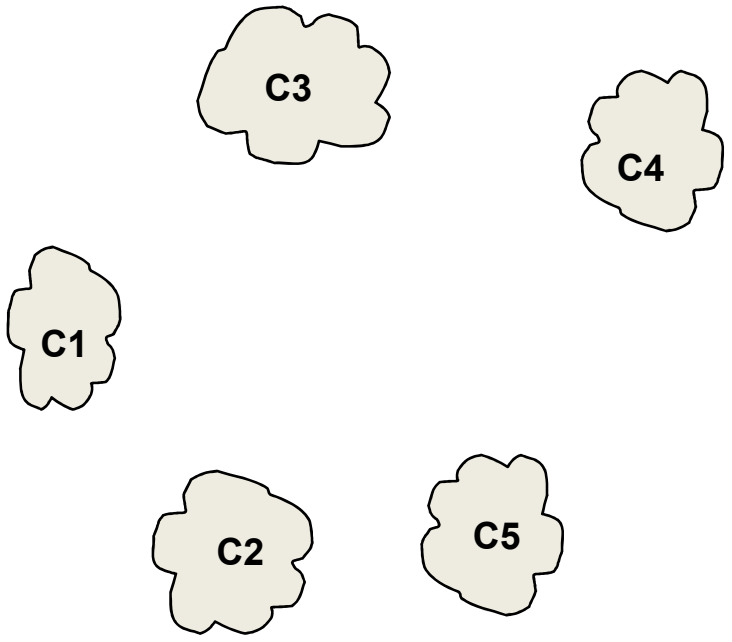
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						

Distance/Proximity Matrix



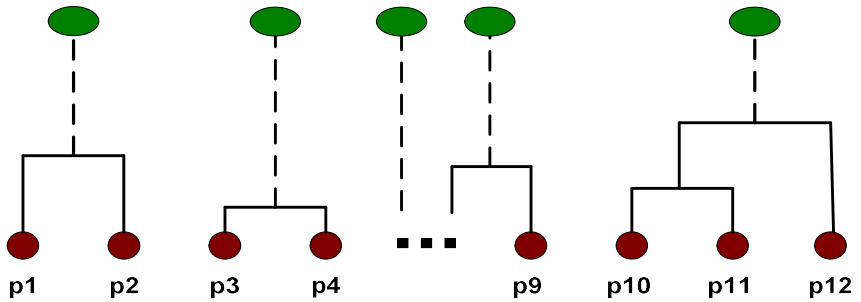
Intermediate State

- After some merging steps, we have some clusters



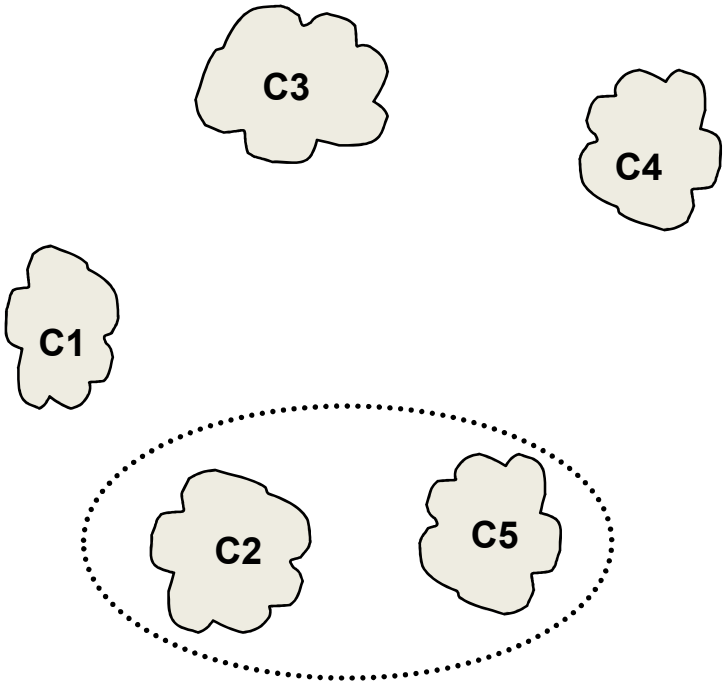
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix



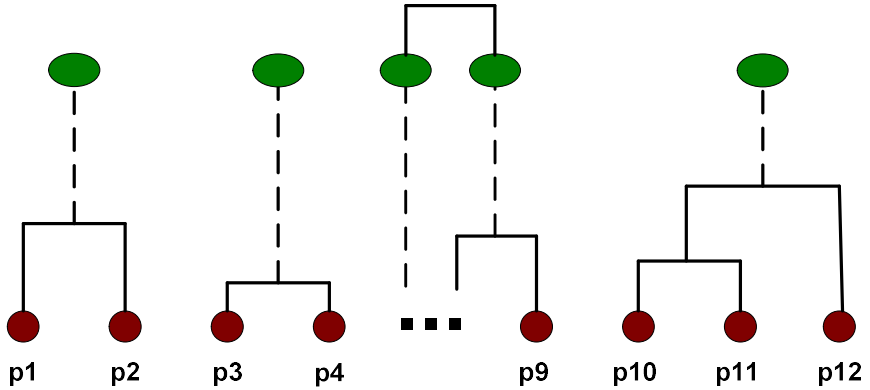
Intermediate State

- Merge the two closest clusters (C2 and C5) and update the distance matrix.



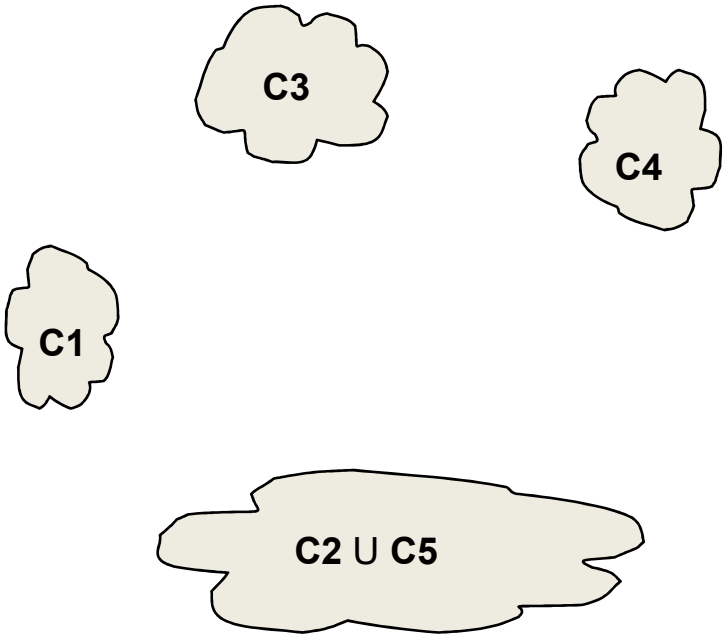
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance/Proximity Matrix

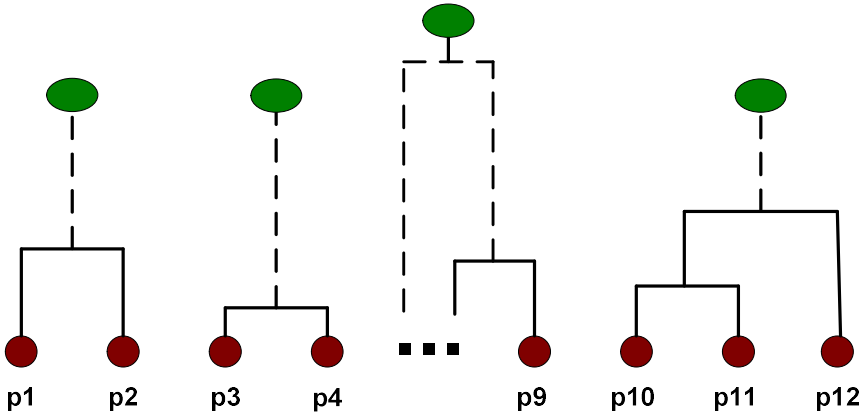


After Merging

- "How do we update the distance matrix?"



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		



Distance between two clusters

- Each cluster is a set of points

- How do we define distance between two sets of points
  - Lots of alternatives
  - Not an easy task

Distance between two clusters

- **Single-link distance** between clusters C_i and C_j is the ***minimum distance*** between any object in C_i and any object in C_j

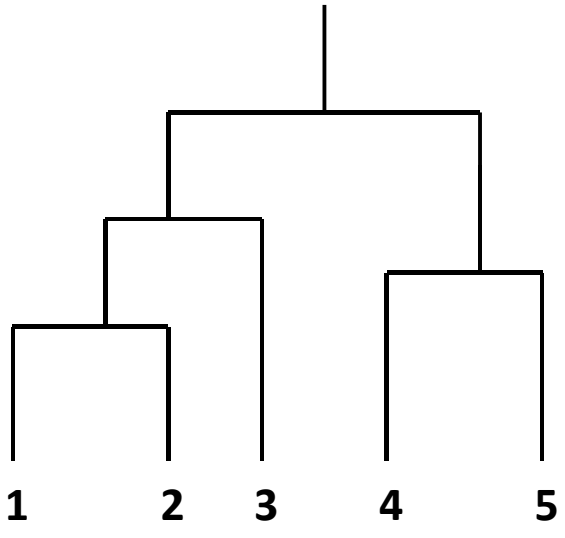
- The distance is **defined by the two most similar objects**

$$D_{sl}(C_i, C_j) = \min_{x,y} \left\{ d(x, y) \middle| x \in C_i, y \in C_j \right\}$$

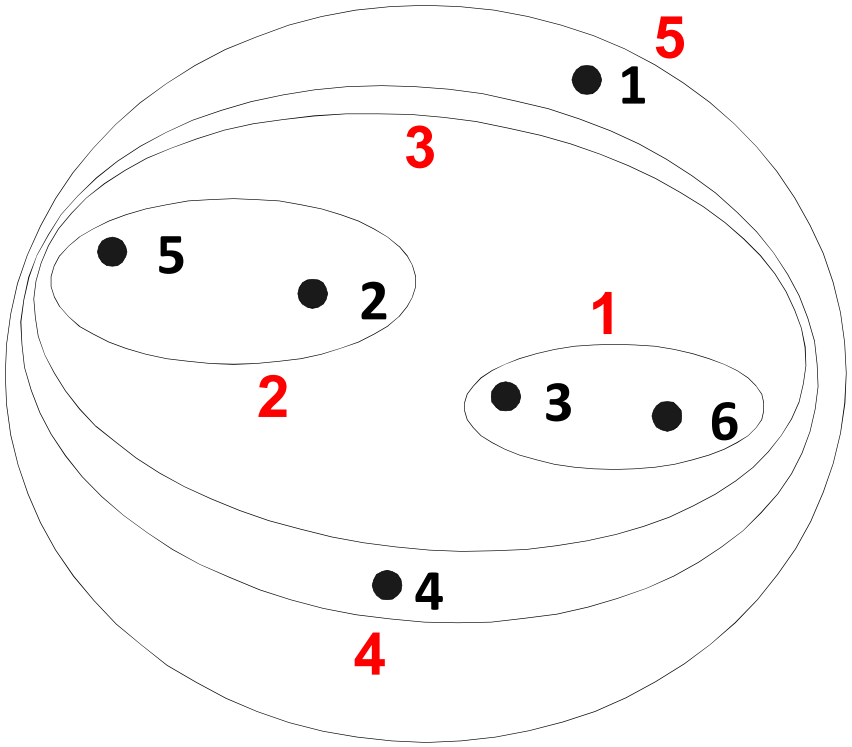
Single-link clustering: example

- Determined by one pair of points, i.e., by one link in the proximity graph.

