数据聚类
Data Clustering

张煦尧 (xyz@nlpr.ia.ac.cn)

2021年12月8日

University of Chinese Academy of Sciences

Hierarchical Clustering
分级聚类

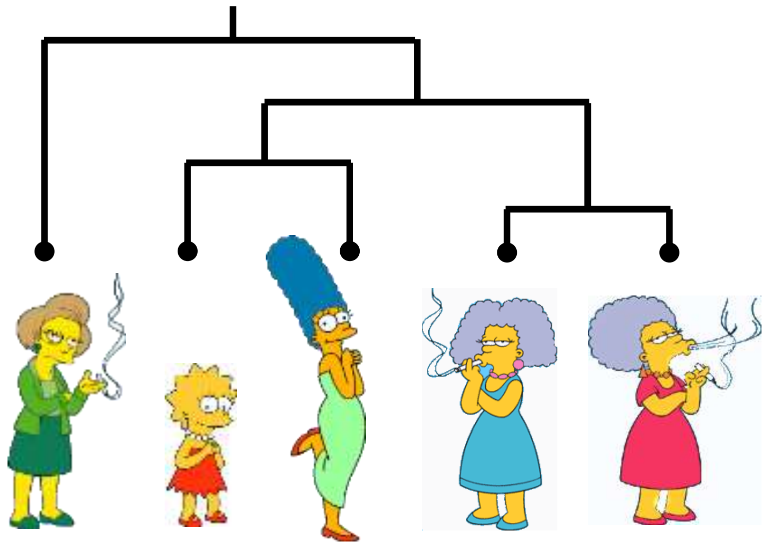
分级聚类(Hierarchical Clustering)

- **生物学上的物种分类**
  - 门、纲、目、科、属、种
  - 最相似的物种被分为“种”
  - 这种分层次归类方法对生物学研究发挥着巨大的作用
    - 生物学意义
    - 生物学研究意义：解决分歧、发现新的物种。
  - 这种思想也可以自然地应用到聚类分析之中，称为**分级聚类、层次聚类**或者**系统聚类**。

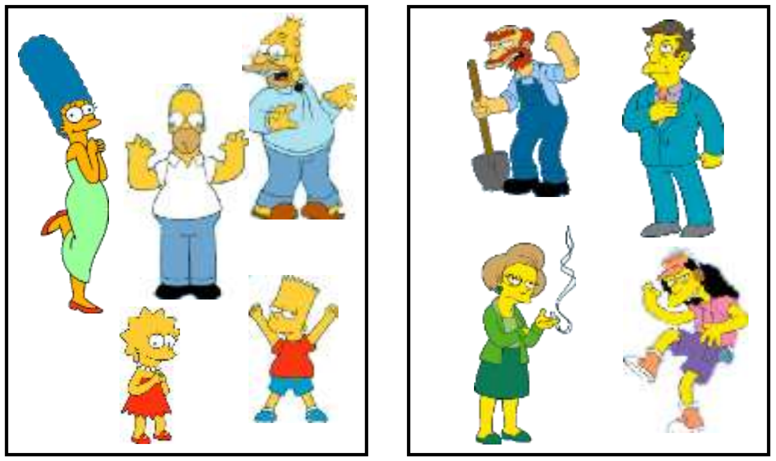
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion

Hierarchical



Partitional



分级聚类

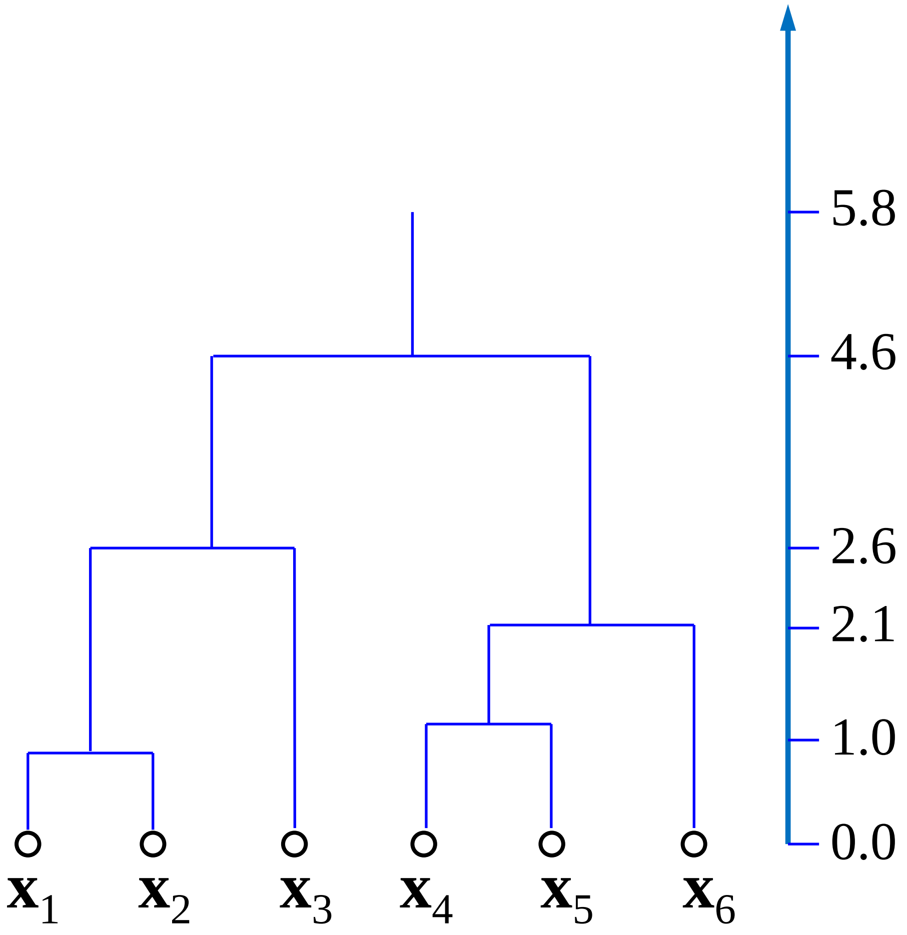
- **分级聚类思想**
  - 对于 n 个样本，极端的情况下，最多可以将数据分成 n 类；最少可以只分成一类，即全部样本都归为一类。
  - **凝聚的层次聚类（自底向上）**
    - 将每个样本作为一个簇，然后根据给定的规则**逐渐合并**一些样本，形成更大的簇，直到所有的样本都被分到一个合适的簇中。
  - **分裂的层次聚类（自顶向下）**
    - 将所有的样本置于一个簇中，然后根据给定的规则**逐渐细分**样本，得到越来越小的簇，直到某个终止条件得到满足。

分级聚类

- **自底向上的分级聚类步骤**
  - (1) 初始化：每个样本形成一个类
  - (2) 合并：计算任意两个类之间的距离（或相似性），将距离最小（或相似性最大）的两个类合并为一个类，记录下这两个类之间的距离（或相似性），其余类不变。
  - (3) 重复步骤 (2): 直到所有样本被合并到两个类之中。

分级聚类

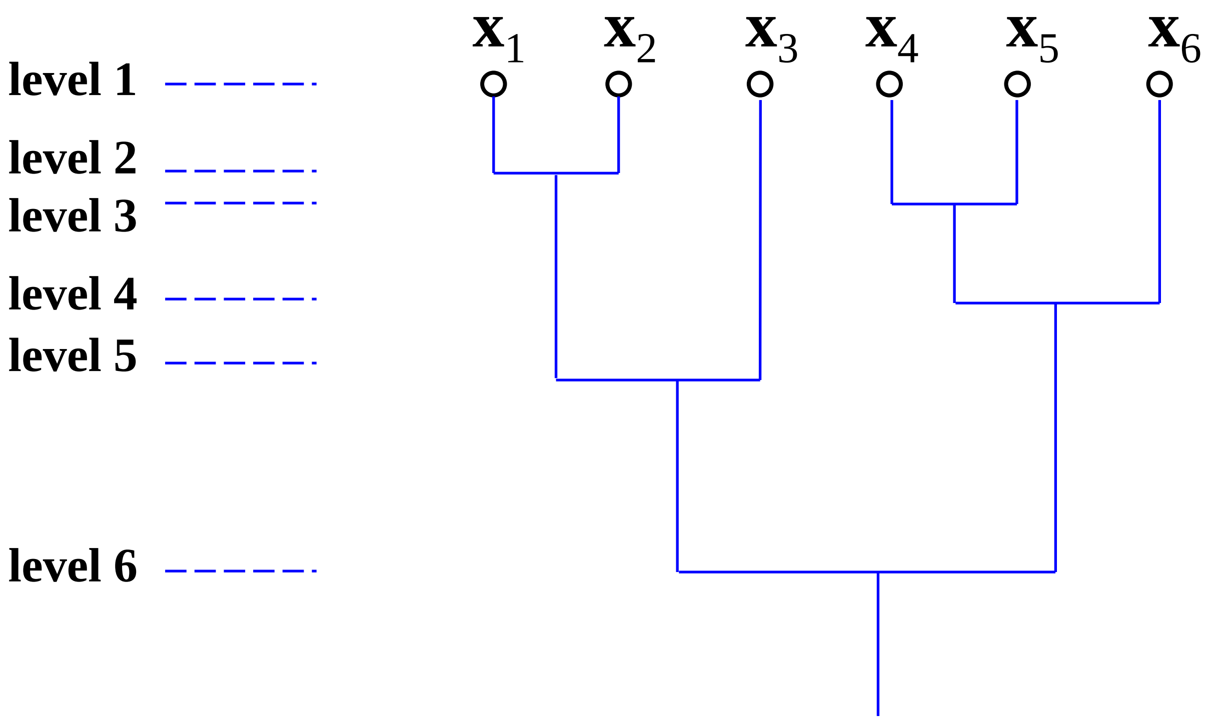
- **系统树**
  - 分级聚类是一种典型的**系统聚类法**。用一棵树来描述分级聚类结果，称为**聚类树**(dendrogram)，或**系统树图**。
  - 如右图，最底层的每个节点表示一个样本，**采用树枝连接两个合并的样本**，树枝的长度反映两个节点之间的距离（或相似性）。
  - 为方便，**采用“水平”来表示分级聚类过程的不同阶段。**开始为水平1，每个样本为一类，然后每执行一次“合并”增加一个水平。



系统树图（采用距离）

分级聚类

- **系统树：** **分级聚类的另一种表示**



系统树图（采用相似性）

分级聚类

- **分级聚类两个核心问题：**
  - 如何度量**样本之间**的距离（或相似性）
  - 如何度量**两个簇之间**的距离（或者相似性）
- **样本之间的距离（或相似性）：**
  - 欧氏距离、马式距离、城区距离、匹配距离、…
  - 相关系数、高斯相似性函数、余弦、距离倒数、…

分级聚类

- 簇与簇之间的距离

$$d_{\min}(D_i, D_j) = \min_{\substack{\mathbf{x} \in D_i \\ \mathbf{x}' \in D_j}} \|\mathbf{x} - \mathbf{x}'\|$$ 最小距离

$$d_{\max}(D_i, D_j) = \max_{\substack{\mathbf{x} \in D_i \\ \mathbf{x}' \in D_j}} \|\mathbf{x} - \mathbf{x}'\|$$ 最大距离

$$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in D_i} \sum_{\mathbf{x}' \in D_j} \|\mathbf{x} - \mathbf{x}'\|$$ 平均距离

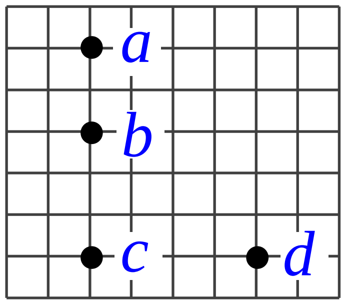
$$d_{mean}(D_i, D_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$ 中心距离

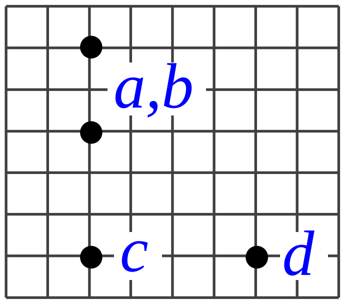
分级聚类

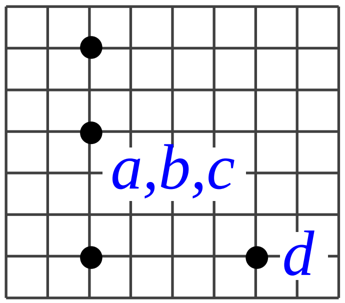
- **实践**
  - 根据数据特性、聚类目标的不同，通常需要采用不同的簇间距离。
  - 对同一数据集，采用不同的簇间连接通常会得到不同的聚类结果。

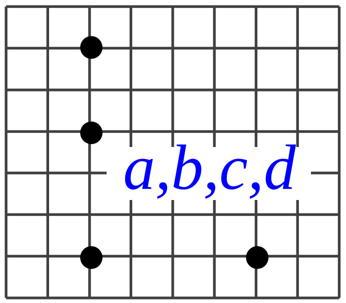
• 例子1：采用最近距离连接簇

(注：*a*到*d*之间的距离取整数)









	b	*c*	*d*
a	2	5	6
b		3	5
c			4

$\rightarrow$

	b	*c*	*d*
a	2	5	6
b		3	5
c			4

$\downarrow$

重新计算距离

	c	*d*
a,b	3	5
c		4

$\nearrow$

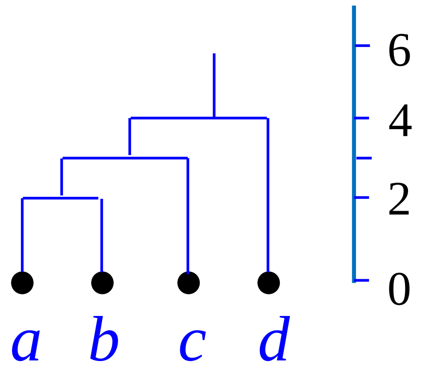
	c	*d*
a,b	3	5
c		4

$\downarrow$

	d
a,b,c	4

$\nearrow$

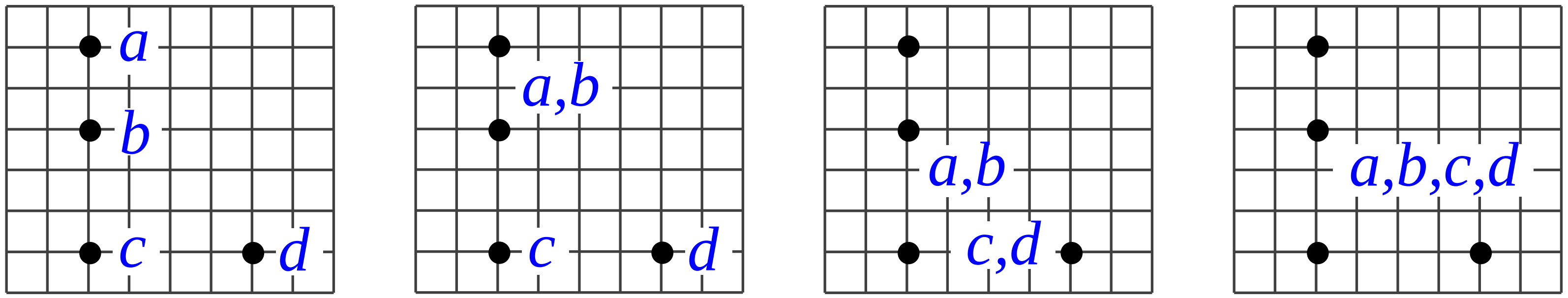
	d
a,b,c	4



簇与样本之间的距离采用最近距离

• 例子2：采用最远距离连接簇 (最大最小准则)

(注：*a*到*d*之间的距离取整数)



	b	c	d
a	2	5	6
b		3	5
c			4

	b	c	d
a	2	5	6
b		3	5
c			4

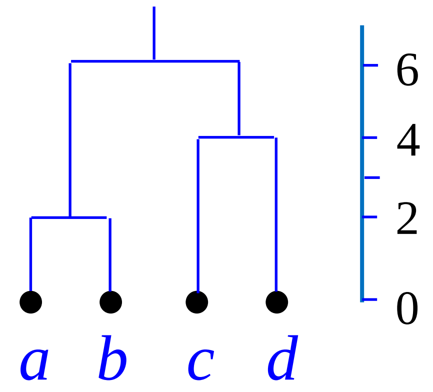
重新计算距离

	c	d
a,b	5	6
c		4

	c	d
a,b	5	6
c		4

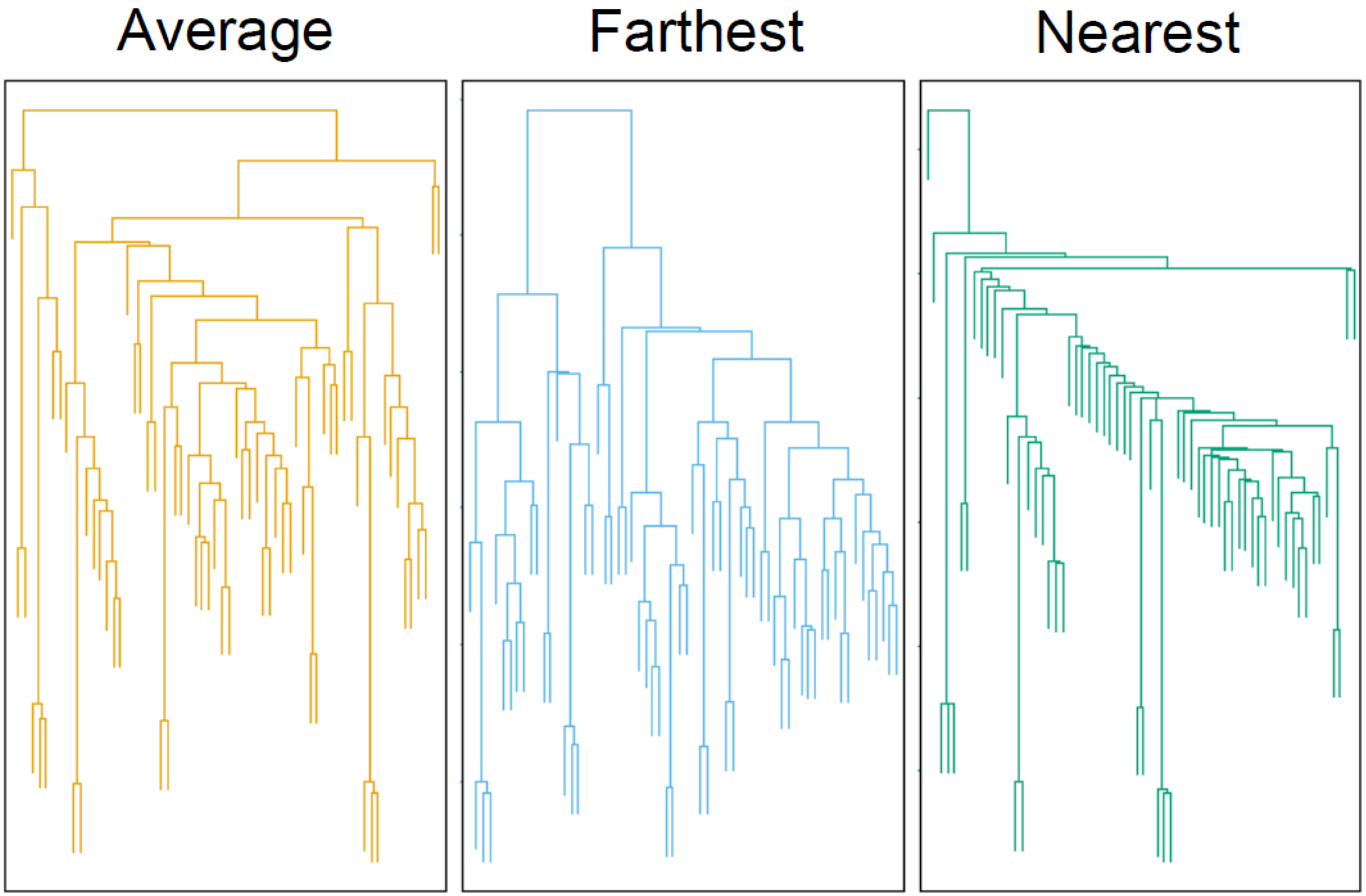
	c,d
a,b	6

	c,d
a,b	6



簇与样本之间的距离采用最远距离

距离决定聚类



- 例子3：请按最小距离准则对如下6个样本进行分级聚类：

$$\mathbf{x}_1 = (0, 3, 1, 2, 0)$$
$$\mathbf{x}_2 = (1, 3, 0, 1, 0)$$
$$\mathbf{x}_3 = (3, 3, 0, 0, 1)$$
$$\mathbf{x}_4 = (1, 1, 0, 2, 0)$$
$$\mathbf{x}_5 = (3, 2, 1, 2, 1)$$
$$\mathbf{x}_6 = (4, 1, 1, 1, 0)$$

解：将每个样本单独看成一类，计算各点对之间的距离，见下表：

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
$\mathbf{x}_1$	0					
$\mathbf{x}_2$	$\sqrt{3}$	0				
$\mathbf{x}_3$	$\sqrt{15}$	$\sqrt{6}$	0			
$\mathbf{x}_4$	$\sqrt{6}$	$\sqrt{5}$	$\sqrt{13}$	0		
$\mathbf{x}_5$	$\sqrt{11}$	$\sqrt{8}$	$\sqrt{6}$	$\sqrt{7}$	0	
$\mathbf{x}_6$	$\sqrt{21}$	$\sqrt{14}$	$\sqrt{8}$	$\sqrt{11}$	$\sqrt{4}$	0

解 (续):

基于上述矩阵，根据最小距离准则，应将 $\mathbf{x}_1$和$\mathbf{x}_2$合并为一类，得到 $G_1=\{\mathbf{x}_1,\mathbf{x}_2\}$，$\{\mathbf{x}_3\}$，$\{\mathbf{x}_4\}$，$\{\mathbf{x}_5\}$，$\{\mathbf{x}_6\}$。**按最小距离原则**重新计算各类之间的距离，见下表：

	G_1	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
G_1	0				
$\mathbf{x}_3$	$\sqrt{6}$	0			
$\mathbf{x}_4$	$\sqrt{5}$	$\sqrt{13}$	0		
$\mathbf{x}_5$	$\sqrt{8}$	$\sqrt{6}$	$\sqrt{7}$	0	
$\mathbf{x}_6$	$\sqrt{14}$	$\sqrt{8}$	$\sqrt{11}$	$\sqrt{4}$	0

解释：比如，类G_1与$\{\mathbf{x}_3\}$之间的距离，按最小距离准则，计算为$\mathbf{x}_2$与$\mathbf{x}_3$的距离

解 (续):

