# MACHINE LEARNING

# 机器学习

## Clustering

## 聚类

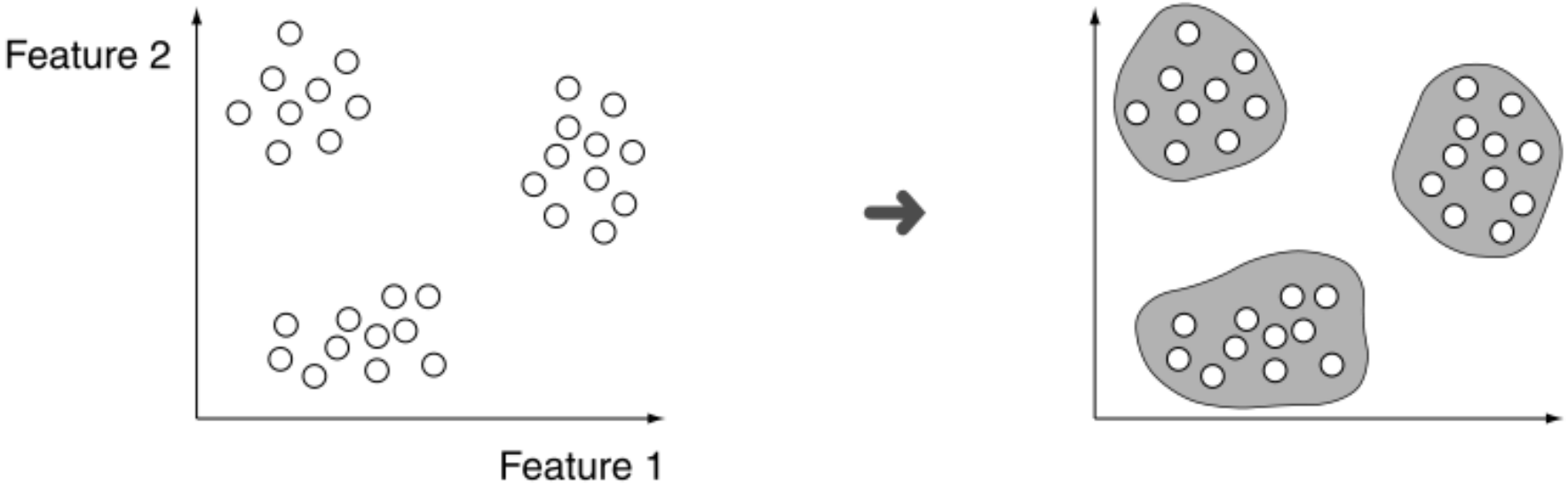

参考：
《机器学习》

Machine Learning Course

# 聚类

- 定义

聚类是一类典型的非监督学习任务，它并没有领域相关的学习目标。聚类任务的学习目标是固定的：希望将务将数据划分成不同的“簇”*cluster*，并且簇应该满足：簇内样本相似，簇间样本相异。



# 原型聚类算法 – K均值

- K-MEANS

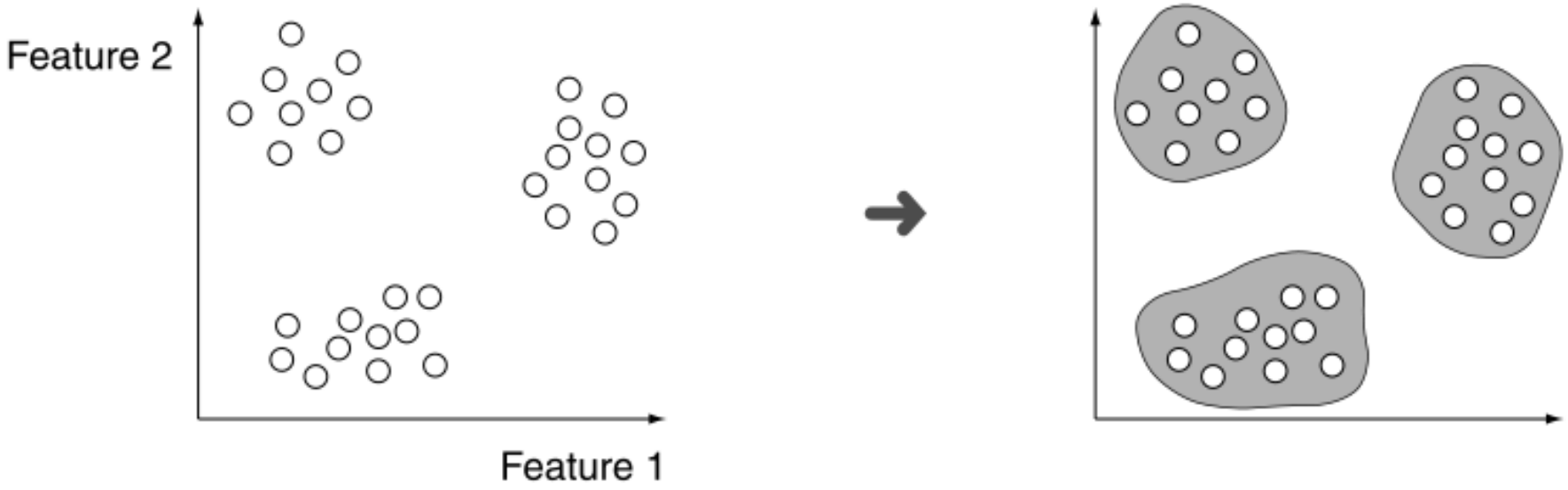
K均值 *K-means* 算法是迄今为止算得上最简单也最基础的聚类算法.

