聚类

聚类的常见用途

- 知识发现 发现事物之间的潜在关系
- 异常值检测
- 特征提取 数据压缩的例子

回顾有监督机器学习

- 给定训练集 X 和 标签Y
- 选择模型
- 学习（目标函数的最优化）- > 生成模型（本质上是一组参数）

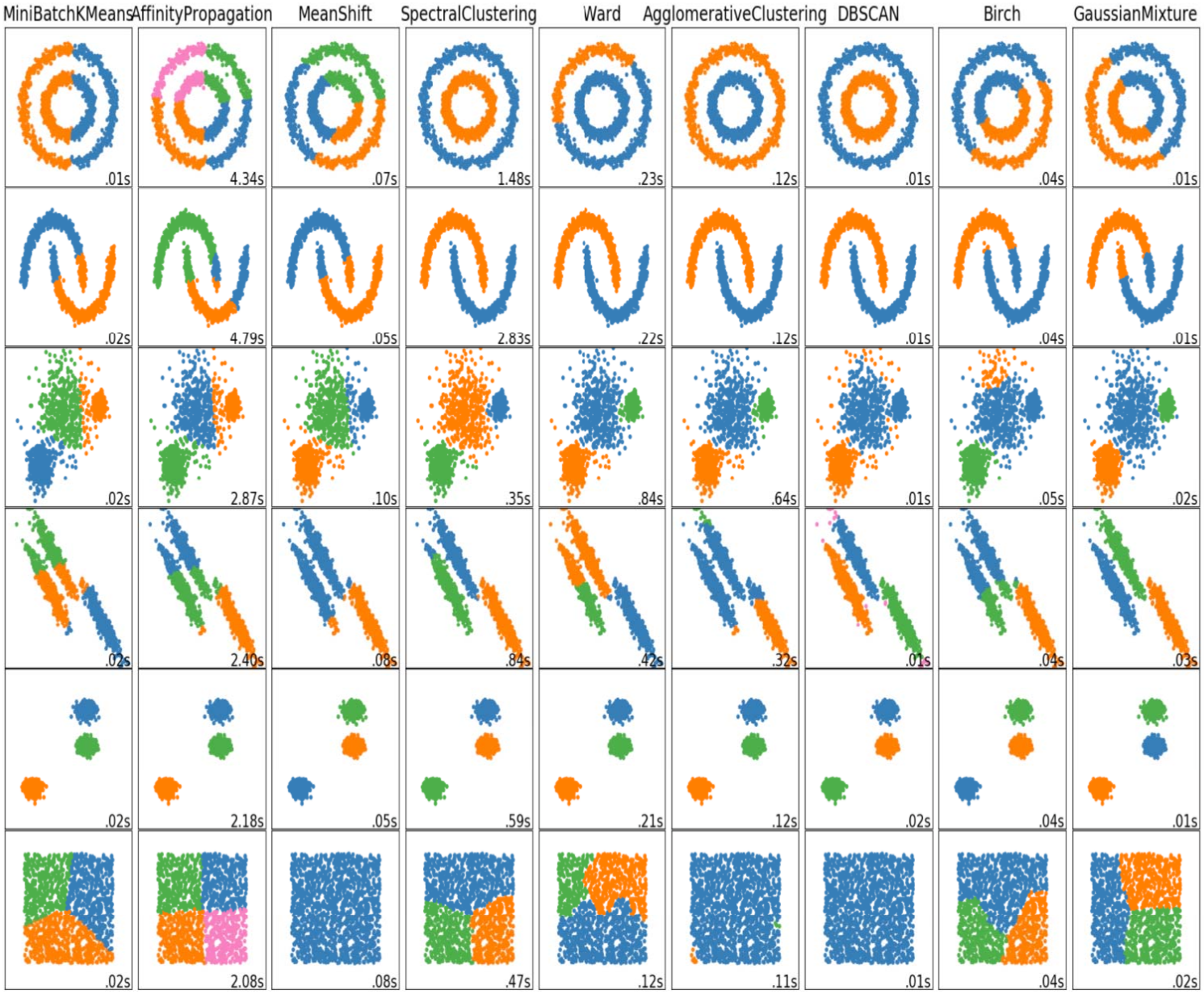
- 根据生成的一组参数进行预测、分类等任务

无监督机器学习

- 拿到的数据只有X ，没有标签 只根据X的相似程度做一些事情

- Clustering 聚类
  - 对于大量未标注的数据集，按照内在相似性来分为多个类别（簇），目标：类别内相似度大，类别间相似度小
  - 也可以用来改变数据的维度，可以将聚类结果作为一个维度添加到训练集中
  - 用one hot编码将维度缩减到类别数
- Dimensionality reduction 降维

聚类算


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
Birch
GaussianMixture
.01s
4.34s
.07s
1.48s
.23s
.12s
.01s
.04s
.01s
.02s
4.79s
.05s
2.83s
.22s
.12s
.01s
.04s
.01s
.02s
2.87s
.10s
.35s
.84s
.64s
.01s
.05s
.02s
.02s
2.40s
.08s
.84s
.42s
.32s
.01s
.04s
.03s
.02s
2.18s
.05s
.59s
.21s
.12s
.02s
.04s
.01s
.02s
2.08s
.08s
.47s
.12s
.11s
.01s
.04s
.02s

相似度

数据间的相似度

- 每一条数据 都可以理解为多维空间中的一个点
- 可以根据点和点之间的距离来评价数据间的相似度
- 欧氏距离

二维空间的公式

$$0\rho = \text{sqrt}((x1-x2)^2+(y1-y2)^2) \quad |\boldsymbol{x}| = \sqrt{(x2 + y2)}$$

三维空间的公式

$$0\rho = \sqrt{((x1-x2)^2+(y1-y2)^2+(z1-z2)^2)} \quad |\boldsymbol{x}| = \sqrt{(x2 + y2 + z2)}$$

数据间的相似度

- 闵可夫斯基距离

$$d(X,Y) = \sqrt[p]{|x_1 - y_1|^p + |x_2 - y_2|^p + ... + |x_n - y_n|^p}$$

- P=1 曼哈顿距离

$$d(X,Y) = |x_1 - y_1| + |x_2 - y_2| + ... + |x_n - y_n|$$

- P=2 欧氏距离

- P=无穷 切比雪夫距离 那个维度差值最大就是哪个差值作为距离

余弦距离

- 将数据看做空间中的点的时候，评价远近可以用欧氏距离或者余弦距离
- 步骤：
  - 将数据映射为高维空间中的点（向量）
  - 计算向量间的余弦值
  - 取值范围[-1,+1] 越趋近于1代表越相似，越趋近于-1代表方向相反，0代表正交

$$\cos\theta = \frac{a \bullet b}{\|a\|\|b\|}$$

$$\cos\theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

余弦距离评价文章相似度

- 想要评价两篇文章是否相似，除了jaccard系数，还可以使用余弦距离
  - 1.将文章分词
  - 2.将文章转变为词向量（ TFIDF）
  - 3.转换为词向量后就可以将文章映射到高维空间变为一个向量
  - 4.文章之间的向量的余弦距离代表文章之间的相似程度

$$\cos\theta = \frac{a \bullet b}{\|a\|\|b\|}$$

$$\cos\theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

TF-IDF

- TF 在给定的文档中某个词出现的概率
  - 某篇文章内部 某词出现的次数/文章的总词数

$$tf_{d,t} = \frac{n_{d,t}}{\sum_k n_{d,k}}$$

- DF 语料库中包含词t的总文章数
- IDF 逆文件频率 $idf_t = \log \frac{|D|}{df_t + 1}$
  - 代表这个词在语料库中的重要程度，越稀有代表越重要，为了减低臭大街的词对于相似度的贡献
- TF-IDF $tfidf_{d,t} = tf_{d,t} * idf_t,$

数据相似度 - Jaccard相似系数

- 用来衡量有限样本集之间的相似程度

$$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|}$$

- 当集合A,B都为空时，定义J（A,B)=1
- 取值范围？ 大小关系？
- Jaccard 距离

$$d_j(A,B)=1-J(A,B)=\frac{|A\cup B|-|A\cap B|}{|A\cup B|}=\frac{A\Delta B}{|A\cup B|}$$

Jaccard 例子

- 假设用户喜欢的商品列表[8，9，17，25，4]
- 两个备选推荐，哪个更好呢？
  - [9,10,17,24,4,8]　　[8,9,25]

- 计算
  - J1 =?　J2=?

