对外经济贸易大学
UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

聚类

信息学院-黄浩

2022年3月14日星期一

聚类方法概述

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类
- 谱聚类

聚类概念

- 聚类分析的输入可以用一组有序对(X, s) 或(X, d)表示，这里X表示一组样本，s和d分别是度量样本间相似度或相异度（距离）的标准。聚类系统的输出是一个分区若$C=\{C_1, C_2,\ldots, C_k\}$，其中$C_i(i=1,2\ldots.,K)$是$X$的子集，且满足：
  - $C_1\cup C_2\cup,\ldots,\cup C_k=X$
  - $C_1\cap C_2=\emptyset, i\neq j$
- C中的成员$C_1, C_2,\ldots, C_k$叫做类或簇（Cluster），每一个类或簇都是通过一些特征描述的，通常有如下几种表示方式：
  - 通过它们的中心或类中关系远的（边界）点表示空间的一类点。
  - 使用聚类树中的结点图形化地表示一个类。
  - 使用样本属性的逻辑表达式表示类。
- 用中心表示一个类是最常见的方式，当类是紧密的或各向同性时用这种方法非常好，然而，当类是伸长的或向各向分布异性时，这种方式就不能正确地表示它们了。

聚类分析的目标

- 聚类分析的目标就是形成的数据簇，并且满足下面两个条件：
  - 一个簇内的数据尽量相似（high intra-class similarity）；
  - 不同簇的数据尽量不相似（low inter-class similarity）。
- 衡量一个聚类分析算法质量，依靠：
  - 相似度测量机制是否合适。
  - 是否能发现数据背后潜在的、手工难以发现的类知识。

常见的距离函数

- 按照距离公理，在定义距离测度时需要满足距离公理的四个条件自相似性、最小性、对称性以及三角不等性。常用的距离函数有如下几种：
  - 明可夫斯基距离（Minkowski）
  - 二次型距离（Quadratic）
  - 余弦距离
  - 二元特征样本的距离度量

明可夫斯基（Minkowski）距离

- 假定x和y是相应的特征，n是特征的维数。 x和y的明可夫斯基距离度量的形式如下：

$$d(x, y) = \left[\sum_{i=1}^{n} |x_i - y_i|^r\right]^{1/r}$$

- 当取不同的值时，上述距离度量公式演化为一些特殊的距离测度：
  - 当r=1时，明可夫斯基距离演变为绝对值距离：

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

- 当r=2时，明可夫斯基距离演变为欧氏距离：

$$d(x,y) = \left[\sum_{i=1}^{n} |x_i - y_i|^2\right]^{1/2}$$

二次型距离

- 二次型距离测度的形式如下：

$$d(x,y)=\left((x-y)^T A(x-y)\right)^{1/2}$$

其中A是非负定矩阵。

- 当取不同的值时，上述距离度量公式演化为一些特殊的距离测度：
  - 当A为单位矩阵时，二次型距离演变为欧氏距离。
  - 当A为对角阵时，二次型距离演变为加权欧氏距离：

$$d(x,y)=\left[\sum_{i=1}^{n} a_{ii}\left|x_i-y_i\right|^2\right]^{1/2}$$

- 当为协方差矩阵时，二次型距离演变为马氏距离。

余弦距离

- 余弦距离的度量形式如下：

$$d(x,y)=\frac{\sum_{i=1}^{n}x_i y_i}{\sqrt{\sum_{i=1}^{n}x_i^2\sum_{i=1}^{n}y_i^2}}$$

二元属性的邻近性度量

- 对象*i* 和对象*j* 的频数表

Object *j*

Object *i*	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

- 对称的二元相异性

$$d(i,j)=\frac{r+s}{q+r+s+t}$$

- 非对称的二元相异性(t被认为不重要，例如：病理化验呈阴性)

$$d(i,j)=\frac{r+s}{q+r+s}$$

二元属性的邻近性度量

- Jaccard系数(非对称的二元相似性):

