机器学习

Machine Learning

主讲人：张敏 清华大学长聘副教授

ML 机器学习

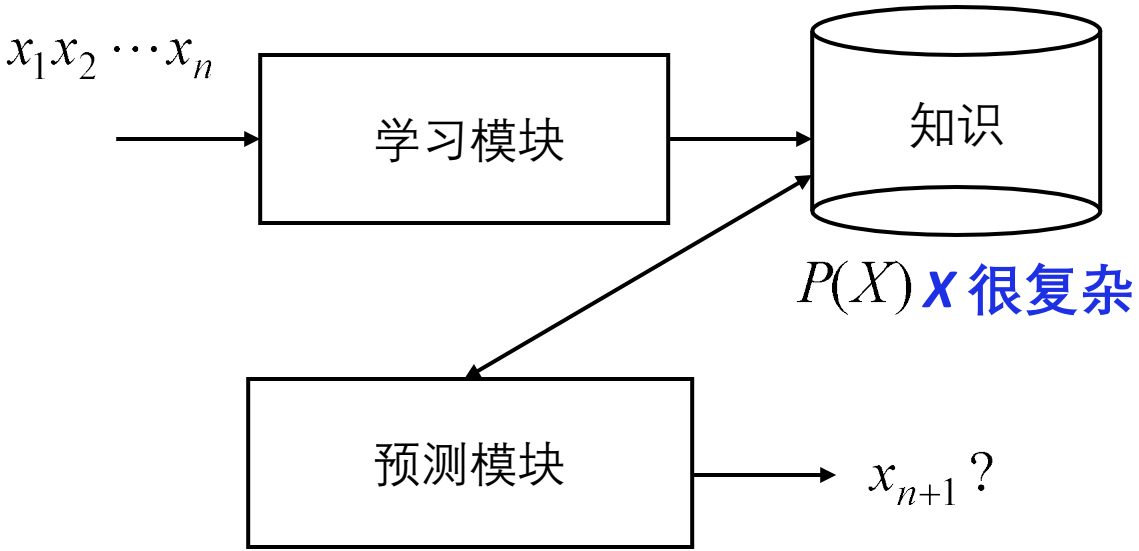
MACHINE LEARNING-MIN ZHANG

Unit.08

无监督学习(II)

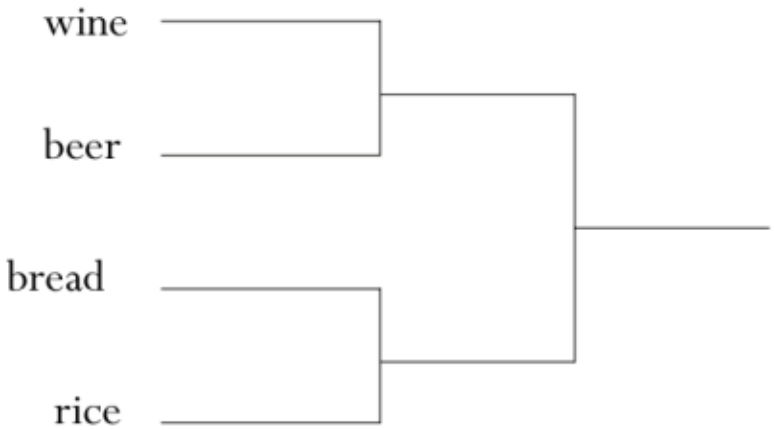
*图片均来自网络或已发表刊物

回顾：无监督学习



目录

层次聚类



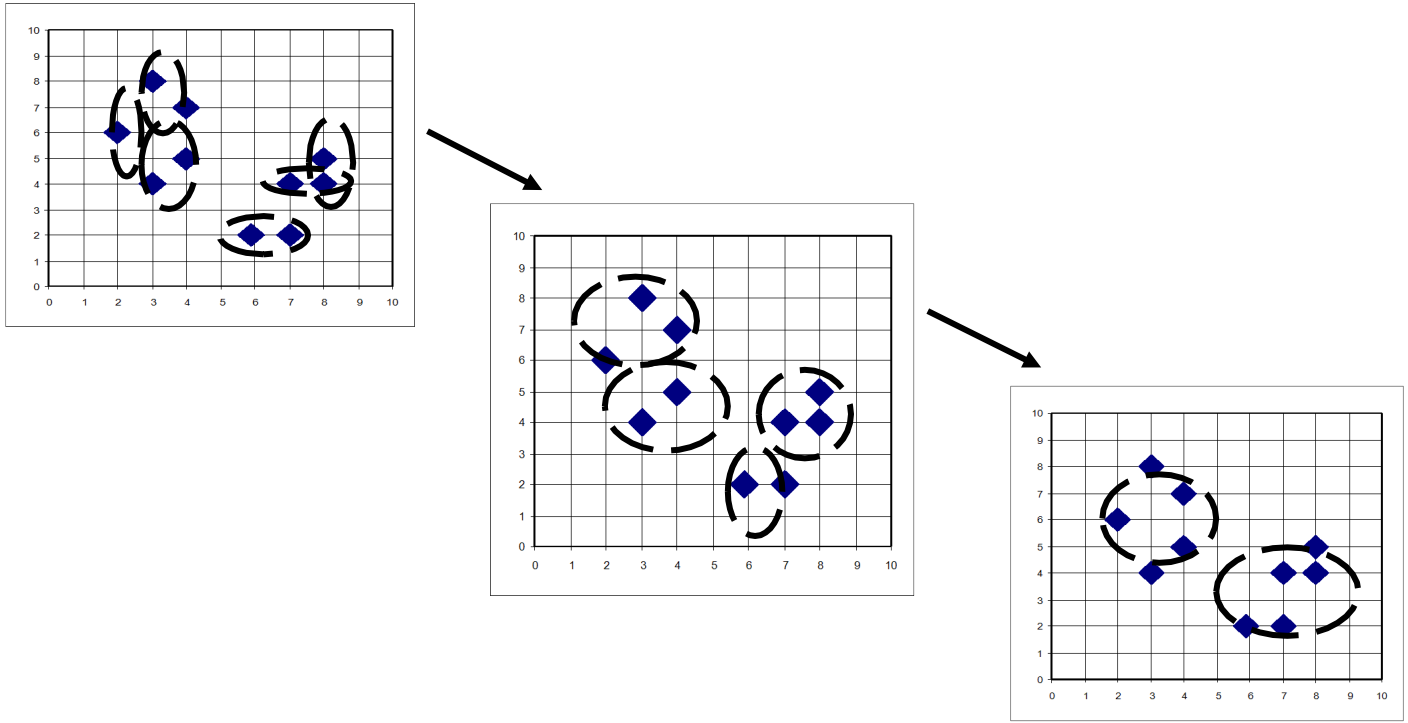
- 层次地构建一个类，比如一个由不同类组成的树状结构
  - 父节点所涵盖的点被分割为兄弟类
- 以不同的粒度解释数据