- 可以应用于 网页去重、文本相似度分析

回顾precision 和 recall

- PRECISION 给出的正确中有多少正确的
- Recall： 所有的正确中有多少给出了

聚类

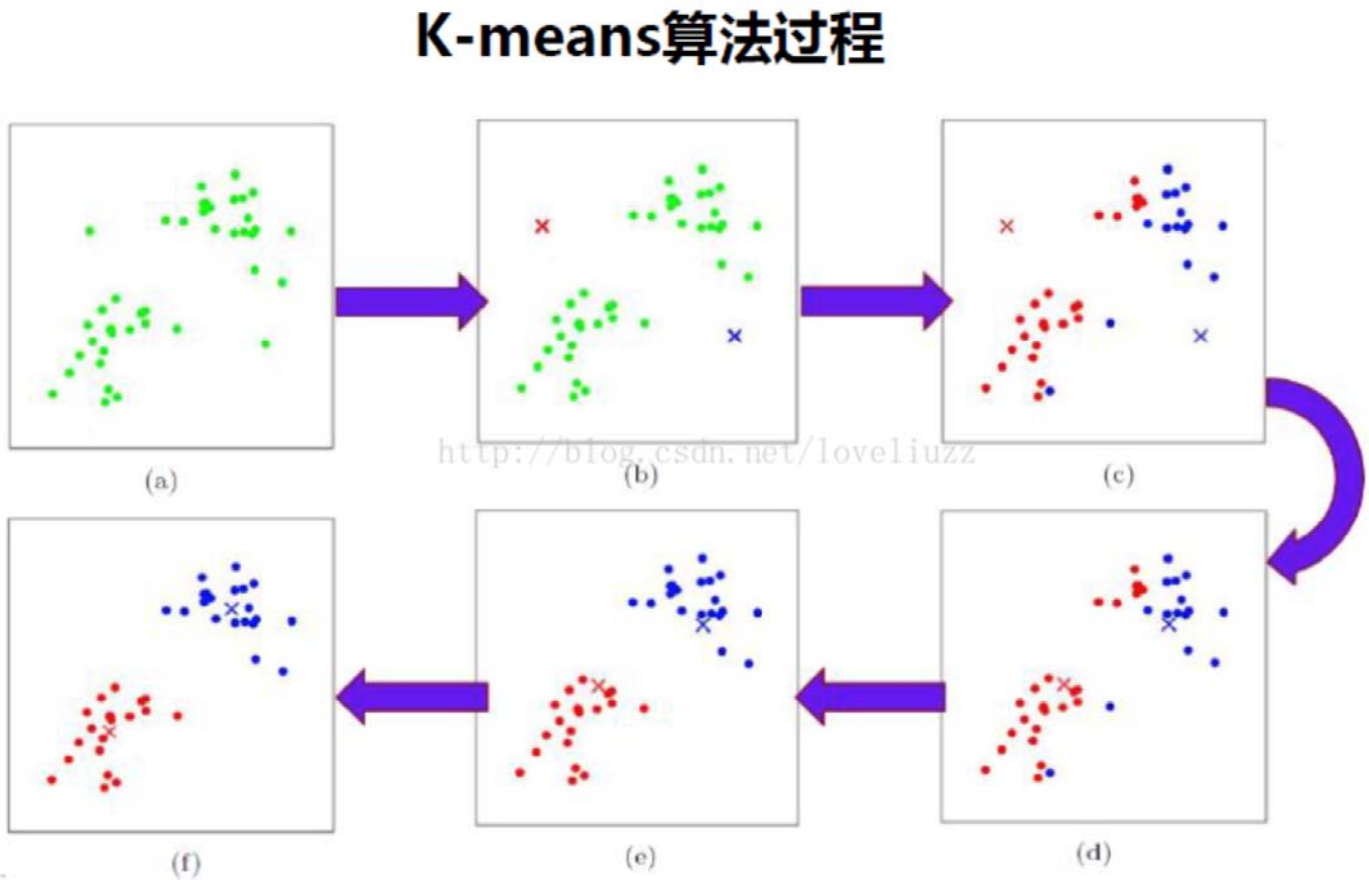
- 将N个样本映射到K个簇中
- 每个簇至少有一个样本

- 基本思路：
  - 先给定K个划分，迭代样本与簇的隶属关系，每次都比前一次好一些
  - 迭代若干次，就能得到比较好的结果

K-means

- 算法步骤：
  - 选择K个初始的簇中心
    - 怎么选？
  - 逐个计算每个样本到中心的距离，将样本归属到距离最小的那个簇中心的簇中
  - 每个簇内部计算平均值，更新簇中心
  - 开始迭代

