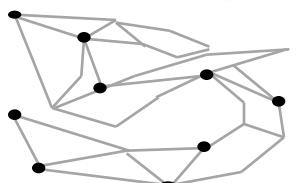


# 聚类 ( Clustering )

数据应用学院

万门大学

Henry Tang



# 概要

---

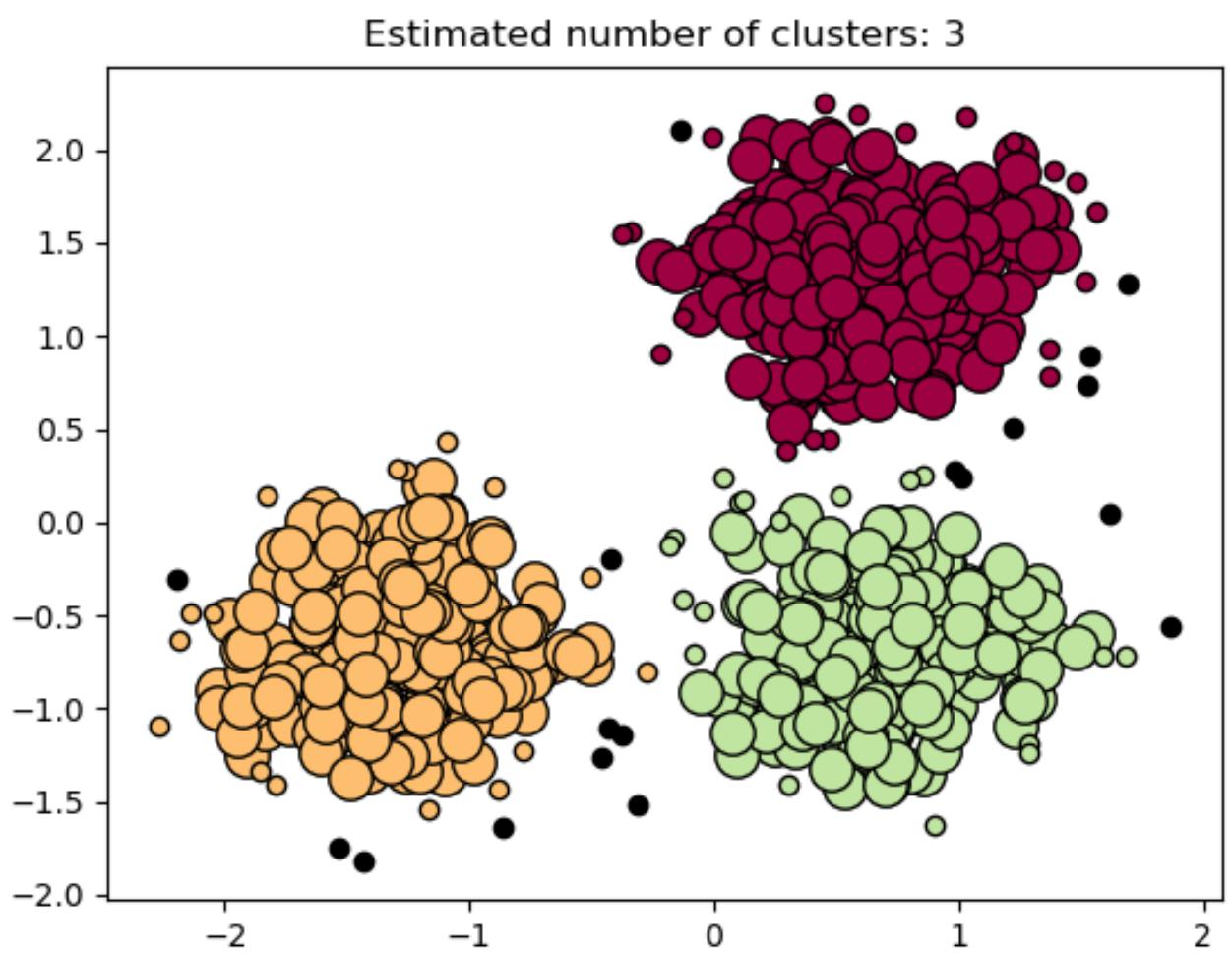
- 聚类简介与历史
- 聚类
  - 基于划分的聚类 (partitioning based clustering) : K均值 (K-means) , Mean shift
  - 层次聚类 (hierarchical clustering): Agglomerative clustering, BIRCH
  - 密度聚类 (density based clustering) : DBSCAN - recommended
  - 基于模型的聚类 (model based clustering) : 高斯混合模型 (GMM)

# 机器学习

- 无监督学习 (Unsupervised learning) : 训练样本的标记信息是未知的, 目标是为了揭露训练样本的内在属性, 结构和信息, 为进一步的数据挖掘提供基础
  - 聚类 (clustering)
  - 降维 (dimensionality reduction)
  - 异常检测 (outlier detection)
  - 推荐系统 (recommendation system)
- 监督学习 (supervised learning) : 训练样本带有信息标记, 利用已有的训练样本信息学习数据的规律预测未知的新样本标签
  - 回归分析 (regression)
  - 分类 (classification)

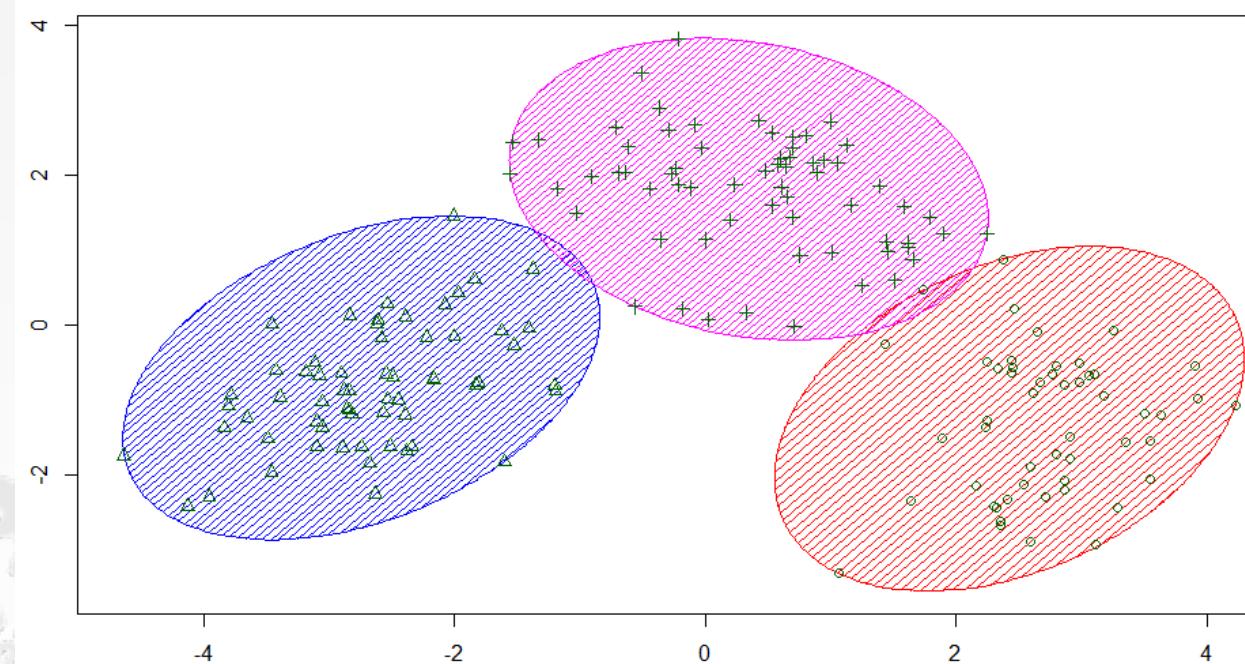
# 聚类

- 聚类：“物以类聚”
  - 按照某一个特定标准（比如距离），把一个数据集分割成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，同时不在同一个簇内的数据对象的差异性也尽可能的大。
- 簇（或类 cluster）：子集合
  - 最大化簇内的相似性
  - 最小化簇与簇之间的相似性
- 聚类可以作为一个单独过程，用于寻找数据内在分布结构，也可以作为其他学习任务前驱过程