凝聚式层次聚类算法(Agglomerative, bottom-up)

- 通过迭代过程得到嵌套式聚类结构
- 算法：（以文档聚类为例）
  - 计算文档对之间的相似性系数
  - 把n个文档中的每一个分配给自己构成一个簇
  - 把最相似的两个簇类 c_i 和 c_j 合并成一个
    - 用新构成的簇类 $c_i \cup c_j$ 代替原来的两个簇 c_i 和 c_j
    - 重新计算其他簇与新生成簇之间的相似性
  - 重复上述过程，直到只剩下k个簇（k可以等于1）

凝聚式层次聚类

- (层次凝聚式聚类)



类相似度

- Single linkage: 两个类中最相似的两个数据点

$$\text{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

$$\text{sim}(c_i \cup c_j, c_k) = \max(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))$$

- Complete linkage: 两个类中最不相似的两个数据点

$$\text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

$$\text{sim}(c_i \cup c_j, c_k) = \min(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))$$

- Average linkage: 两个类中数据点相似度的平均值

$$\text{sim}(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

$$\text{sim}(c_i \cup c_j, c_k) = \frac{1}{|c_i| + |c_j|}(|c_i|\text{sim}(c_i, c_k), |c_j|\text{sim}(c_j, c_k))$$

实例：意大利城市的层次聚类

- 欧式距离
- Single linkage

距离矩阵

	BA	FI	MI	NA	RM	TO
BA	0	622	877	255	412	996
FI		0	295	468	268	400
MI			0	754	564	138
NA				0	219	869
RM					0	669
TO						0

实例：意大利城市的层次聚类

距离矩阵

	BA	FI	MI/TO	NA	RM
BA	0	622	877	255	412
FI		0	295	468	268
MI/TO			0	754	564
NA				0	219
RM					0

实例：意大利城市的层次聚类

距离矩阵

	BA	FI	MI/TO	NA/RM
BA	0	622	877	255
FI		0	295	268
MI/TO			0	564
NA/RM				0

实例：意大利城市的层次聚类

距离矩阵

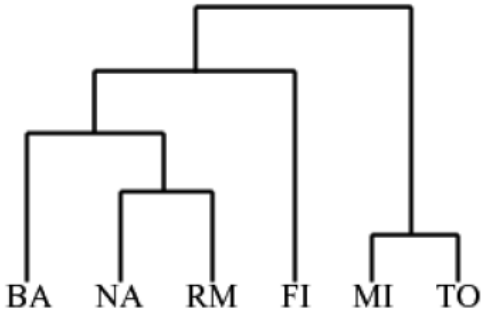
	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI		0	295
MI/TO			0

实例：意大利城市的层次聚类

距离矩阵

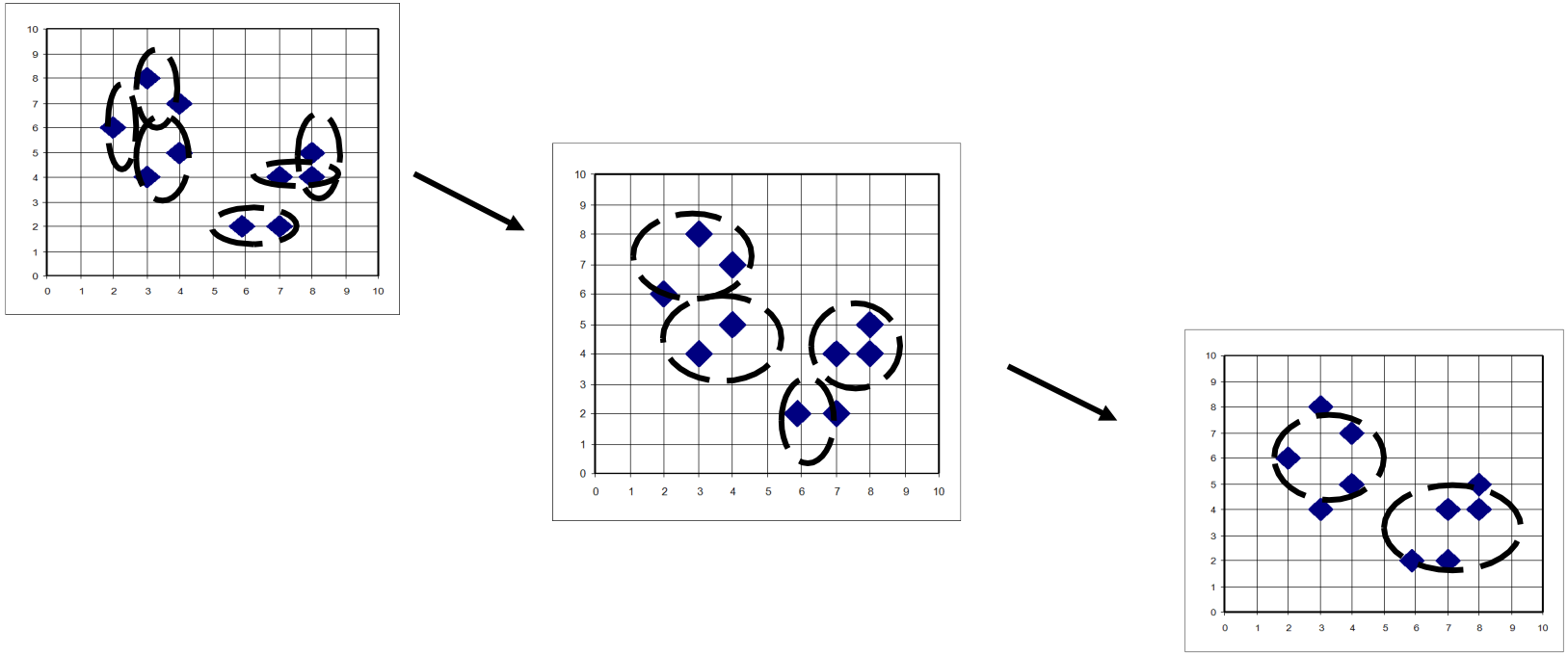
	BA/NA/RM/FI	MI/TO
BA/NA/RM/FI	0	295
MI/TO		0