K-means



K-means的特点

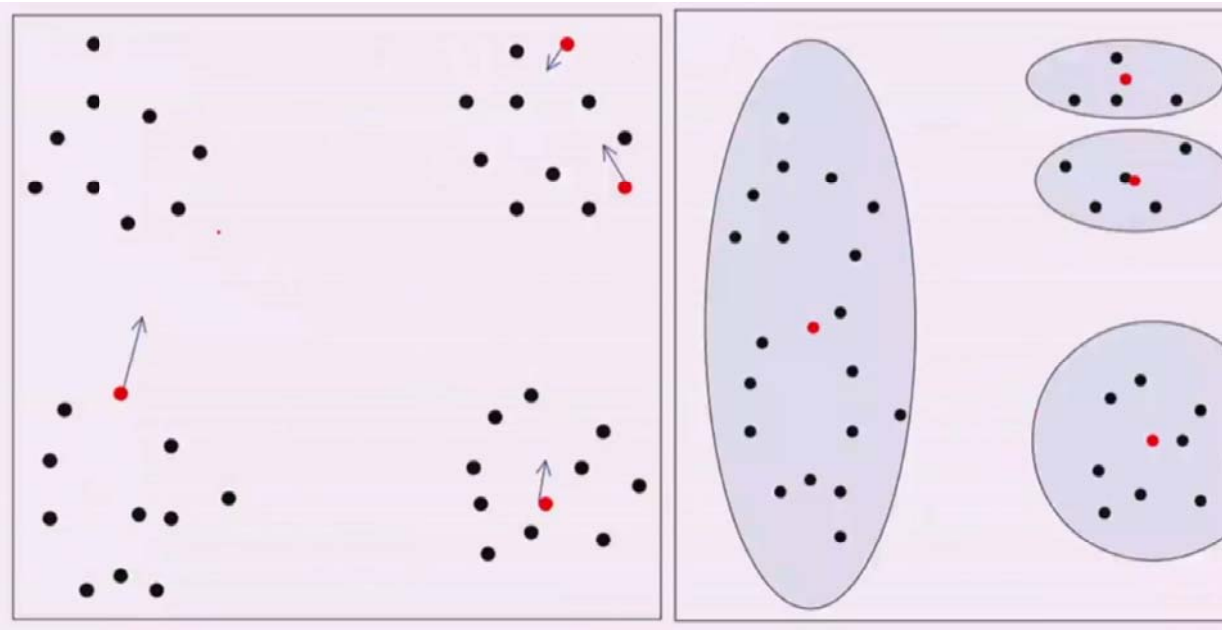
- 优点：
  - 简单，效果不错

- 缺点
  - 对异常值敏感
  - 对初始值敏感
  - 对某些分布聚类效果不好

K-means的变种（优化）

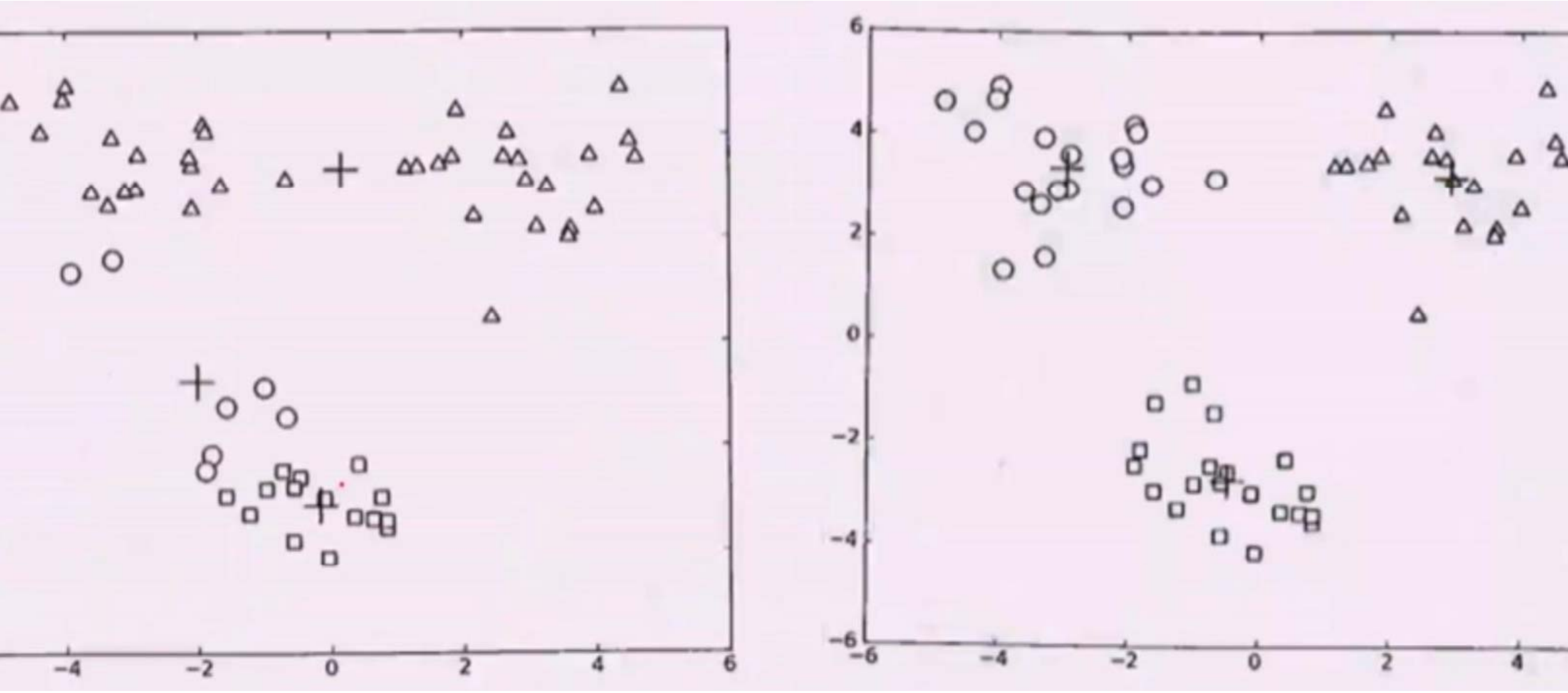
- K-Mediods
  - 计算新的簇中心的时候不再选择均值，而是选择中位数
  - 抗噪能力得到加强

二分K-means



- K-means的损失函数
  - 每个点到中心点的位置 MSE
- 分别计算四个簇的mse，会发现有两个簇的MSE很小，一个簇的MSE很大
- 选择合并簇中心点比较近，MSE很小簇，切分簇中心离其他簇中心比较远，MSE比较大的簇，重新进行K-means聚类

二分Kmeans



K-means++

- K-means选择一个好的初始中心点非常重要
- K-means++ 改变初始中心点的位置
  - 目标：初始化簇中心点稍微远一些
  - 步骤：
    - 随机选择第一个中心点
    - 计算每个样本到第一个中心点的距离
    - 将距离转化为概率
    - 概率化选择

K-means的损失函数

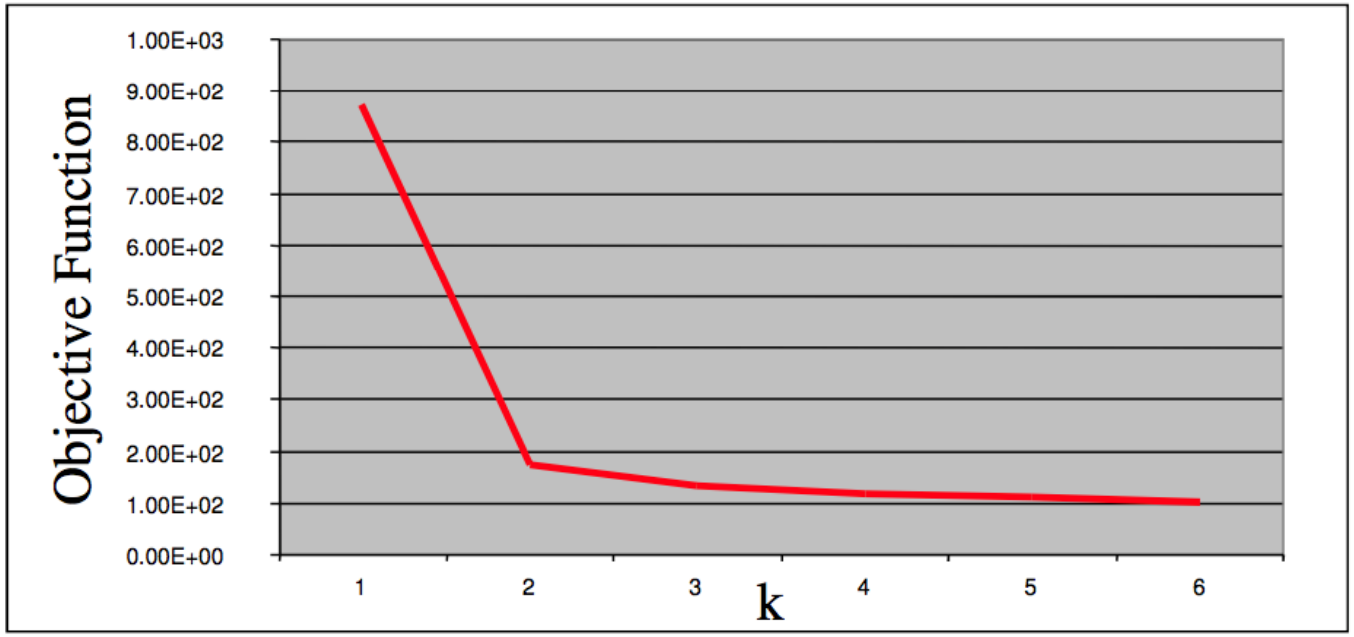
- K-means可以work的理论基础是假定了数据点符合同方差的高斯分布模型

$$J(\mu_1, \mu_2, \cdots \mu_k) = \frac{1}{2} \sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_i - \mu_j)^2$$

