

# 非监督学习 —— 聚类 ( Clustering )

卿来云

# Outline

- 简介
- 距离度量函数
- 聚类性能评价指标
- 聚类算法
  - K均值聚类 ( K-means )
  - 高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )
  - 层次聚类
  - 基于密度的聚类



## 监督学习 vs. 非监督学习

- 监督学习：给定 $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ ，学习输入 $x$ 与输出 $y$ 之间的关系： $y = f(\mathbf{x}; \mathbf{w})$ 
  - 分类： $y$ 类别标签
  - 回归： $y$ 为连续值
  - 排序： $y$ 为有序的数值
- 非监督学习：给定 $\{\mathbf{x}_i\}_{i=1}^N$ ，寻找数据的内在结构： $z = f(\mathbf{x}; \mathbf{w})$ 
  - 概率密度估计： $z$ 为概率密度
  - 聚类： $z$ 为聚类类别
  - 降维/可视化： $z$ 为 $x$ 的低维表示
  - 生成模型：产生类似样本

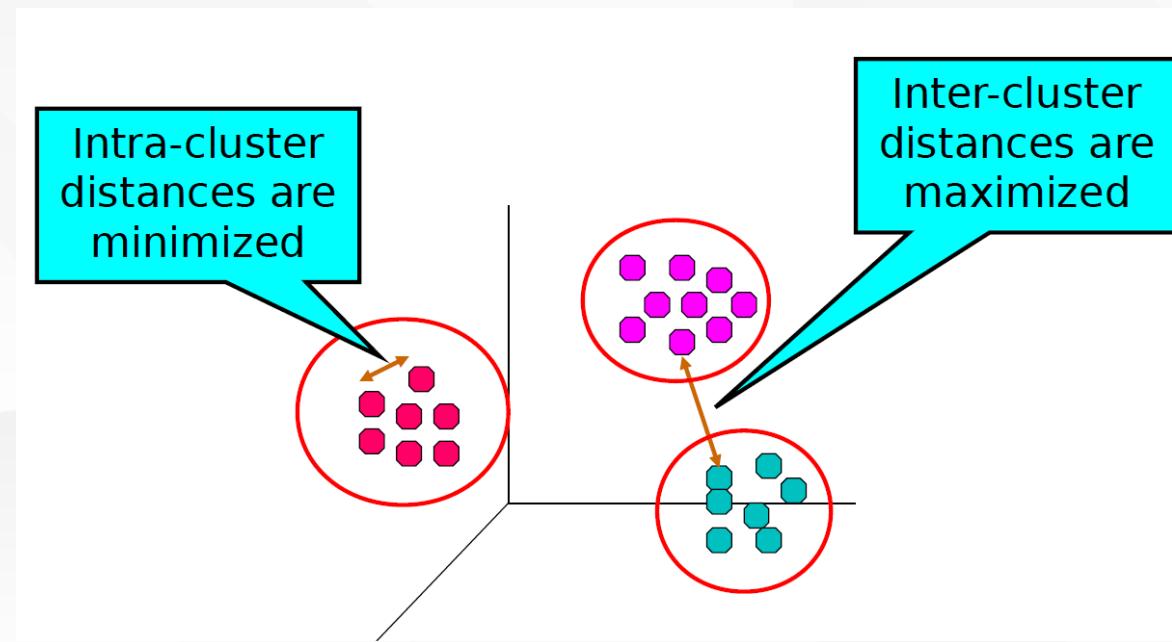
## ➤ 为什么要非监督学习？

- 原始数据容易获得，但标注数据昂贵。
- 降低存储/计算。
- 对高维数据降噪。
- 对数据进行探索性分析（可视化）。
- 非监督学习通常可作为监督学习的预处理步骤。

Unsupervised learning will bring about the next AI revolution.  
—— Yann LeCun

## ➤ 聚类

- 发现数据中分组聚集的结构：根据数据中样本与样本之间的距离或相似度，将样本划分为若干组 / 类 / 簇（cluster）。
- 划分的原则：簇内样本相似、簇间样本不相似



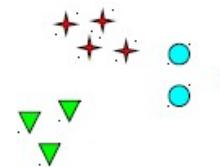
## ➤ 簇的概念并不明确



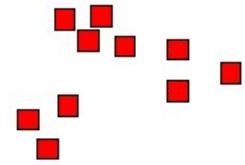
So tell me how  
many clusters do  
you see?



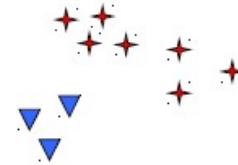
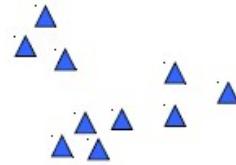
How many clusters?



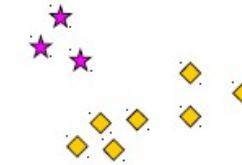
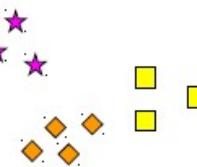
Six Clusters



Two Clusters



Four Clusters



## ➤ 聚类的类型

- 聚类的结果是产生一个簇的集合

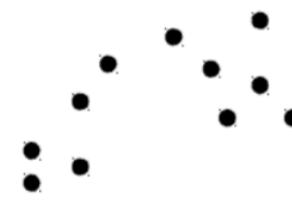
- 基于划分的聚类（无嵌套）

- 将所有样本划分到若干不重叠的子集（簇），且使得每个样本仅属于一个子集

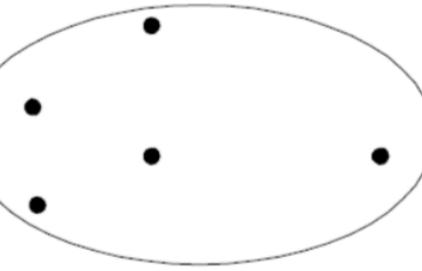
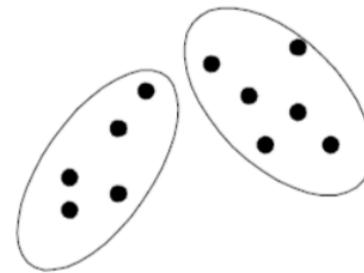
- 层次聚类（嵌套）

- 树形聚类结构，在不同层次对数据集进行划分，簇之间存在嵌套

## ➤ 基于划分的聚类

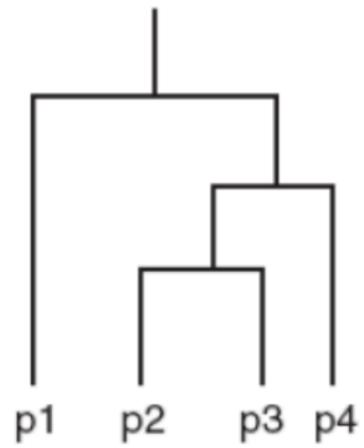


Original Points

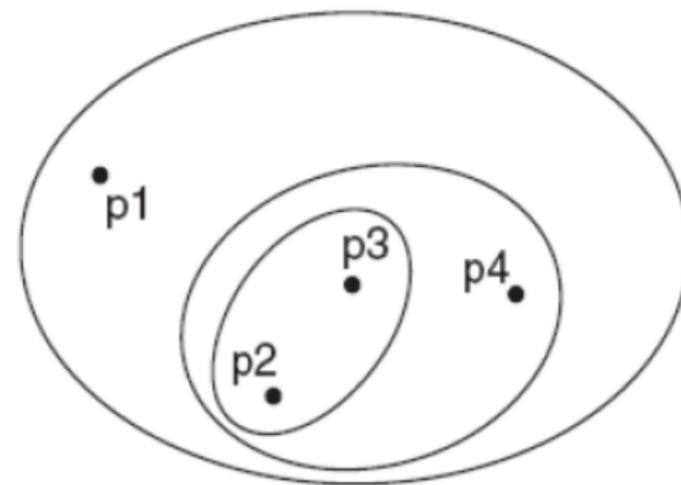


A Partitional Clustering

## ➤ 层次聚类



(a) Dendrogram.



(b) Nested cluster diagram.

树状图

## ➤ 对聚类中簇集合的其他区别

### ■ 独占 ( Exclusive ) vs. 非独占的 ( non-exclusive )

- 在非独占的类簇中，样本点可以属于多个簇

### ■ 模糊 ( Fuzzy ) vs. 非模糊的 ( non-fuzzy )

- 在模糊聚类中，一个样本点以一定权重属于各个聚类簇
- 权重和为1
- 概率聚类有相似的特性

### ■ 部分 ( Partial ) vs. 完备 ( complete )

- 在一些场景，我们只聚类部分数据

### ■ 异质 ( Heterogeneous ) vs. 同质 ( homogeneous )

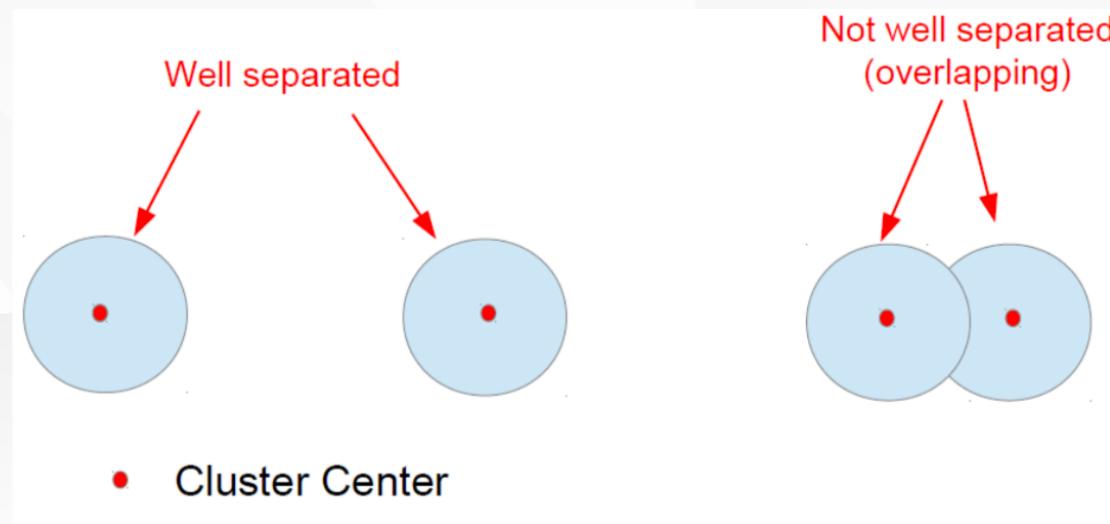
- 簇的大小、形状和密度是否有很大的差别

## ➤ 簇的类型

- 基于中心的簇
- 基于邻接的簇
- 基于密度的簇
- 基于概念的簇

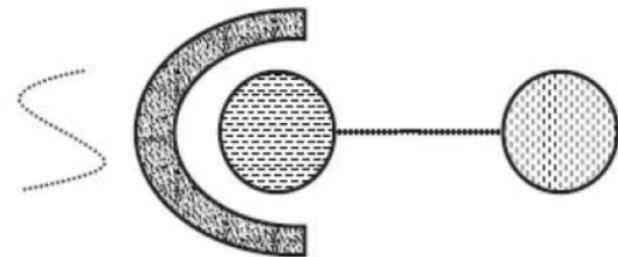
## ➤ 基于中心的簇

- 簇内的点和其“中心”较为相近（或相似），和其他簇的“中心”较远，这样的一组样本形成的簇
- 簇的中心常用质心（metroid）表示，即簇内所有点的平均，或用中心点（medoid）表示，即簇内最有代表性的点



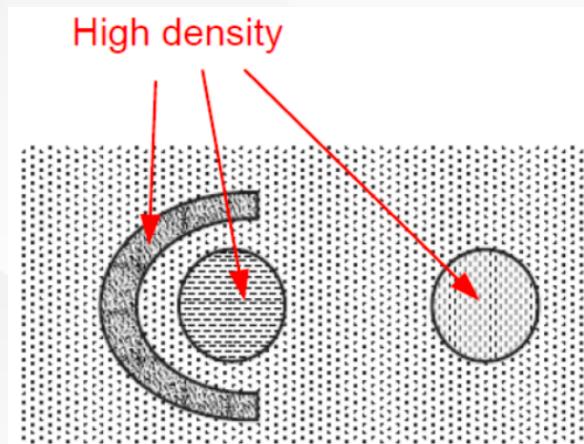
## ➤ 基于连续性和基于密度的簇

- 基于连续性的簇：相比其他任何簇的点，每个点都至少和所属簇的某一个点更近



(c) Contiguity-based clusters. Each point is closer to at least one point in its cluster than to any point in another cluster.

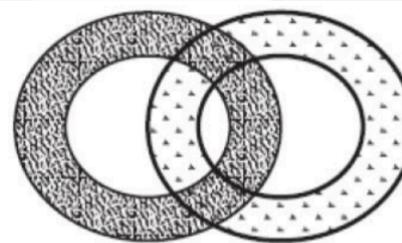
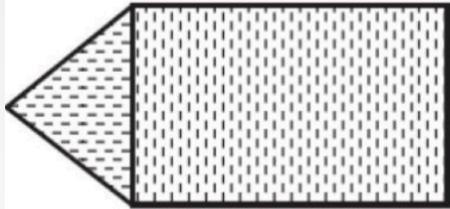
- 基于密度的簇：簇是由高密度的区域形成的，簇之间是一些低密度的区域



(d) Density-based clusters. Clusters are regions of high density separated by regions of low density.

## ➤ 基于概念的簇

■ 基于概念的簇：同一个簇共享某种性质，这个性质是从整个结合推导出来的



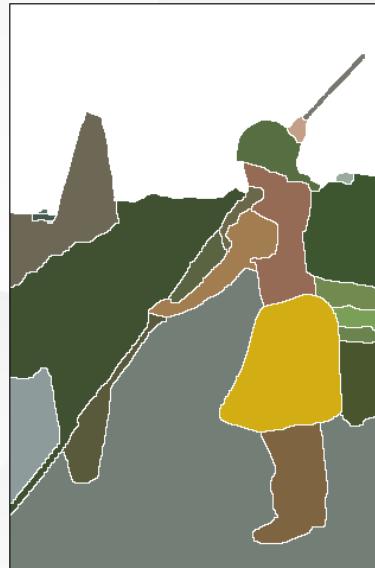
(e) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. (Points in the intersection of the circles belong to both.)

■ 基于概念的簇难检测，它通常不是：

- 基于中心
- 基于关系
- 基于密度

## ➤ 聚类的应用场景

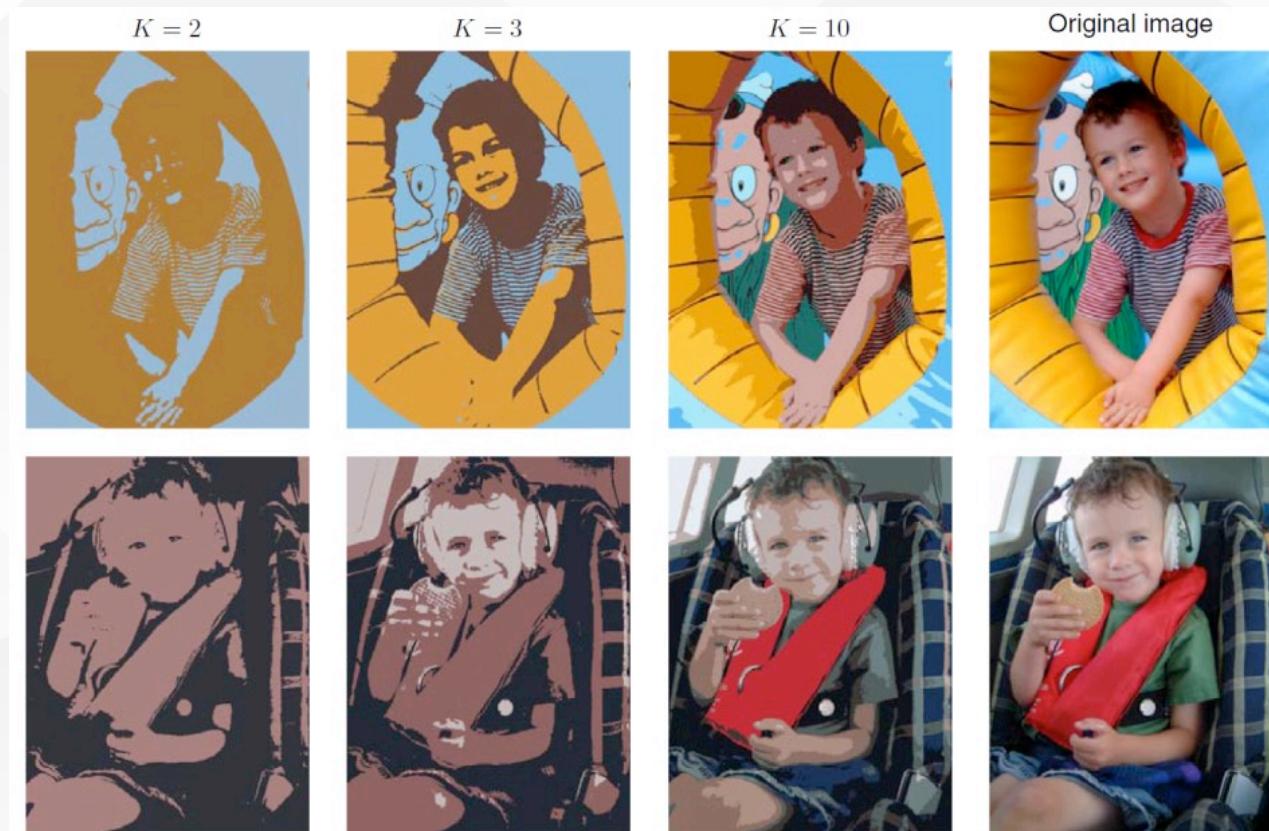
### ■ 图像分割



<https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html>

## ➤ 聚类的应用场景

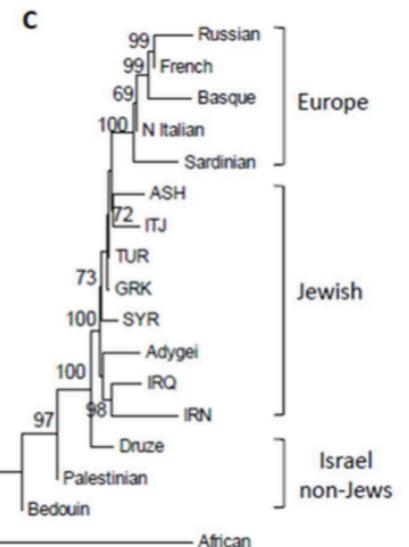
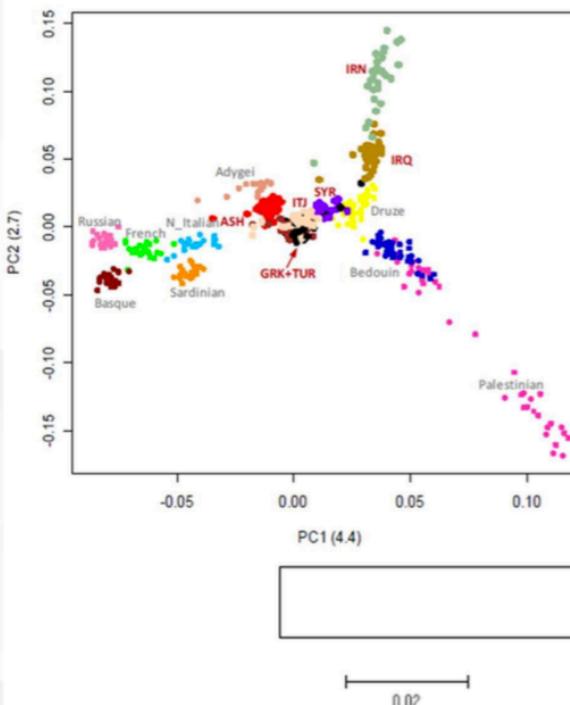
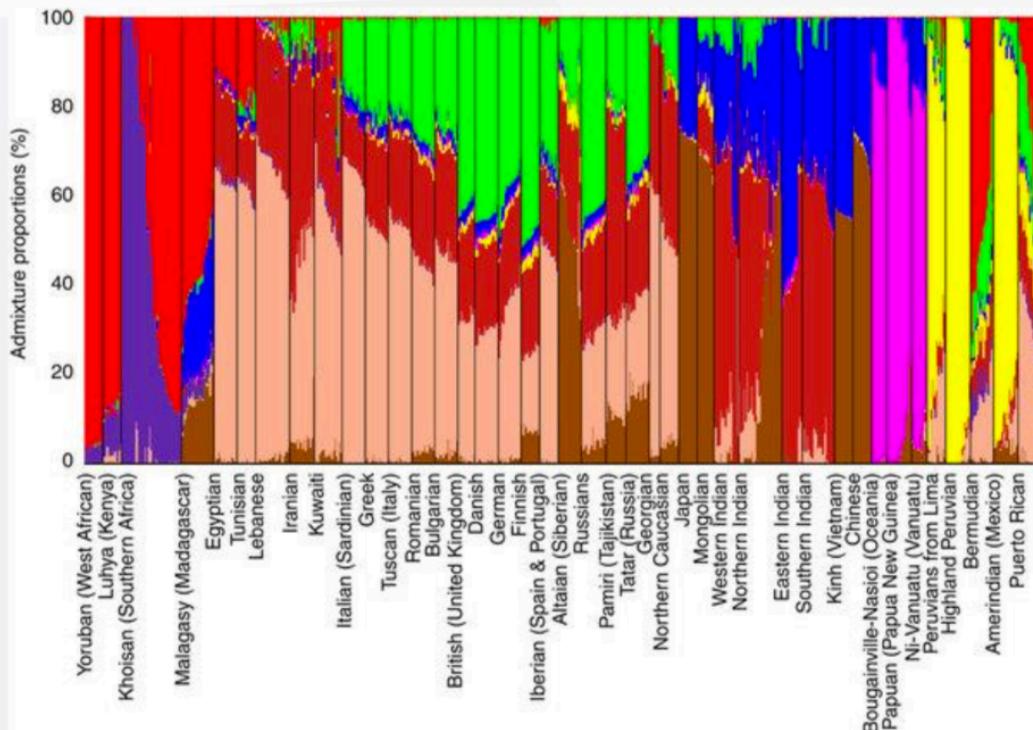
### ■ 图像压缩 • 向量量化