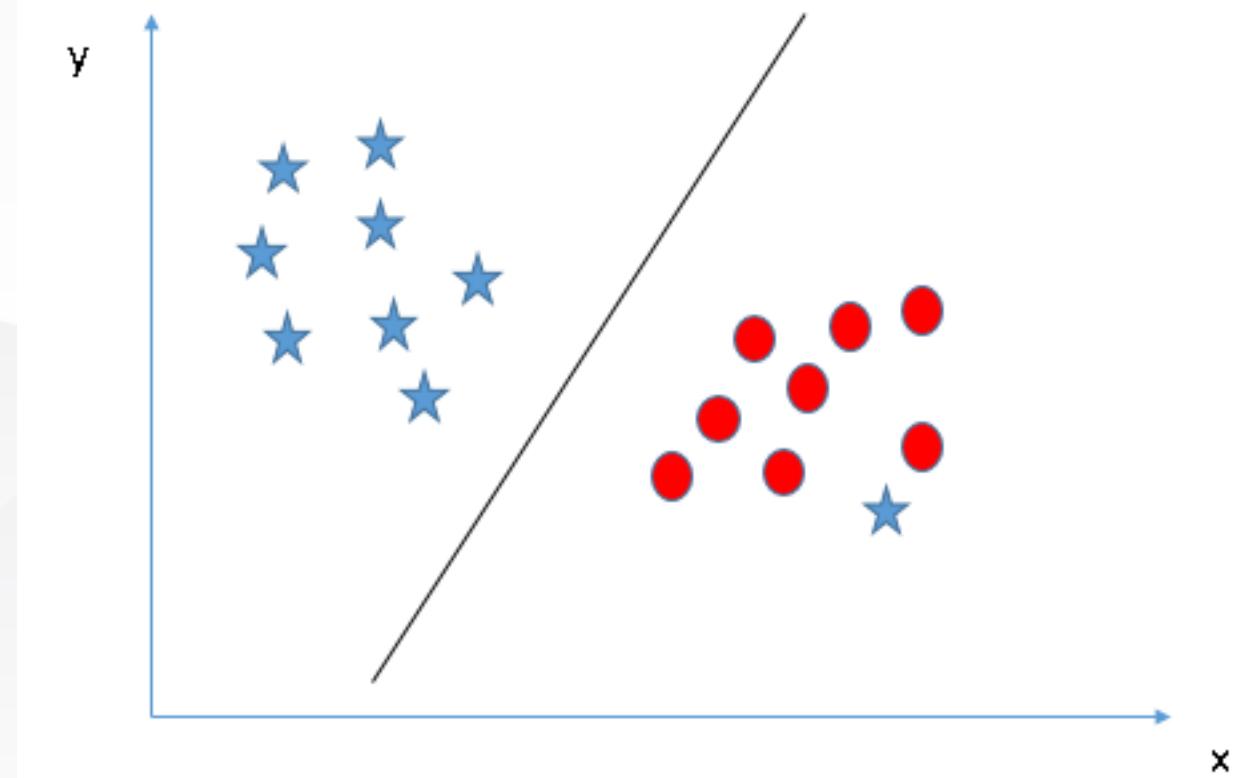


# 聚类和分类的区别

- 聚类和分类的区别：
  - 聚类 (clustering) : 无监督学习任务，不知道真实的样本标记，只把相似度高的样本聚合在一起。
  - 分类 (classification) : 监督学习任务，利用已知的样本标记训练学习器预测未知样本的类别



聚类



分类

# 聚类相似度度量：几何距离

- Minkowski 距离:

$$dist_{mk}(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- 欧式距离 (Euclidean distance) : p = 2 的 Minkowski 距离

$$dist_{mk}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\left( \sum_{i=1}^n |x_i - y_i|^2 \right)}$$

- 曼哈顿距离 (Manhattan distance) : p = 1 的 Minkowski 距离

$$dist_{mk}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^n |x_i - y_i|$$

- 夹角余弦:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}}$$

- 相关系数 (Pearson correlation coefficient) :

$$corr(\mathbf{x}, \mathbf{y}) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

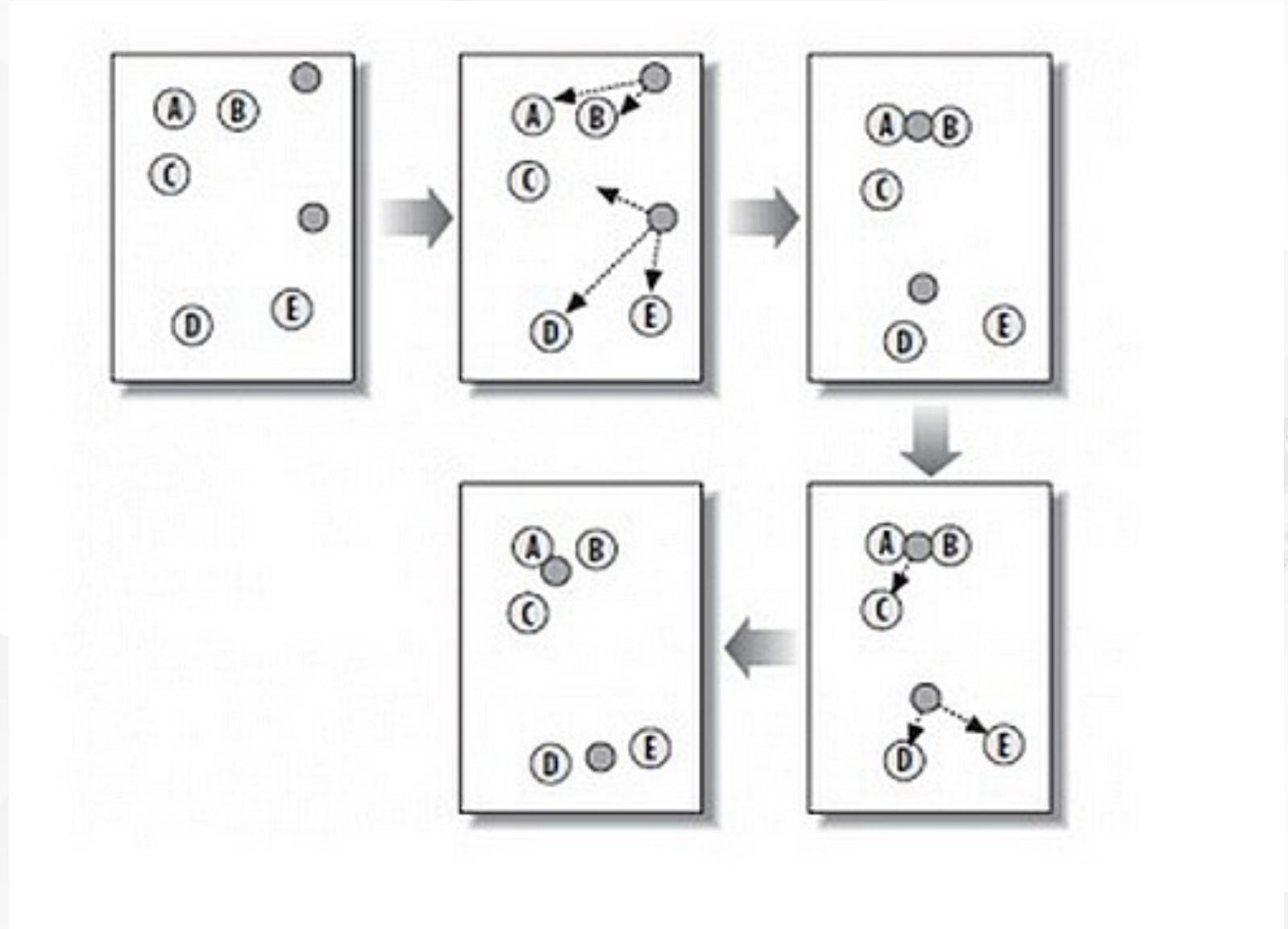
# 聚类类别

- 基于划分的聚类 (partitioning based clustering) : K均值 (K-means) , Mean shift
- 层次聚类 (hierarchical clustering): Agglomerative clustering, BIRCH
- 密度聚类 (density based clustering) : DBSCAN
- 基于模型的聚类 (model based clustering) : 高斯混合模型 (GMM)
- Affinity propagation
- Spectral clustering

# 划分聚类 (partition based clustering)

## K-Means algorithm

- 给定包含N个点的数据集，划分法将构造K个分组
- 每个分组代表一个聚类，这里每个分组至少包含一个数据点，每个数据点属于且仅属于一个分组。
- 对于给定的K值，算法先给出一个初始的分组方法，然后通过反复迭代的方法改变分组，直到准则函数收敛



# K 均值算法 (Kmeans)

- 给定样本集  $D = \{ \mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_m \}$  K均值算法针对聚类所得簇:  $C = \{ C_1, C_2 \dots C_k \}$
- 最小化平方差 :

$$E = \sum_{i=1}^k \sum_{x \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|_2^2$$

其中 :  $\boldsymbol{\mu}_i = \frac{1}{C_i} \sum_{x \in C_i} \mathbf{x}$  是簇  $C_i$  的质心

具体的K均值算法过程:

- 1) 首先, 随机地选择  $k$  个对象, 每个对象初始地代表了一个簇的质心, 即选择  $K$  个初始质心
- 2) 对剩余的每个对象, 根据其与各簇中心的距离, 将它赋给最近的簇
- 3) 重新计算每个簇的平均值。这个过程不断重复, 直到准则函数 (误差的平方和 SSE 作为全局的目标函数) 收敛, 直到质心不发生明显的变化