Final result



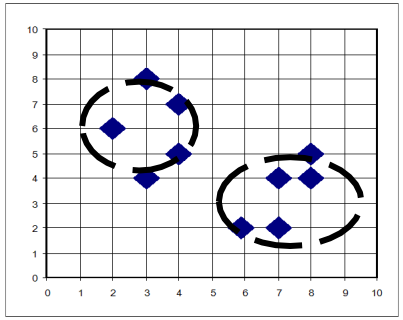
目录

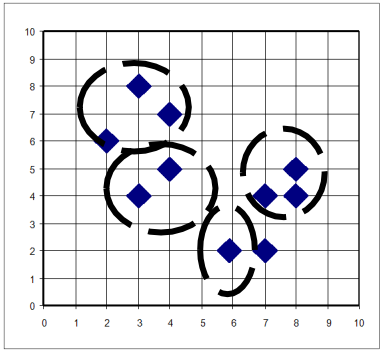
回顾：凝聚式层次聚类 (Agglomerative hierarchical clustering)

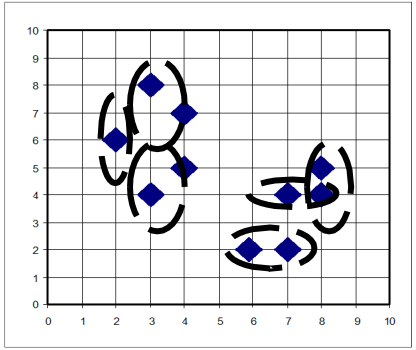


分裂式层次聚类（Divisive, top-down）

根据一个类中最大的间隔进行分裂







1. 最大平均类内距离的点→ Splinter group
2. 其他点 → 保持不变（Old party）

3. 重复以下操作直到不再发生改变:
把满足MinDist_to_Splinter >= MinDis_to_Old的点 → Splinter

分裂式层次聚类vs. 凝聚式层次聚类

- 分裂式
  - 可以利用全局信息
  - 可以只关心局部的细分
  - 需要与其他flat clustering alg.结合使用(e.g. k-means)
  - 寻找分裂点比较复杂
  - 一旦开始被分开，再也不会被聚起来，不能保证距离接近的点一定在一层中的同一个类里