Bishop, PRML

## ➤ 聚类的应用场景

### 人类的种族分析

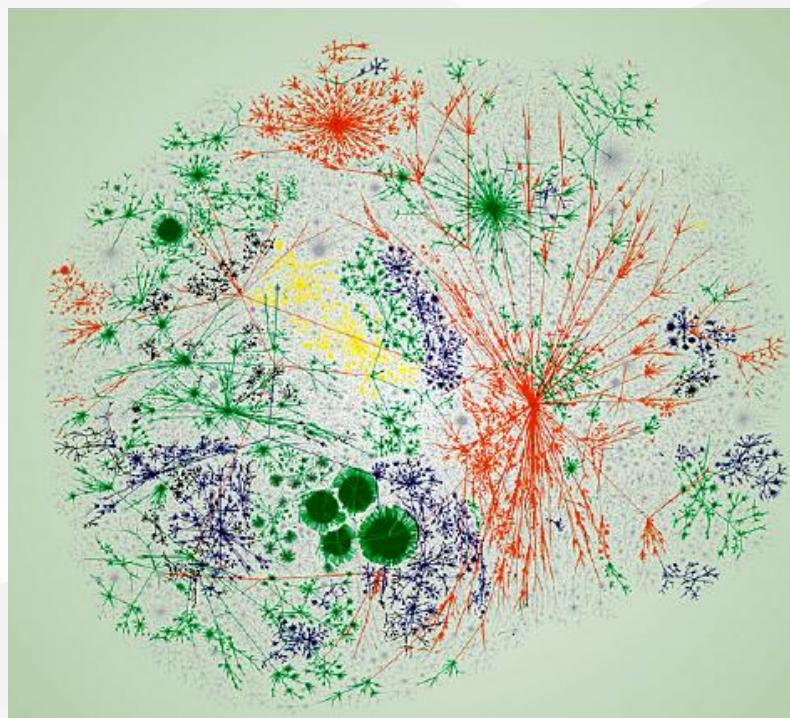


Eran Elhaik et al. Nature

<https://www.nature.com/articles/ncomms4513>

## ➤ 聚类的应用场景

### 复杂网络分析

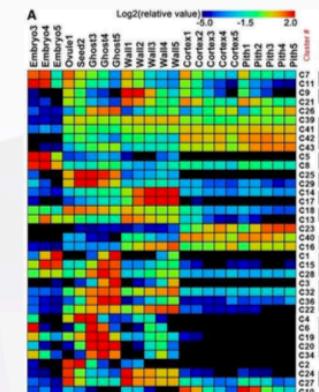
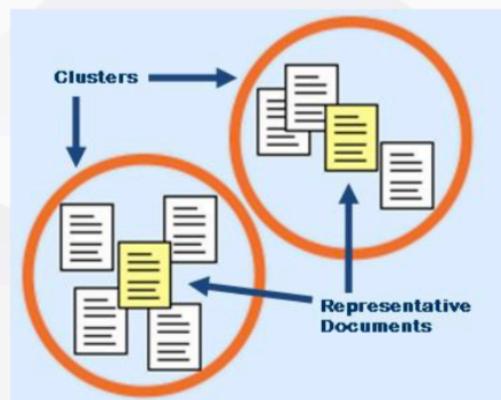


Newman, 2008

## ➤ 其他应用

- 用户画像：基于顾客消费历史对顾客聚类
- 商品分析：基于购买的用户对商品的聚类
- 文本分析：基于相似的词对文档聚类
- 计算生物学：基于编辑距离对DNA序列聚类

|        | Item 1 | Item 2 | Item 3 | Item 4 | Item 5 |
|--------|--------|--------|--------|--------|--------|
| User 1 | 0      | 3      | 0      | 3      | 0      |
| User 2 | 4      | 0      | 0      | 2      | 0      |
| User 3 | 0      | 0      | 3      | 0      | 0      |
| User 4 | 3      | 0      | 4      | 0      | 3      |
| User 5 | 4      | 3      | 0      | 4      | 0      |



## ➤ 聚类分析的“三要素”

- 如何定义样本点之间的“**远近**”
  - 使用相似性/距离函数
- 如何评价聚类出来的簇的**质量**
  - 利用评价函数去评估聚类结果
- 如何**获得**聚类的簇
  - 如何表示簇，如何设计划分和优化算法，算法何时停止

# Outline

- 简介
- **距离度量函数**
- 聚类性能评价指标
- 聚类算法
  - K均值聚类 ( K-means )
  - 高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )
  - 层次聚类
  - 基于密度的聚类

## ➤ 聚类分析的要素之一

- 如何衡量样本之间的“远近”？
  - 文档聚类时，我们如何衡量文档间远近？
  - 图像分割时，我们如何衡量像素点之间的远近？
  - 用户画像时，我们如何衡量用户之间的远近？
- 我们需要量化这些样本，并计算它们之间的距离
- 距离（Distance）/ 相似（similarity）/ 不相似（Dissimilarity）/ 邻近（Proximity）函数的选择与应用相关
- 考虑特征的类型
  - 类别型特征、序数型特征、数值型特征
- 从数据中学习距离度量

## ➤ 距离函数

一个合法的距离度量函数满足：

- $dist(\mathbf{x}_i, \mathbf{x}_j) \geq 0$  (非负性)
- $dist(\mathbf{x}_i, \mathbf{x}_j) = 0 iff \mathbf{x}_i = \mathbf{x}_j$  (identity of indiscernible , 不可分的同一性)
- $dist(\mathbf{x}_i, \mathbf{x}_j) = dist(\mathbf{x}_j, \mathbf{x}_i)$  (对称性)
- $dist(\mathbf{x}_i, \mathbf{x}_j) \leq dist(\mathbf{x}_i, \mathbf{x}_k) + dist(\mathbf{x}_k, \mathbf{x}_j)$  (三角不等式)

## ➤ 闵可夫斯基 ( Minkowski ) 距离

■ 闵可夫斯基距离 :  $dist(\mathbf{x}_i, \mathbf{x}_j) = \left( \sum_{d=1}^D |x_{i,d} - x_{j,d}|^p \right)^{1/p}$

• 当  $p = 2$  时 , 为欧氏距离 :  $dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{d=1}^D (x_{i,d} - x_{j,d})^2}$

• 当  $p = 1$  时 , 为曼哈顿距离 :  $dist(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^D |x_{i,d} - x_{j,d}|$

- 这类距离函数对特征的旋转和平移变换不敏感 , 对数值尺度敏感
- 如果样本特征值尺度不一致 , 将数据标准化 ( 数据是否需要标准化等预处理需考虑特征的物理含义 )

## ➤ 数据预处理

| 操作             | 功能       | 说明                           |
|----------------|----------|------------------------------|
| StandardScaler | 标准化，去量纲  | 对特征矩阵的列，将特征值取值范围标准化（0均值、1方差） |
| MinMaxScaler   | 区间缩放，去量纲 | 对特征矩阵的列，将特征值缩放到区[0, 1]区间     |
| Normalizer     | 归一化      | 对特征矩阵的行，将样本向量转换为“单位向量”（模长为1） |

## ➤ 数据标准化

- 在很多模型中，假设各特征的取值区间相同。如果数据不满足该假设，需要将数据进行变换。
- 标准化是一种常用数据转换方式，将输入特征的均值变为为0，方差为1。Scikit-Learn中用类StandardScaler实现。

```
# 数据标准化
from sklearn.preprocessing import StandardScaler
# 构造输入特征的标准化器
ss_X = StandardScaler()

# 分别对训练和测试数据的特征进行标准化处理
X_train = ss_X.fit_transform(X_train)
X_test = ss_X.transform(X_test)
```

对每维特征单独处理

$$x'_i = \frac{x_i - \mu}{\sigma}$$
$$\mu = \frac{1}{N} \sum_{i=0}^N x_i, \sigma = \sqrt{\frac{1}{N-1} \left( \sum_{i=0}^N x_i - \mu \right)^2}$$

## ➤ 最小最大缩放

■ 另一种数据预处理的方式是将特征取值范围缩放到某个区间（ scaling ），即将样本数据取值限定在特定范围，如  $[0,1]$  （ Scikit-Learn 中用类 [MinMaxScaler](#) 实现）或 将特征取值缩放到  $[-1,1]$  （ [MaxAbsScaler](#) ）。

■ 动机：

- 对非常小的标准偏差的特征更鲁棒
- 在稀疏数据中保留零条目

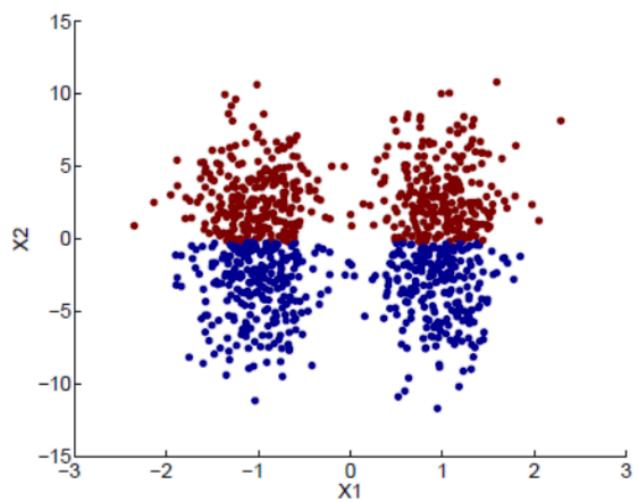
## ➤ 数据归一化/正规化

■ 数据正规化（ normalization ）：将每个样本的模的长度变为单位长度1。Scikit-Learn中用类Normalizer实现。

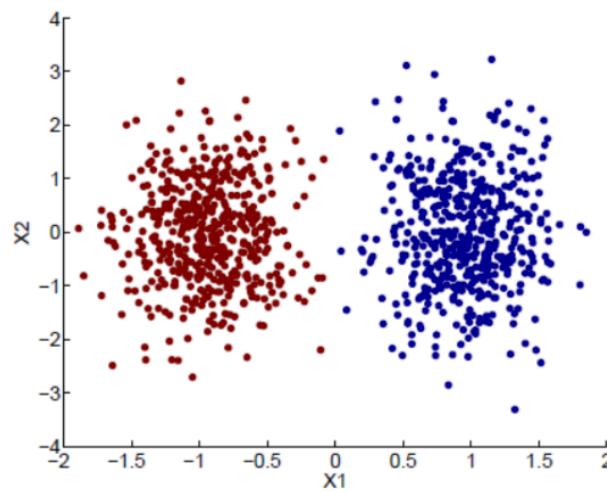
$$\mathbf{x}'_i = \mathbf{x}_i / \|\mathbf{x}_i\|_2 = \mathbf{x}_i / \sqrt{\sum_{d=1}^D x_{i,d}^2}$$

■ 在求欧式距离（相似度度量指标）和文本特征时常用  
• 如KMeans聚类中

## ➤ 例：标准化

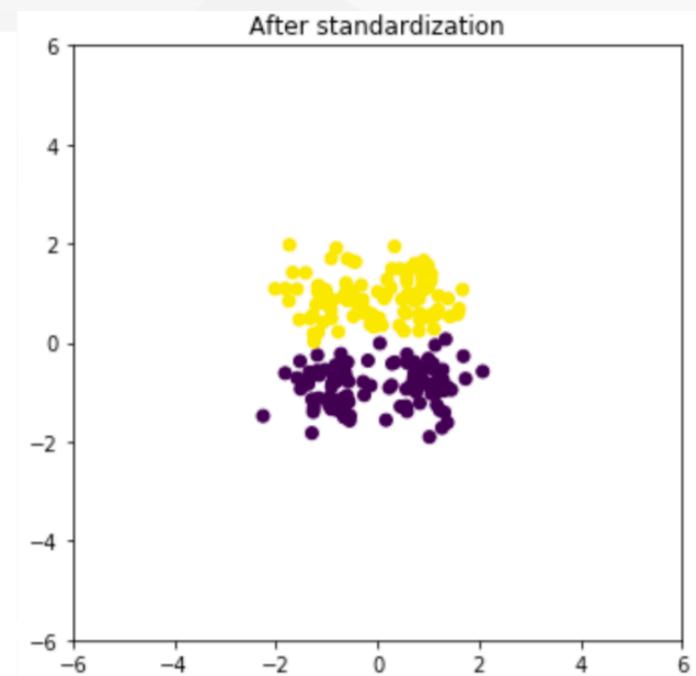
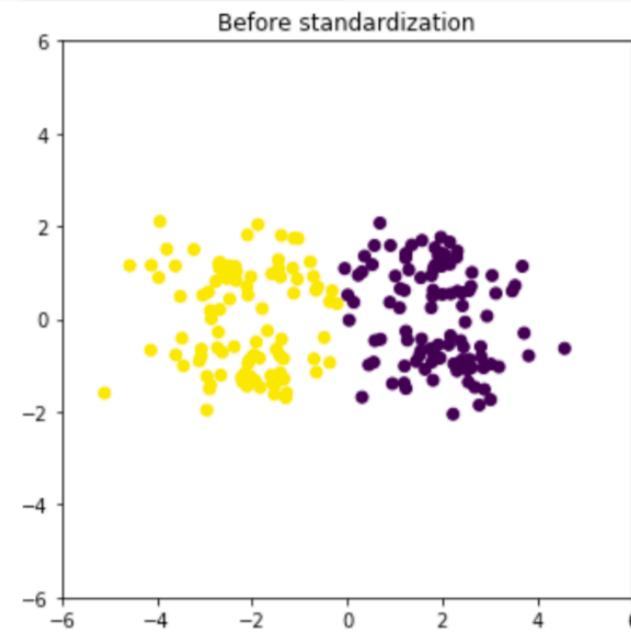
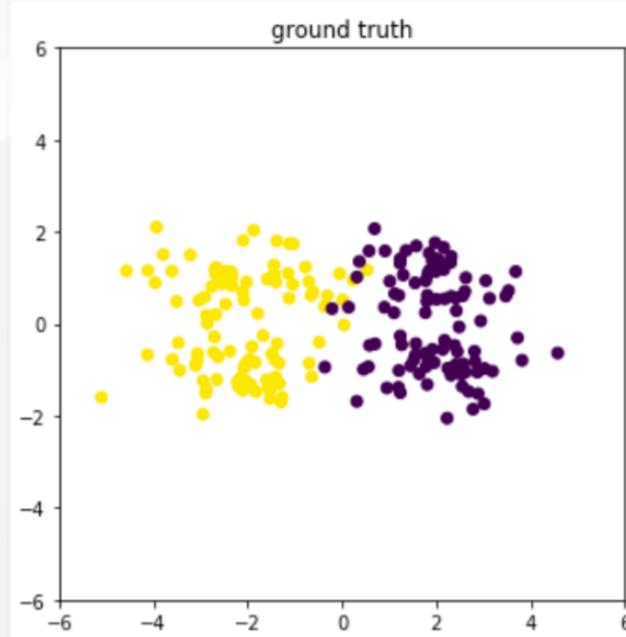


无标准化



标准化

## ➤ 标准化并不一定起效



参考结果

无标准化

标准化

## ➤ 余弦相似度(cosine similarity)

■ 夹角余弦：两变量 $\mathbf{x}_i, \mathbf{x}_j$ 看作 $D$ 维空间的两个向量，这两个向量间的夹角余弦可用下式进行计算

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{d=1}^D x_{i,d} x_{j,d}}{\sqrt{\sum_{d=1}^D x_{i,d}^2} \sqrt{\sum_{d=1}^D x_{j,d}^2}} = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

## ➤ 相关系数

- 相关系数（亦被称为Pearson系数）经常用来度量变量间的相似性。
- 变量 $\mathbf{x}_i, \mathbf{x}_j$ 的相关系数定义为

$$r(\mathbf{x}_i, \mathbf{x}_j) = \frac{cov(\mathbf{x}_i, \mathbf{x}_j)}{\sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_j}} = \frac{\mathbb{E}[(\mathbf{x}_i - \mu_i)(\mathbf{x}_j - \mu_j)]}{\sigma_{\mathbf{x}_i} \sigma_{\mathbf{x}_j}}$$
$$= \frac{\sum_{k=1}^D (x_{i,k} - \mu_{i,k})(x_{j,k} - \mu_{j,k})}{\sqrt{\sum_{k=1}^D (x_{i,k} - \mu_{i,k})^2 \sum_{j=1}^D (x_{j,k} - \mu_{j,k})^2}}$$

当对数据做中心化后，  
 $\mu_i = \mu_j = 0$   
相关系数等于余弦相似度

$$s(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

## ➤ 杰卡德相似系数(Jaccard)

■杰卡德相似系数（交比并）（二值数据）：

$$J(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^D (x_{i,k} \cap x_{j,k})}{\sum_{k=1}^D (x_{i,k} \cup x_{j,k})}$$

# Outline

- 简介
- 距离度量函数
- **聚类性能评价指标**
- 聚类算法
  - K均值聚类 ( K-means )
  - 高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )
- 层次聚类
- 基于密度的聚类

## ➤ 聚类性能评价

### ■聚类性能评价方法：

- 外部评价法 (external criterion) : 聚类结果与参考结果有多相近
- 内部评价法 (internal criterion) : . 聚类的本质特点 (无参考结果)

## ➤ 参考模型 ( reference model )

- 数据集 :  $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ,
- 聚类结果 :  $C = \{C_1, C_2, \dots, C_K\}$ , 其中  $C_k$  表示属于类别  $k$  的样本的集合 ,
- 参考模型 :  $C^* = \{C_1^*, \dots, C_K^*\}$
- $\lambda$  和  $\lambda^*$  分别为  $C$  和  $C^*$  的标记向量
- 计算聚类结果  $C$  和 参考类别结果  $C^*$  的样本对的数目

$$a = \#\{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i, \mathbf{x}_j \in C_k ; \mathbf{x}_i, \mathbf{x}_j \in C_l^*\}$$

( 在两种聚类结果中 , 两个样本的所属的类别相同 )

$$d = \#\{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i \in C_{k1}, \mathbf{x}_j \in C_{k2} ; \mathbf{x}_i \in C_{l1}^*, \mathbf{x}_j \in C_{l2}^*\}$$

( 在两种聚类结果中 , 两个样本的所属的类别不同 )

$$b = \#\{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i, \mathbf{x}_j \in C_k ; \mathbf{x}_i \in C_{l1}^*, \mathbf{x}_j \in C_{l2}^*\}$$

$$c = \#\{(\mathbf{x}_i, \mathbf{x}_j) | \mathbf{x}_i \in C_{k1}, \mathbf{x}_j \in C_{k2} ; \mathbf{x}_i, \mathbf{x}_j \in C_l^*\}$$

$a \uparrow$ , 一致性  $\uparrow$   
 $b \uparrow$ , 一致性  $\downarrow$   
 $c \uparrow$ , 一致性  $\downarrow$   
 $d \uparrow$ , 一致性  $\uparrow$

|      |      | $N(N - 1)/2$ |     | 参考   |     |
|------|------|--------------|-----|------|-----|
|      |      | same         | not | same | not |
| 聚类结果 | same | $a$          | $b$ | same | not |
|      | not  | $c$          | $d$ |      |     |

- 利用  $(a, b, c, d)$  定义外部评价指标

## ➤ 外部索引

### ■ Jaccard 系数 ( JC )

$$JC = \frac{a}{a+b+c}$$

$JC \in [0,1]$ ,  $JC \uparrow$ , 一致性  $\uparrow$

|      |      | $N(N - 1)/2$ |     | 参考   |     |
|------|------|--------------|-----|------|-----|
|      |      | same         | not | same | not |
| 聚类结果 | same | $a$          | $b$ | same | $c$ |
|      | not  | $c$          | $d$ | not  | $d$ |