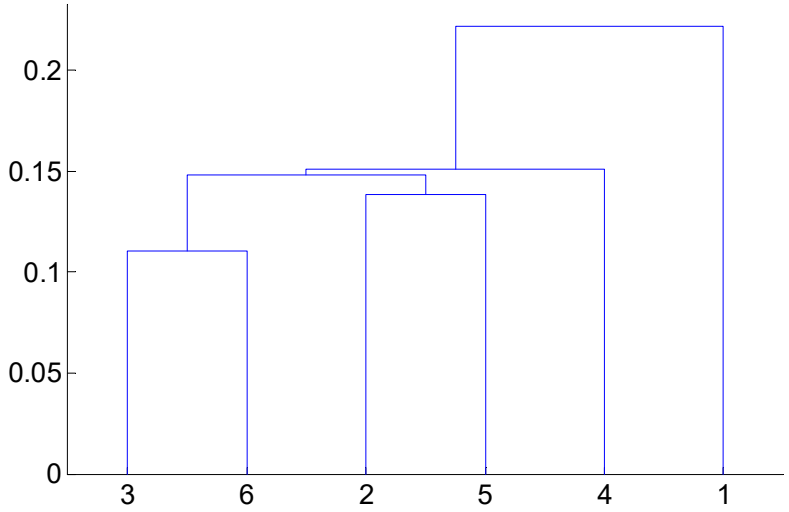
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Single-link clustering: example

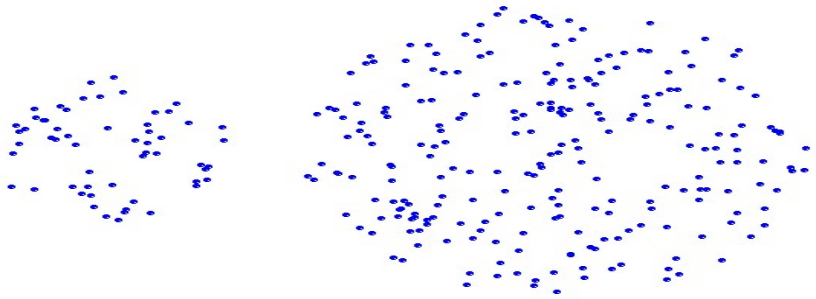


Nested Clusters

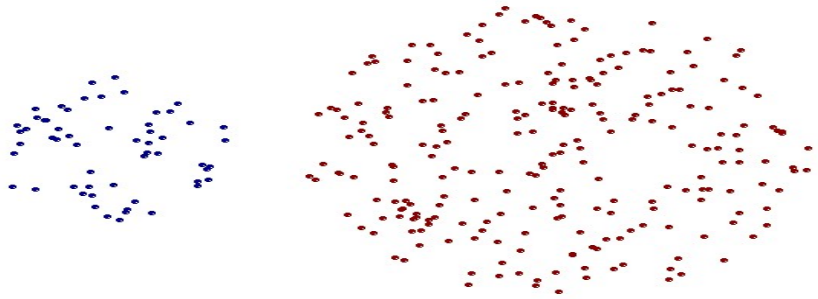


Dendrogram

Strengths of single-link clustering



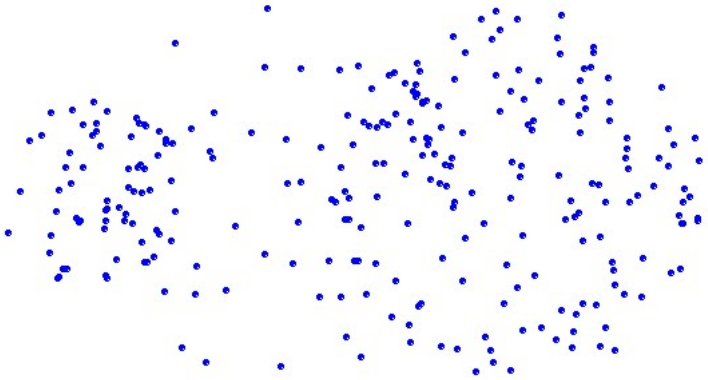
Original Points



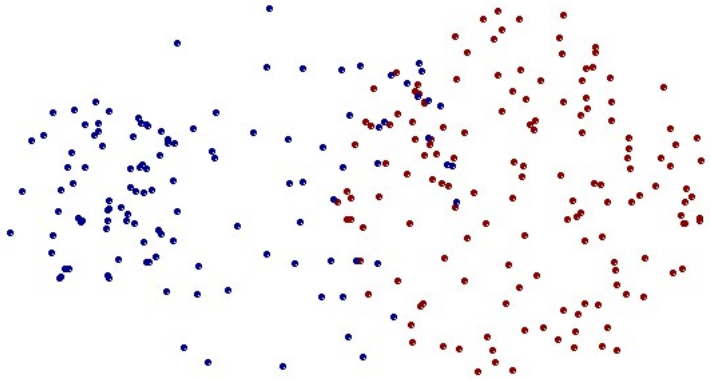
Two Clusters

- Can handle non-elliptical shapes

Limitations of single-link clustering



Original Points



Two Clusters

- Sensitive to noise and outliers
- It produces long, elongated clusters

Distance between two clusters

- **Complete-link distance** between clusters C_i and C_j is the ***maximum distance*** between any object in C_i and any object in C_j

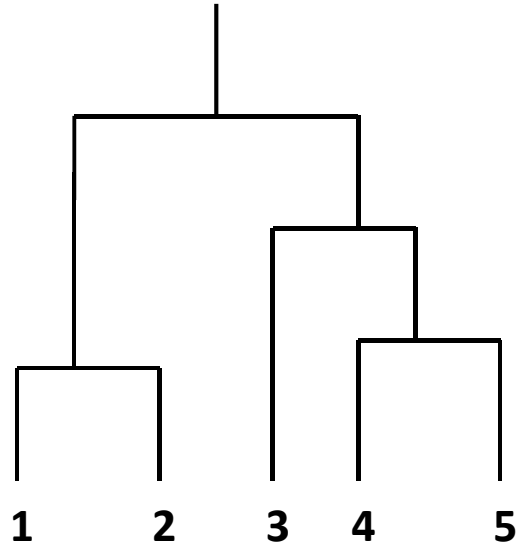
- The distance is **defined by the two most dissimilar objects**

$$D_{cl}(C_i, C_j) = \max_{x,y} \{ d(x, y) \mid x \in C_i, y \in C_j \}$$

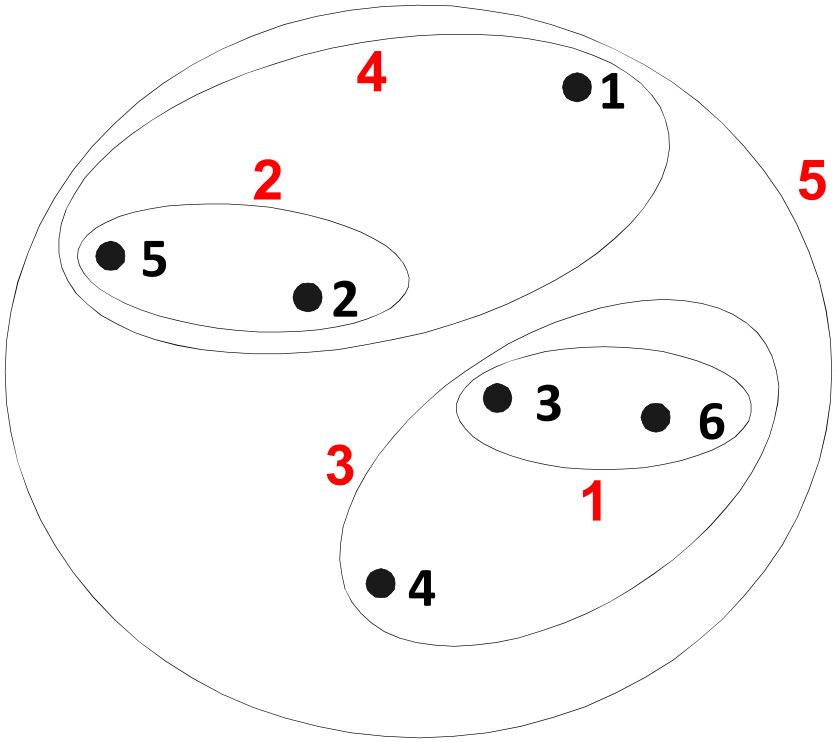
Complete-link clustering: example

- Distance between clusters is determined by the two most distant points in the different clusters

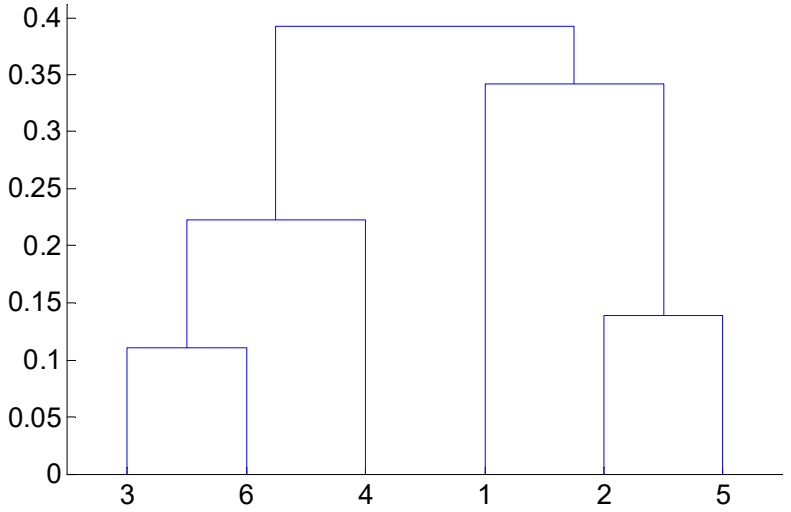
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Complete-link clustering: example

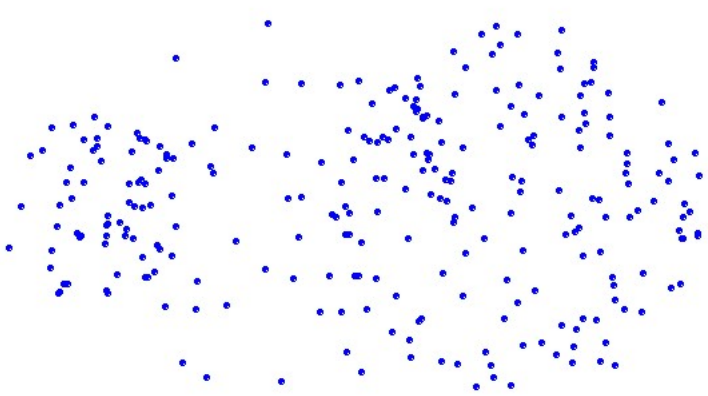


Nested Clusters

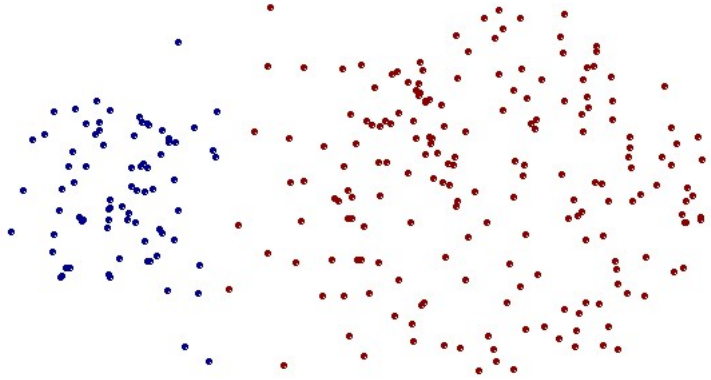


Dendrogram

Strengths of complete-link clustering



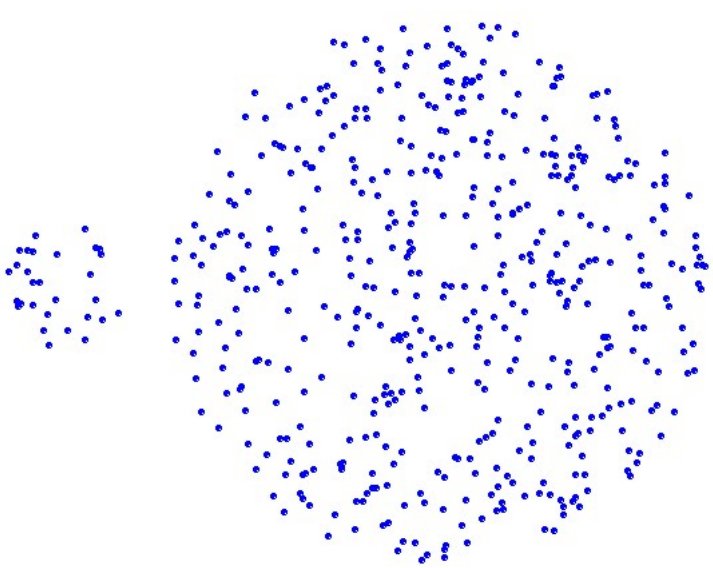
Original Points



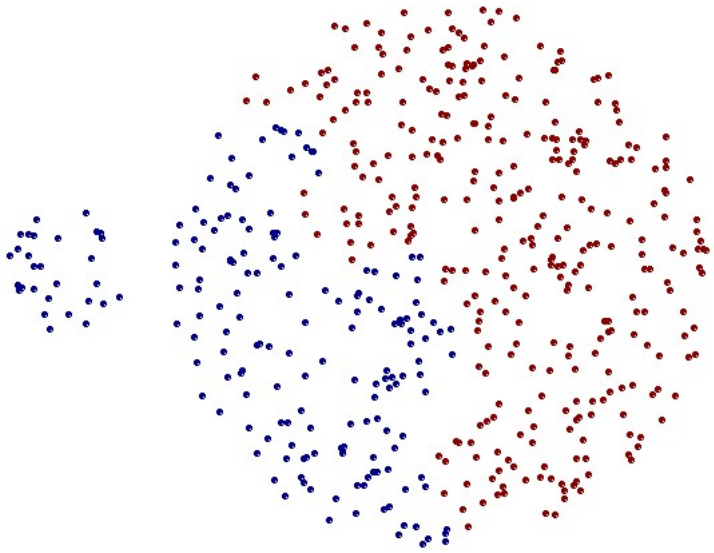
Two Clusters

- More balanced clusters (with equal diameter)
- Less susceptible to noise

Limitations of complete-link clustering



Original Points



Two Clusters

- Tends to break large clusters
- All clusters tend to have the same diameter – small clusters are merged with larger ones