凝聚式

只用局部信息

必须全聚完

不需要其他算法

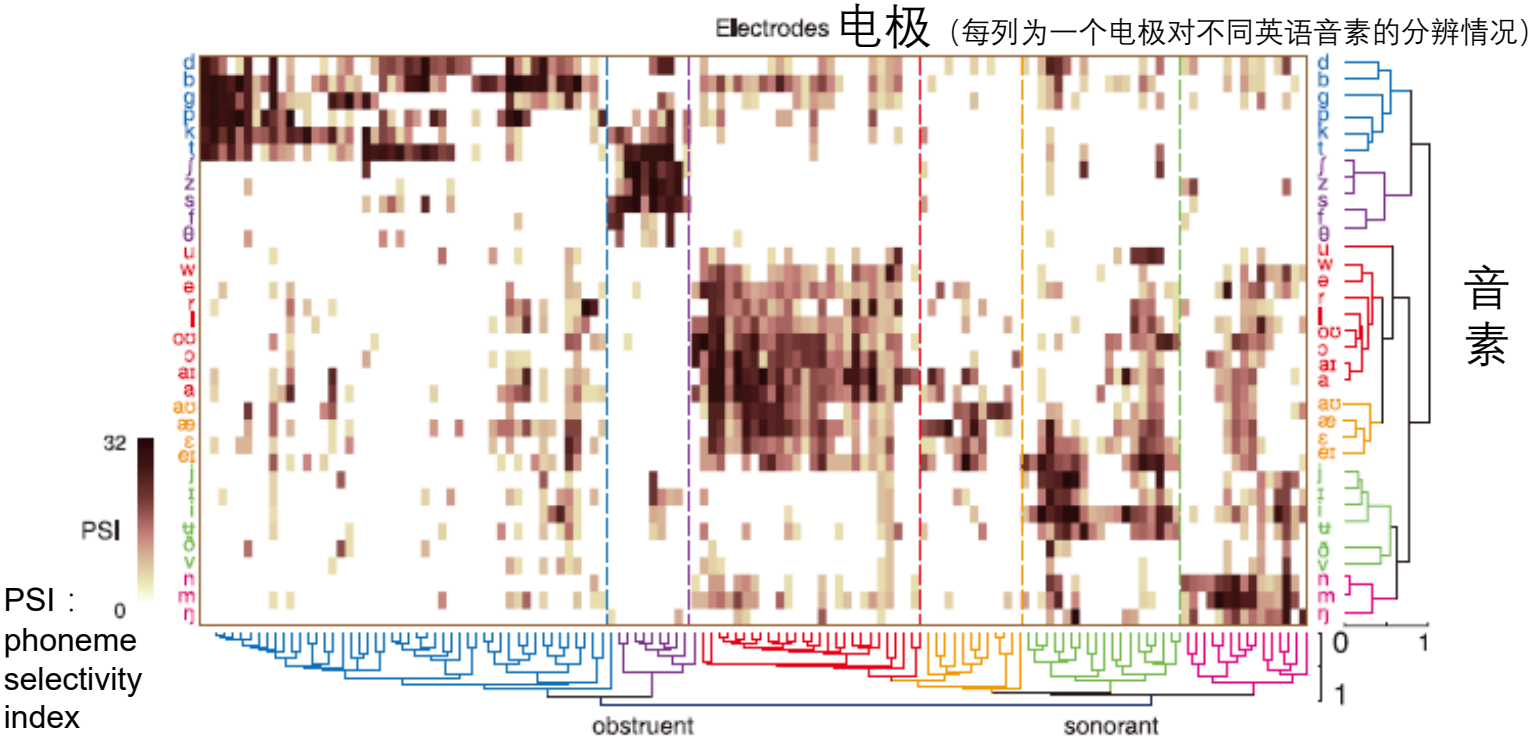
相近的类聚合容易

可以保证

层次聚类的相关讨论

- 优点
  - 可以从不同粒度观察数据，十分灵活
  - 可以方便适应各种形式的相似度定义
  - 因此适用于各种属性类型
- 缺点
  - 停止条件不确定
  - 计算开销大、很难应用到大的数据集上

神经科学数据分析中的应用



在癫痫病人脑中插了很多电极，记录听觉皮层不同位置对于语音的反应

[Mesgarani et al., science, 2014]

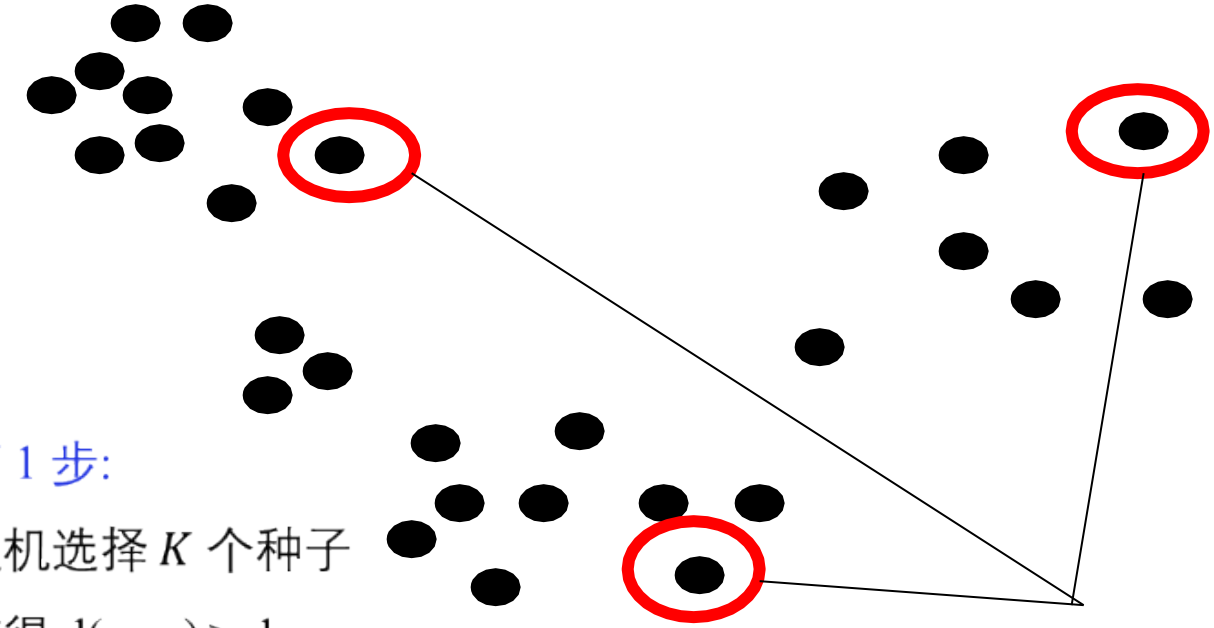
目录

- 无监督学习介绍
- 聚类介绍
- 层次聚类
- K-means 聚类
- K-medoids 聚类

算法

- 给定一个类分配方案 C，确定每个类的均值向量：$\{g_1, \cdots, g_K\}$
- 给定 K 个均值向量的集合 $\{g_1, \cdots, g_K\}$，把每个对象分配给距离均值最近的类
- 重复上述过程直到评价函数值不发生变化

