第十章 数据聚类

§10.1 背景知识

§10.2 典型算法

§10.3 应用举例

§10.1 背景知识

一、问题的引入

二、聚类的度量

问题的引入

感知器方法回顾

- 数 据 集：

$$\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$$

- 模　　型：

$$\hat{y} = \mathbf{sgn}(w^T x + w_0)$$

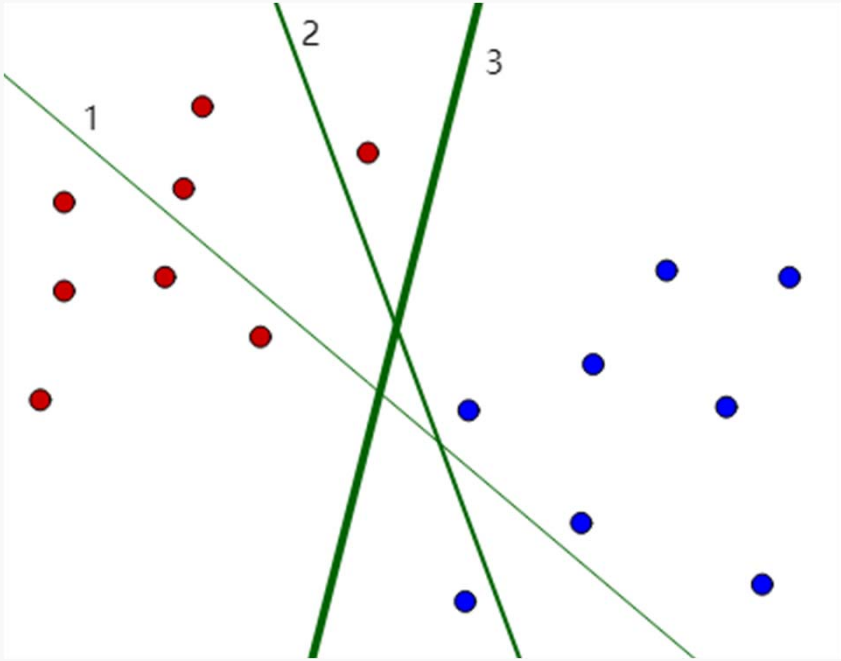
- 惩罚函数：

$$J(w, w_0) = \sum_{y_i(w^T x_i + w_0) \le 0} -y_i(w^T x_i + w_0)$$

- 优化方法：梯度下降法

$$w^{(t+1)} = w^{(t)} - \rho \nabla_w J(w, w_0)$$

$$w_0^{(t+1)} = w_0^{(t)} - \rho \nabla_{w_0} J(w, w_0)$$



感知器收敛过程示意图

问题的引入

- 监督学习方法
  - 需要**带标签**的训练样本
  - 从训练样本中学习出一个**预测模型**
  - 用模型预测新样本（测试数据）的标签
- 如果训练样本不带标签？
  - **标注成本昂贵**，数据量巨大
  - 数据本身的**结构规律**尚待挖掘
  - 可根据样本特征间的**相似性进行聚类（物以类聚）**

问题的引入

聚类问题的定义

- **输入**：m个无标签样本

$$D = \{x_1, x_2, \dots, x_m\}$$

$$x_i = (x_{i1}; x_{i2}; \dots x_{in}) \in R^n$$

- **输出**：k个不相交的簇(clusters)

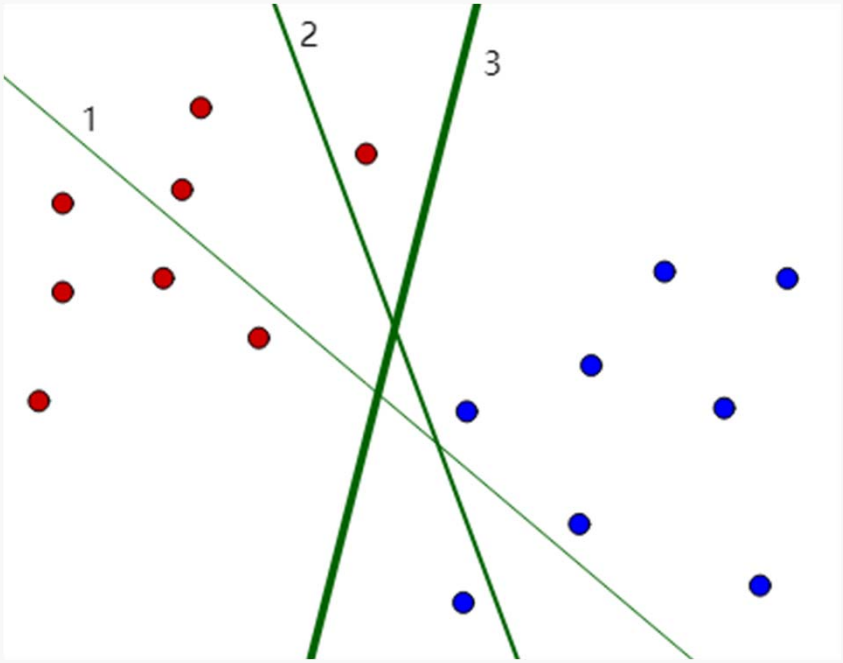
$$\{C_l | l = 1, 2, \dots, k\}$$

$$\text{其中} \begin{cases} \cup_{l=1}^{k} C_l = D, \\ C_{l'} \cap_{l' \neq l} C_l = \emptyset \end{cases}$$

- 定义：

$\lambda_j \in \{1, 2, \dots, k\}$为样本$x_j$的**簇标记**，即

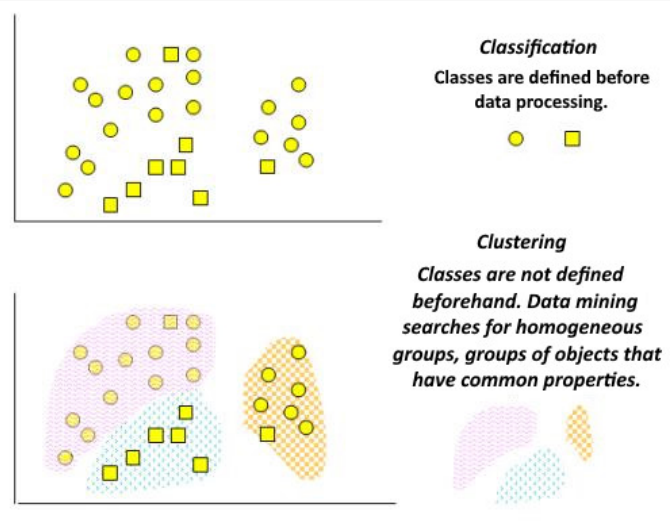
$$x_j \in C_{\lambda_j}$$

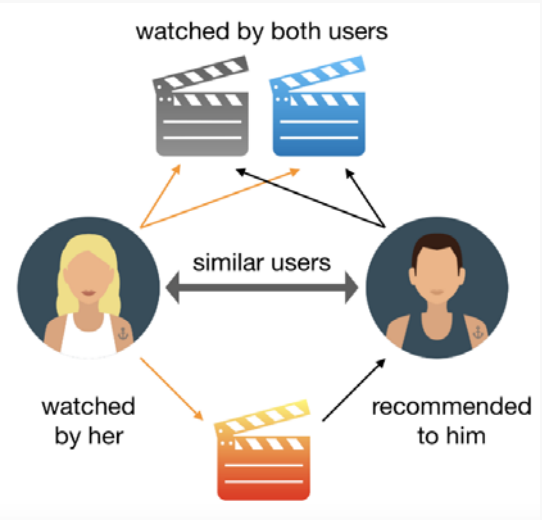


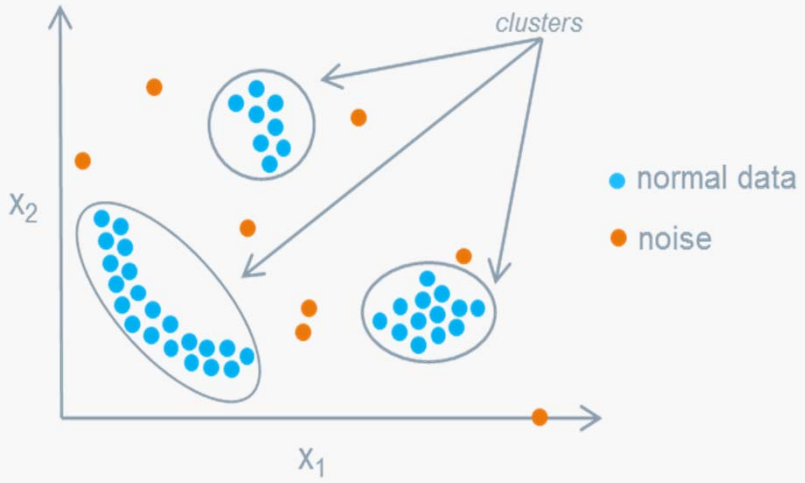
问题的引入

数据聚类的应用

- 数据分类：给无标签数据赋予标签，从而获得带标注数据，**分类的前驱过程**
- 数据推荐：根据之前的习惯进行产品自动推荐，如**商品推荐**、**电影推荐**等
- 异常检测：自动找到不属于任何簇的样本，常被认为是**噪声或异常**







聚类的度量

聚类性能的评价

对于聚类结果$\{C,\lambda\}$：

- 簇内相似度(intra-cluster similarity)高
- 簇间相似度(inter-cluster similarity)低
- 外部指标：以参考模型$\{C^*,\lambda^*\}$为参照标准，定义如下指标

$$
\begin{aligned}
a &= |SS|, \quad & SS &= \{(x_i,x_j)|\lambda_i=\lambda_j,\lambda_i^*=\lambda_j^*,i<j\}\\
b &= |SD|, \quad & SD &= \{(x_i,x_j)|\lambda_i=\lambda_j,\lambda_i^*\neq\lambda_j^*,i<j\}\\
c &= |DS|, \quad & DS &= \{(x_i,x_j)|\lambda_i\neq\lambda_j,\lambda_i^*=\lambda_j^*,i<j\}\\
d &= |DD|, \quad & DD &= \{(x_i,x_j)|\lambda_i\neq\lambda_j,\lambda_i^*\neq\lambda_j^*,i<j\}
\end{aligned}
$$

- Jaccard系数：$JC=\dfrac{a}{a+b+c}$
- FM指数：$FMI=\sqrt{\dfrac{a}{a+b}\cdot\dfrac{a}{a+c}}$
- Rand指数：$RI=\dfrac{2(a+d)}{m(m-1)}$