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

二元属性的相异性（例子）

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Name(姓名)和Gender (性别)是对称属性
- 其他属性是非对称二元属性，假设只针对非对称二元属性进行相异性计算

$$d(i,j) = \frac{r+s}{q+r+s}$$

- 值 Y 和 P 是 1, 值 N 是 0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

类间距离

- 距离函数都是关于两个样本的距离刻画，然而在聚类应用中，最基本的方法是计算类间的距离。
- 设有两个类C_a和C_b，它们分别有m和h个元素，它们的中心分别为γ_a和γ_b。设元素$x \in C_a$，$y \in C_b$，这两个元素间的距离通常通过类间距离来刻画，记为$D(C_a, C_b)$。
- 类间距离的度量主要有：
  - 最短距离法：定义两个类中最靠近的两个元素间的距离为类间距离。
  - 最长距离法：定义两个类中最远的两个元素间的距离为类间距离。
  - 中心法：定义两类的两个中心间的距离为类间距离。
  - 类平均法：它计算两个类中任意两个元素间的距离，并且综合他们为类间距离：

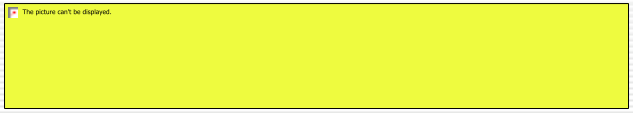
$$D_G(C_a, C_b) = \frac{1}{mh} \sum_{x \in C_a} \sum_{y \in C_b} d(x, y)$$

  - 离差平方和。

中心法

- 中心法涉及到类的中心的概念。假如C_i是一个聚类，x是C_i内的一个数据点，那么类中心定义如下：

其中n_i是第i个聚类中的点数。因此，两个类C_a和C_b的类间距离为：



其中γ_a和γ_b是类C_a和C_b的中心点，d是某种形式的距离公式。

离差平方和

- 离差平方和用到了类直径的概念：
  - 类的直径反映了类中各元素间的差异，可定义为类中各元素至类中心的欧氏距离之和，其量纲为距离的平方：

$$r_a = \sum_{i=1}^{m} (x_i - \overline{x_a})^T (x_i - \overline{x_b})$$

- 根据上式得到两类C_a和C_b的直径分别为γ_a和γ_b，类$C_{a+b} = C_a \cup C_b$的直径为γ_{a+b}，则可定义类间距离的平方为：

$$D_W^2(C_a, C_b) = r_{a+b} - r_a - r_b$$

划分聚类方法

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类
- 谱聚类

划分聚类算法的主要思想

- **定义**：给定一个有n个对象的数据集，划分聚类技术将构造数据k个划分，每一个划分就代表一个簇，$k \leq n$。也就是说，它将数据划分为k个簇，而且这k个划分满足下列条件：
  - 每一个簇至少包含一个对象。
  - 每一个对象属于且仅属于一个簇。
- 对于给定的k，算法首先给出一个初始的划分方法，以后通过反复迭代的方法改变划分，使得每一次改进之后的划分方案都较前一次更好。

聚类设计的评价函数

- 一种直接方法就是观察聚类的类内差异（Within cluster variation）和类间差异(Between cluster variation）。
  - 类内差异：衡量聚类的紧凑性，类内差异可以用特定的距离函数来定义，例如，

$$w(C)=\sum_{i=1}^{k} w(C_i)=\sum_{i=1}^{k}\sum_{x\in C_i} d(x,\overline{x_i})^2$$

  - 类间差异：衡量不同聚类之间的距离，类间差异定义为聚类中心间的距离，例如，

$$b(C)=\sum_{1\le j<i\le k} d(\overline{x_j},\overline{x_i})^2$$

- 聚类的总体质量可被定义为w(c)和b(c)的一个单调组合，比如w(c) / b(c) 。

k-means算法

算法 *k*-means算法

输入：簇的数目*k*和包含*n*个对象的数据库。

输出：*k*个簇，使平方误差准则最小。

(1)assign initial value for means; /*任意选择*k*个对象作为初始的簇中心；*/

(2) REPEAT

(3) FOR *j*=1 to *n* DO assign each X_j to the closest clusters;

(4) FOR *i*=1 to *k* DO $\overline{x_i} = |C_i| \sum_{x \in C_i} x$ /*更新簇平均值*/

(5) Compute $E = \sum_{i=1}^{k} \sum_{x \in C_i} \left| x - \overline{x_i} \right|^2$ /*计算准则函数*E*/