输入： $x_1, \ldots, x_n, x \in \mathbb{R}^d$, 簇个数 $K$

输出: 簇标记向量 $\boldsymbol{c}$

其中$\boldsymbol{c} = (c_1, \ldots, c_n)$, $c_i \in \{1, \ldots, K\}$

$c_i = c_j = k$ 表明 $x_i$ 和 $x_j$ 同属于标记为 $k$ 的簇



# 原型聚类算法 – K均值

- K-MEANS

K均值 *K-means* 算法是迄今为止算得上最简单也最基础的聚类算法.

输入： $x_1, \ldots, x_n, x \in \mathbb{R}^d$, 簇个数 $K$

输出: 簇标记向量 $\boldsymbol{c}$， 以及 $K$ 个簇对应的原型 $\boldsymbol{\mu}$

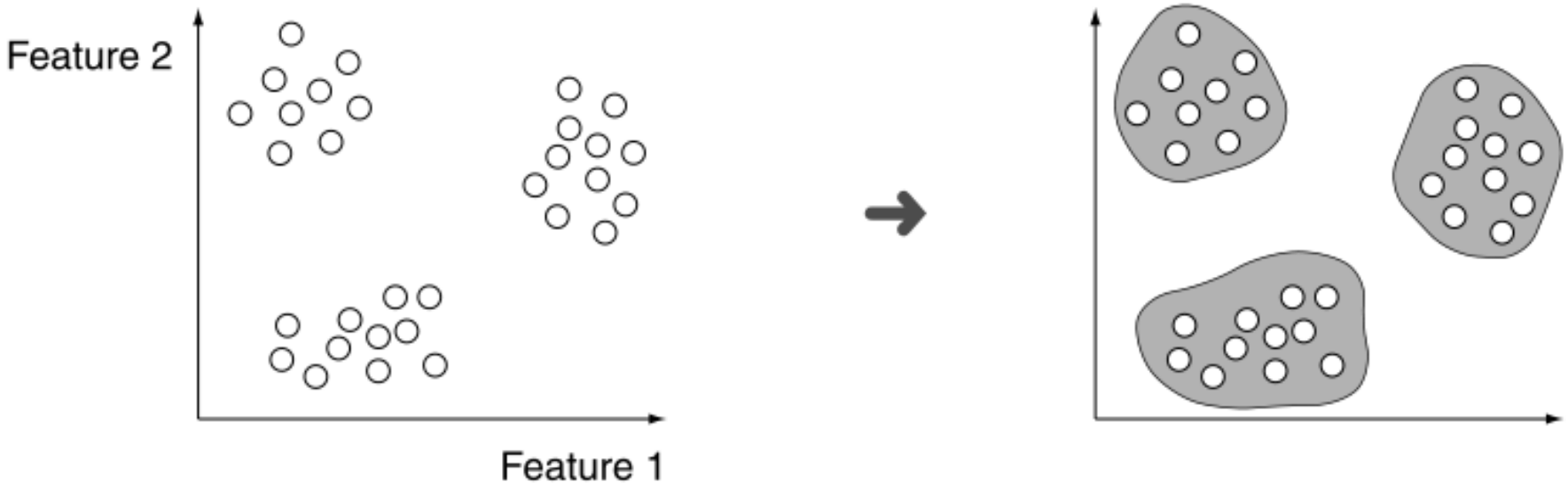
其中$\boldsymbol{c} = (c_1, \ldots, c_n)$, $c_i \in \{1, \ldots, K\}$

$c_i = c_j = k$ 表明 $x_i$ 和 $x_j$ 同属于标记为 $k$ 的簇

其中$\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K), \mu_k \in \mathbb{R}^d$ (所处空间与 $x_i$ 所在的空间相同)

每一个向量 $\mu_k$能够作为一个簇的代表（原型）

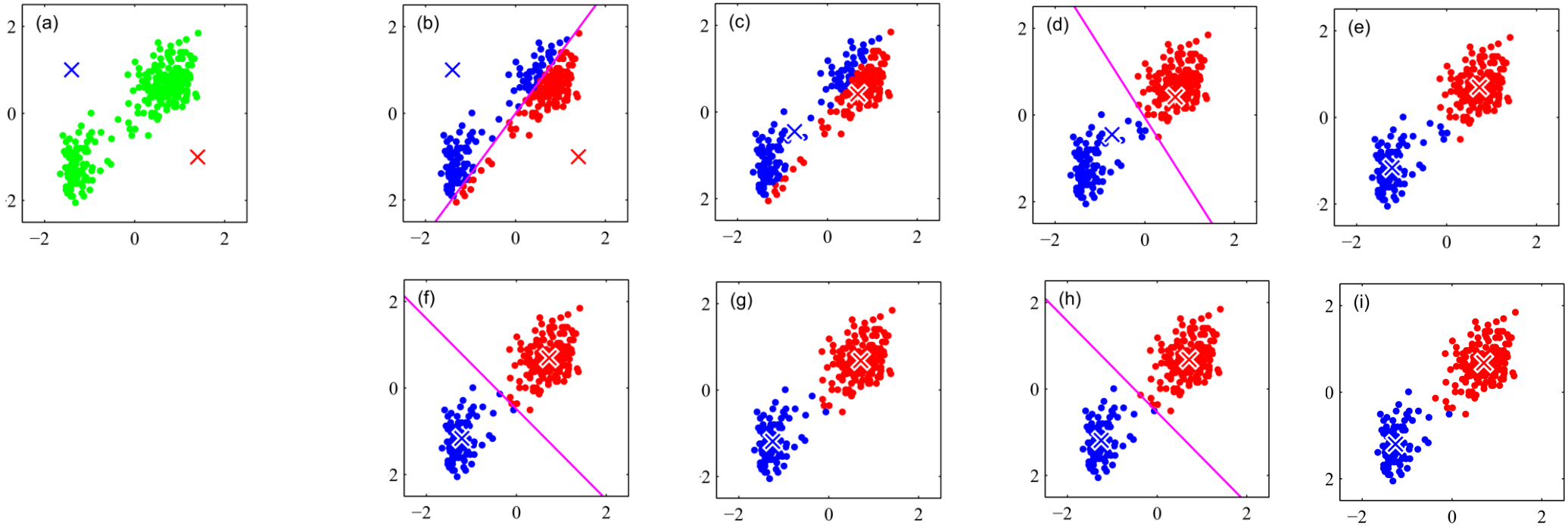
簇标记与原型满足 $c_i = \arg\min_k \ \|x_i - \mu_k\|^2$



