

周志华 著

MACHINE
LEARNING

机器学习

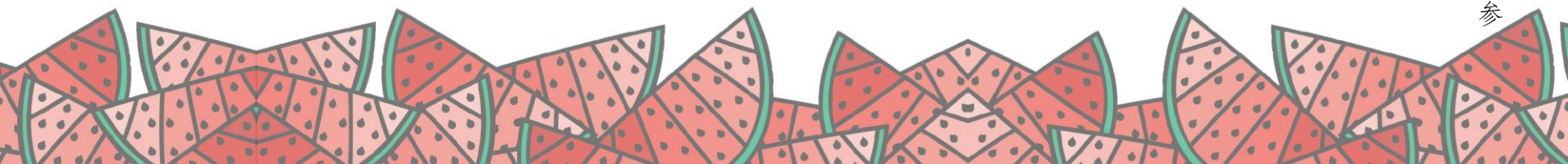
清华大学出版社

本章课件致谢..

魏秀参
周旺

本课件版权所有©LAMD, 为本书教学目的可免费使用,

其他目的需征得本书作者同意



第九章：聚类

大纲

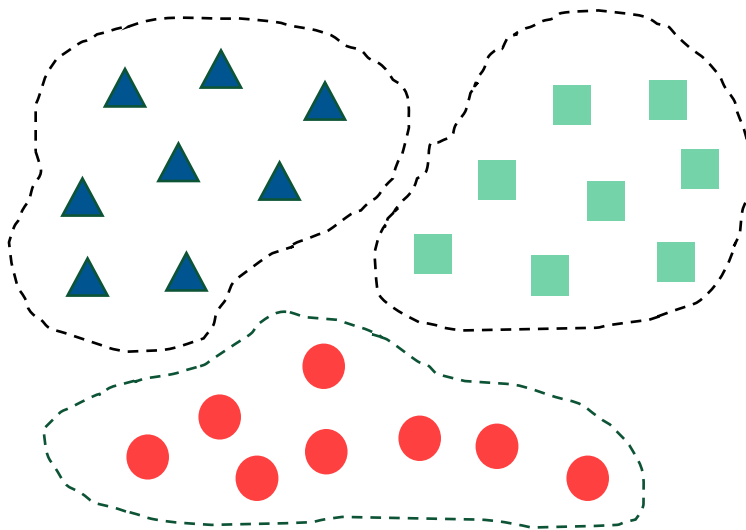
- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

大纲

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

聚类任务

- 在“无监督学习”任务中研究最多、应用最广。
- 聚类目标：将数据集中的样本划分为若干个通常不相交的子集（“簇”， cluster）。
- 聚类既可以作为一个单独过程（用于找寻数据内在的分布结构），也可作为分类等其他学习任务的前驱过程。



聚类任务

□ 形式化描述

假定样本集 $D = \{x_1, x_2, \dots, x_m\}$ 包含 m 个无标记样本, 每个样本 $x_i = (x_{i1}; x_{i2}; \dots; x_{in})$ 是一个 n 维的特征向量, 聚类算法将样本集 D 划分成 k 个不相交的簇 $\{C_l | l = 1, 2, \dots, k\}$ 。其中 $C_{l'} \cap_{l' \neq l} C_l = \phi$, 且 $D = \bigcup_{l=1}^k C_l$ 。

相应地, 用 $\lambda_j \in \{1, 2, \dots, k\}$ 表示样本 x_j 的“簇标记”(即 **cluster label**), 即 $x_j \in C_{\lambda_j}$ 。于是, 聚类的结果可用包含 m 个元素的簇标记向量 $\lambda = \{\lambda_1; \lambda_2; \dots; \lambda_m\}$ 表示。

大纲

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

性能度量

- 聚类性能度量，亦称为聚类“有效性指标” (validity index)
- 直观来讲：

我们希望“物以类聚”，即同一簇的样本尽可能彼此相似，不同簇的样本尽可能不同。换言之，聚类结果的“簇内相似度” (intra-cluster similarity) 高，且“簇间相似度” (inter-cluster similarity) 低，这样的聚类效果较好。

性能度量

□ 聚类性能度量：

- 外部指标 (external index)

将聚类结果与某个“参考模型” (reference model) 进行比较。

- 内部指标 (internal index)

直接考察聚类结果而不用任何参考模型。

性能度量

对数据集 $D = \{x_1, x_2, \dots, x_m\}$, 假定通过聚类得到的簇划分为 $C = \{C_1, C_2, \dots, C_k\}$, 参考模型给出的簇划分为 $C^* = \{C_1^*, C_2^*, \dots, C_s^*\}$. 相应地, 令 λ 与 λ^* 分别表示与 C 和 C^* 对应的簇标记向量.

我们将样本两两配对考虑, 定义

$$a = |SS|, SS = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

$$b = |SD|, SD = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

$$c = |DS|, DS = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

$$d = |DD|, DD = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

性能度量 - 外部指标

□ Jaccard系数 (Jaccard Coefficient, JC)

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

□ FM指数 (Fowlkes and Mallows Index, FMI)

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

□ Rand指数 (Rand Index, RI)

$$RI = \frac{2(a+b)}{m(m-1)}$$

[0,1]区间内,
越大越好.

性能度量 - 内部指标

- 考虑聚类结果的簇划分 $C = \{C_1, C_2, \dots, C_k\}$, 定义簇 C 内样本间的平均距离

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

簇 C 内样本间的最远距离

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

簇 C_i 与簇 C_j 最近样本间的距离

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

簇 C_i 与簇 C_j 中心点间的距离

$$d_{cen}(C) = dist(\mu_i, \mu_j)$$

性能度量 - 内部指标

□ DB指数 (Davies-Bouldin Index, DBI)

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{avg(C_i) + avg(C_j)}{d_{cen}(\mu_i, \mu_j)} \right)$$

越小越好.

□ Dunn指数 (Dunn Index, DI)

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

越大越好.

大纲

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

距离计算

□ 距离度量的性质：

非负性： $dist(x_i, x_j) \geq 0$

同一性： $dist(x_i, x_j) = 0$ 当且仅当 $x_i = x_j$

对称性： $dist(x_i, x_j) = dist(x_j, x_i)$

直递性： $dist(x_i, x_j) \leq dist(x_i, x_k) + dist(x_k, x_j)$

距离计算

□ 距离度量的性质：

非负性： $dist(x_i, x_j) \geq 0$

同一性： $dist(x_i, x_j) = 0$ 当且仅当 $x_i = x_j$

对称性： $dist(x_i, x_j) = dist(x_j, x_i)$

直递性： $dist(x_i, x_j) \leq dist(x_i, x_k) + dist(x_k, x_j)$

□ 常用距离：

闵可夫斯基距离 (Minkowski distance) :

$$dist(x_i, x_j) = \left(\sum_{u=1}^n |x_{iu} - x_{ju}|^p \right)^{\frac{1}{p}}$$

p=2: 欧氏距离 (Euclidean distance) .

p=1: 曼哈顿距离 (Manhattan distance) .

距离计算

□ 属性介绍

- 连续属性 (continuous attribute)
在定义域上有无穷多个可能的取值
- 离散属性 (categorical attribute)
在定义域上是有限个可能的取值

距离计算

□ 属性介绍

- 连续属性 (continuous attribute)

在定义域上有无穷多个可能的取值

- 离散属性 (categorical attribute)

在定义域上是有限个可能的取值

- 有序属性 (ordinal attribute)

例如定义域为 $\{1,2,3\}$ 的离散属性，“1”与“2”比较接近、与“3”比较远，称为“有序属性”。

- 无序属性 (non-ordinal attribute)

例如定义域为{飞机，火车，轮船}这样的离散属性，不能直接在属性值上进行计算，称为“无序属性”。

距离度量

□ Value Difference Metric, VDM (处理无序属性) :

令 $m_{u,a}$ 表示属性 u 上取值为 a 的样本数, $m_{u,a,i}$ 表示在第 i 个样本簇中在属性 u 上取值为 a 的样本数, k 为样本数, 则属性 u 上两个离散值 a 与 b 之间的VDM距离为

$$VDM_p(a, b) = \sum_{i=1}^k \left| \frac{m_{u,a,i}}{m_{u,a}} - \frac{m_{u,b,i}}{m_{u,b}} \right|^p$$

距离度量

□ MinkovDM_p (处理混合属性) :

$$MinkovDM_p(x_i, x_j) = \left(\sum_{u=1}^{n_c} |x_{iu} - x_{ju}|^p + \sum_{u=n_c+1}^n VDM_p(x_{iu}, x_{ju}) \right)^{\frac{1}{p}}$$

□ 加权距离 (样本中不同属性的重要性不同时) :

$$dist(x_i, x_j) = (\omega_1 \cdot |x_{i1} - x_{j1}|^p + \cdots + \omega_n \cdot |x_{in} - x_{jn}|^p)^{\frac{1}{p}}$$

$$\omega_i \geq 0, \sum_{i=1}^n \omega_i = 1$$

大纲

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

原型聚类

□ 原型聚类

也称为“基于原型的聚类” (prototype-based clustering), 此类算法假设聚类结构能通过一组原型刻画。

□ 算法过程:

通常情况下, 算法先对原型进行初始化, 再对原型进行迭代更新求解。

□ 接下来, 介绍几种著名的原型聚类算法

k 均值算法、学习向量量化算法、高斯混合聚类算法。

原型聚类 - k 均值算法

给定数据集 $D = \{x_1, x_2, \dots, x_m\}$, k 均值算法针对聚类所得簇划分 $C = \{C_1, C_2, \dots, C_k\}$ 最小化平方误差

$$E = \sum_{i=1}^k \sum_{x \in C_j} \|x - \mu_i\|_2^2$$

其中, μ_i 是簇 C_i 的均值向量。

E 值在一定程度上刻画了簇内样本围绕簇均值向量的紧密程度, E 值越小, 则簇内样本相似度越高。

原型聚类 - k 均值算法

给定数据集 $D = \{x_1, x_2, \dots, x_m\}$, k 均值算法针对聚类所得簇划分 $C = \{C_1, C_2, \dots, C_k\}$ 最小化平方误差

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

其中, μ_i 是簇 C_i 的均值向量。

E 值在一定程度上刻画了簇内样本围绕簇均值向量的紧密程度, E 值越小, 则簇内样本相似度越高。

□ 算法流程 (迭代优化) :