Distance between two clusters

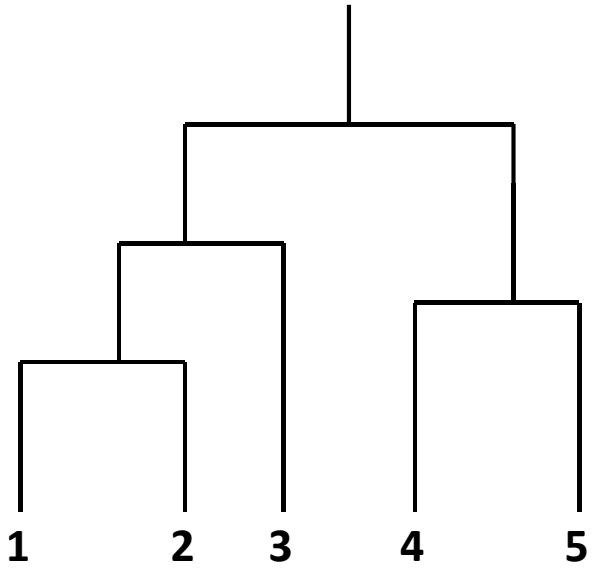
- **Group average distance** between clusters C_i and C_j is the ***average distance*** between any object in C_i and any object in C_j

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

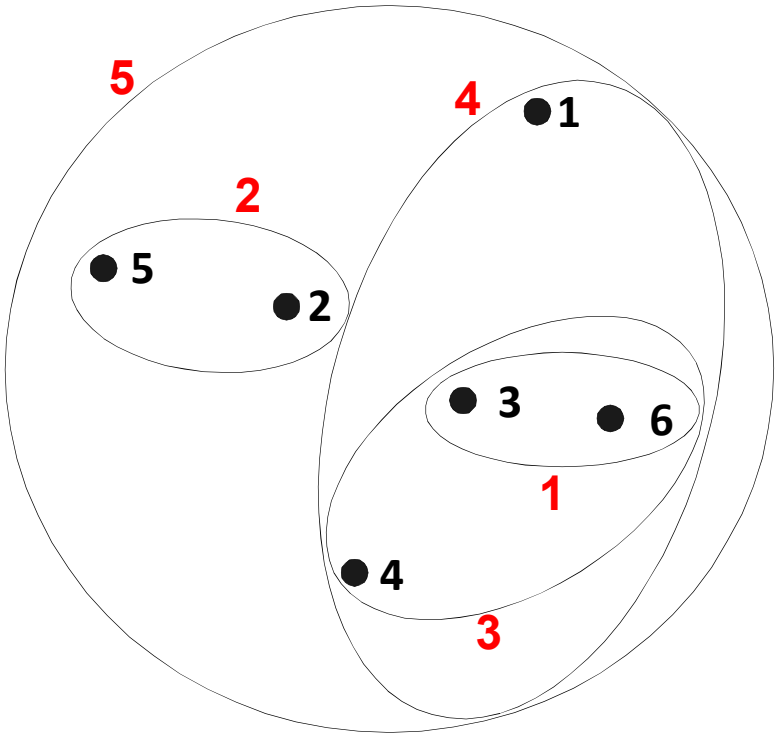
Average-link clustering: example

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

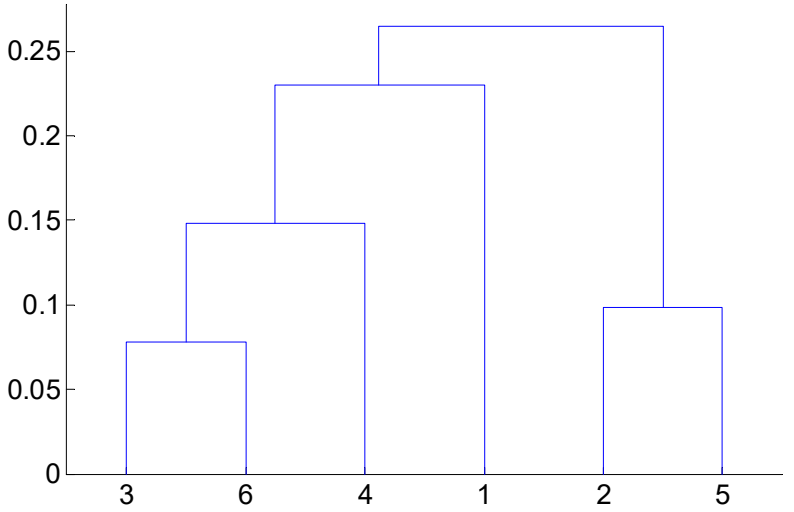
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Average-link clustering: example



Nested Clusters



Dendrogram

Average-link clustering: discussion

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Distance between two clusters

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{centroids}(C_i, C_j) = d(r_i, r_j)$$

Distance between two clusters

- **Ward's distance** between clusters C_i and C_j is the ***difference*** between the ***total within cluster sum of squares for the two clusters separately***, and the ***within cluster sum of squares resulting from merging the two clusters*** in cluster C_{ij}

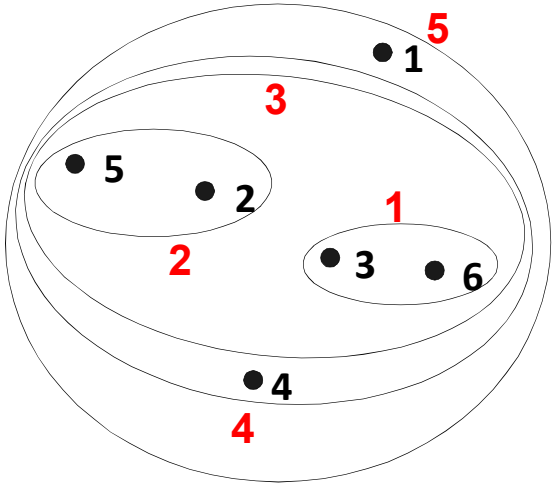
$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i: centroid of C_i
- r_j: centroid of C_j
- r_{ij}: centroid of C_{ij}

Ward's distance for clusters

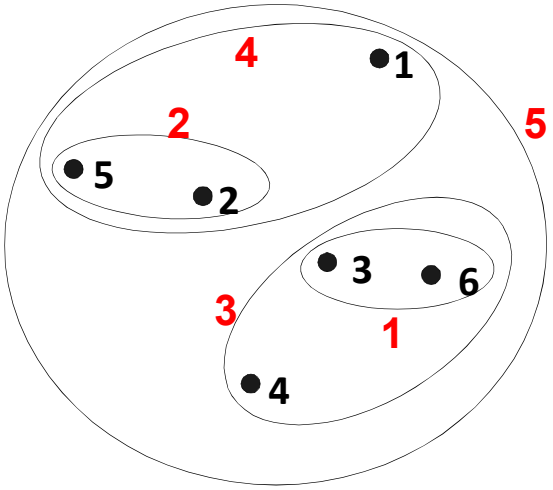
- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

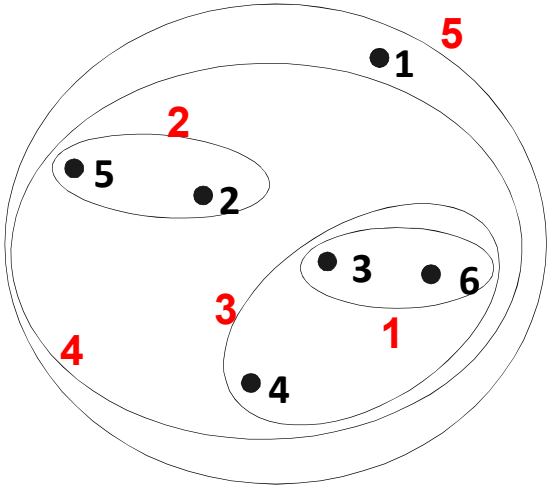
Hierarchical Clustering: Comparison



MIN

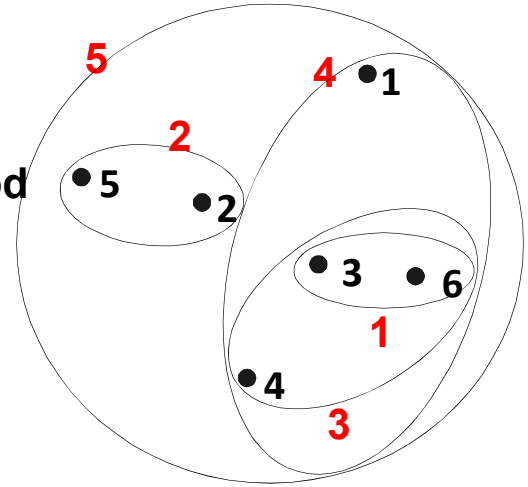
MAX





Group Average

Ward's Method



Hierarchical Clustering: Time and Space requirements

- For a dataset **X** consisting of **n** points
- **$O(n^2)$ space**; it requires storing the distance matrix
- **$O(n^3)$ time** in most of the cases
  - There are **n** steps and at each step the size **n^2** distance matrix must be updated and searched
  - Complexity can be reduced to **$O(n^2 \log(n))$** time for some approaches by using appropriate data structures

Divisive hierarchical clustering

- Start with a single cluster composed of all data points
- Split this into components
- Continue recursively
- *Monothetic* divisive methods split clusters using one variable/dimension at a time
- *Polythetic* divisive methods make splits on the basis of all variables together
- Any intercluster distance measure can be used
- Computationally intensive, less widely used than agglomerative methods

Model-based clustering

- Assume data generated from **k** probability distributions
- ***Goal:*** find the distribution parameters
- ***Algorithm:*** Expectation Maximization (EM)
- ***Output:*** Distribution parameters and a **soft** assignment of points to clusters

Model-based clustering

- Assume **k** probability distributions with parameters: $(\theta_1,\ldots, \theta_k)$
- Given data **X**, compute $(\theta_1,\ldots, \theta_k)$ such that $Pr(X|\theta_1,\ldots, \theta_k)$ [likelihood] or $\ln(Pr(X|\theta_1,\ldots, \theta_k))$ [loglikelihood] is maximized.
- Every point $x \epsilon X$ need not be generated by a single distribution but it can be generated by multiple distributions with some probability [soft clustering]

EM Algorithm

- Initialize **k** distribution parameters **(θ_1,…, θ_k)**; Each distribution parameter corresponds to a cluster center
- Iterate between two steps
  - **E**xpectation step: (probabilistically) assign points to clusters
  - **M**aximation step: estimate model parameters that maximize the likelihood for the given assignment of points

EM Algorithm

- Initialize **k** cluster centers
- Iterate between two steps
  - **E**xpectation step: assign points to clusters

$$\Pr(x_i \in C_k) = \Pr(x_i \mid C_k) \Big/ \sum_j \Pr(x_i \mid C_j)$$

$$w_k = \frac{\sum_i \Pr(x_i \in C_k)}{n}$$

  - **M**aximation step: estimate model parameters

$$r_k = \frac{1}{n} \sum_{i=1}^{n} \frac{\Pr(x_i \in C_k)}{\sum_k \Pr(x_i \in C_j)}$$

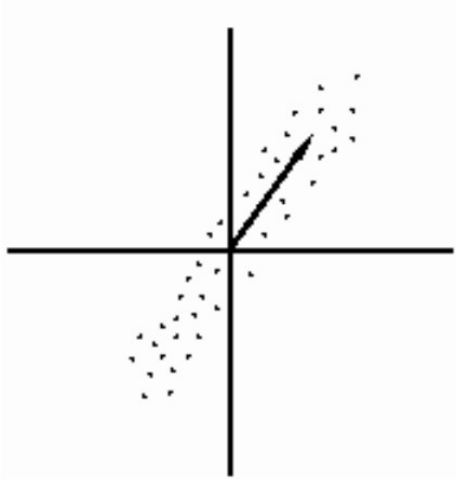
内在结构

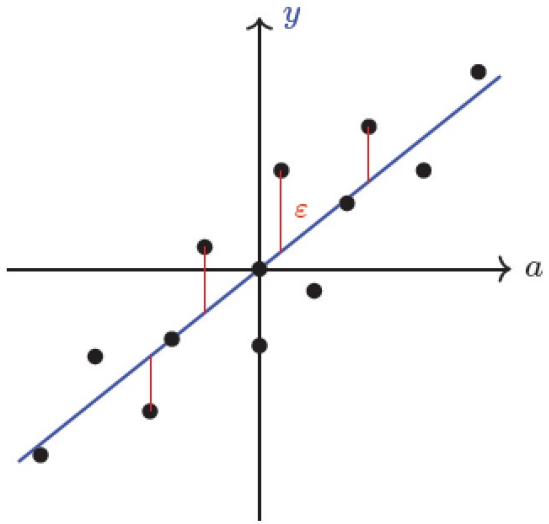
- 哪些数据是一组的（相似的）？

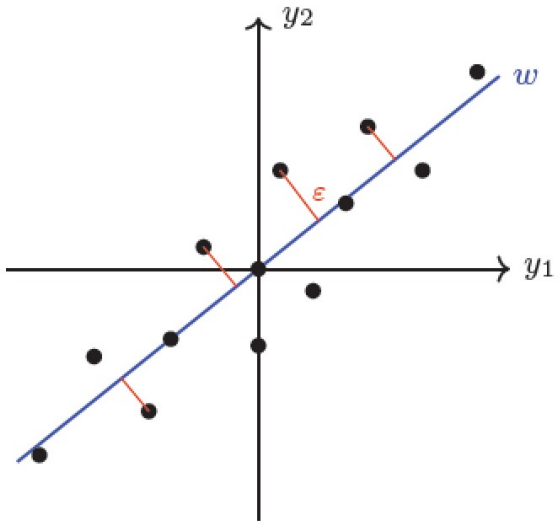
- 相似性的“度量”？
  - 距离
  - 线性空间（同一组基表达）
  - 非线性空间（流形空间）