# 原型聚类算法 – K均值

- K-MEANS

随机初始化 $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$.
进行下面的迭代直到收敛 ( $\boldsymbol{c}$ 和 $\boldsymbol{\mu}$ 不再发生变化)
更新 $c_i$ : $c_i = \arg\min_k \|x_i - \mu_k\|^2$
更新 $\mu_k$: $\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i \mathbb{1}\{c_i = k\}, n_k = \sum_{i=1}^{n} \mathbb{1}\{c_i = k\}$



# 原型聚类算法 – K均值

- K-MEANS

K均值求解$\boldsymbol{\mu}^*, \boldsymbol{c}^*$的过程能否从一个最小损失函数的角度来推导出来？

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{c}} \ ?$$

# 原型聚类算法 – K均值

- K-MEANS

K均值惩罚了簇内离散程度的和，这种离散程度是基于欧式距离来定义的

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{c}} \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2 = \sum_{k=1}^{K}\sum_{i:c_i=k}\|x_i - \mu_k\|^2$$

# 原型聚类算法 – K均值

- K-MEANS

K均值惩罚了簇内离散程度的和，这种离散程度是基于欧式距离来定义的

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{c}} \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2 = \sum_{k=1}^{K}\sum_{i:c_i=k}\|x_i - \mu_k\|^2$$

我们将变量分为$c$和$\mu$两类. 我们很难去同时优化它们，但是我们可以观察到:
固定$\mu$则很容易求得最优的$c$；固定$c$则很容易求得最优的$\mu$

# 原型聚类算法 – K均值

- K-MEANS

K均值惩罚了簇内离散程度的和，这种离散程度是基于欧式距离来定义的

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{c}} \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2 = \sum_{k=1}^{K}\sum_{i:c_i=k}\|x_i - \mu_k\|^2$$

我们将变量分为$c$和$\mu$两类. 我们很难去同时优化它们，但是我们可以观察到:
固定$\mu$则很容易求得最优的$c$；固定$c$则很容易求得最优的$\mu$

所以我们可以利用坐标下降法*coordinate descent*: 每次迭代固定一组参数，然后优化其余的参数。并在后续迭代中更换它们的角色：

# 原型聚类算法 – K均值

- K-MEANS

K均值惩罚了簇内离散程度的和，这种离散程度是基于欧式距离来定义的

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{c}} \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2 = \sum_{k=1}^{K}\sum_{i:c_i=k}\|x_i - \mu_k\|^2$$

我们将变量分为$c$和$\mu$两类. 我们很难去同时优化它们，但是我们可以观察到:
固定$\mu$则很容易求得最优的$c$；固定$c$则很容易求得最优的$\mu$

所以我们可以利用坐标下降法*coordinate descent*: 每次迭代固定一组参数，然后优化其余的参数。并在后续迭代中更换它们的角色：

输入: $x_1, \ldots, x_n$ where $x_i \in \mathbb{R}^d$.
随机初始化 $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$
迭代地进行下面两步:
固定 $\boldsymbol{\mu}$, 找到最优的 $c_i \in \{1, \ldots, K\}$ for $i = 1, \ldots, n$.
固定 $\boldsymbol{c}$, 找到最优的 $\mu_k \in \mathbb{R}^d$ for $k = 1, \ldots, K$.

# 原型聚类算法 – K均值

- K-MEANS

K均值惩罚了簇内离散程度的和，这种离散程度是基于欧式距离来定义的

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\mu,c} \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2 = \sum_{k=1}^{K}\sum_{i:c_i=k}\|x_i - \mu_k\|^2$$

我们将变量分为$c$和$\mu$两类. 我们很难去同时优化它们，但是我们可以观察到:
固定$\mu$则很容易求得最优的$c$；固定$c$则很容易求得最优的$\mu$

$$\mathcal{L} = (\underbrace{\sum_{k=1}^{K}\mathbb{1}\{c_1 = k\}\|x_1 - \mu_k\|^2}_{x_1 \text{与其分配的簇对应的类心距离}}) + \cdots + (\underbrace{\sum_{k=1}^{K}\mathbb{1}\{c_n = k\}\|x_n - \mu_k\|^2}_{x_n \text{与其分配的簇对应的类心距离}})$$

$$c_i = \arg\min_k \|x_i - \mu_k\|^2$$

# 原型聚类算法 – K均值

- K-MEANS

K均值惩罚了簇内离散程度的和，这种离散程度是基于欧式距离来定义的

$$\boldsymbol{\mu}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\mu},\boldsymbol{c}} \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}\{c_i = k\}\|x_i - \mu_k\|^2 = \sum_{k=1}^{K}\sum_{i:c_i=k}\|x_i - \mu_k\|^2$$