初始化每个簇的均值向量

repeat

1. (更新) 簇划分;
2. 计算每个簇的均值向量

until 当前均值向量均未更新

原型聚类 - k 均值算法

□ 算法伪代码:

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

过程:

- 1: 从 D 中随机选择 k 个样本作为初始均值向量 $\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k\}$
- 2: **repeat**
- 3: 令 $C_i = \emptyset$ ($1 \leq i \leq k$)
- 4: **for** $j = 1, \dots, m$ **do**
- 5: 计算样本 $\mathbf{x}_j$ 与各均值向量 $\boldsymbol{\mu}_i$ ($1 \leq i \leq k$) 的距离: $d_{ji} = \|\mathbf{x}_j - \boldsymbol{\mu}_i\|_2$;
- 6: 根据距离最近的均值向量确定 $\mathbf{x}_j$ 的簇标记: $\lambda_j = \arg \min_{i \in \{1, 2, \dots, k\}} d_{ji}$;
- 7: 将样本 $\mathbf{x}_j$ 划入相应的簇: $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$;
- 8: **end for**
- 9: **for** $i = 1, \dots, k$ **do**
- 10: 计算新均值向量: $\boldsymbol{\mu}'_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$;
- 11: **if** $\boldsymbol{\mu}'_i \neq \boldsymbol{\mu}_i$ **then**
- 12: 将当前均值向量 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}'_i$
- 13: **else**
- 14: 保持当前均值向量不变
- 15: **end if**
- 16: **end for**
- 17: **until** 当前均值向量均未更新
- 18: **return** 簇划分结果

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

原型聚类 - k 均值算法

□ k 均值算法实例

接下来以表9-1的西瓜数据集4.0为例，来演示 k 均值算法的学习过程。将编号为 i 的样本称为 x_i 。

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

原型聚类 - k 均值算法

□ k 均值算法实例

假定聚类簇数 $k=3$ ，算法开始时，随机选择3个样本 x_6, x_{12}, x_{27} 作为初始均值向量，即 $\mu_1 = (0.403; 0.237), \mu_2 = (0.343; 0.099), \mu_3 = (0.533; 0.472)$

考察样本 $x_1 = (0.697; 0.460)$ ，它与当前均值向量 μ_1, μ_2, μ_3 的距离分别为0.369, 0.506, 0.166，因此 x_1 将被划入簇 C_3 中。类似的，对数据集中的所有样本考察一遍后，可得当前簇划分为

$$C_1 = \{x_5, x_6, x_7, x_8, x_9, x_{10}, x_{13}, x_{14}, x_{15}, x_{17}, x_{18}, x_{19}, x_{20}, x_{23}\}$$

$$C_2 = \{x_{11}, x_{12}, x_{16}\}$$

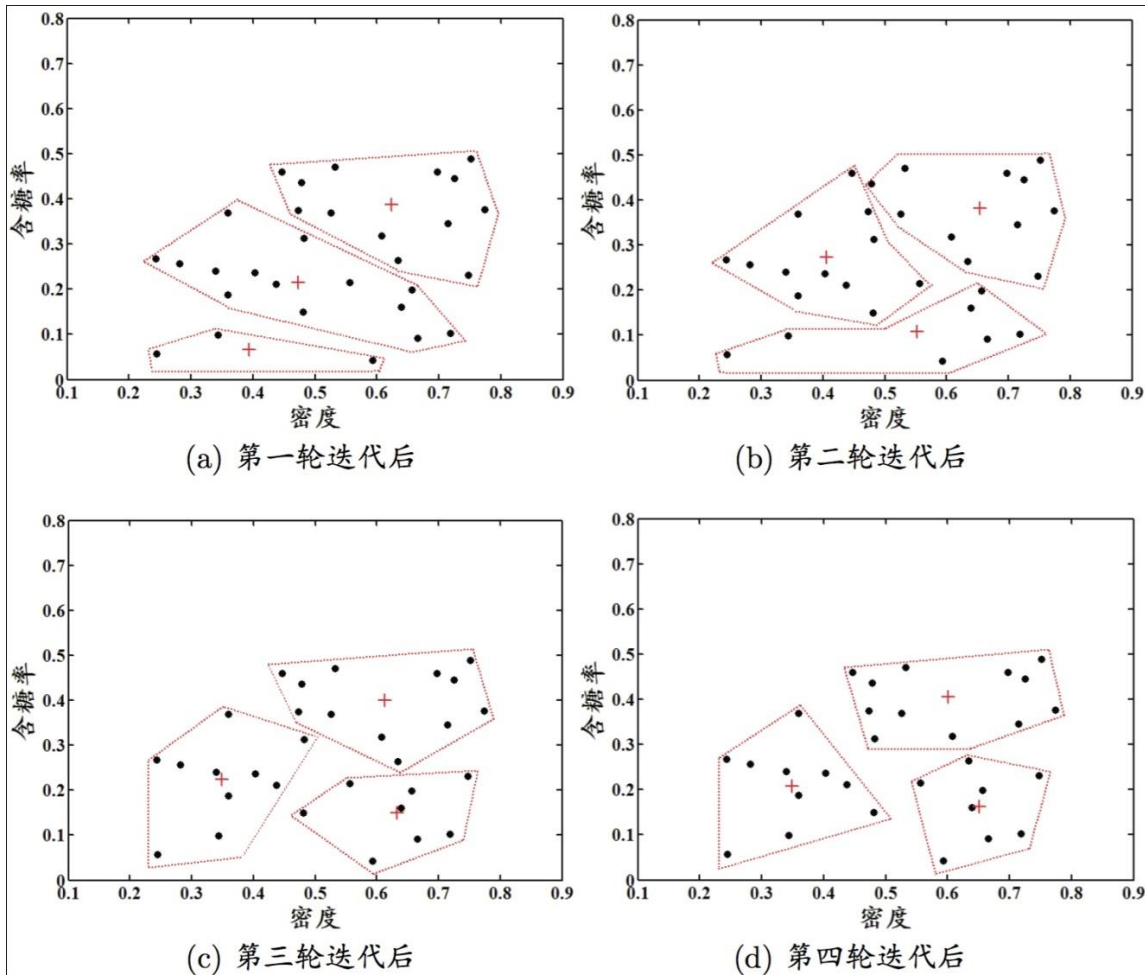
$$C_3 = \{x_1, x_2, x_3, x_4, x_{21}, x_{22}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}\}$$

于是，可以从分别求得新的均值向量

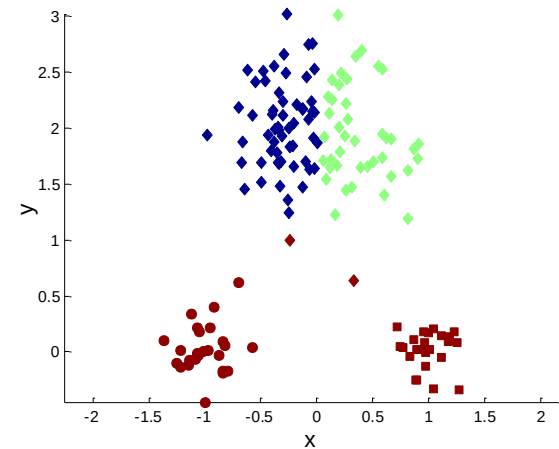
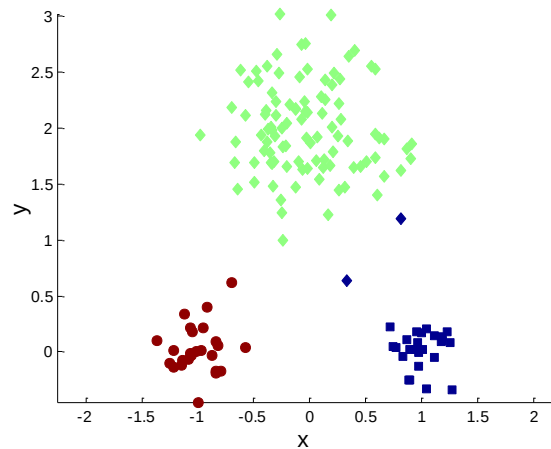
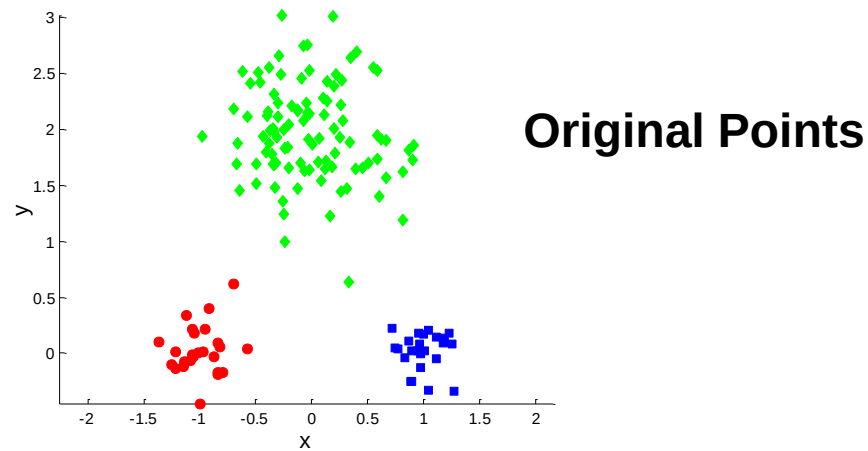
$\mu'_1 = (0.473; 0.214), \mu'_2 = (0.394; 0.066), \mu'_3 = (0.623; 0.388)$
不断重复上述过程，如下图所示。