Principal Component Analysis (PCA)

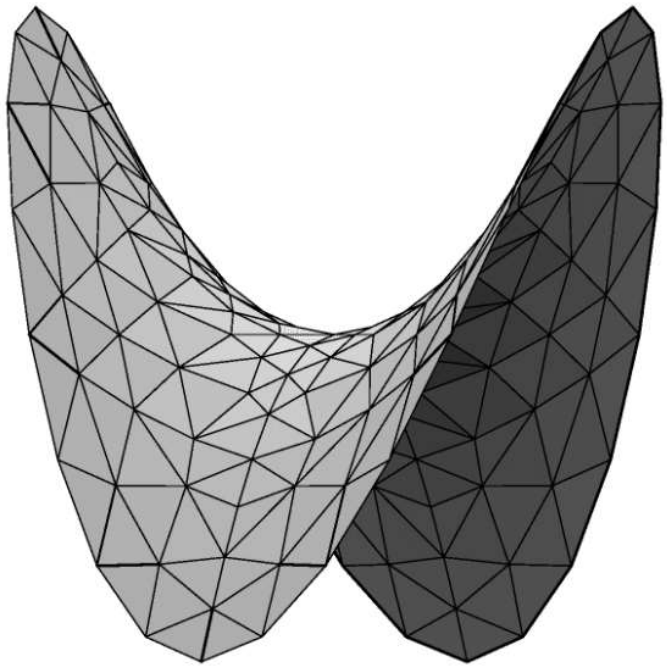
- Find linear subspace projection which preserves the data locations (under quadratic error)





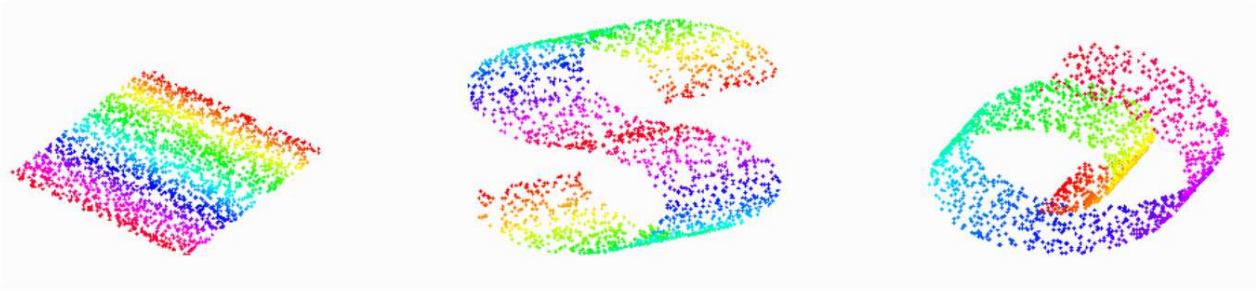


Manifolds



low-D surface
embedded in
high-D space

Manifolds

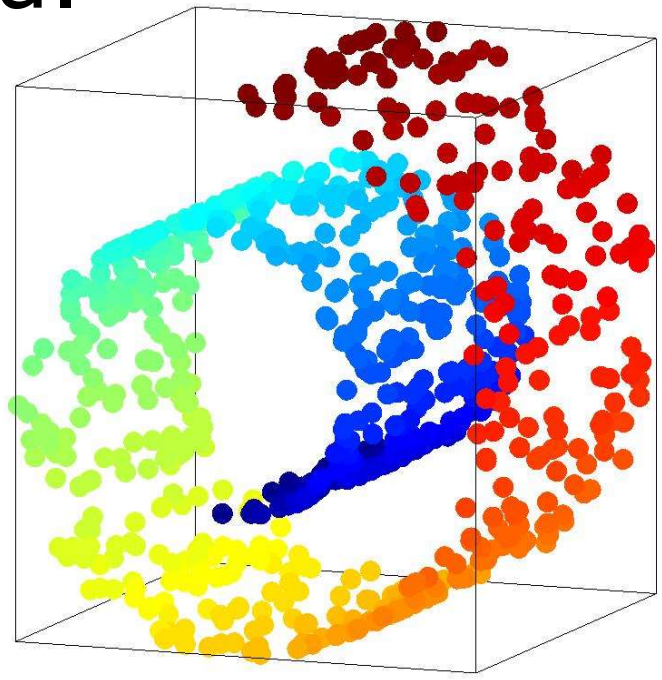


(a) (b) (c)

- PCA : works for (a)
- Doesn’t do much good for (b) or (c)
  - Linear subspace doesn’t explain it well
- What do we mean by “consistent locations”?
  - Preserve local relationships and structure
  - One possibility : preserve distances

Manifold learning

Find a low-D basis for describing high-D data.



X ~= X'
s.t.
dim(X') << dim(X)

uncovers the intrinsic dimensionality

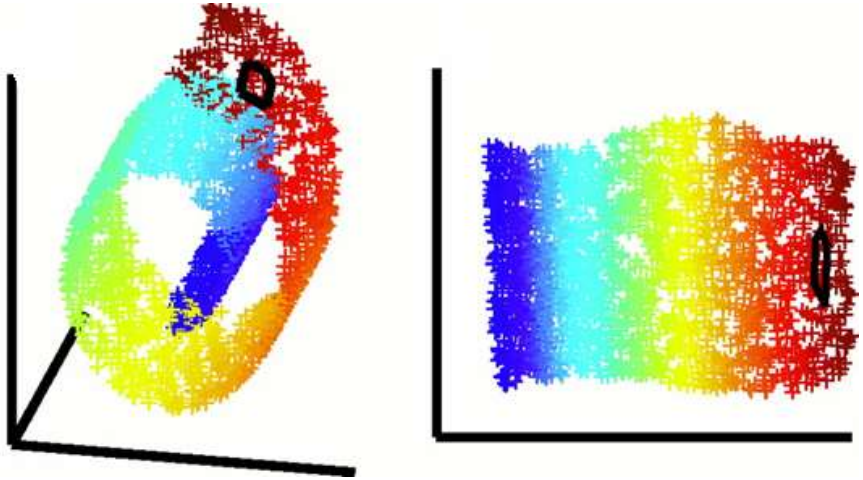
Dimension Reduction

why do manifold learning?

1. data compression
2. “curse of dimensionality”
3. de-noising
4. visualization
5. reasonable distance metrics

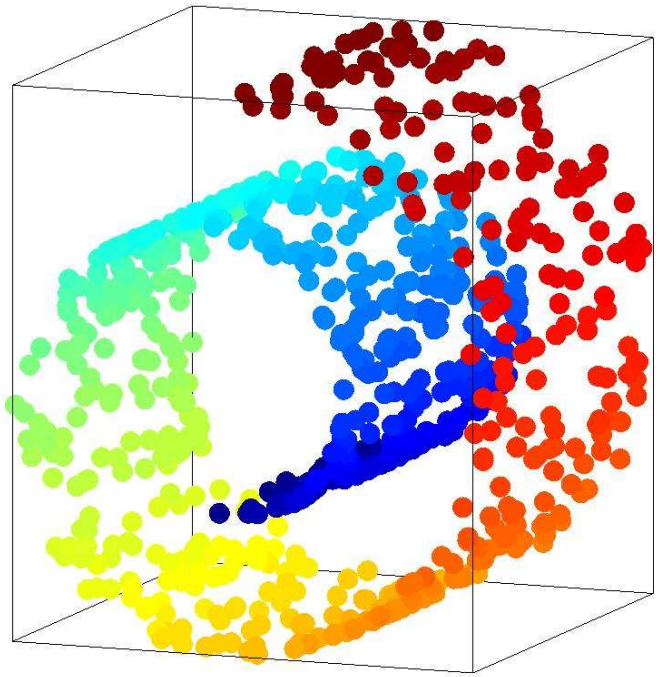
Manifold Learning & Dimension Reduction

- Given a set of data points $x_1, \cdots x_N \in \Re^m$
- $$x_i = f(\tau_i) + \varepsilon_i, i = 1, \cdots N$$
- Dimension Reduction: Find τ_i
- Manifold Learning: Find f



Key Difficulty

- data points unorganized (no adjacency relationship known)



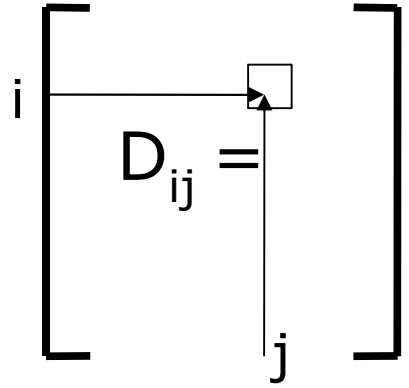
If we knew all pairwise distances…

	Chicago	Raleigh	Boston	Seattle	S.F.	Austin	Orlando
Chicago	0						
Raleigh	641	0					
Boston	851	608	0				
Seattle	1733	2363	2488	0			
S.F.	1855	2406	2696	684	0		
Austin	972	1167	1691	1764	1495	0	
Orlando	994	520	1105	2565	2458	1015	0

Distances calculated with geobytes.com/CityDistanceTool

Multidimensional Scaling (MDS)

For *n* data points, and a distance matrix D,

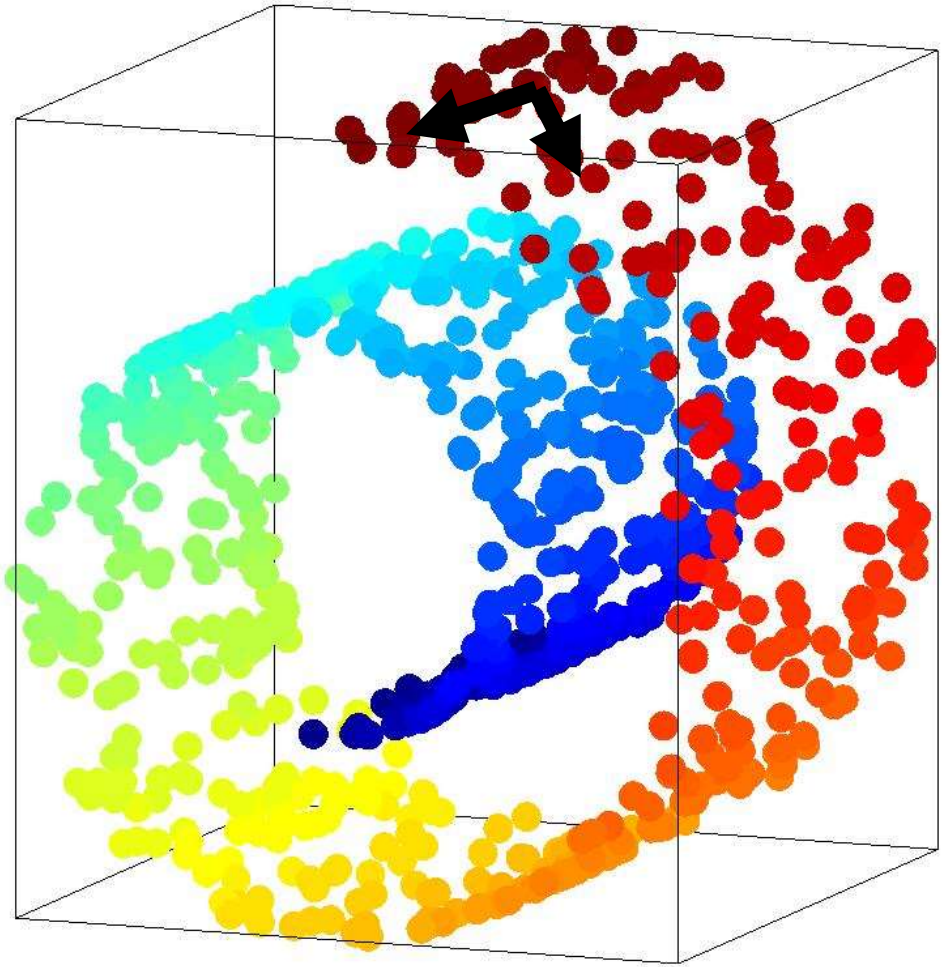


...we can construct a *m*-dimensional space to preserve inter-point distances by using the top eigenvectors of D scaled by their eigenvalues