### ■ Fowlkes and Mallows指数 ( FMI ) : precision 和recall的几何均值

$$FMI = \sqrt{\frac{a}{a+b} \frac{a}{a+c}}$$

$FMI \in [0,1]$ ,  $FMI \uparrow$ , 一致性  $\uparrow$

### ■ Rand 指数 (Rand Index), RI

$$RI = \frac{2(a+d)}{N(N-1)}$$

$RI \in [0,1]$ ,  $RI \uparrow$ , 一致性  $\uparrow$

## ➤ 无参考模型

■大部分时候只有聚类结果，没有参考模型，只能用内部评价法评估聚类的性能

- 簇内相似度越高，聚类质量越好
- 簇间相似度越低，聚类质量越好

## ➤ 簇内相似度

■ 平均距离 :  $avg(C_k) = \frac{1}{|C_k|(|C_k|-1)} \sum_{\mathbf{x}_i, \mathbf{x}_j \in C_k} dist(\mathbf{x}_i, \mathbf{x}_j)$

■ 最大距离 :  $diam(C_k) = \max_{\mathbf{x}_i, \mathbf{x}_j \in C_k} dist(\mathbf{x}_i, \mathbf{x}_j)$

■ 簇的半径 ( diameter ) :  $diam(C_k) = \sqrt{\frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} (dist(\mathbf{x}_i, \boldsymbol{\mu}_k))^2}$

• 其中  $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$

## ➤ 簇间相似度

■ 最小距离 :  $d_{min}(C_k, C_l) = \min_{\mathbf{x}_i \in C_k, \mathbf{x}_j \in C_l} dist(\mathbf{x}_i, \mathbf{x}_j)$

■ 类中心之间的距离 :  $d_{cen}(C_k, C_l) = dist(\boldsymbol{\mu}_k, \boldsymbol{\mu}_l),$

- 其中  $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$

## ➤ 内部评价指标

### ■ DB 指数(DBI)

$$DBI = \frac{1}{K} \sum_{k=1}^K \max_{k \neq l} \frac{\text{avg}(C_k) + \text{avg}(C_l)}{d_{cen}(C_k, C_l)},$$

簇内距离/簇间距离

$DBI \downarrow$ , 聚类质量  $\uparrow$

### ■ Dunn 指数(DI)

$$DI = \min_{1 \leq k < l \leq K} \frac{d_{min}(C_k, C_l)}{\max_{1 \leq k \leq K} diam(C_k)},$$

簇间距离/簇的半径 ( 簇内距离 )

$DI \uparrow$ , 聚类质量  $\uparrow$

## ➤ 内部评价指标

### ■ Calinski-Harabaz Index (CHI)

$$CHI = \frac{Tr(\mathbf{B})}{Tr(\mathbf{W})} \times \frac{N - M}{M - 1}, \quad DI \uparrow, \text{聚类质量} \uparrow \quad \text{计算快}$$

- 其中  $\mathbf{W}$  为簇内散度矩阵 :  $\mathbf{W} = \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$ 
  - 其中簇中心  $\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$
  - $Tr(\mathbf{W}) = \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k} dist(\mathbf{x}_i, \boldsymbol{\mu}_k)$
- $\mathbf{B}$  为簇间散度矩阵 :  $\mathbf{B} = \sum_{k=1}^K |C_k| (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T$ 
  - 其中  $\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$
  - $Tr(\mathbf{B}) = \sum_{k=1}^K |C_k| dist(\boldsymbol{\mu}_k, \boldsymbol{\mu})$

## ➤ Silhouette index

■ 对于其中的一个样本点 $i$ ，记：

- $a(i)$ ：样本点 $i$ 到与其所属簇中其它点的平均距离
- $\overline{d(i, C_k)}$ ：样本点 $i$ 到其他类 $C_k$  ( $x_i \notin C_k$ ) 内所有点的平均距离
- $b(i)$ ：所有 $\overline{d(i, C_k)}$ 的最小值

■ 则样本点 $i$  的轮廓宽度为： $s(i) = \frac{b(i)-a(i)}{\max(a(i), b(i))}$

■ 平均Silhouette值为： $SI = \frac{1}{N} \sum_{i=1}^N s(i)$   $SI \uparrow$ , 聚类质量  $\uparrow$

# Outline

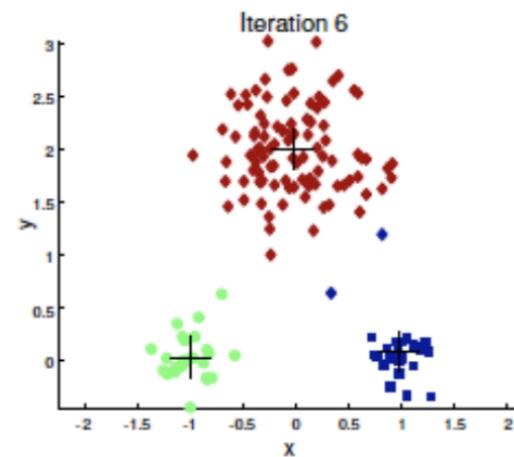
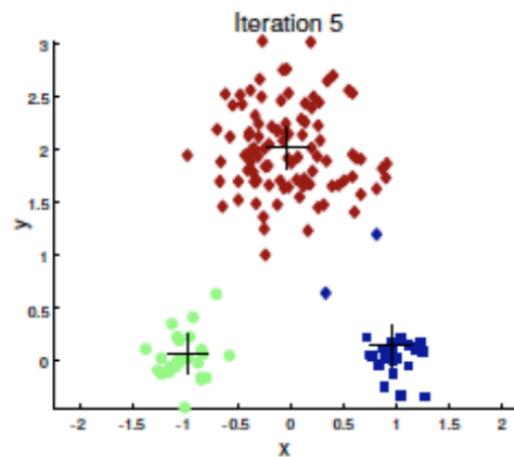
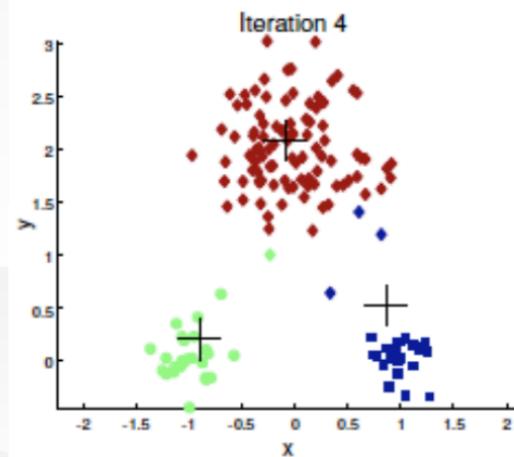
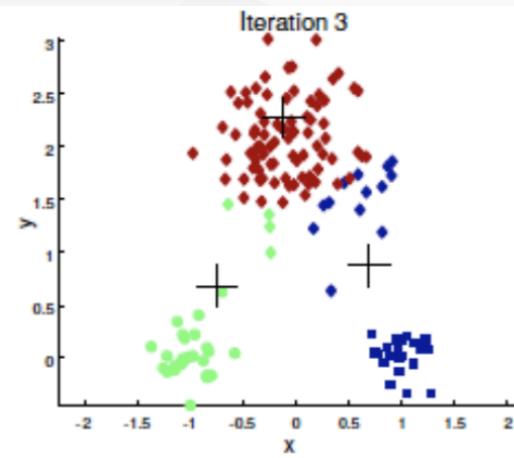
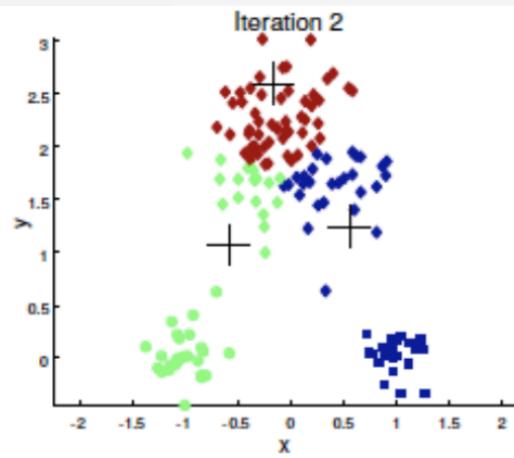
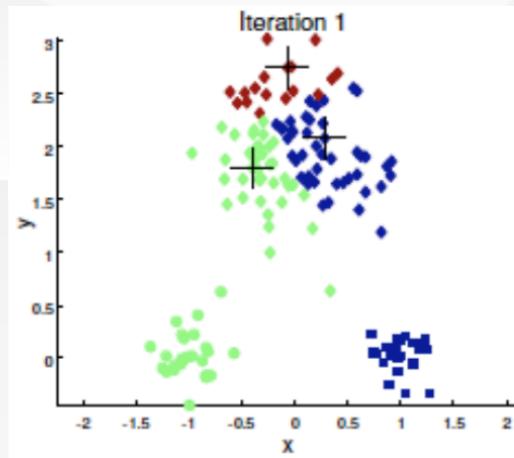
- 简介
- 距离度量函数
- 聚类性能评价指标
- 聚类算法
  - **K均值聚类 ( K-means )**
  - 高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )
  - 层次聚类
  - 基于密度的聚类

## ➤ K均值聚类

- 问题：给定 $N$ 个样本点 $X = \{\mathbf{x}_i\}_{i=1}^N$  进行聚类
  - 输入：数据  $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ ，簇数目为 $K$
- 

1. 随机选择 $K$ 个种子数据点(seeds)作为 $K$ 个簇的中心
  2. repeat
  3.     for each  $\mathbf{x}_i \in \mathcal{D}$  do
  4.         计算 $\mathbf{x}_i$ 与每一个簇中心的距离
  5.         将 $\mathbf{x}_i$ 指配到距离最近的簇中心
  6.     end for
  7.     用当前的簇内点重新计算 $K$ 个簇中心位置
  8. until 当前簇中心未更新
-

## ➤ K均值聚类运行示意



## ➤ K均值聚类的基本假设与优化目标

■ K均值聚类：基于划分的聚类方法

1. 如何表示簇？

- 每个簇都用其质心（centroid）或原型（prototype） $\mu_k$ 表示

2. 如何划分节点？

- 使用欧式距离进行距离度量

- 每个节点都划分到最近的那个质心的簇中

- $r_{i,k} \in \{0,1\}$  为从属度，指示样本 $x_i$ 是否属于簇 $k$ ，且  $\sum_{k=1}^K r_{i,k} = 1$ （每个样本点属于且仅属于一个类）

3. 优化目标：损失函数： $J = \sum_{i=1}^N \sum_{k=1}^K r_{i,k} \|x_i - \mu_k\|^2$ ，平方误差和(SSE)

K-means暗含了对簇的什么假设？

## ➤ 如何优化？

■ 如何最小化损失函数  $J(r_{i,k}, \mu_k)$  ?

■ Chicken and egg problem

- 如果中心点  $\mu_k$  已知，可以计算样本的类别归属  $r_{i,k}$ ；
- 如果样本的类别归属  $r_{i,k}$  已知，可以计算中心点  $\mu_k$ 。

■ 可用迭代求解（坐标轴下降法）：

- 固定  $\mu_k$ ，优化  $r_{i,k}$ ；
- 固定  $r_{i,k}$ ，优化  $\mu_k$ 。

## ➤ 如何优化？

■ 初始化  $K$  个类中心： $\mu_k$

■ 迭代求解：

- 固定  $\mu_k$ ，最小化  $J(r_{i,k})$ ：分配每个样本点到其最近的中心点所在的簇

$$\lambda_i = \operatorname{argmin}_{k'} \operatorname{dist}(x_i, \mu_{k'})$$

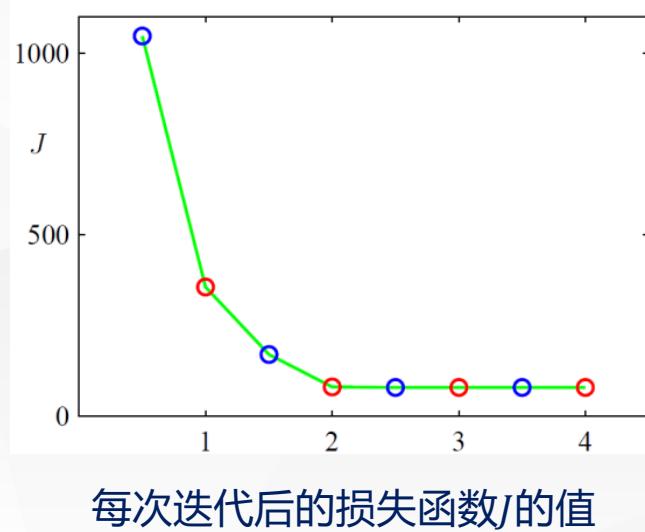
$$\begin{cases} r_{i,k} = 1, k = \lambda_i \\ r_{i,k} = 0, k \neq \lambda_i \end{cases}$$

- 固定  $r_{i,k}$ ，最小化  $J(\mu_k)$ ： $\mu_k$  为簇中所有样本的中心/均值：

$$\mu_k = \frac{\sum_i r_{i,k} \mathbf{x}_i}{\sum_i r_{i,k}}$$

## ➤ K-means收敛性

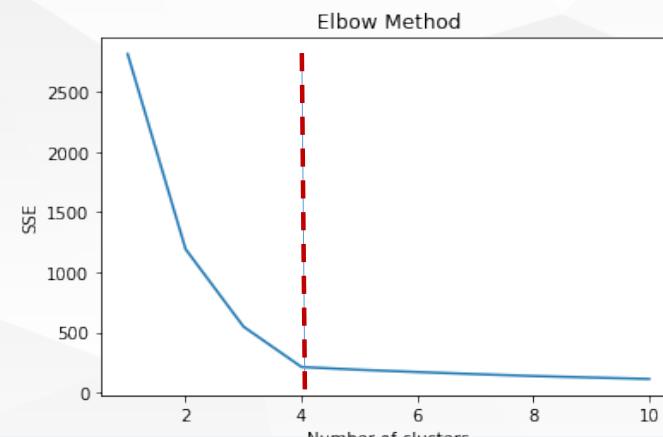
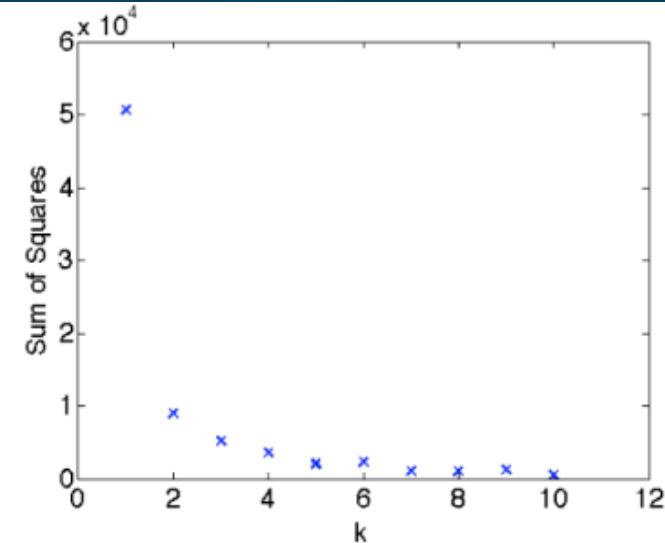
- K-means是在损失函数上进行坐标轴下降(coordinate descent)优化
- 损失函数 $J$ 单调下降，所以损失函数值会收敛，所以聚类结果也会收敛
- K-means有可能会在不同聚类结果间震荡，但在实际中较少发生
- $J$ 是非凸的(non-convex)，所以损失函数 $J$ 上应用坐标下降法不能保证收敛到全局的最小值。一个常见的方法是运行K-means多次，选择最好的结果



## ➤ K的选择

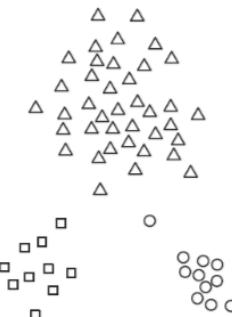
■ 训练集上  $K$  越大，损失越小

- 1. 根据评价指标如CHI、 ...
- 2. 非监督学习/聚类通常是监督学习任务的一部分，用监督学习任务校验集上的评价指标选择  $K$
- 3. 肘部法

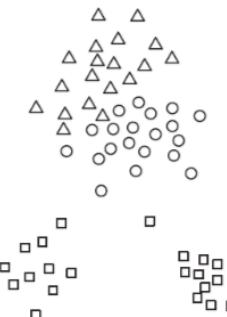


# ➤ 如何初始化K-means ?

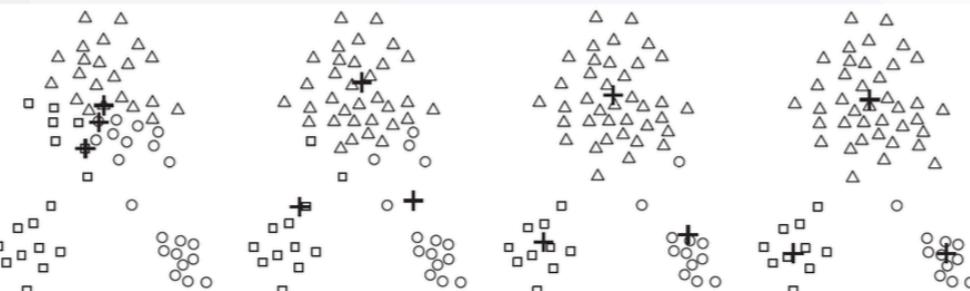
不同初始点的选择



(a) Optimal clustering.



(b) Suboptimal clustering.



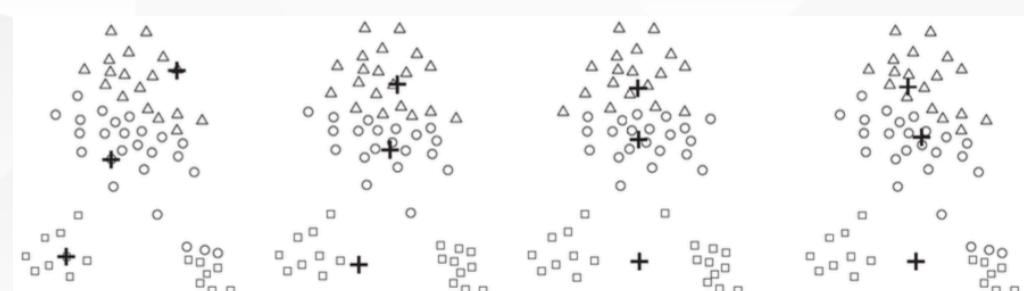
(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

(d) Iteration 4.

聚类较好



(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

(d) Iteration 4.

聚类较差

## ➤ 如何初始化K-means ?

■ 即便存在 $K$ 个真实的簇，正好选到 $K$ 个簇的中心的机会也很小

■ 一些启发式做法

- 随机确定第一个类的中心，其他类中心的位置尽量远离已有类中心
- Scikit-Learn中 $K$ -means实现中参数`init`可设置初始值的设置方式
  - 默认值为 ‘`k-means++`’，将初始化 centroids（质心）彼此远离，得到比随机初始化更好的结果。

## ➤ 预处理和后处理

### ■ 预处理

- 归一化数据 ( e.g. 缩放到单位标准差 )
- 消除离群点

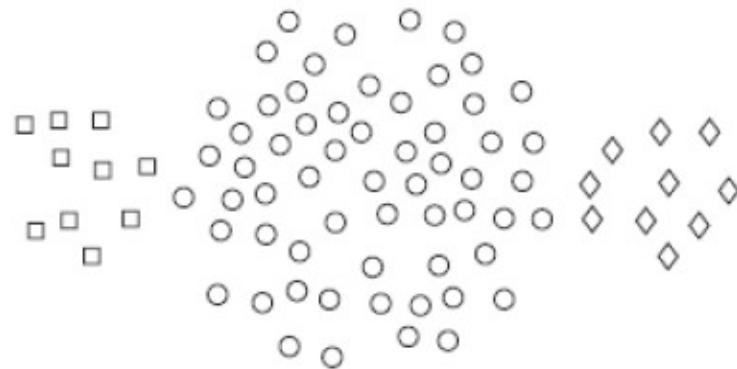
### ■ 后处理

- 删除小的簇 : 可能代表离群点
- 分裂松散的簇 : 簇内节点间距离之和很高
- 合并距离较近的簇

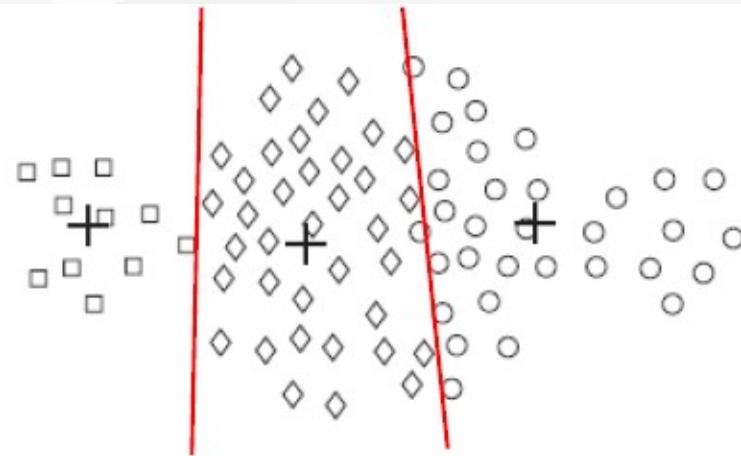
## ➤ K-means的局限性

- K-means 存在问题，当簇具有不同的
  - 尺寸
  - 密度
  - 非球形
- K-means可能得不到理想的聚类结果

## ➤ K-means的局限性: 不同的尺寸

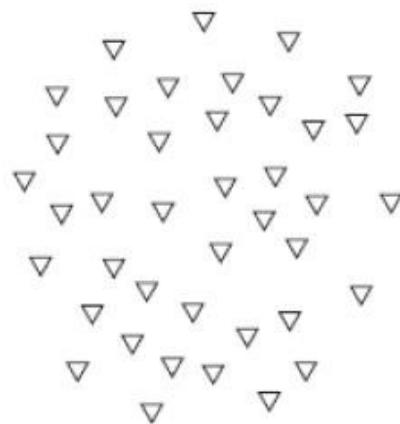


(a) Original points.