(6) UNTIL *E*不再明显地发生变化。

- 算法首先随机地选择*k*个对象，每个对象初始地代表了一个簇的平均值或中心。对剩余的每个对象根据其与各个簇中心的距离，将它赋给最近的簇。然后重新计算每个簇的平均值。这个过程不断重复，直到准则函数收敛。
- 准则函数试图使生成的结果簇尽可能地紧凑和独立。

k-means例子

样本数据

序号	属性 1	属性 2
1	1	1
2	2	1
3	1	2
4	2	2
5	4	3
6	5	3
7	4	4
8	5	4

根据所给的数据通过对其实施*k*-means (设*n*=8，*k*=2)，其主要执行执行步骤：

第一次迭代：假定随机选择的两个对象，如序号1和序号3当作初始点，分别找到离两点最近的对象，并产生两个簇{1，2}和{3，4，5，6，7，8}。

对于产生的簇分别计算平均值，得到平均值点。

对于{1，2}，平均值点为（1.5，1）（这里的平均值是简单的相加出2）；

对于{3，4，5，6，7，8}，平均值点为（3.5，3）。

第二次迭代：通过平均值调整对象的所在的簇，重新聚类，即将所有点按离平均值点（1.5，1）、（3.5，1）最近的原则重新分配。得到两个新的簇：{1，2，3，4}和{5，6，7，8}。重新计算簇平均值点，得到新的平均值点为（1.5，1.5）和（4.5，3.5）。

第三次迭代：将所有点按离平均值点（1.5，1.5）和（4.5，3.5）最近的原则重新分配，调整对象，簇仍然为{1，2，3，4}和{5，6，7，8}，发现没有出现重新分配，而且准则函数收敛，程序结束。

迭代次数	平均值（簇1）	平均值（簇2）	产生的新簇	新平均值（簇1）	新平均值（簇2）
1	（1，1）	（1，2）	{1，2}，{3，4，5，6，7，8}	（1.5，1）	（3.5，3）
2	（1.5，1）	（3.5，3）	{1，2，3，4}，{5，6，7，8}	（1.5，1.5）	（4.5，3.5）
3	（1.5，1.5）	（4.5，3.5）	{1，2，3，4}，{5，6，7，8}	（1.5，1.5）	（4.5，3.5）

k-means算法的性能分析

- 主要优点：
  - 是解决聚类问题的一种经典算法，简单、快速。
  - 对处理大数据集，该算法是相对可伸缩和高效率的。
  - 当结果簇是密集的，它的效果较好。
- 主要缺点
  - 在簇的平均值被定义的情况下才能使用，可能不适用于某些应用。
  - 必须事先给出*k*（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
  - 不适合于发现非凸面形状的簇或者大小差别很大的簇。而且，它对于“噪声”和孤立点数据是敏感的。

PAM算法基本思想

- PAM作为最早提出的*k*-中心点算法之一，它选用簇中位置最中心的对象作为代表对象，试图对*n*个对象给出*k*个划分。
- 代表对象也被称为是中心点，其他对象则被称为非代表对象。
- 最初随机选择*k*个对象作为中心点，该算法反复地用非代表对象来代替代表对象，试图找出更好的中心点，以改进聚类的质量。
- 在每次迭代中，所有可能的对象对被分析，每个对中的一个对象是中心点，而另一个是非代表对象。
- 对可能的各种组合，估算聚类结果的质量。一个对象O_j被可以产生最大平方-误差值减少的对象代替。在一次迭代中产生的最佳对象集合成为下次迭代的中心点。

PAM算法基本思想(续)

- 为了判定一个非代表对象O_h是否是当前一个代表对象O_i的好的替代，对于每一个非中心点对象O_j，下面的四种情况被考虑：
  - **第一种情况：**O_j当前隶属于中心点对象O_i。如果O_i被O_h所代替作为中心点，且O_j离一个O_m最近，$i \neq m$，那么O_j被重新分配给O_m。
  - **第二种情况：**O_j当前隶属于中心点对象O_i。如果O_i被O_h代替作为一个中心点，且O_j离O_h最近，那么O_j被重新分配给O_h。
  - **第三种情况：**O_j当前隶属于中心点O_m，$m \neq i$。如果O_i被O_h代替作为一个中心点，而O_j依然离O_m最近，那么对象的隶属不发生变化。
  - **第四种情况：**O_j当前隶属于中心点O_m，$m \neq i$。如果O_i被O_h代替作为一个中心点，且O_j离O_h最近，那么O_i被重新分配给O_h。