- 通过最大似然估计得到的k-means的迭代方法
- 这个函数是个非凸函数，根据初始值不同只能得到局部最优解

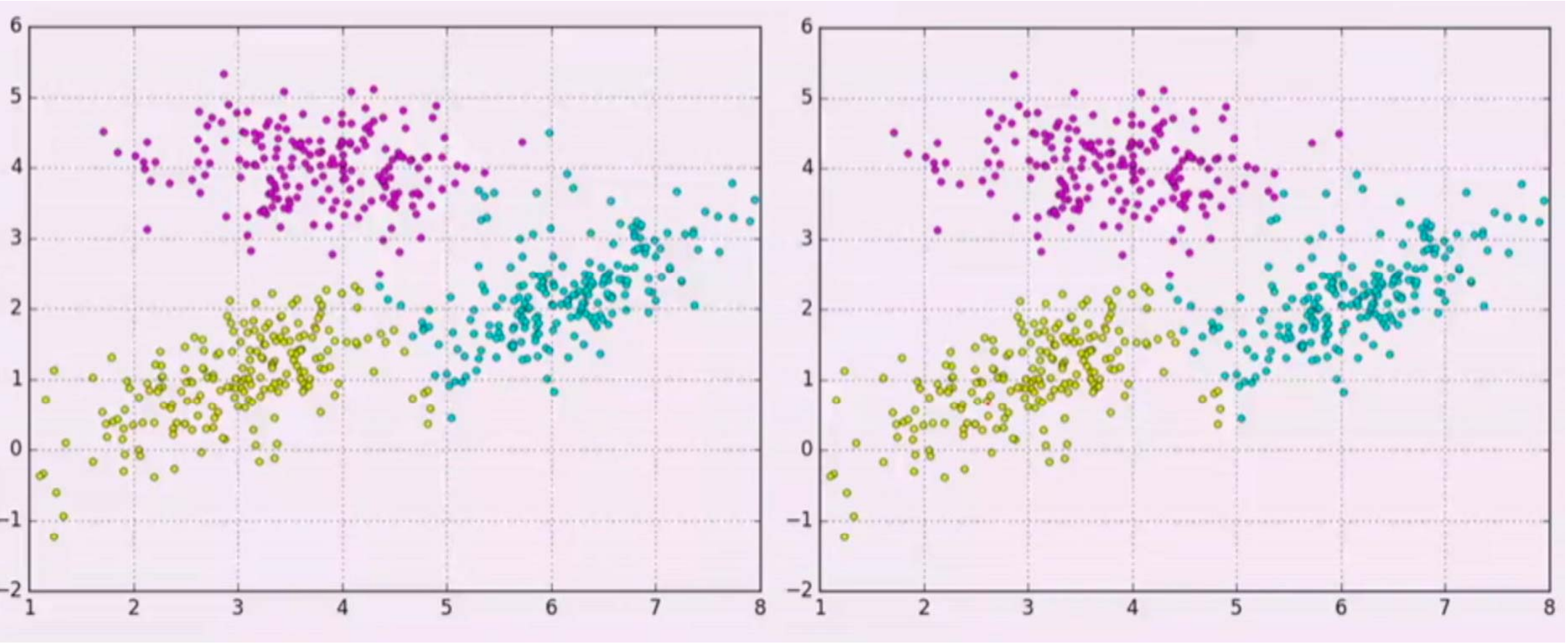
K的选择

- K=N的时候 损失函数为0
- K=1 损失函数=原始数据的方差
- 选择一开始下降快的

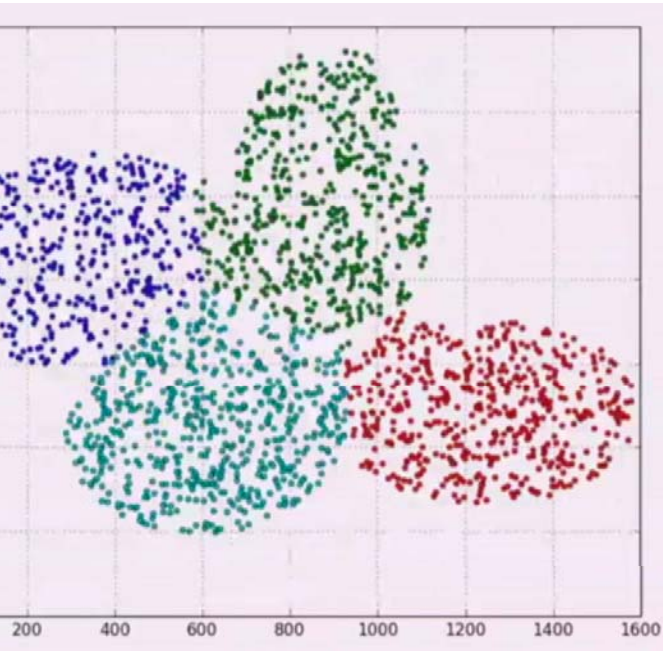


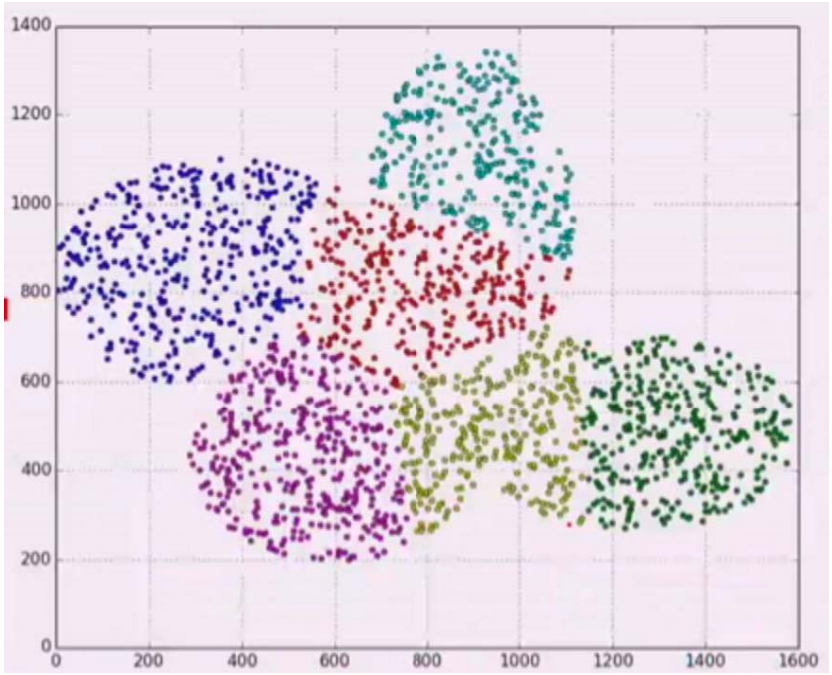
Mini-batch Kmeans

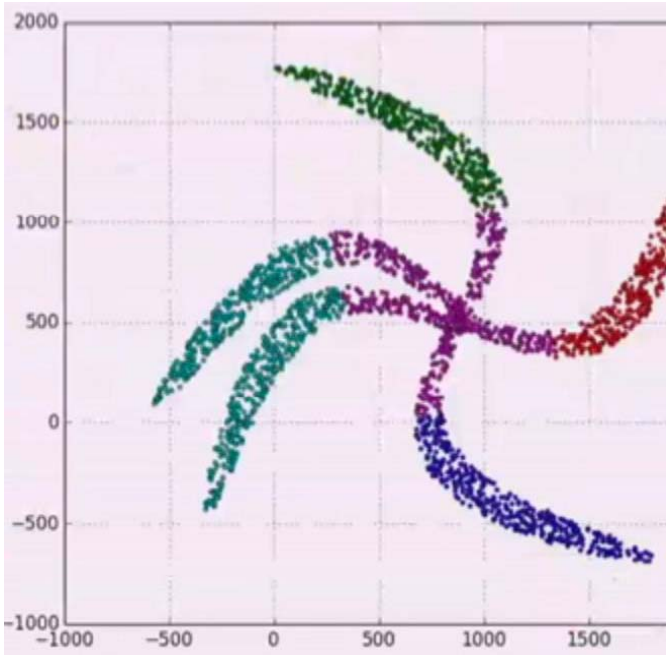
- 随机选择一部分的数据求均值



不同的k


200
400
600
800
1000
1200
1400
1600

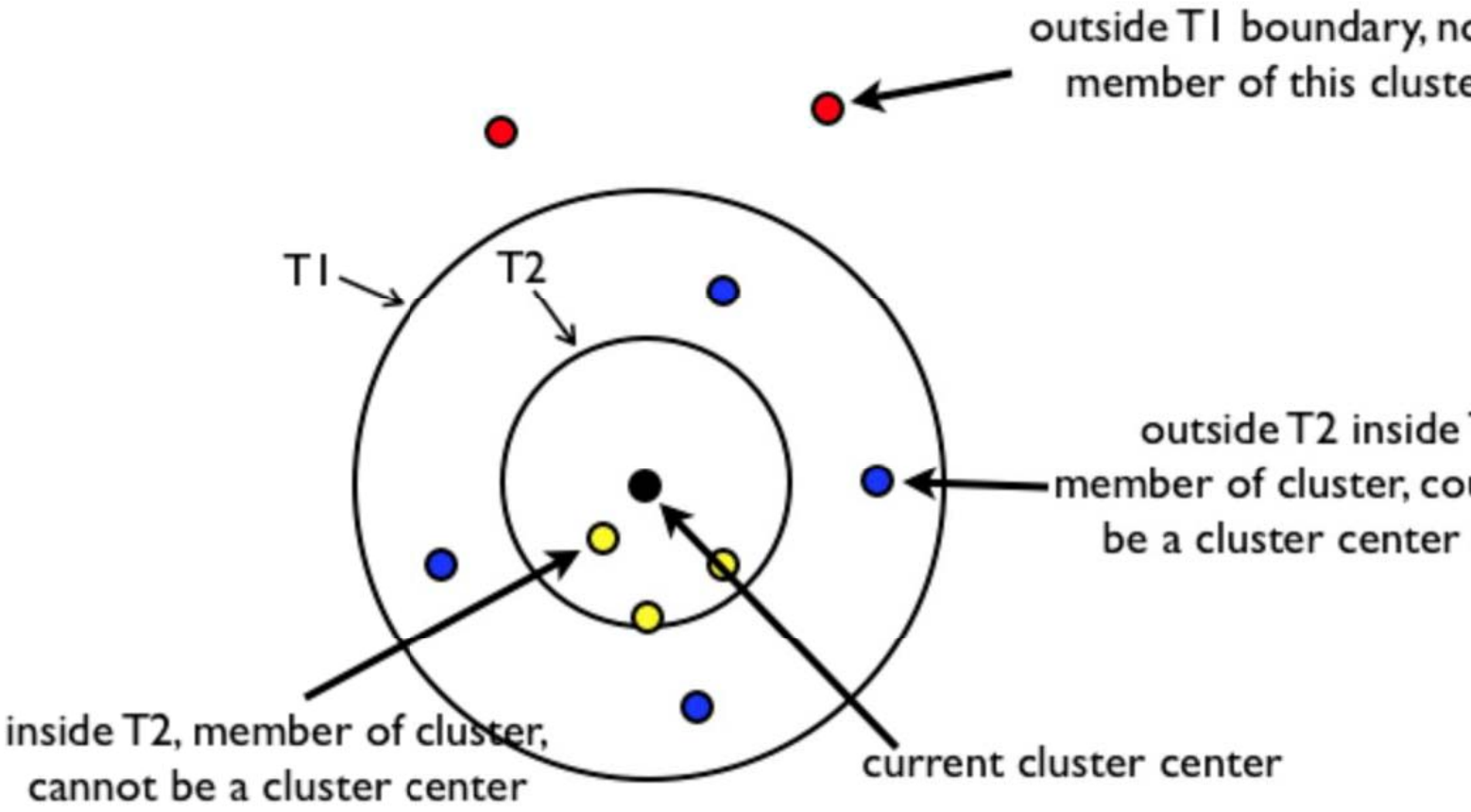

0
200
400
600
800
1000
1200
1400
1600


2000
1500
1000
500
0
−500
−1000
−1000
−500
0
500
1000
1500

Canopy聚类

- 一次迭代出结果
- T1 T2 超参数

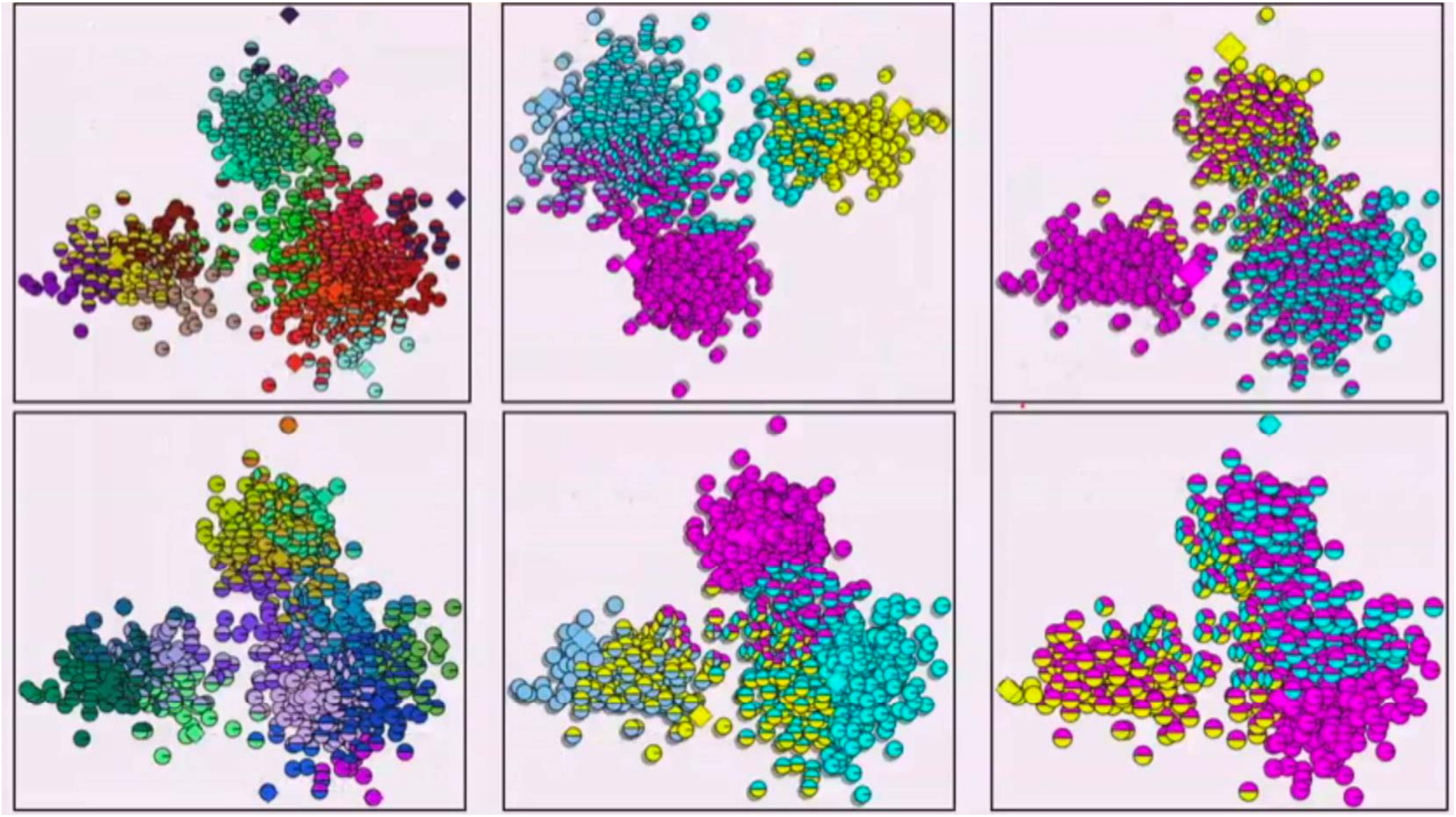
- 很少单独使用
- 结合kmeans



Canopy聚类的步骤

- 设置超参数T1, T2 T1>T2
  - WHILE D 非空：
    - 随机产生d属于D作为中心点
    - 计算所有点到d得距离
    - 所有距离<t1的点归属于d的中心点
    - 从D中删除d 及距离小于t2的点

Canopy聚类的效果



聚类的评估方法

- Given Label
  - 均一性 homogeneity　　metrics.homogeneity_score
  - 完整性 completeness　　metrics.completeness_score

  - V-Measure　　metrics.v_measure_score(labels_true, labels_pred)

$$v_\beta = \frac{(1+\beta)\cdot h \cdot c}{\beta \cdot h + c}$$

ARI评估

- 也是given-label情况下的评估指标
- 类似于分类问题中的ACC

Rand Index

- 维基百科：https://en.wikipedia.org/wiki/Rand_index

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

- a, the number of pairs of elements in S that are in the same subset in X and in the same subset in Y
- b, the number of pairs of elements in S that are in different subsets in X and in different subsets in Y
- c, the number of pairs of elements in S that are in the same subset in X and in different subsets in Y
- d, the number of pairs of elements in S that are in different subsets in X and in the same subset in Y

Adjusted Rand Index

- 1. 计算相依表（**contingency table**）