我们将变量分为$c$和$\mu$两类. 我们很难去同时优化它们，但是我们可以观察到:
固定$\mu$则很容易求得最优的$c$；固定$c$则很容易求得最优的$\mu$

$$\mathcal{L} = \underbrace{\left(\sum_{i=1}^{n}\mathbb{1}\{c_i = 1\}\|x_i - \mu_1\|^2\right)}_{\text{簇1的簇内离散程度 (欧式距离定义)}} + \cdots + \underbrace{\left(\sum_{i=1}^{n}\mathbb{1}\{c_i = K\}\|x_i - \mu_K\|^2\right)}_{\text{簇}K\text{的簇内离散程度 (欧式距离定义)}}$$

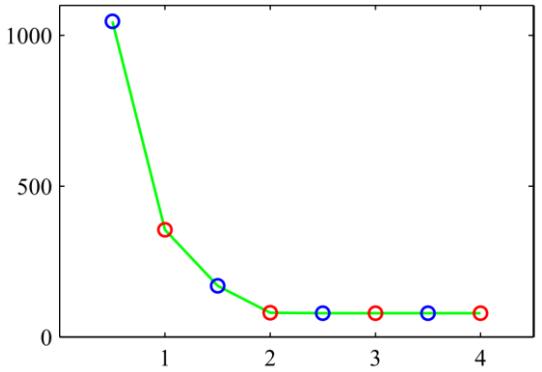
$$\mu_k = \arg\min_{\mu}\sum_{i=1}^{n}\mathbb{1}\{c_i = k\}\|x_i - \mu\|^2$$

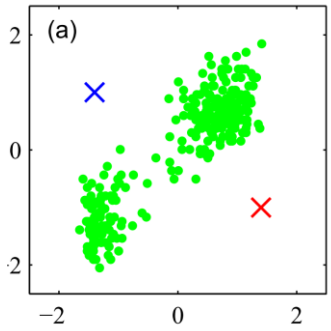
$$\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i\mathbb{1}\{c_i = k\}, n_k = \sum_{i=1}^{n}\mathbb{1}\{c_i = k\}$$

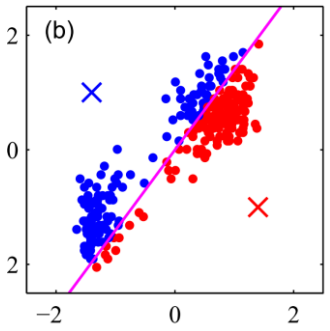
# 原型聚类算法 – K均值

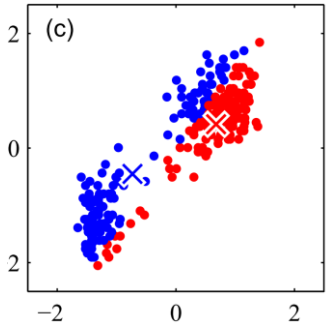
- K-MEANS

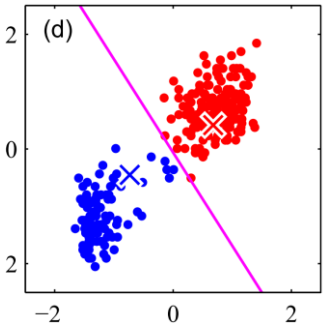
随机初始化 $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$.
进行下面的迭代直到收敛 ( $\boldsymbol{c}$ 和 $\boldsymbol{\mu}$ 不再发生变化)
更新 $c_i$ : $c_i = \arg\min_k \|x_i - \mu_k\|^2$
更新 $\mu_k$: $\mu_k = \frac{1}{n_k}\sum_{i=1}^{n} x_i \mathbb{1}\{c_i = k\}, n_k = \sum_{i=1}^{n} \mathbb{1}\{c_i = k\}$

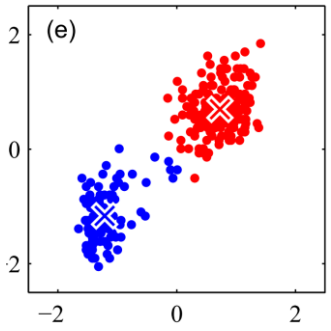


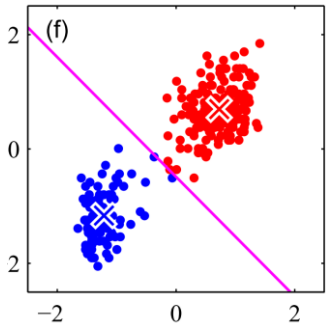



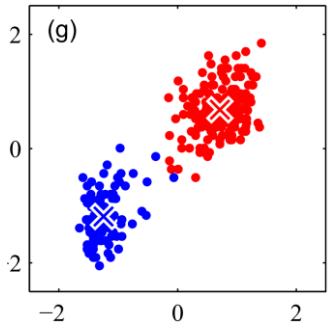



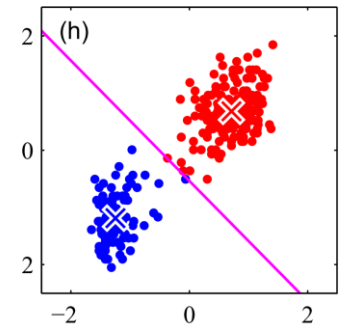



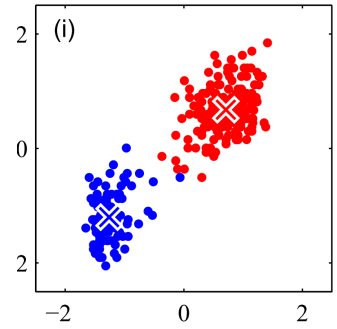



# 原型聚类算法 – K均值

- K-MEANS

收敛性可以得到保证：

- 每次更新完 $c_i$ 和 $\mu_k$ 后 $\mathcal{L}$ 的值便会下降.
- 即 $\mathcal{L}$ 时单调递减的 monotonically decreasing.
- $\mathcal{L} \geq 0$, 说明其会收敛于某个值 (不过不一定是0)