基于上述矩阵，根据最小距离准则，应将 $\mathbf{x}_5$和$\mathbf{x}_6$合并为一类，得到 $G_1=\{\mathbf{x}_1,\mathbf{x}_2\}$， $\{\mathbf{x}_3\}$， $\{\mathbf{x}_4\}$， $G_2=\{\mathbf{x}_5,\mathbf{x}_6\}$。**按最小距离原则**重新计算各类之间的距离，见下表：

	G_1	$\mathbf{x}_3$	$\mathbf{x}_4$	G_2
G_1	0			
$\mathbf{x}_3$	$\sqrt{6}$	0		
$\mathbf{x}_4$	$\sqrt{5}$	$\sqrt{13}$	0	
G_2	$\sqrt{8}$	$\sqrt{6}$	$\sqrt{7}$	0

解释：比如，类G_1与类G_2之间的距离，按最小距离准则，计算为$\mathbf{x}_2$与$\mathbf{x}_5$的距离

解 (续):

基于上述矩阵，根据最小距离准则，应将 G_1和$\mathbf{x}_3$合并为一类，得到 $G_3=G_1\cup\{\mathbf{x}_4\}$，$\{\mathbf{x}_3\}$，$G_2=\{\mathbf{x}_5,\mathbf{x}_6\}$。**按最小距离原则**重新计算各类之间的距离，见下表:

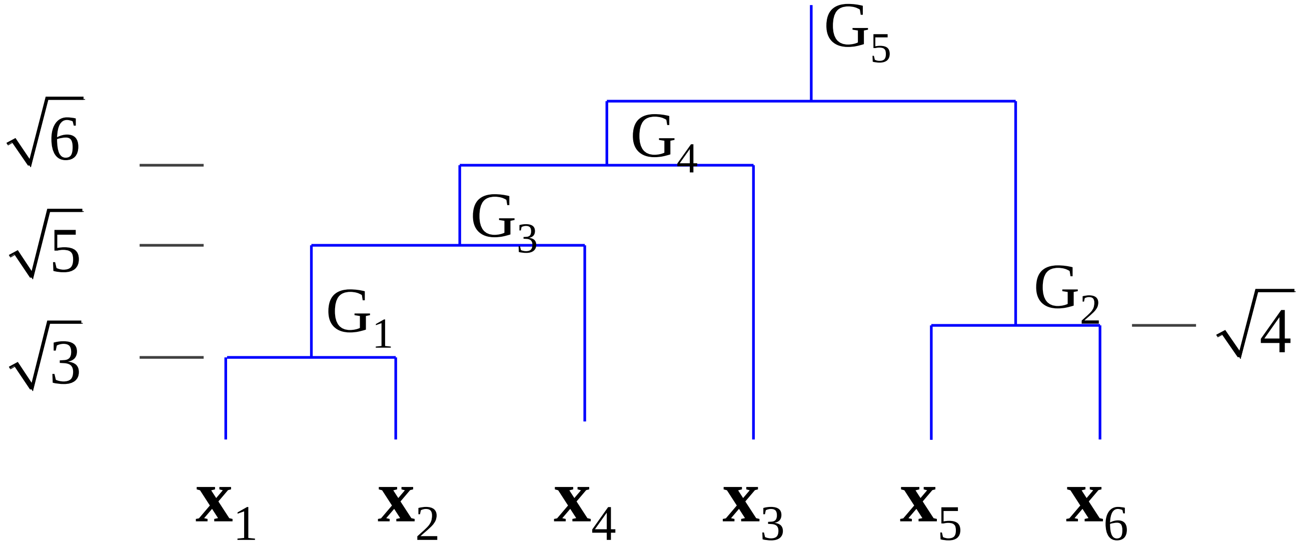
	G_3	$\mathbf{x}_3$	G_2
G_3	0		
$\mathbf{x}_3$	$\sqrt{6}$	0	
G_2	$\sqrt{7}$	$\sqrt{6}$	0

解释：比如，类G_3与$\{\mathbf{x}_3\}$之间的距离，按最小距离准则，计算为$\mathbf{x}_2$与$\mathbf{x}_3$的距离

中国科学院大学 University of Chinese Academy of Sciences

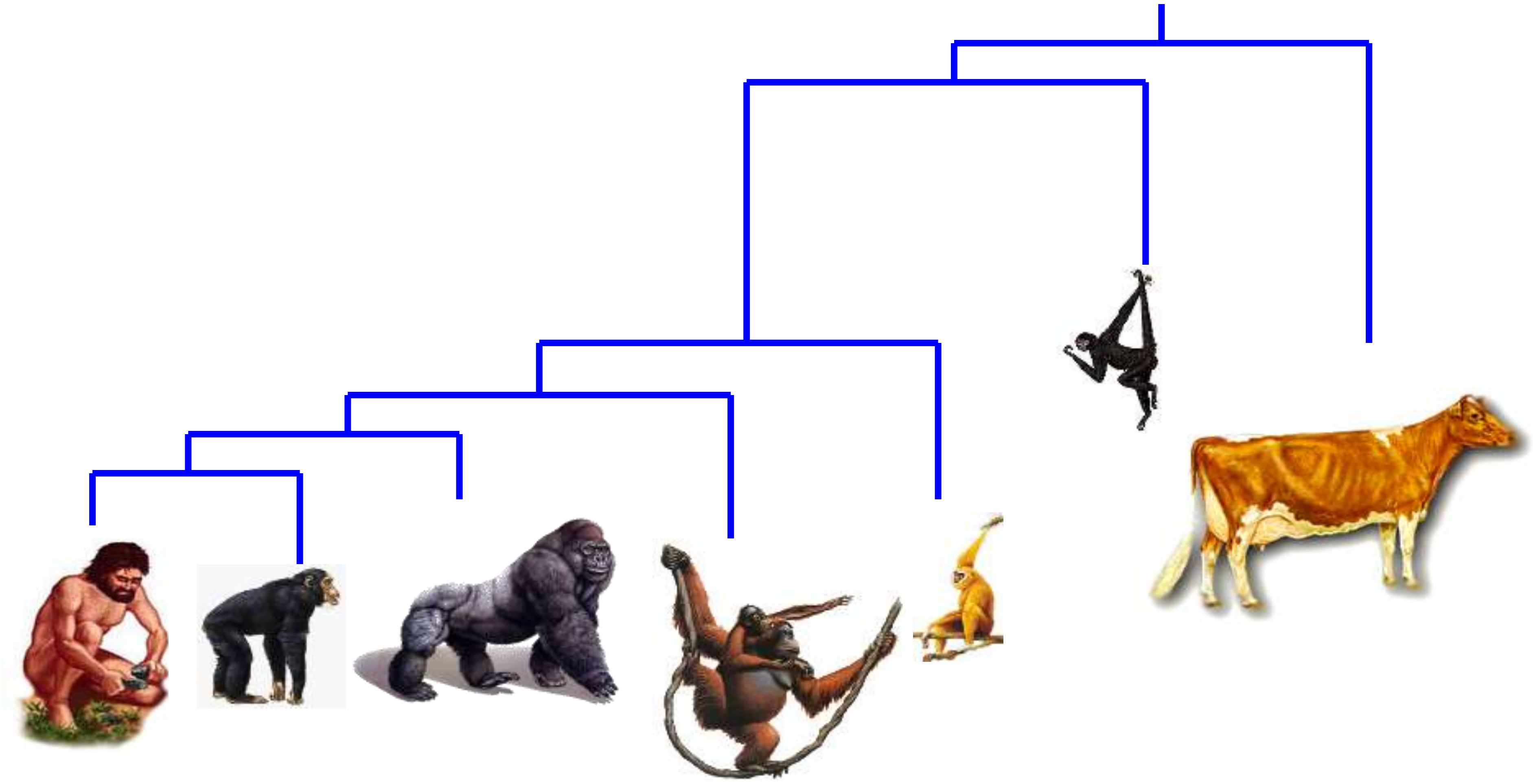
解 (续):

基于上述矩阵，根据最小距离准则，应将 G_3和$\mathbf{x}_3$合并为一类，当然也可以将 G_2 和 $\mathbf{x}_3$合并为一类。如选择前者，得到 $G_4=G_3\cup\{\mathbf{x}_3\}$，$G_2=\{\mathbf{x}_5,\mathbf{x}_6\}$ 。最后， G_4和 G_2 合并为一类，这样所有数据将被合并在一起。

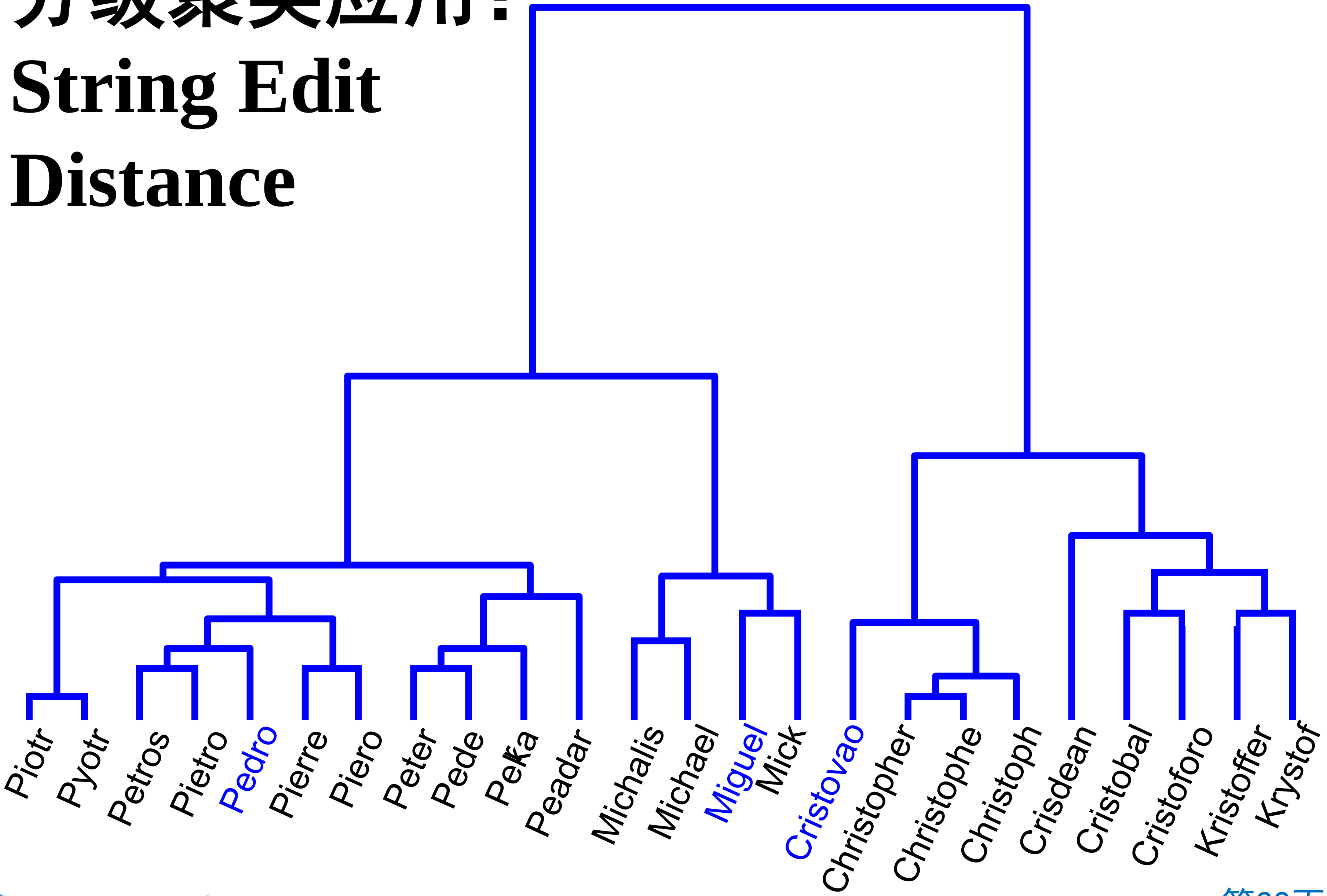


系统树图

分级聚类应用



分级聚类应用：String Edit Distance



String Edit Distance

To measure the similarity between two objects, transform one into the other, and measure how much effort it took. The measure of effort becomes the distance measure.

The distance between Patty and Selma:

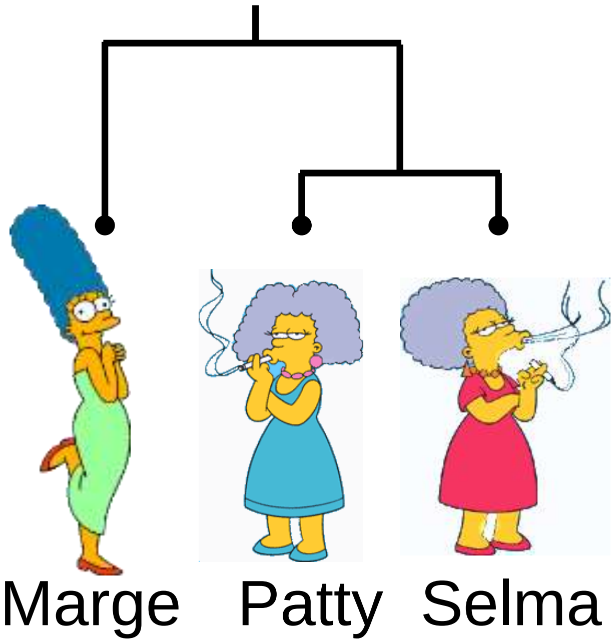
```
Change dress color,    1 point
Change earring shape,  1 point
Change hair part,      1 point
```

D(Patty,Selma) = **3**

The distance between Marge and Selma:

```
Change dress color,    1 point
Add earrings,          1 point
Decrease height,       1 point
Take up smoking,       1 point
Lose weight,           1 point
```

D(Marge,Selma) = **5**



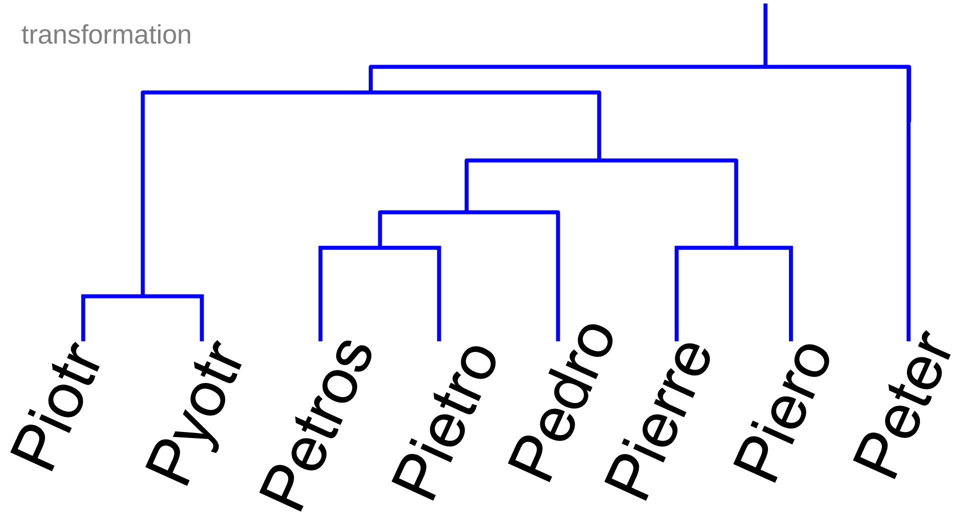
This is called the "edit distance" or the "transformation distance"

Edit Distance Example

It is possible to transform any string *Q* into string *C*, using only *Substitution*, *Insertion* and *Deletion*.
Assume that each of these operators has a cost associated with it.

The similarity between two strings can be defined as the cost of the cheapest transformation from *Q* to *C*.

Note that for now we have ignored the issue of how we can find this cheapest transformation



How similar are the names "Peter" and "Piotr"?

Assume the following cost function

Substitution	1 Unit
Insertion	1 Unit
Deletion	1 Unit

D(**Peter**,**Piotr**) is 3

Peter

↓ Substitution (i for e)

Piter

↓ Insertion (o)

Pioter

↓ Deletion (e)

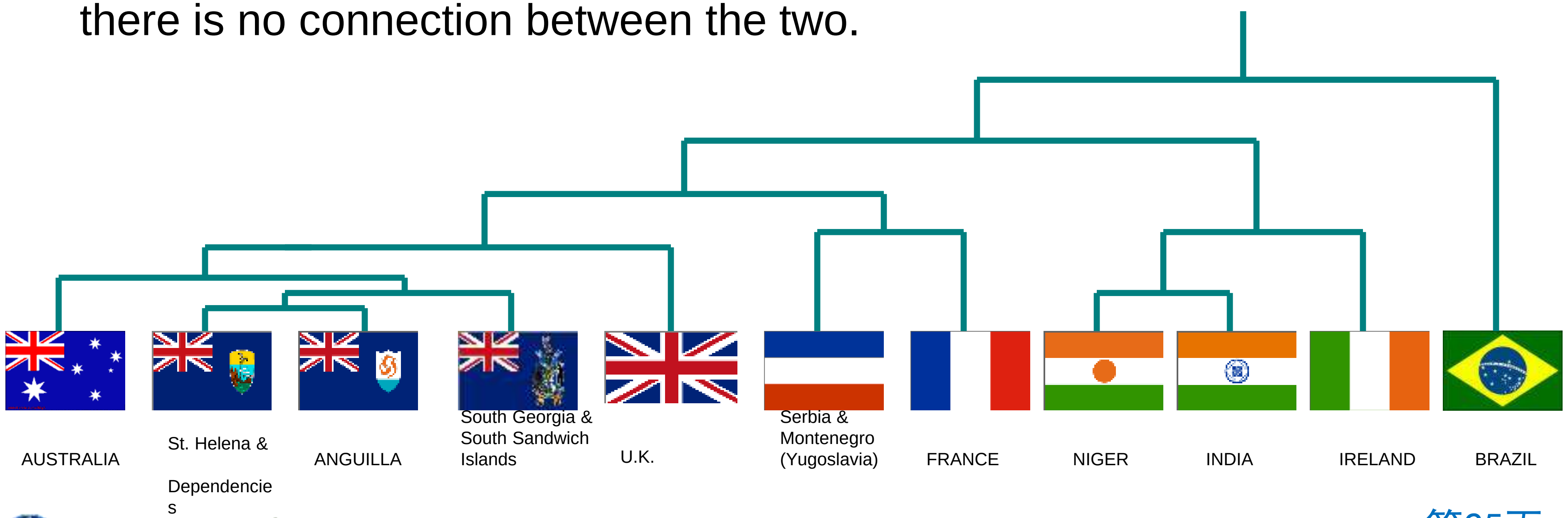
Piotr

分级聚类应用

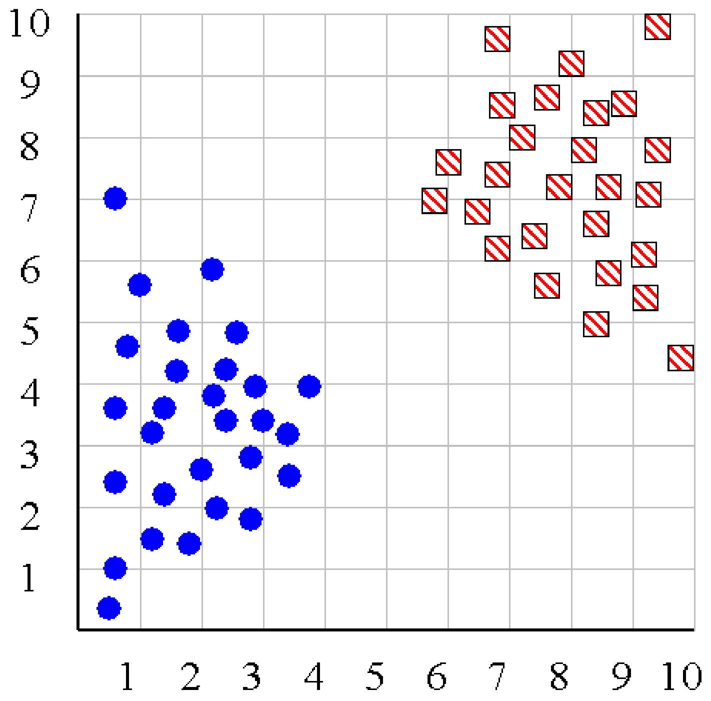
Hierarchal clustering can sometimes show patterns that are meaningless or spurious

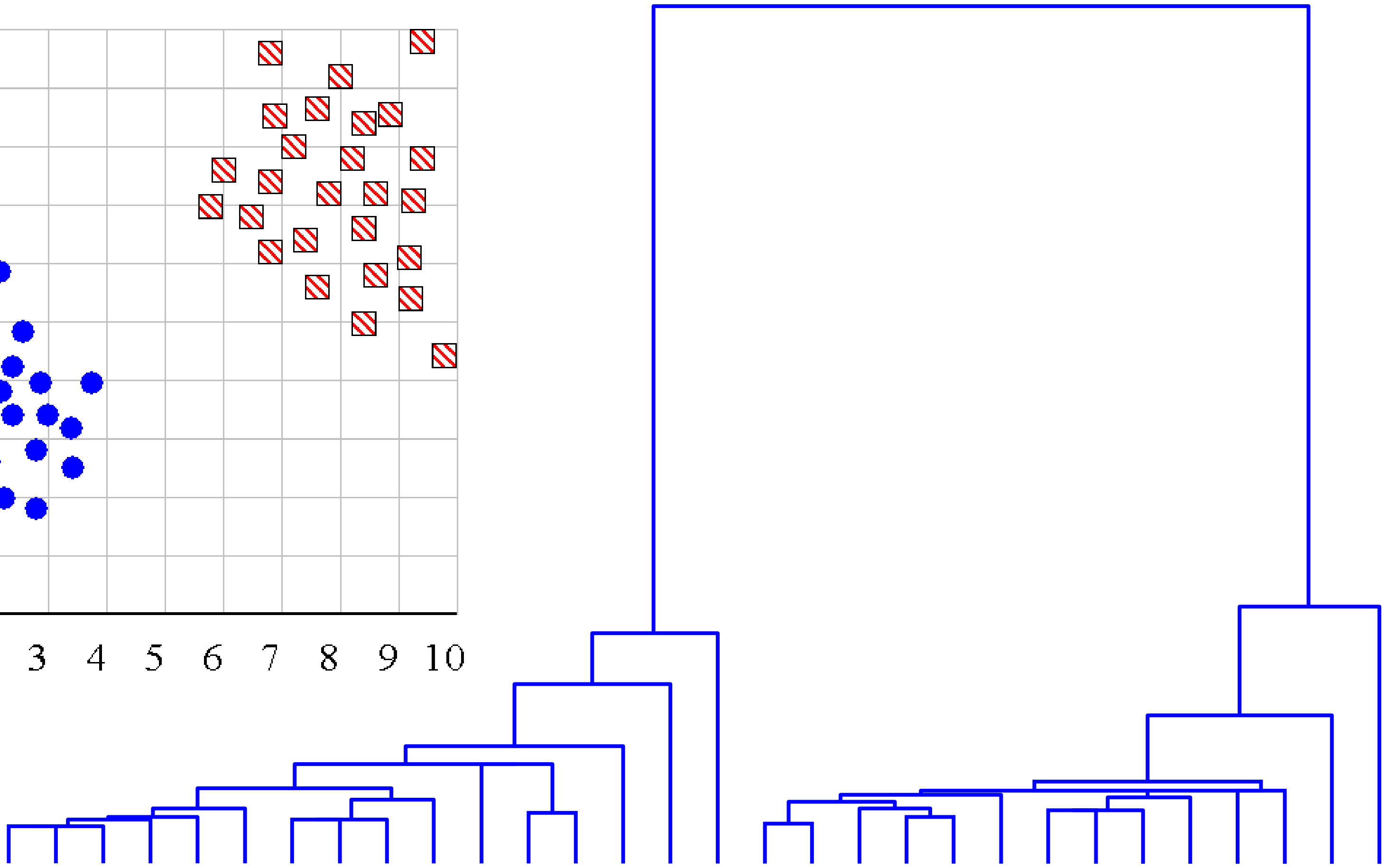
The tight grouping of Australia, Anguilla, St. Helena etc is meaningful; all these countries are former UK colonies

However the tight grouping of Niger and India is completely spurious; there is no connection between the two.