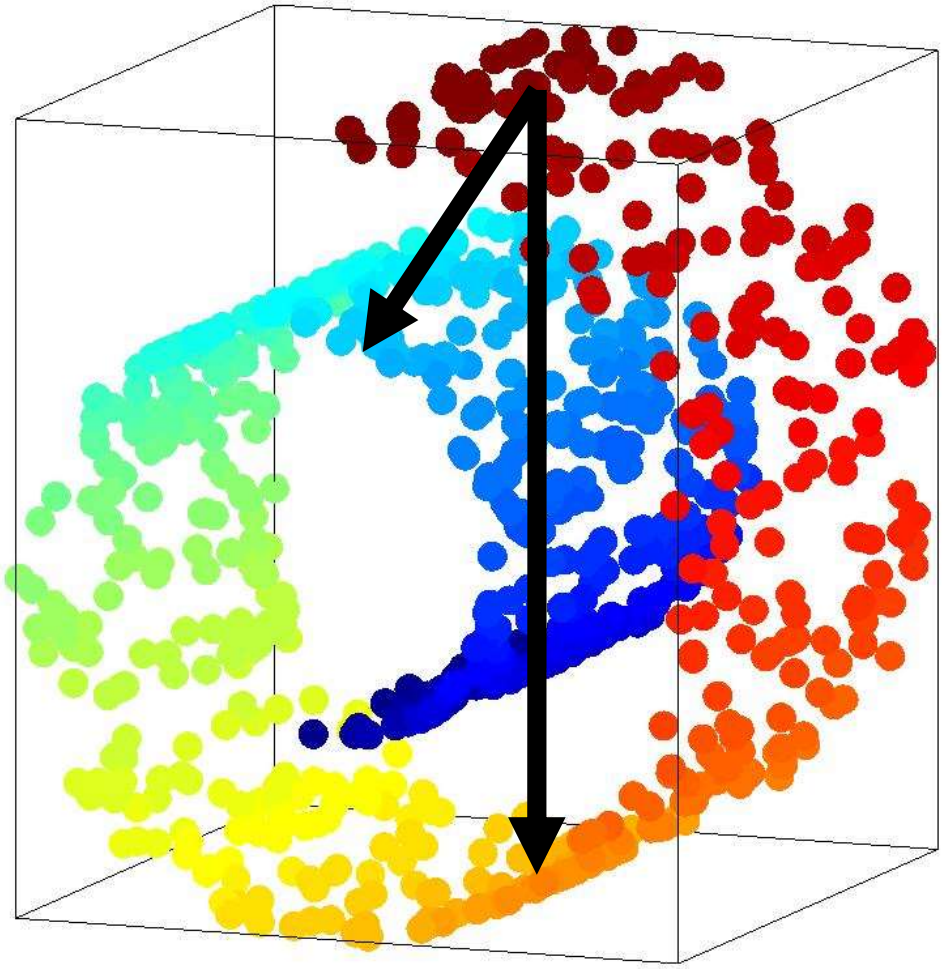
MDS result in 2D

Actual plot of cities

Don’t know distances



Don’t know distnaces



Taxonomy of Dimension Reduction Techniques

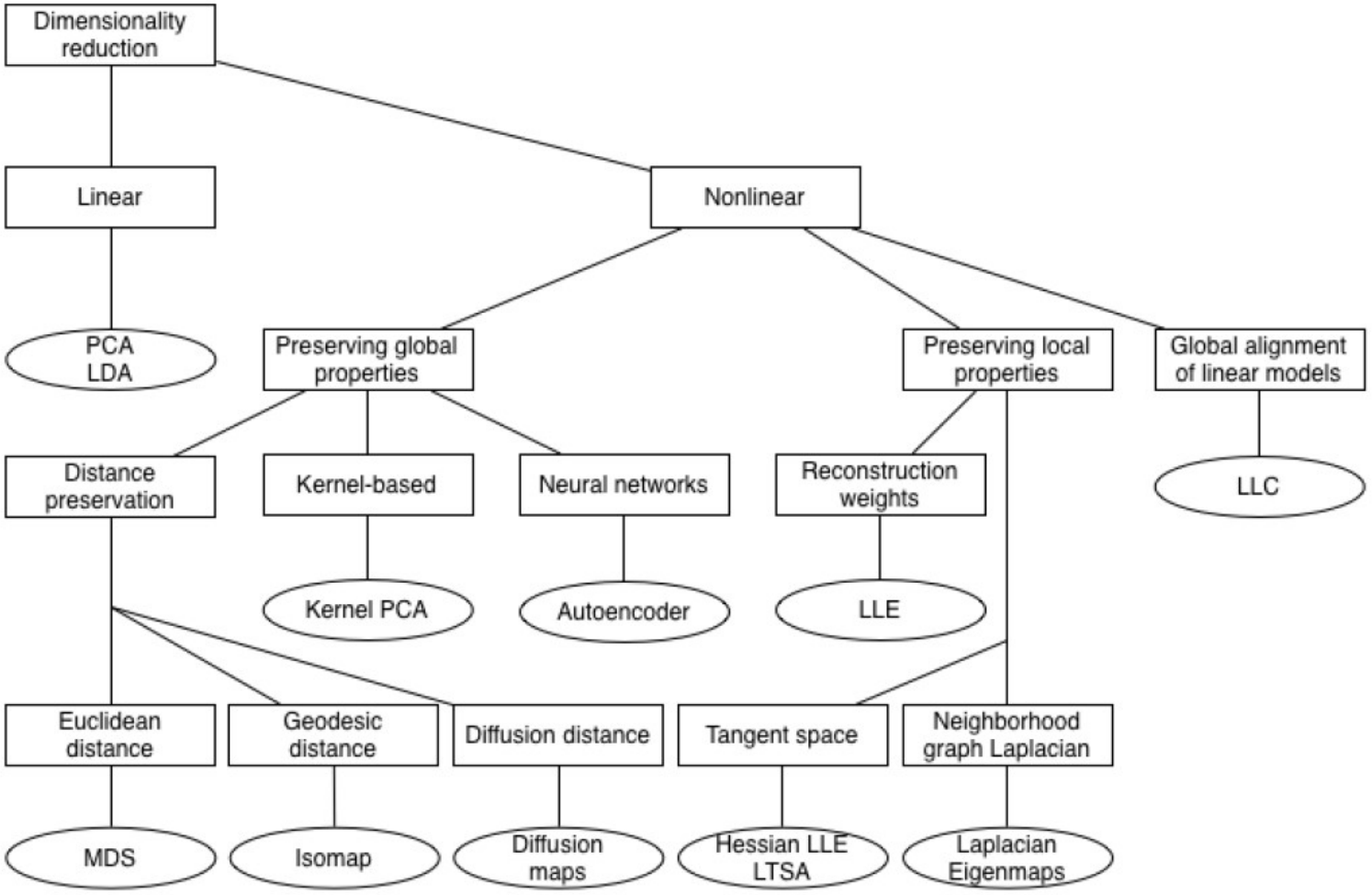


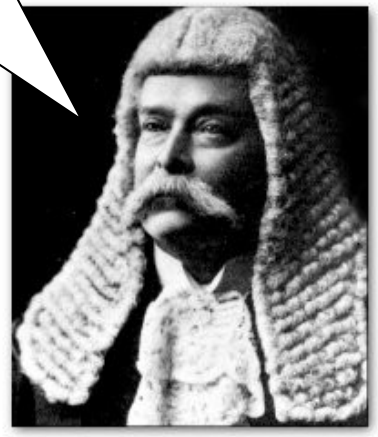
Figure From: L.J.P. van der Maaten et. al., *Dimensionality Reduction: A Comparative Review*

Methods of Manifold Learning

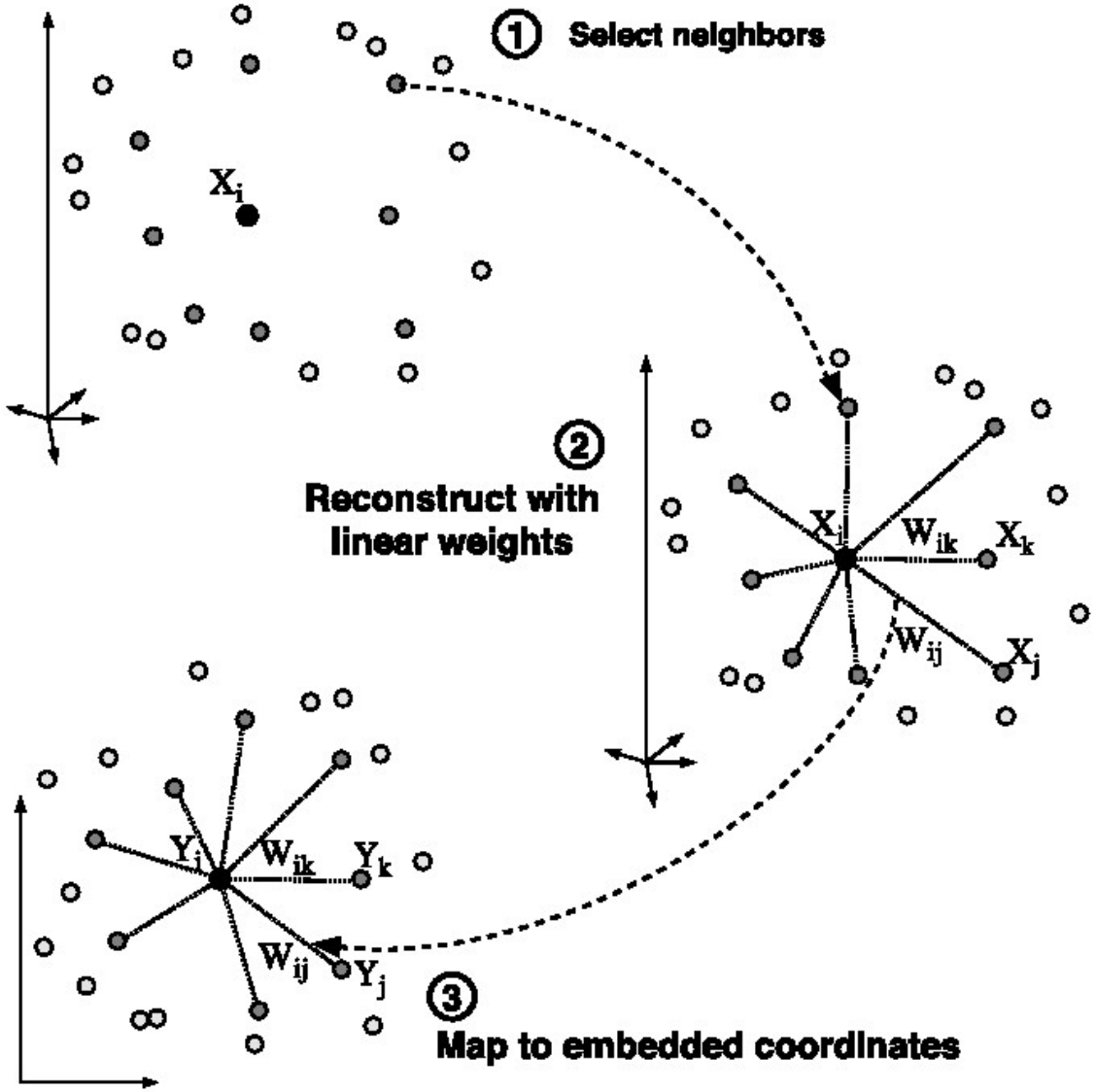
1. Locally Linear Embedding (LLE)

[Science, 2000]

Find a mapping to preserve local linear relationships between neighbors



Locally Linear Embedding



LLE: Two key steps

1. Find weight matrix W of linear coefficients:

$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum\nolimits_j W_{ij} \vec{X}_j \right|^2$$

Enforce sum-to-one constraint.

LLE: Two key steps

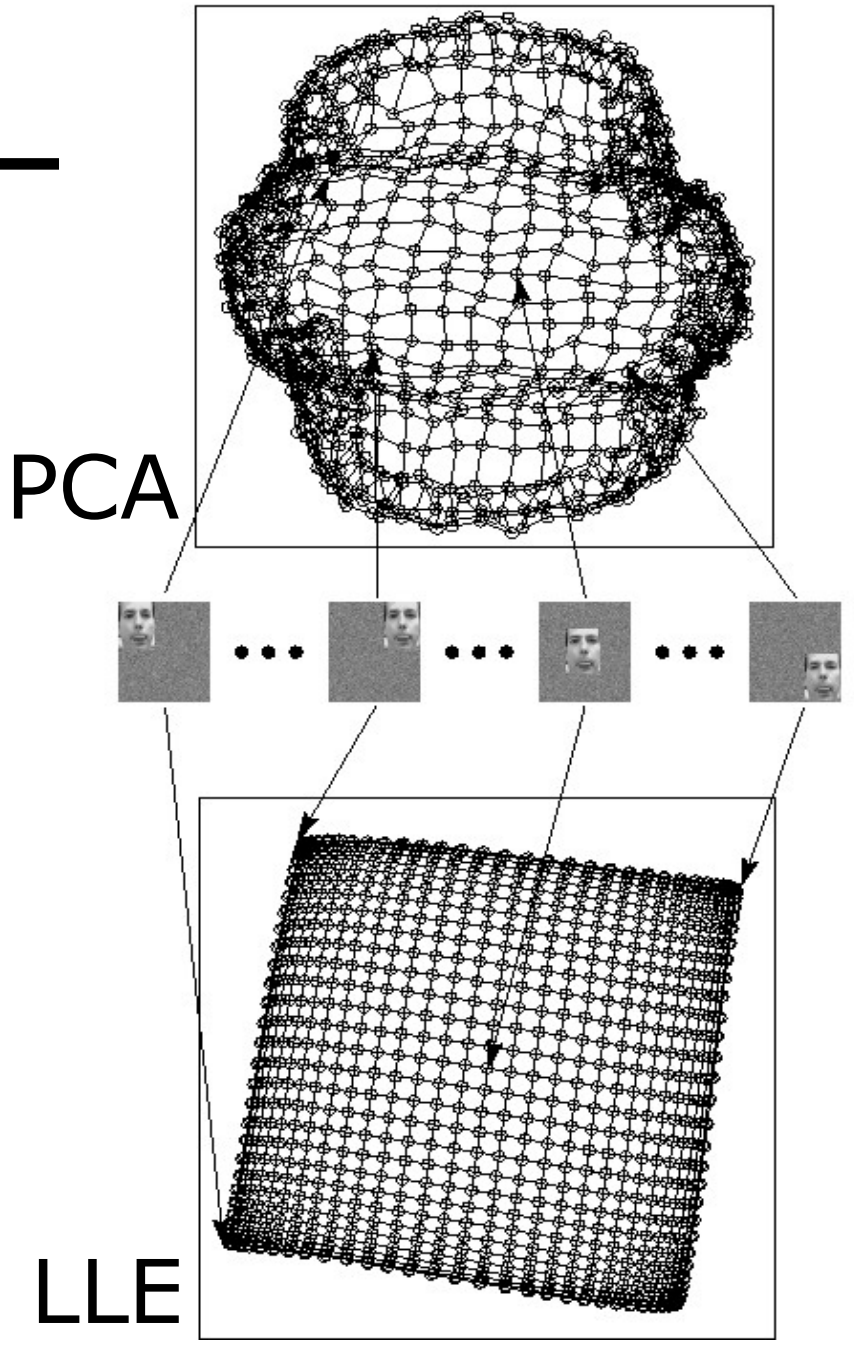
2. Find projected vectors Y to minimize reconstruction error