原型聚类 - k 均值算法

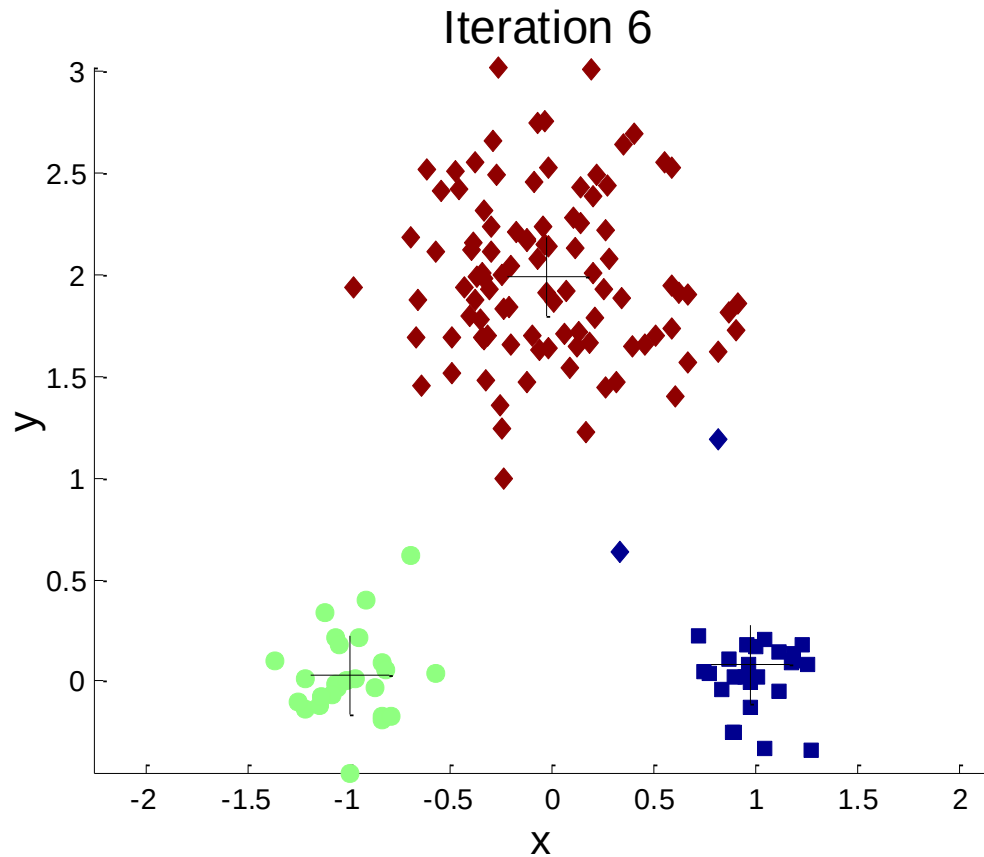
□ 聚类结果：



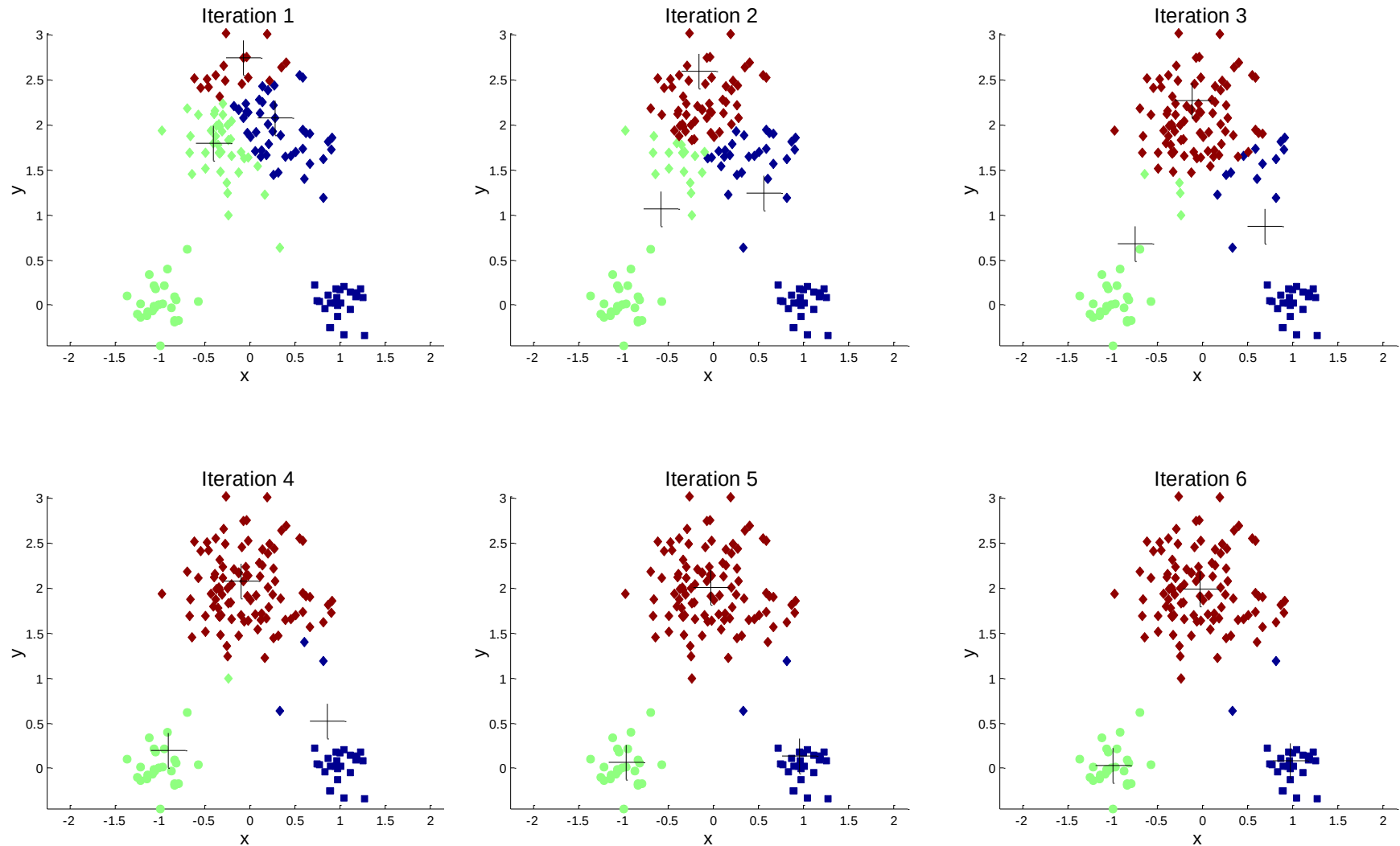
Two different K-means Clusterings



Importance of Choosing Initial Centroids



Importance of Choosing Initial Centroids



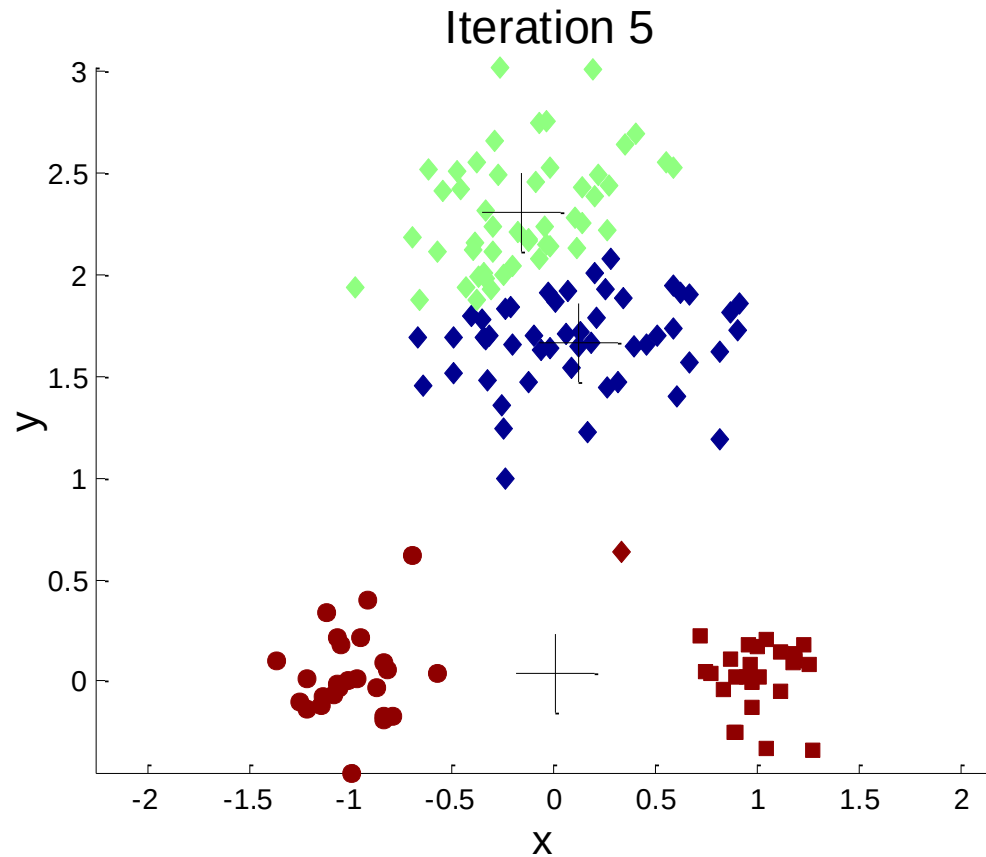
Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
 - For each point, the error is the distance to the nearest cluster
 - To get SSE, we square these errors and sum them.