聚类的度量

聚类性能的评价

对于聚类结果$\{C, \lambda\}$：

- 内部指标：直接考察聚类结果$\{C, \lambda\}$，无需参考模型，定义如下指标

$$\text{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \le i < j \le |C|} \text{dist}(x_i, x_j)$$

$$\text{diam}(C) = \max_{1 \le i < j \le |C|} \text{dist}(x_i, x_j)$$

$$d_{\min}(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} \text{dist}(x_i, x_j)$$

$$d_{\text{cen}}(C_i, C_j) = \text{dist}(x_i, x_j)$$

- **DB指数**：

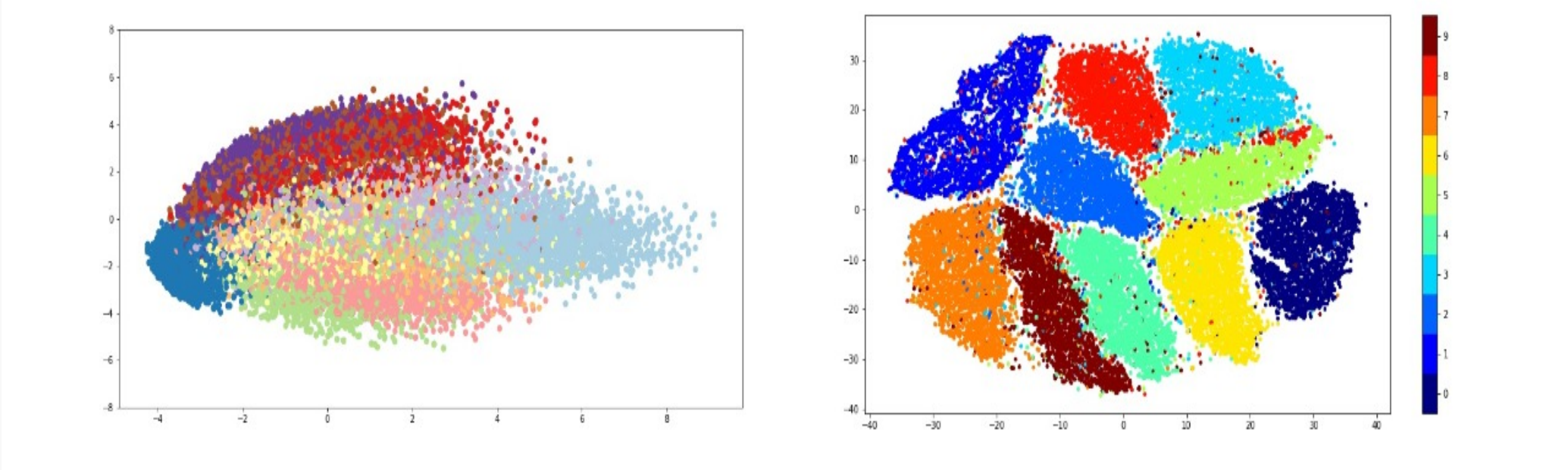
$$\text{DBI} = \frac{1}{k} \sum_{i=1}^{k} \max_{j \ne i} \left(\frac{\text{avg}(C_1) + \text{avg}(C_2)}{d_{\text{cen}}(\mu_i, \mu_j)} \right)$$

- **Dunn指数**：

$$\text{DI} = \min_{1 \le i \le k} \left\{ \min_{j \ne i} \left(\frac{d_{\min}(C_i, C_j)}{\max_{1 \le l \le k} \text{diam}(C_l)} \right) \right\}$$

聚类的度量

聚类性能的评价



MNIST数据集在不同算法下的聚类结果

§10.2 典型算法

一、k均值聚类算法

二、高斯混合聚类算法

三、层次化聚类算法

k均值聚类算法

- 给定数据集$\boldsymbol{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$，$k$均值聚类算法**假定每个聚类簇(cluster)存在一个位于簇中心点的原型(prototype)**，根据样本之间的相似性，给每个样本$\boldsymbol{x}_j$赋予该原型的标签$\boldsymbol{y}_{x_j}\in[1,k]$（$k$已给出）
- 对于簇划分$\boldsymbol{C}=\{C_1,C_2,\ldots,C_k\}$，$k$均值聚类算法最小化如下平方误差：

$$\boldsymbol{E}=\sum_{i=1}^{k}\sum_{x\in C_i}\left||x-\mu_i|\right|_2^2$$

其中$\boldsymbol{\mu_i}=\frac{1}{|C_i|}\sum_{x\in C_i}x$为簇$\boldsymbol{C_i}$的**均值向量**

- 找到最小化E的一组簇$\boldsymbol{C}$需要遍历所有可能的情况：NP-Hard
- 解决办法：**贪心搜索(greedy search)算法**

k均值聚类算法

- k均值聚类算法的贪心搜索
- 重复以下两步

Step(1)：固定各个样本x_j的簇标签y_{x_j}，更新各个簇的中心点μ_i

Step(2)：固定更新各个簇的中心点μ_i，更新各个样本x_j的簇标签y_{x_j}

k均值聚类算法

k均值聚类算法的贪心搜索

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
聚类簇数 k.

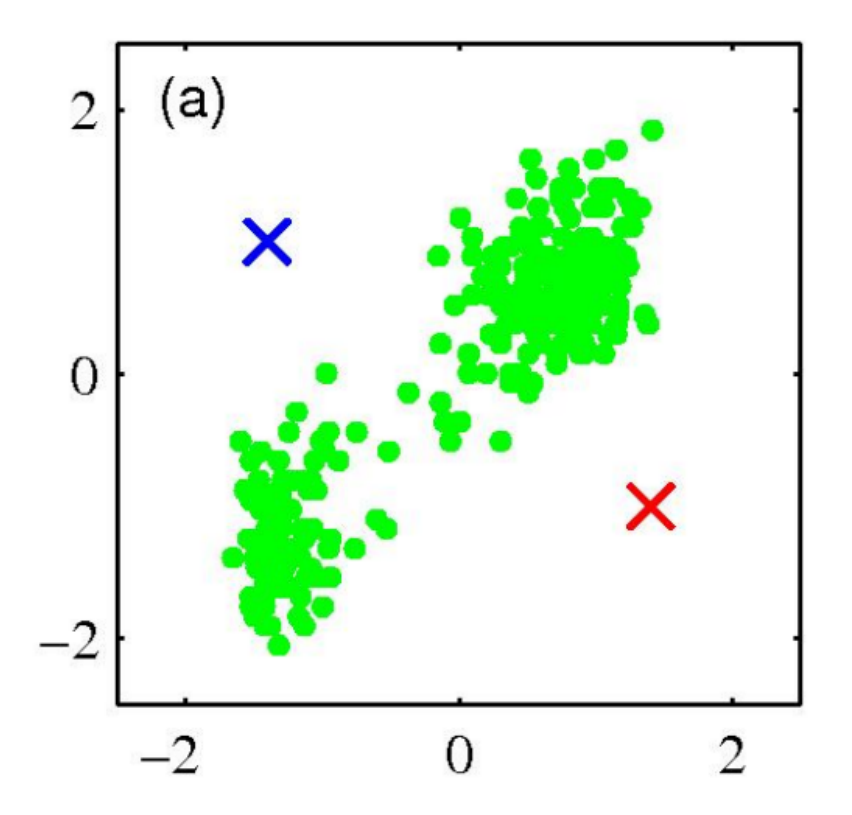
过程:

从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\}$
repeat
  令 $C_i = \varnothing \ (1 \leqslant i \leqslant k)$
  for $j = 1, 2, \ldots, m$ **do**
    计算样本 $\boldsymbol{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i \ (1 \leqslant i \leqslant k)$ 的距离: $d_{ji} = ||\boldsymbol{x}_j - \boldsymbol{\mu}_i||_2$;
    根据距离最近的均值向量确定 $\boldsymbol{x}_j$ 的簇标记: $\lambda_j = \arg\min_{i \in \{1,2,\ldots,k\}} d_{ji}$;
    将样本 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \bigcup \{\boldsymbol{x}_j\}$;
  end for
  for $i = 1, 2, \ldots, k$ **do**
    计算新均值向量: $\boldsymbol{\mu}'_i = \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x}$;
    if $\boldsymbol{\mu}'_i \neq \boldsymbol{\mu}_i$ **then**
      将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}'_i$
    else
      保持当前均值向量不变
    end if
  end for
until 当前均值向量均未更新