AUSTRALIA
St. Helena & Dependencies
ANGUILLA
South Georgia & South Sandwich Islands
U.K.
Serbia & Montenegro (Yugoslavia)
FRANCE
NIGER
INDIA
IRELAND
BRAZIL

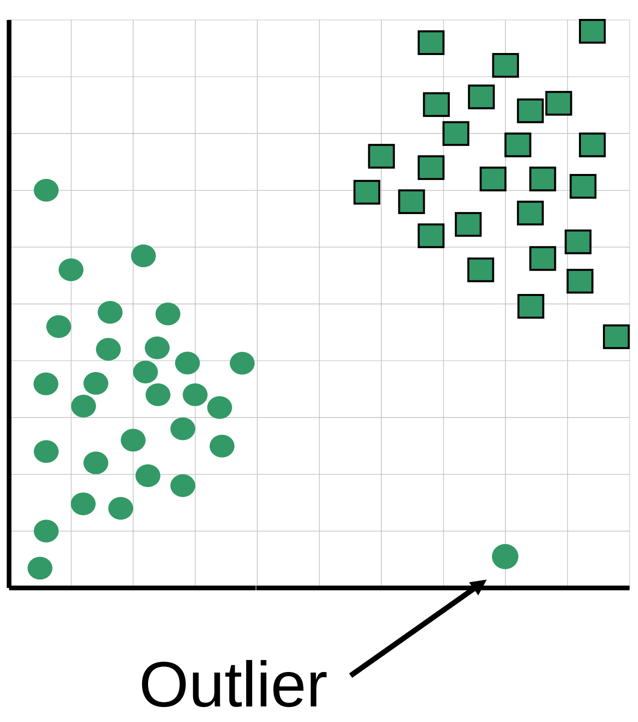
分级聚类用于选择类别数

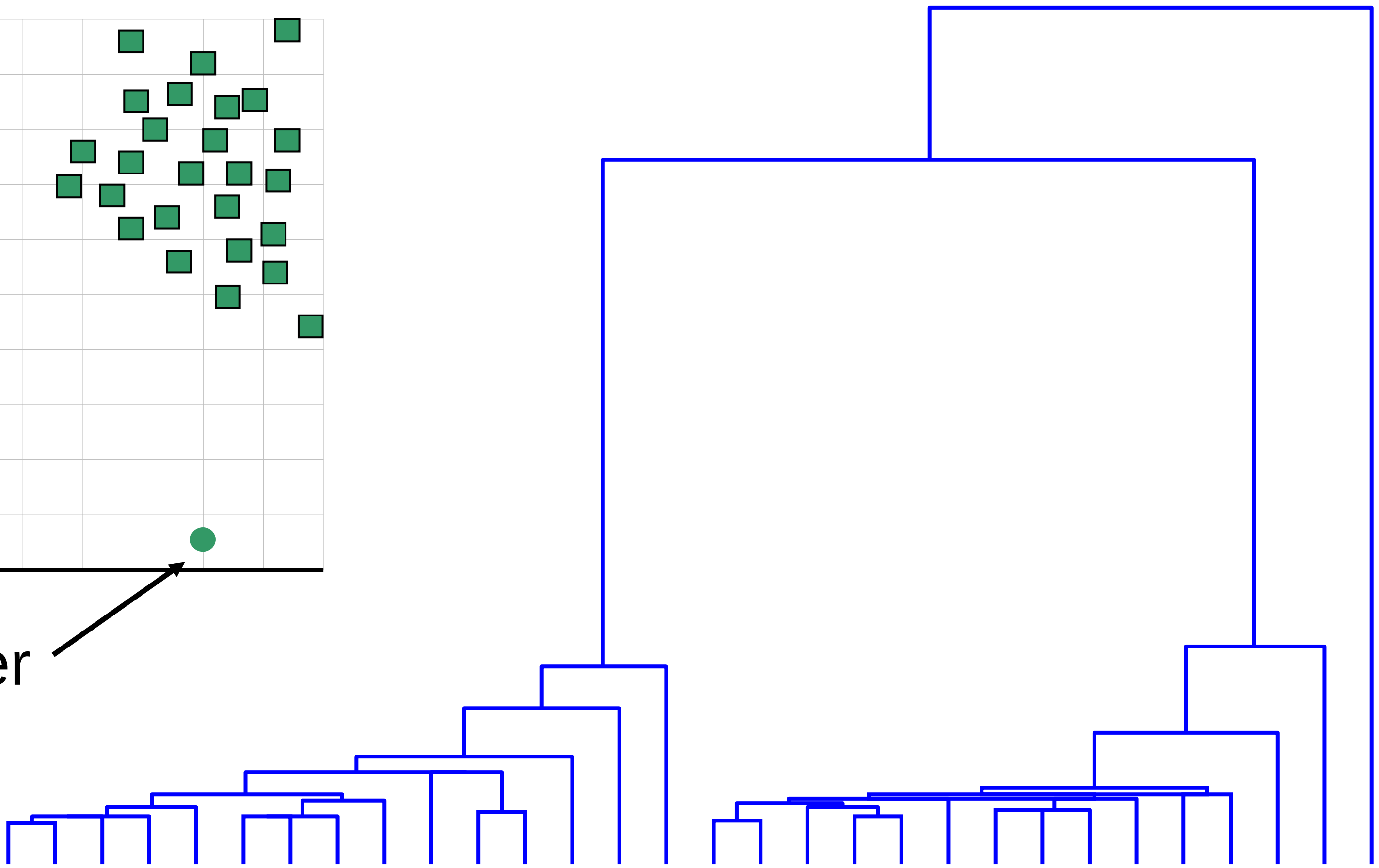




分级聚类用于发现Outlier

The single isolated branch is suggestive of a data point that is very different to all others

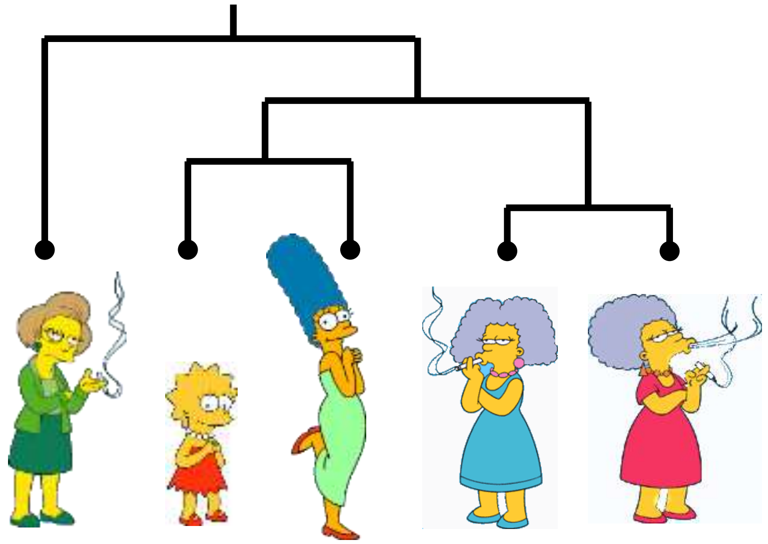




分级聚类

The number of dendrograms with n leafs $= (2n-3)!/[(2^{(n-2)})(n-2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	…
10	34,459,425



Since we cannot test all possible trees we will have to heuristic search of all possible trees. We could do this..

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

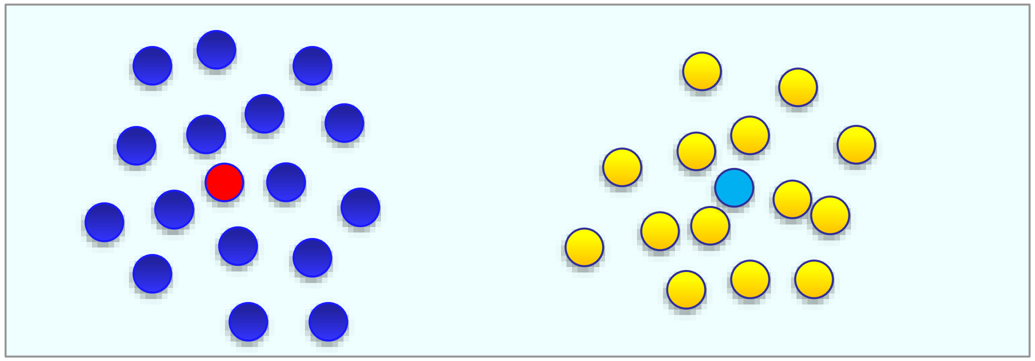
Hierarchal Clustering Methods Summary

- No need to specify the number of clusters in advance
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least O(n^2), where n is the number of total objects
- Like any heuristic search algorithms, local optima are a problem
- Interpretation of results is subjective

Spectral Clustering
谱聚类
→ *K*-means in spectrum space

引言

- **回忆K-means聚类**

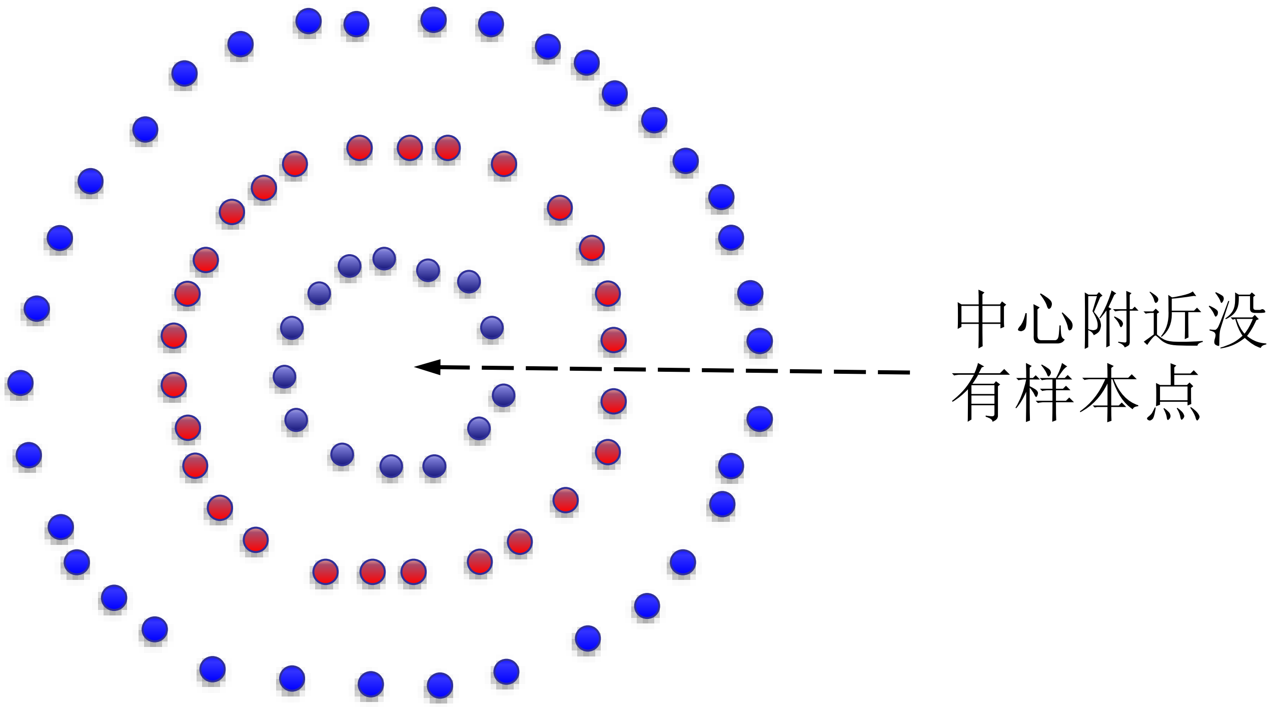


每个簇均可以用中心点来表示

(特别适合于单个簇符合高斯分布的情形)

引言

- 对于其它分布情形



Spectral clustering allows us to address these sorts of clusters!

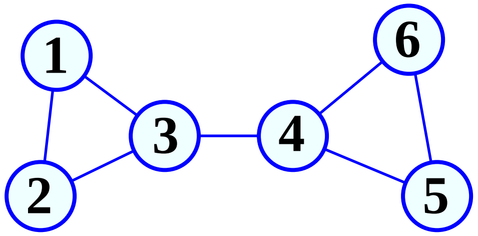
引言

- **谱学习方法**
  - 广义上讲，任何在学习过程中应用到矩阵特征值分解的方法均叫**谱学习方法**，比如主成分分析（PCA）、线性判别成分分析（LDA）、流形学习中的谱嵌入方法、谱聚类、等等。

- **谱聚类**
  - 谱聚类算法建立在**图论的谱图理论**基础之上，其本质是将聚类问题转化为一个**图上的关于顶点划分的最优问题**。
  - 谱聚类算法建立在**点对亲和性**基础之上，理论上能对任意分布形状的样本空间进行聚类。
  - 最早关于谱聚类的研究始于1973年，主要用于计算机视觉和VLSI设计领域。从2000年开始，谱聚类逐渐成为机器学习领域中的一个研究热点。

图论基本概念

- **图 G：** 由顶点集 V 和边集 E 所构成，记为 G(V,E)。根据边是否有向，可以分为无向图或者有向图。
- **图 G 的邻接矩阵 W：**
  – 行数和列数等于矩阵顶点的个数；
  – 矩阵元素为 0 或 1。1 表示对应的一对顶点有边相连，0表示没有边相连。

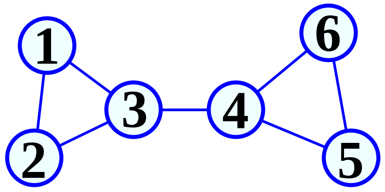


$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

中国科学院大学
University of Chinese Academy of Sciences

图论基本概念

- **顶点的度：等于与顶点相连接的边的条数。**
- **度矩阵：　为一个对角矩阵。**将邻接矩阵各行元素累加至对应的主对角元素，可得到**度矩阵 D**。

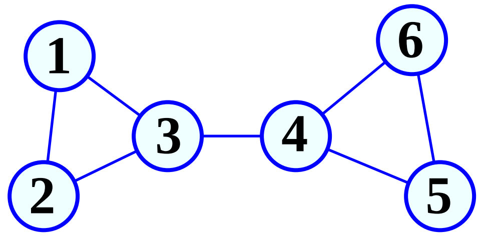


$$\mathbf{W}=\begin{pmatrix}0&1&1&0&0&0\\1&0&1&0&0&0\\1&1&0&1&0&0\\0&0&1&0&1&1\\0&0&0&1&0&1\\0&0&0&1&1&0\end{pmatrix},\quad \mathbf{D}=\begin{pmatrix}2&&&&&\\&2&&&&\\&&3&&&\\&&&3&&\\&&&&2&\\&&&&&2\end{pmatrix}$$

图论基本概念

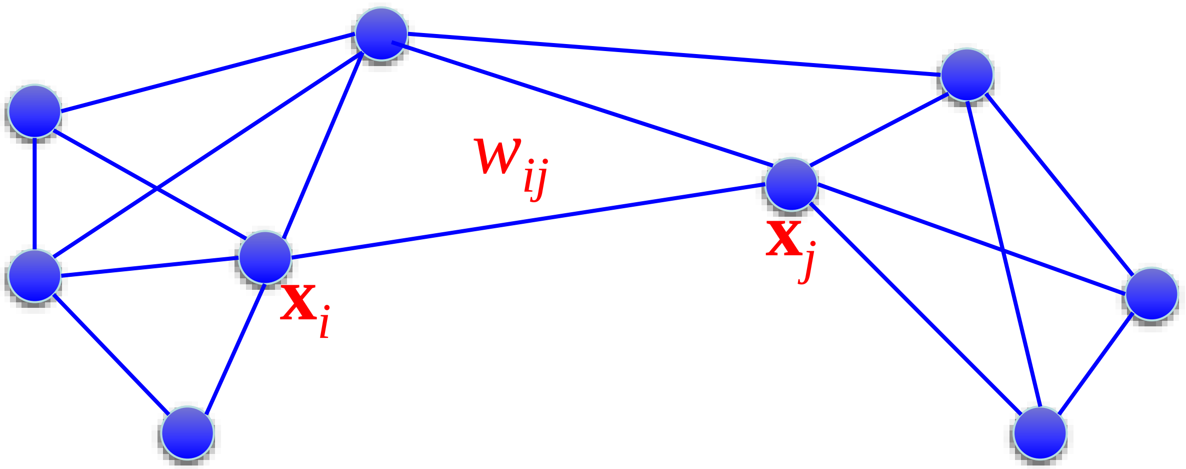
- **拉普拉斯矩阵**
  - 度矩阵减去邻接矩阵得到拉普拉斯矩阵 $\mathbf{L}$。

$$\mathbf{D}=\begin{pmatrix}2&&&&&\\&2&&&&\\&&3&&&\\&&&3&&\\&&&&2&\\&&&&&2\end{pmatrix},\quad \mathbf{L}=\mathbf{D}-\mathbf{W}=\begin{pmatrix}2&-1&-1&0&0&0\\-1&2&1&0&0&0\\-1&-1&3&-1&0&0\\0&0&-1&3&-1&-1\\0&0&0&-1&2&-1\\0&0&0&-1&-1&2\end{pmatrix}$$



图论基本概念

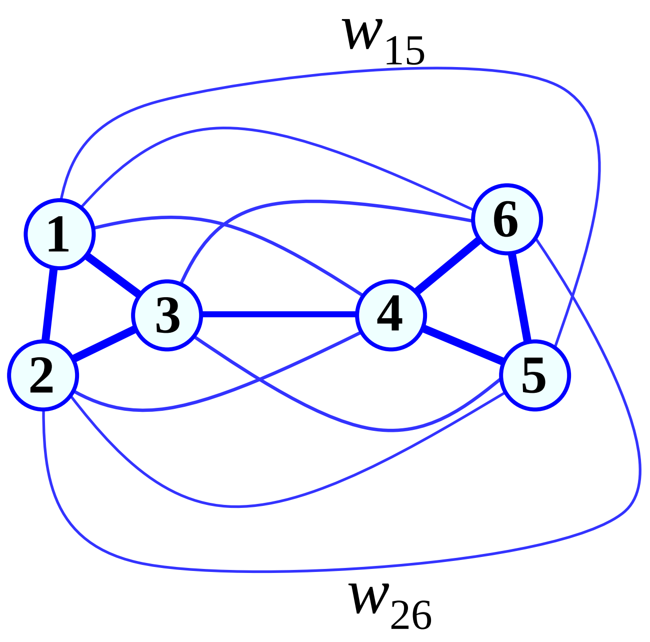
- **基于数据集的图构造**
  - 将每一个数据点视为图的一个顶点，顶点之间可以有边相连。每条边上加上一些权重，用来反映点对亲和性（即相似性）。



比如，采用高斯函数计算点对亲和性：$w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$

图论基本概念

- **图构造 G(V, E)**
  - 根据某种测度构建点对相似度矩阵

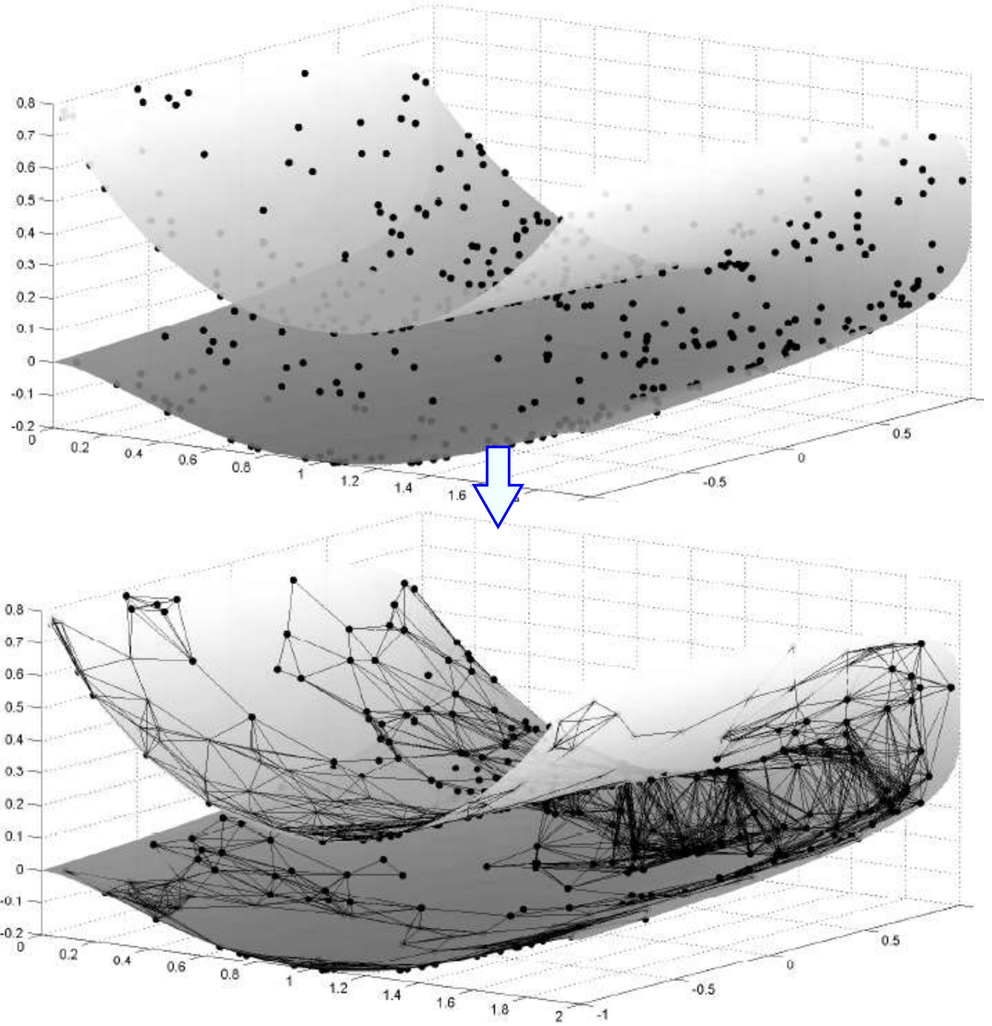


$$\begin{pmatrix} 0 & w_{12} & w_{13} & w_{14} & w_{15} & w_{16} \\ & 0 & w_{23} & w_{24} & w_{25} & w_{26} \\ & & 0 & w_{34} & w_{35} & w_{36} \\ & \text{对} & & 0 & w_{45} & w_{46} \\ & & \text{称} & & 0 & w_{56} \\ & & & & & 0 \end{pmatrix}$$