PAM算法基本思想(续)

- 每当重新分配发生时，平方-误差E所产生的差别对代价函数有影响。因此，如果一个当前的中心点对象被非中心点对象所代替，代价函数计算平方-误差值所产生的差别。替换的总代价是所有非中心点对象所产生的代价之和。
  - 如果总代价是负的，那么实际的平方-误差将会减小，O_i可以被O_h替代。
  - 如果总代价是正的，则当前的中心点O_i被认为是可接受的，在本次迭代中没有变化。

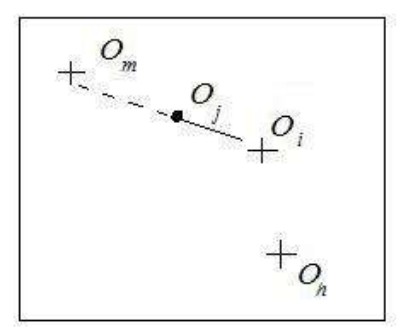
总代价定义如下：

$$TC_{ih} = \sum_{j=1}^{n} C_{jih}$$

其中，C_{jih}表示O_j在O_i被O_h代替后产生的代价。下面我们将介绍上面所述的四种情况中代价函数的计算公式，其中所引用的符号有：O_i和O_m是两个原中心点，O_h将替换O_i作为新的中心点。

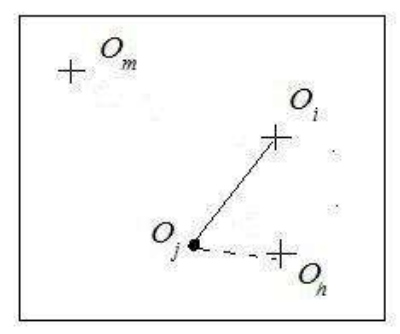
PAM算法代价函数的四种情况

第一种情况



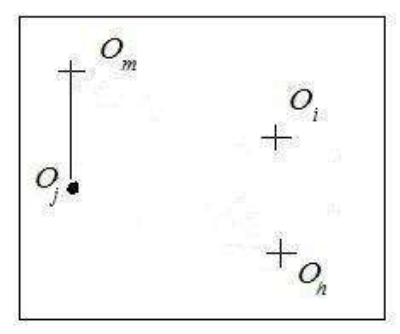
O_j被重新分配给O_m,
$C_{jih} = d(j, m) - d(j, i)$

第二种情况



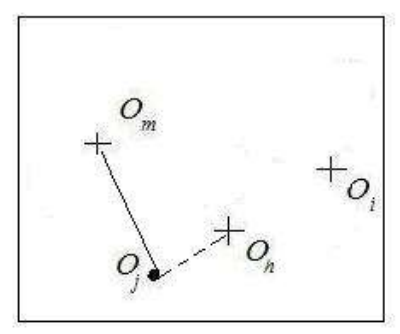
O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, i)$

第三种情况



O_j的隶属不发生变化，
$C_{jih} = 0$

第四种情况



O_j被重新分配给O_h,
$C_{jih} = d(j, h) - d(j, m)$

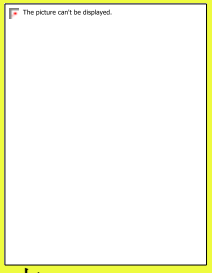
PAM算法基本思想(续)

算法 PAM（*k*-中心点算法）
输入：簇的数目*k*和包含*n*个对象的数据库。
输出：*k*个簇，使得所有对象与其最近中心点的相异度总和最小。
（1） 任意选择*k*个对象作为初始的簇中心点；
（2） REPEAT
（3） 指派每个剩余的对象给离它最近的中心点所代表的簇；
（4） REPEAT
（5） 选择一个未被选择的中心点O_j；
（6） REPEAT
（7） 选择一个未被选择过的非中心点对象O_h；
（8） 计算用O_h代替O_j的总代价并记录在*S*中；
（9） UNTIL 所有的非中心点都被选择过；
（10） UNTIL 所有的中心点都被选择过；
（11） IF 在*S*中的所有非中心点代替所有中心点后的计算出的总代价有小于0的存在 THEN 找出*S*中的用非中心点替代中心点后代价最小的一个，并用该非中心点替代对应的中心点，形成一个新的*k*个中心点的集合；
（12）UNTIL 没有再发生簇的重新分配，即所有的*S*都大于0.