$X \backslash Y$	Y_1	Y_2	$\dots$	Y_s	Sums
X_1	n_{11}	n_{12}	$\dots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\dots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\dots$	n_{rs}	a_r
Sums	b_1	b_2	$\dots$	b_s	

Given a set S of n elements, and two groupings or partitions (*e.g.* clusterings) of these elements, namely $X = \{X_1, X_2, \dots, X_r\}$ $Y = \{Y_1, Y_2, \dots, Y_s\}$, the overlap between X and Y can be summarized in a contingency table $[n_{ij}]$ where each entry n_{ij} denot the number of objects in common between X_i and Y_j : $n_{ij} = |X_i \cap Y_j|$.

Adjusted Rand Index

- 根据相依表中的值计算ARI

$$\overbrace{ARI}^{\text{Adjusted Index}} = \frac{\overbrace{\sum_{ij}\binom{n_{ij}}{2}}^{\text{Index}} - \overbrace{[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}^{\text{Expected Index}}}{\underbrace{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}]}_{\text{Max Index}} - \underbrace{[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}_{\text{Expected Index}}}$$

where n_{ij}, a_i, b_j are values from the contingency table.

```
metrics.adjusted_rand_score(labels_true,labels_pred)
```

互信息评价法

- 互信息的定义，用来描述两个随机
- 变量表述出的信息的相似程度

$$MI(X,Y)=\sum_{i=1}^{r}\sum_{j=1}^{s}P(i,j)\log\frac{P(i,j)}{P(i)P(j$$

- AdjustedMI

$$AMI(X,Y)=\frac{MI(X,Y)-E[MI(X,Y)]}{\max\{H(X),H(Y)\}-E[MI(X$$

```
metrics.adjusted_mutual_info_score(labels_true, labels_pred)
```

非给定标签情况下的评价指标

- 轮廓系数：
- 1.计算同簇内每个样本到其他样本的距离，求平均a(i)
- 2.计算同簇内每个样本到其他簇样本的平均距离，若干个簇中最小的距离 b(i)

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

```
metrics.silhouette_score(X, labels, metric='euclidean')
```