# K 均值算法 (Kmeans)

输入： 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_m\}$  聚类簇的数量k

过程：

1. 随机选择k个样本点作为初始簇的质心

2. Repeat :

for j from 1 to m do:

    计算样本  $\mathbf{x}_j$  与k个质心的距离

    根据最近的均值向量将  $\mathbf{x}_j$  归为相应的簇

end for

for i from 1 to k do:

    重新计算更新新的簇质心（根据均值计算）

end for

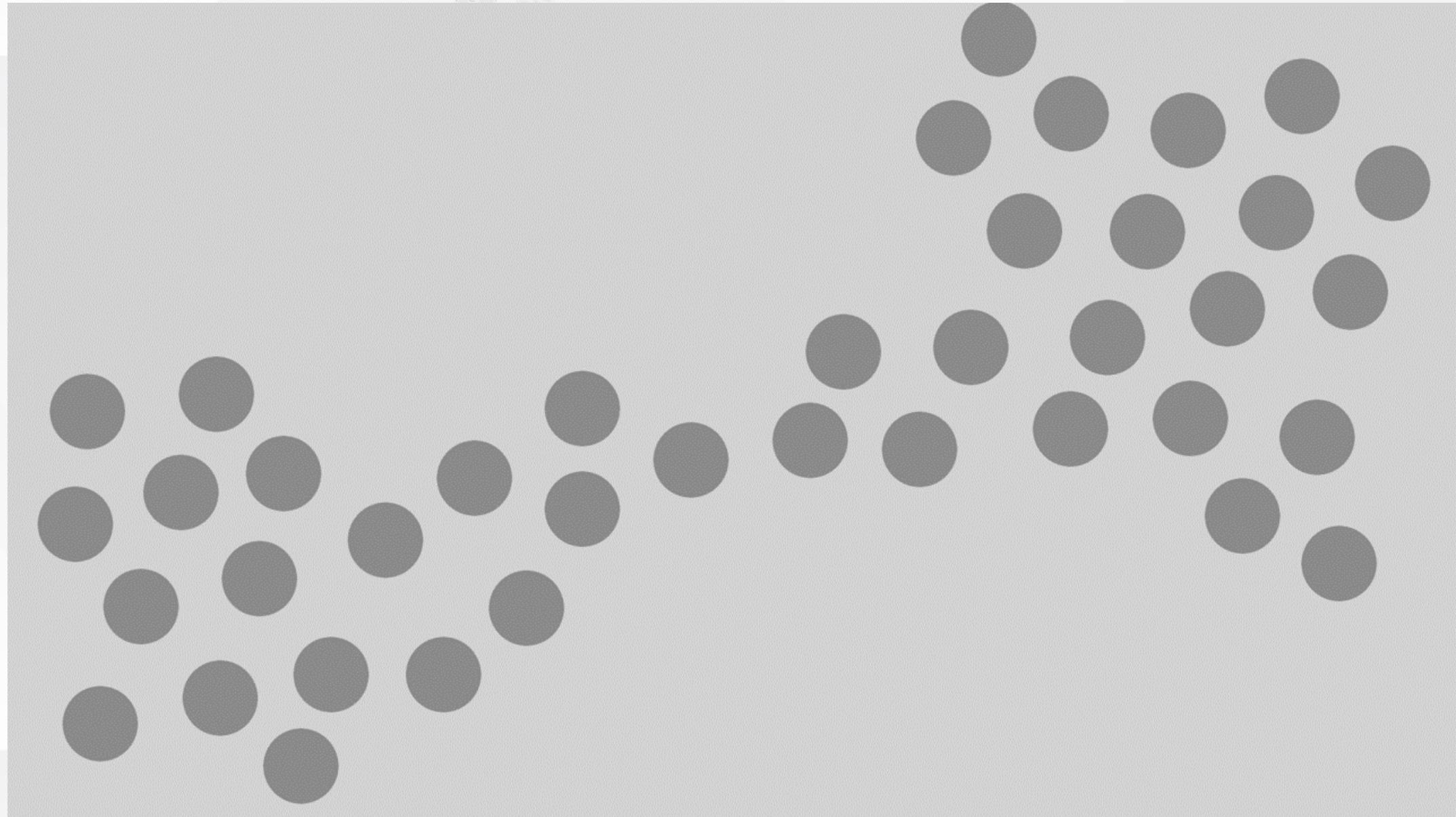
Until 质心收敛，当前均值向量未更新 or 达到最大迭代次数

输出：

$$C = \{C_1, C_2 \dots C_k\}$$

# K 均值算法 (K-means)

K 均值算法的可视化



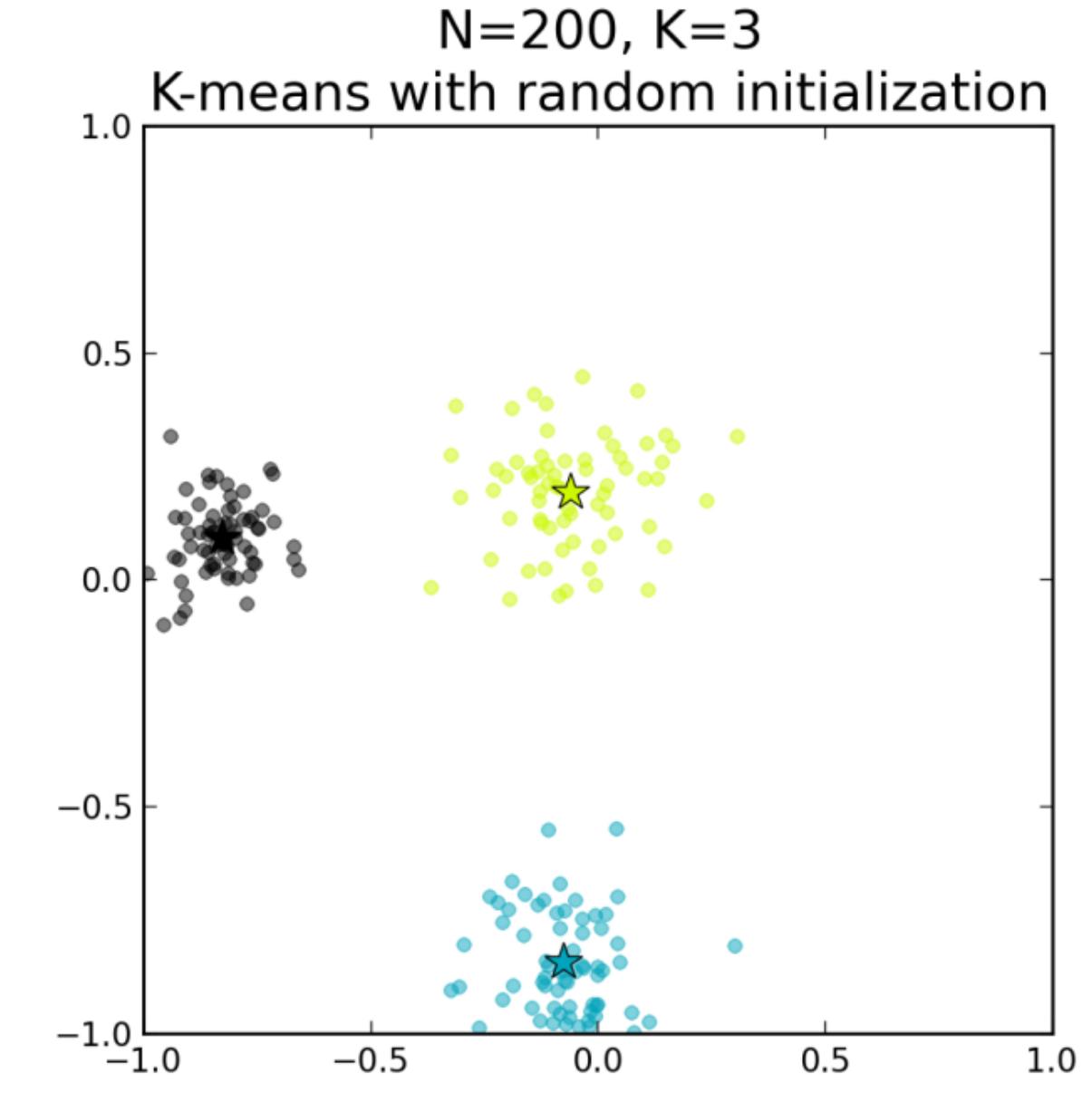
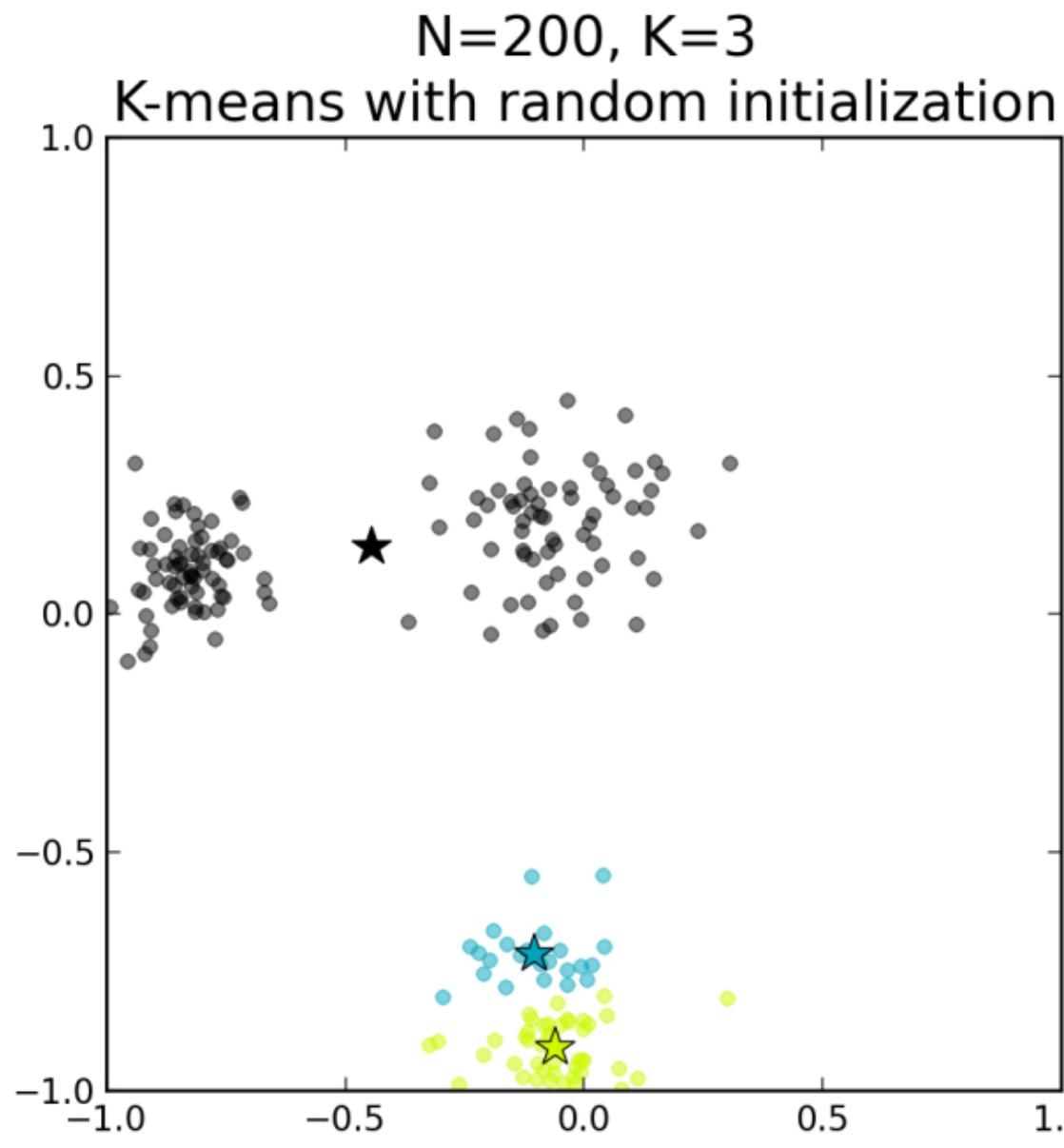
<https://www.youtube.com/watch?v=JLCwNeqf1v0>