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

k均值聚类算法

- k均值聚类算法的贪心搜索举例
  - 初始化

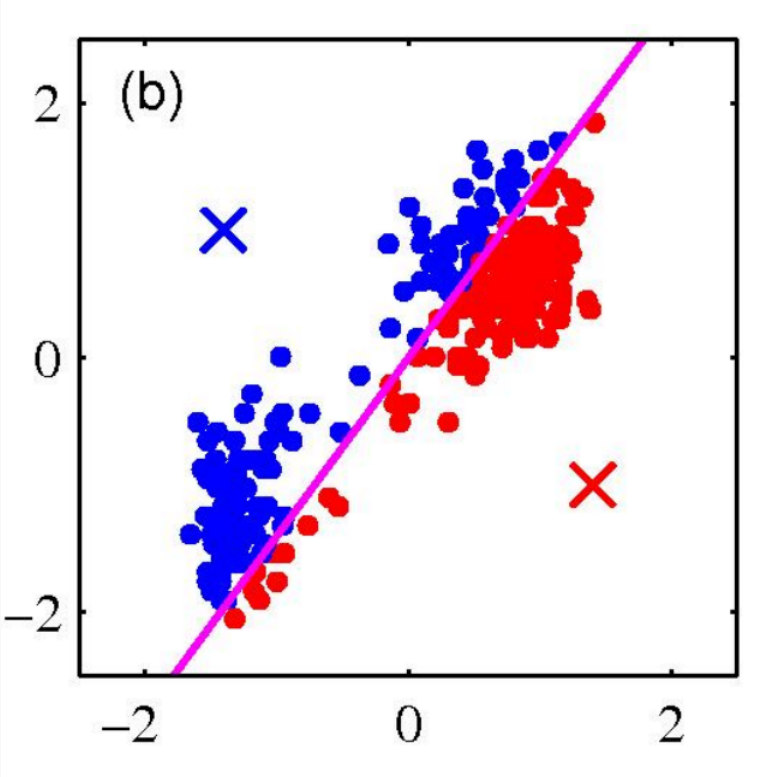


· 随机选取k个点作为簇中心点

· K=2

k均值聚类算法

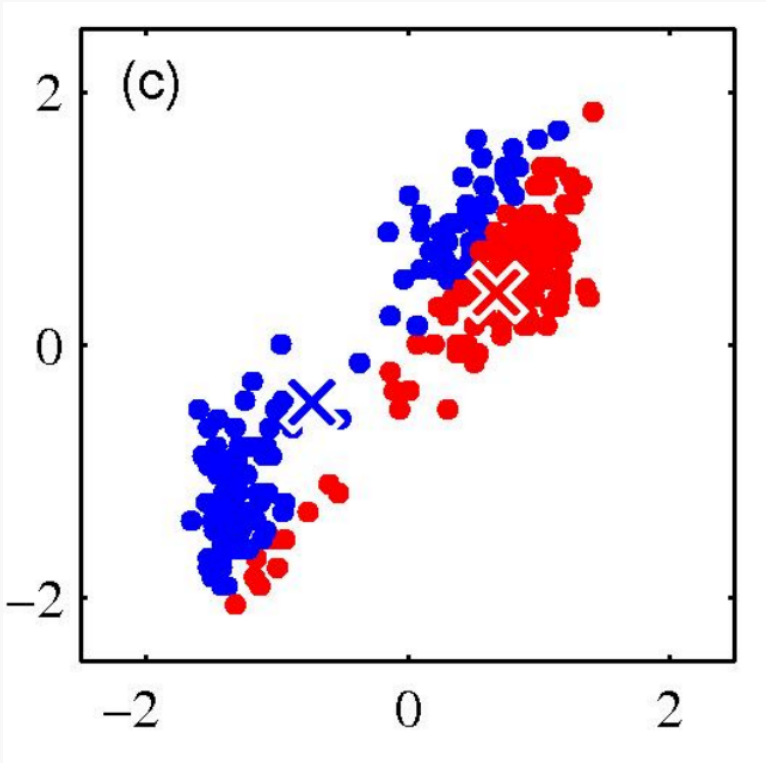
- k均值聚类算法的贪心搜索举例
  - 迭代过程



1. 寻找距离样本x_i最近的簇中心点

k均值聚类算法

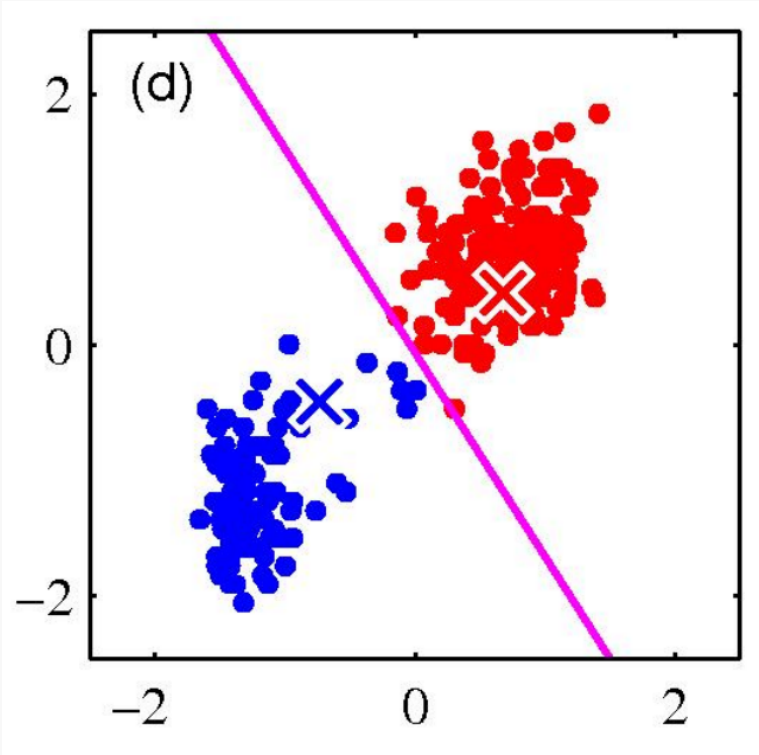
- k均值聚类算法的贪心搜索举例
  - 迭代过程



1. 更新簇中心点坐标

k均值聚类算法

- k均值聚类算法的贪心搜索举例
  - 迭代过程

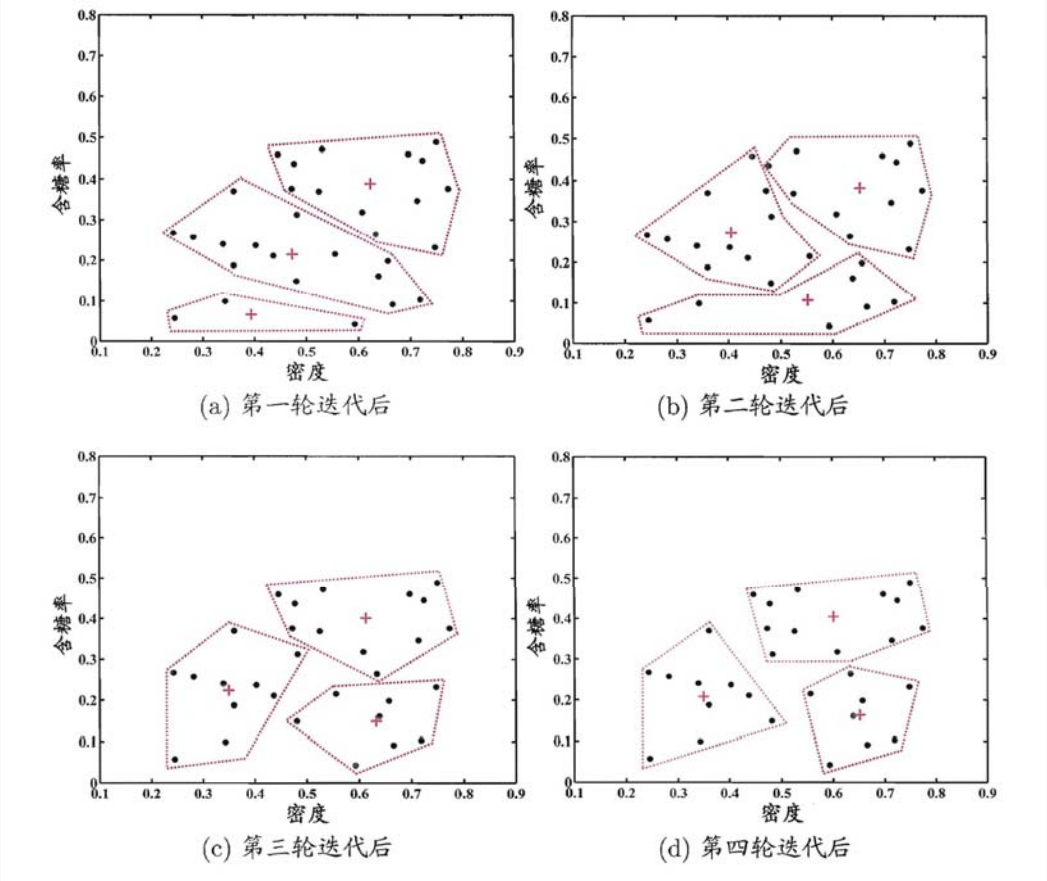


1. 寻找距离样本x_i最近的簇中心点
2. 更新簇中心点坐标

不断迭代1、2，直至收敛

k均值聚类算法

- k均值聚类算法的贪心搜索举例



西瓜数据集k均值聚类算法结果

k均值聚类算法

k均值聚类算法的收敛性

- 目标函数：$\min_{\mu}\min_{C}\sum_{x\in C_i}|x-\mu_i|^2$
- 回顾算法的迭代过程：

1、固定μ，优化C：

$$\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2=\min_{C}\sum_{i}^{n}|x_i-\mu_{\underset{j}{argmin}\,dist(x_i,\mu_j)}|^2$$

对应于第一步的寻找最近邻

2、固定C，优化μ：$\min_{\mu}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$

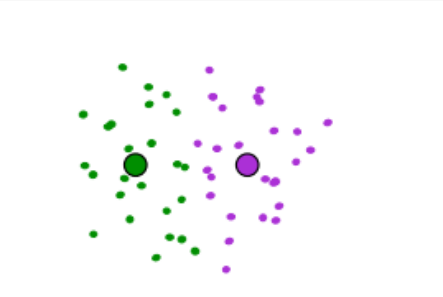
- 令$\frac{\partial\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2}{\partial\mu_i}=0$，得到：

$$\mu_i=\frac{1}{|C_i|}\sum_{x\in C_i}x$$

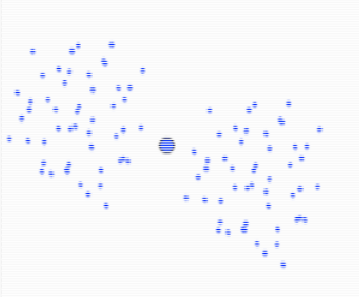
对应于第二步的均值更新

k均值聚类算法

- k均值聚类算法的优缺点
  - 优点
    - 比较高效：$O(Nkm)$，N为迭代次数上限，k为簇的数量，m为样本数
    - 在局部最优解处终止：迭代过程保障了目标函数不断下降直至收敛
  - 缺点
    - k作为先验需提前指定



K=1更好



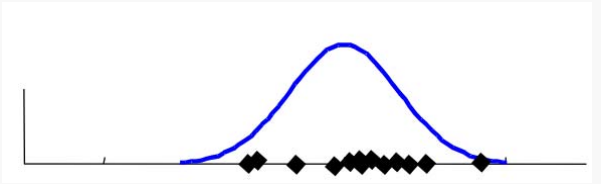
K=2更好

- 无法处理噪声数据(noisy data)和离群数据(outliers)
- 不适用于非凸形状的簇

高斯混合聚类算法

高斯函数

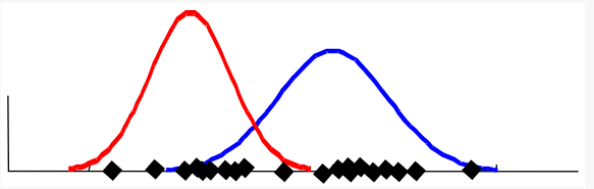
$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\eta)^2}{2\sigma^2}}$$



高斯混合模型

位于第i个高斯模型的先验概率 $P(C = i) = w_i$

第i个高斯模型的概率密度函数 $P(x|C = i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$



给定高斯混合模型θ，样本x的似然概率

$$P(x|\theta) = \sum_i P(C = i, x|\theta) = \sum_i P(x|C = i, \theta)P(C = i|\theta)$$

$$= \sum_i w_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$$

高斯混合聚类算法

- 高斯混合模型的性质
  - 假设已知GMM成分的数目(k)、它们各自的先验概率(w_i)和各个高斯模型的参数(μ_i, Σ_i)
  - 通过GMM生成新的样本
    - 根据先验概率分布($\sum_i w_i = 1$)随机选取某个高斯模型，如第j个高斯模型
    - 从第j个高斯模型的分布$N(\mu_j, \Sigma_j)$中采样，从而得到新的样本数据
  - 高斯混合模型的似然函数

$$p(x_1, \dots, x_m|\theta) = \prod_{i=1}^{m}(\sum_{j=1}^{k} p(x_i|C = j)w_j)$$

高斯混合聚类算法

高斯混合聚类算法流程

输入: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$;
高斯混合成分个数 k.