Calinski-Harabaz Index

- 计算公式

$$s(k) = \frac{tr(B_k)}{tr(W_k)} \frac{m-k}{k-1}$$

- m为样本数，k为类别数 Bk为类别之间的协方差矩阵，Wk为类别内的协方差矩阵
- W越小越好，B越大越好

层次聚类

- 解决了k-means k值选择和初始中心点选择的问题
- 分为 分裂法 和 凝聚法

分裂法

- 1.将所有样本归为一个簇
  - While 不满足条件（k个数或距离阈值）：
    - 在同一个簇C中计算样本间距离，选最远的距离的两个样本ab（终止条件检测）
    - 将样本a，b划入c1 c2（终止条件检测）
    - 计算原簇c中样本离谁近，划入谁

例子

dis	p0	p1	p2	p3	p4	p5
po	0	1	sqrt(2)	sqrt(18)	5	sqrt(41)
p1	1	0	1	sqrt(13)	sqrt(18)	sqrt(32)
p2	sqrt(2)	1	0	sqrt(8)	sqrt(13)	5
p3	sqrt(18)	sqrt(13)	sqrt(8)	0	1	sqrt(5)
p4	5	sqrt(18)	sqrt(13)	1	0	sqrt(2)
p5	sqrt(41)	sqrt(32)	5	sqrt(5)	sqrt(2)	0

凝聚法

- 1.将所有点看做一个独立的簇
  - While 不满足条件：
    - 计算两两簇之间的距离，找到最小距离的簇C1 和C2（多种计算方式）
    - 合并C1 C2
  - 合并C1 C2的方式：
    - 1. 两个簇间距离最小的样本距离
    - 2. 两个簇间最远的两个点的距离
    - 3. 两个簇之间两两求距离的平局值
    - 4. 两个粗之间两两求距离的中位数
    - 5. 求每个集合的中心点，用中心点的距离代表簇的距离

层次聚类对比k-means

K-means这种扁平聚类产出一个聚类结果(都是独立的)

层次聚类能够根据你的聚类程度不同，有不同的结果

K-means需要指定聚类个数K，层次聚类不用

K-means比层次聚类要快一些(通常说来)

K-means用的多，可以用K-Median

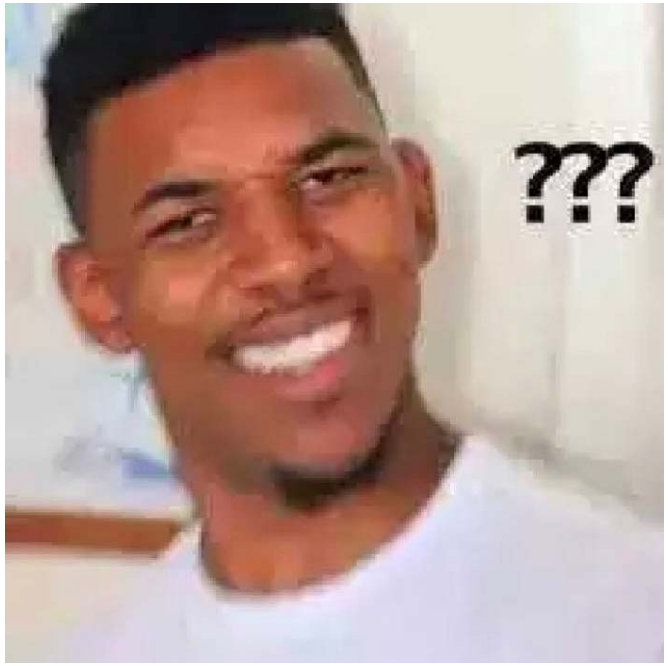
密度聚类-DBSCAN

- DBSCAN（Density-Based Spatial Clustering of Applications with Noise)

- 一个基于密度聚类的算法。与层次聚类不同，它将簇定义为密度相连的点的最大集合，能够把具有高密度的区域划分为簇，并可有效地对抗噪声

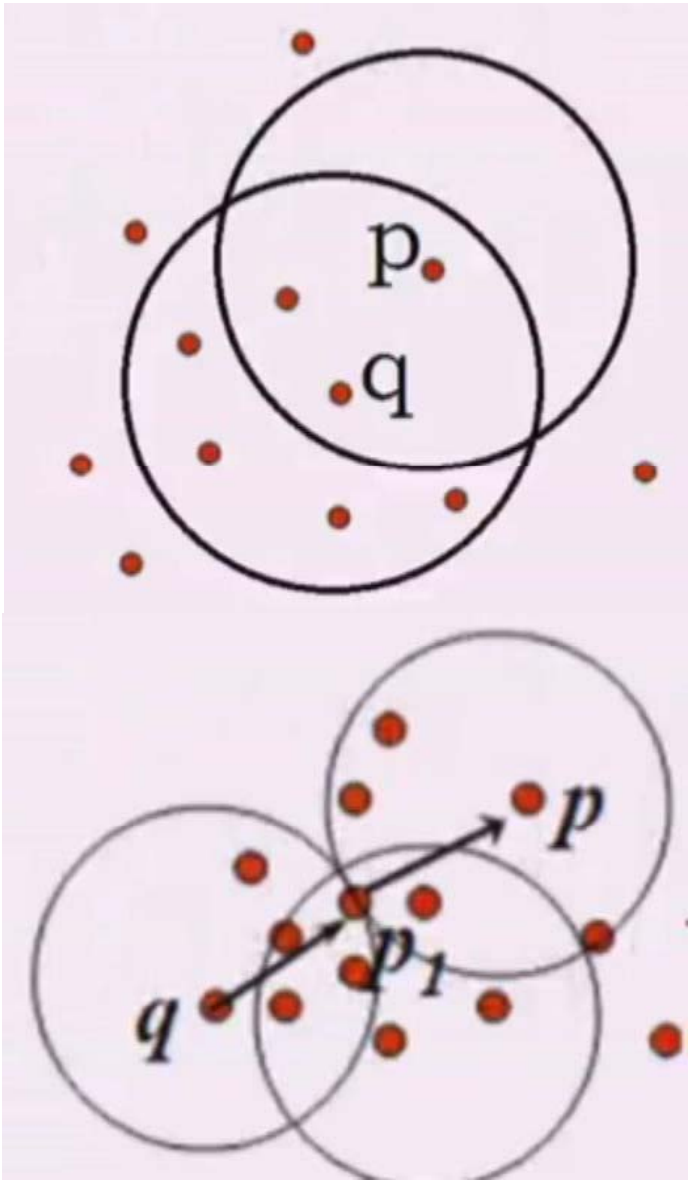
密度聚类-DBSCAN

- 将簇定义为密度相连的点的最大集合



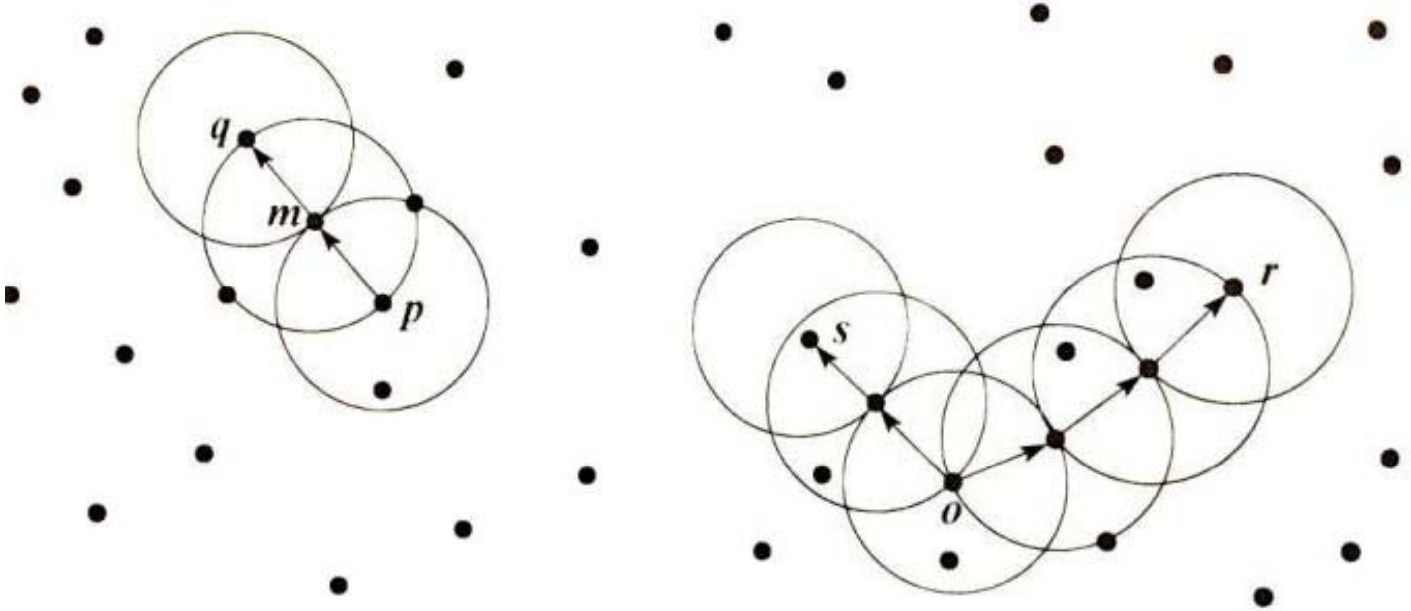
直接密度可达

- 对象的e邻域，给定对象在半径e内的区域
- 核心对象，给定一个数目m，如果对象的e邻域中有至少m个对象，该对象为核心对象
- 直接密度可达：给定一个对象集合D，如果p在q的e邻域内，而q是一个核心对象，p从q出发是直接密度可达的