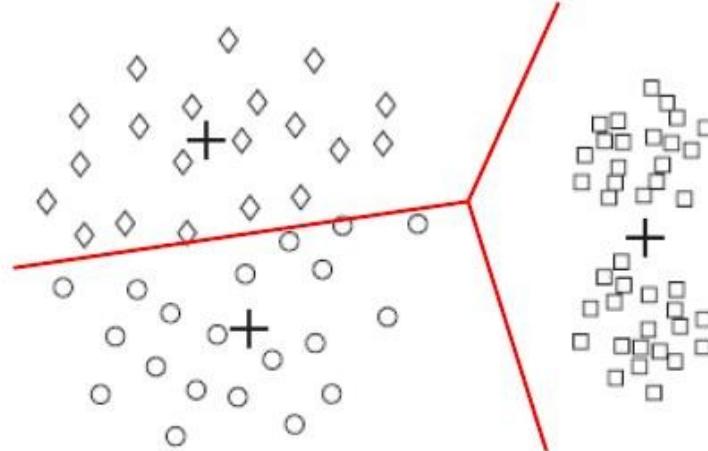
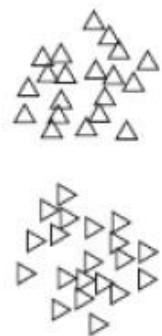


(b) Three K-means clusters.

## ➤ K-means的局限性: 不同的密度

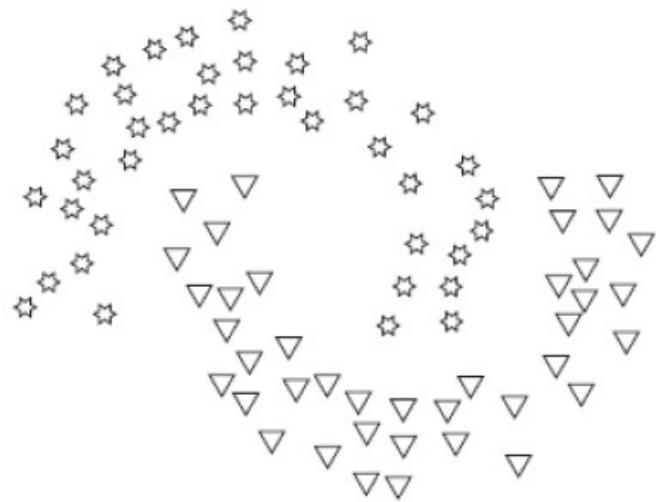


(a) Original points.

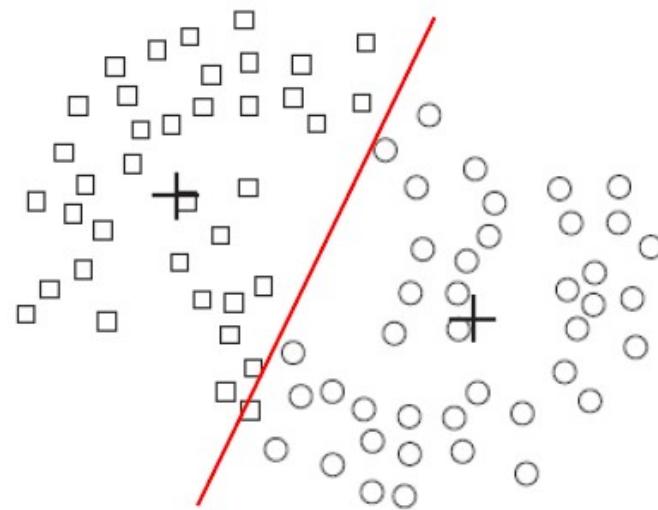


(b) Three K-means clusters.

## ➤ K-means的局限性: 非凸的形状



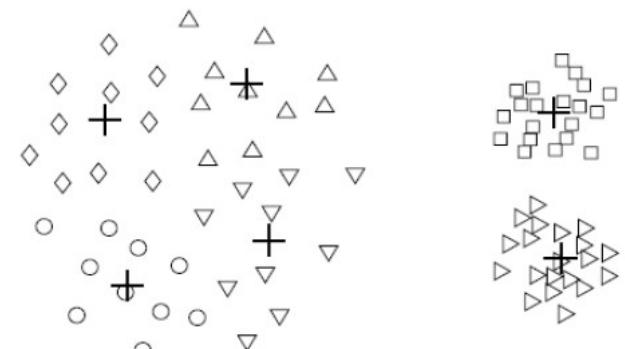
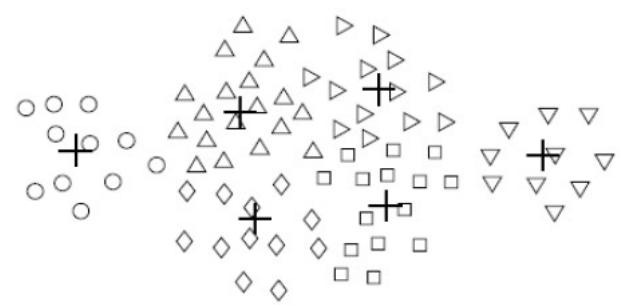
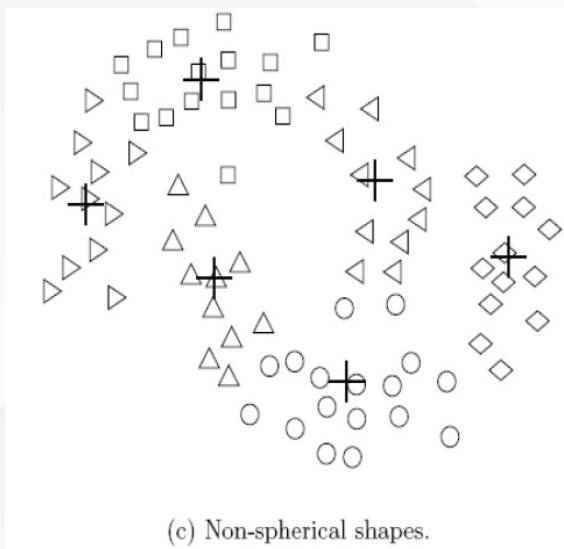
(a) Original points.



(b) Two K-means clusters.

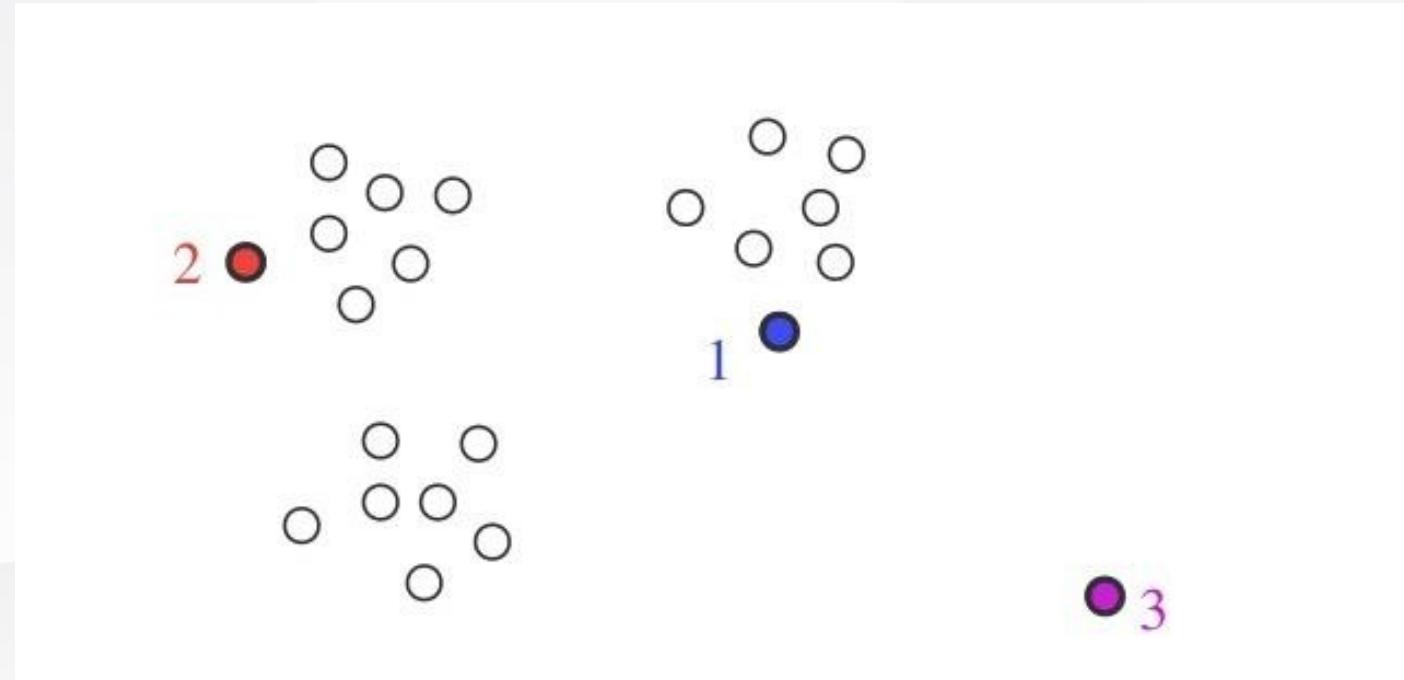
## ➤ 克服K-means的这些局限性

- 使用更大数量的簇
- 几个小的簇表示一个真实的簇
- 使用基于密度的方法



## ➤ K-means的局限性

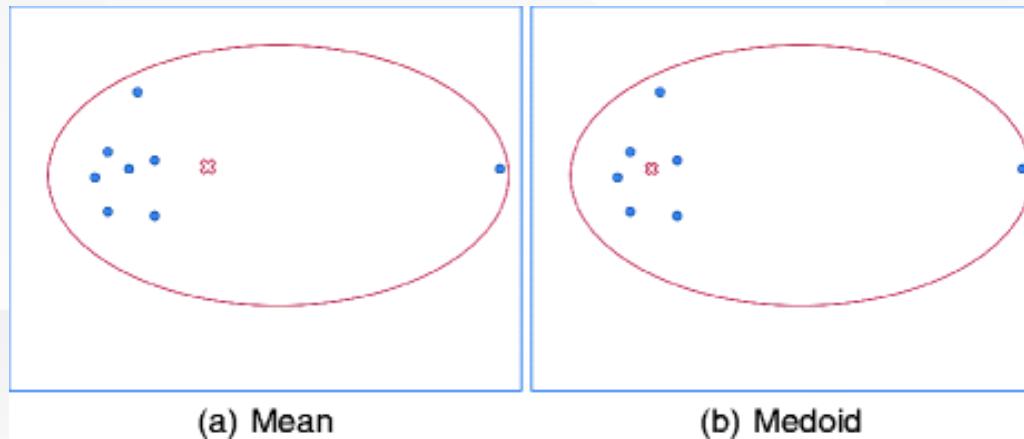
### ■ 离群点带来的问题



## ➤ K-medoids

### ■ 均值作为中心易受影响，换用簇的中位点(median)做为中心

- 均值极有可能是一个不存在的样本点，不足以代表该簇中的样本，而中值是一个样本集合中真实存在的一个样本点
- 相对均值而言，中值对噪声（孤立点、离群点）不那么敏感



## ➤ 非向量空间的聚类

- 对K-means聚类算法，数据必须在某个向量空间，这样欧氏距离是一个很自然的相似性度量。
- 也可以采用类似核函数 $k(\mathbf{x}_i, \mathbf{x}_k)$ 度量样本之间的相似性，然后基于这种相似性进行聚类。

## ➤ K-means聚类的优点

- 一种经典的聚类算法，简单、快速
- 能处理大规模数据，可扩展型好
- 当簇接近高斯分布时，效果较好

## ➤ K-means的局限性

- 硬划分数据点到簇，当数据上出现一些小的扰动，可能导致一个点划分到另外的簇
- 假定簇为球形且每个簇的概率相等
- 解决方案：高斯混合模型

# Outline

- 简介
- 距离度量函数
- 聚类性能评价指标
- 聚类算法
  - K均值聚类 ( K-means )
  - **高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )**
  - 层次聚类
  - 基于密度的聚类

## ➤ 多变量高斯分布

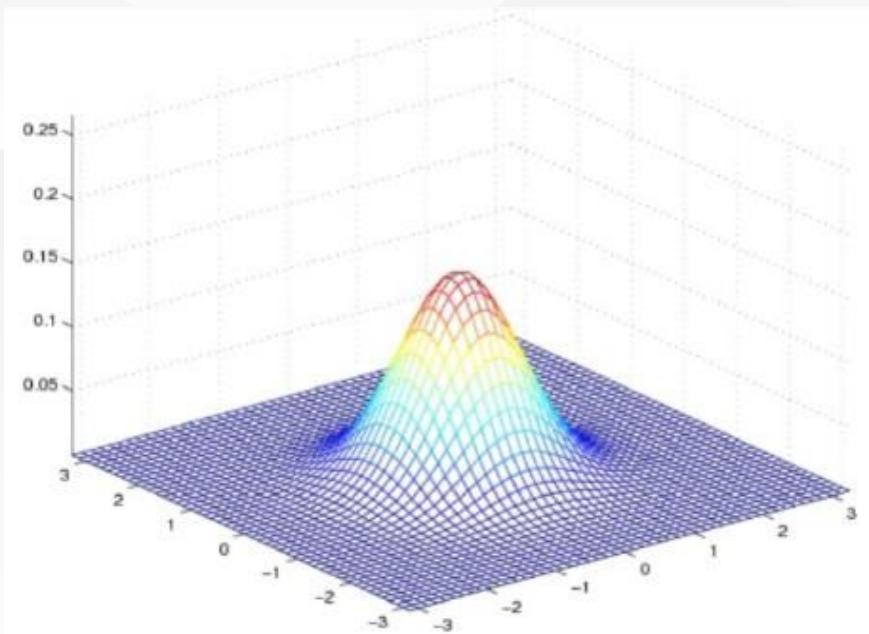


Figure 6: Multivariate Normal Distribution

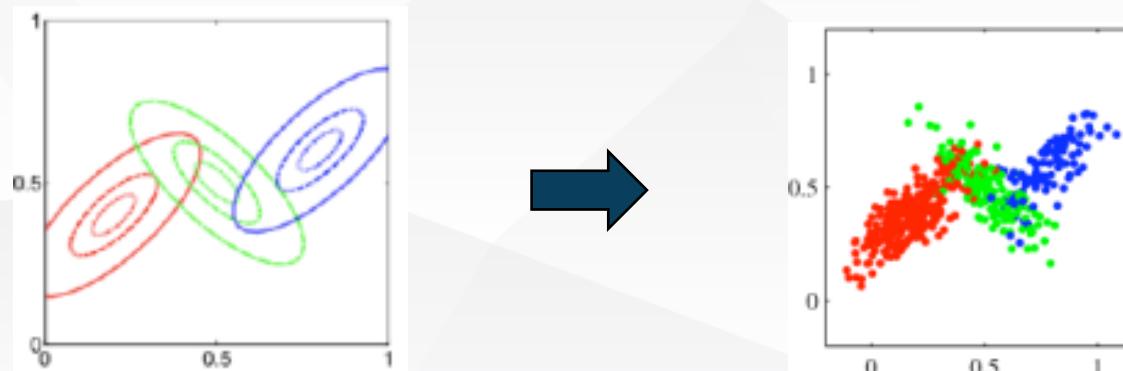
$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

## ➤ 高斯混合模型

- 概率解释: 假设有 $K$ 个簇，每一个簇服从高斯分布，以概率 $\pi_k$ 随机选择一个簇 $k$ ，从其分布中采样出一个样本点，如此得到观测数据
- 概率密度函数

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \text{ where } \sum_{k=1}^K \pi_k = 1, \quad 0 \leq \pi_k \leq 1$$



## ➤ 为高斯混合模型引入隐变量

- 为每个样本点 $\mathbf{x}$ 关联一个 $K$ 维的隐变量  $\mathbf{z} = (z_1, \dots, z_K)$ ，指示样本 $\mathbf{x}$ 所属的簇，因此 $\mathbf{z}$ 采用独热(one-hot)表示

$$P(z_k = 1) = \pi_k$$

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

## ➤ 从属度

- 给定 $\mathbf{x}$ ，可以计算 $\mathbf{z}$ 的条件概率

$$\gamma(z_k) = P(z_k = 1 | \mathbf{x}) = \frac{P(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{k'=1}^K P(z_{k'} = 1)p(\mathbf{x}|z_{k'} = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}$$

- $\gamma(z_k)$ 可以看做是对 $\mathbf{x}$ 从属于第 $k$ 个簇的一种估计或者“解释”。

## ➤ 混合高斯模型的学习过程困难

### ■ 参数学习：极大似然估计

- 最大化以下的对数似然函数(log likelihood)

$$\begin{aligned} \ln(P(\mathbf{X}|\boldsymbol{\theta})) &= \sum_{i=1}^N \ln(p(\mathbf{x}_i|\boldsymbol{\theta})) \\ &= \sum_{i=1}^N \ln \left( \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \end{aligned}$$

### ■ 为什么这个函数优化很困难？

- $\ln$ 里有求和，导致求导困难，所有参数耦合在一起

$P$ 表示随机事件的概率， $p$ 表示概率密度函数。

## ➤ 如何求解？

$$\ln((P(\mathbf{X}|\boldsymbol{\theta})) = \sum_{i=1}^N \log \left( \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

■ 先计算似然函数取最大值时满足的条件， $\ln((P(\mathbf{X}|\boldsymbol{\theta}))$  对  $\boldsymbol{\mu}_k$  求导

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

$$\begin{aligned} 0 = \frac{\partial \ln((P(\mathbf{X}|\boldsymbol{\theta}))}{\partial \boldsymbol{\mu}_k} &= - \sum_{i=1}^N \frac{\pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \times (-\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)) \\ &= - \sum_{i=1}^N \gamma(z_{i,k}) \times (-\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)) \end{aligned}$$

■ 得到： $\boldsymbol{\mu}_k = \frac{\sum_i \gamma(z_{i,k}) \mathbf{x}_i}{\sum_i \gamma(z_{i,k})}$

## ➤ 如何去解？

■ 类似地，我们得到

$$\pi_k = \frac{\sum_i \gamma(z_{i,k})}{N}, \quad \Sigma_k = \frac{\sum_i \gamma(z_{i,k})(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i \gamma(z_{i,k})}$$

■ 注意这不是封闭解， $\gamma(z_{i,k})$ 依赖于参数

■ 这暗示了一种寻找解的迭代方案，这就是EM算法在GMM下的实例  
■ E步：给定当前参数的估计值计算后验概率，或者叫从属度  
M步：基于当前的从属度，重新估计参数的值

## ➤ Expectation-Maximization (EM) 算法

1. 初始化参数并计算对数似然
2. E步: 基于当前参数计算从属度

$$\gamma(z_{i,k}) \triangleq \mathbb{E}(z_{i,k}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}$$

3. M步: 基于当前从属度重新估计参数

$$\pi_k = \frac{\sum_i \gamma(z_{i,k})}{N}, \boldsymbol{\mu}_k = \frac{\sum_i \gamma(z_{i,k}) \mathbf{x}_i}{\sum_i \gamma(z_{i,k})}, \boldsymbol{\Sigma}_k = \frac{\sum_i \gamma(z_{i,k}) (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i \gamma(z_{i,k})}$$