过程:

1: 初始化高斯混合分布的模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$
2: **repeat**
3: **for** $j = 1, 2, \ldots, m$ **do**
4: 根据式(9.30)计算 $\boldsymbol{x}_j$ 由各混合成分生成的后验概率, 即
$\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \boldsymbol{x}_j)$ $(1 \leqslant i \leqslant k)$
5: **end for**
6: **for** $i = 1, 2, \ldots, k$ **do**
7: 计算新均值向量: $\boldsymbol{\mu}_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{m} \gamma_{ji}}$;
8: 计算新协方差矩阵: $\boldsymbol{\Sigma}_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} (\boldsymbol{x}_j - \boldsymbol{\mu}_i')(\boldsymbol{x}_j - \boldsymbol{\mu}_i')^{\mathrm{T}}}{\sum_{j=1}^{m} \gamma_{ji}}$;
9: 计算新混合系数: $\alpha_i' = \frac{\sum_{j=1}^{m} \gamma_{ji}}{m}$;
10: **end for**
11: 将模型参数 $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leqslant i \leqslant k\}$ 更新为 $\{(\alpha_i', \boldsymbol{\mu}_i', \boldsymbol{\Sigma}_i') \mid 1 \leqslant i \leqslant k\}$
12: **until** 满足停止条件
13: $C_i = \varnothing$ $(1 \leqslant i \leqslant k)$
14: **for** $j = 1, 2, \ldots, m$ **do**
15: 根据式(9.31)确定 $\boldsymbol{x}_j$ 的簇标记 λ_j;
16: 将 $\boldsymbol{x}_j$ 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \bigcup \{\boldsymbol{x}_j\}$
17: **end for**

输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

$$p_{\mathcal{M}}(z_j = i|\boldsymbol{x}_j) = \frac{P(z_j = i) \cdot p_{\mathcal{M}}(\boldsymbol{x}_j|z_j = i)}{p_{\mathcal{M}}(\boldsymbol{x}_j)}$$

$$= \frac{\alpha_i \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(\boldsymbol{x}_j|\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}$$

高斯混合聚类算法

- 高斯混合聚类算法用于数据聚类
  - 第一步：确定聚类簇的个数：k
  - 第二步：随机初始化高斯混合模型的参数θ(包括了μ, Σ, w等)
  - E-step：得到每个样本的$p_{i,j}$，每个高斯分布的p_i
  - M-step：基于E-step的输出，对参数进行调整

$$\mu_i \leftarrow \sum_j \frac{p_{i,j} x_j}{p_i}$$

$$\Sigma_i \leftarrow \sum_j \frac{p_{i,j} x_j x_j^T}{p_i}$$

$$w_i \leftarrow \frac{p_i}{\sum_j p_j}$$

高斯混合聚类算法

- 高斯混合聚类算法的优缺点
  - 优点
    - 比较高效：$O(Nkm)$, N为迭代次数上限，k为簇的数量，m为样本数
    - 可解释性较强：为各个簇学习了各自的生成概率模型，能生成新的数据
    - 优化目标较为直观：数据关于分布的似然函数
  - 缺点
    - k作为先验需要提前指定
    - 容易陷入局部最优
    - 不适用于非凸形状的簇

高斯混合聚类算法

- 对无标注的西瓜数据使用EM算法聚类 $x=$(密度，含糖率)
  - 第一步：**初始化参数**
    - 令高斯混合成分的个数$k=3$，各成分的先验概率为$w_1=w_2=w_3=1/3$
    - 令均值$\mu_1=x_6$，$\mu_2=x_{22}$，$\mu_3=x_{27}$，令方差$\Sigma_1=\Sigma_2=\Sigma_3=\begin{pmatrix}0.1 & 0\\ 0 & 0.1\end{pmatrix}$

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

高斯混合聚类算法

- 第二步：进行EM算法的“E”步，即计算样本由各混合成分生成的后验概率
  - 以x_1为例，后验概率如下：

$$p_{1,1} = p(C=1|x_1) = \frac{p(x_1|C=1)w_1}{\sum_k p(x_1|C=k)w_k} = 0.219,\ p_{2,1} = 0.404,\ p_{3,1} = 0.377$$

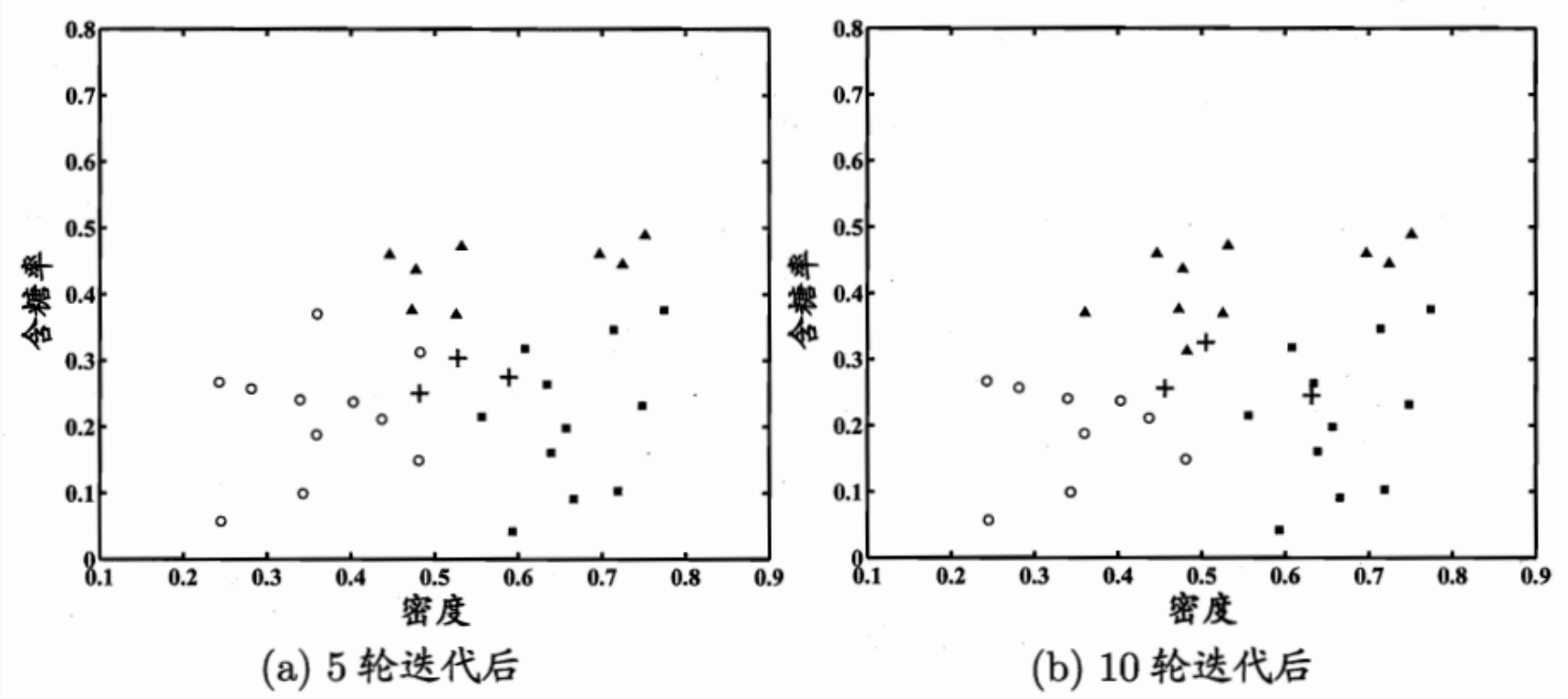
  - 和x_1类似，计算得到所有样本的后验概率$\{p_{i,j}\}$和$p_i = \sum_j p_{i,j}$
- 第三步：进行EM算法的“M”步，即根据后验概率更新模型参数

$$\mu_i \leftarrow \sum_j \frac{p_{i,j}x_j}{p_i} \qquad \Sigma_i \leftarrow \sum_j \frac{p_{i,j}(x_j-\mu_i)(x_j-\mu_i)^T}{p_i} \qquad w_i \leftarrow \frac{p_i}{\sum_j p_j}$$

  - 得到更新后的$w_1 = 0.361,\ w_2 = 0.323,\ w_3 = 0.316$
  - $\mu_1 = (0.491; 0.251),\ \mu_2 = (0.571; 0.281),\ \mu_3 = (0.534; 0.295)$
  - $\Sigma_1 = \begin{pmatrix} 0.025 & 0.004 \\ 0.004 & 0.016 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.023 & 0.004 \\ 0.004 & 0.017 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.024 & 0.005 \\ 0.005 & 0.016 \end{pmatrix}$
- 重复第二步和第三步，直至达到最大迭代轮数或者模型收敛

高斯混合聚类算法

- 不同迭代轮数之后，样本的聚类结果如下图所示



(a) 5 轮迭代后　　(b) 10 轮迭代后

加号表示各类簇的均值

层次化聚类算法

- 初始化：每一个样本自成一簇

  ⇒ m个点，m个簇

- 聚类过程：找到最“相似”的两个簇，将其合并

  ⇒ m个点，$m-1$个簇

- 终止条件：直至所有样本都属于一个簇 或簇的数目降到预设值

  ⇒ m个点，$k(1 \leq k \leq m)$个簇

簇之间的相似性（集合之间的距离）如何定义?