密度可达

- 密度可达：如果存在一个对象链p1p2...pn ,令p1=p,pn=q,pi+1是关于e和m直接密度可达的，则对象p是从对象q关于e和m密度可达的
- 密度相连：如果集合D中存在一个对象o 使o->p 密度可达，o->q 密度可达　那么p和q就是关于e和m密度相连的



DB-SCAN算法

- DB-SCAN 通过检查数据集中每个对象的e邻域来寻找聚类
- 如果一个点p的e邻域中多余m个对象，则创建一个p为核心的新簇，依据p来反复寻找密度相连的集合（有可能合并原有已经生成的簇）当没有任何新的点可以被添加到簇中的时候，寻找结束

密度最大值聚类

- 密度最大值聚类是另一种基于密度的聚类算法，可以识别各种形状的类簇，还能够确定聚类数目
- 定义：局部密度

$$\rho_i = \sum_j \chi(d_{ij} - d_c), \quad \text{其中，} \chi(x) = \begin{cases} 1 & x < 0 \\ 0 & otherwise \end{cases}$$

- D_c是一个截断距离, ρi 即到对象i的距离小于d_c 的对象的个数，即：ρi = 任何一个点以d_c为半径的圆内的样本点的数量，Dc的设定经验，使平均每个点的邻居数目是所有点的1%-2%

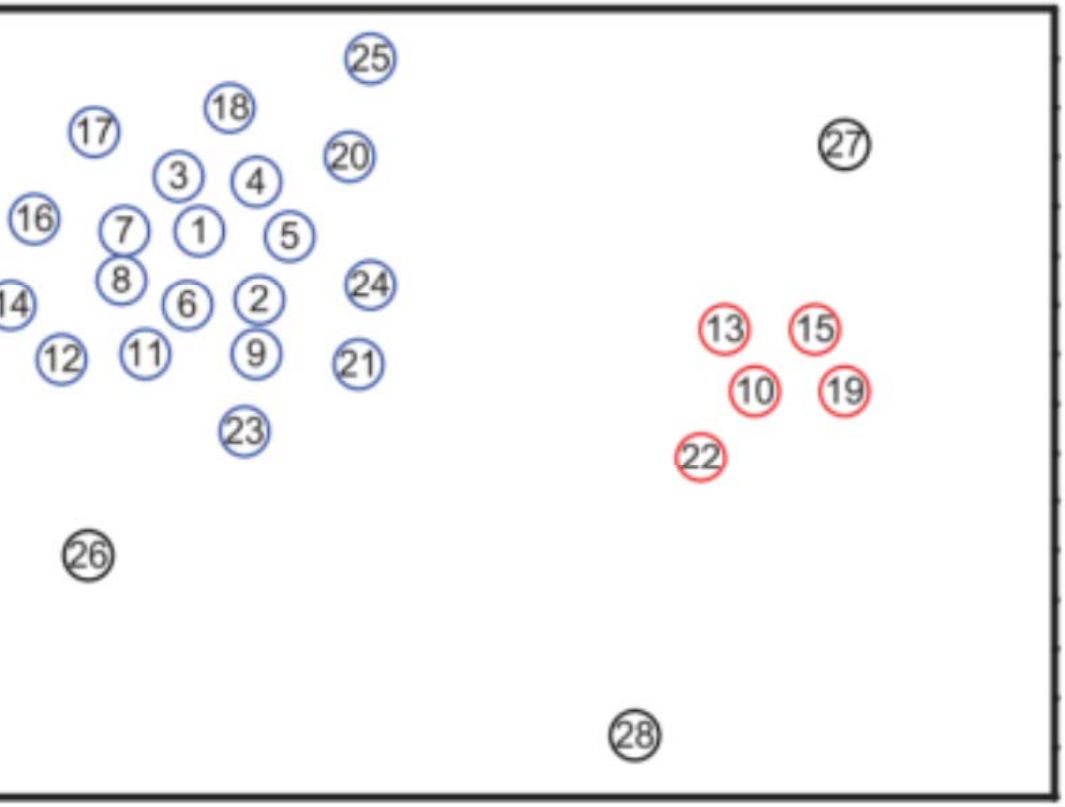
密度最大值聚类

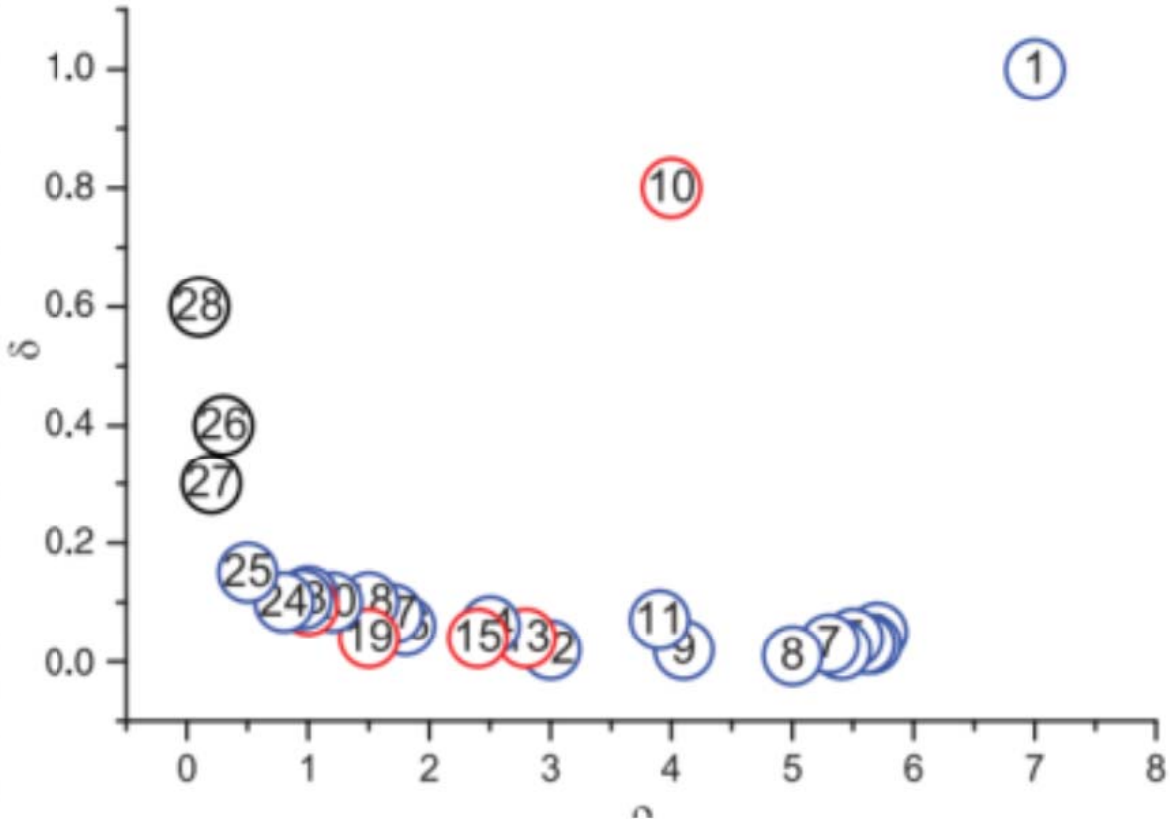
- 高局部密度点距离：

$$\delta_i = \min_{j:\rho_j > \rho_i} \left(d_{ij} \right)$$

- 在密度高于对象i的所有对象中，离i最近的点到i得距离
- 对于密度最大的对象，设置高局部密度距离为max（dij)