4. 迭代到似然函数收敛

■ 该算法将收敛到似然函数的局部最优

## ➤ EM算法的另一个视角

- 目标函数是对数似然函数： $\ln(P(X|\theta)) = \ln \sum_Z (P(X, Z|\theta))$
- 如果我们知道完整数据 $\{X, Z\}$ ，则完整数据的对数似然函很直接： $\ln(P(X, Z|\theta))$
- 然而，关于隐变量 $Z$ 的值的信息只能从后验分布得到 $p(Z|X, \theta^{old})$ 。
- 因此，我们不妨考虑其**期望值** (即E步)

$$Q(\theta, \theta^{old}) = \mathbb{E}_Z \{ \ln(P(X, Z|\theta)) \} = p(Z|X, \theta^{old}) \ln(P(X, Z|\theta))$$

- 在 $M$ 步，我们**最大化**期望： $\theta^{new} = \operatorname{argmax}_{\theta} Q(\theta, \theta^{old})$

## ➤ 通用的EM算法

- 初始参数  $\theta^{old}$
- E步：计算  $p(Z|X, \theta^{old})$
- M步：计算  $\theta^{new} : \theta^{new} = \underset{\theta}{\operatorname{argmax}} Q(\theta, \theta^{old})$
- 其中  $Q(\theta, \theta^{old}) = \mathbb{E}_z \{ \ln(P(X, Z|\theta)) \} = p(Z|X, \theta^{old}) \ln(P(X, Z|\theta))$
- 检查似然函数或者参数的值是否收敛，如果没有达到收敛条件，则令  $\theta^{old} \leftarrow \theta^{new}$ ，转E步。

## ➤ 高斯混合模型回顾

■ 完整数据的对数似然函数：

$$\begin{aligned} \ln(P(X, Z | \pi, \mu, \Sigma)) &= \ln(P(Z | \pi)) \ln(P(X | Z, \mu, \Sigma)) \\ &= \sum_{i=1}^N \sum_{k=1}^K z_{i,k} (\log(\pi_k) + \ln(\mathcal{N}(x_i | \mu_k, \Sigma_k))) \end{aligned}$$

- $\pi_k$  和  $(\mu_k, \Sigma_k)$  在完整数据似然函数中解耦合，存在平凡的封闭解
- 期望为：

$$\begin{aligned} \mathbb{E}_z \{ \ln(P(X, Z | \theta)) \} &= \sum_{i=1}^N \sum_{k=1}^K \mathbb{E}_z(z_{i,k}) (\ln(\pi_k) + \ln(\mathcal{N}(x_i | \mu_k, \Sigma_k))) \\ &= \sum_{i=1}^N \sum_{k=1}^K \gamma_{i,k} (\ln(\pi_k) + \ln(\mathcal{N}(x_i | \mu_k, \Sigma_k))) \end{aligned}$$

■ Vs. 不完整数据的对数似然函数：

$$\ln(P(X | \pi, \mu, \Sigma)) = \sum_{i=1}^N \ln \left( \sum_{k=1}^K \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \right)$$

## ➤ 通用EM算法

- EM算法是一种寻找含有隐变量的概率模型的最大似然解的通用技术
- 为什么我们通过这种启发式的方式推导出来的**EM算法**，但确实是在最大化似然函数？

$$\theta^{new} = \underset{\theta}{\operatorname{argmax}} Q(\theta, \theta^{old}) \quad \rightarrow \quad \theta = \underset{\theta}{\operatorname{argmax}} P(X, \theta)$$

## ➤ 通用EM算法

### ■ 考虑一个概率模型

- 可观测变量 $X$  和隐变量 $Z$
- 联合概率分布为 $P(X, Z|\theta)$

■ 我们的目标是最大化如下的似然函数： $P(X|\theta) = \sum_Z P(X, Z|\theta)$

■ 直接优化 $P(X|\theta)$ 很困难, 但优化完整数据的似然函数 $P(X, Z|\theta)$ 容易得多

## ➤ The EM Algorithm in General

■ E步：用参数的当前值 $\theta$ ，计算给定观测 $x$ 隐含变量 $z$ 的分布 $q$ ：

$$q(z) = p(z|x, \theta)$$

■ M步：最大化 $\ln(p(x, z|\theta))$  的期望，其中期望在E步得到的分布 $q$ 下进行： $\theta = \operatorname{argmax}_{\theta} \mathbb{E}_q [\ln(p(x, z|\theta))]$

## ➤ 证明

### ■ 定义

$$\begin{aligned} F(q, \theta) &= \mathbb{E}_q[\log(p(\mathbf{x}, \mathbf{z}|\theta))] - \mathbb{E}_q[\log(q(\mathbf{z}))] \\ &= \mathbb{E}_q[\log(p(\mathbf{x}|\theta) \times p(\mathbf{z}|\mathbf{x}, \theta))] - \mathbb{E}_q[\log(q(\mathbf{z}))] \\ &= \mathbb{E}_q[\log(p(\mathbf{x}|\theta))] + \mathbb{E}_q[\log(p(\mathbf{z}|\mathbf{x}, \theta))] - \mathbb{E}_q[\log(q(\mathbf{z}))] \\ &= \log(p(\mathbf{x}|\theta)) - \mathbb{E}_q[\log(q(\mathbf{z})/p(\mathbf{z}|\mathbf{x}, \theta))] \\ &= \log(p(\mathbf{x}|\theta)) - KL(q||p) \end{aligned}$$

## ➤ 证明

■ E步求使得  $F(q, \theta)$  最大的  $q$

- 当时  $q(z) = p(z|x, \theta)$  , KL散度  $KL(q||p)$  最小

$$F(q, \theta) = \ln(p(x|\theta)) - KL(q||p)$$

■ M步求使得  $F(q, \theta)$  最大  $\theta$

- $\mathbb{E}_q[\ln(q(z))]$  与  $\theta$  无关。

$$F(q, \theta) = \mathbb{E}_q[\ln(p(x, z|\theta))] - \mathbb{E}_q[\ln(q(z))]$$

■ 当  $F(q, \theta)$  最大时 ,  $p(x|\theta)$  也最大

- 否则存在某个  $\theta^*$  ,  $p(x|\theta^*) > p(x|\theta)$  , 则  $F(q^*, \theta^*) > F(q, \theta)$  , 其中  $q^*(z) = p(z|x, \theta^*)$ .

## ➤ 例：EM for混合贝努利分布

■ 1. 初始值： $\pi^{(0)} = 0.5, p^{(0)} = 0.5, q^{(0)} = 0.5$

■ 2. 计算隐含变量的概率

$$\begin{aligned} P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) &= \frac{P(z_i = 1|\boldsymbol{\theta}^{(t)})P(x_i|z_i = 1, \boldsymbol{\theta}^{(t)})}{P(z_i = 1|\boldsymbol{\theta}^{(t)})P(x_i|z_i = 1, \boldsymbol{\theta}^{(t)}) + P(z_i = 0|\boldsymbol{\theta}^{(t)})P(x_i|z_i = 0, \boldsymbol{\theta}^{(t)})} \\ &= \frac{\pi^{(0)}(p^{(0)})^{x_i}(1 - p^{(0)})^{(1-x_i)}}{\pi^{(0)}(p^{(0)})^{x_i}(1 - p^{(0)})^{(1-x_i)} + (1 - \pi^{(0)})(q^{(0)})^{x_i}(1 - q^{(0)})^{(1-x_i)}} \end{aligned}$$

$$P(z_i = 1|x_i = 1, \boldsymbol{\theta}^{(0)}) = \frac{0.5 \times 0.5}{0.5 \times 0.5 + 0.5 \times 0.5} = 0.5, P(z_i = 0|x_i = 1, \boldsymbol{\theta}^{(0)}) = 0.5$$

$$P(z_i = 1|x_i = 0, \boldsymbol{\theta}^{(0)}) = \frac{0.5 \times 0.5}{0.5 \times 0.5 + 0.5 \times 0.5} = 0.5, P(z_i = 0|x_i = 0, \boldsymbol{\theta}^{(0)}) = 0.5$$

## ➤ 例：EM for混合贝努利分布

### ■ 3. 计算使得函数 $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$ 最大的参数值

$$\begin{aligned}
 Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{(t)}) \ln(P(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})) \\
 &= \sum_{i=1}^N \sum_{k=1}^1 P(z_i = k|x_i, \boldsymbol{\theta}^{(t)}) \left( \ln(p(z_i)) + \ln(Bernoulli(x_i|p, q)) \right) \\
 &= \sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) (\ln(\pi) + (x_i \ln(p) + (1 - x_i) \ln(1 - p))) \\
 &\quad + \sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)}) (\ln(1 - \pi) + (x_i \ln(q) + (1 - x_i) \ln(1 - q)))
 \end{aligned}$$

$$0 = \frac{\partial Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})}{\partial \pi} = \frac{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)})}{\pi} - \frac{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)})}{1 - \pi}$$

$$\boxed{\pi = \frac{1}{N} \sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)})}$$

$$\pi^{(1)} = \frac{1}{N} \sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(0)}) = 0.5$$

## ➤ 例：EM for混合贝努利分布

■ 3. 计算使得函数  $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$  最大的参数值

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) (\ln(\pi) + (x_i \ln(p) + (1 - x_i) \ln(1 - p))) \\ + \sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)}) (\ln(1 - \pi) + (x_i \ln(q) + (1 - x_i) \ln(1 - q)))$$

$$0 = \frac{\partial Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})}{\partial p} = \frac{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) x_i}{p} - \frac{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) (1 - x_i)}{1 - p}$$

$$p = \frac{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) x_i}{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)})}$$

$$p^{(1)} = \frac{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(0)}) x_i}{\sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(0)})} = 0.6$$

## ➤ 例：EM for混合贝努利分布

■ 3. 计算使得函数  $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$  最大的参数值

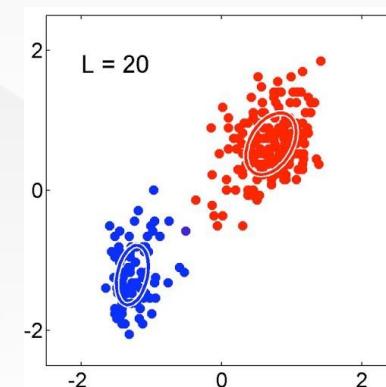
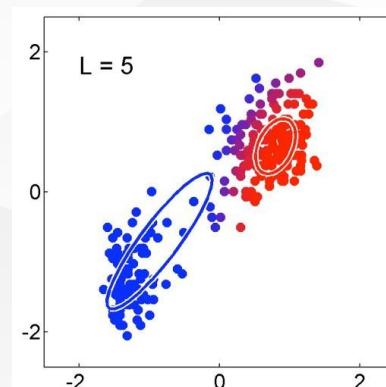
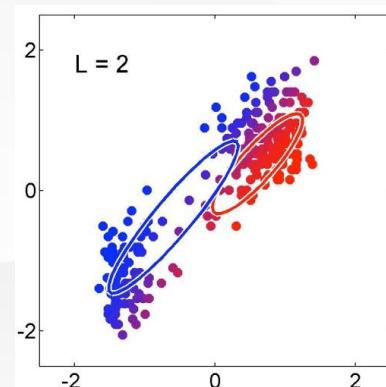
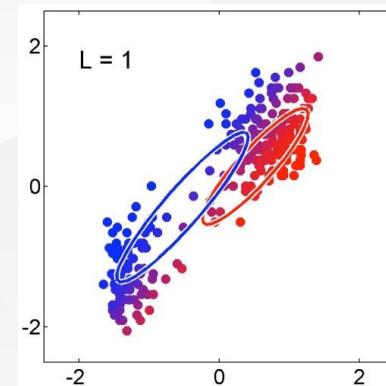
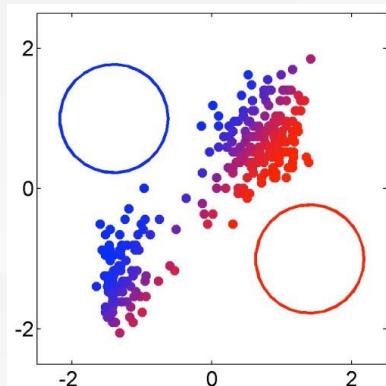
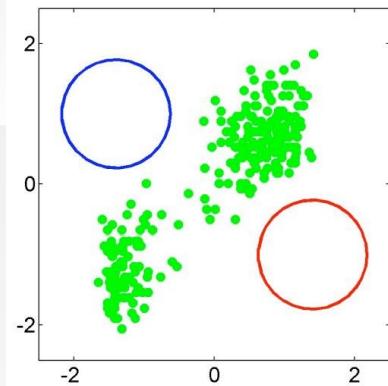
$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^N P(z_i = 1|x_i, \boldsymbol{\theta}^{(t)}) (\ln(\pi) + (x_i \ln(p) + (1 - x_i) \ln(1 - p))) \\ + \sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)}) (\ln(1 - \pi) + (x_i \ln(q) + (1 - x_i) \ln(1 - q)))$$

$$0 = \frac{\partial \ln Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})}{\partial q} = \frac{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)}) x_i}{q} - \frac{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)}) (1 - x_i)}{1 - q}$$

$$q = \frac{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)}) x_i}{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(t)})}$$

$$q^{(1)} = \frac{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(0)}) x_i}{\sum_{i=1}^N P(z_i = 0|x_i, \boldsymbol{\theta}^{(0)})} = 0.6$$

➤ 例：



## ➤ 高斯混合模型: 与K-means 的联系

- 高斯混合模型的E步是一个软划分版本的 K-means. :  $r_{i,k} \in [0,1]$
- 高斯混合模型的M步估计除了估计均值外还估计协方差矩阵。
- 当所有 $\pi_k$ 相等 ,  $\Sigma_k = \delta^2 I$  , 当 $\delta^2 \rightarrow 0$ ,  $r_{ik} \rightarrow \{0, 1\}$  , 则两个方法是一致的。

## ➤ K-means vs 高斯混合模型(GMM)

### K-means

- 损失函数: 最小化平方距离的和
- 样本点硬划分到某个簇
- 假定样本属于每个簇的概率相等，且为球形簇

### GMM

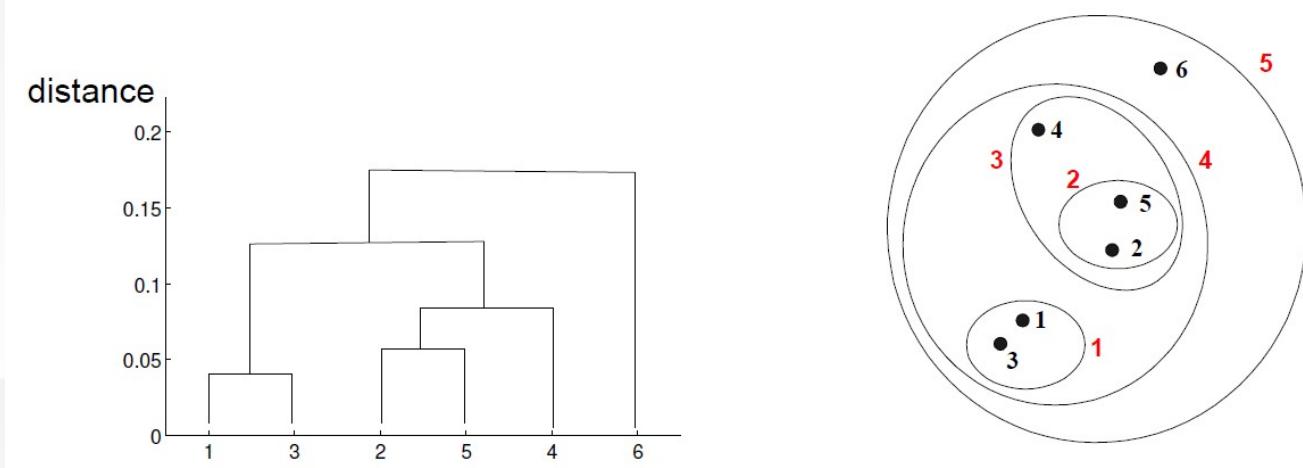
- 最小化负对数似然
- 点到簇的从属关系为软分配
- 可以被用于非球形簇，且各个簇概率不同

# Outline

- 简介
- 距离度量函数
- 聚类性能评价指标
- 聚类算法
  - K均值聚类 ( K-means )
  - 高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )
- 层次聚类
- 基于密度的聚类

## ➤ 层次聚类

- 产生树形嵌套的聚类簇
- 可以被可视化为树状图(dendrogram)
  - 树形的示意图，记录了簇合并或分割的序列



## ➤ 层次聚类的优点

### ■ 不需要提前假定聚类的簇数

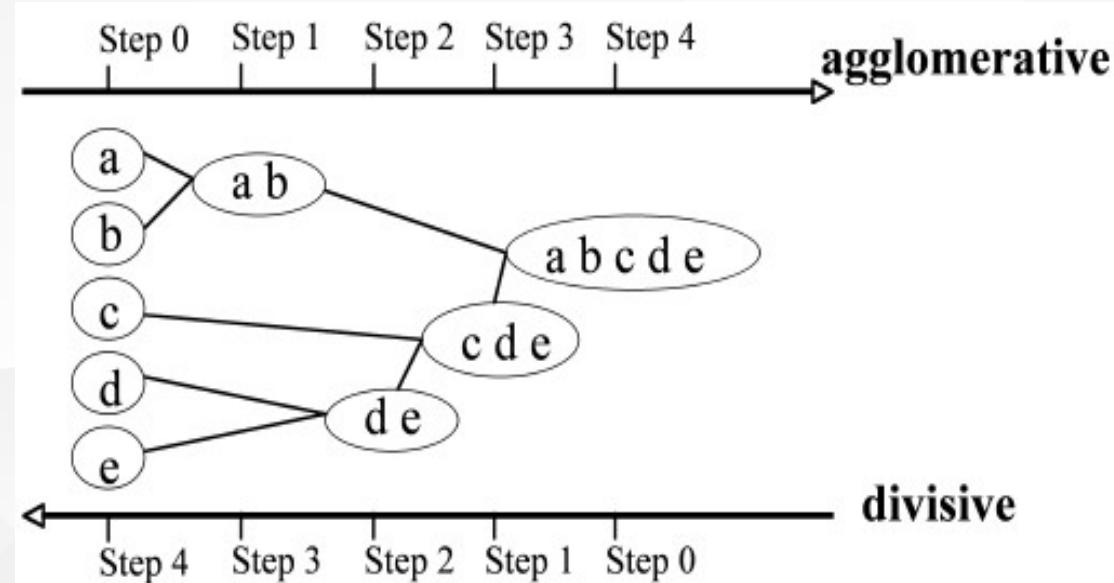
- 通过选择树状图的某一层可以获得任意簇数量的聚类结构

### ■ 聚类结果可能对应着有意义的分类体系

- 例如在生物科学中 (e.g., 门纲目科, 人类种系, ...)