点对相似度矩阵

图论基本概念

- 图构造 G(V, E)
  - 全连接
  - 局部连接
    - *k* - 近邻
    - *ε* - 半径

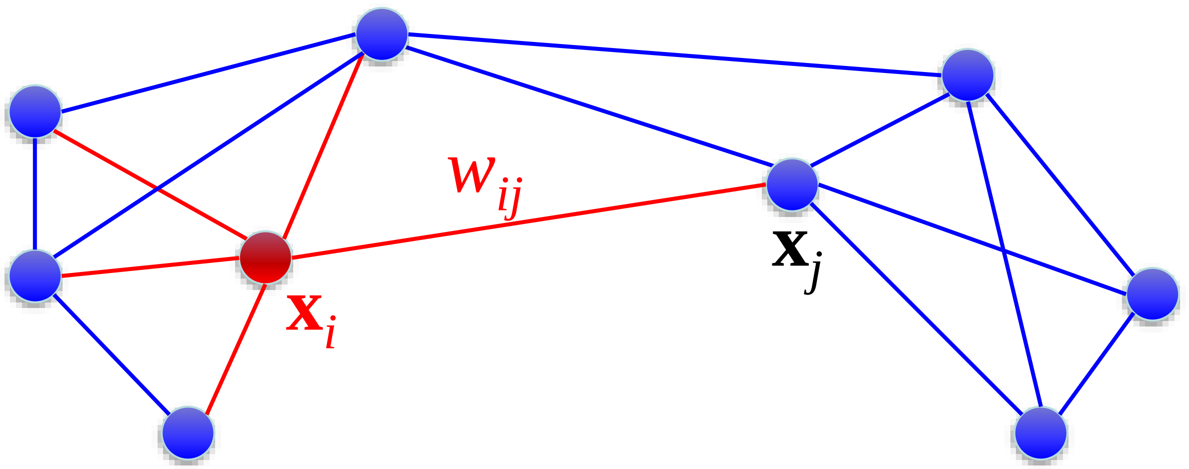


***k*-近邻：** 对每个数据点 $\mathbf{x}_i$，首先在所有样本中找出不包含 $\mathbf{x}_i$ 的 *k* 个最邻近的样本点，然后 $\mathbf{x}_i$ 与每个邻近样本点均有一条边相连，从而完成图构造。

为了保证W矩阵的对称性，可以令W=(W^T+W)/2

图论基本概念

- **顶点的度：**所有与该顶点相连接的边的权重之和。



$$d_i = \sum_{j \in V} w_{ij}$$

（如果顶点 $\mathbf{x}_j$ 不与 $\mathbf{x}_i$ 相边接，则 $w_{ij} = 0$。）

图论基本概念

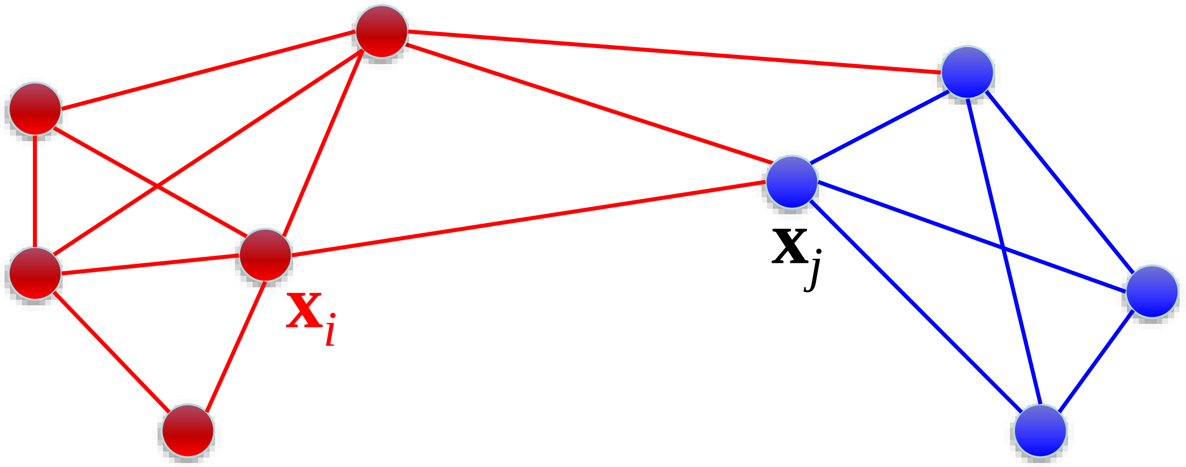
- **拉普拉斯矩阵(Laplacian matrix)**
  - 拉普拉斯矩阵是描述图的一种矩阵。给定一个具有 n 个顶点的图，其拉普拉斯矩阵描述为：

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

  - 其中，$\mathbf{D}$ 为一个对角矩阵，主对角元素表示顶点的度。$\mathbf{W}$ 为亲和度矩阵，其元素 w_{ij} 表示顶点 $\mathbf{x}_i$ 与 $\mathbf{x}_j$ 之间的亲和程度（即相似度）。

图论基本概念

- **子图 A⊂V 的势 |A|：**　　　　等于其所包含的顶点个数。
- **子图 A⊂V 的体积 *vol(A)*：**等于其中所有顶点的度之和。



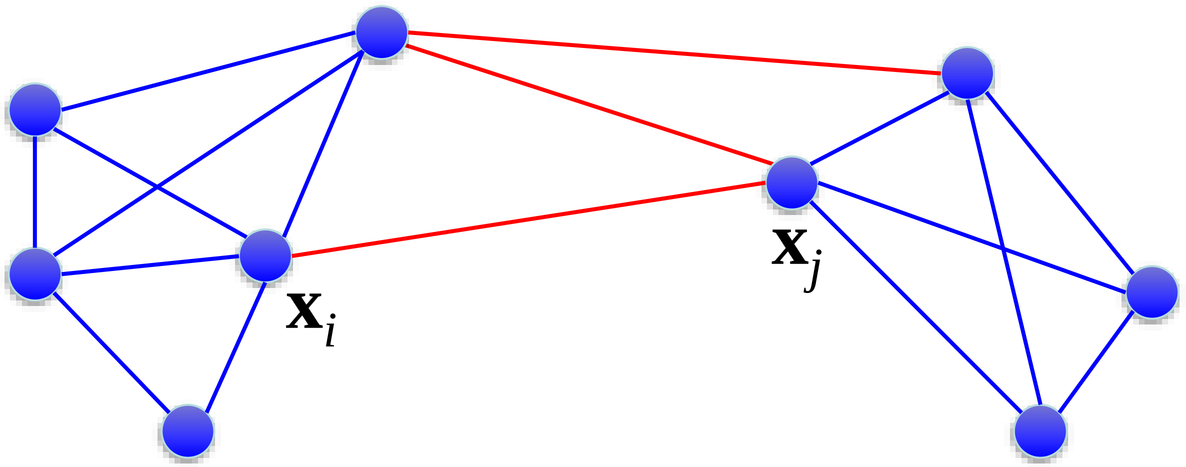
$$vol(A) = \sum_{i \in A} d_i$$

图论基本概念

- **子图A的补图：** V 中去掉 A 的顶点所构成的子图$\bar{A}$：

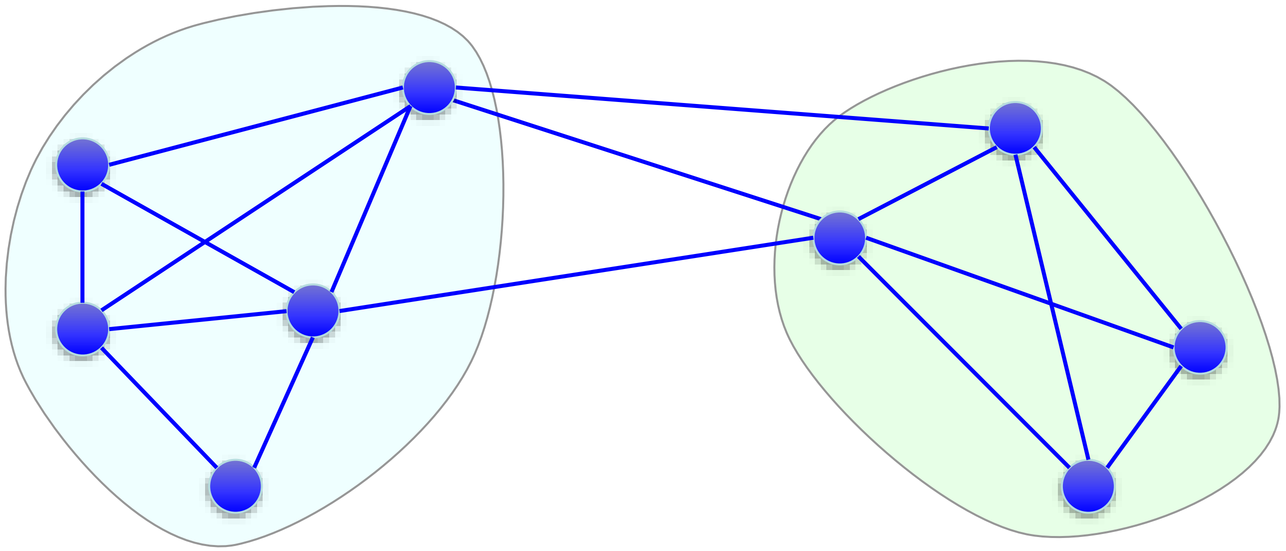
$$\bar{A} = V - A$$

- **边割：** 指边 E 的一个子集，去掉该子集中的边，图就变成两个连通子图。



图论基本概念

- **图切割：**
  - 设 $A_1, A_2, \ldots, A_k$ 为顶点集合 A 的非空连通子集，如果 $A_i \cap A_j = \varnothing, i \neq j$，且 $A_1 \cup A_2 \cup \ldots \cup A_k = V$，则称 $A_1, A_2, \ldots, A_k$ 为图 G 的一个分割。



图论基本概念

- **子图相似度：子图 A 与子图 B 的相似度**定义为连接两个子图所有边的权重之和：

$$W(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

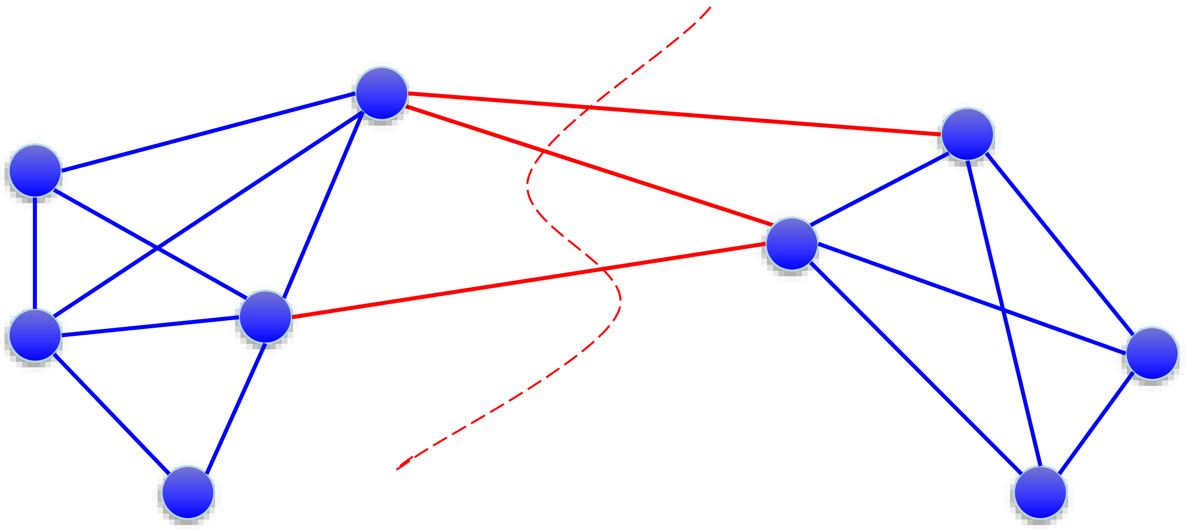
- **子图之间的切割：子图 A 与子图 B 的切割定义:**

$$cut(A,B) = W(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

注：如果两个顶点不相连，则权重为零。

图切割

- **最小二分切割 (Minimum bipartitional cut)**
  - 在所有的图切割中，找一个最小代价的切割，将图分为两个不连通的子图。也就是说，切开之后，两个子图之间的相似性要最小。



图切割

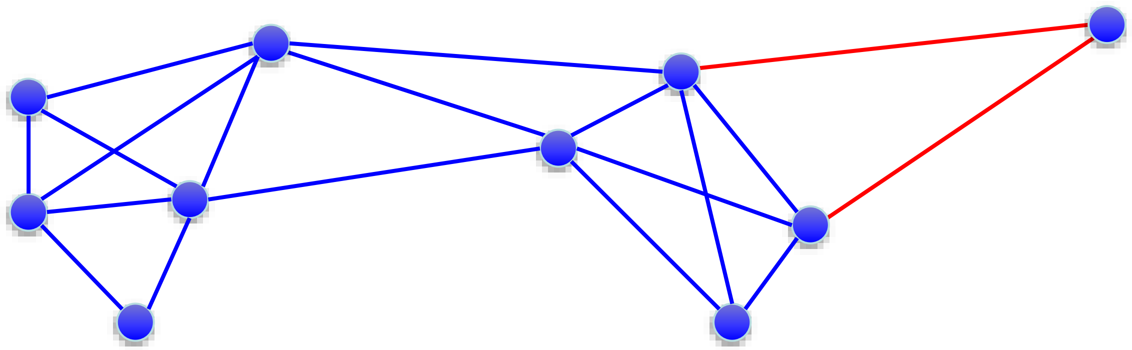
- **最小二分切割**
  - 在所有的图切割中，找一个最小代价的切割，将图分为两个不连通的子图。也就是说，切开之后，两个子图之间的相似性要最小。**最优化问题如下：**

$$\min_{A} \quad \underline{\mathrm{cut}(A,\bar{A}) := W(A,\bar{A}) = \sum_{i\in A,\, j\in\bar{A}} w_{ij}}$$

$$\begin{aligned} s.t. \quad & A \neq \varnothing, \\ & A \cap \bar{A} = \varnothing, \\ & A \cup \bar{A} = V \end{aligned}$$

图切割

- **最小二分切割**
  - **在实践中，上述目标函数通常将一个点(比如野点)从其余各点中分离出来**。从聚类的角度看，这并不是我们所期望的。



图切割

- **归一化最小二分切割**
  - 出现上述问题的原因在于**对子图的规模没有加以限制。**
  - 一个基本的假设是希望两个子图的**规模不要相差太大。**
  - 一个基本的做法是采用**子图的势或者体积**来对切割进行归一化，即定义如下目标函数：
    - 采用子图的势：

$$\mathrm{Ratiocut}(A,\bar{A}) := \frac{1}{2}\left(\frac{cut(A,\bar{A})}{|A|}+\frac{cut(A,\bar{A})}{|\bar{A}|}\right)$$

    - 采用子图的体积：

$$\mathrm{Ncut}(A,\bar{A}) := \frac{1}{2}\left(\frac{cut(A,\bar{A})}{vol(A)}+\frac{cut(A,\bar{A})}{vol(\bar{A})}\right)$$

图切割

- **K-切割 (k > 2)：**
  - 考虑将图分成 k 个子图：A_1，A_2，…，A_k。一种直观的方法是将图切割问题理解为多个二分切割问题的综合。
- **未归一化切割目标函数：**

$$\mathrm{cut}(A_1, A_2, \cdots, A_k) := \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i)$$

- **比例切割目标函数：**

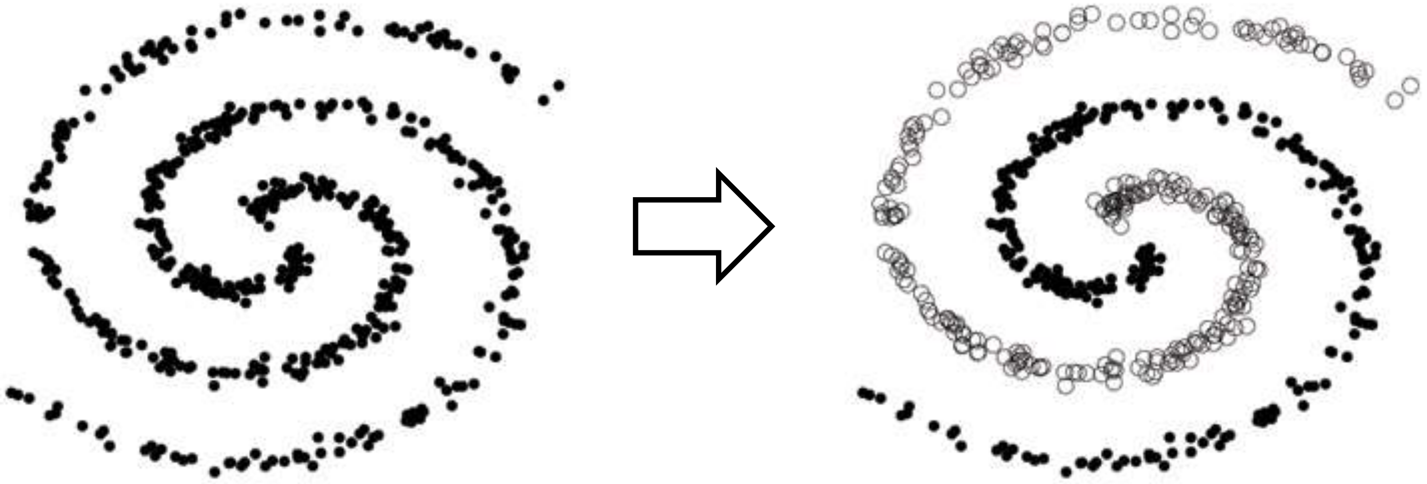
$$\mathrm{Ratiocut}(A_1, A_2, \cdots, A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \bar{A}_i)}{|A_i|}$$

- **归一化切割目标函数：**

$$\mathrm{Ncut}(A_1, A_2, \cdots, A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{\mathrm{vol}(A_i)} = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \bar{A}_i)}{\mathrm{vol}(A_i)}$$

Spectral Clustering

- Spectral Clustering
  - Simple, but powerful method of clustering
  - Requires less assumptions on the form of clusters
  - Outperforms the traditional approaches, such as *k*-means clustering.



Spectral Clustering

- Spectral Clustering?
  - Spectral analysis in linear algebra
    - Basic features of **matrices** : eigenpairs (eigenvalue, eigenvector)
    - Methods of using the eigenpairs to solve given problems

- Spectral Clustering!
  - Methods of **using the eigenvectors of some matrices** to find a partition of the data such that points in the same group are similar

Spectral Clustering

- Similarity Graphs
  - *ε*-neighborhood graph
    - Connect all points whose pairwise distances are smaller than *ε*.
  - *k*-nearest neighbor graph
    - Connect two points if one is among the *k*-nearest neighbors of the other (and vice versa for *mutual k*-nearest neighbor graph).
    - Each edge is weighted by the similarity of their endpoints.
  - fully connected graph
    - Connect all points and weight all edges by similarity of their endpoints.

Spectral Clustering

- In spectral clustering, the Gaussian similarity is used to represent local neighborhood relationships.

$$W_{ij} = \exp\left\{-\frac{\|v_i - v_j\|^2}{2\sigma^2}\right\} \quad (i \neq j)$$

- W : adjacency matrix of similarity graph
- D : degree matrix

$$D_{ij} = \begin{cases} \sum_k W_{ik} & i = j \\ 0 & i \neq j \end{cases}$$

$$\begin{aligned} x_1 &= (6,\ 8) \\ x_2 &= (3,\ 1) \\ x_3 &= (7,\ 9) \\ x_4 &= (3,\ 2) \\ x_5 &= (9,\ 8) \\ x_6 &= (2,\ 4) \end{aligned}$$

$$W = \begin{pmatrix} 0 & 0.04 & 0.89 & 0.08 & 0.61 & 0.17 \\ 0.04 & 0 & 0.01 & 0.95 & 0.01 & 0.57 \\ 0.89 & 0.01 & 0 & 0.03 & 0.76 & 0.06 \\ 0.08 & 0.95 & 0.03 & 0 & 0.02 & 0.76 \\ 0.61 & 0.01 & 0.76 & 0.02 & 0 & 0.03 \\ 0.17 & 0.57 & 0.06 & 0.76 & 0.03 & 0 \end{pmatrix}$$

$$(\sigma = 3)$$

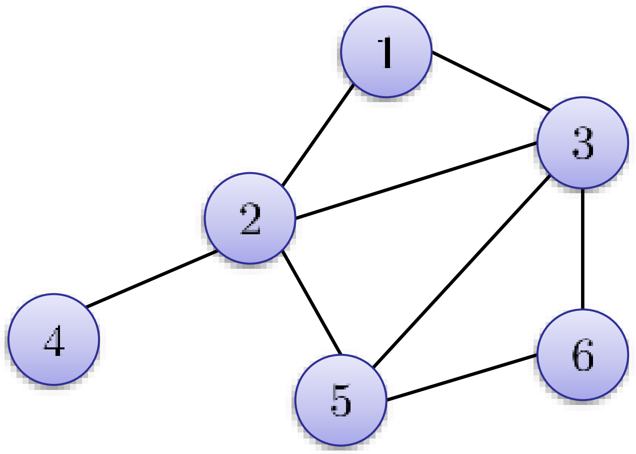
Spectral Clustering

- Graph Laplacian
  - unnormalized graph Laplacian : $L = D - W$
  - normalized graph Laplacian

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} W \longleftarrow \text{related to random walk}$$

  - Example



$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 \\ -1 & -1 & 4 & 0 & -1 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

Assume the weights of edges are 1.