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum\nolimits_j W_{ij} \vec{Y}_j \right|^2$$

must solve for whole dataset simultaneously

LLE: Result

preserves local topology



LLE: pro and con

- no local minima, one free parameter

- incremental & fast

- simple linear algebra operations

- can distort global structure

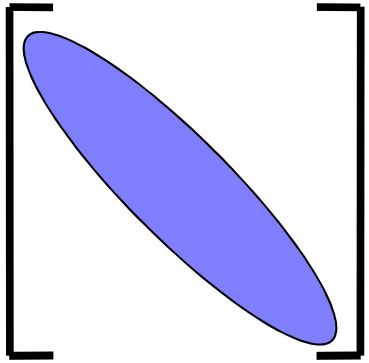
2. Isomap

[Science, 2000]

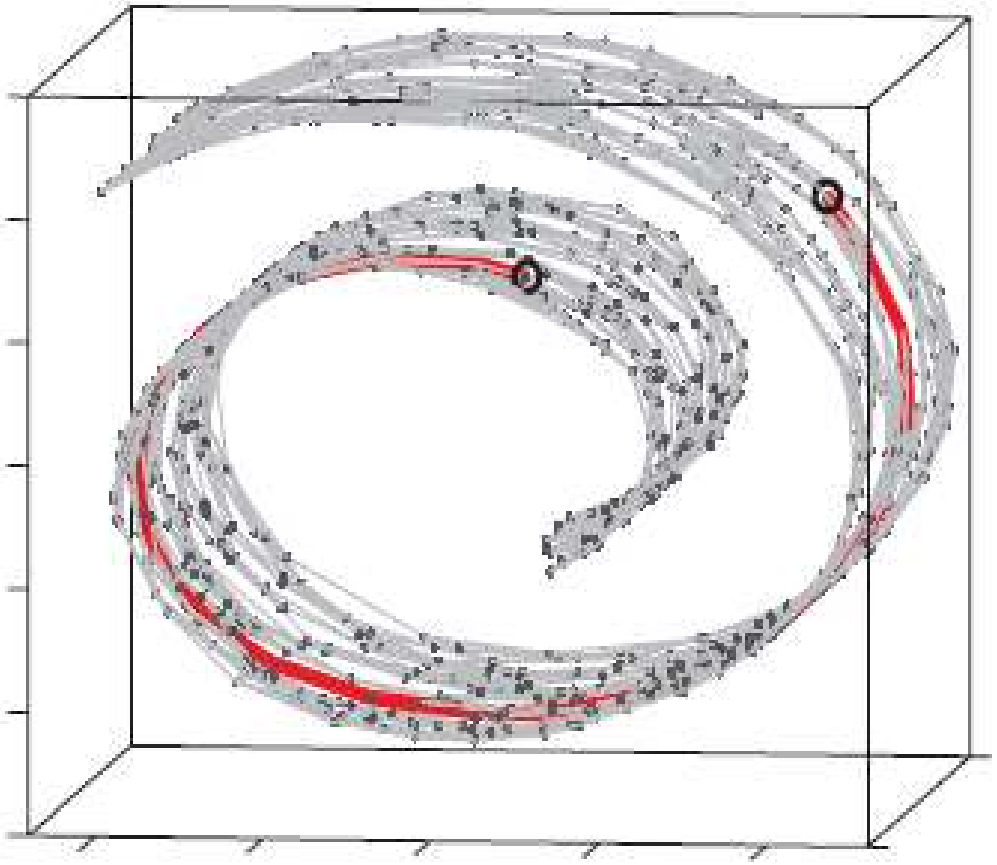
- Build a data graph G.
  - (u,v) is an edge iff SSD(u,v) is small
- For any two points, we approximate the distance between them with the “shortest path” on G

Isomap

1. Build a sparse graph with K-nearest neighbors

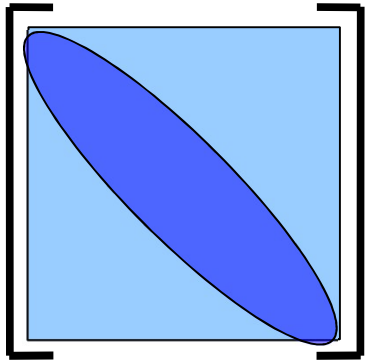
$D_g =$ 

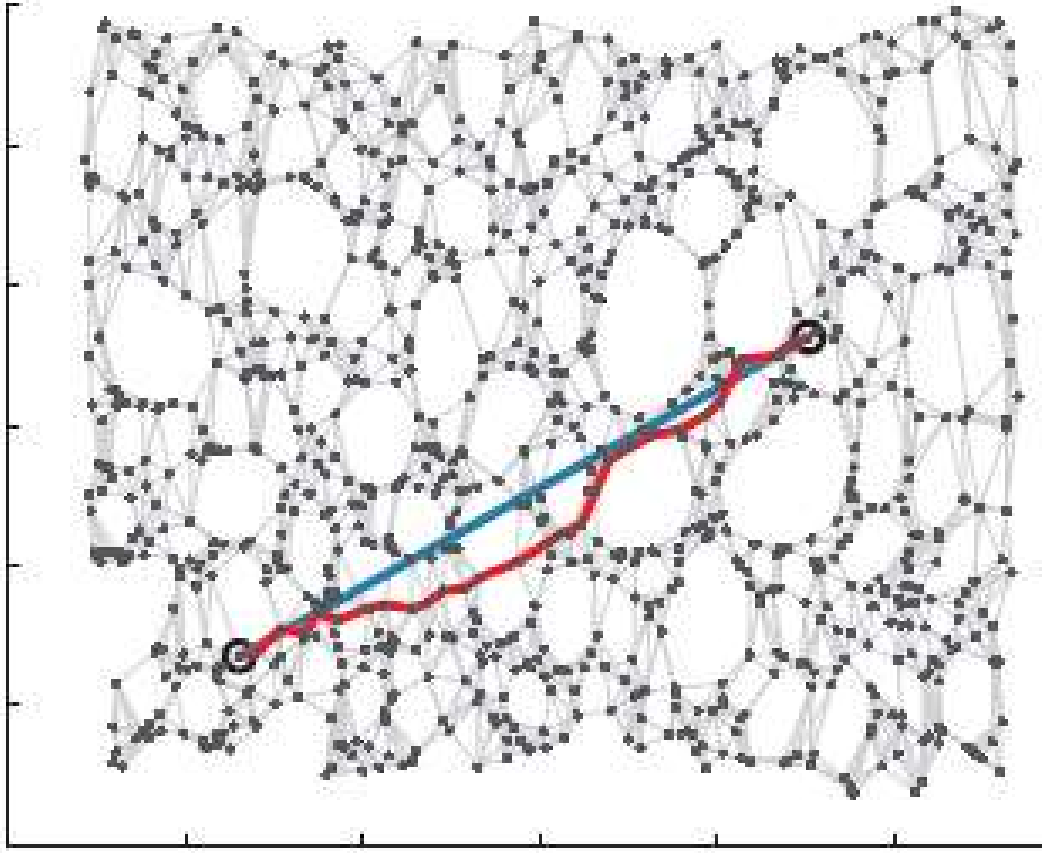
(distance matrix is sparse)



Isomap

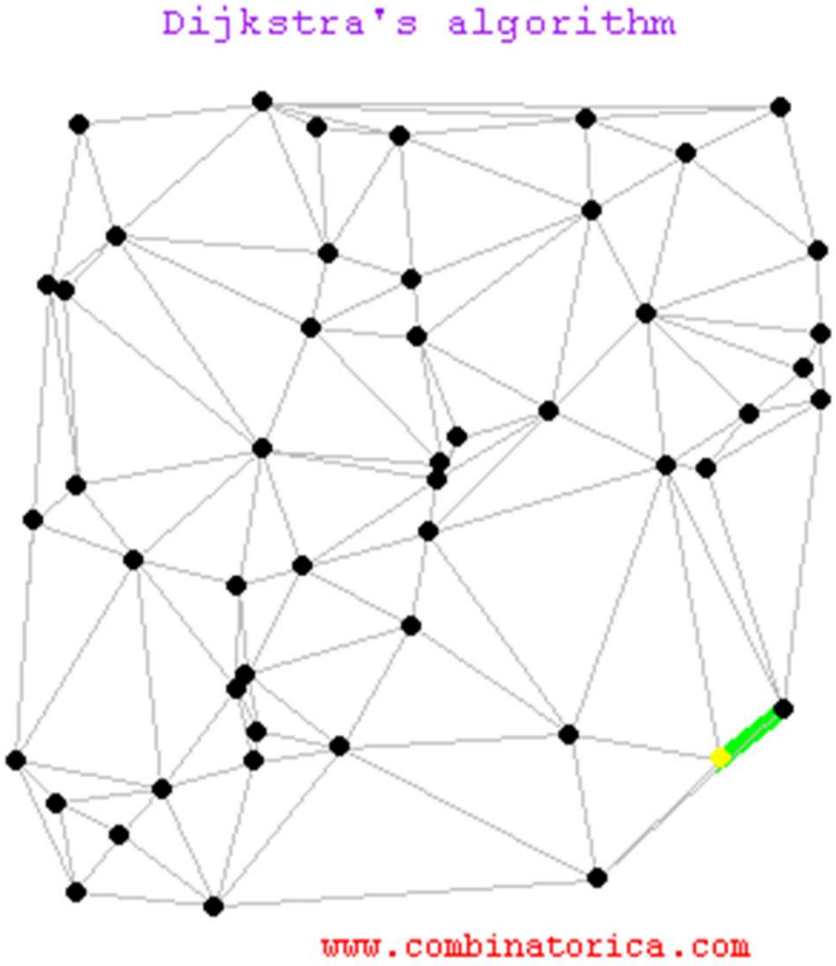
2. Infer other interpoint distances by finding shortest paths on the graph (Dijkstra's algorithm).

$D_g =$ 



Isomap

shortest-distance on a graph is easy to compute



Isomap: pro and con

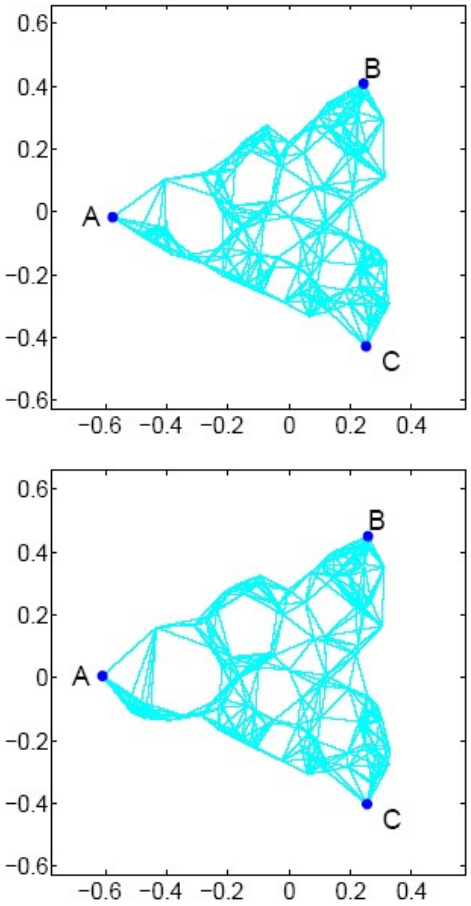
- preserves global structure

- few free parameters

- sensitive to noise, noise edges

- computationally expensive (dense matrix eigen-reduction)

Leakage problem



LLE和Isomap有效的原因

- 它们都是非参数的方法，不需要对流形的很多的参数假设.
- 它们是非线性的方法，都基于流形的内在几何结构，更能体现现实中数据的本质.
- 它们的求解简单，都转化为求解特征值问题，而不需要用迭代算法.