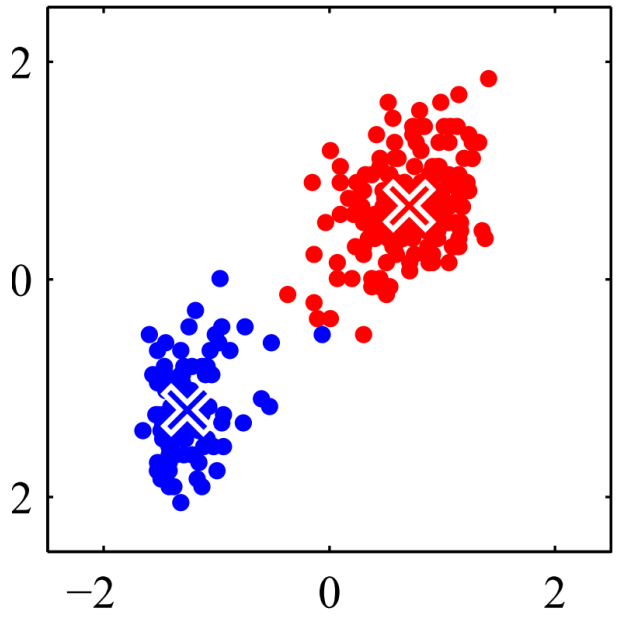
但注意到，K-means算法不一定能够收敛到全局最优解。当$c$不再改变时，表示其收敛到一个局部最优解 *local optimal solution*. 这说明不同的随机初始化可能导致不同的聚类结果。

现实中，可以运行多次算法，然后取目标函数的最小值对应的结果。

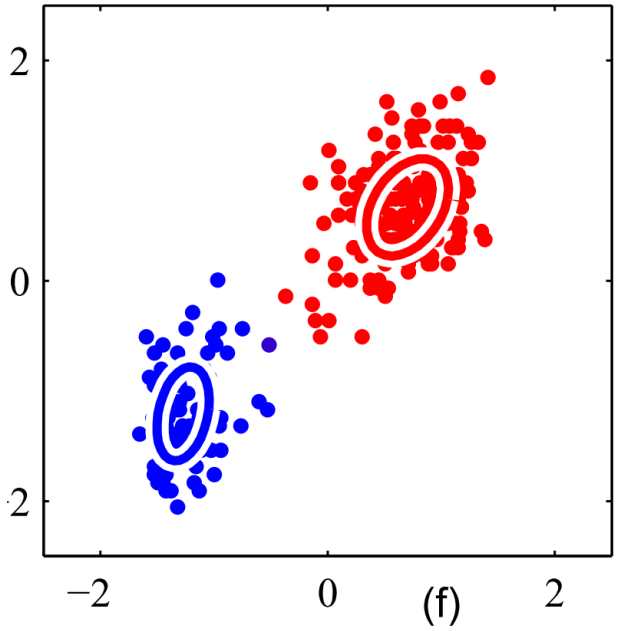
# 原型聚类算法

- 定义

原型聚类 *prototype-based clustering* 假设聚类结果能够通过一组原型刻画，在现实聚类任务中极为常用。通常情形下，算法先对原型进行初始化，然后对原型进行迭代更新求解。K-means 就是一种典型的原型聚类。




均值向量作为原型



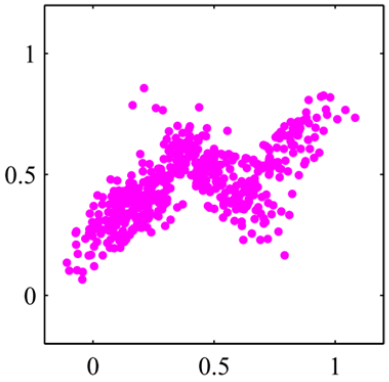

高斯分布作为原型

# 原型聚类算法 – 高斯混合模型

- 混合模型

假设数据是拥有一个潜在的“类型”属性的，“类型”的概念对应于参数不同的分布。这样的假设下，数据对应的生成分布应该是一个混合分布，混合模型对这种混合分布进行了设定：
通过将更基本的概率分布（例如高斯分布）用线性组合的方式进行叠加，可以被形式化为概率模型，被称为混合模型 *mixture models*。

例如高斯混合模型：




$$p_{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{K} \alpha_i \cdot p(\boldsymbol{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

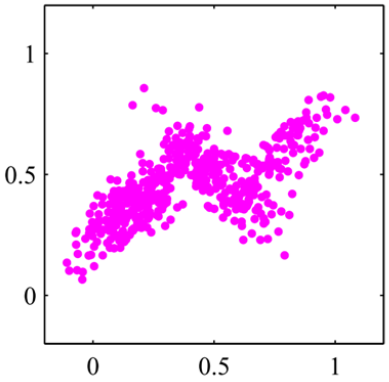
$$\sum_{i=1}^{K} \alpha_i = 1, \alpha_i \geq 0$$

# 原型聚类算法 – 高斯混合模型

- 混合模型

假设数据是拥有一个潜在的“类型”属性的（类似聚类的想法）。这样的假设下，数据对应的生成分布应该是一个混合分布，对应混合模型：
通过将更基本的概率分布（例如高斯分布）用线性组合的方式进行叠加，可以被形式化为概率模型，被称为混合模型 *mixture models*。