Spectral Clustering Algorithm 1

- Unnormalized Spectral Clustering

1. Construct a similarity graph and compute the unnormalized graph Laplacian L.
2. Compute the k smallest eigenvectors $u_1, u_2, \cdots, u_k$ of L.
3. Let $U = [\, u_1 \; u_2 \; \cdots \; u_k \,] \in \mathbb{R}^{n \times k}$.
4. Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the ith row of U.

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{bmatrix} = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix}$$

5. Thinking of y_i's as points in $\mathbb{R}^k$, cluster them with k-means algorithms.

Spectral Clustering Algorithm 2

- Normalized Spectral Clustering [Shi2000]

1. Construct a similarity graph and compute the unnormalized graph Laplacian L.
2. Compute the k smallest **generalized** eigenvectors $u_1, u_2, \cdots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.
3. Let $U = [\, u_1\ u_2\ \cdots\ u_k \,] \in \mathbb{R}^{n \times k}$.
4. Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the ith row of U.

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1k} \\ u_{21} & u_{22} & \cdots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nk} \end{bmatrix} = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix}$$

5. Thinking of y_i's as points in $\mathbb{R}^k$, cluster them with k-means algorithms.

Spectral Clustering Algorithm 3

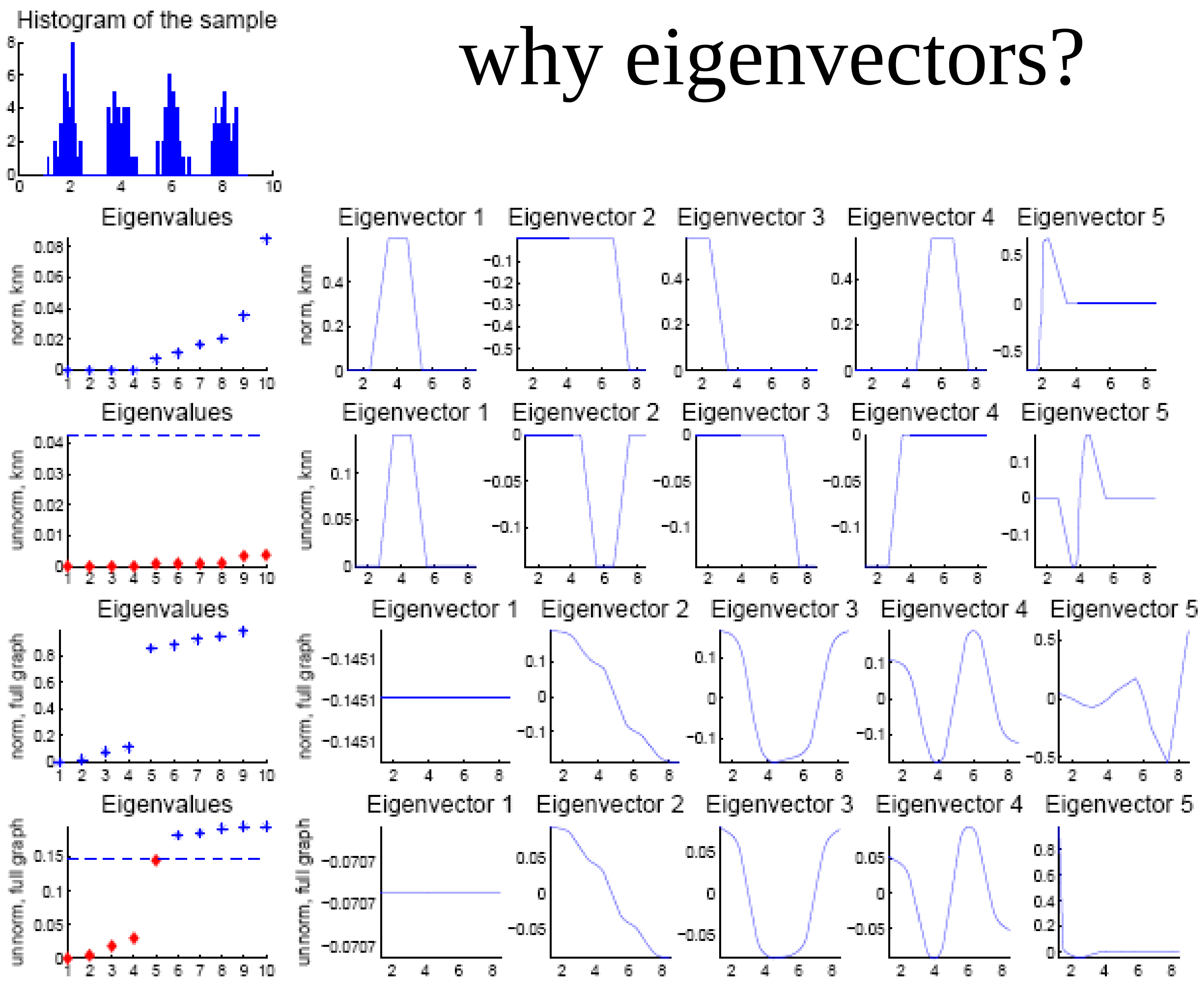
- Normalized Spectral Clustering [Ng2002]

1. Construct a similarity graph and compute the normalized graph Laplacian L_{sym}.
2. Compute the k smallest eigenvectors $u_1, u_2, \cdots, u_k$ of L_{sym}.
3. Let $U = [\, u_1 \; u_2 \; \cdots \; u_k \,] \in \mathbb{R}^{n \times k}$.
4. Normalized the rows of U to norm 1.

$$U_{ij} \leftarrow \frac{U_{ij}}{(\sum_k U_{ik}^2)^{1/2}}$$

5. Let $y_i \in \mathbb{R}^k$ be the vector corresponding to the ith row of U.
6. Thinking of y_i's as points in $\mathbb{R}^k$, cluster them with k-means algorithms.

why eigenvectors?



Simulated example: Graph of 3 connected components

$$W=\begin{bmatrix} 1&1&1&0&0&0&0&0\\ 1&1&1&0&0&0&0&0\\ 1&1&1&0&0&0&0&0\\ 0&0&0&1&1&1&0&0\\ 0&0&0&1&1&1&0&0\\ 0&0&0&1&1&1&0&0\\ 0&0&0&0&0&0&1&1\\ 0&0&0&0&0&0&1&1 \end{bmatrix}$$

```
Eigenvalues
[3.0  3.0  3.0  3.0  2.0  0.0  0.0 -0.0]

Eigenvectors
 0.0  0.0  0.0  0.8 -0.0  0.0 -0.6  0.0
 0.0  0.0  0.7 -0.4 -0.0  0.0 -0.6  0.0
 0.0  0.0 -0.7 -0.4 -0.0  0.0 -0.6  0.0
 0.0  0.8  0.0  0.0 -0.0 -0.6  0.0  0.0
 0.7 -0.4  0.0  0.0 -0.0 -0.6  0.0  0.0
-0.7 -0.4  0.0  0.0 -0.0 -0.6  0.0  0.0
 0.0  0.0  0.0  0.0  0.7  0.0  0.0  0.7
 0.0  0.0  0.0  0.0 -0.7  0.0  0.0  0.7
```

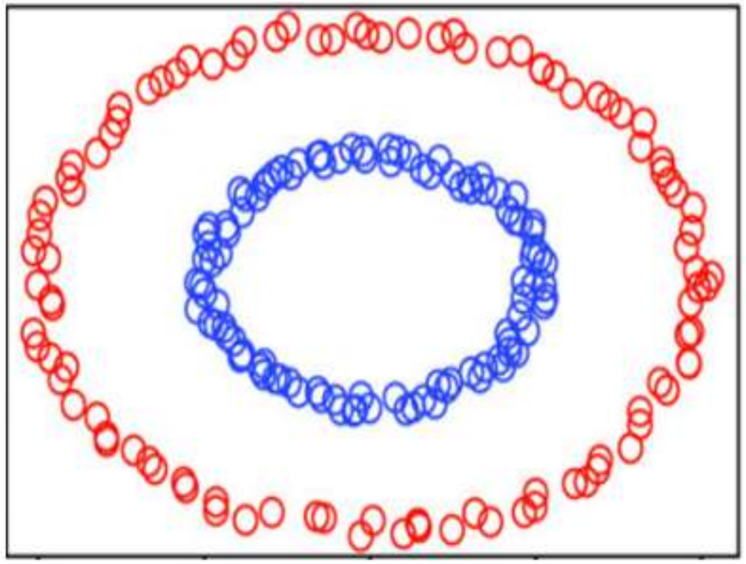
$$W=\begin{bmatrix} 1&1&0&0&0&1&0&0\\ 1&1&0&0&0&1&0&0\\ 0&0&1&0&1&0&0&1\\ 0&0&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1\\ 1&1&0&0&0&1&0&0\\ 0&0&0&1&0&0&1&0\\ 0&0&1&0&1&0&0&1 \end{bmatrix}$$

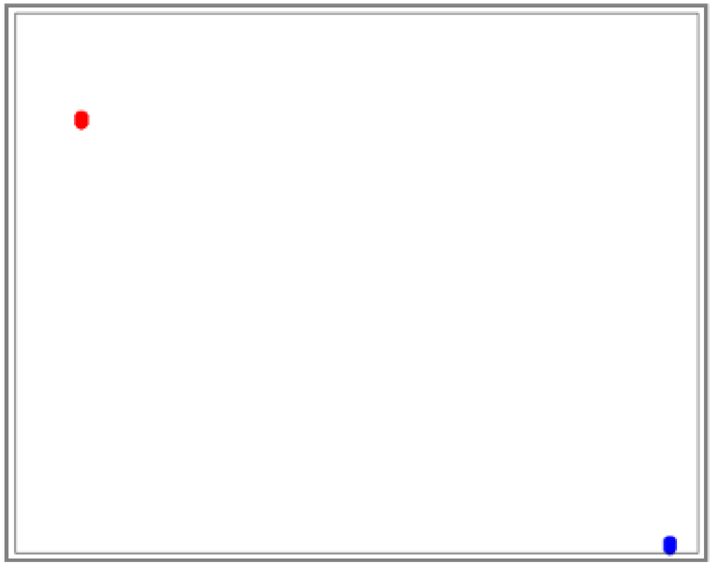
```
Eigenvalues
[3.0  3.0  3.0  3.0  2.0  0.0  0.0  0.0]

Eigenvectors
 0.8  0.0  0.0  0.0  0.0  0.0 -0.0 -0.6
-0.4  0.0 -0.0 -0.7  0.0 -0.0 -0.0 -0.6
 0.0 -0.0  0.8  0.0 -0.0  0.0  0.6  0.0
 0.0 -0.0 -0.0  0.0 -0.7  0.7  0.0  0.0
 0.0 -0.7 -0.4  0.0  0.0  0.0  0.6  0.0
-0.4 -0.0  0.0  0.7 -0.0  0.0  0.0 -0.6
 0.0  0.0  0.0  0.0  0.7  0.7 -0.0  0.0
 0.0  0.7 -0.4  0.0 -0.0 -0.0  0.6  0.0
```

中国科学院大学
University of Chinese Academy of Sciences

Represents the data using K eigenvectors and obtains the clusters





Two Explanations

- Graph partitioning point of view
  - Based on mincut problem in a similarity graph
  - Find a partition such that the edges between clusters have a very low weight and the edges within a cluster have high weight.

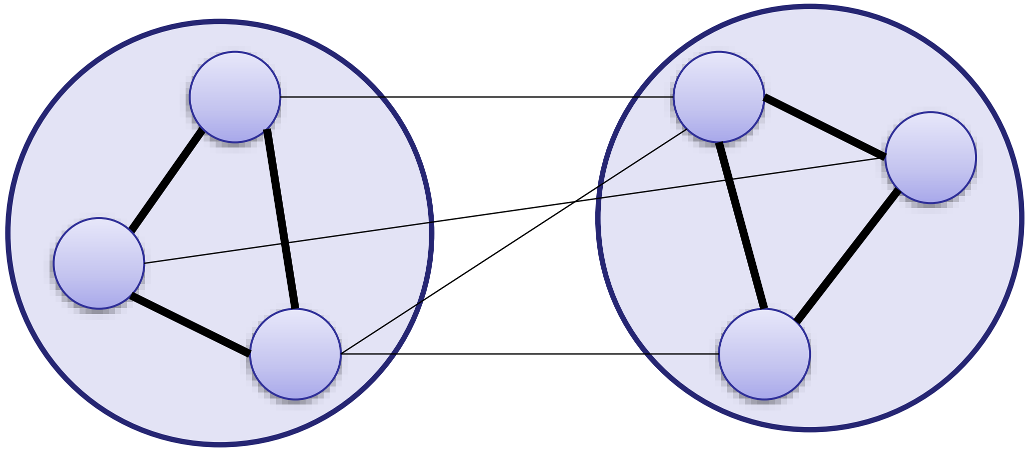
- Random walks point of view
  - Based on random walks on the similarity graph
  - Find a partition such that random walk stays long within the same cluster and seldom jumps to other clusters.

Graph Partitioning Point of View

- Mincut problem
  - Given a number *k*, find a partition $A_1, \cdots, A_k$ which minimizes

$$\mathrm{cut}(A_1, \cdots, A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \overline{A_i})$$

$$W(A_i, \overline{A_i}) = \sum_{p\in A_i,\, q\in \overline{A_i}} W_{pq}$$



- In practice, the mincut problem results in a size imbalance in the partition.

Graph Partitioning Point of View

- Two common objective functions

$$\mathrm{RatioCut}(A_1, \cdots, A_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, \overline{A_i})}{|A_i|} = \sum_{i=1}^{k}\frac{\mathrm{cut}(A_i, \overline{A_i})}{|A_i|}$$

$$\mathrm{NCut}(A_1, \cdots, A_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(A_i, \overline{A_i})}{\mathrm{vol}(A_i)} = \sum_{i=1}^{k}\frac{\mathrm{cut}(A_i, \overline{A_i})}{\mathrm{vol}(A_i)}$$

$$\mathrm{vol}(A_i) = \sum_{k\in A_i} D_{kk}$$

- Normalized mincut problem
  - Given a number *k*, find a partition $A_1, \cdots, A_k$ which minimizes

$\mathrm{RatioCut}$ or NCut

Graph Partitioning Point of View

- Represent partitions by *k* indicator vectors $f_1, \cdots, f_k$.

$$f_{ij} = \begin{cases} 1 & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

- Relationship between mincut problem and graph Laplacian

$$f_i^T L f_i = \mathrm{cut}(A_i, \overline{A_i}) \quad \Rightarrow$$

$$\mathrm{cut}(A_1, \cdots, A_k) = \sum_{i=1}^{k} \mathrm{cut}(A_i, \overline{A_i}) = \sum_{i=1}^{k} f_i^T L f_i = \mathrm{tr}(F^T L F)$$

$$F = [\, f_1 \; f_2 \; \cdots \; f_k \,] \in \mathbb{R}^{n \times k}$$

  – Mincut problem is converted into

$$\underset{F \in \mathbb{R}^{n\times k}}{\arg\min}\, \mathrm{tr}(F^T L F)$$
$$\text{subject to } f_{ij} \in \{0, 1\},\ f_i^T f_j = 0 \text{ if } i \neq j$$

Graph Partitioning Point of View

Defined as

$$L = D - W$$

Key property: for all $f \in \mathbb{R}^n$

$$\begin{aligned}
f'Lf &= f'Df - f'Wf \\
&= \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij} \\
&= \frac{1}{2}\left(\sum_i (\sum_j w_{ij}) f_i^2 - 2\sum_{ij} f_i f_j w_{ij} + \sum_j (\sum_i w_{ij}) f_j^2\right) \\
&= \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2
\end{aligned}$$

Graph Partitioning Point of View

- Relaxing the form of indicator vectors, normalized mincut problems can be expressed with graph Laplacian.

$$f_{ij} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{RatioCut Problem} \quad \Rightarrow \quad \begin{array}{l} \underset{F\in\mathbb{R}^{n\times k}}{\arg\min}\, \operatorname{tr}(F^T L F) \\ \text{subject to } F^T F = I,\ f_i\text{'s are indicator vectors} \end{array}$$

$$f_{ij} = \begin{cases} 1/\sqrt{\operatorname{vol}(A_i)} & \text{if } v_i \in A_j \\ 0 & \text{otherwise} \end{cases}$$

$$\text{NCut Problem} \quad \Rightarrow \quad \begin{array}{l} \underset{F\in\mathbb{R}^{n\times k}}{\arg\min}\, \operatorname{tr}(F^T L F) \\ \text{subject to } F^T D F = I,\ f_i\text{'s are indicator vectors} \end{array}$$

Graph Partitioning Point of View

- The optimization problems are NP-hard.
  - Relax the discreteness condition of vectors $f_1, \cdots, f_k$.

$$\text{RatioCut Problem} \Rightarrow \begin{array}{l} \arg\min_{F \in \mathbb{R}^{n \times k}} \operatorname{tr}(F^T L F) \\ \text{subject to } F^T F = I \end{array}$$

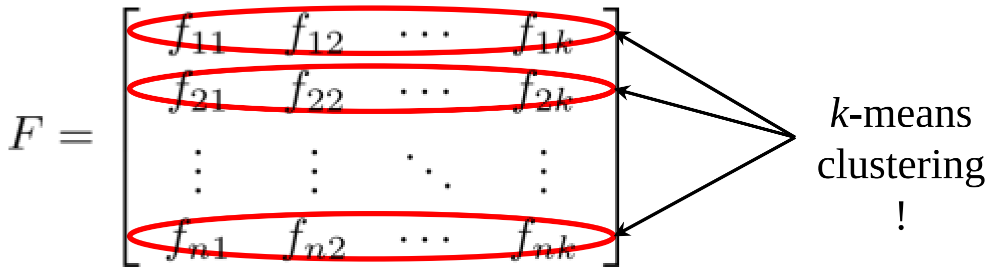
$$\text{NCut Problem} \Rightarrow \begin{array}{l} \arg\min_{F \in \mathbb{R}^{n \times k}} \operatorname{tr}(F^T L F) \\ \text{subject to } F^T D F = I \end{array}$$

- By the Rayleigh-Ritz theorem, the solutions of above problems are the matrix that contains the *k* smallest eigenvectors of the graph Laplacian L and normalized one L_{sym}, repectively.

University of Chinese Academy of Sciences

Graph Partitioning Point of View

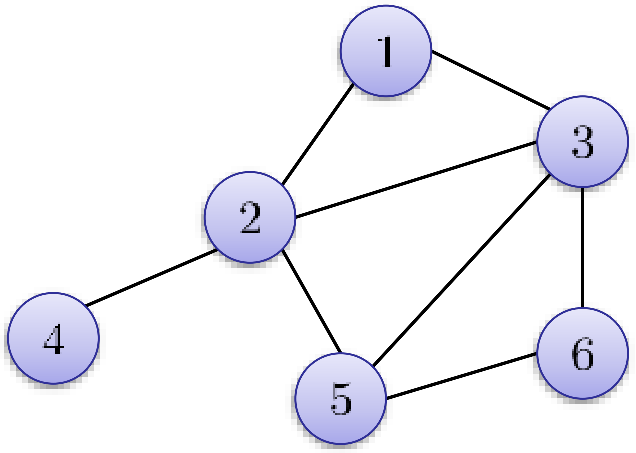
- Note that the solutions of the relaxed optimization problems does NOT indicate which nodes are included in which groups!
- However, we hope that if the data are *well-separate*, the eigenvectors of graph Laplacians are close to piecewise constant (and close to indicator vectors).
- Thinking of the rows of the solution matrices as another representation of data points, *k*-mean clustering is a way of finding an appropriate group for each point.


$F = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1k} \\ f_{21} & f_{22} & \cdots & f_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & \cdots & f_{nk} \end{bmatrix}$
k-means clustering !

Random Walks Point of View

- Tradition Probability Matrix

$$P_{ij} = \frac{W_{ij}}{\sum_k W_{ik}} \quad \Rightarrow \quad P = D^{-1}W$$



$$P = \begin{pmatrix} 0 & 1/4 & 1/4 & 0 & 0 & 0 \\ 1/2 & 0 & 1/4 & 1 & 1/3 & 0 \\ 1/2 & 1/4 & 0 & 0 & 1/3 & 1/2 \\ 0 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 1/4 & 0 & 0 & 1/2 \\ 0 & 0 & 1/4 & 0 & 1/3 & 0 \end{pmatrix}$$

Assume the weights of edges are 1.