# K 均值算法 (Kmeans)

- 优点：
  - 简单直观，易于理解实现
  - 复杂度相对比较低，在K不是很大的情况下，Kmeans的计算时间相对很短
  - Kmeans 会产生紧密度比较高的簇，反应了簇内样本围绕质心的紧密程度的一种算法
- 缺点：
  - 很难预测到准确的簇的数目
  - 对初始值设置很敏感 (Kmeans++)
  - Kmeans 主要发现圆形或者球形簇，对不同形状和密度的簇效果不好
  - Kmeans对噪声和离群值非常敏感 (Kmedians 对噪声和离群值不敏感)

# Kmeans 初始值敏感

Disadvantage of K-Means



# 初始质心优化：Kmeans++

How to solve the disadvantage? - Optimize Initial K-Mean assignment.

输入: 样本集  $D = \{x_1, x_2 \dots x_m\}$ , 聚类簇的数量k

选取初始质心的过程:

1. 随机从m个样本点中选择一个样本作为第一个簇的质心C1
2. 计算所有的样本点到质心C1的距离 :  $D_i^2 = \|x_i - C_1\|^2$
3. 从每个点的概率分布  $D_i^2 / \sum_j D_j^2$  中随机选取一个点作为第二个质心C2。离C1越远的点，被选择的概率越大
4. 重新计算所有样本点到质心的距离:
5. 重复上述过程，直到初始的k个质心被选择完成  $D_i^2 = \min(\|x_i - C_1\|^2, \|x_i - C_2\|^2)$
6. 按照Kmeans的算法步骤完成聚类

输出:

$$C = \{C_1, C_2 \dots C_k\}$$

# 层次聚类 (hierarchical clustering)

## e.g. Agglomerative clustering algorithm

- 主要在不同层次对数据集进行逐层分解，直到满足某种条件为止
- 先计算样本之间的距离。每次将距离最近的点合并到同一个类。然后，再计算类与类之间的距离，将距离最近的类合并为一个大类。不停的合并，直到合成了一个类
- 自底向上 (bottom-up) 和自顶向下 (top-down) 两种方法
  - top-down : 一开始每个个体都是一个初始的类，然后根据类与类之间的链接 (linkage) 寻找同类，最后形成一个最终的簇
  - bottom-up : 一开始所有样本都属于一个大类，然后根据类与类之间的链接 (linkage) 排除异己，达到聚类的目的

# 层次聚类 (hierarchical clustering)

类与类的距离的计算方法有：

最短距离法，最长距离法，中间距离法，平均距离法等

$$\text{最小距离: } d_{min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} dist(\mathbf{x}, \mathbf{y})$$

$$\text{最大距离: } d_{max}(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} dist(\mathbf{x}, \mathbf{y})$$

$$\text{平均距离: } d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} dist(\mathbf{x}, \mathbf{y})$$