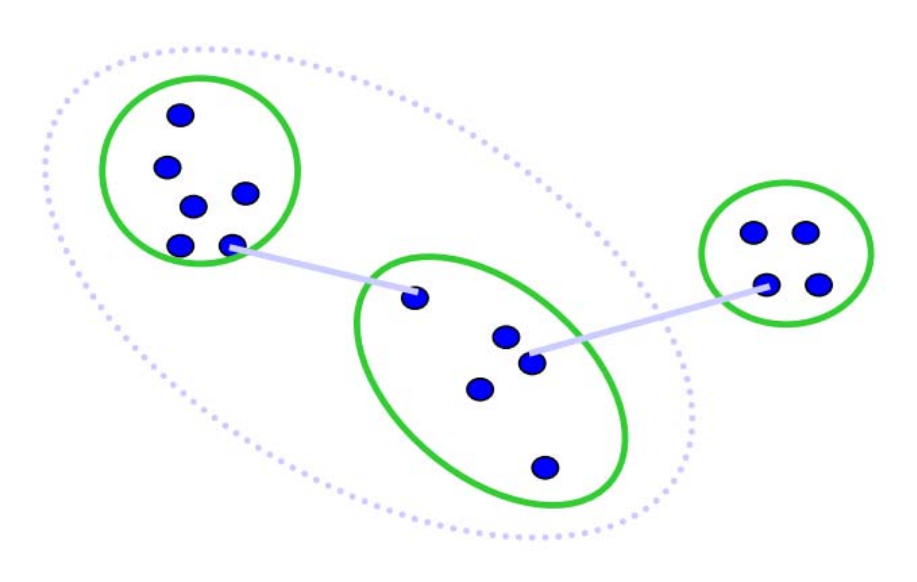
层次化聚类算法

- Cluster与Cluster之间的距离
  - Single Link

    两个簇之间样本对的最小距离

$$d_{min}(C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z)$$

样本之间的相似性度量 $dist()$ 已定义



容易产生形状较为狭长的cluster

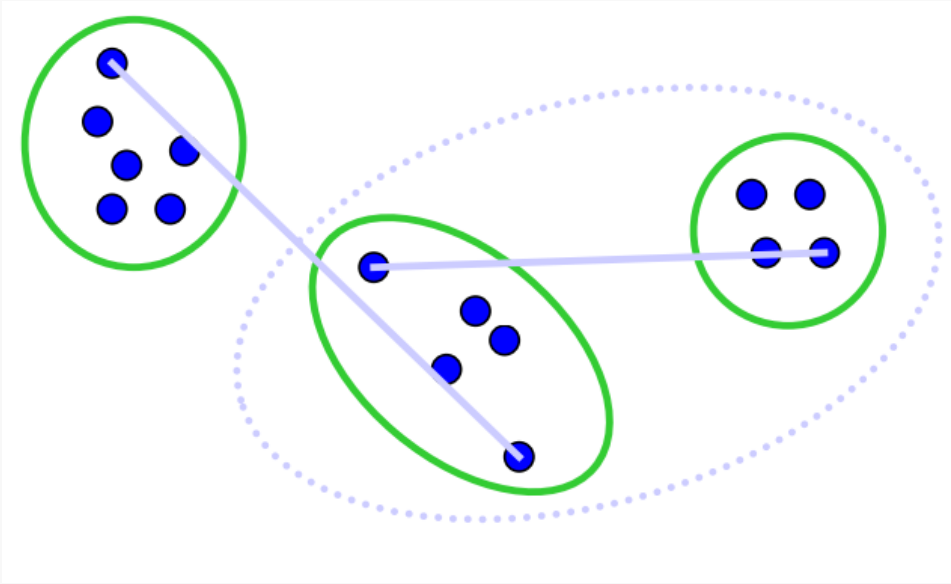
层次化聚类算法

- Cluster与Cluster之间的距离
  - Complete Link

两个簇之间样本对的最大距离

样本之间的相似性度量 $dist()$ 已定义

$$d_{max}(C_i, C_j) = \max_{x \in C_i, z \in C_j} dist(x, z)$$



容易产生形状较为紧致的cluster

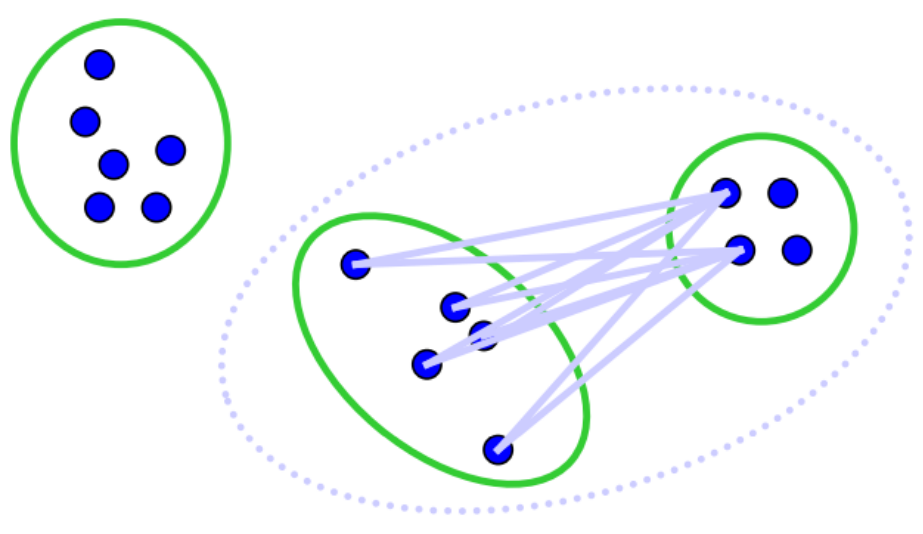
层次化聚类算法

- Cluster与Cluster之间的距离
  - Average Link

两个簇之间所有样本对的平均距离

样本之间的相似性度量 $dist()$ 已定义

$$d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{x \in C_j} dist(x, z)$$



最广泛使用

对噪声鲁棒

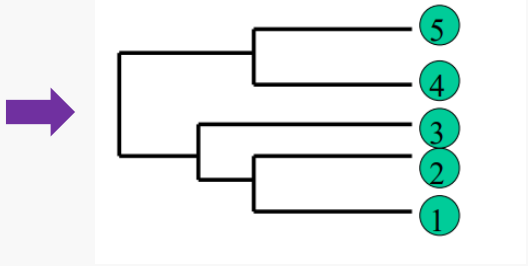
层次化聚类算法

层次化聚类算法的优缺点

优点

- 不过于依赖预先设定的簇数量k
- 有灵活的相似性（距离）度量方式
- 聚类结果的表示方式相对符合人们的认知

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & & & & \\ 2 & 2 & 0 & & & \\ 3 & 6 & 3 & 0 & & \\ 4 & 10 & 9 & 7 & 0 & \\ 5 & 9 & 8 & 5 & 4 & 0 \end{array}$$