密度最大值聚类


25
18
17
20
27
3
4
16
7
1
5
8
24
14
6
2
13
15
12
11
9
21
10
19
23
22
26
28


1.0
0.8
0.6
0.4
0.2
0.0
δ
1
10
28
26
27
25
11
9
8
0
1
2
3
4
5
6
7
8

簇中心的识别

- ρ大δ大 --- 簇中心
- ρ小δ大 ---- 异常值
- 确定簇中心后， 按照距离找组织

谱聚类

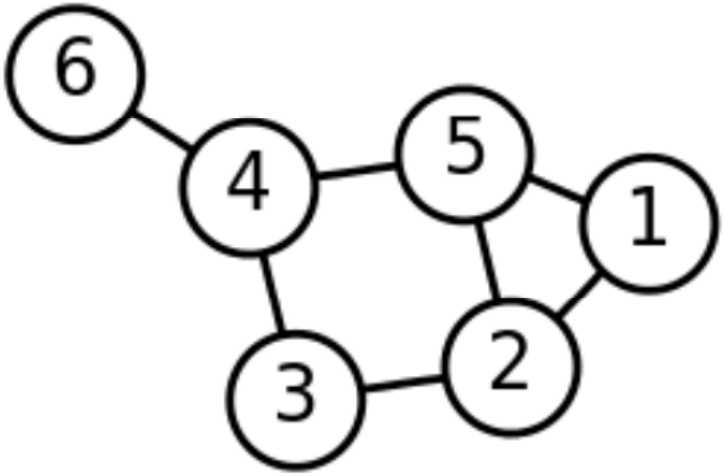
谱聚类的特点

- 1.对数据的结构没有假设（适应性广）
- 2.经过特殊的构图处理后计算很快
- 3.不会像kmeans一样将一些离散的小簇聚在一起

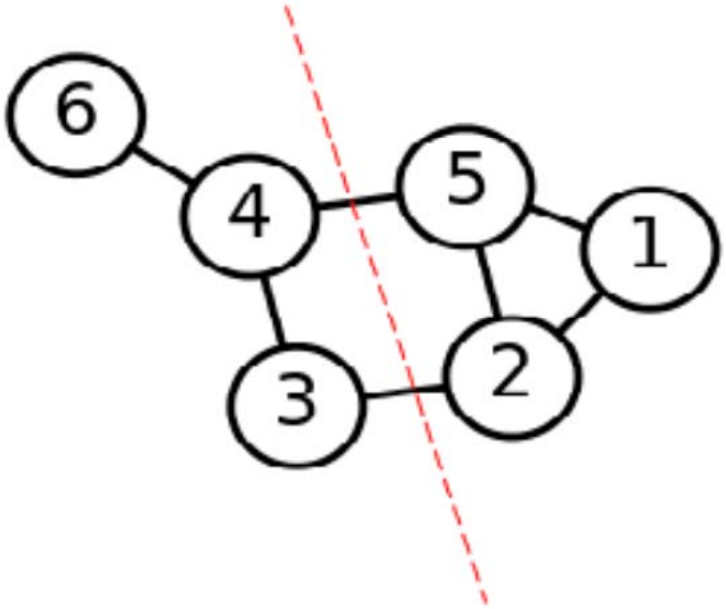
- 1.对于不同的构图方式比较敏感
- 2.对于超参数设置比较敏感

谱聚类整体思路

- 1.构图



2. 切图



构图

- 根据训练集，计算相似度矩阵
- 根据相似度矩阵采用某种构图方法计算w权重矩阵
- 根据w权重矩阵计算D矩阵和拉普拉斯矩阵

相似度矩阵

- 根据n个样本彼此之间的举例（可以选择欧氏距离或者高斯距离）生成一个NxN的相似度矩阵

- 欧式距离： $s_{i,j} = \|x_i - x_j\|^2$

- 高斯距离： $s_{i,j} = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$

- 得到了 S 矩阵

根据构图方式计算W矩阵

- 常见构图方式：
    1. ε-neighborhood
    2. k-nearest neighborhood
    3. fully connected

根据构图方式计算W矩阵

- ε-neighborhood
  - 选取一个阈值ε
- 根据规则： $W_{i,j} = \begin{cases} 0, & if \quad s_{i,j} > \varepsilon \\ \varepsilon, & if \quad s_{i,j} \leq \varepsilon \end{cases}$ 生成W矩阵

根据构图方式计算W矩阵(邻接矩阵）

- k-nearest neighborhood
- 遍历所有样本，找到每个样本的k近邻
  - 构图：2中思路
    - 1. i在j的k近邻中或j在i的k近邻中时 保留sij作为wij 其余取0

$$W_{i,j} = W_{j,i} = \begin{cases} 0, & if \quad x_i \notin KNN(X_j) \quad and \quad x_j \notin KNN(X_i) \\ e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}, & if \quad x_i \in KNN(X_j) \quad or \quad x_j \in KNN(X_i) \end{cases}$$

    - 2. i在j的k近邻中且j在i的k近邻中时 保留sij作为wij 其余取0

$$W_{i,j} = W_{j,i} = \begin{cases} 0, & if \quad x_i \notin KNN(X_j) \quad or \quad x_j \notin KNN(X_i) \\ e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}, & if \quad x_i \in KNN(X_j) \quad and \quad x_j \in KNN(X_i) \end{cases}$$

根据构图方式计算W矩阵

- fully connected

- 直接保留相似度矩阵作为权重矩阵

计算D矩阵和拉普拉斯矩阵

$$D_{i,j} = \begin{cases} 0, & if \quad i \neq j \\ \sum_j w_{i,j}, & if \quad i = j \end{cases}$$

$\sum_j w_{i,j}$ 表示某个点与其他点相连的所有边的权重之和，D矩阵是个NxN对角矩阵

拉普拉斯矩阵：

$$L = D - W$$

切图

- 对于原始图的任意两个子图 A B 满足（ A∩B=∅ ）
- 定义切图权重为：W(A,B) = ∑i∈A,j∈B wij

- 衡量最终切图结果：
- 假设原始图V切为了k个子图（A1,A2,...AK ）有

$$A_1 \cup A_2 \cup \cdots \cup A_k = V且A_1 \cap A_2 \cap \cdots \cap A_k = \varnothing,$$

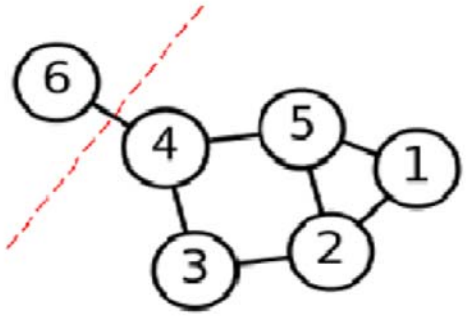
定义 $cut(A_1, A_2, \cdots, A_k) = \frac{1}{2}\sum_{i}^{k} W(A_i, \bar{A}_i)$ 为该种切法的切边权重和

切图目的

- 切图的目标：
  - 每个子图内部连边的权重平均都较大
  - 每个子图间则尽量没有边相连，或者连边的权重很低

$$min \quad cut(A_1, A_2, \cdots, A_k)$$

- 问题：



RatioCut

- 一种思路：子图内部的点越多越好，这样就能排除掉刚才的情况

$$Ratiocut(A_1, A_2, \cdots, A_k) = \frac{1}{2}\sum_{i}^{k}\frac{W(A_i, \bar{A}_i)}{|A_i|}$$

- 问题变为了： $min \quad Ratiocut(A_1, A_2, \cdots, A_k)$

- 另一种思路： 子图内部所有的边的权重和越大越好，也能排除掉刚才的情况

$$Ncut(A_1, A_2, \cdots, A_k) = \frac{1}{2}\sum_{i}^{k}\frac{W(A_i, \bar{A}_i)}{vol(A_i)}$$

分图方法

- RatioCut
  - 对于L矩阵取最小的k1个特征值对应的特征向量（每个向量的形状是是1*n）
  - 将k1个列向量拼成一个N行k1列的矩阵H
  - 对这个H矩阵按行做标准化

$$h_{ij}^{*} = \frac{h_{ij}}{(\sum_{t=1}^{k} h_{it}^{2})^{1/2}}$$

  - 设定超参数k2，对标准化后的H矩阵进行k-means聚类，得到的结果便是按照RatioCut标准划分出来的子集