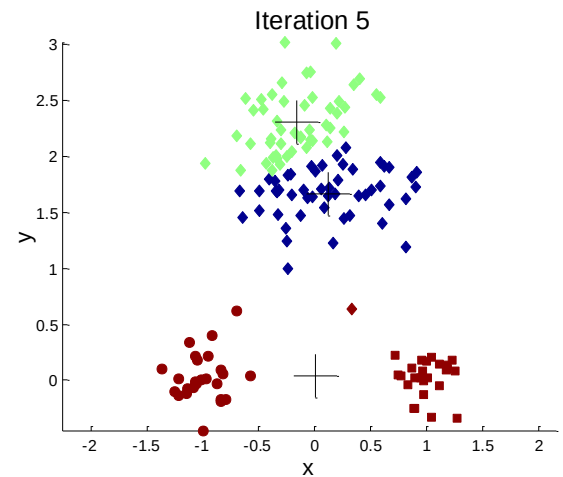
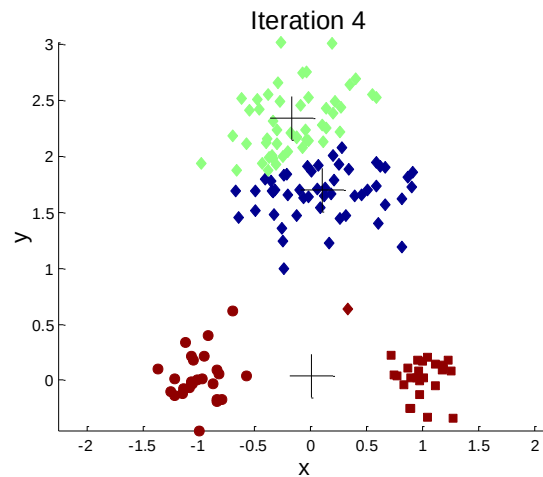
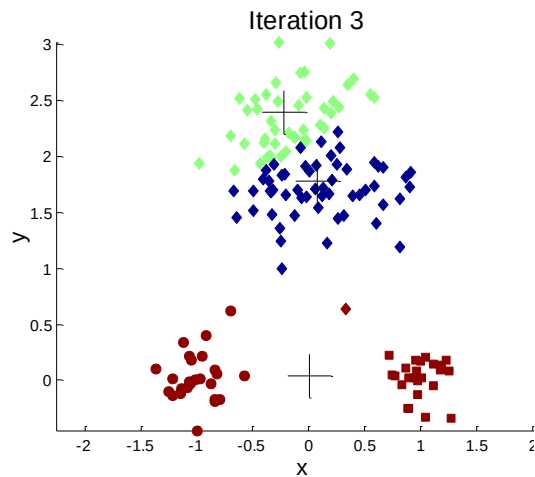
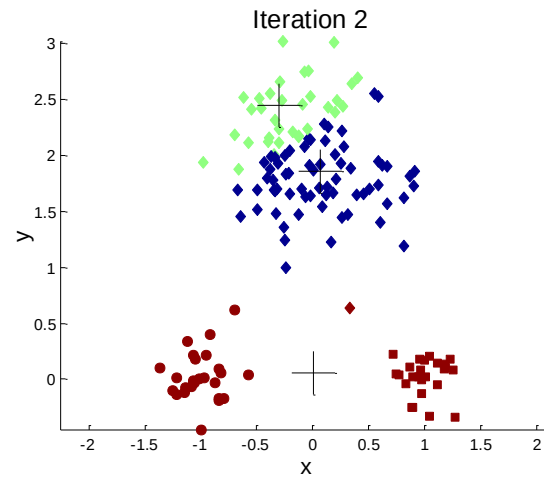
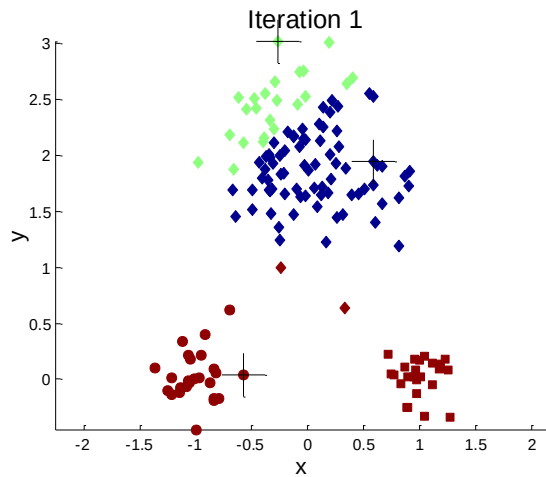
$$SSE = \sum_{i=1}^K \sum_{x \in C_i} \text{dist}^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
 - can show that m_i corresponds to the center (mean) of the cluster
- Given two clusterings, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K , the number of clusters
 - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Importance of Choosing Initial Centroids ...



Importance of Choosing Initial Centroids ...



Problems with Selecting Initial Points

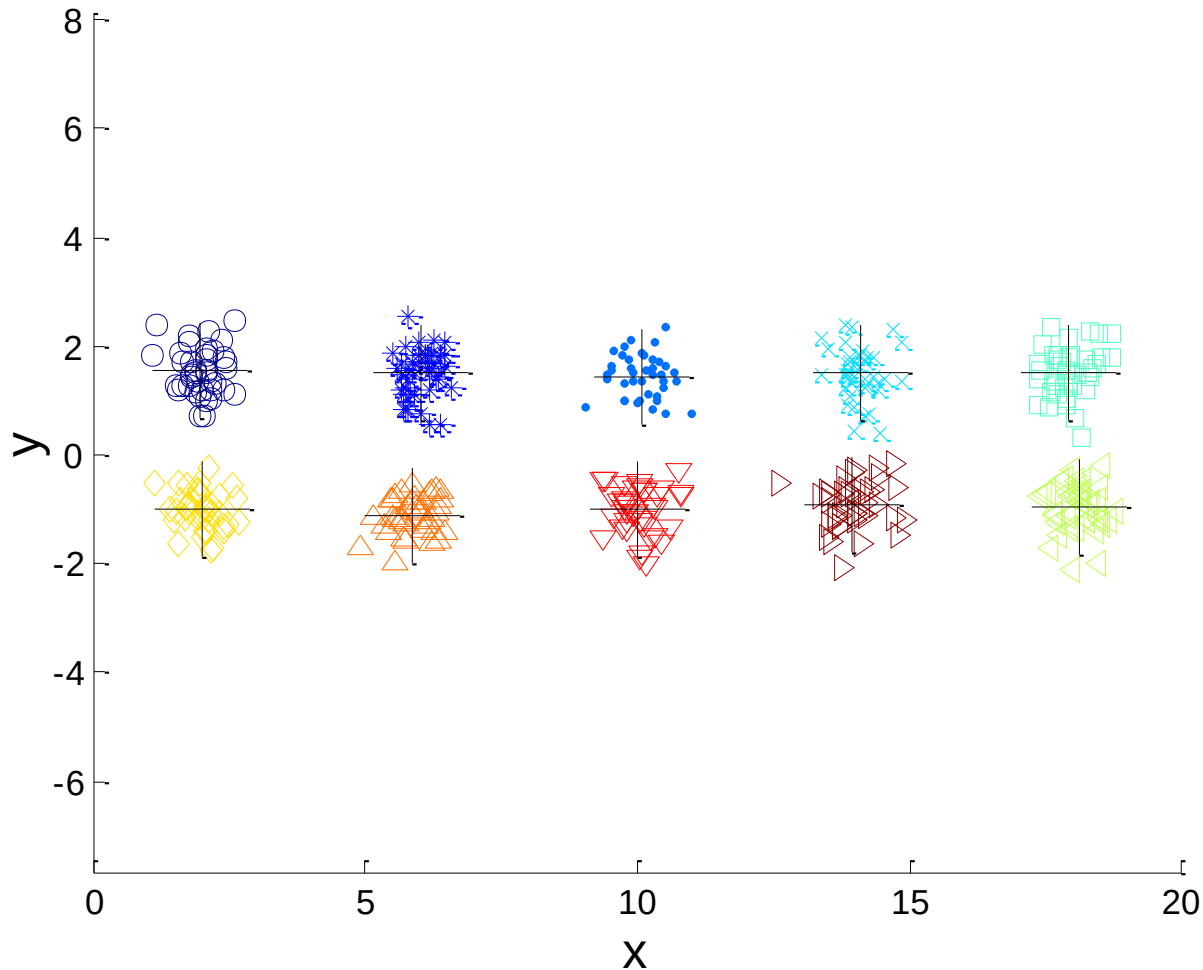
- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
 - Chance is relatively small when K is large
 - If clusters are the same size, n , then

- For example, if $K = 10$, then probability = $10!/10^{10} = 0.00036$
- Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
- Consider an example of five pairs of clusters

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

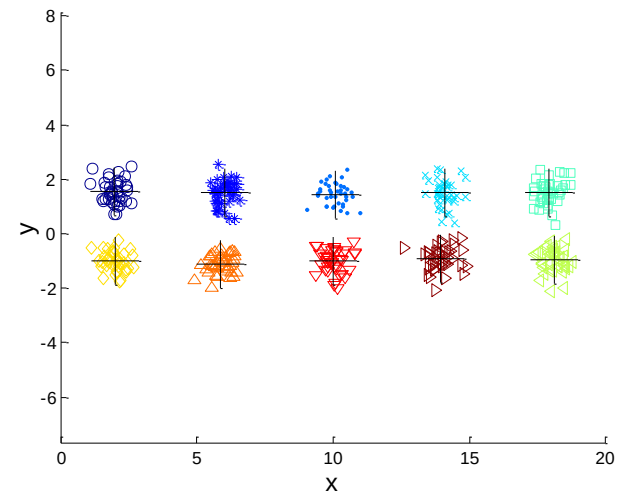
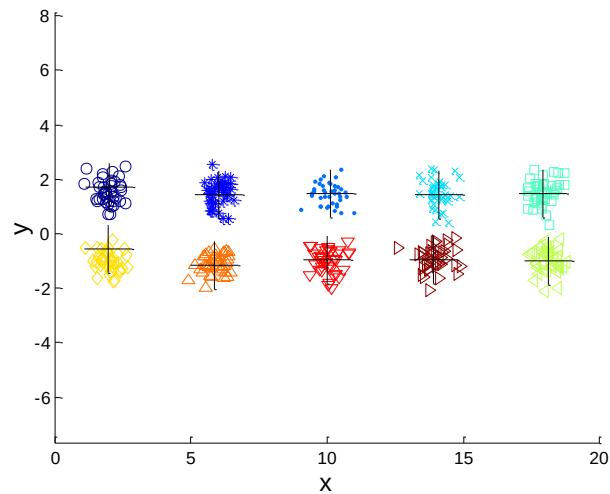
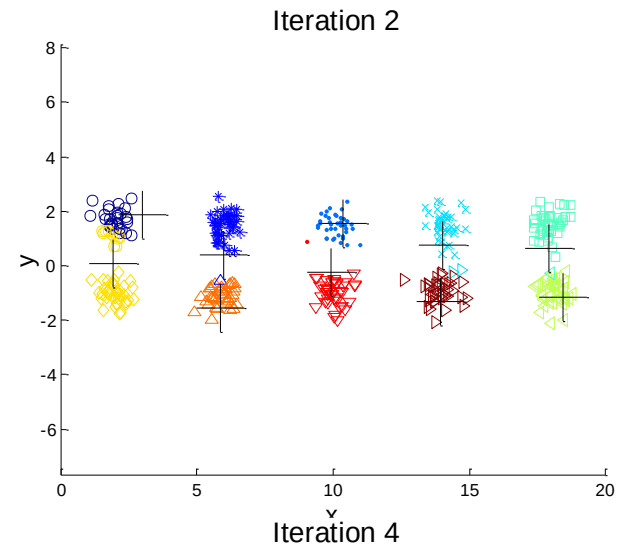
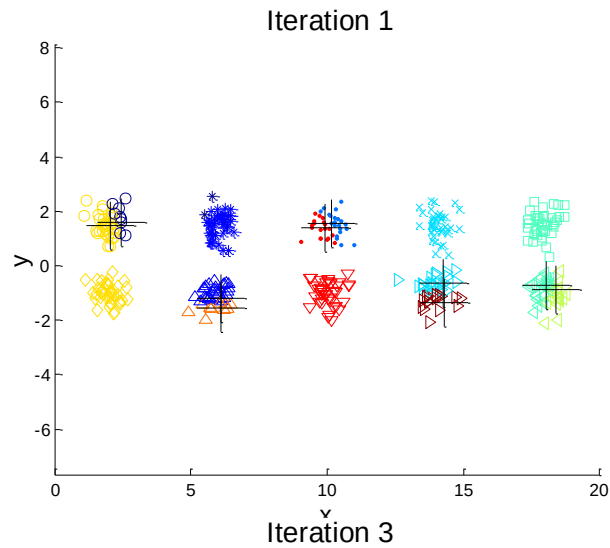
10 Clusters Example