树形式的结果，可用于
亲属关系聚类、物种基因聚类

缺点

- 效率低，$O(m^3)$
- 簇结果有一定主观性（树形图需要进一步人为划分成簇）

层次化聚类算法

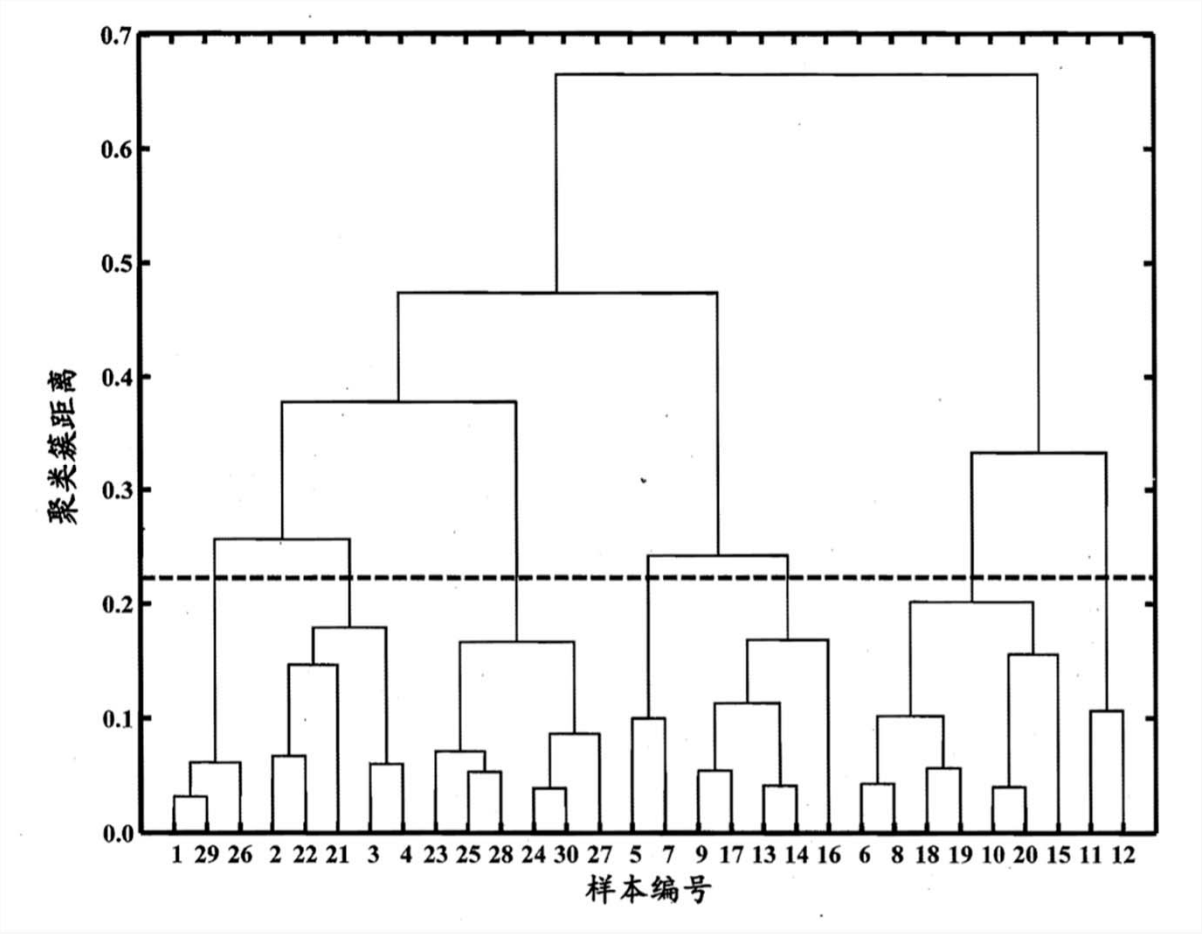
- 以表9.1西瓜数据集4.0为例

表 9.1 西瓜数据集 4.0

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

层次化聚类算法

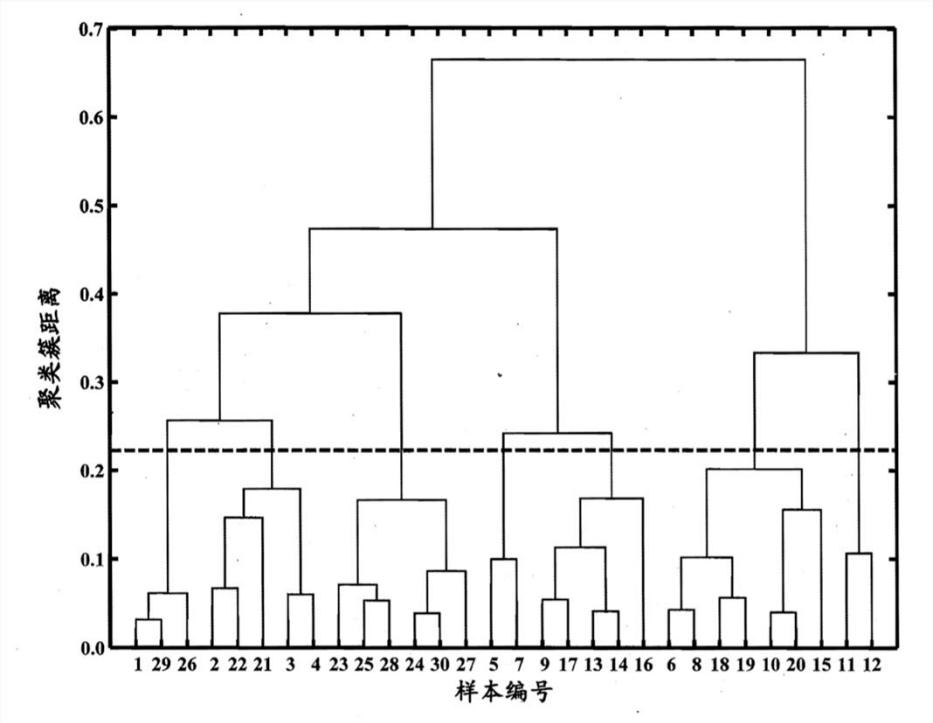
- 执行AGNES算法直到所有样本出现在同一个簇中，得到如下树状图



层次化聚类算法

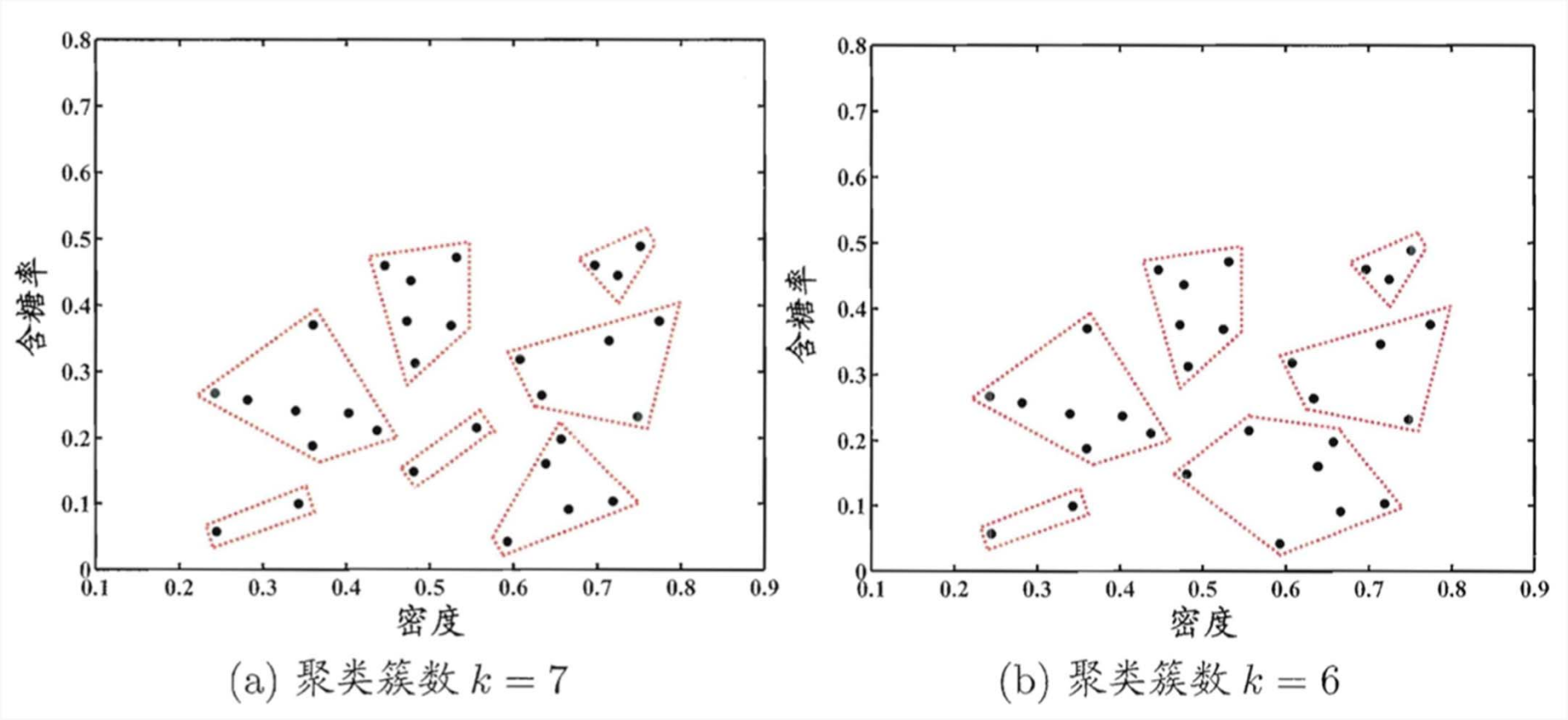
- 树状图中每层链接一组聚类簇
- 在树状图的特定层次上进行分割，可以得到相应的簇划分结果
  - 以图中虚线为例
  - 可以得到7个聚类簇的结果

$$C_1 = \{x_1, x_{26}, x_{29}\}$$
$$C_2 = \{x_2, x_3, x_4, x_{21}, x_{22}\}$$
$$C_3 = \{x_{23}, x_{24}, x_{25}, x_{27}, x_{28}, x_{30}\}$$
$$C_4 = \{x_5, x_7\}$$
$$C_5 = \{x_9, x_{13}, x_{14}, x_{16}, x_{17}\}$$
$$C_6 = \{x_6, x_8, x_{10}, x_{15}, x_{18}, x_{19}, x_{20}\}$$
$$C_7 = \{x_{11}, x_{12}\}$$



层次化聚类算法

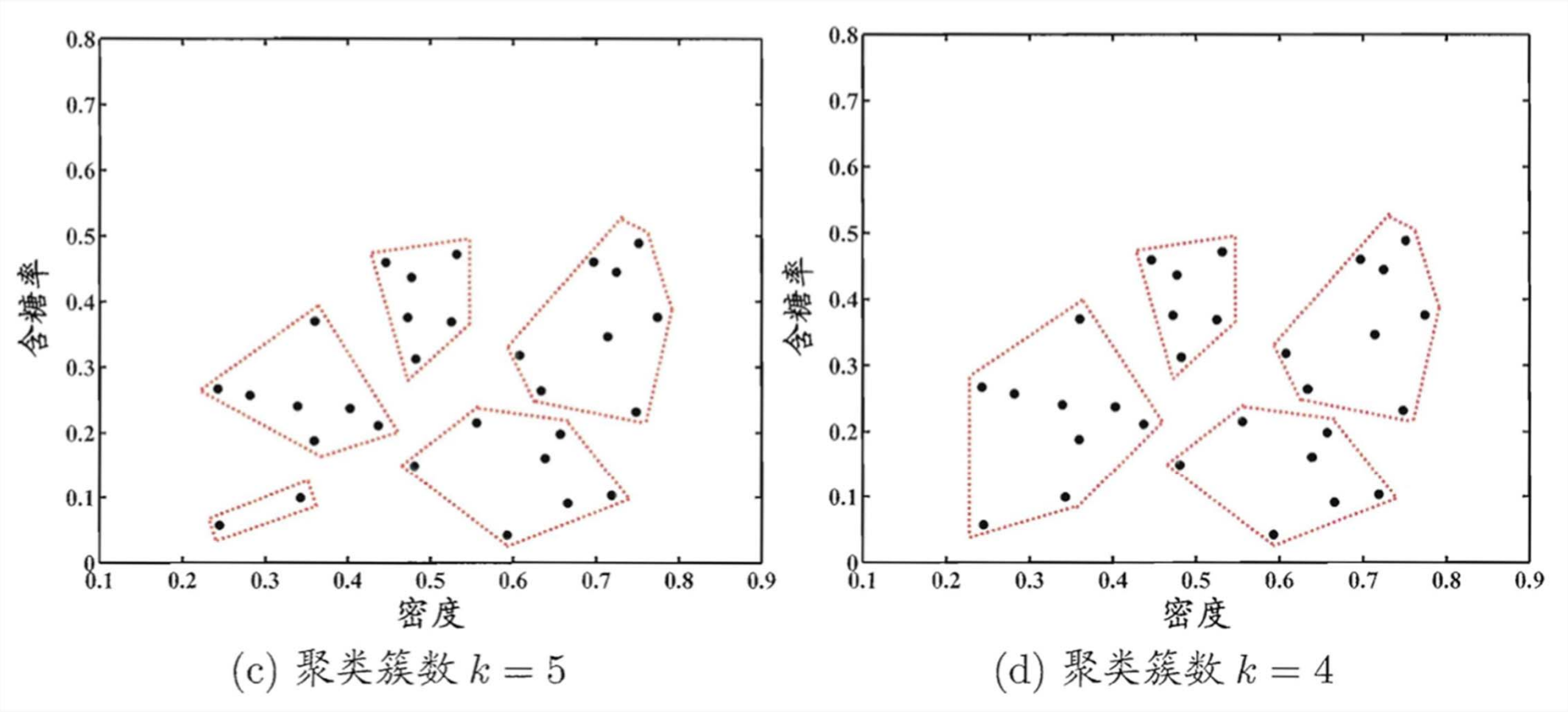
- 下图分别展示了聚类簇数为7和6的划分结果



(a) 聚类簇数 $k = 7$　　(b) 聚类簇数 $k = 6$

层次化聚类算法

- 下图展示了聚类簇数为5和4的划分结果



三种聚类方法的对比

	k均值聚类	高斯混合聚类	层次化聚类
运行时间	较快 （复杂度关于k、样本数线性增长）	最快 （复杂度关于k、样本数线性增长）	较慢，$O(m^3)$
先验知识要求	仅需已知相似性/距离度量	先验要求较高	先验要求最高 （高斯分布）
输入参数	None	k（簇的数目）	k（簇的数目）
簇的特征	需要人为对簇进行主观上的划分（算法仅输出一个树形结构）	恰好k个簇	恰好k个簇