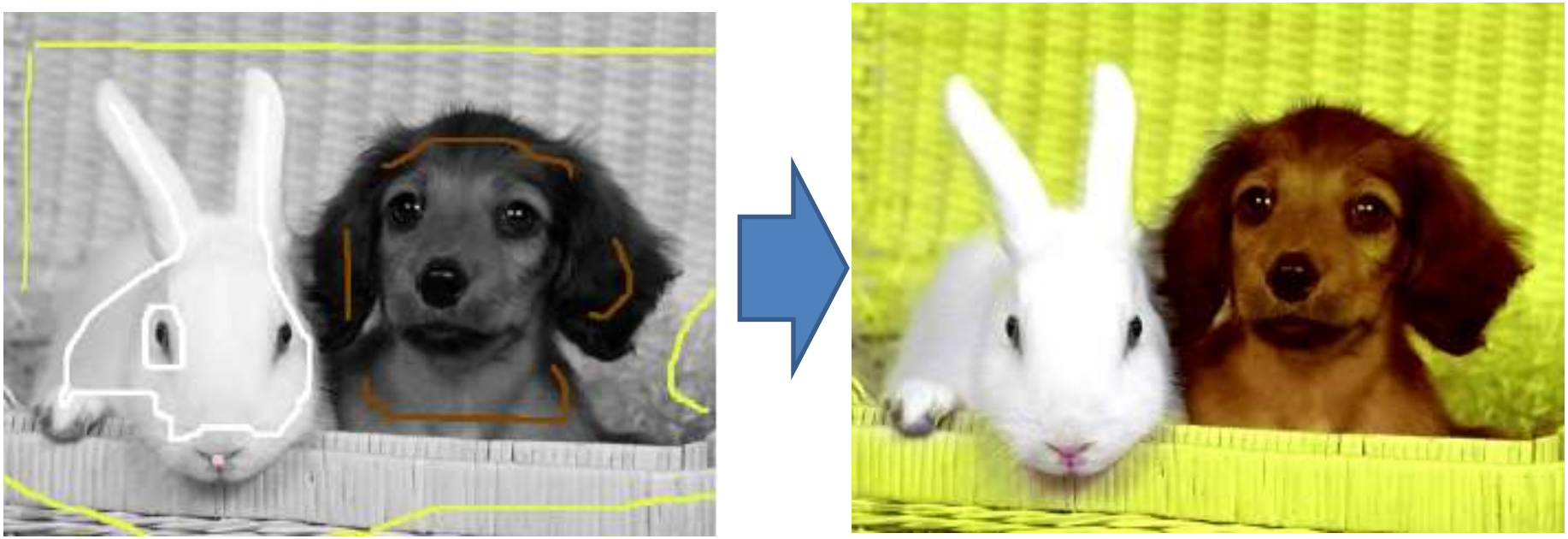
流形学习问题探讨

- 求解方法
  - 对嵌入映射或者低维流形作出某种特定的假设, 或者以保持高维数据的某种性质不变为目标.
  - 将问题转化为求解优化问题.
  - 提供有效的算法.

- 如何确定低维目标空间的维数?
- 当采样数据很稀疏时, 怎样进行有效的学习?
- 将统计学习理论引入流形学习对其泛化性能进行研究

Manifold Preserving Image Editing Propagation

Editing Propagation

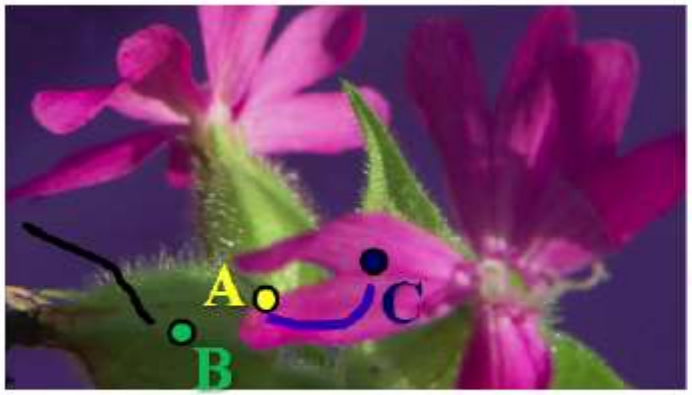


Colorization

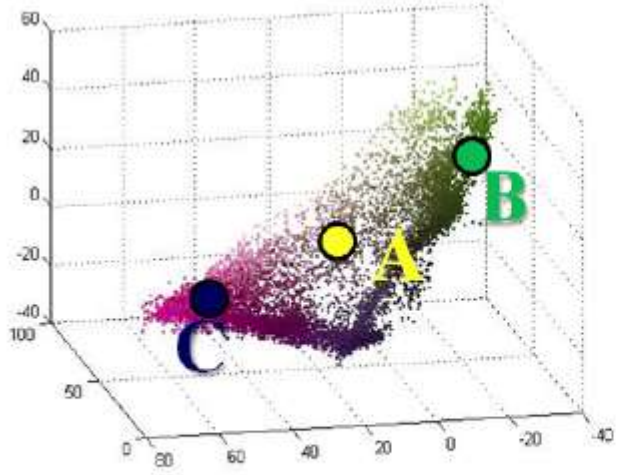
Chen et al. Manifold Preserving Image Editing Propagation. Siggraph Asia 2012.

Manifold Preserving Propagation

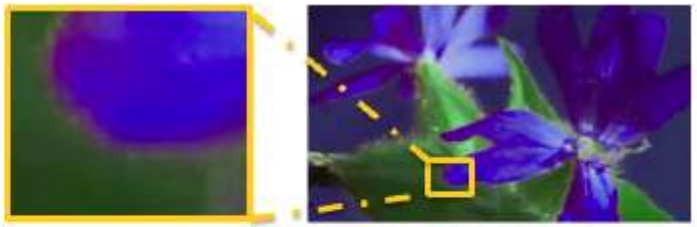
[Siggraph Asia, 2012]



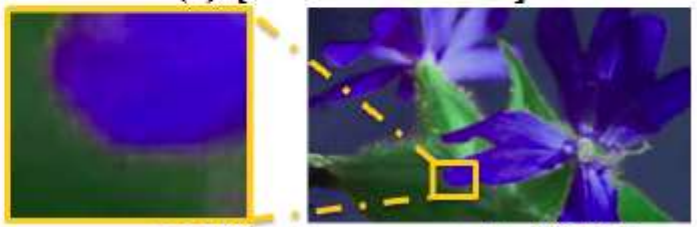
(a)



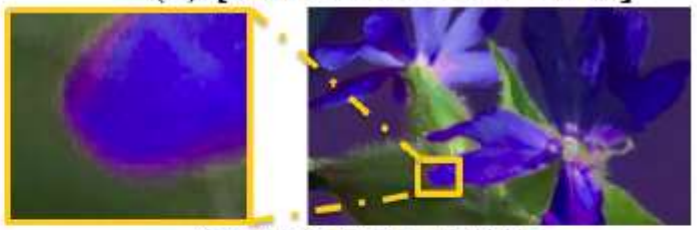
(b)



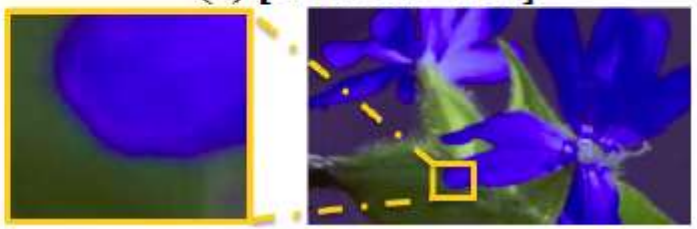
(c) [Xu et al. 2009]



(d) [Farbman et al. 2010]



(e) [Li et al. 2010]