Iteration 4



Starting with two initial centroids in one cluster of each pair of clusters

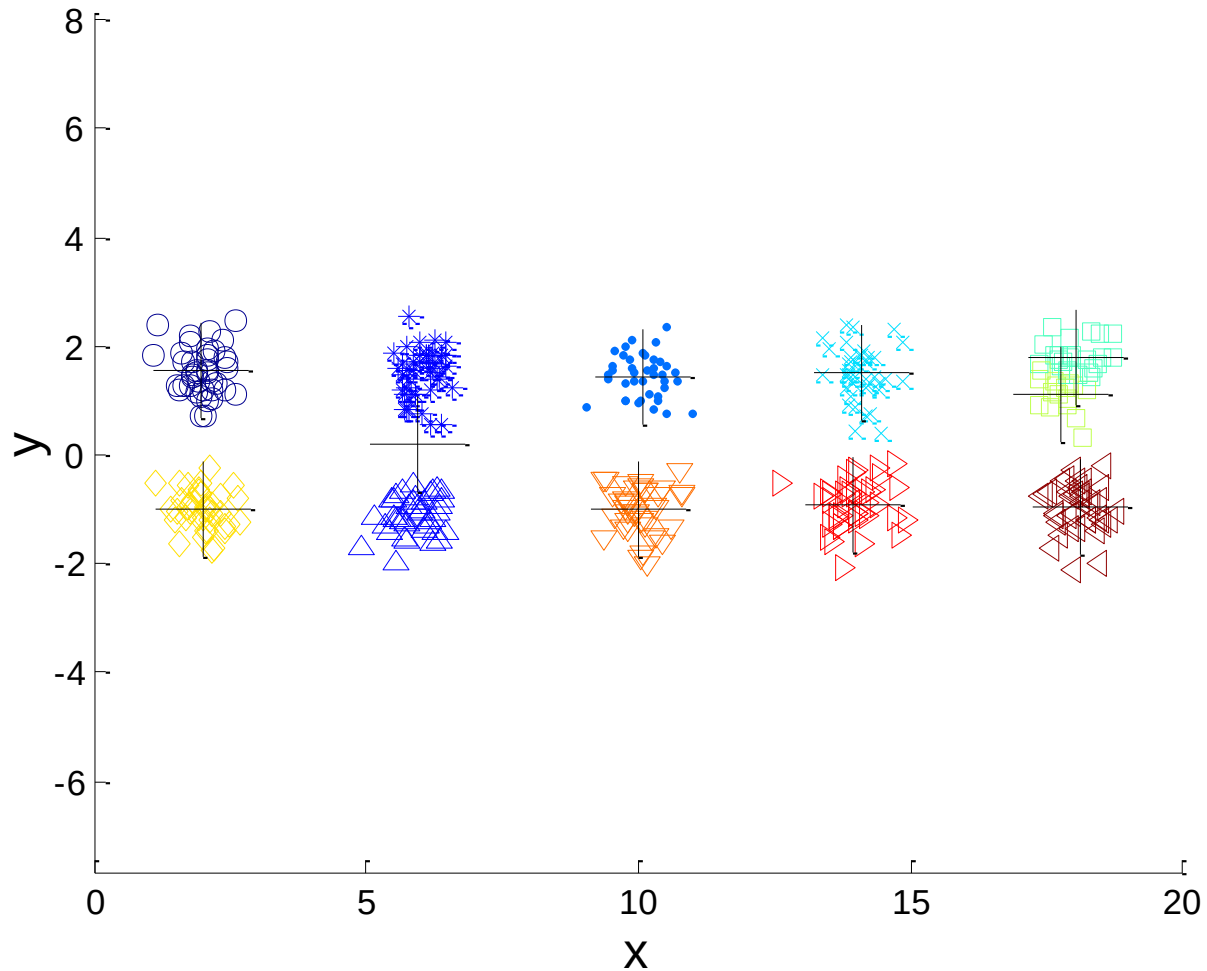
10 Clusters Example



Starting with two initial centroids in one cluster of each pair of clusters

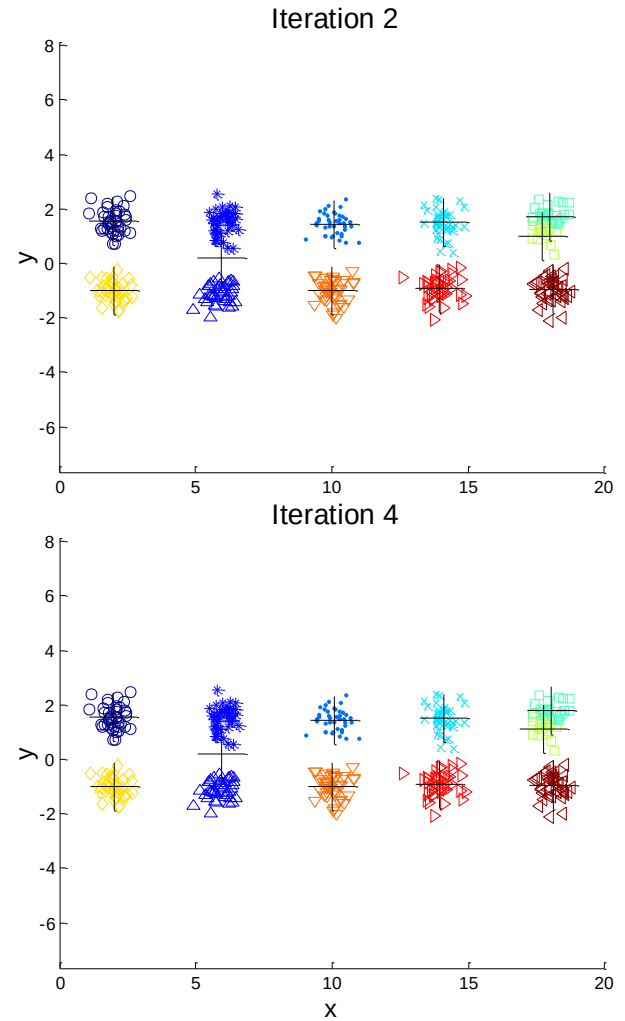
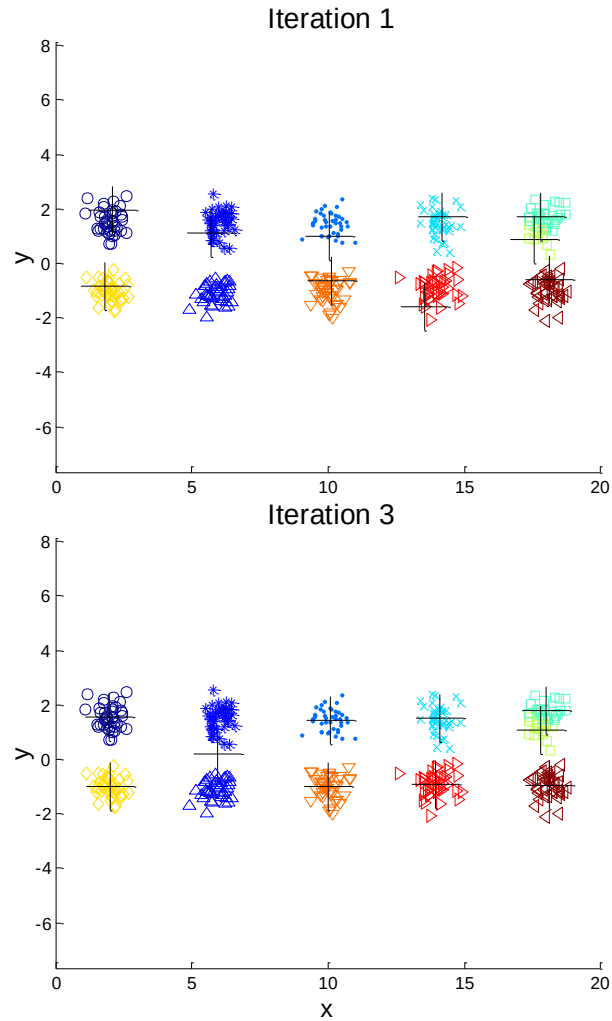
10 Clusters Example

Iteration 4



Starting with some pairs of clusters having three initial centroids, while other have only one.

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

原型聚类 - 学习向量量化

□ 学习向量量化 (Learning Vector Quantization, LVQ)

与一般聚类算法不同的是, LVQ假设数据样本带有类别标记, 学习过程中利用样本的这些监督信息来辅助聚类.

给定样本集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, LVQ的目标是学得一组 n 维原型向量 $\{p_1, p_2, \dots, p_q\}$, 每个原型向量代表一个聚类簇。

原型聚类 - 学习向量量化

□ 算法伪代码:

输入: 样本集 $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$;
原型向量个数 q , 各原型向量预设的类别标记 $\{t_1, t_2, \dots, t_q\}$;
学习率 $\eta \in (0, 1)$.