例如高斯混合模型：

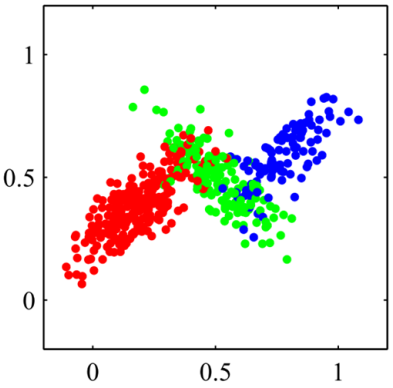



$$p_{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{K} \alpha_i \cdot p(\boldsymbol{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

$$\sum_{i=1}^{K} \alpha_i = 1, \alpha_i \geq 0$$

【假设】针对于每个 $x_j, j = 1, \dots n$ 引入 $z_j \in \{1, \dots K\}$ 表示它是由哪个高斯分布生成的。高斯混合模型假设数据的生成过程是:

1. 第 $j$ 个样本首先被分配了一个类型 $z_j \sim \text{Discrete}(\alpha)$
2. 然后通过类型对应的高斯分布被采样了出来: $x_j \sim \mathcal{N}\left(\mu_{z_j}, \Sigma_{z_j}\right)$

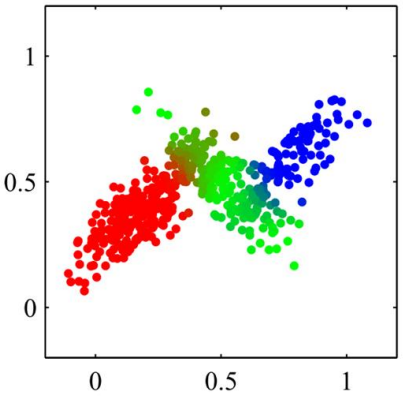



【I/O】给定数据集$x_1, \dots, x_n$，高斯混合模型负责求解出$\boldsymbol{\mu}_{1..K}, \boldsymbol{\Sigma}_{1..K}, \alpha_{1\dots K}$，之后可以得到：

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

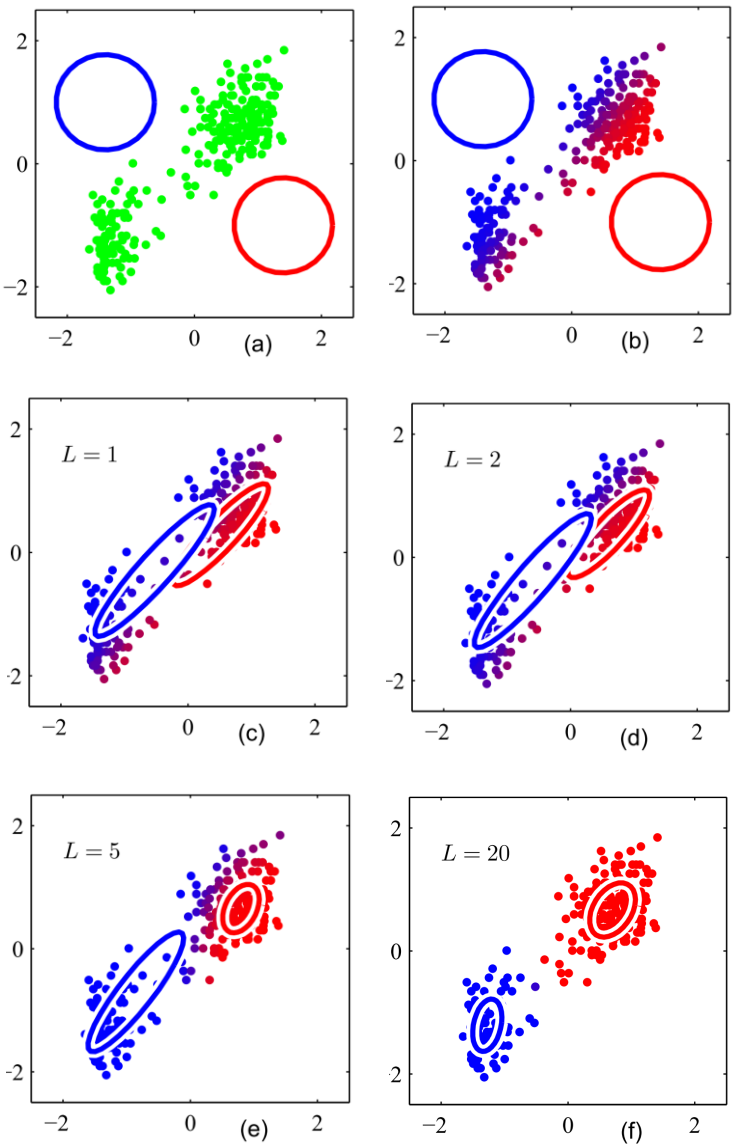
$$= \frac{\alpha_i \cdot p(\boldsymbol{x}_j \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{l=1}^{K} \alpha_l \cdot p(\boldsymbol{x}_j \mid \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} = \gamma_{ji}$$

$$c_j = \arg\max_{i \in \{1,2,\dots,K\}} \gamma_{ji}$$



# 原型聚类算法 – 高斯混合模型

- $\boldsymbol{\mu}_{1..K}, \boldsymbol{\Sigma}_{1..K}, \alpha_{1...K}$的学习： EM算法




$$\Theta = \{\alpha_1, \dots \alpha_k, \boldsymbol{\mu}_1, \dots \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots \boldsymbol{\Sigma}_K\}$$

$$\Theta_{\text{MLE}} = \arg\max_{\Theta} \mathcal{L}(\Theta \mid X)$$

$$= \arg\max_{\Theta} \left( \sum_{j=1}^{n} \log \sum_{i=1}^{K} \alpha_i \mathcal{N}(X \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \right)$$

$$\boldsymbol{\mu}_{1..K}, \boldsymbol{\Sigma}_{1..K}, \alpha_{1...K} \longleftrightarrow \gamma_{ji}$$