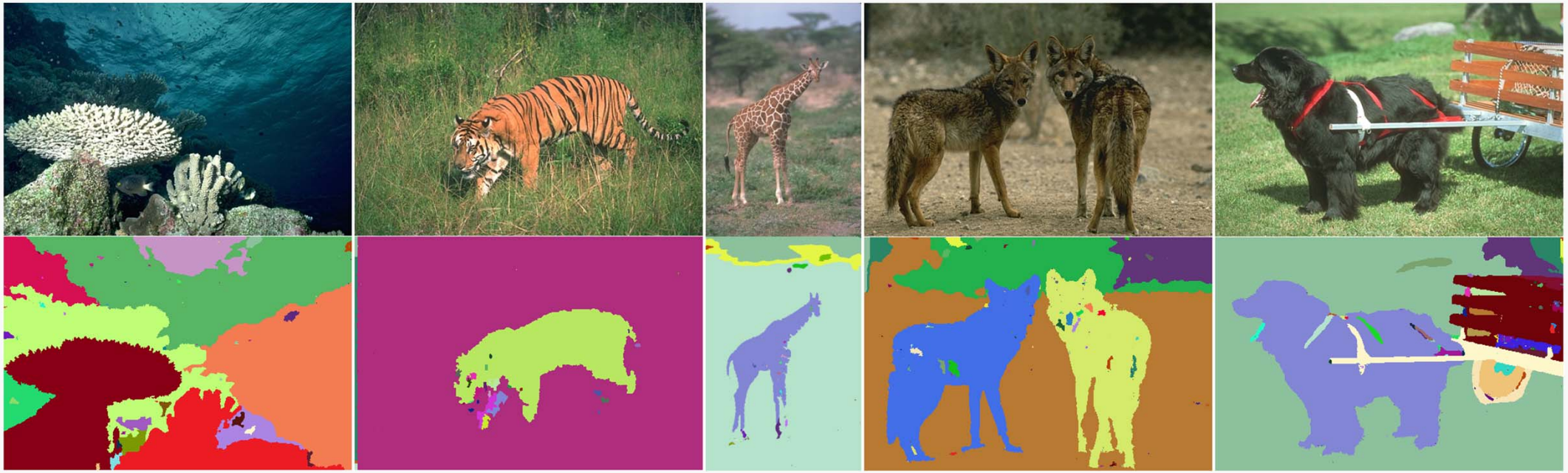
§10.3 应用举例

一、非监督图像分割

二、大规模数据聚类

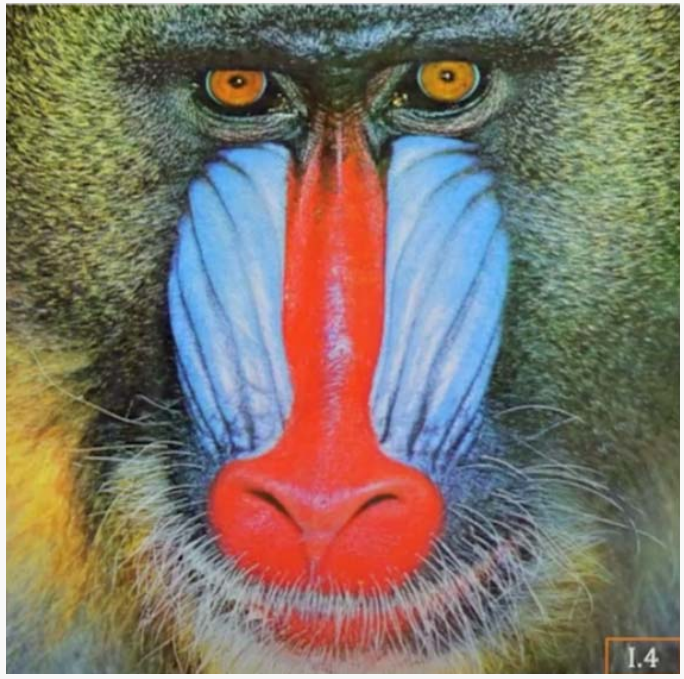
应用实例：非监督图像分割

- 非监督图像分割
  - **定义**：为每个图像的像素分配一个标签（非事先定义）
  - **挑战**：样本特征选取、图像标签数目的确定

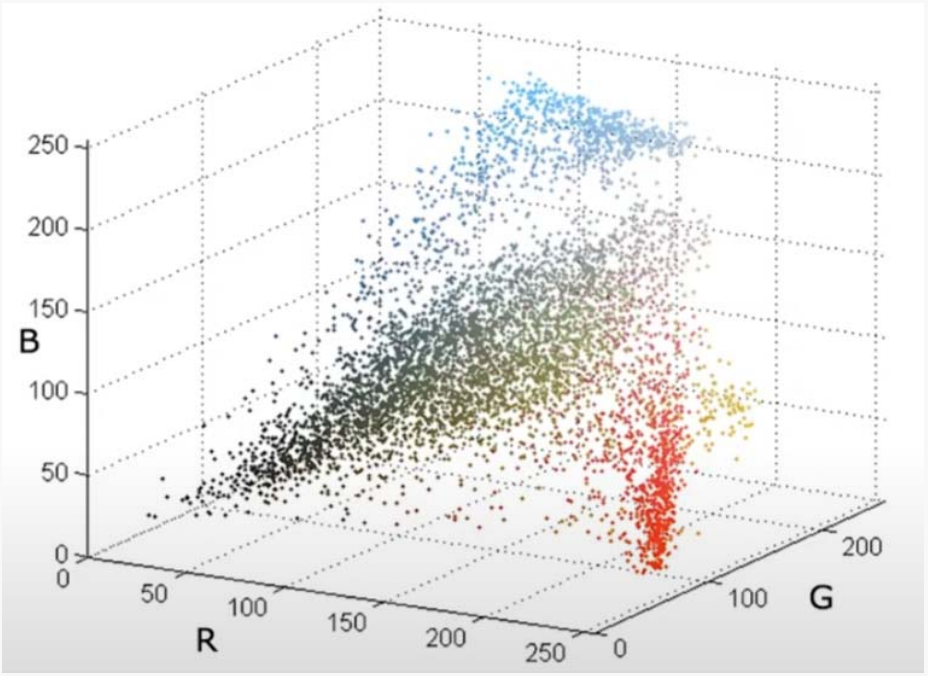


应用实例：非监督图像分割

- k均值聚类算法用于非监督图像分割
  - 图像像素在Euclidean空间下的可视化



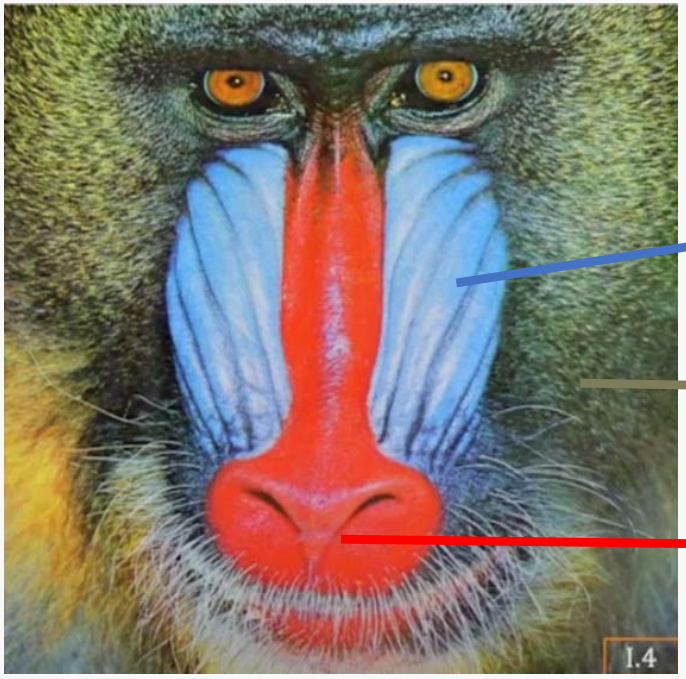
输入图像



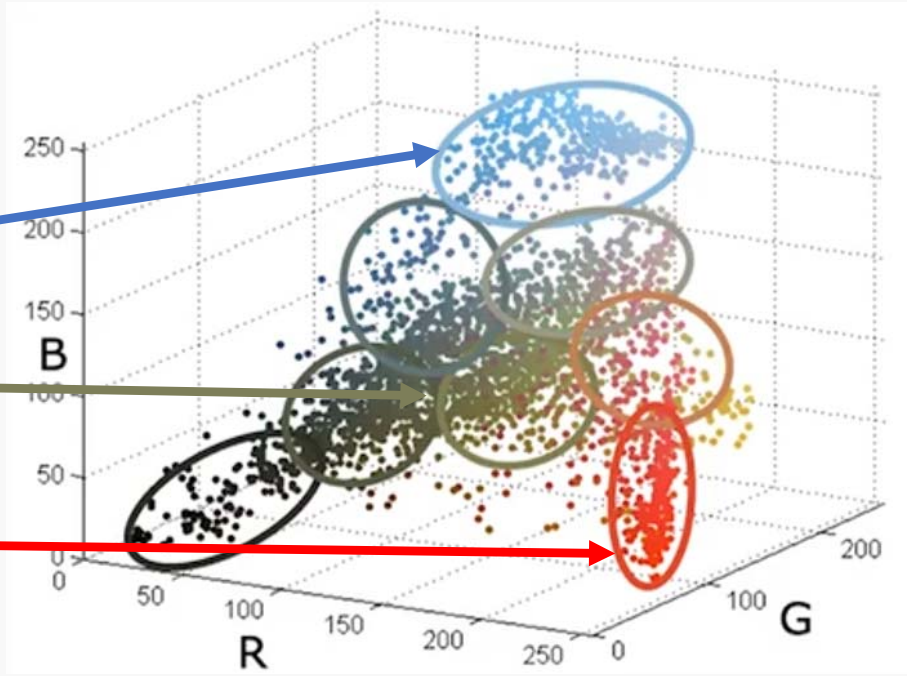
图像像素的RGB空间分布

应用实例：非监督图像分割

- k均值聚类算法用于非监督图像分割
  - 在RGB空间对图像像素特征使用k均值聚类算法



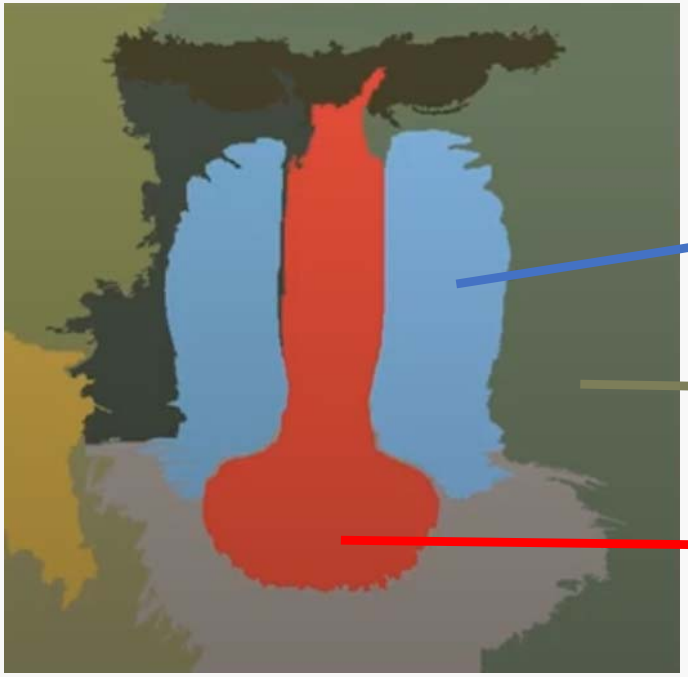
输入图像



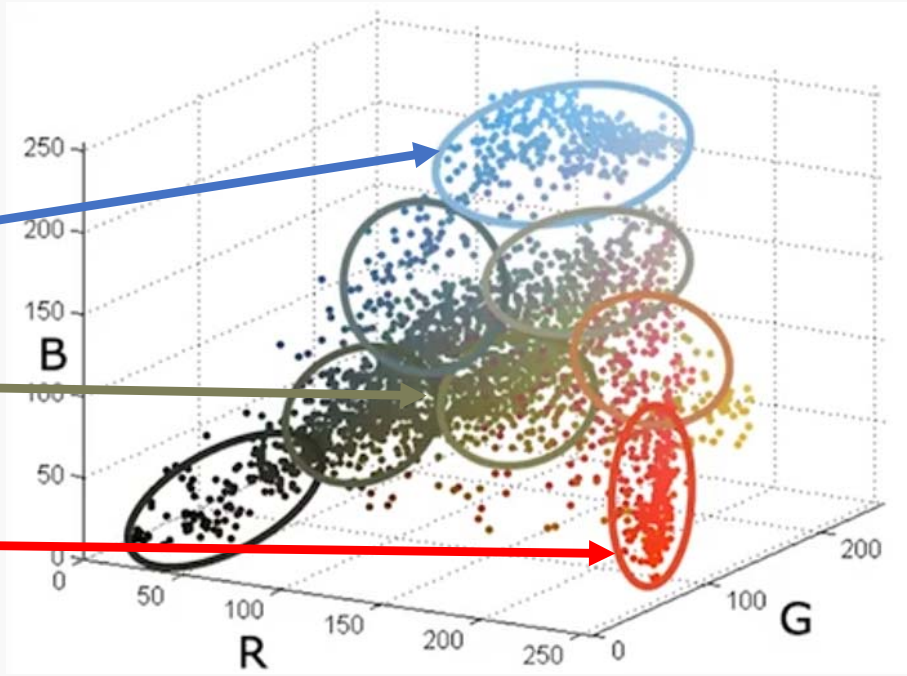
图像像素的k均值算法结果

应用实例：非监督图像分割

- k均值算法用于非监督图像分割
  - 在RGB空间对图像像素特征使用k均值聚类算法



图像的分割结果



图像像素的k均值算法结果

应用实例：非监督图像分割

- k均值算法用于非监督图像分割
  - 定义：为每个图像的像素分配一个标签（非事先定义）
  - 困难：样本特征选取、图像标签数目的确定

除RGB外，像素坐标、深度值也可用于特征相似性度量

k-means中，k为人为指定

- Mean-Shift算法用于非监督图像分割
  - 不需要提前指定k — k-means需要提前指定
  - 适用于非凸形状的簇 — k-means仅适用于凸的簇
  - 对离群点较为鲁棒 — k-means易受噪声影响，对初始化较为敏感