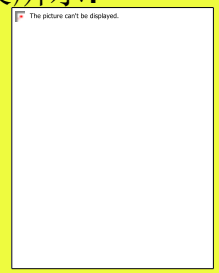
PAM算法基本思想(续)

假如空间中的五个点｛A、B、C、D、E｝如图1所示，各点之间的距离关系如表1所示，根据所给的数据对其运行PAM算法实现划分聚类（设k=2）。 样本点间距离如下表所示:

样本点	A	B	C	D	E
A	0	1	2	2	3
B	1	0	2	4	3
C	2	2	0	1	5
D	2	4	1	0	3
E	3	3	5	3	0



样本点



起始中心点为A,B

第一步 建立阶段：假如从5个对象中随机抽取的2个中心点为{A，B},则样本被划分为{A、C、D}和{B、E}，如图所示。

第二步 交换阶段：假定中心点A、B分别被非中心点}和{C、D、E}替换，根据PAM算法需要计算下列代价TC_{AC}、 TC_{AD}、 TC_{AE}、TC_{BC}、TC_{BD}、 TC_{BE}。

我们以$TCAC$为例说明计算过程：

a) 当A被C替换以后，A不再是一个中心点，因为A离B比A离C近，A被分配到B中心点代表的簇，$C_{AAC}=d(A,B)-d(A,A)=1$。

b) B是一个中心点，当A被C替换以后，B不受影响，$C_{BAC}=0$。

c) C原先属于A中心点所在的簇，当A被C替换以后，C是新中心点，符合PAM算法代价函数的第二种情况$C_{CAC}=d(C,C)-d(C,A)=0-2=-2$。

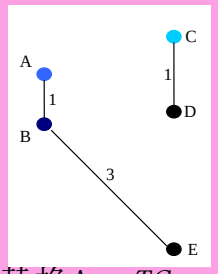
d) D原先属于A中心点所在的簇，当A被C替换以后，离D最近的中心点是C，根据PAM算法代价函数的第二种情况$C_{DAC}=d(D,C)-d(D,A)=1-2=-1$。

e) E原先属于B中心点所在的簇，当A被C替换以后，离E最近的中心仍然是 B，根据PAM算法代价函数的第三种情况$C_{EAC}=0$。

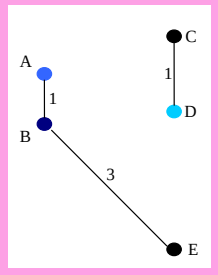
因此，$TC_{AC}=CA_{AC}+CB_{AC}+CB_{AC}+CD_{AC}+CE_{AC}=1+0-2-1+0=-2$。

PAM算法基本思想(续)

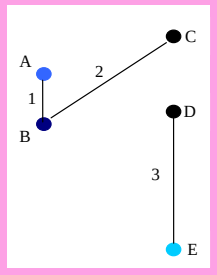
在上述代价计算完毕后，我们要选取一个最小的代价，显然有多种替换可以选择，我们选择第一个最小代价的替换（也就是C替换A），根据图5-4（a）所示，样本点被划分为{ B、A、E}和{C、D}两个簇。图5-4（b）和图5-4（c）分别表示了D替换A，E替换A的情况和相应的代价