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

$$\alpha_i = \frac{\sum_{j=1}^{n} \gamma_{ji}}{n}$$

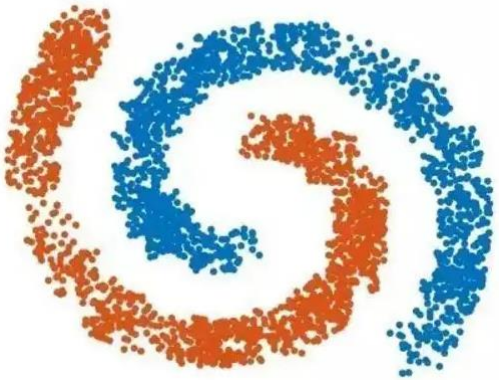
$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{n} \gamma_{ji} \left( \boldsymbol{x}_j - \boldsymbol{\mu}_i \right) \left( \boldsymbol{x}_j - \boldsymbol{\mu}_i \right)^{\mathrm{T}}}{\cdot \sum_{j=1}^{n} \gamma_{ji}}$$

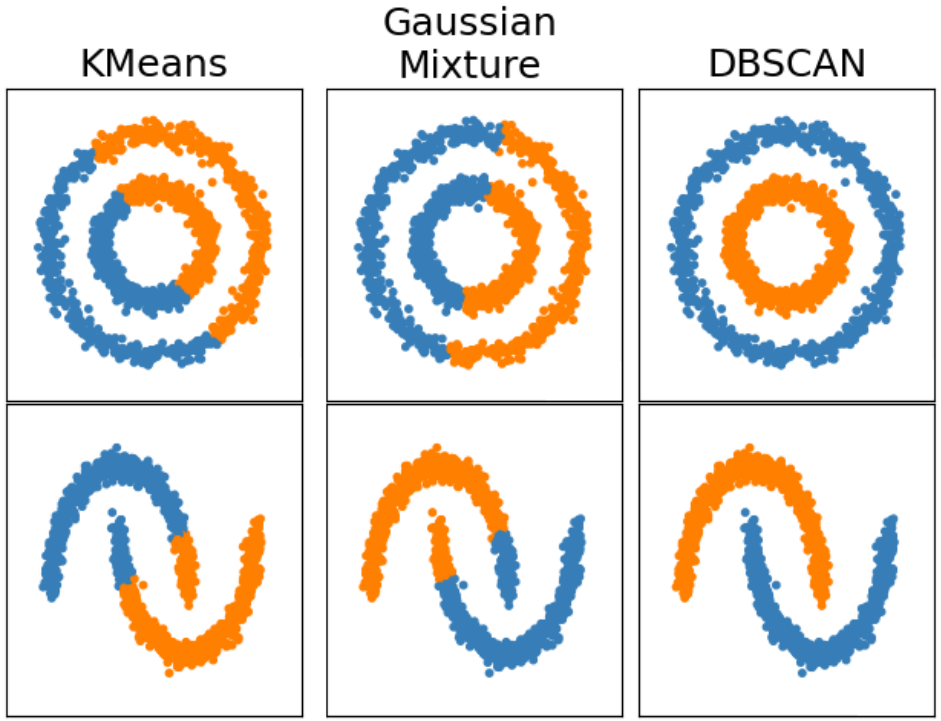
$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} \gamma_{ji} \boldsymbol{x}_j}{\sum_{j=1}^{m} \gamma_{ji}}$$

# 密度聚类算法

- 定义

密度聚类是基于密度来聚类的，此类算法假设聚类结构能通过样本分布的紧密程度确定。通常情形下，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以获得最终的聚类结果。





# 密度聚类算法 – DBSCAN

- DBSCAN

DBSCAN 是一种著名的密度聚类算法，它基于一组“邻域” 参数 $(\epsilon, MinPts)$ 来刻画样本的地位与样本之间的紧密程度。

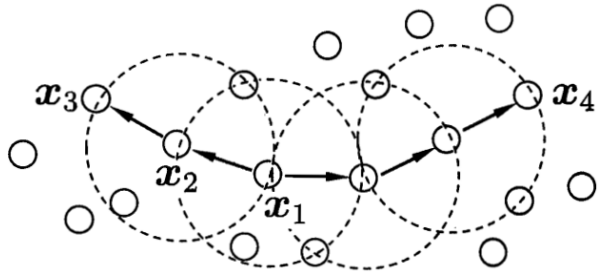
核心对象(core object): 若 $\boldsymbol{x}_j$ 的 $\epsilon$-邻域至少包含 $MinPts$ 个样本, 即 $\left|N_\epsilon(\boldsymbol{x}_j)\right| \geqslant MinPts$, 则 $\boldsymbol{x}_j$ 是一个核心对象;

$\epsilon$-邻域: 对 $\boldsymbol{x}_j \in D$, 其 $\epsilon$-邻域包含样本集 $D$ 中与 $\boldsymbol{x}_j$ 的距离不大于 $\epsilon$ 的样本, 即 $N_\epsilon(\boldsymbol{x}_j) = \{\boldsymbol{x}_i \in D \mid \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) \leqslant \epsilon\}$;

密度直达(directly density-reachable): 若 $\boldsymbol{x}_j$ 位于 $\boldsymbol{x}_i$ 的 $\epsilon$-邻域中, 且 $\boldsymbol{x}_i$ 是 核心对象, 则称 $\boldsymbol{x}_j$ 由 $\boldsymbol{x}_i$ 密度直达;

密度可达(density-reachable): 对 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$, 若存在样本序列 $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n$, 其中 $\boldsymbol{p}_1 = \boldsymbol{x}_i, \boldsymbol{p}_n = \boldsymbol{x}_j$ 且 $\boldsymbol{p}_{i+1}$ 由 $\boldsymbol{p}_i$ 密度直达, 则称 $\boldsymbol{x}_j$ 由 $\boldsymbol{x}_i$ 密度可达;

密度相连 (density-connected): 对 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$, 若存在 $\boldsymbol{x}_k$ 使得 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 均由 $\boldsymbol{x}_k$ 密度可达, 则称 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 密度相连.