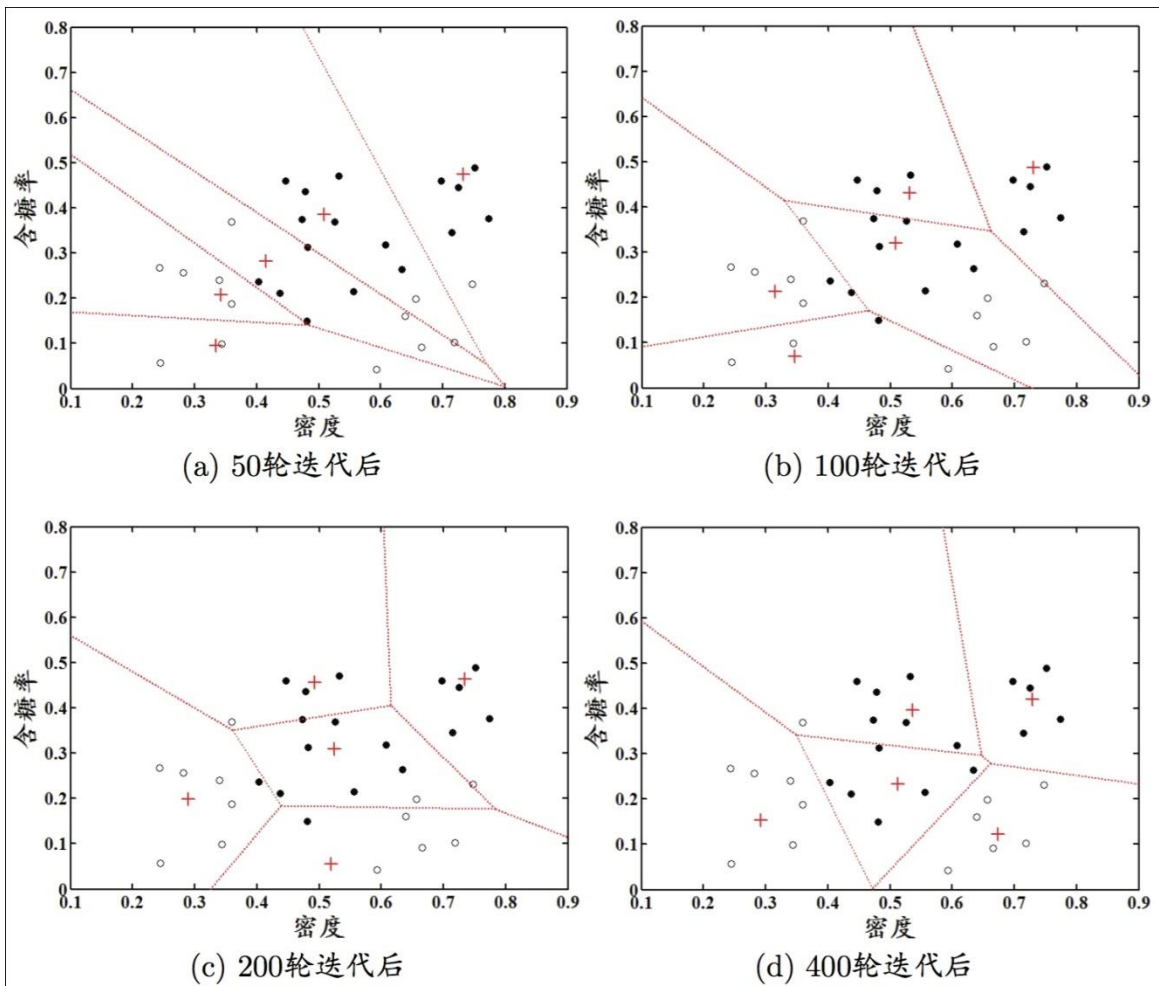
过程:

- 1: 初始化一组原型向量 $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$
- 2: **repeat**
- 3: 从样本集 D 随机选取样本 $(\mathbf{x}_j, y_j)$;
- 4: 计算样本 $\mathbf{x}_j$ 与 $\mathbf{p}_i$ ($1 \leq i \leq q$) 的距离: $d_{ji} = \|\mathbf{x}_j - \mathbf{p}_i\|_2$;
- 5: 找出与 $\mathbf{x}_j$ 距离最近的原型向量; $i^* = \arg \min_{i \in \{1, 2, \dots, q\}} d_{ji}$;
- 6: **if** $y_j = t_{i^*}$ **then**
- 7: $\mathbf{p}' = \mathbf{p}_{i^*} + \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*})$
- 8: **else**
- 9: $\mathbf{p}' = \mathbf{p}_{i^*} - \eta \cdot (\mathbf{x}_j - \mathbf{p}_{i^*})$
- 10: **end if**
- 11: 将原型向量 $\mathbf{p}_{i^*}$ 更新为 $\mathbf{p}'$
- 12: **until** 满足停止条件
- 13: **return** 当前原型向量

输出: 原型向量 $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_q\}$

原型聚类 - 学习向量量化

□ 聚类效果:



原型聚类 - 高斯混合聚类

与 k 均值、LVQ用原型向量来刻画聚类结构不同，高斯混合聚类 (Mixture-of-Gaussian) 采用概率模型来表达聚类原型：

□ 多元高斯分布的定义

对 n 维样本空间中的随机向量 x ，若 x 服从高斯分布，其概率密度函数为

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

其中 μ 是 n 维均值向量， Σ 是 $n \times n$ 的协方差矩阵。也可将概率密度函数记作 $p(x|\mu, \Sigma)$ 。

原型聚类 - 高斯混合聚类

□ 高斯混合分布的定义

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

该分布由 k 个混合分布组成，每个分布对应一个高斯分布。其中， μ_i 与 Σ_i 是第 i 个高斯混合成分的参数。而 $\alpha_i > 0$ 为相应的“混合系数”， $\sum_{i=1}^k \alpha_i = 1$

原型聚类 - 高斯混合聚类

□ 假设样本的生成过程由高斯混合分布给出：

首先，根据 $\alpha_1, \alpha_2, \dots, \alpha_k$ 定义的先验分布选择高斯混合成分，其中 α_i 为选择第 i 个混合成分的概率；

然后，根据被选择的混合成分的概率密度函数进行采样，从而生成相应的样本。

原型聚类 - 高斯混合聚类

□ 模型求解：最大化（对数）似然

$$\begin{aligned} LL(D) &= \ln \left(\prod_{j=1}^m p_M(x_j) \right) \\ &= \sum_{j=1}^m \ln \left(\sum_{i=1}^k \alpha_i \cdot p(x_j | \mu_i, \Sigma_i) \right) \end{aligned}$$

令：

$$\frac{\partial LL(D)}{\partial \mu_i} = 0 \quad \longrightarrow \quad \mu_i = \frac{\sum_{j=1}^m \gamma_{ji} x_j}{\sum_{j=1}^m \gamma_{ji}}$$

高斯混合聚类 - 模型求解 (续)

令：

$$\frac{\partial LL(D)}{\partial \Sigma_i} = 0 \quad \longrightarrow \quad \Sigma_i = \frac{\sum_{j=1}^m \gamma_{ji} (x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^m \gamma_{ji}}$$

高斯混合聚类

□ 算法伪代码:

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

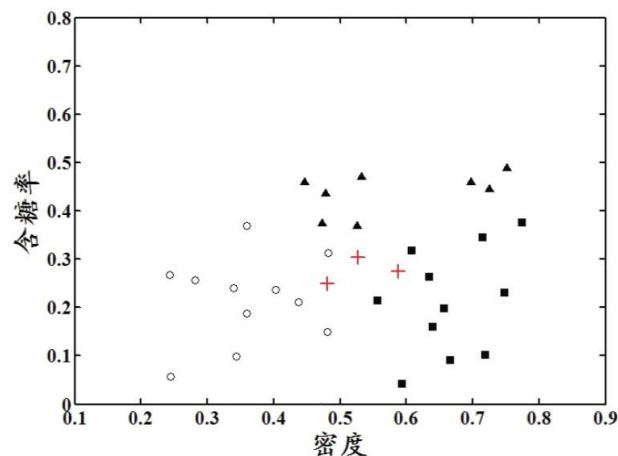
过程:

```
1: 初始化高斯混合分布的模型参数  $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$ 
2: repeat
3:   for  $j = 1, \dots, m$  do
4:     根据(9.30)计算  $\mathbf{x}_j$  由各混合成分生成的后验概率, 即
        $\gamma_{ji} = p_{\mathcal{M}}(z_j = i \mid \mathbf{x}_j) \ (1 \leq i \leq k)$ 
5:   end for
6:   for  $i = 1, \dots, k$  do
7:     计算新均值向量:  $\boldsymbol{\mu}'_i = \frac{\sum_{j=1}^m \gamma_{ji} \mathbf{x}_j}{\sum_{j=1}^m \gamma_{ji}}$ ;
8:     计算新协方差矩阵:  $\boldsymbol{\Sigma}'_i = \frac{\sum_{j=1}^m \gamma_{ji} (\mathbf{x}_j - \boldsymbol{\mu}'_i)(\mathbf{x}_j - \boldsymbol{\mu}'_i)^\top}{\sum_{j=1}^m \gamma_{ji}}$ ;
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  $\{(\alpha_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \mid 1 \leq i \leq k\}$  更新为  $\{(\alpha'_i, \boldsymbol{\mu}'_i, \boldsymbol{\Sigma}'_i) \mid 1 \leq i \leq k\}$ 
12: until 满足停止条件
13:  $C_i = \emptyset \ (1 \leq i \leq k)$ 
14: for  $j = 1, \dots, m$  do
15:   根据(9.31)确定  $\mathbf{x}_j$  的簇标记  $\lambda_j$ ;
16:   将  $\mathbf{x}_j$  划入相应的簇:  $C_{\lambda_j} = C_{\lambda_j} \cup \{\mathbf{x}_j\}$ 
17: end for
18: return 簇划分结果
```