(a) C替换A， TC_{AC}=-2



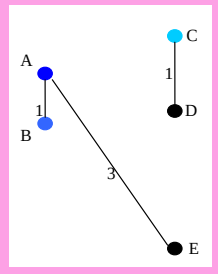
(b) D替换A， TC_{AD}=-2



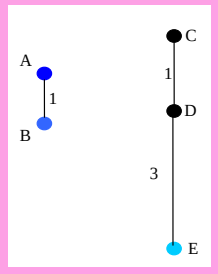
(c) E替换A， TC_{AE}=-1

图5-4　替换中心点A

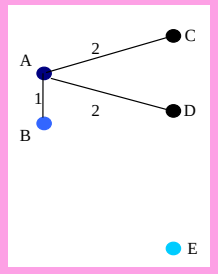
图5-5（a）、（b）、（c）分别表示了用C、D、E替换B的情况和相应的代价。



(a)　C替换B， TC_{BC}=-2



(b) D替换B， TC_{BD}=-2



(c) E替换B， TC_{BE}=-2

图5-5　替换中心点B

通过上述计算，已经完成了PAM算法的第一次迭代。在下一迭代中，将用其他的非中心点{A、D、E}替换中心点{B、C}，找出具有最小代价的替换。一直重复上述过程，直到代价不再减小为止。

层次聚类方法

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类
- 谱聚类

层次聚类方法概述

- 层次聚类方法对给定的数据集进行层次的分解，直到某种条件满足为止。具体又可分为：
  - 凝聚的层次聚类：一种自底向上的策略，首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到某个终结条件被满足。
  - 分裂的层次聚类：采用自顶向下的策略，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到达到了某个终结条件。
- 层次凝聚的代表是AGNES算法。层次分裂的代表是DIANA算法。

AGNES算法

- AGNES (AGglomerative NESting)算法最初将每个对象作为一个簇，然后这些簇根据某些准则被一步步地合并。两个簇间的相似度由这两个不同簇中距离最近的数据点对的相似度来确定。聚类的合并过程反复进行直到所有的对象最终满足簇数目。

算法 AGNES（自底向上凝聚算法）

输入：包含n个对象的数据库，终止条件簇的数目k。

输出：k个簇，达到终止条件规定簇数目。

(1) 将每个对象当成一个初始簇；

(2) REPEAT

(3) 根据两个簇中最近的数据点找到最近的两个簇；

(4) 合并两个簇，生成新的簇的集合；

(5) UNTIL 达到定义的簇的数目；

AGNES算法例子

序号	属性 1	属性 2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

第1步：根据初始簇计算每个簇之间的距离，随机找出距离最小的两个簇，进行合并，最小距离为1，合并后1，2点合并为一个簇。

第2步：，对上一次合并后的簇计算簇间距离，找出距离最近的两个簇进行合并，合并后3，4点成为一簇。

第3步：重复第2步的工作，5，6点成为一簇。

第4步：重复第2步的工作，7，8点成为一簇。

第5步：合并{1，2}，{3，4}成为一个包含四个点的簇。

第6步：合并{5，6}，{7，8}，由于合并后的簇的数目已经达到了用户输入的终止条件程序结束。

步骤	最近的簇距离	最近的两个簇	合并后的新簇
1	1	{1}，{2}	{1，2}，{3}，{4}，{5}，{6}，{7}，{8}
2	1	{3}，{4}	{1，2}，{3，4}，{5}，{6}，{7}，{8}
3	1	{5}，{6}	{1，2}，{3，4}，{5，6}，{7}，{8}
4	1	{7}，{8}	{1，2}，{3，4}，{5，6}，{7，8}
5	1	{1，2}，{3，4}	{1，2，3，4}，{5，6}，{7，8}
6	1	{5，6}，{7，8}	{1，2，3，4}，{5，6，7，8}结束

AGNES性能分析

- AGNES算法比较简单，但经常会遇到合并点选择的困难。假如一旦一组对象被合并，下一步的处理将在新生成的簇上进行。已做处理不能撤消，聚类之间也不能交换对象。如果在某一步没有很好的选择合并的决定，可能会导致低质量的聚类结果。
- 这种聚类方法不具有很好的可伸缩性，因为合并的决定需要检查和估算大量的对象或簇。
- 假定在开始的时候有n个簇，在结束的时候有1个簇，因此在主循环中有n次迭代，在第i次迭代中，我们必须在$n-i+1$个簇中找到最靠近的两个聚类。另外算法必须计算所有对象两两之间的距离，因此这个算法的复杂度为 $O(n^2)$，该算法对于n很大的情况是不适用的。