示意：初始化



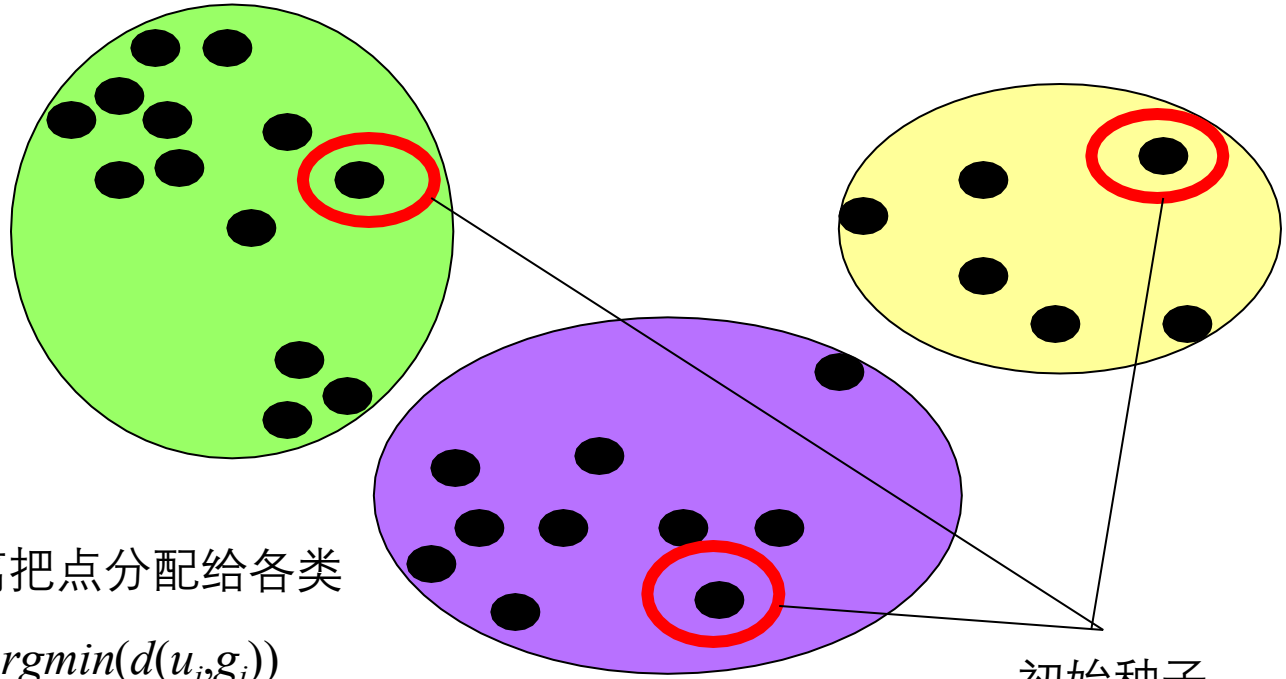
第 1 步:

随机选择 K 个种子

使得 $d(g_i, g_j) > d_{min}$

初始种子 (当 $K = 3$)

示意：类的分配



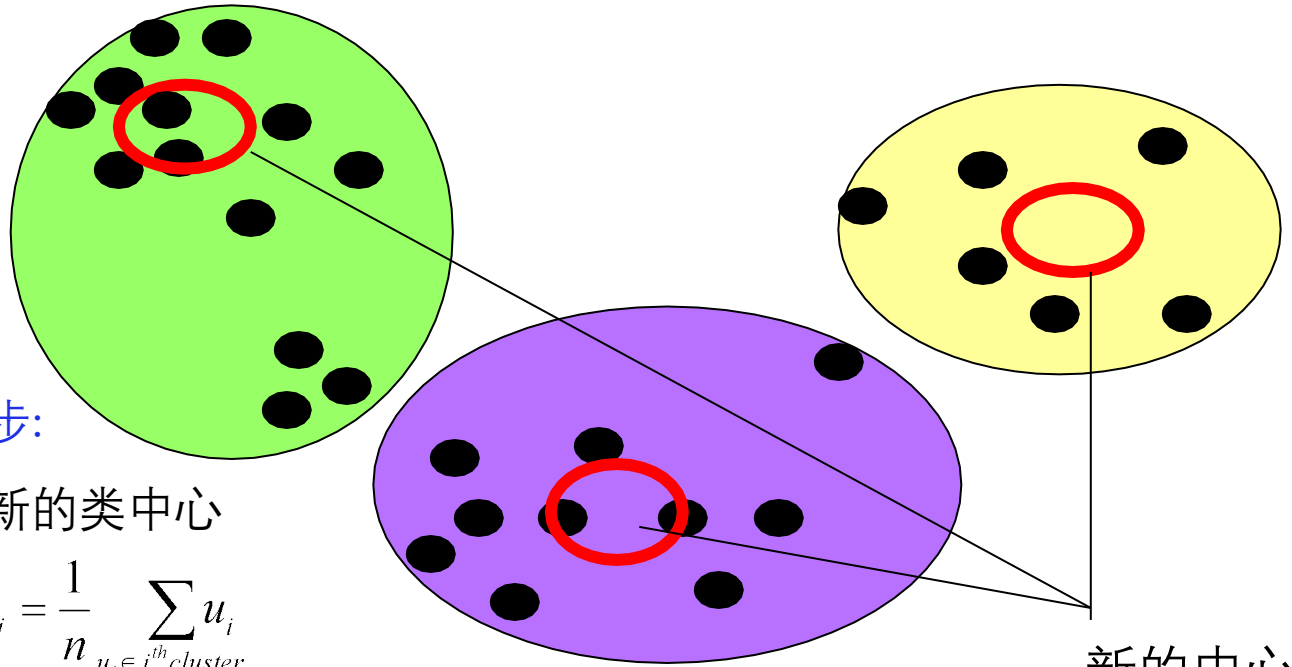
第 2 步:

根据最近距离把点分配给各类

$Cluster(u_i) = argmin(d(u_i,g_i))$

$g_i \in \{g_1,\ldots,g_K\}$

示意：更新中心



第 3 步:

计算新的类中心

$$g_j = \frac{1}{n} \sum_{u_i \in j^{th} cluster} u_i$$

新的中心

接着重复第 2&3 步

算法

- 给定一个类分配方案 C，确定每个类的均值向量：$\{g_1, \cdots, g_K\}$
- 给定 K 个均值向量的集合 $\{g_1, \cdots, g_K\}$，把每个对象分配给距离均值最近的类
- 重复上述过程直到评价函数值不发生变化
- 不保证找到最优解