## ➤ 层次聚类

- 自底向上（凝聚式）：递归的合并相似度最高/距离最近的两个簇
- 自顶向下（分裂式）：递归地分裂最不一致的簇（例如：具有最大直径的簇）
- 用户可以在层次化的聚类中选择一个分割，得到一个最自然的聚类结果（例如：各个簇的簇间相似性高于一定阈值）



## ➤ 凝聚式聚类算法

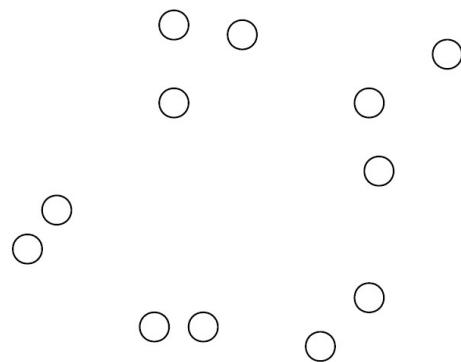
- 相较于分裂式，凝聚式是更加流行的层次聚类技术
- 基本算法非常直观

1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4.     Merge the two closest clusters
5.     Update the proximity matrix
6. Until only a single cluster remains

- 关键是如何计算簇之间的相似度 (proximity) → 不同的定义簇间距离的方法，将得到不同的聚类算法

## ➤ 起始状态

- 开始时每个点是一个簇，计算一个相似度矩阵



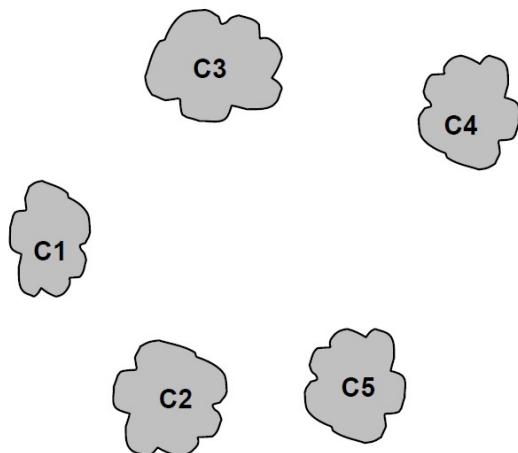
|    | p1 | p2 | p3 | p4 | p5 | .. |
|----|----|----|----|----|----|----|
| p1 |    |    |    |    |    |    |
| .  |    |    |    |    |    |    |

**Proximity Matrix**



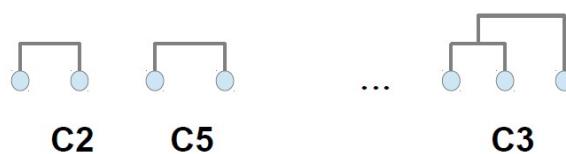
## ➤ 中间过程

■ 经过一些合并步骤后，我们可以获得一些簇



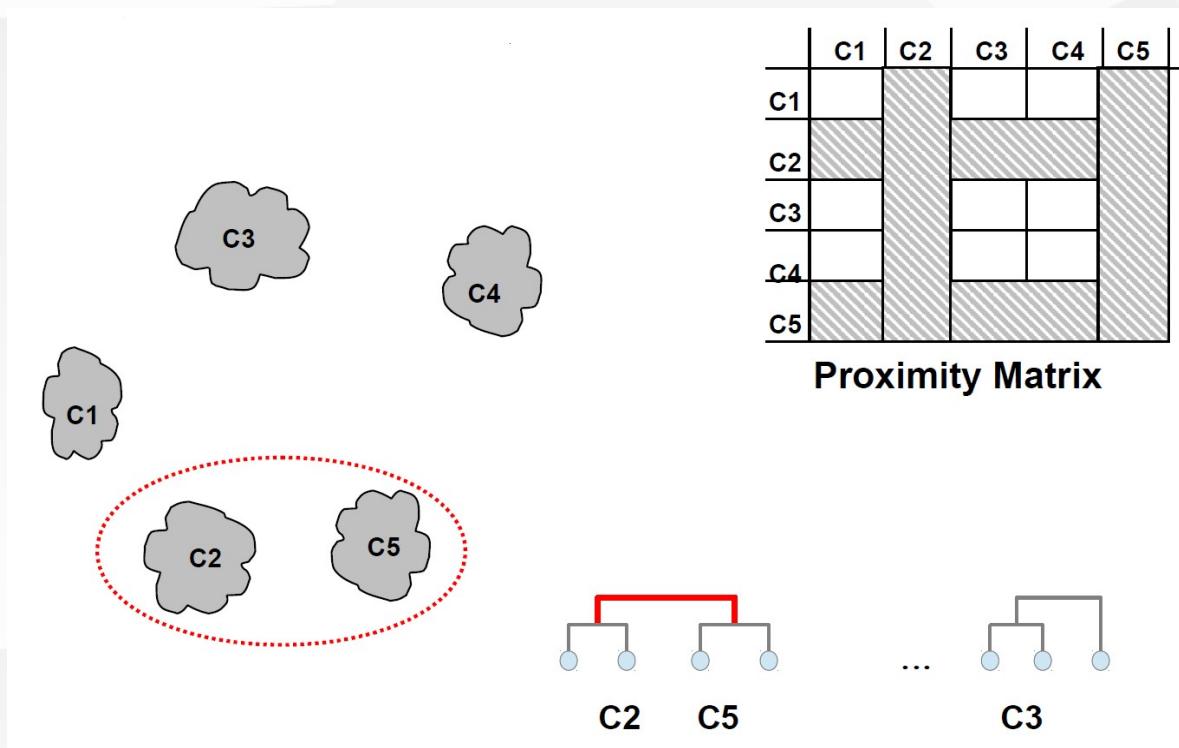
|    | c1 | c2 | c3 | c4 | c5 |
|----|----|----|----|----|----|
| c1 |    |    |    |    |    |
| c2 |    |    |    |    |    |
| c3 |    |    |    |    |    |
| c4 |    |    |    |    |    |
| c5 |    |    |    |    |    |

Proximity Matrix



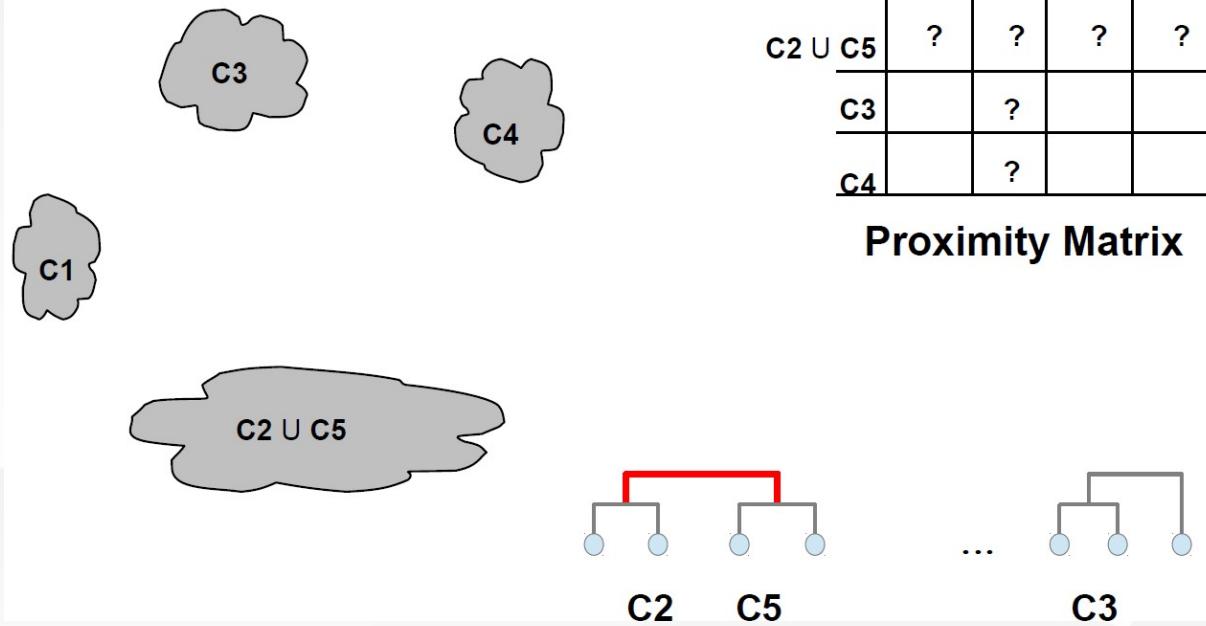
## ➤ 中间过程

- 我们合并最近的两个簇 (C2 和 C5) , 并更新相似度矩阵



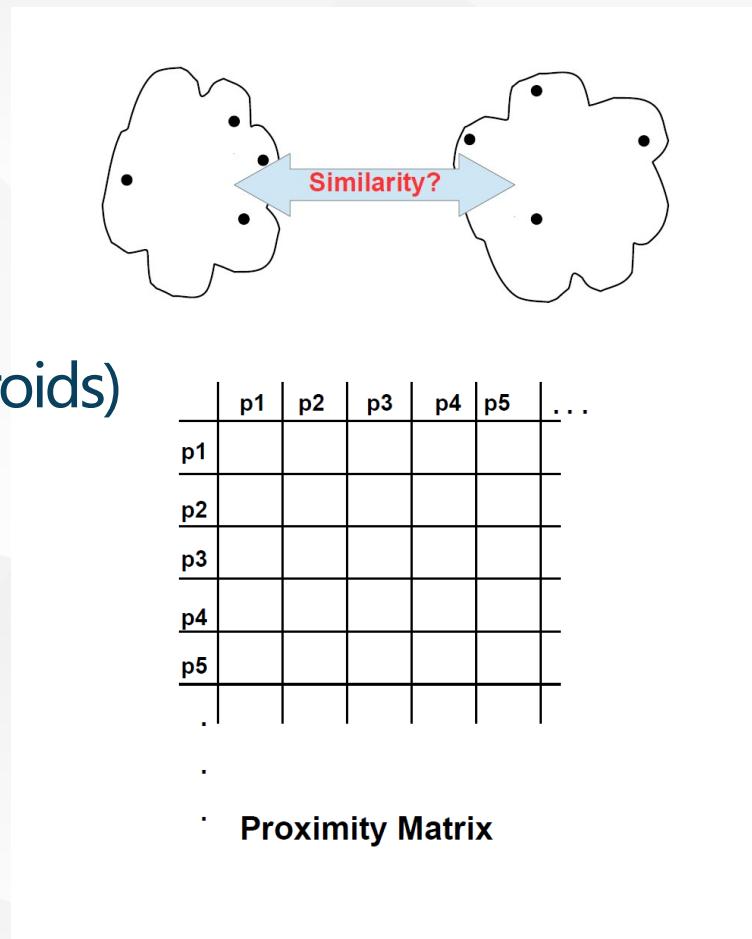
## ➤ 合并后

■ 问题在于 “我们如何更新相似度矩阵？”



## ➤ 如何定义簇间相似性

- 最小距离(MIN)
- 最大距离(MAX)
- 平均距离(Group Average)
- 中心点距离 (Distance Between Centroids)
- 其他由某种目标函数推导出来的方法
  - Ward' s 方法使用平方误差

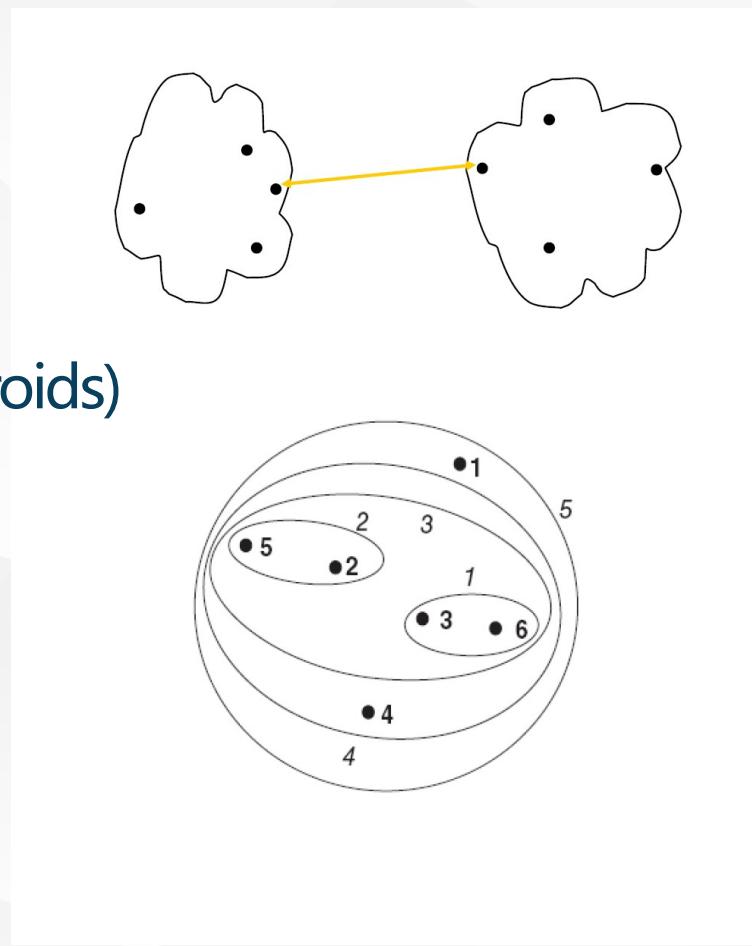


## ➤ 如何定义簇间相似性

- 最小距离(MIN) (Single link)
- 最大距离(MAX)
- 平均距离(Group Average)
- 中心点距离 (Distance Between Centroids)
- 其他由某种目标函数推导出来的方法
  - Ward' s 方法使用平方误差

优势: 可形成非球形、非凸的簇

问题: 链式效应

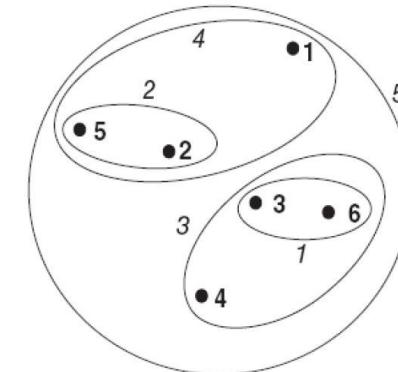
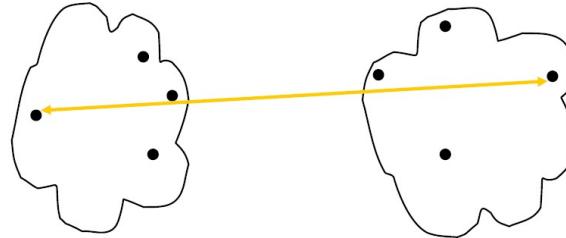


## ➤ 如何定义簇间相似性

- 最小距离(MIN)
- 最大距离(MAX) (complete link)
- 平均距离(Group Average)
- 中心点距离(Distance Between Centroids)
- 其他由某种目标函数推导出来的方法
  - Ward' s 方法使用平方误差

优势: 对噪声更加鲁棒 (不成链)

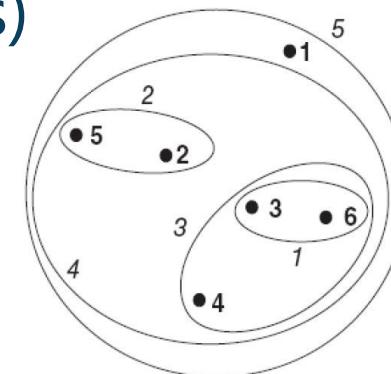
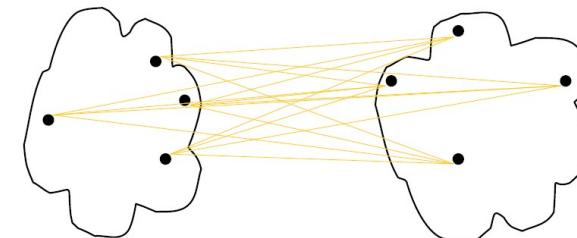
问题: 趋向于拆开大的簇,偏好球形(globular/round)簇



## ➤ 如何定义簇间相似性

- 最小距离(MIN)
- 最大距离(MAX)
- 平均距离 (Group Average)(average-linkage)
- 中心点距离 (Distance Between Centroids)
- 其他由某种目标函数推导出来的方法
  - Ward' s 方法使用平方误差

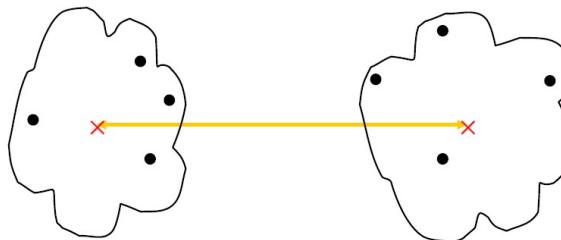
MIN 和 MAX 的折中方案



## 如何定义簇间相似性

- 最小距离(MIN)
  - 最大距离(MAX)
  - 平均距离(Group Average)
  - 中心点距离 (Distance Between Centroids)
  - 其他由某种目标函数推导出来的方法
    - Ward' s 方法使用平方误差

问题: 反向效应 (后边合并的簇间距离可能 比之前合并的簇间距离更近)



|    | p1 | p2 | p3 | p4 | p5 | ... |
|----|----|----|----|----|----|-----|
| p1 |    |    |    |    |    |     |
| p2 |    |    |    |    |    |     |
| p3 |    |    |    |    |    |     |
| p4 |    |    |    |    |    |     |
| p5 |    |    |    |    |    |     |

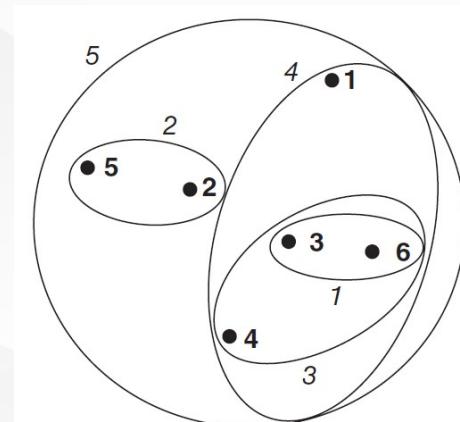
## Proximity Matrix

## ➤ 如何定义簇间相似性

- 最小距离(MIN)
- 最大距离(MAX)
- 平均距离(Group Average)
- 中心点距离 (Distance Between Centroids)
- 其他由某种目标函数推导出来的方法
  - Ward' s 方法使用平方误差

两个簇的相似性基于两个簇融合后平方误差的增加

- 更少受噪声和离群点影响
- 倾向于球形簇
- K-means的层次化版本
  - 可以用于初始化K-means



## ➤ 层次聚类的限制

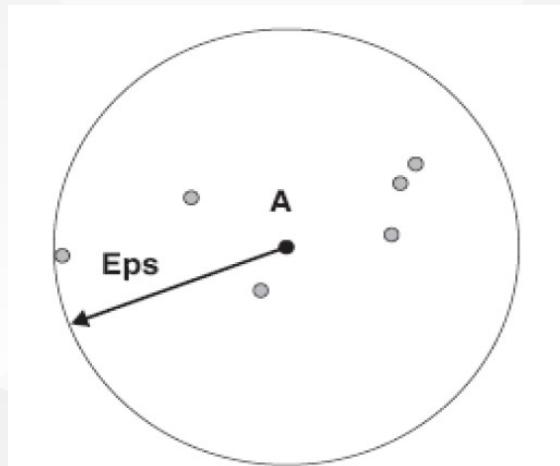
- 贪心:一旦簇被合并或者拆分，过程不可逆
- 没有优化一个全局的目标函数
- 不同方法存在一个或多个以下问题:
  - 对噪声和离群点敏感
  - 比较难处理不同尺寸的簇和凸的簇
  - 成链, 误把大簇分裂

# Outline

- 简介
- 距离度量函数
- 聚类性能评价指标
- 聚类算法
  - K均值聚类 ( K-means )
  - 高斯混合模型和EM算法 ( Gaussian Mixture Models and EM Algorithm )
- 层次聚类
- **基于密度的聚类**

## ➤ DBSCAN

- DBSCAN ( density-based spatial clustering of application with Noise )
- 密度(Density) = 给定半径 ( $\varepsilon$ )内点的个数

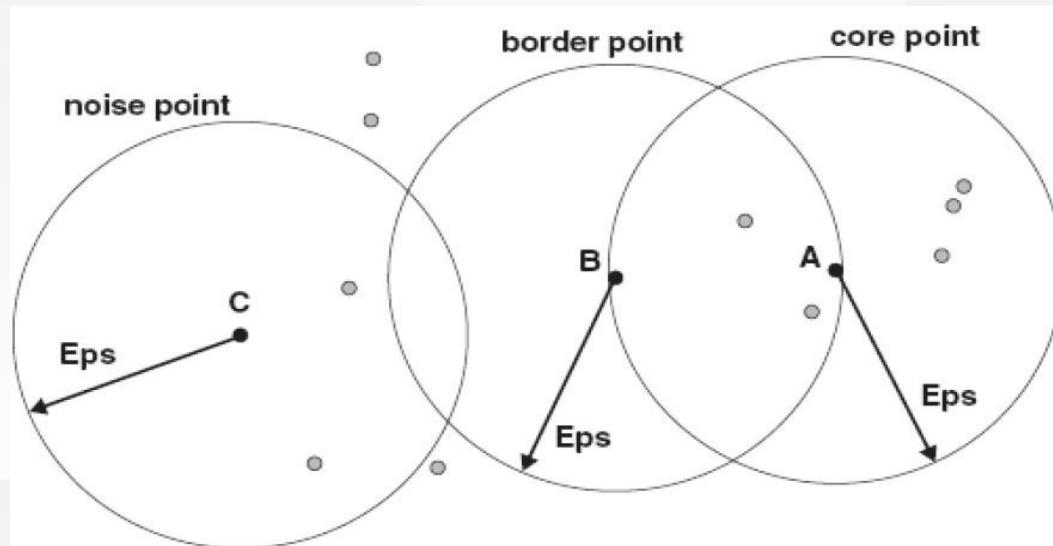


Density = 7 points

Ester et al. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD). 1996.

## ➤ 预备知识

- 核心点 ( Core point ) : 指定半径 $\varepsilon$ 内多于指定数量 $MinPts$ 个点
- 边界点 ( Border point ) : 半径 $\varepsilon$ 内有少于 $MinPts$ 个点，但在某个核心点的邻域内
- 噪声点 ( Outliers ) : 核心点和边界点之外的点.



## ➤ 预备知识

- 点 $q$ 由点 $p$ 密度可达：连接两个点的路径上所有的点都是核心点
  - 如果 $p$ 是核心点，那么由它密度可达的点形成一个簇
- 点 $q$ 和点 $p$ 是密度相连的，如果存在点 $o$ 从其密度可达点 $q$ 和点 $p$
- 聚类的簇满足以下两个性质：
  - 连接性：簇内的任意两点点是密度相连的；
  - 最大性：如果一个点从一个簇中的任意一点密度可达，则该点属于该簇

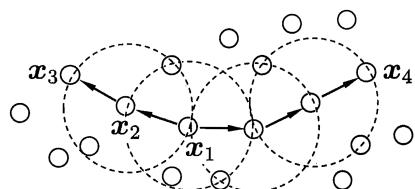


图 9.8 DBSCAN 定义的基本概念( $MinPts = 3$ )：虚线显示出  $\epsilon$ -邻域， $x_1$  是核心对象， $x_2$  由  $x_1$  密度直达， $x_3$  由  $x_1$  密度可达， $x_4$  与  $x_1$  密度相连。

# DBSCAN 算法

输入: 样本集  $D = \{x_1, x_2, \dots, x_m\}$ ;  
邻域参数  $(\epsilon, MinPts)$ .