应用实例：非监督图像分割

- Mean-Shift用于非监督图像分割
  - 输入：N个像素在特征（RGB、坐标、深度）空间的分布$\{f_1, f_2, \ldots, f_N\}$
  - Mean-Shift的目标：寻找该分布的模点(modes)
- 算法流程
  - S1：对第i个像素，设置初始簇均值$\mu_i = f_i$
  - S2：对每个簇均值μ_i，迭代以下过程：
    - S2.1：以μ_i为中心点建立滑窗W_i，可以为圆形或方形
    - S2.2：在W_i内计算该滑窗内的样本均值μ
    - S2.3：若$\left|\left|\mu - \mu_i\right|\right|^2 \geq \epsilon$，则更新$\mu_i \leftarrow \mu$；反之停止迭代，当前$\mu_i$为一个新的模点
  - S3：将属于同一个模点的像素标记为同一个簇中的样本

应用实例：非监督图像分割

- 图像分割结果

输入图像

k-means分割结果(k=16)

Mean-Shift分割结果(半径=23)

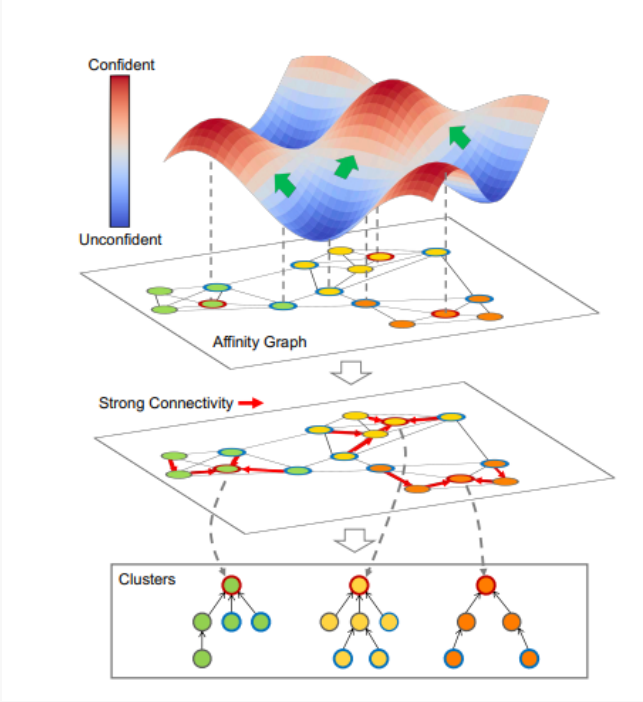
应用实例：非监督图像分割

- Mean-Shift在其他图像上的分割结果

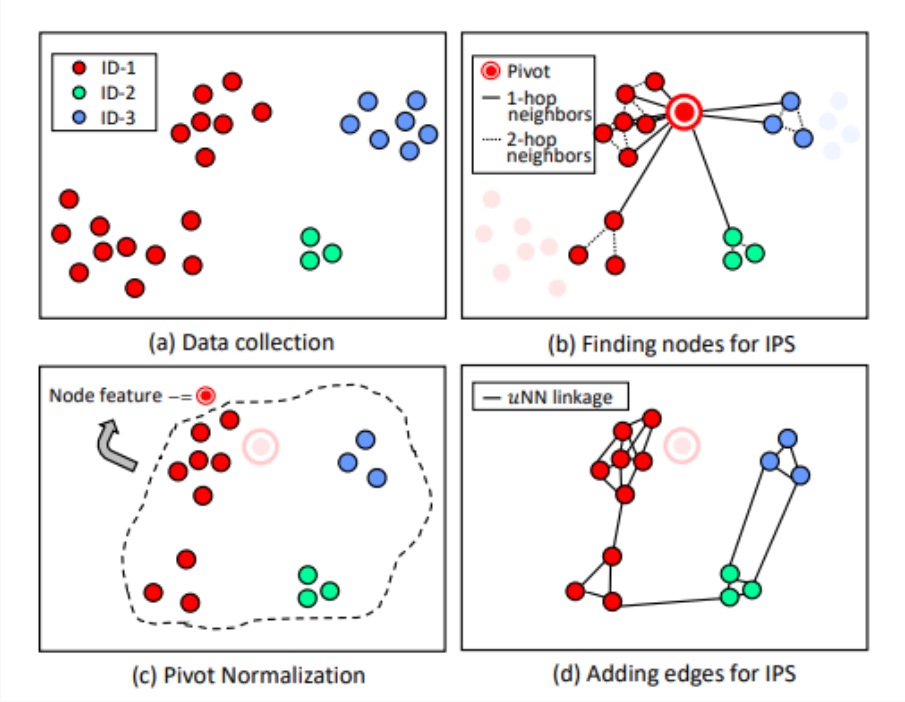



应用实例：大规模图像聚类

- 大规模人脸图像聚类
  - **应用**：批量标注人脸图像、对大规模人脸图像进行组织管理
  - 通常用图网络处理该类问题，可分为**全局方法**和**局部方法**



全局方法



局部方法

应用实例：大规模图像聚类

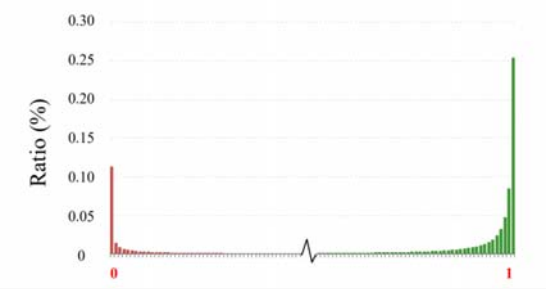
大规模人脸图像聚类

	全局方法	局部方法
优点	用图卷积网络使用全图作为输入，**训练测试速度快**	在局部区域内进行训练与测试，可以处理大规模数据集
缺点	由于图卷积网络使用全图作为输入，无法处理大规模数据集，网络也无法做到很深	不能很好地把握全局结构，无法在全图间进行信息传递，只能在局部进行信息传播，**训练测试非常慢**

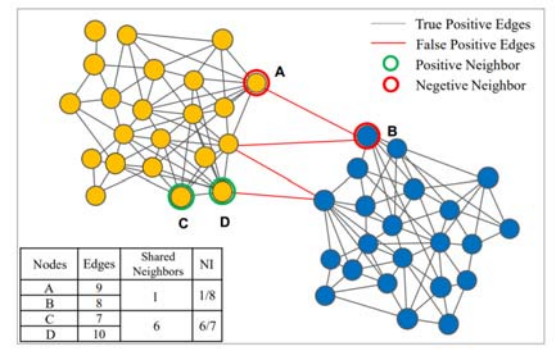
应用实例：大规模数据聚类

大规模人脸图像聚类

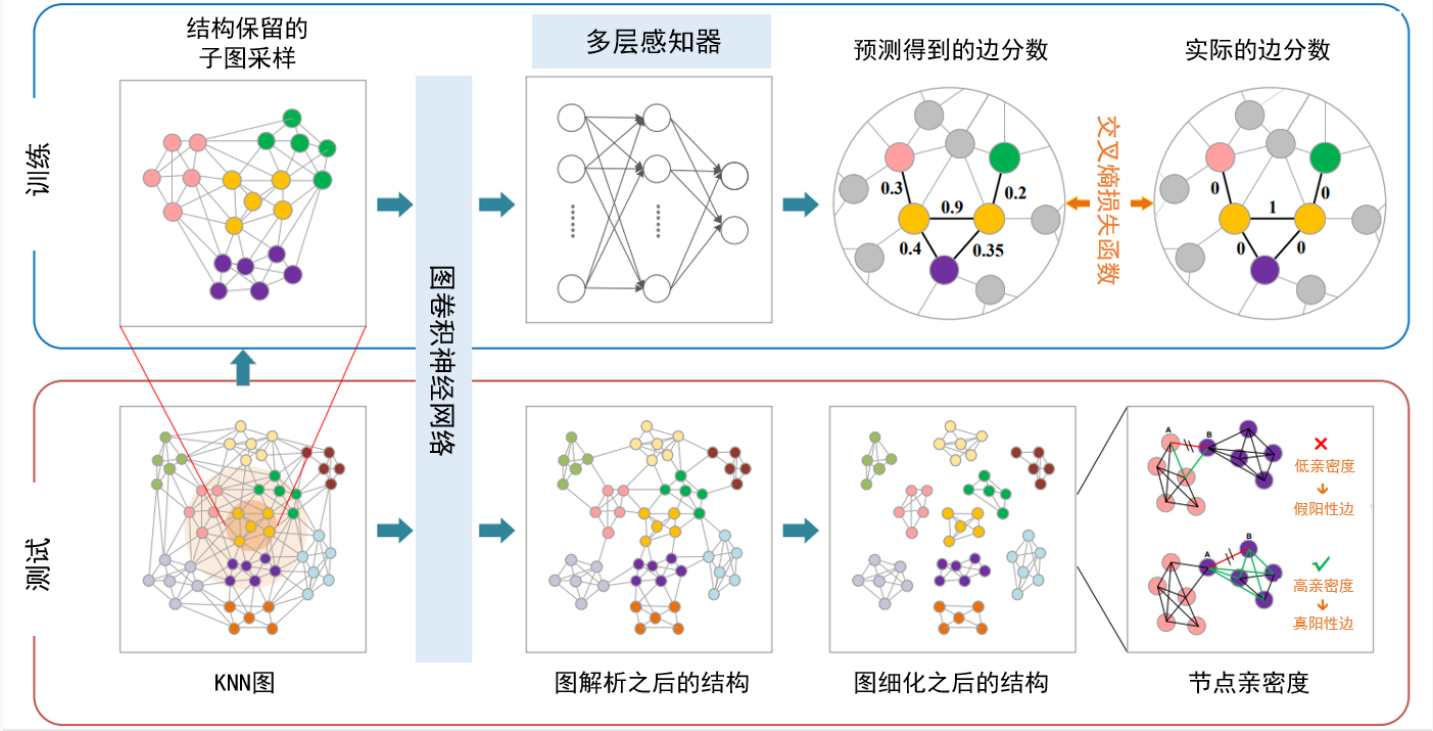
- **我们方法**：解析得到子图，再在子图之间进行信息传递



图解析



图细化



[1] S. Shen, W. Li, Z. Zhu, G. Huang, D. Du, J. Lu, and J. Zhou, Structure-aware Face Clustering on a Large-Scale Graph with 10^7 Nodes. *CVPR*, 2021.

应用实例：大规模图像聚类

- 大规模人脸图像聚类