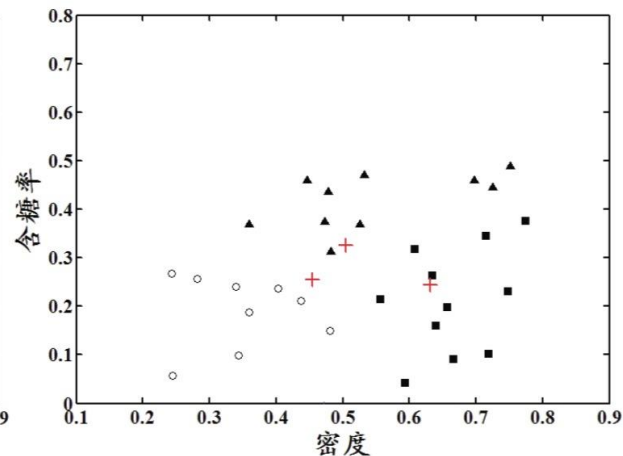
输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

高斯混合聚类

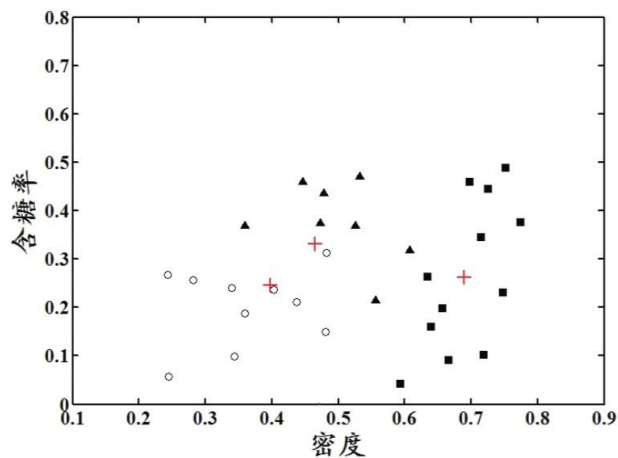
□ 聚类效果：



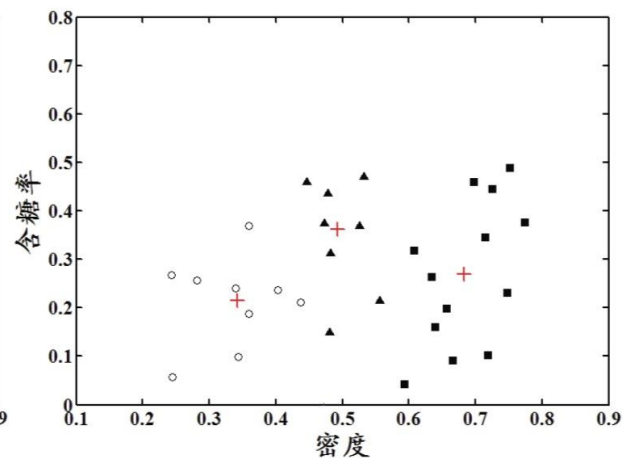
(a) 5轮迭代后



(b) 10轮迭代后



(c) 20轮迭代后



(d) 50轮迭代后

大纲

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

密度聚类

□ 密度聚类的定义

密度聚类也称为“基于密度的聚类” (density-based clustering)。

此类算法假设聚类结构能通过样本分布的紧密程度来确定。

通常情况下，密度聚类算法从样本密度的角度来考察样本之间的可连接性，并基于可连接样本不断扩展聚类簇来获得最终的聚类结果。

接下来介绍**DBSCAN**这一密度聚类算法。

密度聚类

□ DBSCAN算法：基于一组“邻域”参数 $(\epsilon, MinPts)$ 来刻画样本分布的紧密程度。

□ 基本概念：

- ϵ 邻域：对样本 $x_j \in D$ ，其 ϵ 邻域包含样本集 D 中与 x_j 的距离不大于 ϵ 的样本；
- 核心对象：若样本 x_j 的 ϵ 邻域至少包含 $MinPts$ 个样本，则该样本点为一个核心对象；
- 密度直达：若样本 x_j 位于样本 x_i 的 ϵ 邻域中，且 x_i 是一个核心对象，则称样本 x_j 由 x_i 密度直达；
- 密度可达：对样本 x_i 与 x_j ，若存在样本序列 $p_1, p_2, \dots, p_n$ ，其中 $p_1 = x_i, p_n = x_j$ 且 p_{i+1} 由 p_i 密度直达，则该两样本密度可达；
- 密度相连：对样本 x_i 与 x_j ，若存在样本 x_k 使得两样本均由 x_k 密度可达，则称该两样本密度相连。

密度聚类

□ 一个例子

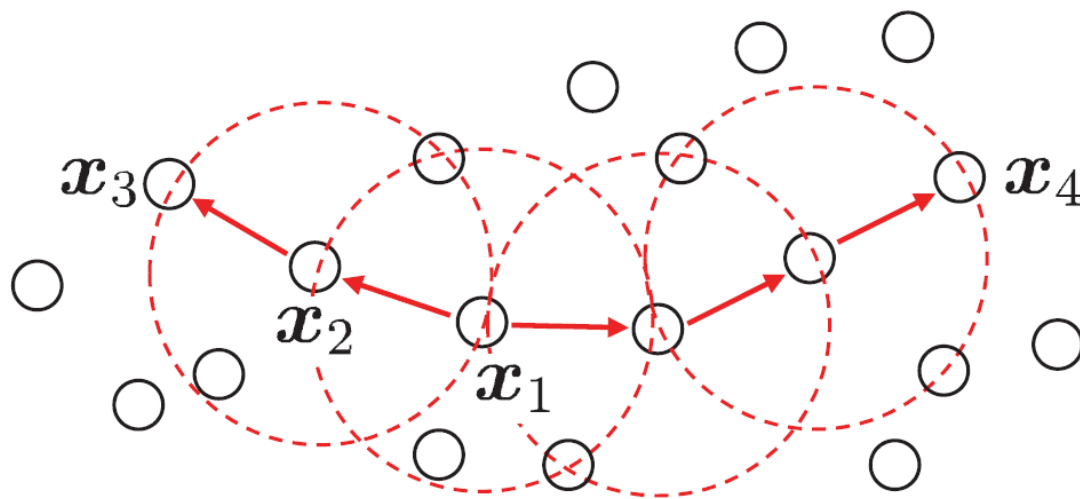
令 $MinPts = 3$, 则
虚线显示出 ϵ 领域。

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_3 与 x_4 密度相连。



密度聚类

□ 对“簇”的定义

由密度可达关系导出的最大密度相连样本集合。

□ 对“簇”的形式化描述

给定领域参数，簇是满足以下性质的非空样本子集：

连接性： $x_i \in C, x_j \in C \Rightarrow x_i$ 与 x_j 密度相连

最大性： $x_i \in C, x_i$ 与 x_j 密度可达 $\Rightarrow x_j \in C$

实际上，若 x 为核心对象，由 x 密度可达的所有样本组成的集合记为 $X = \{x' \in D \mid x' \text{ 由 } x \text{ 密度可达}\}$ ，则 X 为满足连接性与最大性的簇。

密度聚类

□ DBSCAN算法伪代码:

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

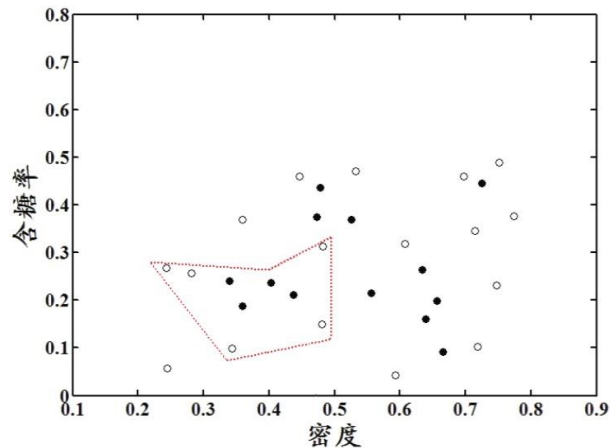
过程:

- 1: 初始化核心对象集合: $\Omega = \emptyset$
- 2: **for** $j = 1, \dots, m$ **do**
- 3: 确定样本 x_j 的 ϵ -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 $Q = \langle o \rangle$;
- 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 Δ 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**
- 25: **return** 簇划分结果

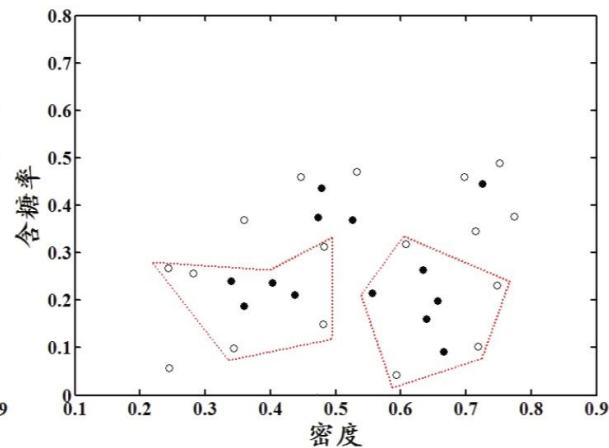
输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