过程:

```
1: 初始化核心对象集合:  $\Omega = \emptyset$ 
2: for  $j = 1, 2, \dots, m$  do
3:   确定样本  $x_j$  的  $\epsilon$ -邻域  $N_\epsilon(x_j)$ ;
4:   if  $|N_\epsilon(x_j)| \geq MinPts$  then
5:     将样本  $x_j$  加入核心对象集合:  $\Omega = \Omega \cup \{x_j\}$ 
6:   end if
7: end for
8: 初始化聚类簇数:  $k = 0$ 
9: 初始化未访问样本集合:  $\Gamma = D$ 
10: while  $\Omega \neq \emptyset$  do
11:   记录当前未访问样本集合:  $\Gamma_{old} = \Gamma$ ;
12:   随机选取一个核心对象  $o \in \Omega$ , 初始化队列  $Q = < o >$ ;
13:    $\Gamma = \Gamma \setminus \{o\}$ ;
14:   while  $Q \neq \emptyset$  do
15:     取出队列  $Q$  中的首个样本  $q$ ;
16:     if  $|N_\epsilon(q)| \geq MinPts$  then
17:       令  $\Delta = N_\epsilon(q) \cap \Gamma$ ;
18:       将  $\Delta$  中的样本加入队列  $Q$ ;
19:        $\Gamma = \Gamma \setminus \Delta$ ;
20:     end if
21:   end while
22:    $k = k + 1$ , 生成聚类簇  $C_k = \Gamma_{old} \setminus \Gamma$ ;
23:    $\Omega = \Omega \setminus C_k$ 
24: end while
输出: 簇划分  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ 
```

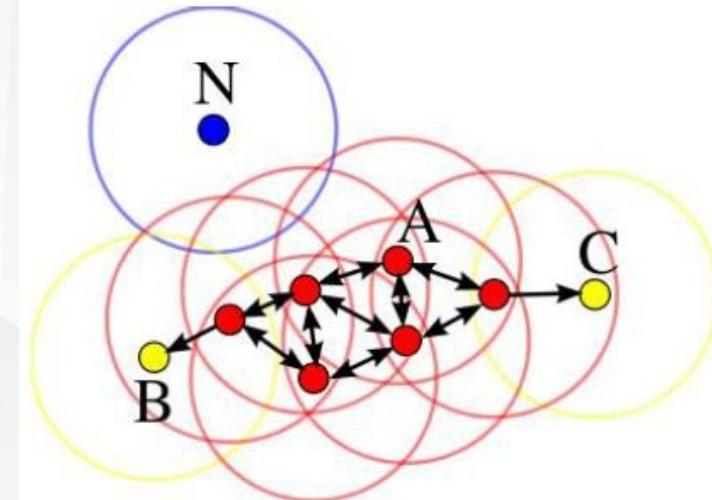


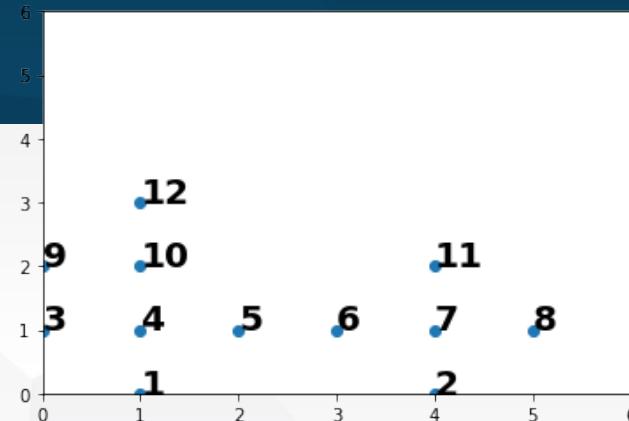
图 9.9 DBSCAN 算法

# DBSCAN算法举例

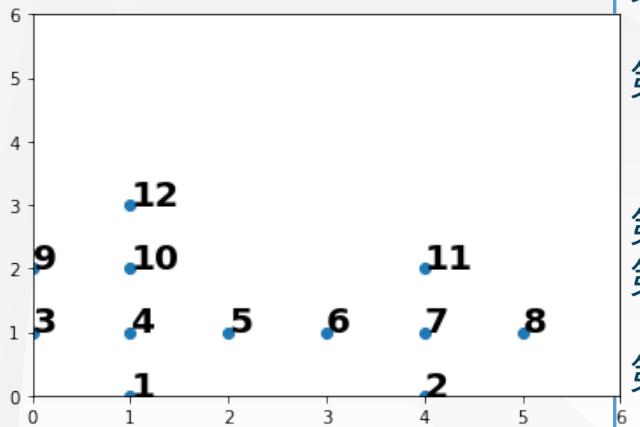
- 训练数据如左所示，样本数 $N = 12$
- DBSCAN算法参数： $\varepsilon = 1$ ， $K = 4$

| 序号 | 属性 1 | 属性 2 |
|----|------|------|
| 1  | 1    | 0    |
| 2  | 4    | 0    |
| 3  | 0    | 1    |
| 4  | 1    | 1    |
| 5  | 2    | 1    |
| 6  | 3    | 1    |
| 7  | 4    | 1    |
| 8  | 5    | 1    |
| 9  | 0    | 2    |
| 10 | 1    | 2    |
| 11 | 4    | 2    |
| 12 | 1    | 3    |

| 步骤 | 选择的点 | 在 $\varepsilon$ 中点的个数 | 通过计算可达点而找到的新簇                     |
|----|------|-----------------------|-----------------------------------|
| 1  | 1    | 2                     | 无                                 |
| 2  | 2    | 2                     | 无                                 |
| 3  | 3    | 3                     | 无                                 |
| 4  | 4    | 5                     | 簇 $C_1$ : {1, 3, 4, 5, 9, 10, 12} |
| 5  | 5    | 3                     | 已在一个簇 $C_1$ 中                     |
| 6  | 6    | 3                     | 无                                 |
| 7  | 7    | 5                     | 簇 $C_2$ : {2, 6, 7, 8, 11}        |
| 8  | 8    | 2                     | 已在一个簇 $C_2$ 中                     |
| 9  | 9    | 3                     | 已在一个簇 $C_1$ 中                     |
| 10 | 10   | 4                     | 已在一个簇 $C_1$ 中,                    |
| 11 | 11   | 2                     | 已在一个簇 $C_2$ 中                     |
| 12 | 12   | 2                     | 已在一个簇 $C_1$ 中                     |



## ➤ DBSCAN算法举例（cont.）



第1步，在数据库中选择一点1，由于在以它为圆心的，以1为半径的圆内包含2个点（小于4），因此它不是核心点，选择下一个点。

第2步，在数据库中选择一点2，由于在以它为圆心的，以1为半径的圆内包含2个点，因此它不是核心点，选择下一个点。

第3步，在数据库中选择一点3，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。

第4步，在数据库中选择一点4，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点（直接可达4个，间接可达2个），聚出的新类{1, 3, 4, 5, 9, 10, 12}，选择下一个点。

第5步，在数据库中选择一点5，已经在簇1中，选择下一个点。

第6步，在数据库中选择一点6，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。

第7步，在数据库中选择一点7，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点，聚出的新类{2, 6, 7, 8, 11}，选择下一个点。

第8步，在数据库中选择一点8，已经在簇2中，选择下一个点。

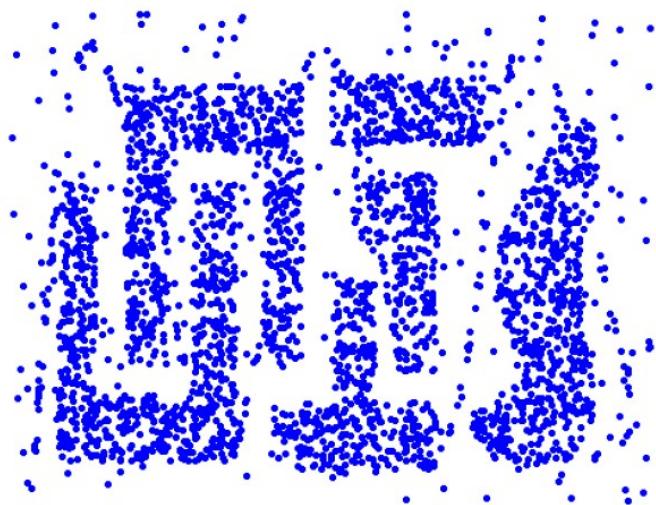
第9步，在数据库中选择一点9，已经在簇1中，选择下一个点。

第10步，在数据库中选择一点10，已经在簇1中，选择下一个点。

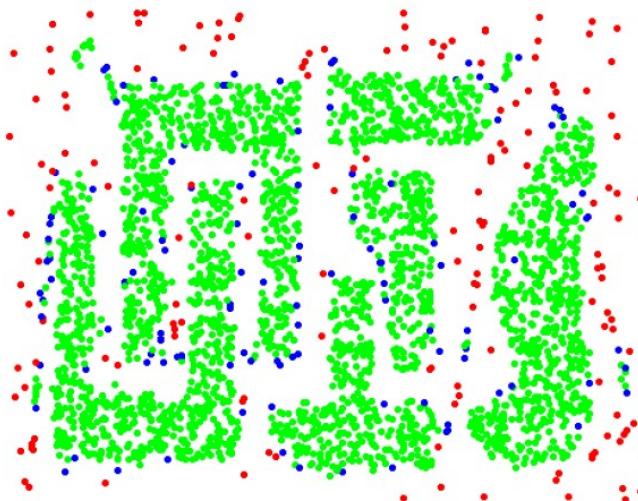
第11步，在数据库中选择一点11，已经在簇2中，选择下一个点。

第12步，选择12点，已经在簇1中，由于这已经是最后一点所有点都以处理，程序终止。

## ➤ DBSCAN : 核心点, 边界点和噪声点



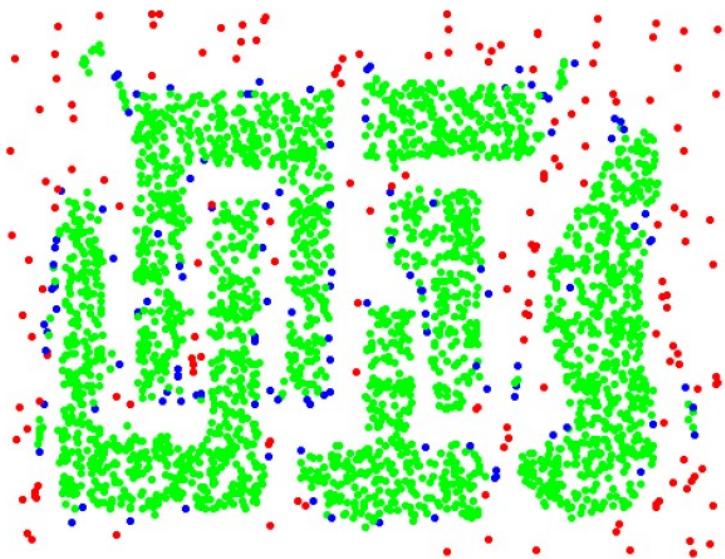
Original Points



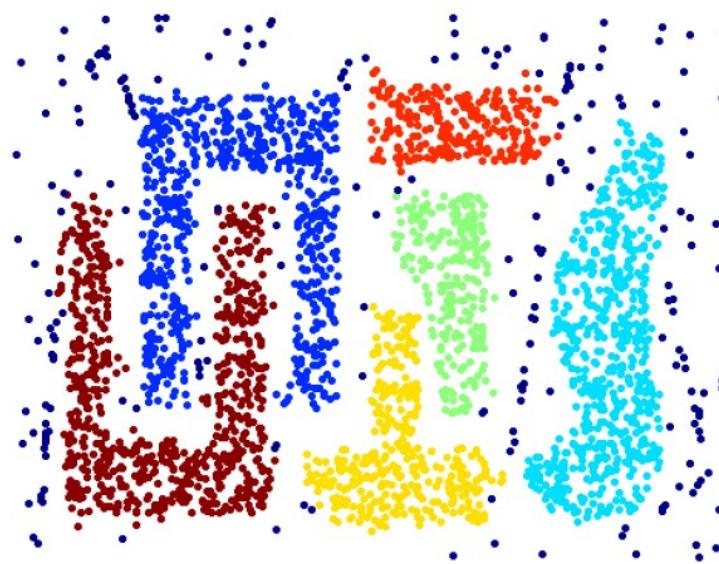
Point types: **core**,  
**border** and **noise**

Eps = 10, MinPts = 4

## ➤ DBSCAN : 确定簇



Point types: **core**,  
**border** and **noise**



Clusters

## ➤ 对DBSCAN的分析

### ■ 优势

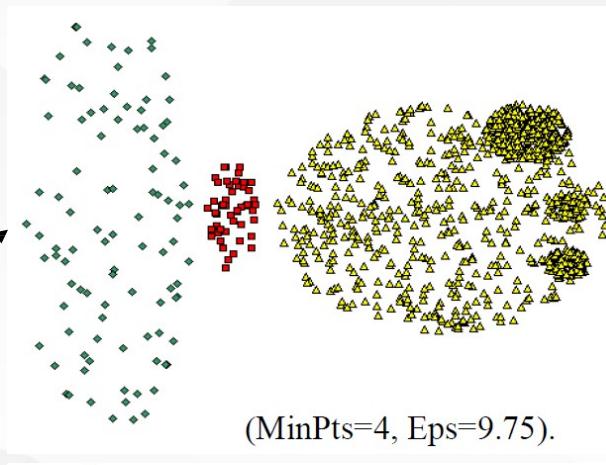
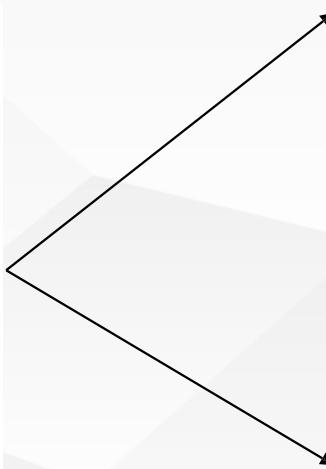
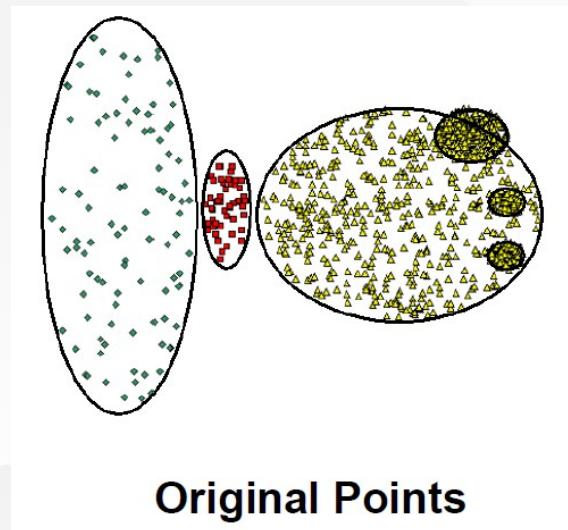
- 不需要明确簇的数量
- 任性形状的簇
- 对离群点(outliers)较为鲁棒

### ■ 劣势

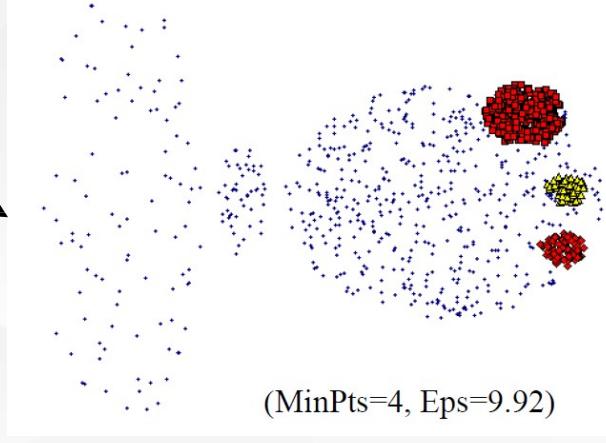
- 参数选择比较困难( $MinPts, \varepsilon$ )
- 不适合密度差异较大的数据集
- 需计算每个样本的邻居点，操作耗时 ( $O(N^2)$ )；即使采用k-d tree索引等技术加速计算，算法的时间复杂也为 $ONlog(N)$ 。

## ➤ DBSCAN 什么时候表现不好

- 密度变化
- 高维数据



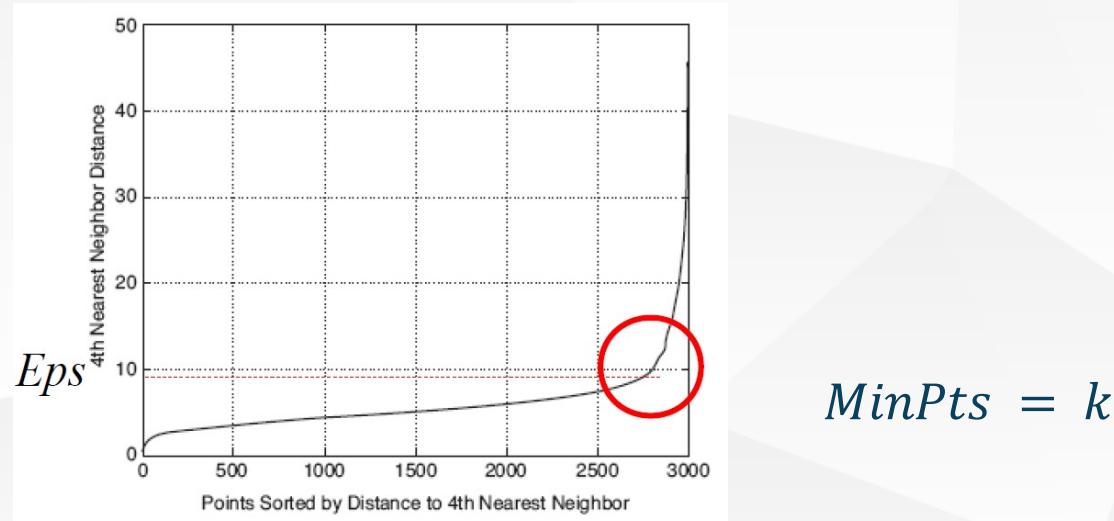
( $\text{MinPts}=4$ ,  $\text{Eps}=9.75$ ).