算法的收敛性

- 问题描述
  - 用 $u_1, \ldots, u_N$ 表示数据，其中 $u_N \in R^D$
  - 用 g_v 表示类 v 的均值，其中 $v = 1, \ldots, K$
  - 目标是将所有点分配到各个类中，使得分配后的 g_v 满足：

    所有点到对应的 g_v 距离平均值最小
- 形式化
  - 对每个数据点 u_n，引入一个二值的指示变量 $r_{nv} \in \{0,1\}$
    - $r_{nv} = 1$ 表示u_n这个点属于类v;
    - $r_{nv} = 0$ 表示u_n这个点不属于类 v;
  - 即 $r_{n,\cdot} = (0, \ldots, 0,1,0, \ldots, 0)$ (只有一个元素是 1)

算法的收敛性（续）

- 问题可以形式化成

$$\min \quad J = \frac{1}{N} \sum_{n=1}^{N} \sum_{v=1}^{K} r_{nv} \|u_n - g_v\|^2$$

  - 所有点到对应类的中点平均距离最小
- 如何解决这个问题？
  - 变量 r_{nv} 和 g_v 是对应的
  - 固定一个优化另一个，直到收敛
- 步骤 (a): 固定 g_v 优化 r_{nv}

$$r_{nv} = \begin{cases} 1 & \text{for } v = \arg\min_k \|u_n - g_k\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

将类分配给各个点

算法的收敛性（续）

- 步骤 (b): 固定 r_{nv} 优化 g_v

$$\min \quad J = \frac{1}{N}\sum_{n=1}^{N}\sum_{v=1}^{K} r_{nv}\|u_n - g_v\|^2$$

$$\frac{\partial J}{\partial g_v} = \frac{2}{N}\sum_{n} r_{nv}(g_v - u_n) = 0$$

$$\rightarrow \quad g_v = \frac{\sum_n r_{nv} u_n}{\sum_n r_{nv}}$$

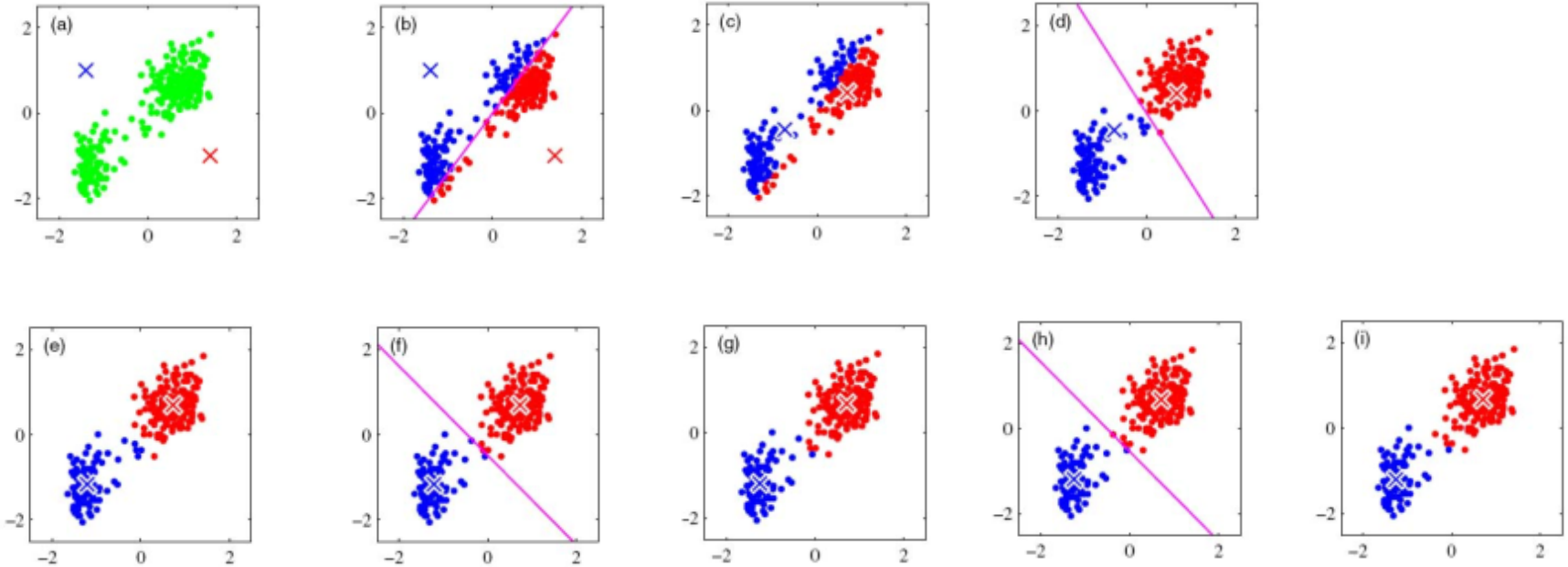
更新类中心

代价函数 J 逐步下降直到收敛!