密度聚类

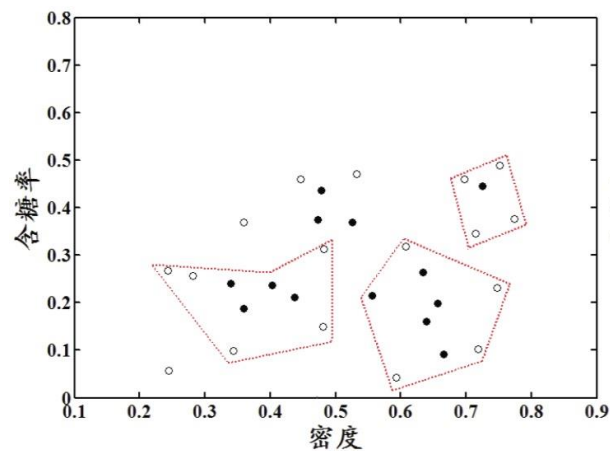
□ 聚类效果：



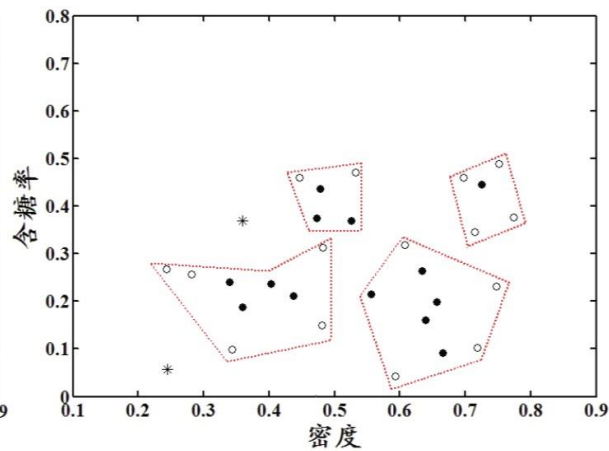
(a) 生成聚类簇 C_1



(b) 生成聚类簇 C_2



(c) 生成聚类簇 C_3



(d) 生成聚类簇 C_4

大纲

- 聚类任务
- 性能度量
- 距离计算
- 原型聚类
- 密度聚类
- 层次聚类

层次聚类

□ 层次聚类试图在不同层次对数据集进行划分，从而形成树形的聚类结构。数据集划分既可采用“自底向上”的聚合策略，也可采用“自顶向下”的分拆策略。

□ **AGNES**算法（自底向上的层次聚类算法）

首先，将样本中的每一个样本看做一个初始聚类簇，然后在算法运行的每一步中找出距离最近的两个聚类簇进行合并，该过程不断重复，直到达到预设的聚类簇的个数。

这里两个聚类簇 C_i 和 C_j 的距离，可以有3种度量方式。

层次聚类

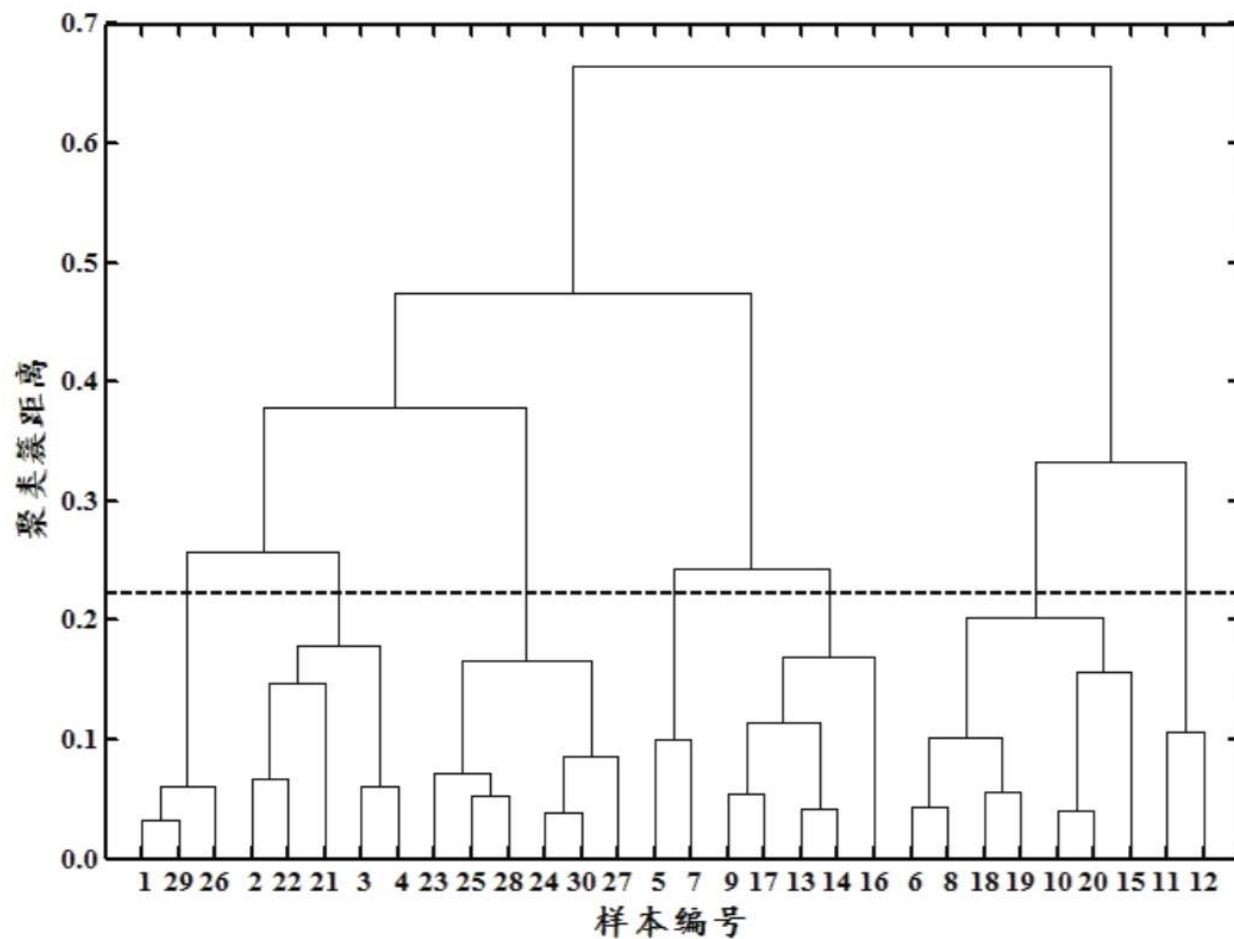
最小距离:
$$d_{min}(C_i, C_j) = \min_{x \in C_i, z \in C_j} dist(x, z)$$

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

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

层次聚类 - 树状图

□ AGNES算法树状图：



层次聚类 – AGNES算法

□ AGNES算法伪代码:

输入: 样本集 $D = \{x_1, x_2, \dots, x_m\}$;
聚类簇距离度量函数 $d \in \{d_{\min}, d_{\max}, d_{\text{avg}}\}$;
聚类簇数 k .

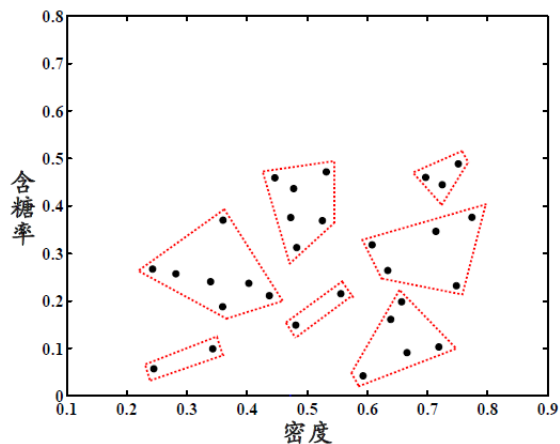
过程:

```
1: for  $j = 1, \dots, m$  do
2:    $C_j = \{x_j\}$ 
3: end for
4: for  $i = 1, \dots, m$  do
5:   for  $j = i, \dots, m$  do
6:      $M(i, j) = d(C_i, C_j)$ ;
7:      $M(j, i) = M(i, j)$ 
8:   end for
9: end for
10: 设置当前聚类簇个数:  $q = m$ 
11: while  $q > k$  do
12:   找出距离最近的两个聚类簇  $(C_{i^*}, C_{j^*})$ ;
13:   合并  $(C_{i^*}, C_{j^*})$ :  $C_{i^*} = C_{i^*} \cup C_{j^*}$ ;
14:   for  $j = j^* + 1, \dots, q$  do
15:     将聚类簇  $C_j$  重编号为  $C_{j-1}$ 
16:   end for
17:   删除距离矩阵  $M$  的第  $j^*$  行与第  $j^*$  列;
18:   for  $j = 1, \dots, q - 1$  do
19:      $M(i^*, j) = d(C_{i^*}, C_j)$ ;
20:      $M(j, i^*) = M(i^*, j)$ 
21:   end for
22:    $q = q - 1$ 
23: end while
24: return 簇划分结果
```

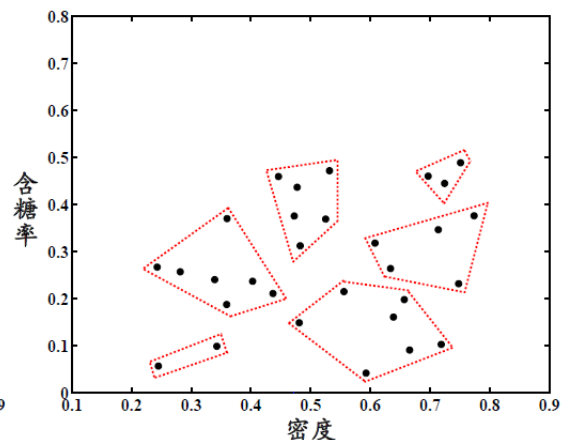
输出: 簇划分 $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$

层次聚类

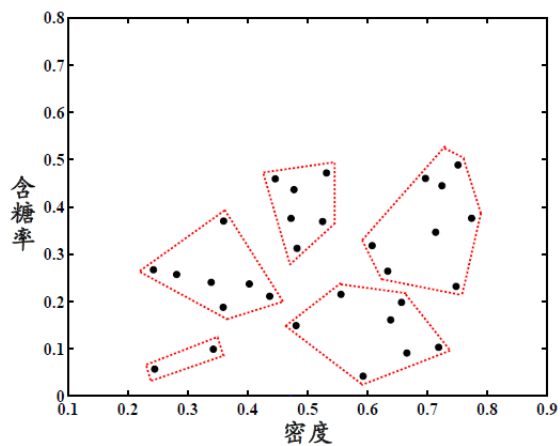
AGNES算法聚类效果：



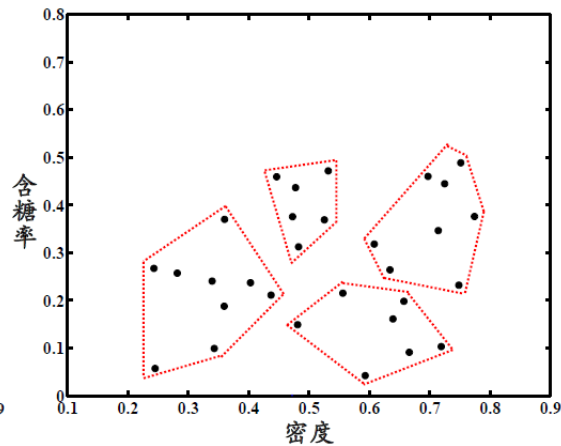
(a) 聚类簇数 $k = 7$



(b) 聚类簇数 $k = 6$



(c) 聚类簇数 $k = 5$



(d) 聚类簇数 $k = 4$