( $\text{MinPts}=4$ ,  $\text{Eps}=9.92$ )

## ➤ DBSCAN : 如何确定 $\varepsilon$ 和 $MinPts$

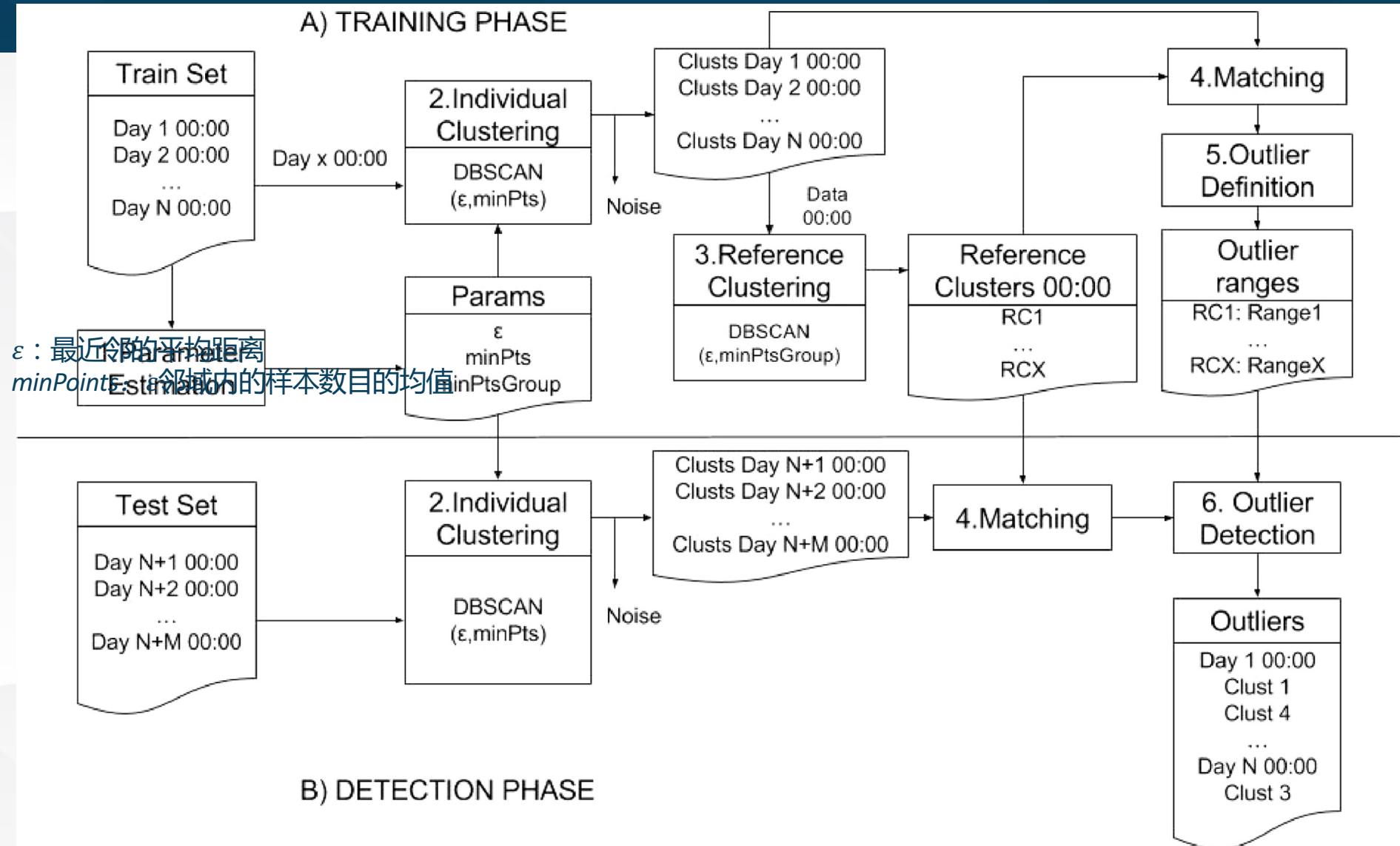
- 直观想法：同一个簇内的点，它们第 $k$ 个最近邻大约相同距离
- 噪声点到其第 $k$ 最近邻距离较远
- 方法：画出每个点到其第 $k$ 最近邻的距离



## ➤ 例：采用DBSCAN聚类分析城市异常事件

- 输入数据：基于位置的社交网络（LBSN）
- 输出：特定时间检测特定区域是否出现异常情况
- 第一阶段：每周每天每时某地的密度（城市 $7 \times 24$ 小时的脉搏）
  - 纽约市6个月的LBSN数据
  - 聚类，得到正常状态的聚类结果（参考聚类结果）
- 第二阶段：收集某地的LBSN数据，检测异常情况
  - 对测试数据进行聚类，聚类结果与训练好的参考聚类结果比较

Sensing the city with Instagram: Clustering geolocated data for outlier detection, 2017



## ➤ 快速寻找密度峰值的聚类

■ Rodriguez, Alex, and Alessandro Laio. *Clustering by fast search and find of density peaks*. Science 344.6191 (2014): 1492-1496.

■ 假设：

- **密度**：聚类中心的密度要大，即围绕着该中心点的样本应要尽可能多
- **距离**：与其他聚类中心的距离要尽可能的远

## ➤ Cluster Center

### ■ $\rho_i$ : 点*i*的局部密度 ( Local density )

- 截断核：  $\rho_i = \sum_{j \in \mathcal{D} \setminus i} \chi(d_{ij} - d_c)$ ，其中  $\chi(x) = \begin{cases} 1 & x < 0 \\ 0 & otherwise \end{cases}$

- $d_c$  为截止距离
- $\rho_i$  为到点*i* 距离小于  $d_c$  的点的数目

- Gaussian核：  $\rho_i = \sum_{j \in \mathcal{D} \setminus i} e^{-\frac{d_{ij}}{d_c}}$

■ 类似以点*i*为中心，以  $d_c$  为半径画一个圆，计算落在圆里的样本数， Gaussian核更加soft一点，根据距离来表示数据点与中心点的权重。离的越近，权重越高；离得越远，权重就越低。

## ➤ Cluster Center

■  $\delta_i$ : 到其他高密度点的距离 ( 相对距离 )

- $\{q_i\}_{i=1}^N$  为  $\{\rho_i\}_{i=1}^N$  的降序排序的索引 , 即  $\rho_{q_1} \geq \rho_{q_2} \dots \geq \rho_{q_N}$

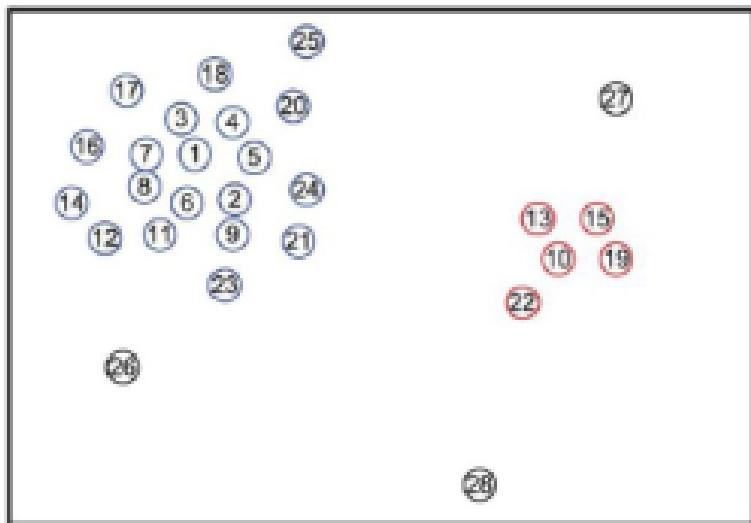
$$\delta_i = \begin{cases} \min_{q_j, j < i} \{d_{q_i q_j}\} & i \geq 2 \\ \min_{j \geq 2} \{\delta_{q_j}\} & i = 1 \end{cases}$$

从所有局部密度大于该点的数据点中 , 离该点最近的距离  
密度最大的那个点 , 除该点之外的局部密度最大值

- 局部密度  $\rho_i$  大 , 且相对距离  $\delta_i$  大的点 , 可以看成是一个聚类中心。

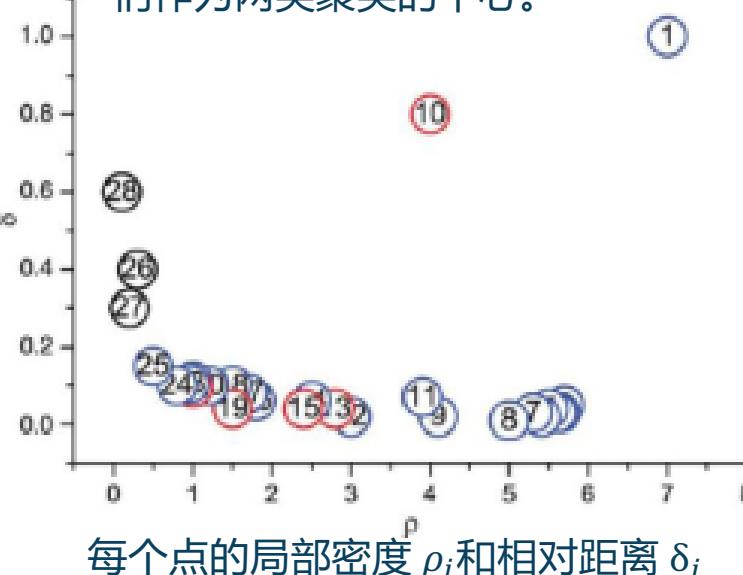
# ➤ Decision Graph

A



B

1号和10号点脱颖而出，因此聚类的时候把他们作为两类聚类的中心。



- (A) point distribution. Data points are ranked in order of decreasing density.
- (B) **Decision graph**. Different colors corresponds to difference clusters.

## ➤ Cluster Numbers

- $\gamma_i = \rho_i \delta_i$
- $\gamma$ 越大，越可能成为聚类中心。

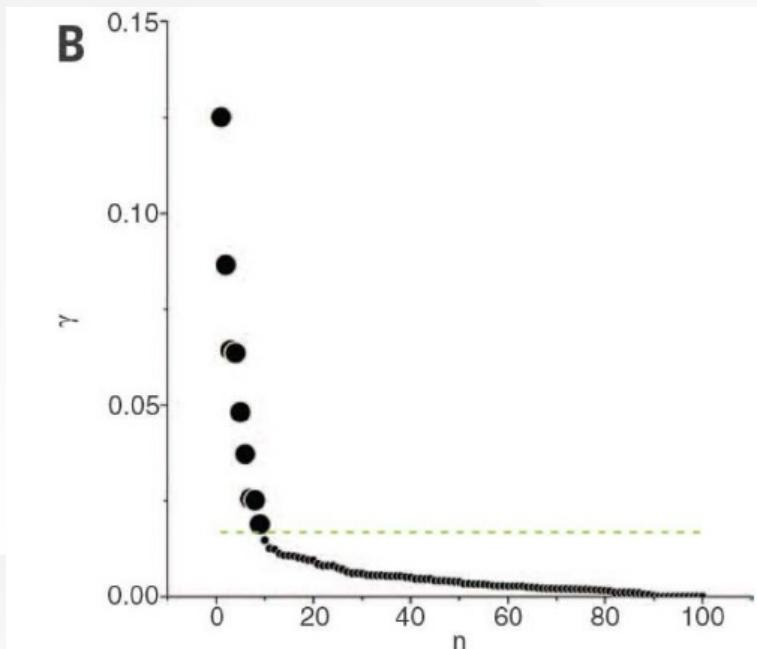
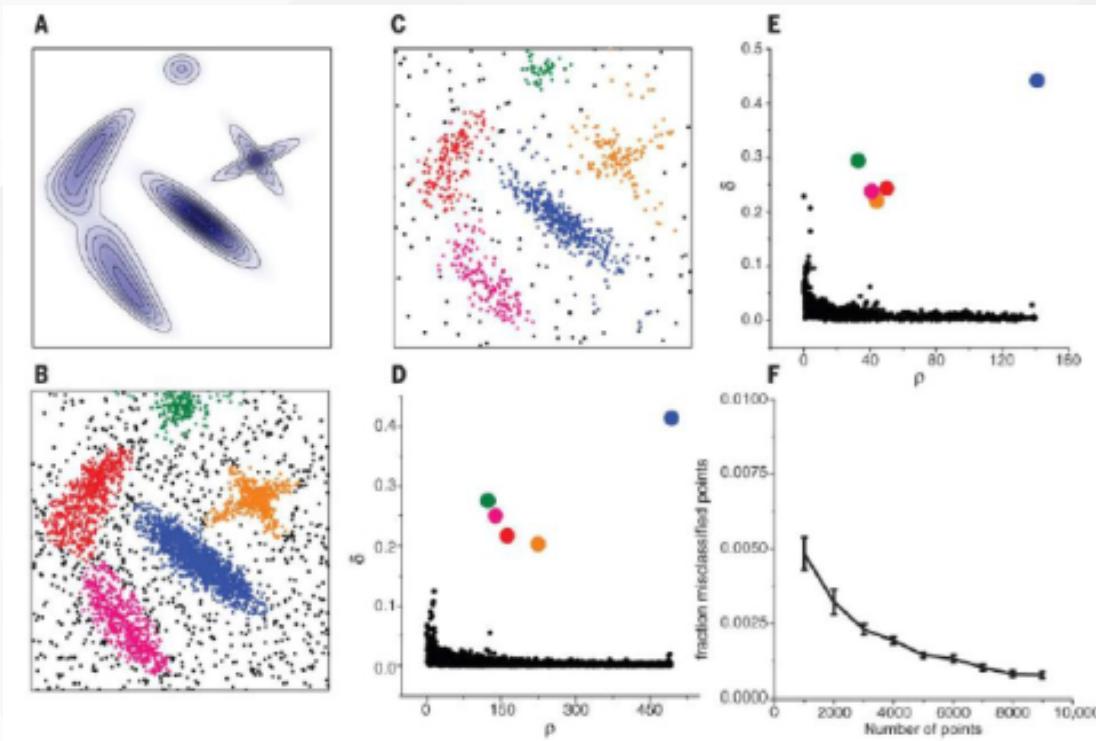


图 3 降序排列的  $\gamma$  值示意图

非聚类中心的 $\gamma$ 值比较平滑  
从非聚类中心到聚类中心的 $\gamma$ 值有一个明显的跳跃

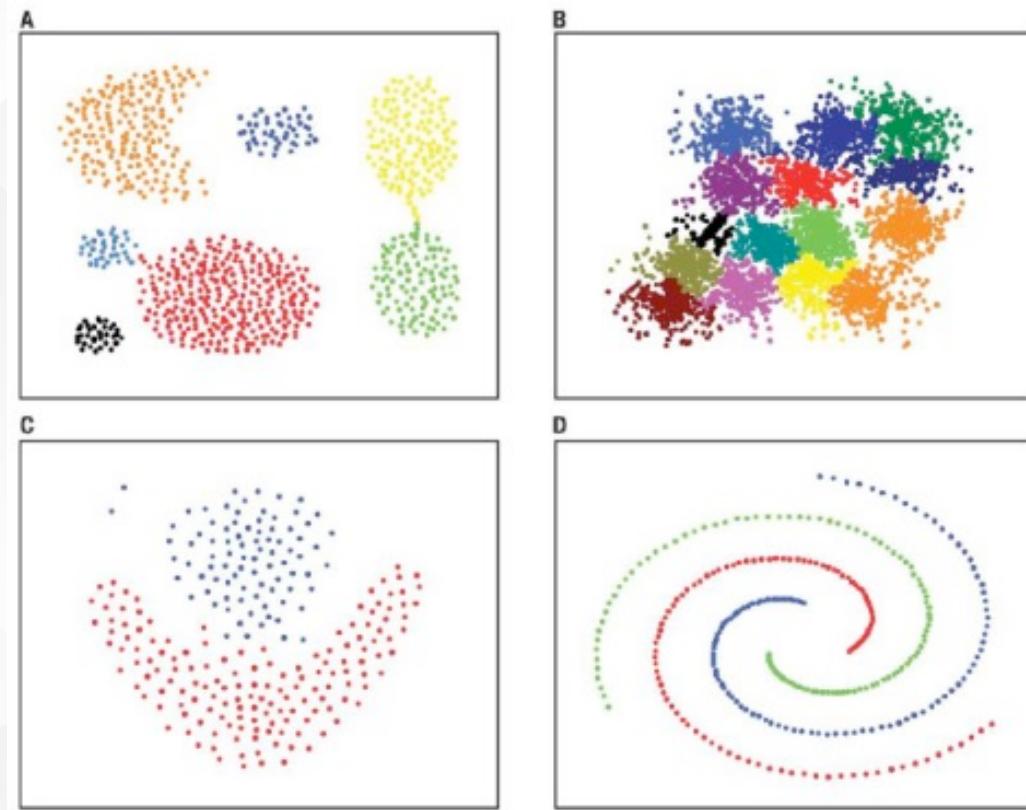
## ➤ Experimental Results



一个非球形类分布图，加入黑色噪音点

- A : 类的概率分布；B、C : 4000个和1000个样本点；
- D、E : decision graph，可以明显看出类别中心及个数；
- F : 随着样本数目增加，错误指派点的比率。

## ➤ Experimental Results



<https://blog.csdn.net/u011089523/article/details/76586072>

<https://yq.aliyun.com/ziliao/417016>

<http://www.cnblogs.com/nolonely/p/6964852.html>

## » 一些其他的聚类算法

### ■ 基于中心的聚类

- Fuzzy c-means
- PAM (Partitioning Around Medoids)

### ■ 层次化

- CURE (Clustering Using Representatives): shrinks points toward center
- BIRCH (balanced iterative reducing and clustering using hierarchies)

### ■ 基于图的聚类

- Graph partitioning on a sparsified proximity graph
- Shared nearest-neighbor (SNN graph)

### ■ 谱聚类

### ■ 子空间聚类

### ■ 数据流聚类

### ■ 协同聚类

## ➤ 基于深度学习的聚类算法

利用非线性的特征转换去学习一个更好的聚类特征表示

- 基于**AutoEncoder**
  - 学习一个encoder和一个decoder
  - encoder用来映射原始数据到表示
  - decoder从表示重建原始数据
- 基于**CDNN**
  - CNN , FCN , DBN
  - 非常深的网络可以被预训练
  - 训练的损失函数为某些聚类损失
- 基于**VAE**
  - 深度生成模型、AE的生成式版本
  - 恢复数据的真实结构，生成样本
  - GMM作为隐表达的一个loss
- 基于**GAN**
  - 流行的深度生成模型
  - 生成和判别

A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture  
<https://xifengguo.github.io/papers/Access-survey.pdf>

## ➤ 聚类前沿研究

### ■聚类公平性

- 意义: 聚类主体与人相关
- 多种公平性定义
- 多种聚类算法下的公平性

### ■Multi-View数据聚类

- 数据来自多个源
- 数据的属性不一致 ( text + image + voice )

### ■聚类算法的加速

- 例如DBSCAN++ : 减少密度计算量 , 提升算法应用速度
- 提高密度聚类并行性

## ➤ 参考文献

■ 周志华，机器学习

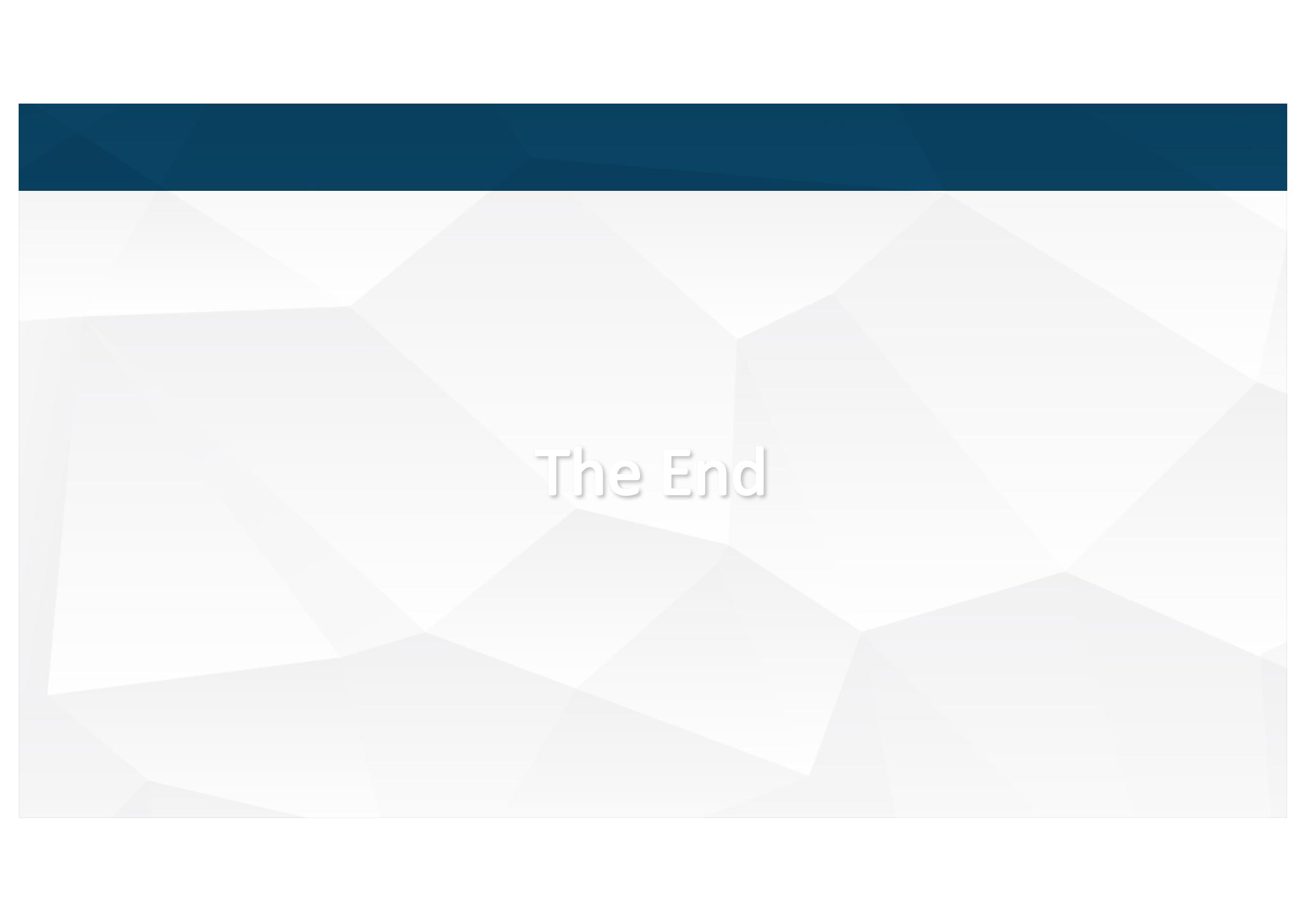
- 第9章

■ 李航，统计学习方法 第二版

- 第13、14、9章

■ Christopher Bishop , Pattern Recognition and Machine Learning

- 第9章



The End