单链接 (single-linkage) : 根据最小距离算法

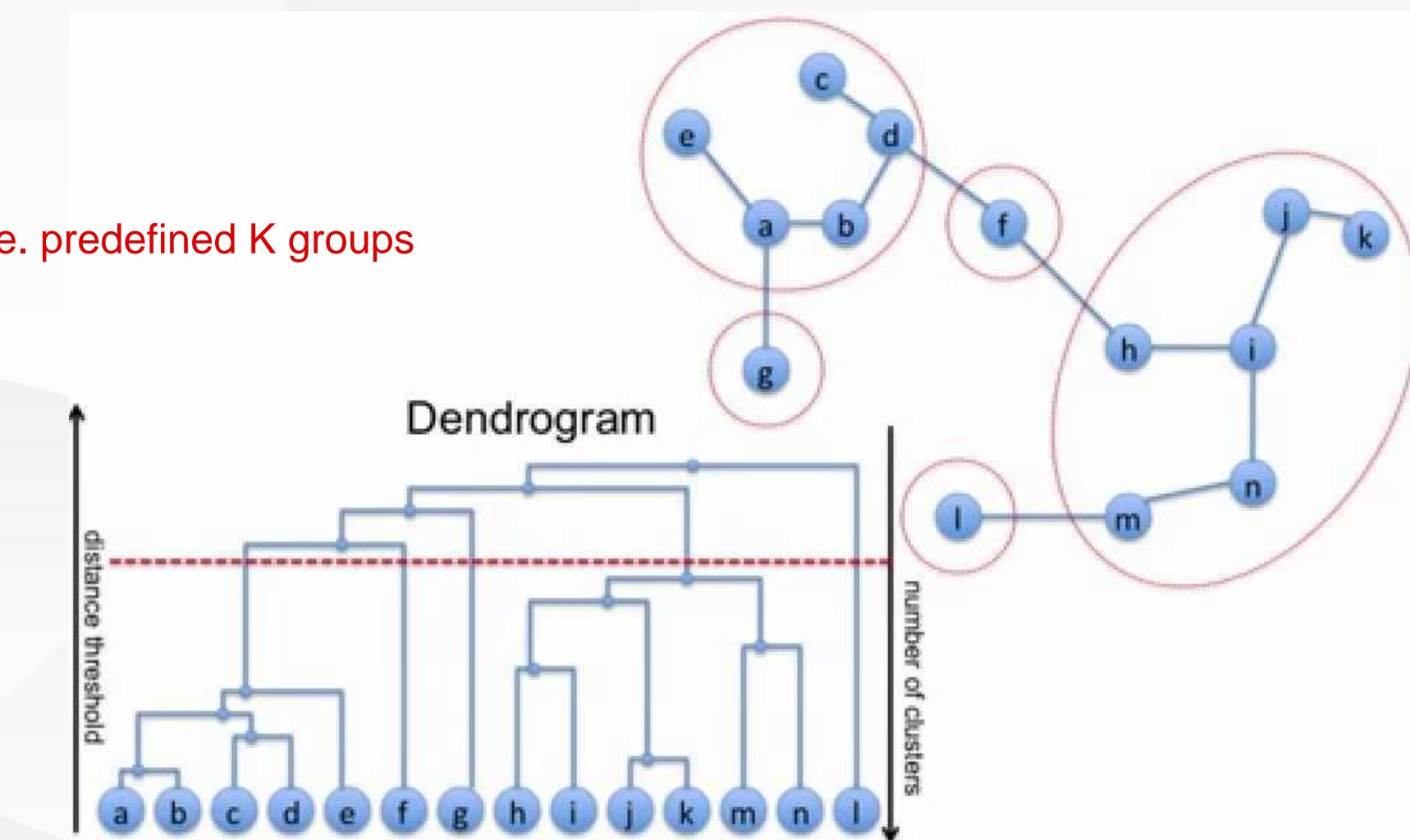
全链接 (complete-linkage) : 根据最大距离算法

均链接 (average-linkage) : 根据平均距离算法

# Agglomerative clustering 算法

凝聚层次聚类具体算法流程：

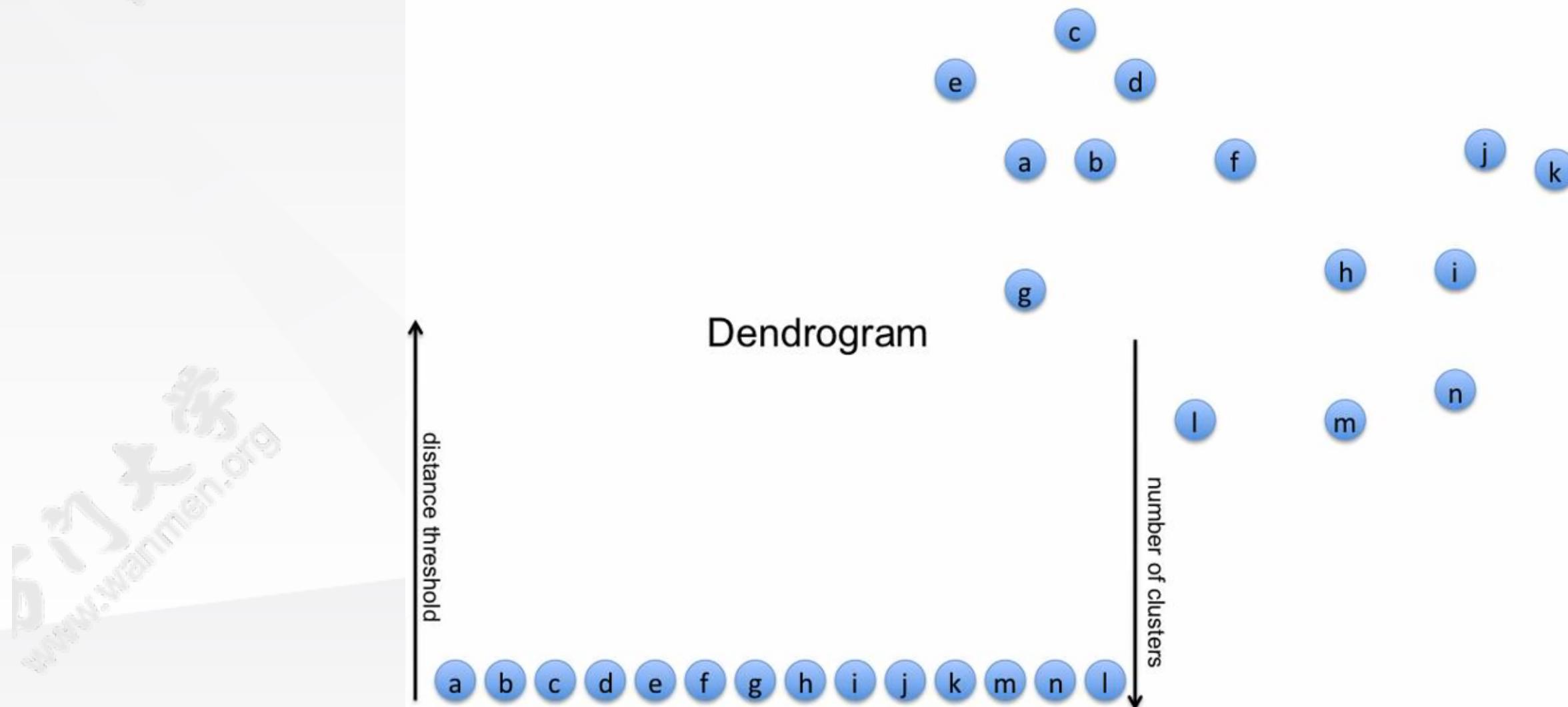
- 1) 给定样本集，决定聚类簇距离度量函数以及聚类簇数目k
- 2) 将每个样本看作一类，计算两两之间的距离
- 3) 将距离最小的两个类合并成一个新类
- 4) 重新计算新类与所有类之间的距离
- 5) 重复 (3 - 4) , 直到达到所需要的簇的数目 i.e. predefined K groups



# Agglomerative clustering 算法

Agglomerative 聚类算法可视化

Agglomerative clustering: example



<https://www.youtube.com/watch?v=XJ3194AmH40>

# Agglomerative clustering 算法

- 优点：
  - 可以得到任意形状的簇，没有Kmeans对形状上的限制
  - 可以发现类之间的层次关系
  - ~~不要制定簇的数目~~ -> Still need predefined K groups (see two pages above)
- 缺点
  - 通常来说，计算复杂度高（很多merge/split）
  - 噪声对层次聚类也会产生很大影响
  - 不适合大样本的聚类

# 密度聚类 (density based clustering)

## DBSCAN Algorithm (RECOMMENDED)

- 基于密度的方法的特点是不依赖于距离，而是依赖于密度，从而克服K均值只能发现“球形”聚簇的缺点。
- 核心思想：只要一个区域中点的密度大于某个阈值，就把它加到与之相近的聚类中去
- 密度算法从样本密度的角度来考察样本的可连接性，并基于可连接样本不断扩展聚类簇以获得最终的聚类结果
- 对噪声和离群值的处理有效
- 经典算法：DBSCAN (density based spatial clustering of applications with noise)

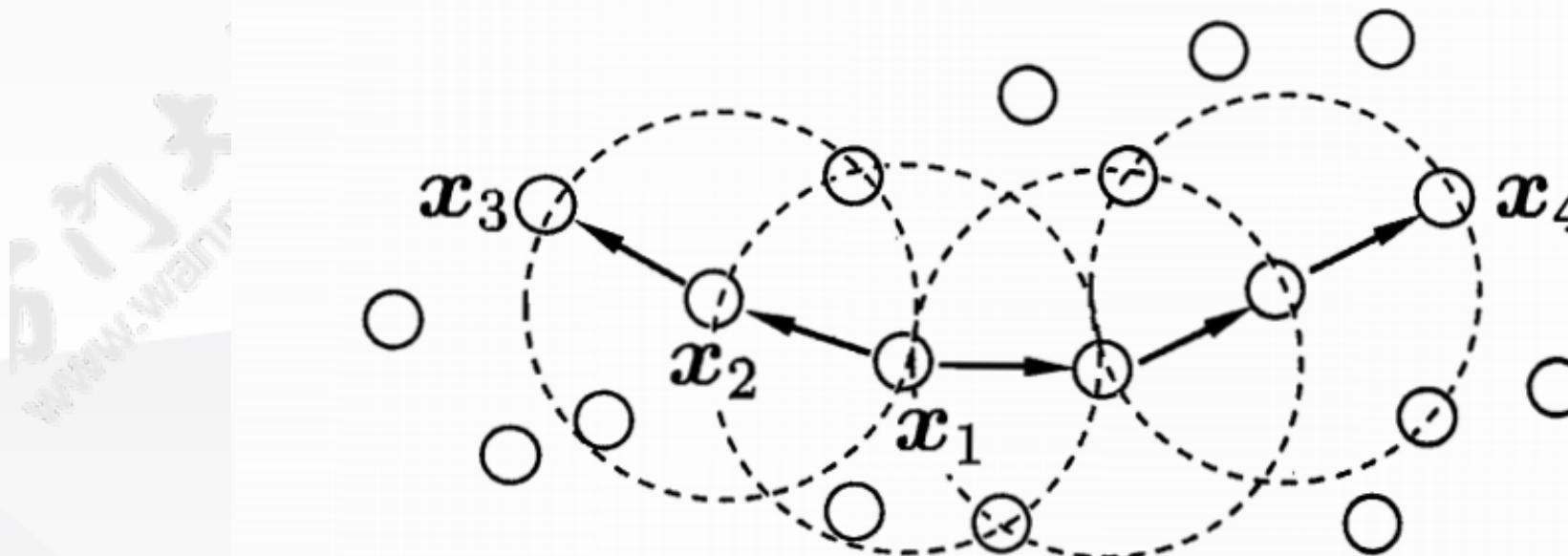
# DBSCAN

DBSCAN 基于近邻域 (neighborhood) 参数 ( $\epsilon$ , MinPts) 刻画样本分布的紧密程度的一种算法

基本概念：

- 样本集：  $D = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_m\}$
- 阈值：  $\epsilon$
- $\epsilon$ -邻域： 对样本点  $\mathbf{x}_j$  的  $\epsilon$ -邻域包括样本集中与  $\mathbf{x}_j$  距离不大于  $\epsilon$  的样本，即
- 核心对象 (core object)： 如果  $\mathbf{x}_j$  的  $\epsilon$ -邻域至少包含 MinPts 个样本，那么  $\mathbf{x}_j$  就是一个核心对象