虚线显示出 $\epsilon$-邻域

$\boldsymbol{x_1}$ 是核心对象

$\boldsymbol{x_2}$ 由 $\boldsymbol{x_1}$ 密度直达

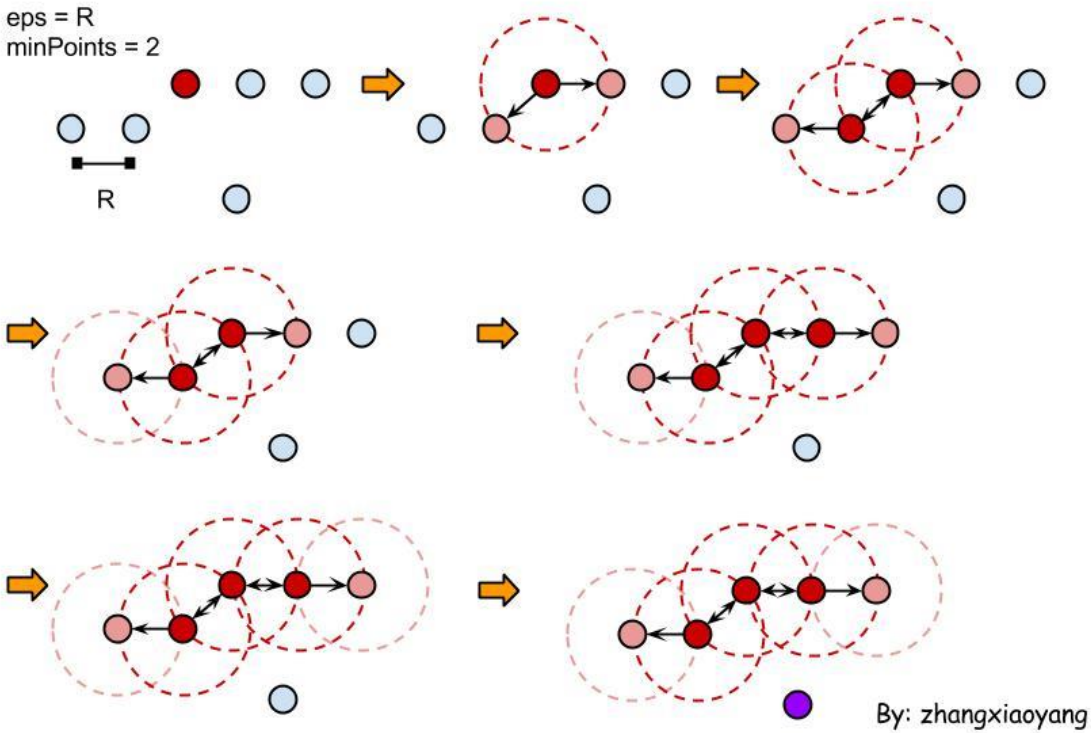
$\boldsymbol{x_3}$ 由 $\boldsymbol{x_1}$ 密度可达

$\boldsymbol{x_3}$ 与 $\boldsymbol{x_4}$ 密度相连

# 密度聚类算法 – DBSCAN

- DBSCAN

DBSCAN通过检查数据集中每点的$\epsilon$邻域来搜索簇，
如果点$\boldsymbol{x}_j$为核心对象，则尝试创建一个簇
创建的方法是陆续将$\boldsymbol{x}_j$密度可达的对象加入这个簇
迭代上述过程（可能涉及一些密度可达簇的合并）
当没有新的点添加到任何簇时，该过程结束.



# 评估

- 外部指标

“外部指标” (external index) 将聚类结果与某个“参考模型” (reference model)进行比较。假设聚类模型输出为 $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$, 参考模型的输出为 $\mathcal{C}^* = \{C_1^*, C_2^*, \dots, C_s^*\}$, 记 $\boldsymbol{\lambda}, \boldsymbol{\lambda}^*$ 为两个输出对应的簇标记。计算以下数量:

$$a = |SS|,\ SS = \left\{\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \mid \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\right\}$$
$$b = |SD|,\ SD = \left\{\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \mid \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\right\}$$
$$c = |DS|,\ DS = \left\{\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \mid \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\right\}$$
$$d = |DD|,\ DD = \left\{\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \mid \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\right\}$$

由此得出以下指标：

Rand Index

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

# 评估

- 内部指标

“内部指标”(internal index)直接考察聚类结果而不利用任何参考模型

假设聚类模型输出为$\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$
通常以“簇内相似，簇间相异”的出发点来评估聚类输出，可定义

$$\text{avg}(C) = \frac{2}{|C|(|C|-1)} \sum_{1 \leqslant i < j \leqslant |C|} \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

$$d_{\text{cen}}(C_i, C_j) = \text{dist}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$$

然后设计指标 DB Index

$$\text{DBI} = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left( \frac{\text{avg}(C_i) + \text{avg}(C_j)}{d_{\text{cen}}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \right)$$