- If the graph is connected and non-bipartite, then the random walk always possesses a unique stationary distribution

$$\pi = [\,\pi_1, \cdots, \pi_n\,]^T$$

Random Walks Point of View

- Relationship between *Ncut* problem and random walks
  - For the random walk X_0 in the stationary distribution,

$$P(B \mid A) = P(X_1 \in B \mid X_0 \in A) \quad (A \cap B = \phi)$$

  - The problem of finding a partition such that random walk does not have many opportunities to jump between clusters is equivalent to *Ncut* problem due to:

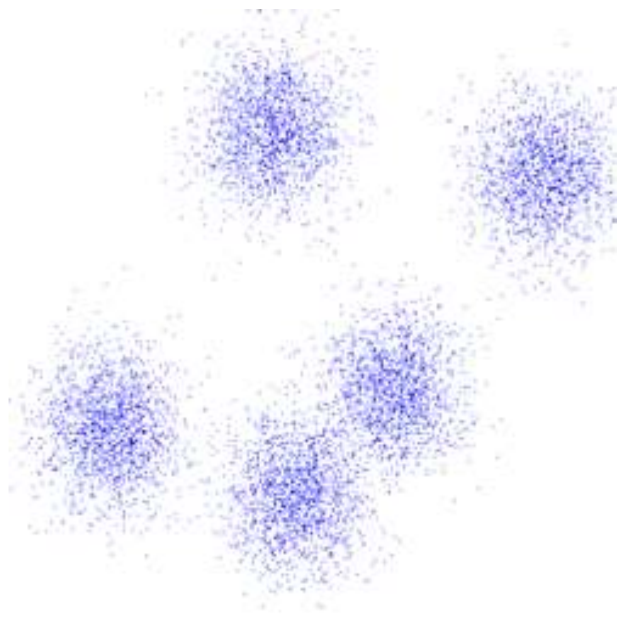
$$\mathrm{NCut}(A, \overline{A}) = P(\overline{A} \mid A) + P(A \mid \overline{A})$$

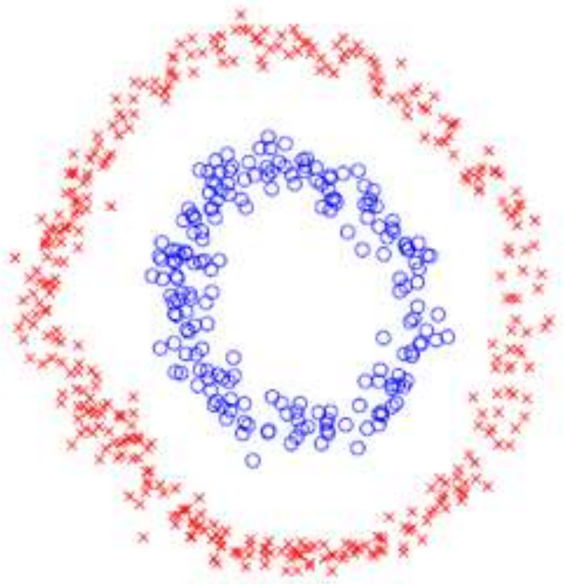
- Relationship between L_{rw} and P

$$Lx = \lambda Dx \quad \Leftrightarrow \quad Px = (1-\lambda)x$$

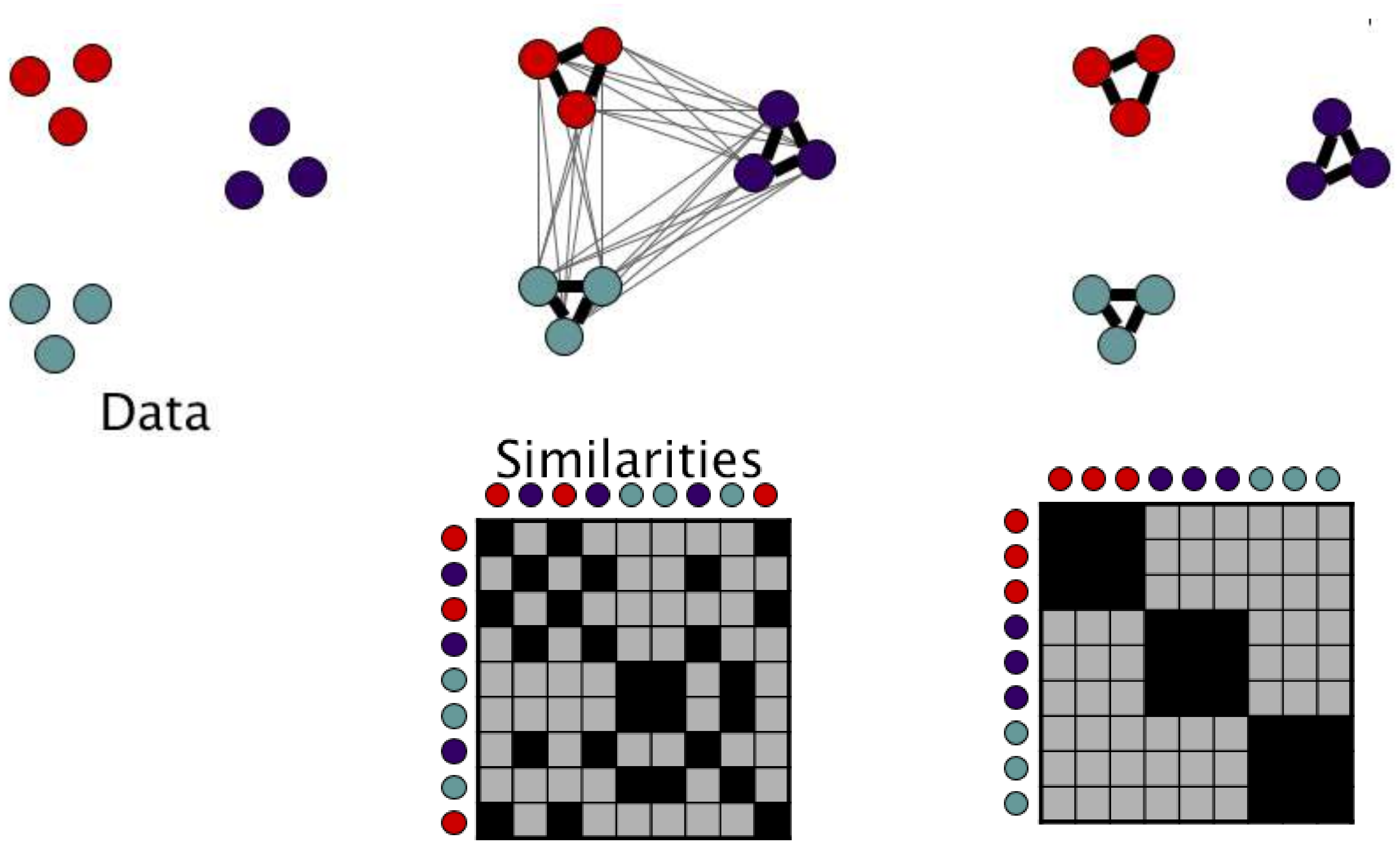
Spectral v.s. *K*-means

- Two different criteria
  - Compactness, e.g., k-means, mixture models
  - Connectivity, e.g., spectral clustering

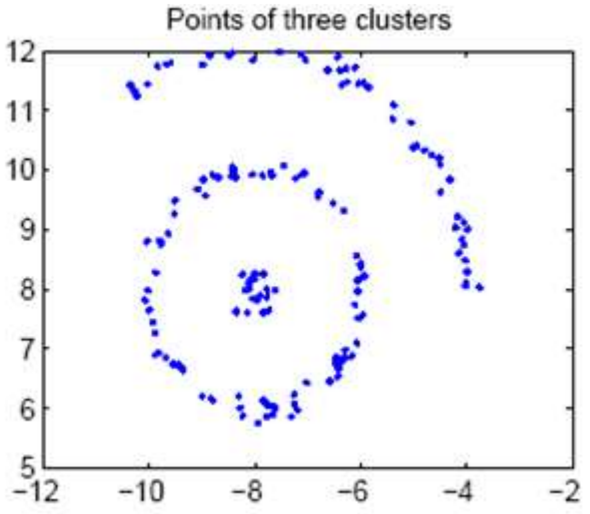

Compactness

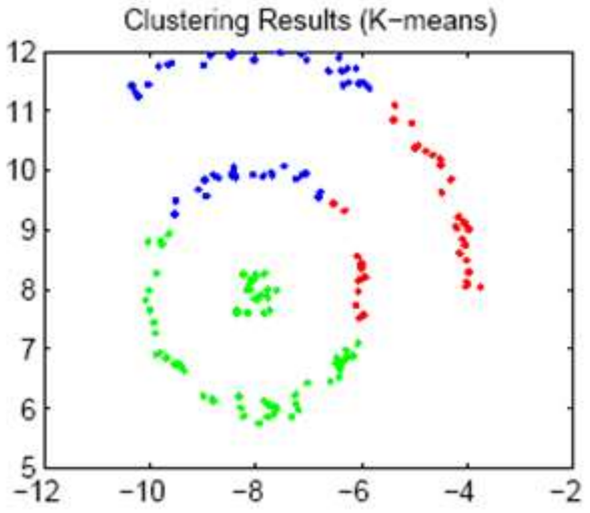

Connectivity

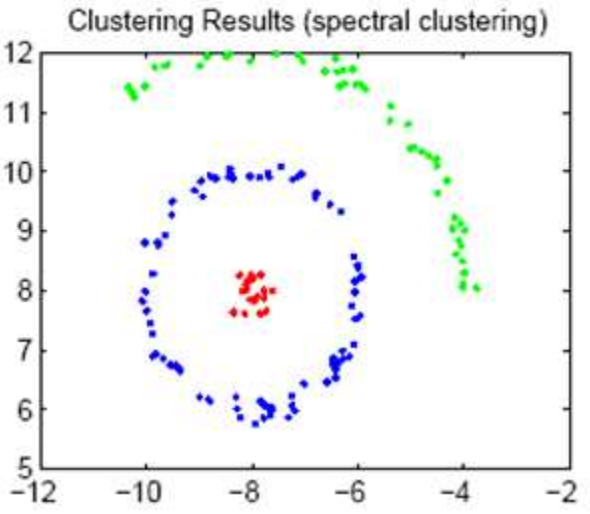
Spectral Clustering



Spectral v.s. *K*-means







Spectral v.s. *K*-means

从优化目标的角度来看：

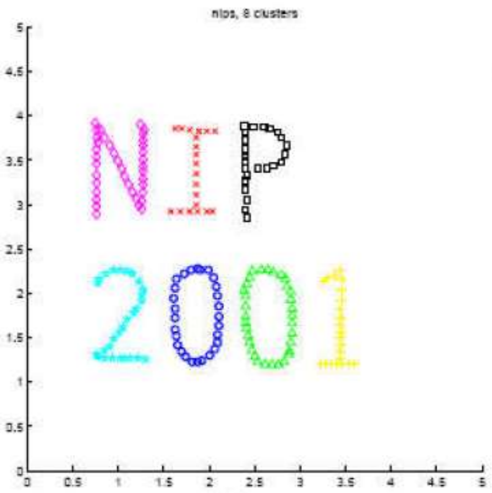
1. K-means致力于最小化类内的平均距离
2. 谱聚类致力于最大化类内的平均相似度

从图的观点来看：

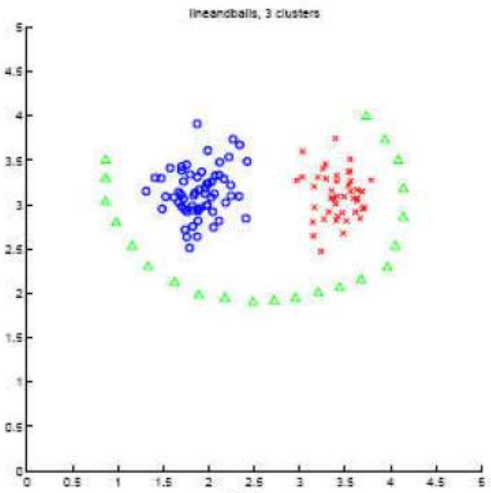
local v.s. global

1. K-means采用的是全图
   即任意两个样本之间的距离都没有被忽略
2. 谱聚类采用的是近邻图
   一般只计算每个样本与其近邻之间的相似度

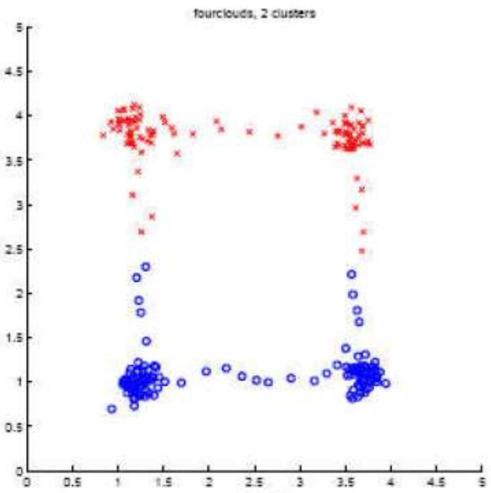
Toy examples



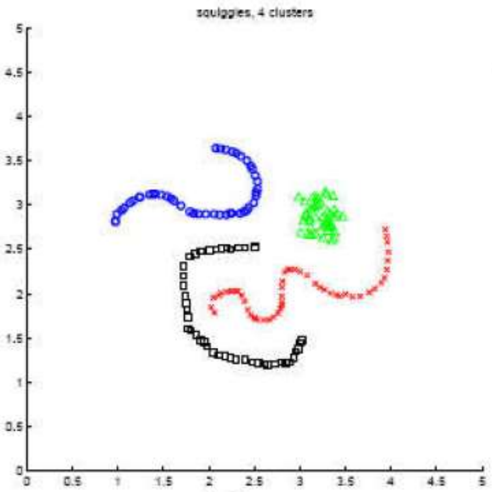
(a)



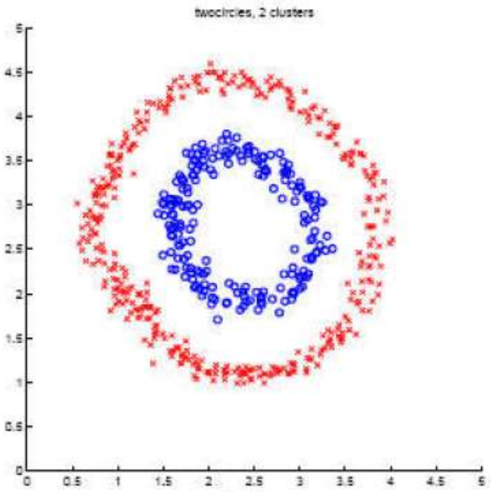
(b)



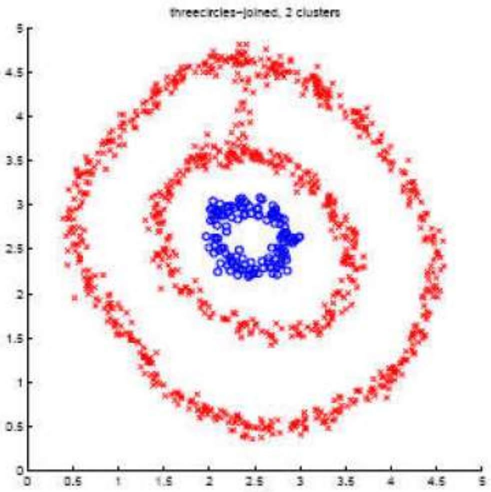
(c)



(d)



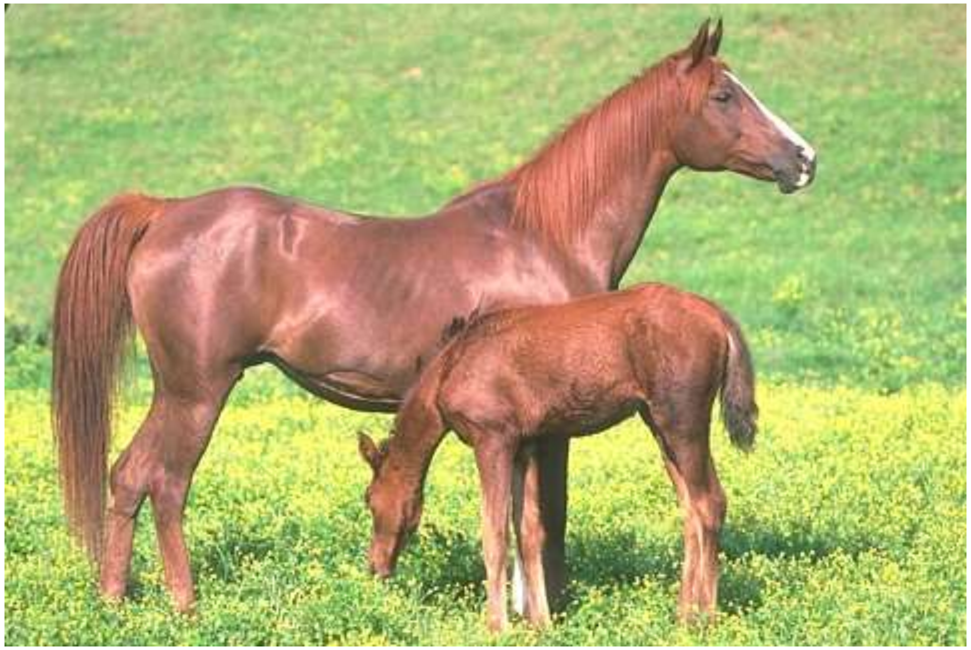
(e)



(f)