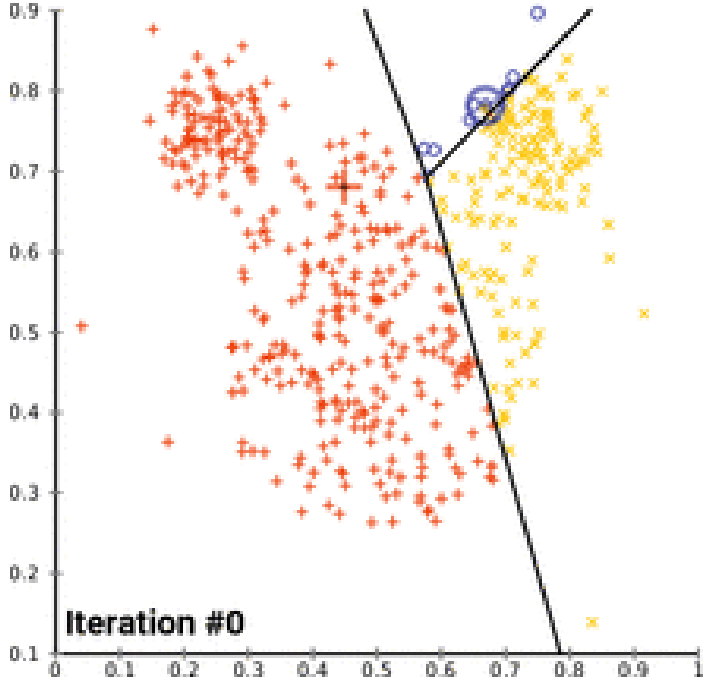
K-means 算法特性小结

- 模型: 向量空间模型
- 策略: 最小化类内对象的欧式距离
- 算法: 迭代
- 硬聚类
- 非层次

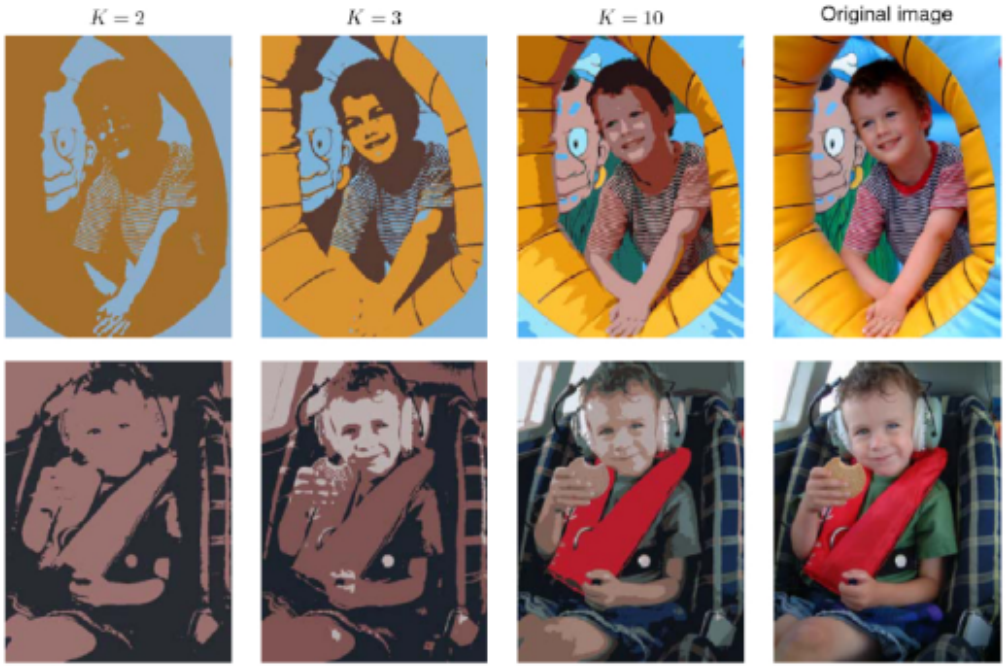
K-means 算法举例



聚类过程动画示例



应用举例：不仅仅是聚类 —— 图像压缩



数据:

所有的像素

特征:

R,G,B 值

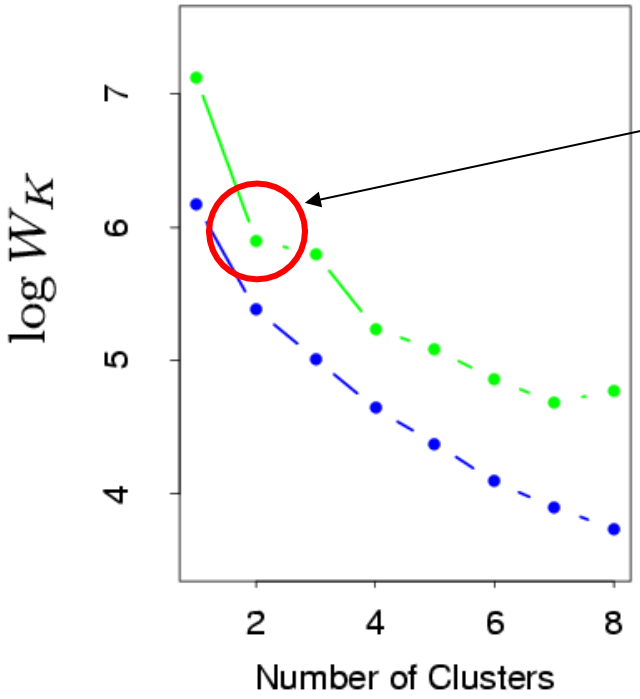
不是位置！

每个像素根据所属类的中心对应的 {R,G,B} 值进行重画

K-means讨论：如何确定“k”?

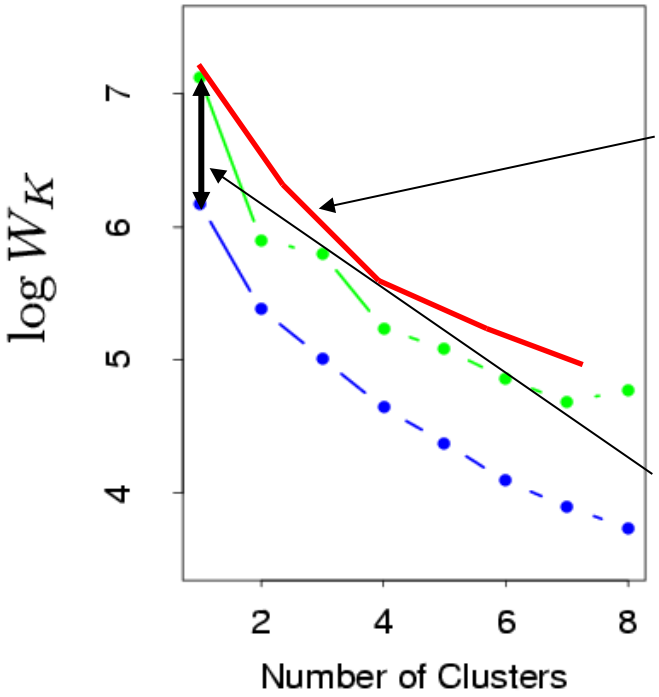
- 问题驱动
  - 通常问题本身会设定一个需要的 K 值
- 只有满足下列条件之一时，可以是 ”数据驱动” 的
  - 数据不稀疏
  - 度量的维度没有明显噪音
- 如果 K 值没有给定
  - 计算类间不相似度 W_k (与类间相似度相反) (或者检验类内相似度) —— 与 K 相关的函数
  - 一般来说， K 值增加，W_k 值降低