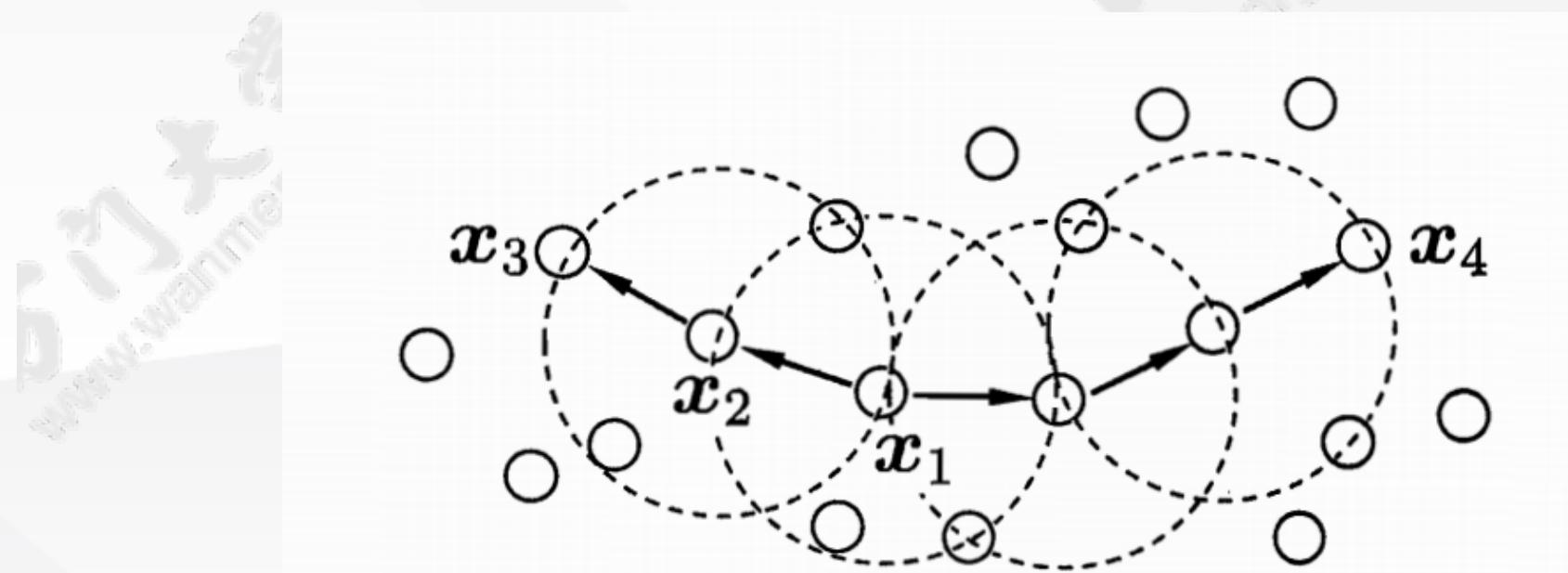
$$N_\epsilon(\mathbf{x}_j) = \{\mathbf{x}_i \in D \mid \text{dist}(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon\}$$



假设 MinPts = 3，虚线表示为  $\epsilon$ -邻域

# DBSCAN

- 密度直达 (directly density-reachable) : 如果  $x_j$  位于  $x_i$  的  $\varepsilon$ -邻域中，并且  $x_i$  是核心对象，那么  $x_j$  由  $x_i$  密度直达
- 密度可达 (density-reachable) : 对  $x_j$  和  $x_i$ ，如果存在一串样本点  $p_1, p_2, \dots, p_n$ ,  $p_1 = x_j$ ,  $p_n = x_i$ , 且  $p_{i+1}$  由  $p_i$
- 密度直达，则称  $x_j$  由  $x_i$  密度可达
- 密度相连：存在样本集合中一点  $O$ ，如果  $x_j$  和  $x_i$  均由  $O$  密度可达，那么  $x_j$  和  $x_i$  密度相连

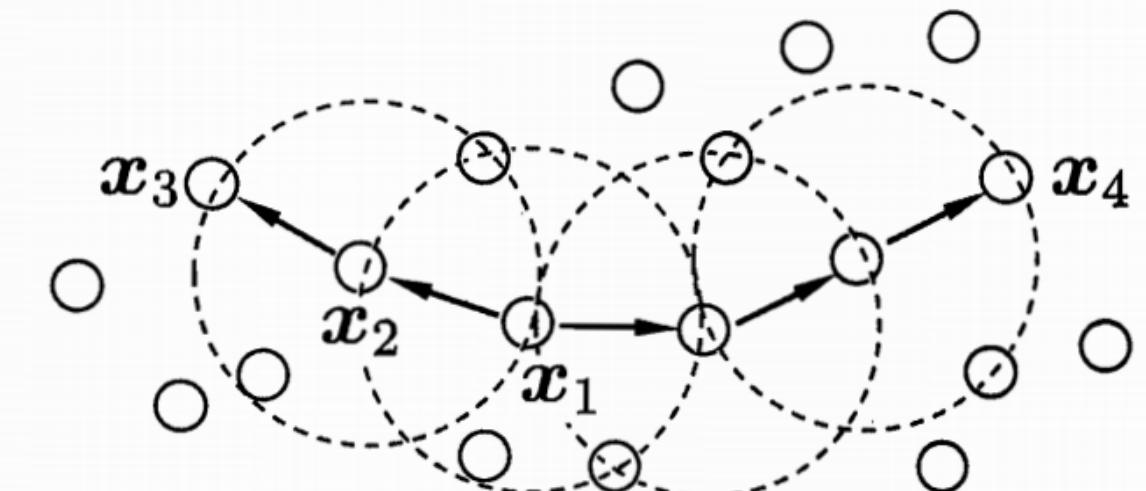


$x_1$  是核心对象，那么从  $x_1$  出发  
 $x_2$  由  $x_1$  密度直达  
 $x_3$  由  $x_1$  密度可达  
 $x_3$  与  $x_4$  密度相连