(f) Our result

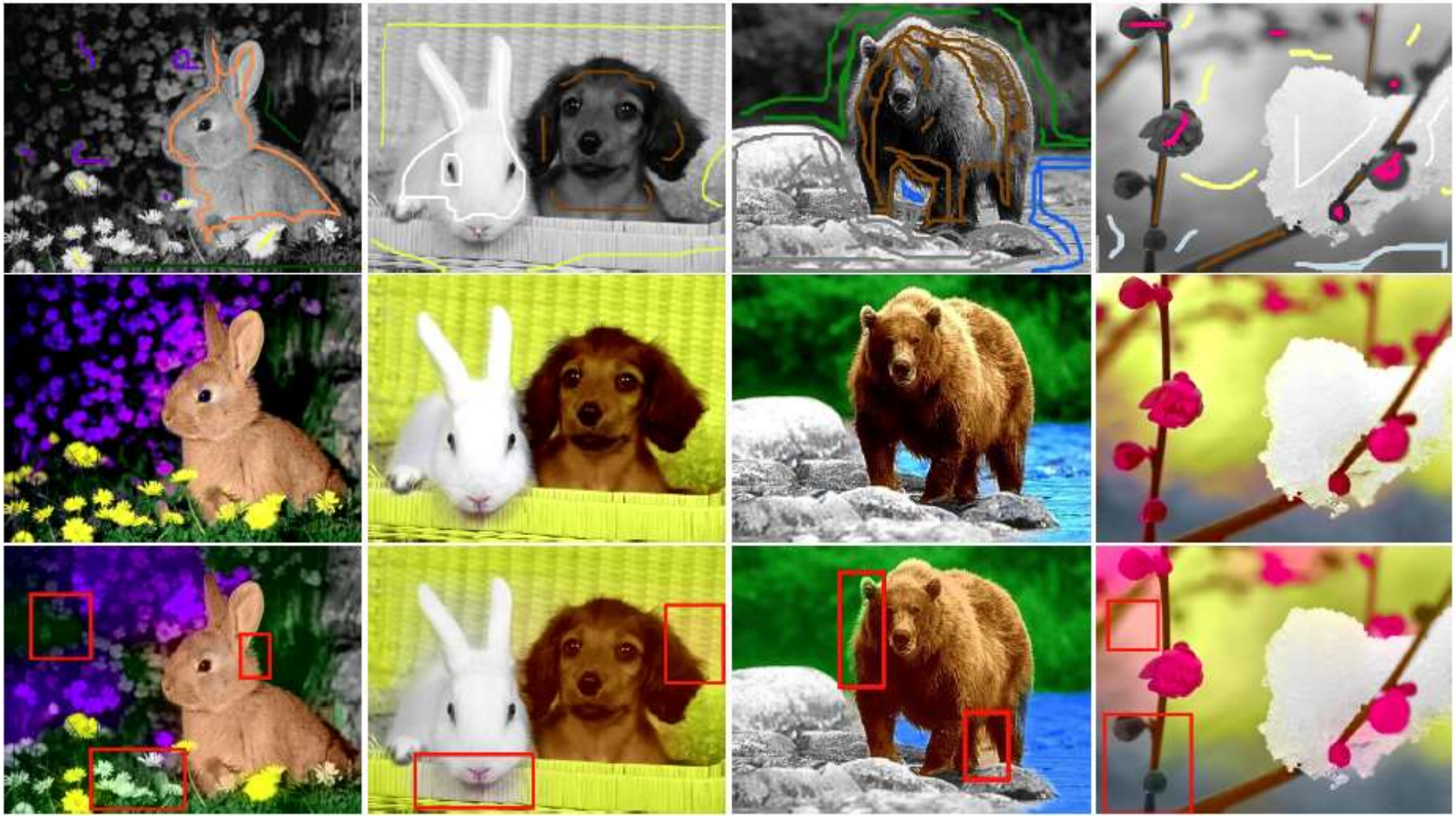
Recoloring

User input

Photoshop

This method

More Results

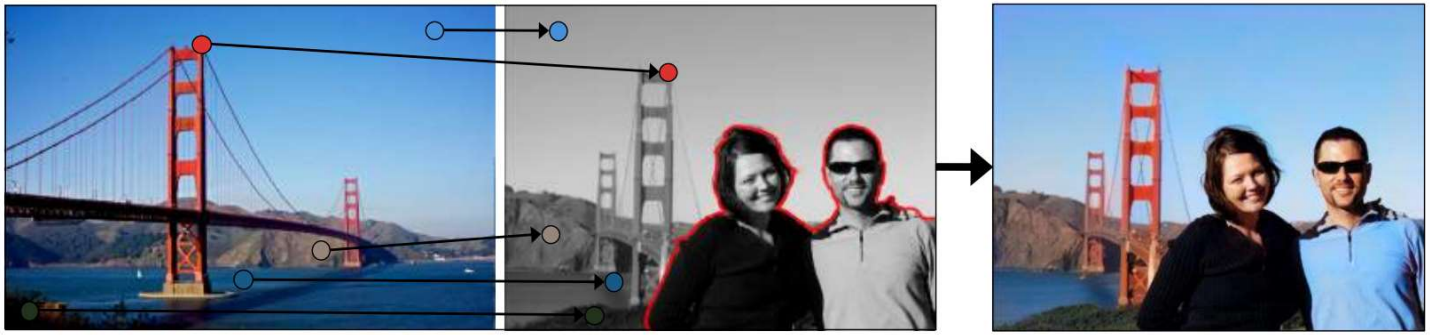


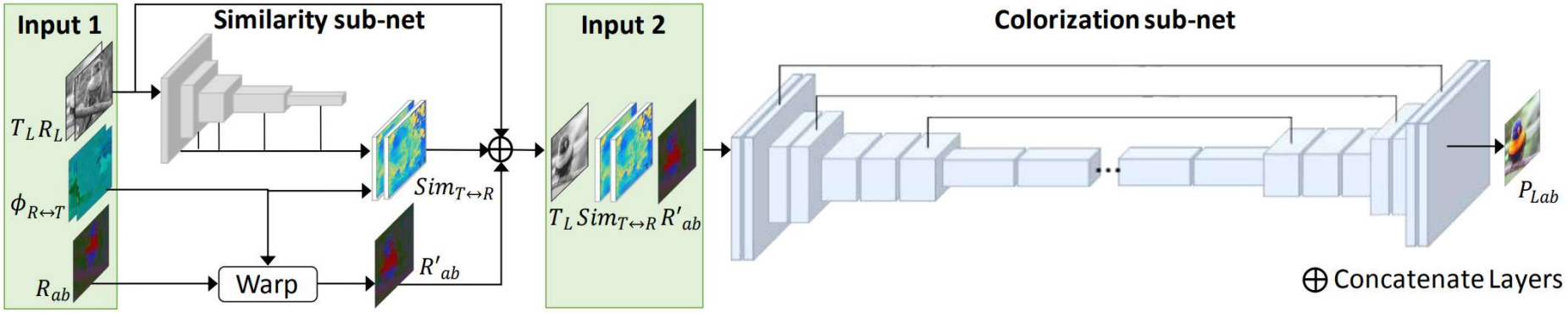
Learning based Methods

Insight: Mapping Composition
数据变换

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} \rightarrow \begin{pmatrix} l \\ \alpha \\ \beta \end{pmatrix} \rightarrow \begin{pmatrix} l' \\ \alpha' \\ \beta' \end{pmatrix} \rightarrow \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix}$$

Deep Exemplar-based Colorization





He et al. Deep Exemplar-based Colorization. Siggraph 2018.
https://liaojing.github.io/html/index.html

Deep Exemplar-based Colorization

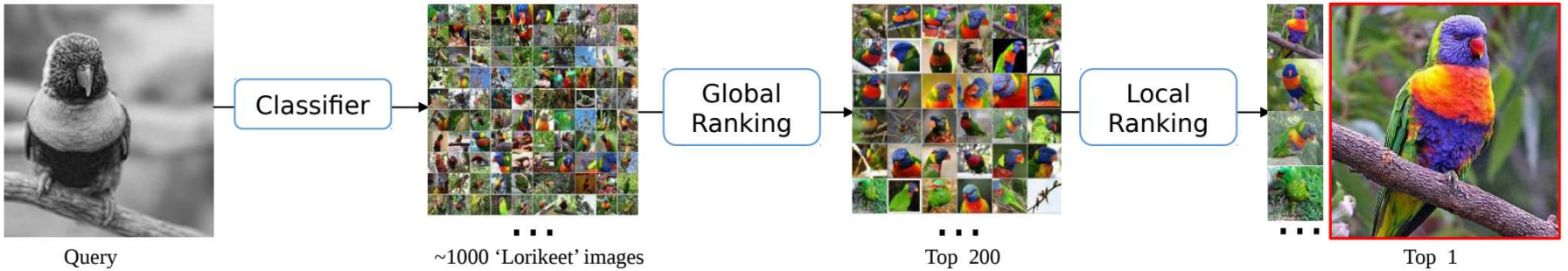
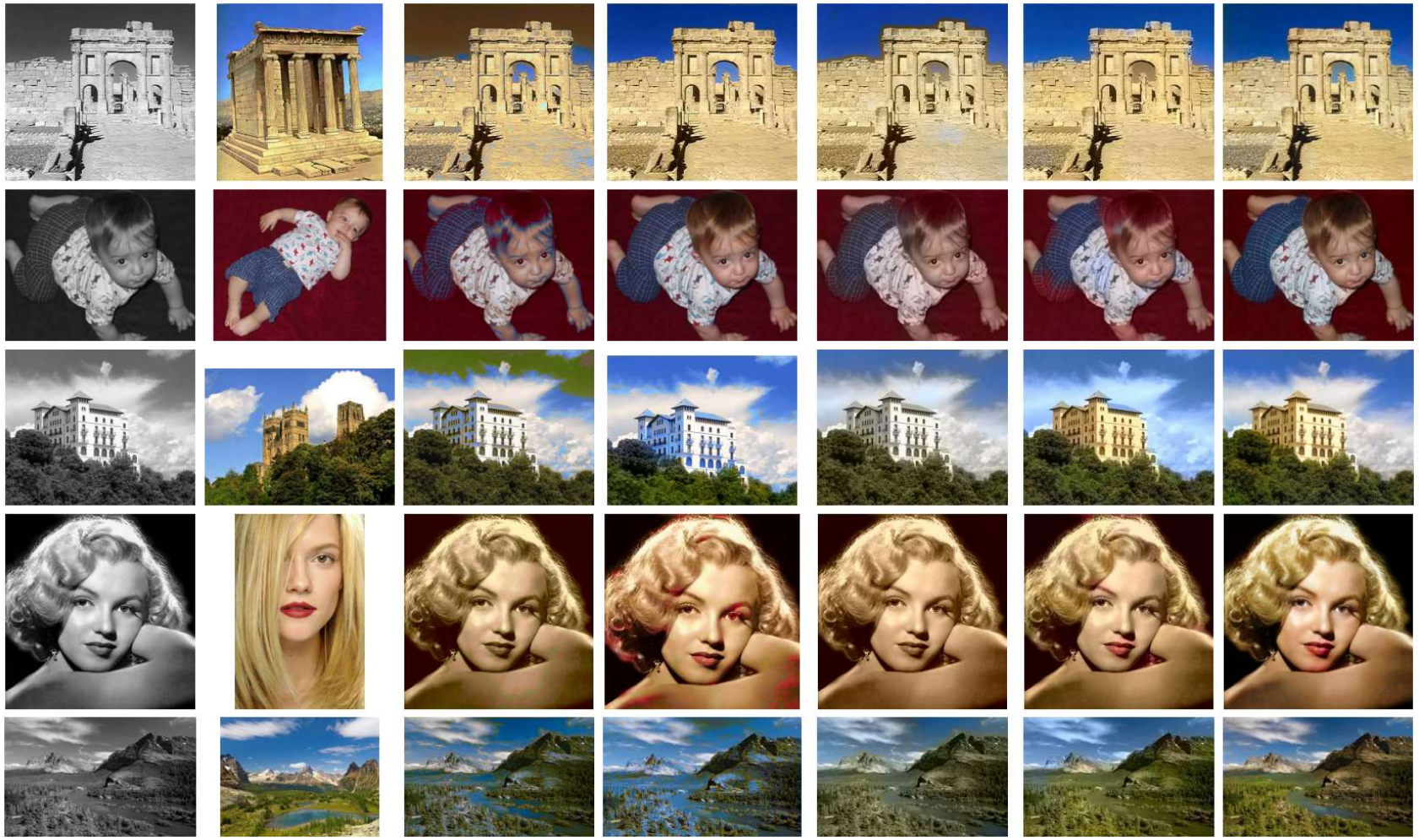


Figure 6: *Color reference recommendation pipeline. Input images: ImageNet dataset.*

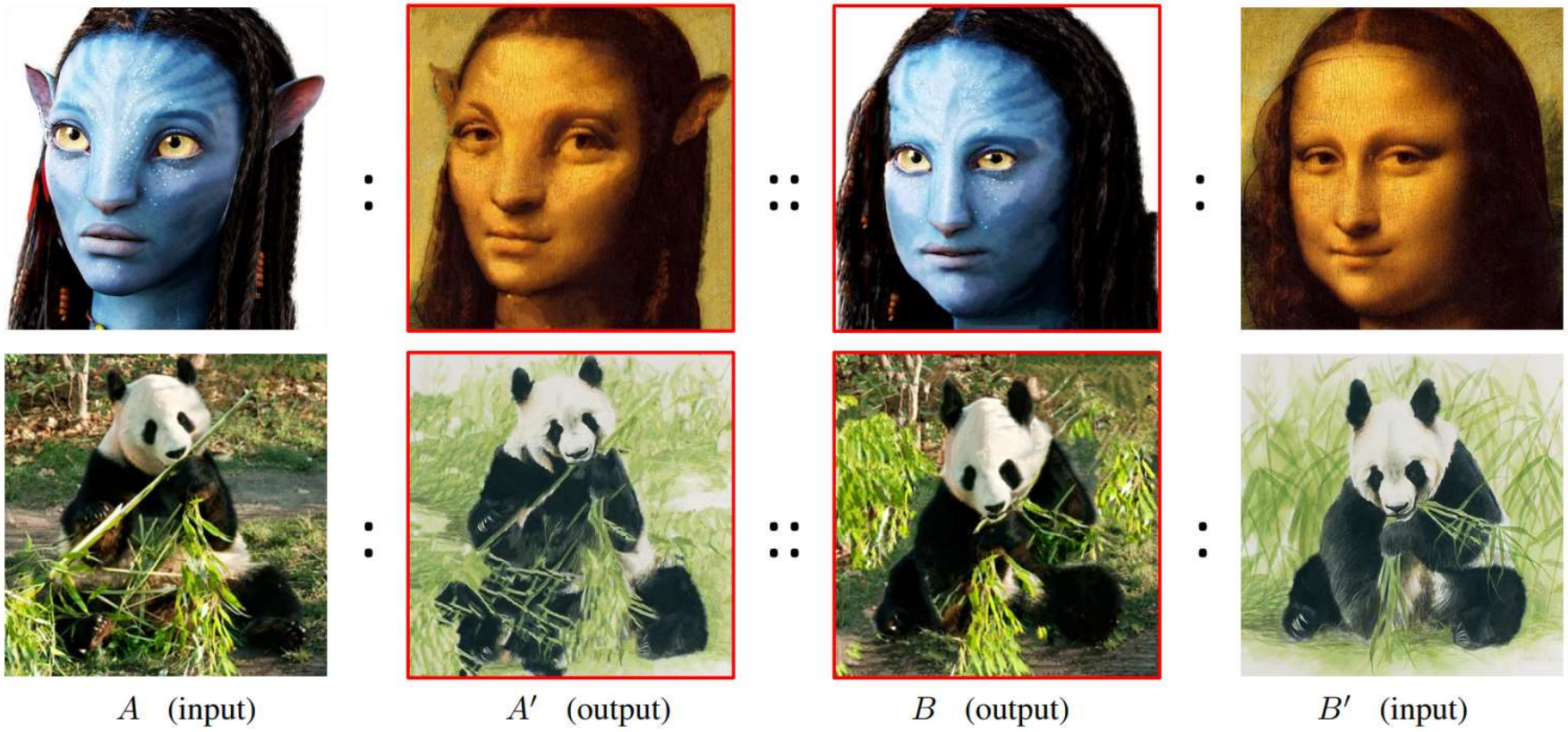
He et al. Deep Exemplar-based Colorization. Siggraph 2018.

Deep Exemplar-based Colorization



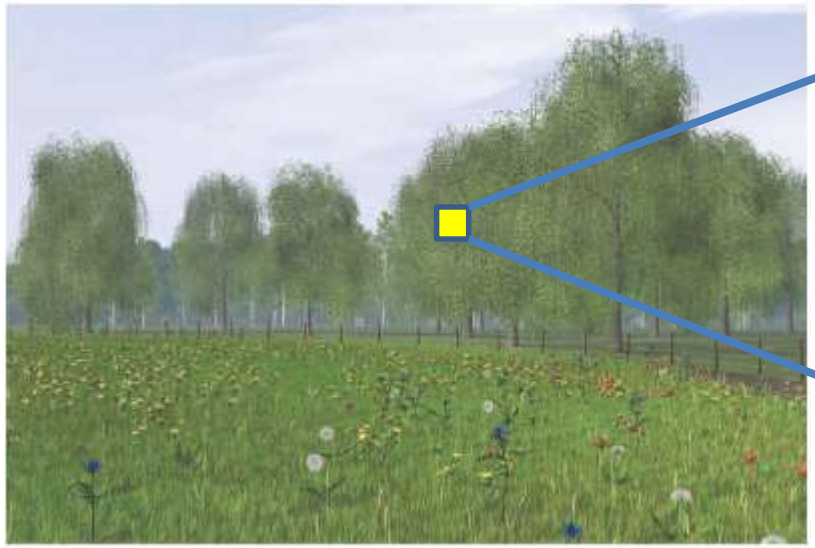
He et al. Deep Exemplar-based Colorization. Siggraph 2018.

Visual Attribute Transfer through Deep Image Analogy



Liao et al. Visual Attribute Transfer through Deep Image Analogy. Siggraph 2017.
https://liaojing.github.io/html/index.html

Feature Space?



(r, g, b)

(r, g, b, u, v)

Thanks!