怎么确定 “k”？（方法 1）



观察到 W_k 急剧下降

怎么确定 “k”? （方法 2）



在均匀分布的数据上用同样的聚类算法

衡量间隔

K-means ：更多讨论

- 当数据呈几个紧凑且互相分离的云状时效果很好
- 对于非凸边界的类或类大小非常不一致的情况也适用
- 对噪声和离群点非常敏感

目录

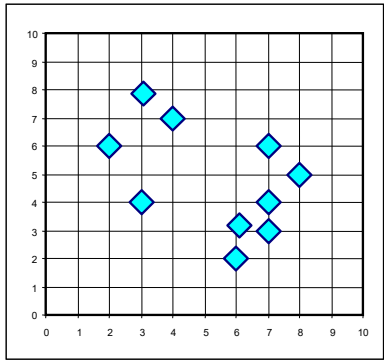
K- medoids

- 用 medoid – 用最靠近类中心的对象作为类的参考点
  - 而不是用类的均值
- 基本策略：
  - 找到 n 对象中的 k 个类，随机确定每个类的代表对象
  - 迭代：
    - 其他所有对象根据距离最近的类中心进行类的分配
    - 计算使得cost最小的类中心
  - 重复直到不再发生变化
  - 代价函数：类内对象与类中心的平均不相似度

K- medoids改进算法：PAM

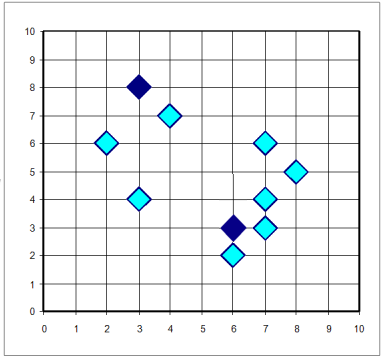
- **Partitioning Around Medoids**
- 基本策略：
  - 找到 n 对象中的 k 个类，随机确定每个类的代表对象
  - 迭代：
    - 其他所有对象根据距离最近的类中心进行类的分配
    - 随机用一个非中心对象替换类中心
    - 类的质量提高则保留替换
- 类的质量
  - 代价函数：类内对象与类中心的平均不相似度

K-Medoids（PAM） 举例



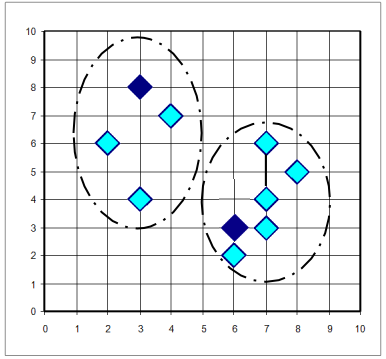
K=2

随机选择 k 个对象作为初始中心

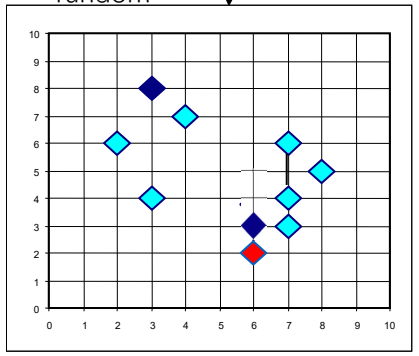


将每个对象分配给最近的中心

总代价 = 20



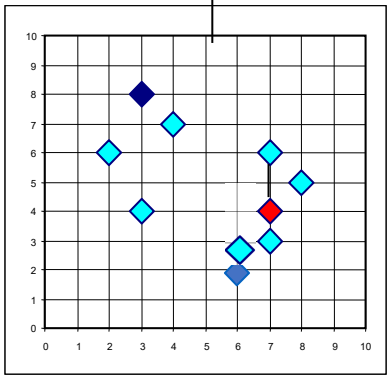
随机选择一个非中心点 O_{random}



计算总代价

代价= 26

不交换



如果质量提升则交换 O 和 O_{random}

循环直到不再发生变化

K-Medoids讨论

- 优点：
  - 当存在噪音和孤立点时，K-medoids 比 K-means 更鲁棒
  - 如果能够迭代所有情况，那么最终得到的划分一定是最优的划分，即聚类结果最好
- 缺点：
  - K-medoids 对于小数据集工作得很好，但不能很好地用于大数据集
    - 计算中心的步骤时间复杂度是O(n^2)，运行速度较慢

基于大样本的改进算法：CLARA

- **Clustering LARge Applications**
- 基本策略：当面对大样本量时：
  - 每次随机选取样本量中的一小部分进行PAM聚类
  - 将剩余样本按照最小中心距离进行归类
  - 在各次重复抽样聚类的结果中，选取误差最小，即中心点代换代价最小的结果作为最终结果

无监督学习总结

- 有监督 v.s. 无监督学习
- 聚类
  - 数据及相似度度量
- 层次聚类
  - 凝聚式 (从下到上)
  - 分裂式 (从上到下)
- K-means 聚类
- K-medoids 聚类（及其变种与改进：PAM，CLARA）