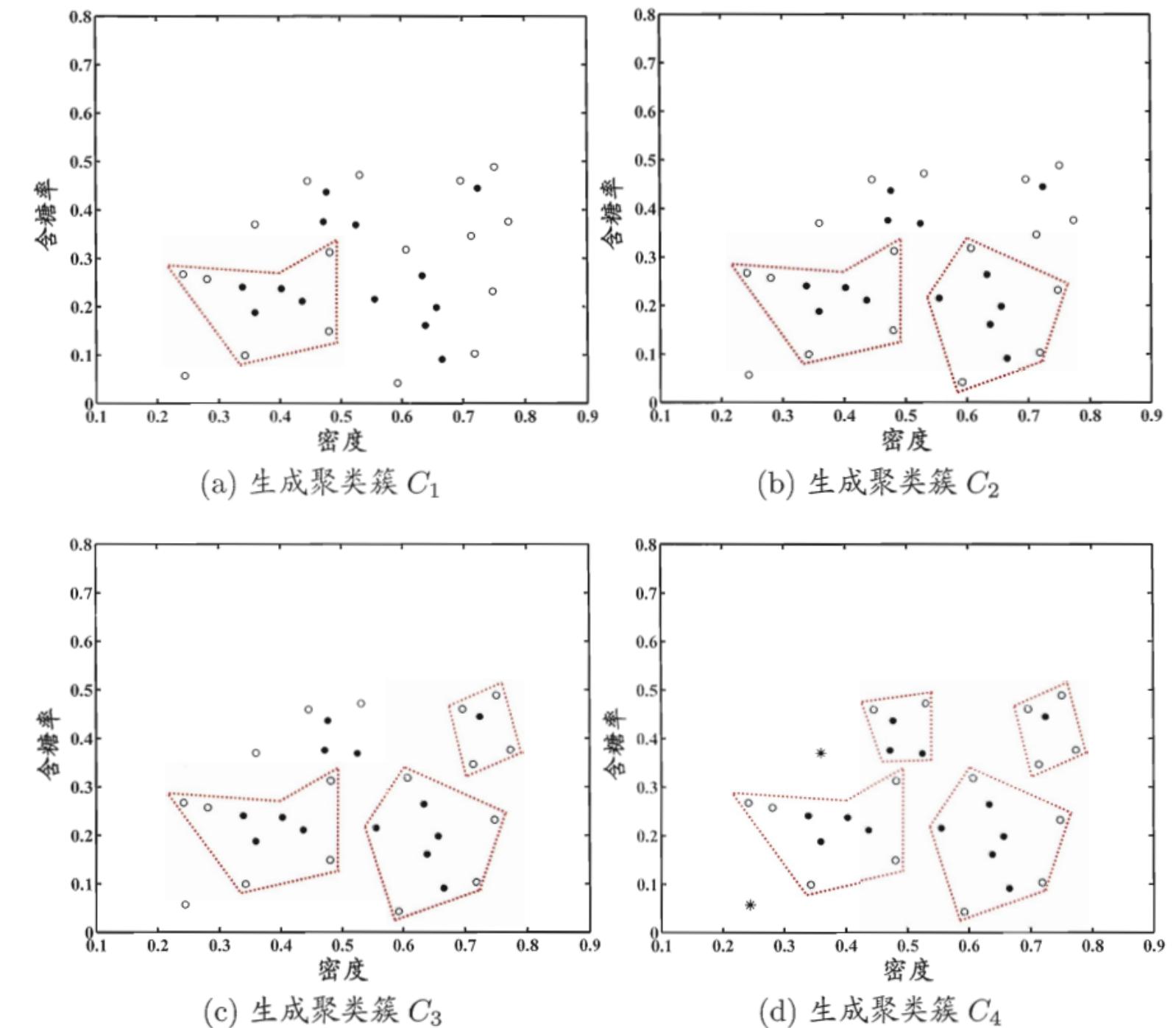
# DBSCAN

## DBSCAN算法的过程：

- 1) 首先根据邻域参数 ( $\varepsilon$ , MinPts) 确定样本集合D中所有的核心对象，存在集合P中。加入集合P的条件为 $\varepsilon$ -邻域有不少于MinPts的样本数
- 2) 然后从核心对象集合P中任意选取一个核心对象作为初始点，找出其密度可达的样本生成聚类簇，构成第一个聚类簇C1
- 3) 将C1内的所有核心对象从P中去除，再从更新后的核心对象集合任意选取下一个种子样本
- 4) 重复步骤 (2 - 3) ，直到核心对象被全部选择完，也就是P为空集



# DBSCAN



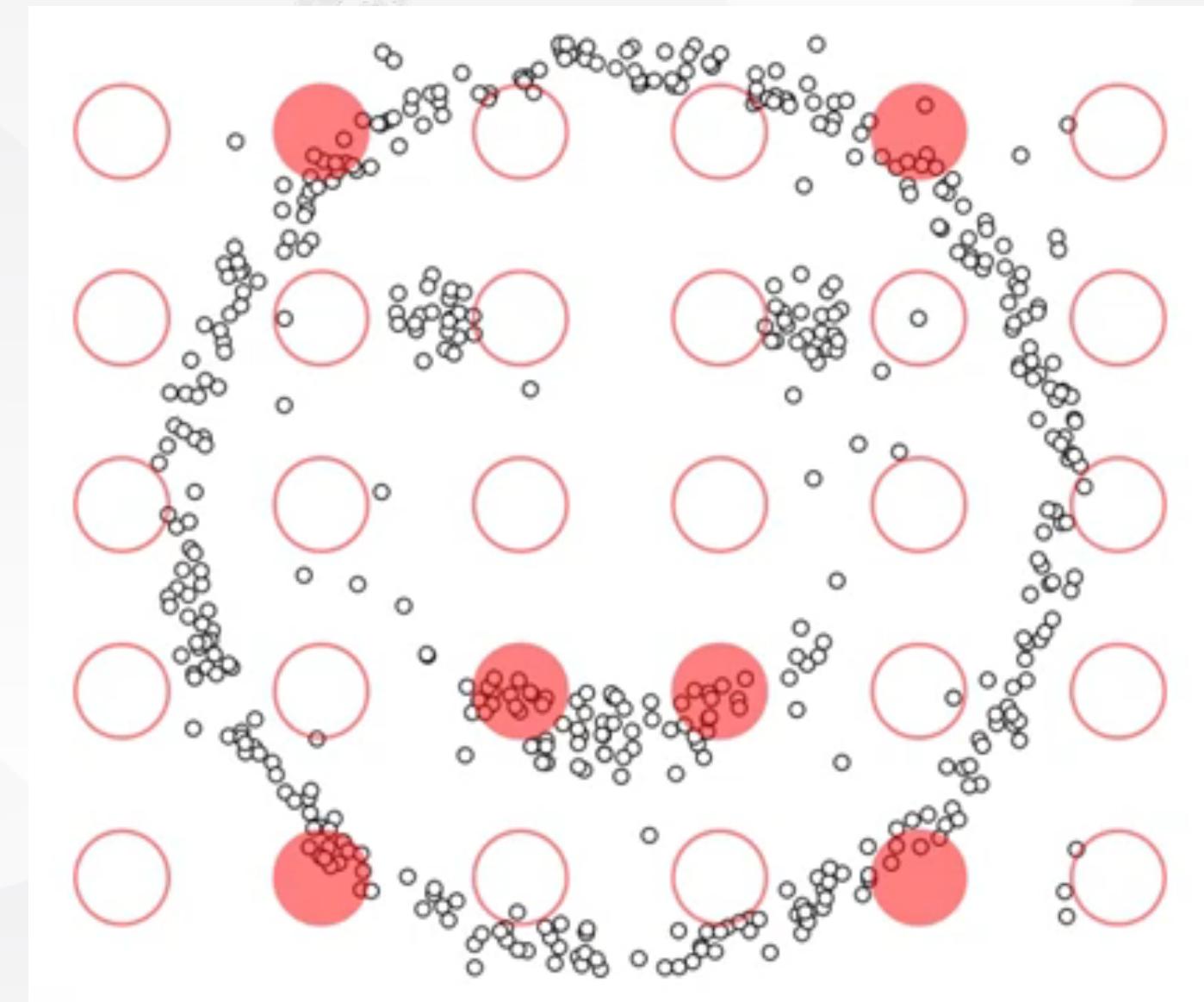
## DBSCAN算法的展示

经过DBSCAN聚类后，会产生三种样本：

- : 核心对象
- : 非核心对象
- \* : 噪声 (不属于任何一个cluster)

# DBSCAN

## DBSCAN算法演示可视化



<https://www.youtube.com/watch?v=h53WMllmUuc>

# DBSCAN

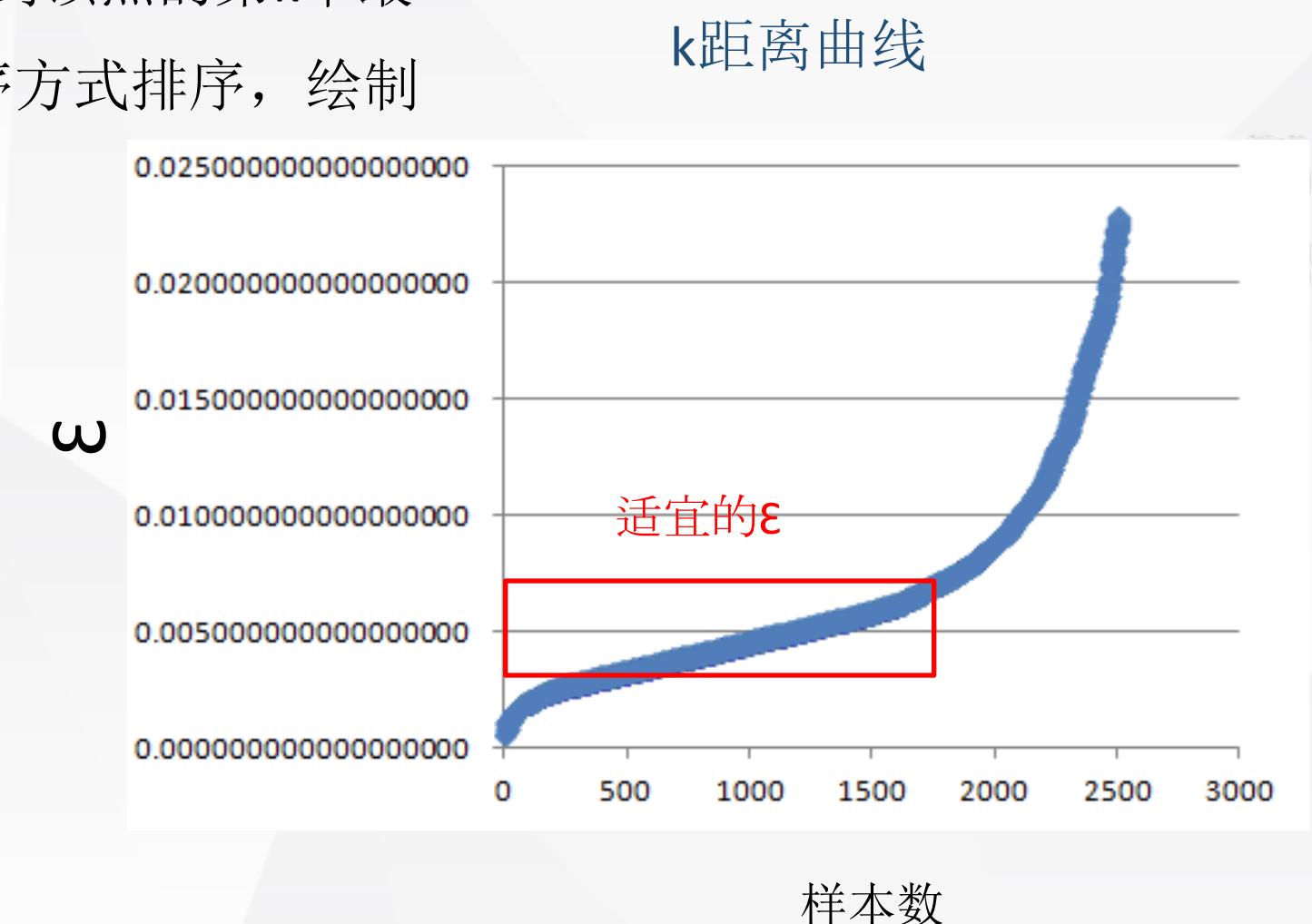
- DBSCAN 两个重要参数  $\epsilon$  和 MinPts, 决定了DBSCAN聚类的效果
- 对于 MinPts, 一般将k值设为4 --> Depends on project needs. (minimum sample points within one group)
  - MinPts 过小, 那么稀疏簇中的由于密度小于MinPts, 不利于簇的进一步扩展
  - MinPts 过大, 很多点会被视为噪声点