谱聚类应用

- 采用谱聚类将图像的前景目标分割出来

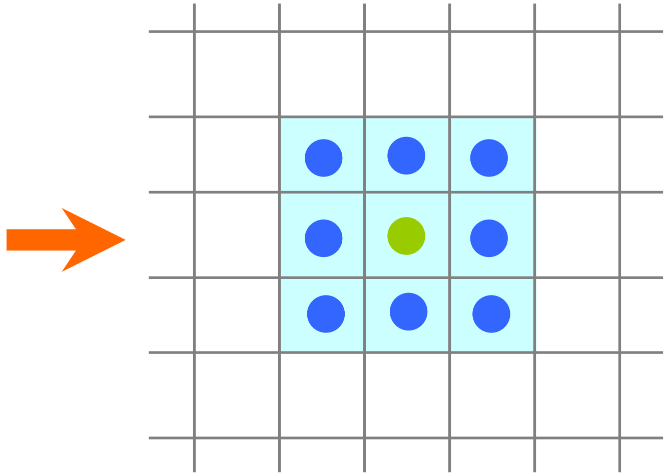


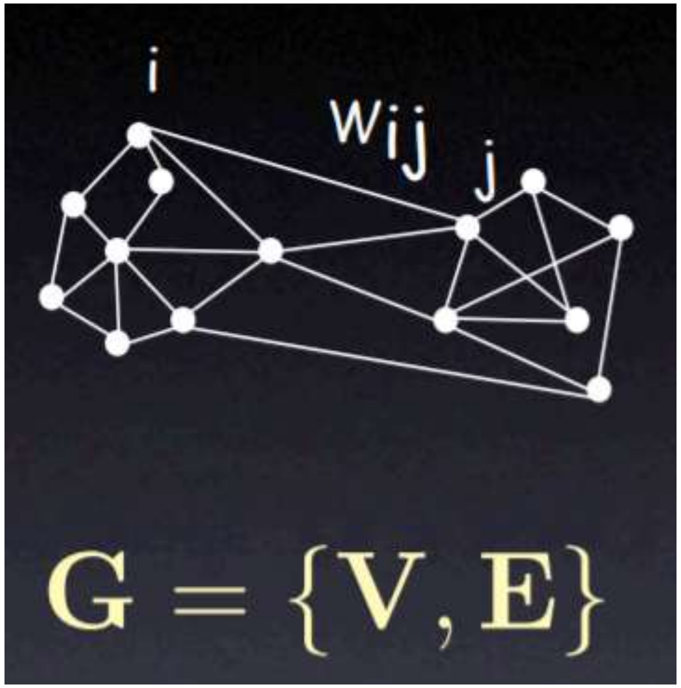
如何构造近邻图？

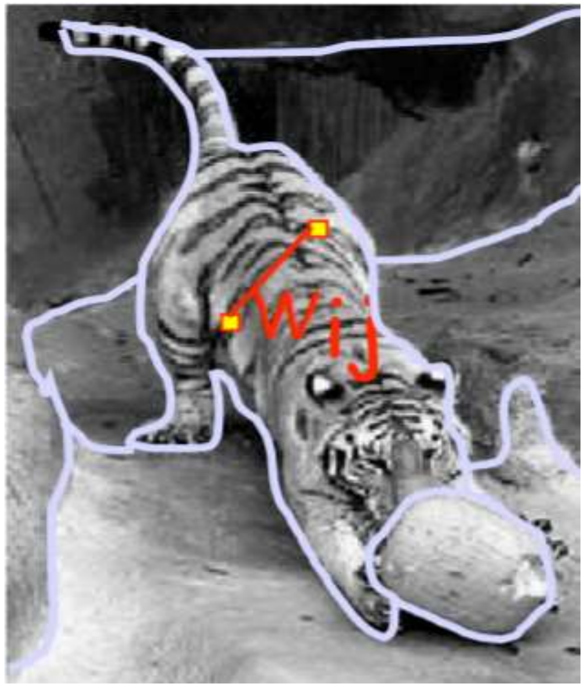
谱聚类应用

- 采用谱聚类将图像的前景目标分割出来
  - 以每个像素为一个顶点，以3X3为一个基本邻域，连接各像素，构建一个图。

图像


格子图

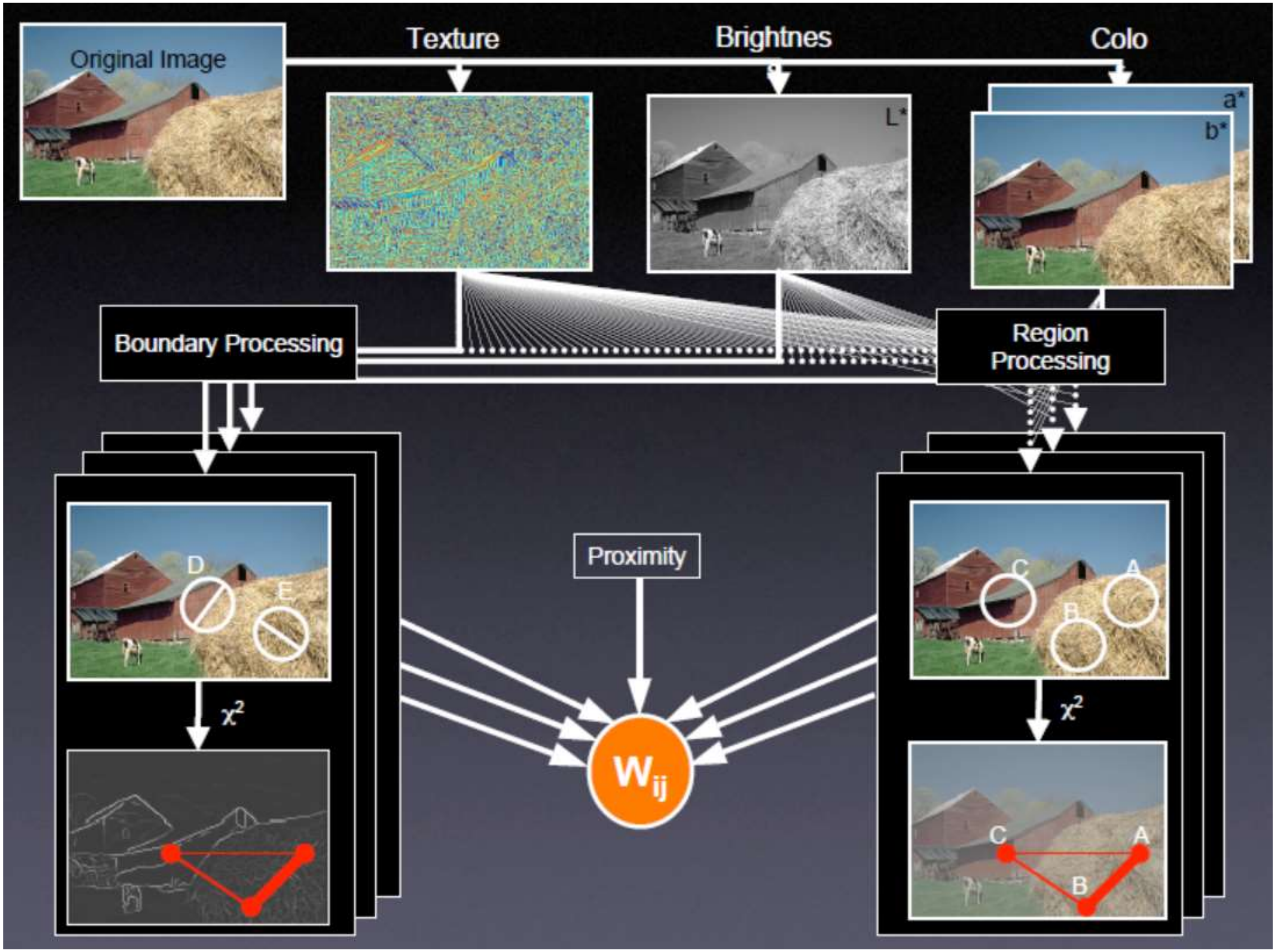


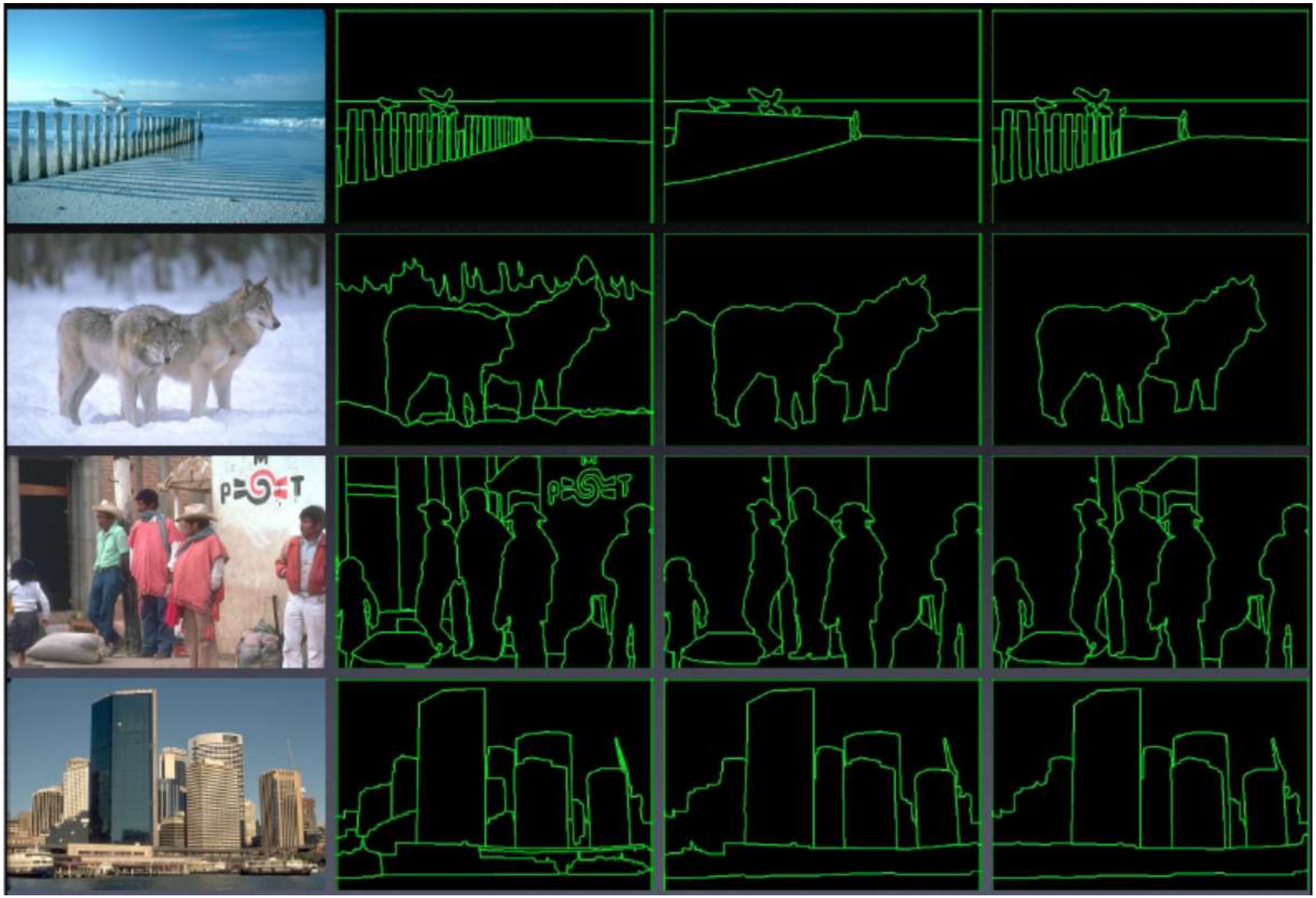


V: graph nodes
E: edges connection nodes

↔

Image = { pixels }
Pixel similarity


Original Image
Texture
Brightnes
Colo
L*
a*
b*
Boundary Processing
Region Processing
D
E
C
A
B
χ^2
Proximity
W_{ij}


P S T

存在的问题

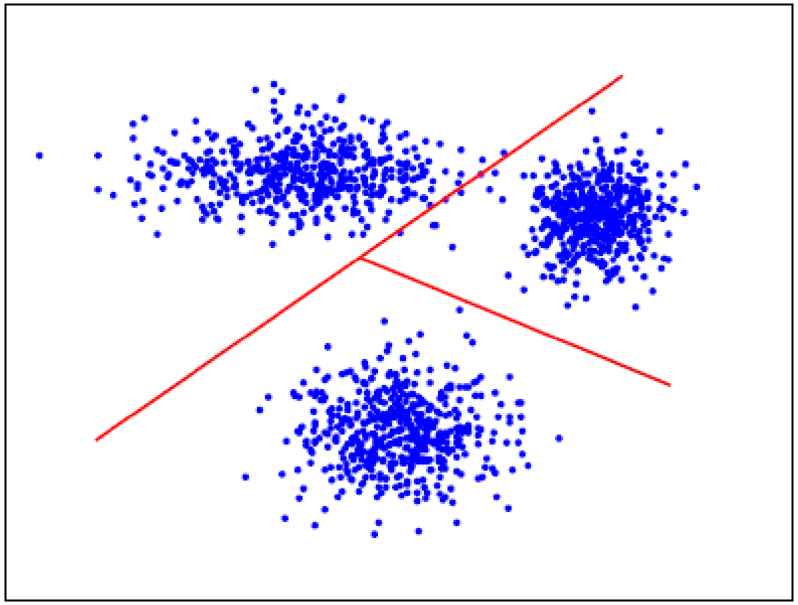
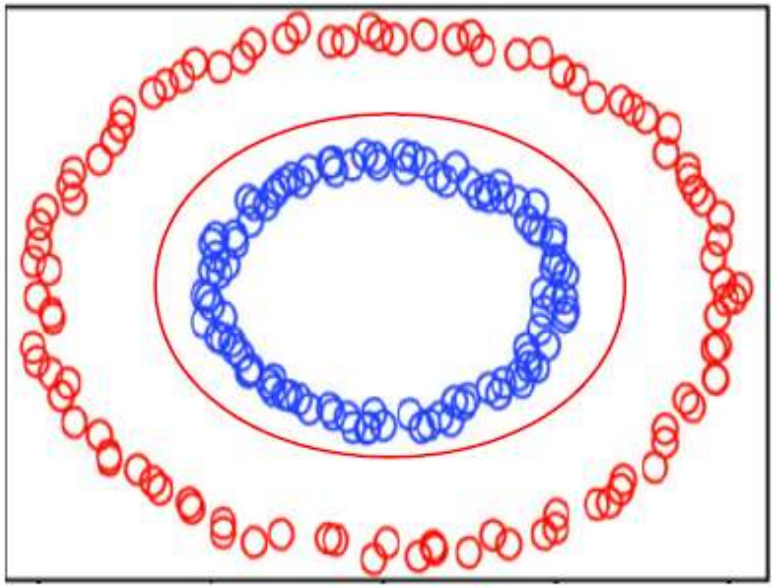
- **算法细节**
  - 核心问题是图构造
    - **局部连接 k 近邻 (ε-半径) 取多大?**
    - **点对权值如何计算?**
  - 特征值分解问题：对于超大型矩阵，计算仍然不稳定，可能会引起结果很差。
    - To handle large data sets
    - Spectral clustering on reduced set selected by sampling or quantization
  - 最后采用 K-means 聚类问题，也可能会影响聚类结果。
  - 当然，聚类数目的多少是一个open problem。
    - Eigenvalue gap: choose the number k such that the gap between λ_k and λ_{k+1} is big

Kernel Clustering

核聚类

→ *K*-means in kernel space

Motivation

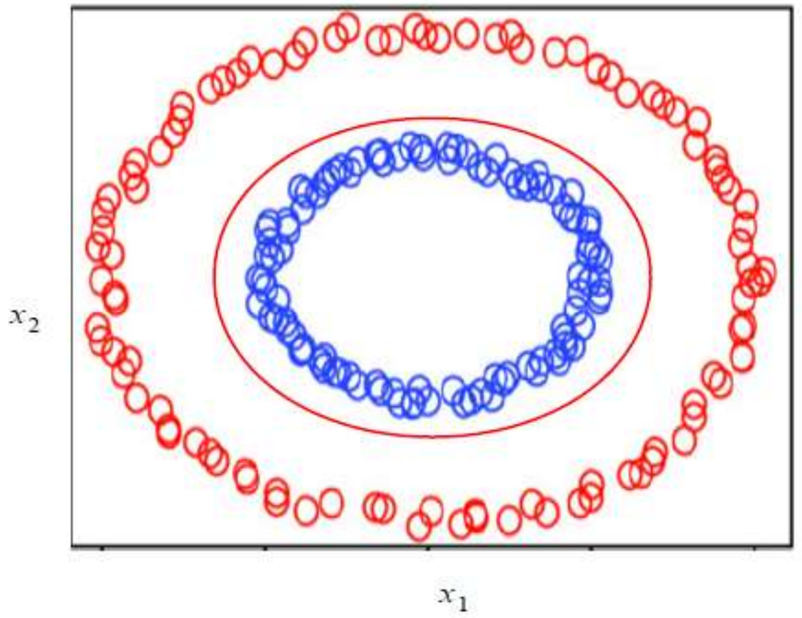



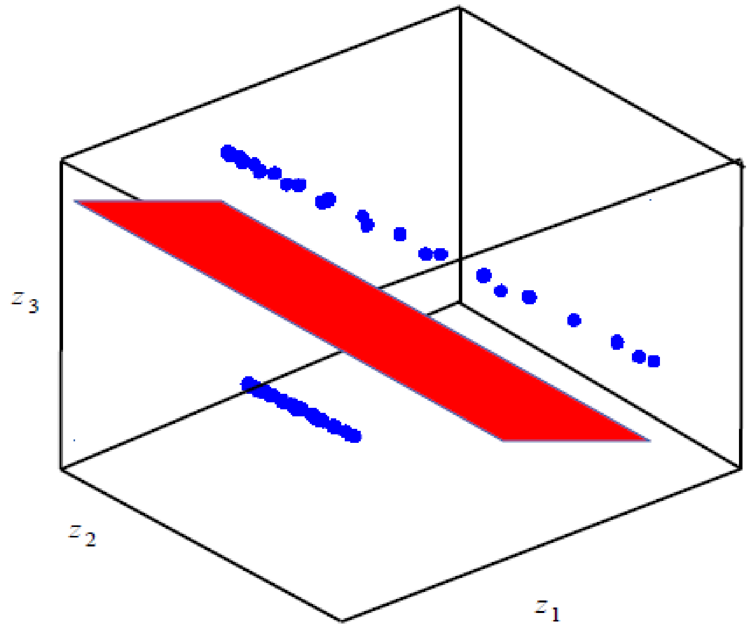
Euclidean distance assumes unit covariance and linear separability

中国科学院大学 University of Chinese Academy of Sciences

Motivation

Any data set is linearly separable in sufficiently high dimensional space





$$\varphi : X \to H$$

Polynomial $\varphi(x) = \left(x_1^2, \sqrt{2}\, x_1 x_2, x_2^2\right)$

What is kernel?

Euclidean distance in H

$$\|\varphi(x)-\varphi(y)\|_2^2=\underbrace{\varphi(x)^T\varphi(x)}-2\,\underbrace{\varphi(x)^T\varphi(y)}+\underbrace{\varphi(y)^T\varphi(y)}$$

Polynomial map

$$\varphi([x_1,x_2])=(x_1^2,\sqrt{2}x_1x_2,x_2^2)$$

$$\varphi(x)^T\varphi(y)=\begin{pmatrix}x_1^2 & \sqrt{2}x_1x_2 & x_2^2\end{pmatrix}\begin{pmatrix}y_1^2\\ \sqrt{2}y_1y_2\\ y_2^2\end{pmatrix}=\boxed{(x^Ty)^2}$$

$\Downarrow$

$$\boxed{\kappa(x,y)=\varphi(x)^T\varphi(y)}$$

Kernel function

$\kappa(x,y)$

Different kernels

Polynomial (Linear)

$$\kappa(x, y) = (x^T y)^p$$

Gaussian

$$\kappa(x, y) = \exp(-\lambda \|x - y\|_2^2)$$

Chi-square

$$\kappa(x, y) = 1 - \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{0.5(x_i + y_i)}$$

Histogram intersection

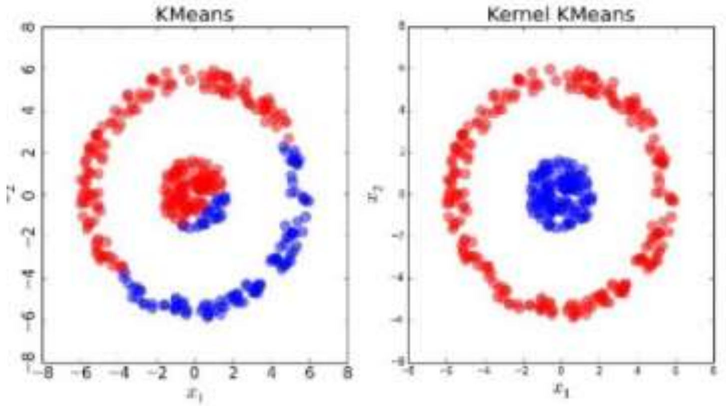
$$\kappa(x, y) = \sum_{i=1}^{m} \min(\text{hist}(x), \text{hist}(y))$$

Kernel K-means

- Basic idea: Replace the Euclidean distance/similarity computations in K-means by the kernelized versions. E.g., $d(\boldsymbol{x}_n, \boldsymbol{\mu}_k) = ||\phi(\boldsymbol{x}_n) - \phi(\boldsymbol{\mu}_k)||$ by

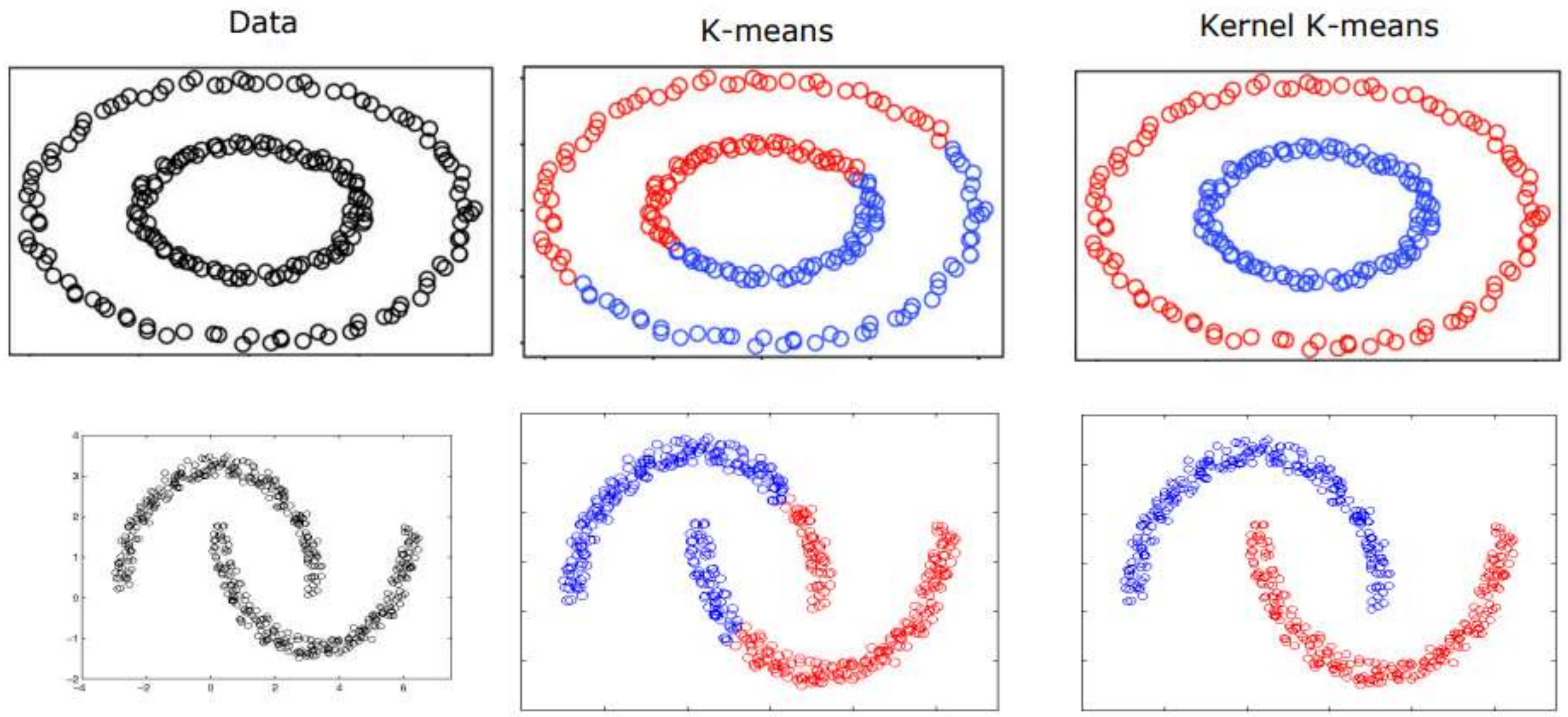
$$\begin{aligned} ||\phi(\boldsymbol{x}_n) - \phi(\boldsymbol{\mu}_k)||^2 &= ||\phi(\boldsymbol{x}_n)||^2 + ||\phi(\boldsymbol{\mu}_k)||^2 - 2\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{\mu}_k) \\ &= k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{\mu}_k, \boldsymbol{\mu}_k) - 2k(\boldsymbol{x}_n, \boldsymbol{\mu}_k) \end{aligned}$$

- Here $k(.,.)$ denotes the kernel function and ϕ is its (implicit) feature map
- Note: ϕ doesn't have to be computed/stored for data $\{\boldsymbol{x}_n\}_{n=1}^N$ or the cluster means $\{\boldsymbol{\mu}_k\}_{k=1}^K$ because computations only depend on kernel evaluations



- A small technical note: When computing $k(\boldsymbol{\mu}_k, \boldsymbol{\mu}_k)$ and $k(\boldsymbol{x}_n, \boldsymbol{\mu}_k)$, remember that $\phi(\boldsymbol{\mu}_k)$ is the average of ϕ's the data points assigned to cluster k

K-means v.s. Kernel *K*-means



Kernel K-means is able to find "complex" clusters.

Kernel *K*-means Challenges

Scalability

- $O(n^2)$ complexity to calculate kernel
- More expensive than K-means

Out-of-sample clustering

- No explicit representation for the centers.
- Expensive to assign a new point to a cluster.

Kernel selection

- The best kernel is application and data-dependent.
- Wrong kernel can lead to results worse than K-means

Scalability

No. of Objects (n)	No. of operations	
	K-means	Kernel K-means
	O(nCd)	O(n^2C)
1M	10^{11} (3.2)	10^{15}
10M	10^{12} (34.9)	10^{17}
100M	10^{13}(5508.5)	10^{19}
1B	10^{14}	10^{21}

d = 100; C=10

No. of Objects (n)	No. of operations	
	K-means	Kernel K-means
	O(nCd)	O(n^2C)
1M	10^{13} (6412.9)	10^{16}
10M	10^{14}	10^{18}
100M	10^{15}	10^{20}
1B	10^{16}	10^{22}

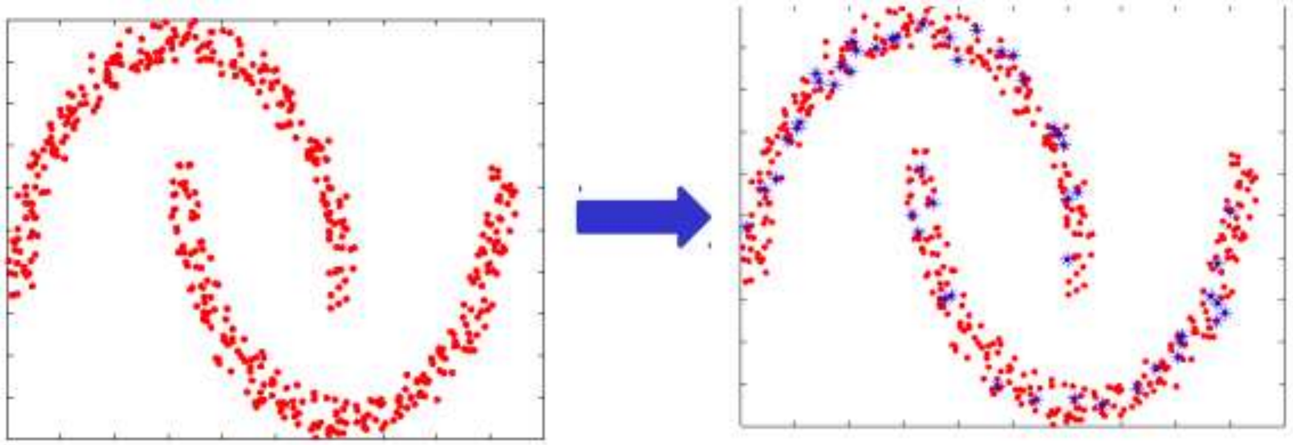
d = 10,000; C=10

* Runtime in seconds on Intel Xeon 2.8 GHz processor using 12 GB memory

Sampling based approximations

- Kernel matrix approximation
- Non-linear Random Projections

Approx. Kernel *K*-means



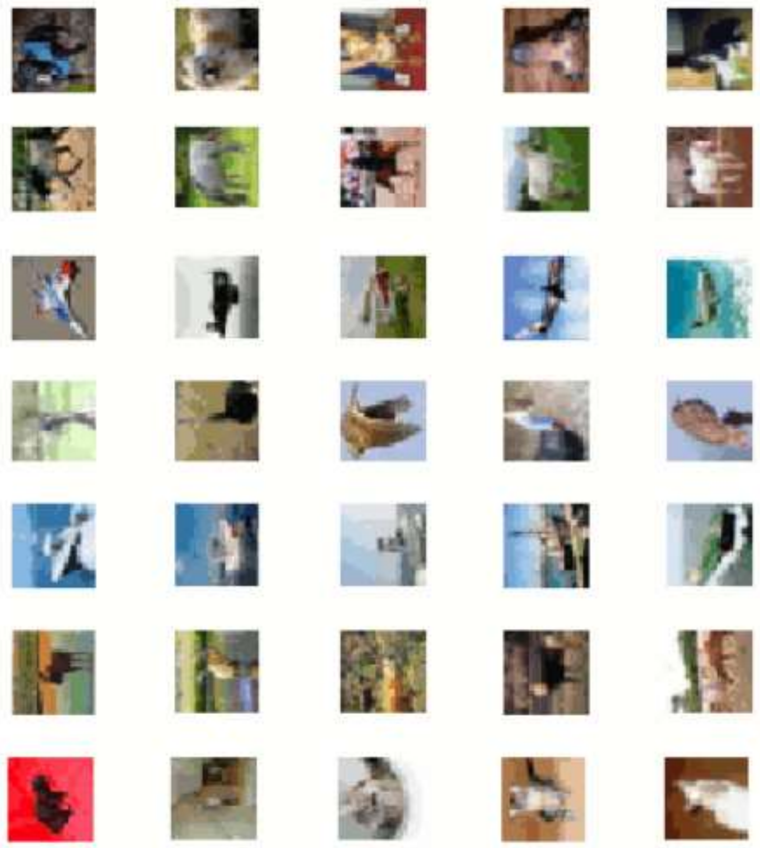
Randomly sample m points $\{y_1, y_2, \ldots y_m\}, m \ll n$ and compute the kernel similarity matrices

K_A (m x m) and K_B (n x m)

$$(K_A)_{ij} = \varphi(y_i)^T \varphi(y_j) \qquad (K_B)_{ij} = \varphi(x_i)^T \varphi(y_j)$$

Equivalent to running K-means on $K_B K_A^{-1} K_B^T$ Nystrom approximation

Clustering 80M Tiny Images


Example Clusters

Average clustering time into 100 clusters	
Approximate kernel K-means (m=1,000)	8.5 hours
K-means	6 hours

2.4 Ghz, 150 GB memory

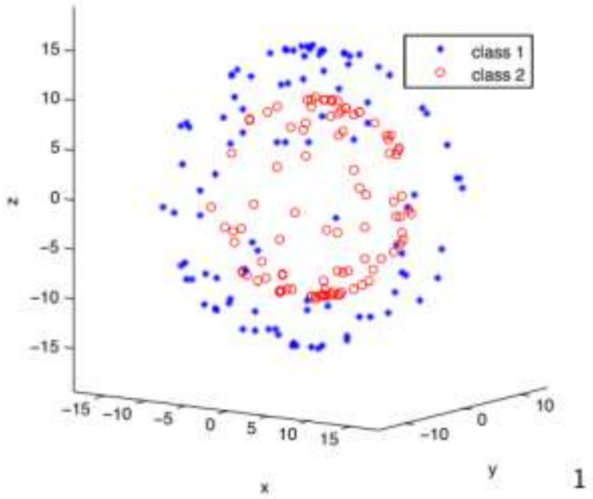
Clustering accuracy on CIFAR-10	
Kernel K-means	29.94
Approximate kernel K-means (m = 5,000)	29.76
Nystrom approximation based spectral clustering**	27.09
K-means	26.70

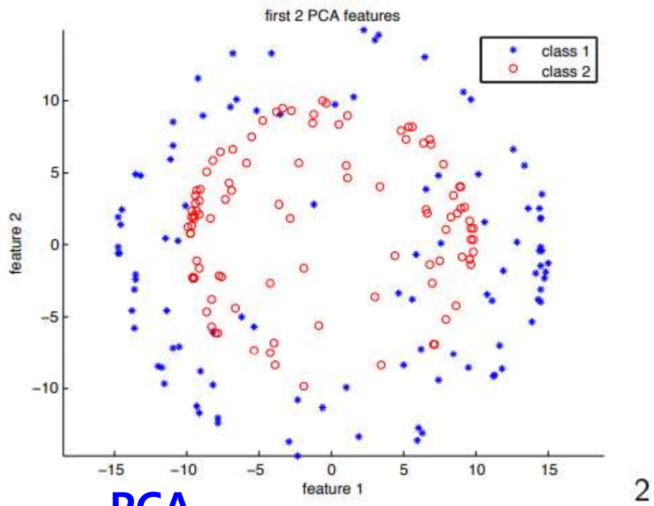
Kernel PCA + *K*-means

PCA: Eigen decomposition of **covariance** matrix

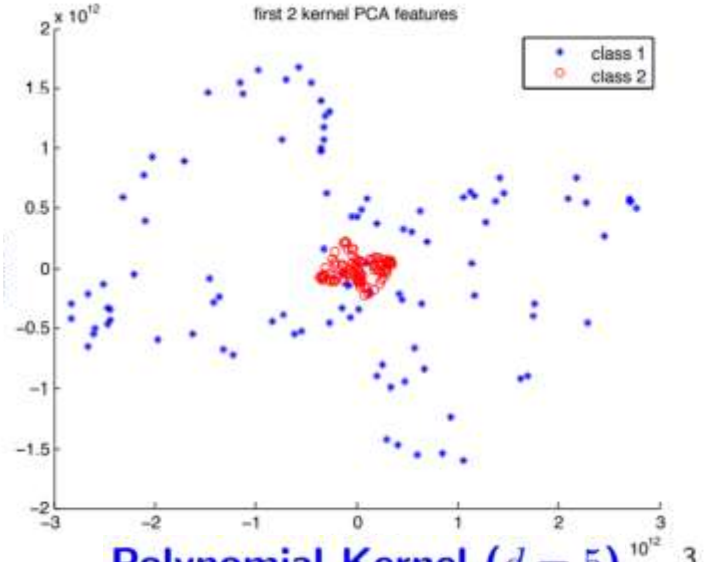
V.S.

KPCA: Eigen decomposition of **normalized kernel** matrix

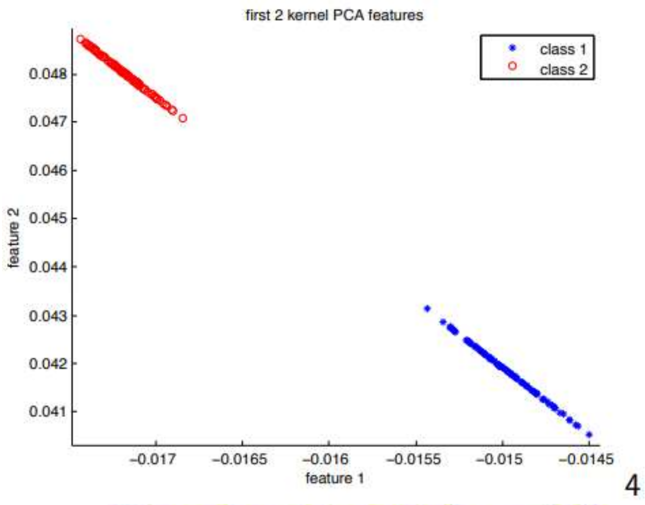




PCA

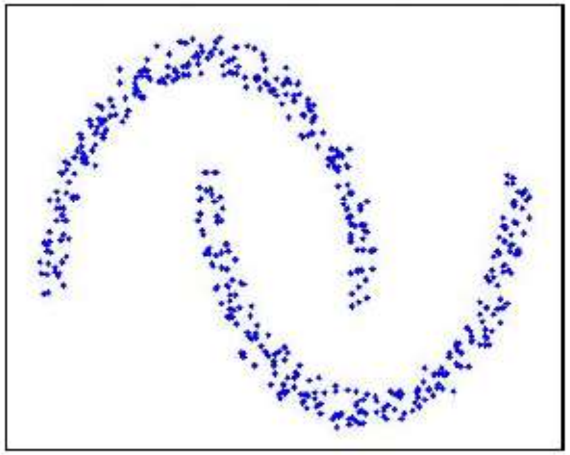


Polynomial Kernel ($d = 5$)



Gaussian Kernel ($\sigma = 20$)

Other Feature Space ?

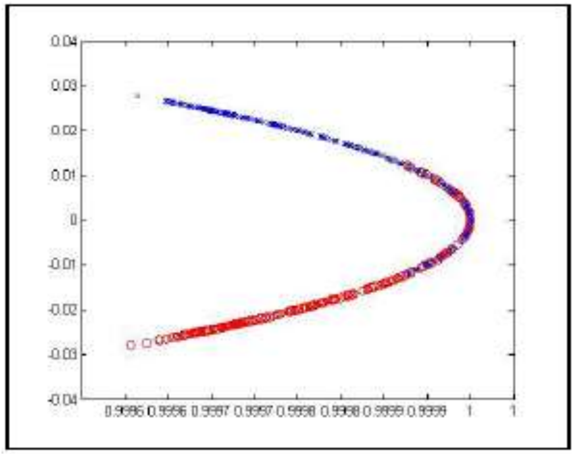


Efficient Kernel Clustering Using **Random Fourier Features**, ICDM 2012

Randomly sample m vectors $\{\omega_1, \omega_2, \ldots \omega_m\}, m \ll n$

$$z(x) = \frac{1}{\sqrt{m}} \left[\cos(\omega_1^T x) \ldots \cos(\omega_m^T x) \ldots \sin(\omega_1^T x) \sin(\omega_m^T x) \right]$$

$$H = \left[z(x_1)^T \;\; z(x_2)^T \ldots z(x_n)^T \right]$$



Cluster H using K-means and obtain the partition

Deep Clustering
深度聚类
→ *K*-means in deep space

Clustering-style Self-Supervised Learning

Mathilde Caron - FAIR Paris & Inria Grenoble

June 20th, 2021

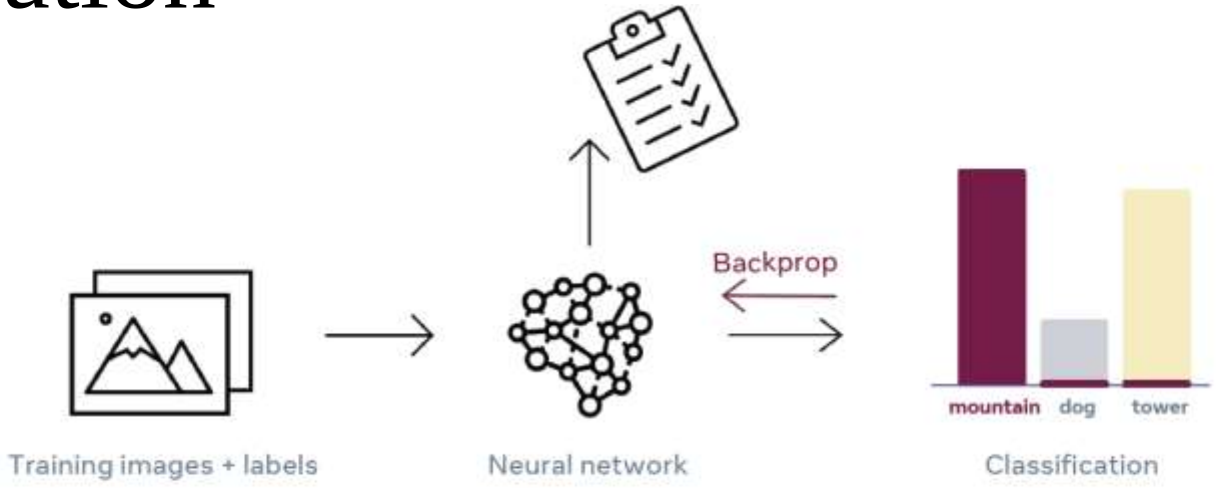
CVPR 2021 Tutorial:

Leave Those Nets Alone:

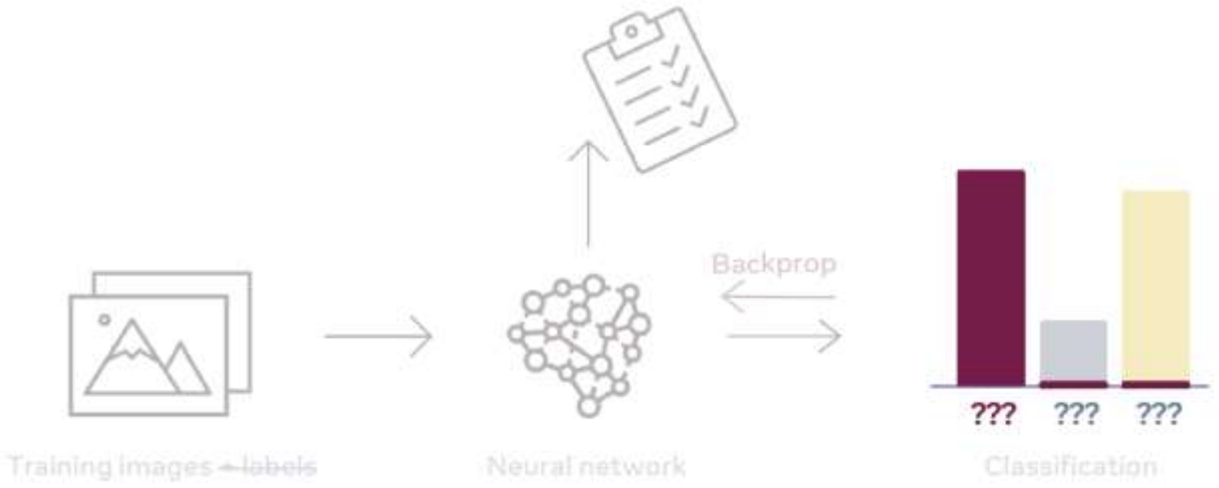
Advances in Self-Supervised Learning

FACEBOOK AI

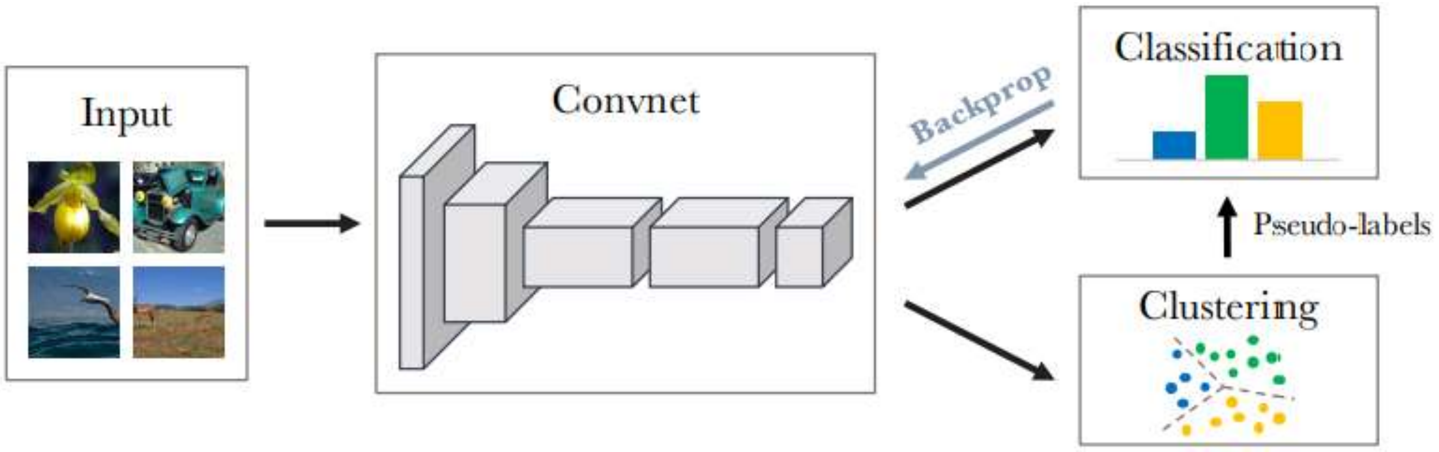
Motivation



Can we replace labels with clustering ?



DeepCluster: Deep Clustering for Unsupervised Learning of Visual Features (ECCV 2018)



We iteratively cluster deep features (*K*-means) and use the cluster assignments as pseudo-labels to learn the parameters of the convnet.

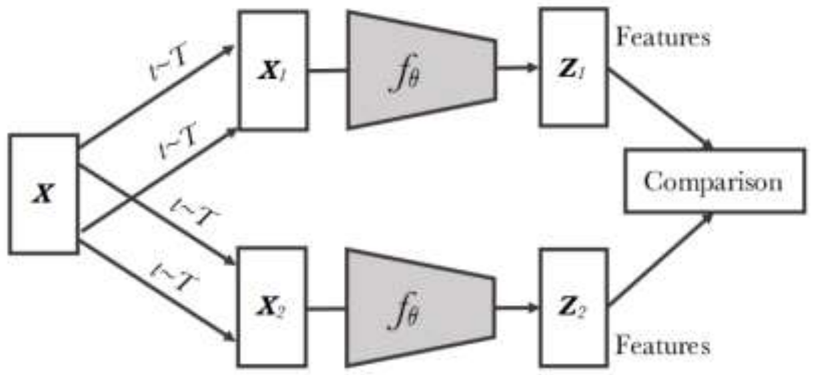
Problem 1: empty cluster

When a cluster becomes empty, we randomly select a non-empty cluster and use its centroid with a small random perturbation as the new centroid for the empty cluster. We then reassign the points belonging to the non-empty cluster to the two resulting clusters.

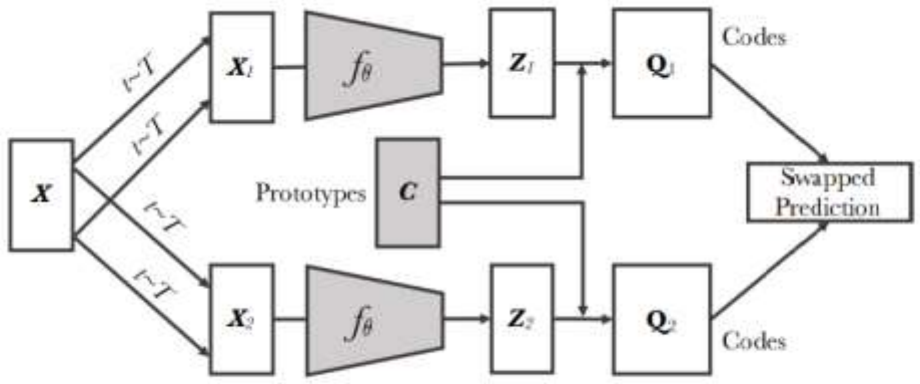
Problem 2: trivial parametrization: predict same outpt for every input

A strategy to circumvent this issue is to sample images based on a uniform distribution over the classes, or pseudo-labels. This is equivalent to weight the contribution of an input to the loss function by the inverse of the size of its assigned cluster.

SwAV: Unsupervised Learning of Visual Features by Contrasting Cluster Assignments (NeurIPS 2020)



Contrastive instance learning



Swapping Assignments between Views (Ours)

Contrastive instance learning (left)

- features from different transformations of the same image are compared to each other.

SwAV (right)

- first obtain "codes" by assigning features to prototype vectors.
- then solve a "swapped" prediction problem.
- does not directly compare image features.
- prototype vectors are learned along with the ConvNet.

SwAV: Unsupervised Learning of Visual Features by Contrasting Cluster Assignments (NeurIPS 2020)

Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Colorization [65]	R50	24	39.6
Jigsaw [46]	R50	24	45.7
NPID [58]	R50	24	54.0
BigBiGAN [15]	R50	24	56.6
LA [68]	R50	24	58.8
NPID++ [44]	R50	24	59.0
MoCo [24]	R50	24	60.6
SeLa [2]	R50	24	61.5
PIRL [44]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [37]	R50	24	65.9
SimCLR [10]	R50	24	70.0
MoCov2 [11]	R50	24	71.1
SwAV	R50	24	**75.3**

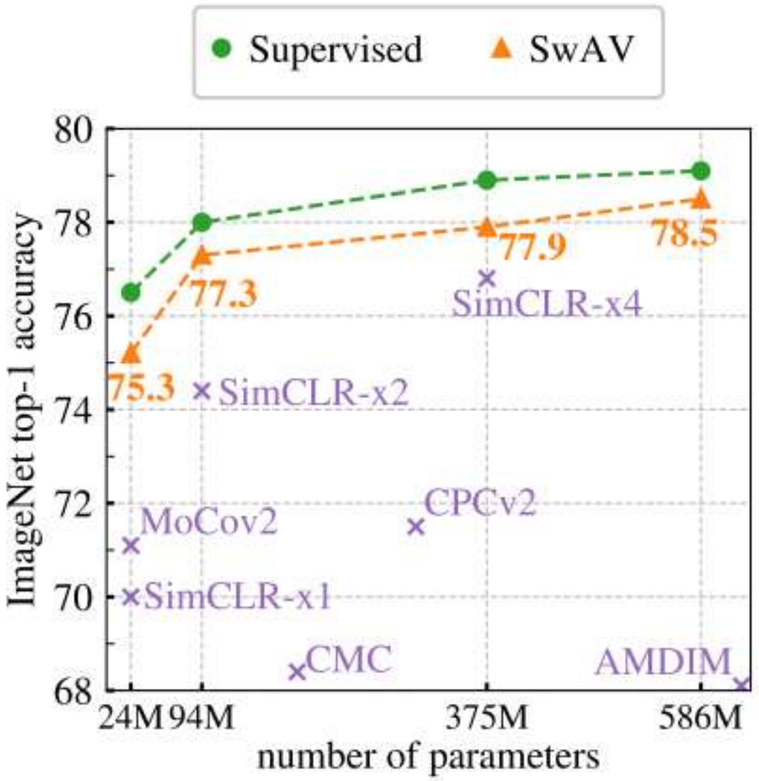


Figure 2: **Linear classification on ImageNet.** Top-1 accuracy for linear models trained on frozen features from different self-supervised methods. **(left)** Performance with a standard ResNet-50. **(right)** Performance as we multiply the width of a ResNet-50 by a factor $\times 2$, $\times 4$, and $\times 5$.

University of Chinese Academy of Sciences

总结与讨论

- K-means clustering
  - K-medians, Competitive learning
  - Mean-shift clustering, DBSCAN
- Gaussian Mixture Model
- Hierarchical Clustering
- Spectral clustering
- Kernel clustering
- Deep clustering

Thank All of You!

(Questions?)