# DBSCAN

- DBSCAN 两个重要参数  $\epsilon$  和 MinPts, 决定了DBSCAN聚类的效果  
How to find optimal

- 对于  $\varepsilon$ , 由  $k$ -距离曲线 ( $k$ -dist graph) 得到。这里的  $k = \text{MinPts}$

- $k$ 距离曲线：对于样本集中每个点，计算出样本集其他点到该点的第 $k$ 个最近邻距离，并将数据集所有点对应的最近邻距离按照升序方式排序，绘制出 $k$ 距离曲线
  - $\epsilon$ 过大：那么多个簇会被归并到一起
  - $\epsilon$ 过小：大部分数据不能够有效聚类



# DBSCAN

- 优点：
  - 与Kmeans相比， DBSCAN不需要知道簇的数目
  - 可以发现任意形状的簇， 不局限于球形
  - 同时， DBSCAN能够很有效的识别噪音点， 这是Kmeans和Agglomerative 聚类不具备的
- 缺点：
  - 对  $\epsilon$  和 MinPts 数据的选择很敏感
  - 对密度很稀疏或者变化的数据集， 表现的效果不是很好

## e.g. GMM (Gaussian Mixed Model) algorithm

- 基于模型的方法给每一个聚类假定一个模型，然后去寻找能很好的拟合模型的数据集。模型可能是数据点在空间中的密度分布函数或者其它
- 这样的方法通常包含的潜在假设是：数据集是由一系列的潜在概率分布生成的，属于概率生成模型 -> GMM: every sample is not 100% into one cluster, it can have x% in cluster 1, y% in cluster 2, ...
- 高斯混合模型 (Gaussian mixed model or GMM) 就是采用多元高斯分布来实现聚类的方法，是期望最大 (Expectation maximization or EM) 算法的经典体现，EM 算法是含有隐变量的概率模型参数的极大似然估计法。对于GMM模型，简单来说，隐变量就是类

# 高斯混合模型聚类

- 多于高斯分布：

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

对于高斯混合分布，假设一共有k个类，那么有k个混合成分

$$p_M(x) = \sum_{i=1}^k \alpha_i \cdot p(x|\mu_i, \Sigma_i)$$

$$\alpha_i \geq 0; \sum_{i=1}^k \alpha_i = 1; \alpha_i \text{ 为混合系数}$$

隐随机变量  $z_j \in \{1, 2, 3, \dots, k\}$ , 表示为生成样本  $x_j \in D$  的高斯混合成分

根据Bayes,  $z_j$  的后验分布概率  $\gamma_{ji}$  为：

$$\gamma_{ji} = p_M(z_j = i | x_j) = \frac{p(z_j = i) p_M(x_j | z_j = i)}{p_M(x_j)} = \frac{\alpha_i p(x_j | \mu_i, \Sigma_i)}{\sum_{l=1}^k \alpha_l p(x_j | \mu_l, \Sigma_l)}$$

# 高斯混合模型聚类

那么对样本划分为  $C = \{C_1, C_2, \dots, C_k\}$  个簇，  
只需要得到样本  $x_j$  由第  $i$  个高斯混合成分的后验概率

$$\lambda_j = \operatorname{argmax}_{i \in \{1, 2, \dots, k\}} \gamma_{ji}$$

最大化对数似然：

$$\text{LogLikelihood}(D) = \log \left( \prod_{j=1}^m p_M(x_j) \right) = \sum_{j=1}^m \log \left( \sum_{i=1}^k \alpha_i p(x_j | \mu_i, \Sigma_i) \right)$$

$$\nabla_{\mu_i} \text{LogLikelihood}(D) = 0 \Rightarrow \mu_i = \frac{\sum_{j=1}^m \gamma_{ji} x_j}{\sum_{j=1}^m \gamma_{ji}}$$

$$\nabla_{\Sigma_i} \text{LogLikelihood}(D) = 0 \Rightarrow \Sigma_i = \frac{\sum_{j=1}^m \gamma_{ji} (x_j - \mu_i)^T (x_j - \mu_i)}{\sum_{j=1}^m \gamma_{ji}}$$

# 高斯混合模型聚类

- 同时除了最大化似然函数，还需要满足混合系数加和为1的约束
- 构造拉格朗日函数：

$$L = \text{LogLikelihood}(D) + \lambda \left( \sum_{i=1}^k \alpha_i - 1 \right) = \sum_{j=1}^m \log \left( \sum_{i=1}^k \alpha_i p(x_j | \mu_i, \Sigma_i) \right) + \lambda \left( \sum_{i=1}^k \alpha_i - 1 \right)$$

$$\nabla_{\alpha_i} L = 0 \Rightarrow \alpha_i = \frac{1}{m} \sum_{j=1}^m \gamma_{ji}$$

即每个高斯成分的混合系数由样本属于该成分的平均后验概率确定

# 高斯混合模型聚类

GMM 模型的过程:

1) 初始化高斯混合分布的模型参数  $\{(\alpha_i, \mu_i, \Sigma_i) | 1 \leq i \leq k\}$

2) repeat :

for j from 1 to m, do:

$$\gamma_{ji} = p_M(z_j = i | x_j) = \frac{\alpha_i p(x_j | \mu_i, \Sigma_i)}{\sum_{l=1}^k \alpha_l p(x_j | \mu_l, \Sigma_l)}$$

end for

for i from 1 to k do:

$$\mu'_i = \frac{\sum_{j=1}^m \gamma_{ji} x_j}{\sum_{j=1}^m \gamma_{ji}} \quad \Sigma'_i = \frac{\sum_{j=1}^m \gamma_{ji} (x_j - \mu_i)^T (x_j - \mu_i)}{\sum_{j=1}^m \gamma_{ji}}$$

end for

$$\alpha'_i = \frac{1}{m} \sum_{j=1}^m \gamma_{ji}$$

until 满足条件

3) 根据所得到的最大后验概率值对每个样本进行归类

# 高斯混合模型聚类

- 优点：
  - 和Kmeans不同， GMM对每个类的划分并不是那么绝对， 而是以一种概率形式表现。
  - 实际上Kmeans是一种特殊情况下的GMM， 每个样本只属于一个类， 而属于其他类的概率为0， 这也就是为什么Kmeans只产生球形的类， 而GMM并没有这种限制。
  - GMM有着很好的统计解释， GMM的混合特征让其在某种情况下更适用（比如新闻会属于很多的话题类， topic modelling）
- 缺点：
  - 计算复杂度高， 效率低。相对于Kmeans等其他算法， GMM往往需要大量的完全协方差高斯分布来描述数据集
  - 不能够很好的scalable

# 聚类算法总结

- 基于划分的聚类 (partitioning based clustering) : K均值 (K-means) , Kmeans++
- 层次聚类 (hierarchical clustering): Agglomerative 聚类
- 密度聚类 (density based clustering) : DBSCAN
- 基于模型的聚类 (model based clustering) : 高斯混合模型 (GMM)