DIANA算法

- DIANA **(Divisive ANAlysis)**算法是典型的分裂聚类方法。
- 在聚类中，用户能定义希望得到的簇数目作为一个结束条件。同时，它使用下面两种测度方法：
  - 簇的直径：在一个簇中的任意两个数据点的距离中的最大值。
  - 平均相异度（平均距离）：$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x \in C_i} \sum_{y \in C_j} |x - y|$

算法 DIANA（自顶向下分裂算法）
输入：包含n个对象的数据库，终止条件簇的数目k。
输出：k个簇，达到终止条件规定簇数目。
（1）将所有对象整个当成一个初始簇；
（2） FOR （i=1; i≠k; i++) DO BEGIN
（3） 在所有簇中挑出具有最大直径的簇C；
（4） 找出C中与其它点平均相异度最大的一个点p并把p放入splinter group，剩余的放在old party中；
（5）. REPEAT
（6） 在old party里找出到最近的splinter group中的点的距离不大于到old party中最近点的距离的点，并将该点加入splinter group。
（7） UNTIL 没有新的old party的点被分配给splinter group；
（8） splinter group和old party为被选中的簇分裂成的两个簇，与其它簇一起组成新的簇集合。
（9） END.

DIANA算法例子

序号	属性 1	属性 2
1	1	1
2	1	2
3	2	1
4	2	2
5	3	4
6	3	5
7	4	4
8	4	5

第1步，找到具有最大直径的簇，对簇中的每个点计算平均相异度（假定采用是欧式距离）。

1的平均距离：（1+1+1.414+3.6+4.24+4.47+5）/7=2.96

类似地，2的平均距离为2.526；3的平均距离为2.68；4的平均距离为2.18；5的平均距离为2.18；6的平均距离为2.68；7的平均距离为2.526；8的平均距离为2.96。

挑出平均相异度最大的点1放到splinter group中，剩余点在old party中。

第2步，在old party里找出到最近的splinter group中的点的距离不大于到old party中最近的点的距离的点，将该点放入splinter group中，该点是2。

第3步，重复第2步的工作，splinter group中放入点3。

第4步，重复第2步的工作，splinter group中放入点4。

第5步，没有在old party中的点放入了splinter group中且达到终止条件（k-2），程序终止。如果没有到终止条件，因该从分裂好的簇中选一个直径最大的簇继续分裂。

步骤	具有最大直径的簇	splinter group	Old party
1	{1，2，3，4，5，6，7，8}	{1}	{2，3，4，5，6，7，8}
2	{1，2，3，4，5，6，7，8}	{1，2}	{3，4，5，6，7，8}
3	{1，2，3，4，5，6，7，8}	{1，2，3}	{4，5，6，7，8}
4	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8}
5	{1，2，3，4，5，6，7，8}	{1，2，3，4}	{5，6，7，8} 终止

层次聚类方法

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类
- 谱聚类

密度聚类方法

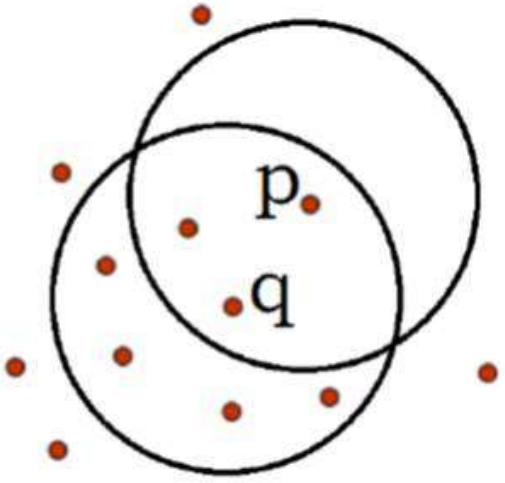
- 密度聚类方法的指导思想是，只要样本点的密度大于某阈值，则将该样本添加到最近的簇中。
- 这类算法能克服基于距离的算法只能发现“类圆形”(凸)的聚类的缺点，可发现任意形状的聚类，且对噪声数据不敏感。但计算密度单元的计算复杂度大，需要建立空间索引来降低计算量。
  - DBSCAN
  - 密度最大值算法

DBSCAN算法

- DBSCAN(Density-Based Spatial Clustering of Applications with Noise)
- 一个比较有代表性的基于密度的聚类算法。与划分和层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在有“噪声”的数据中发现任意形状的聚类。

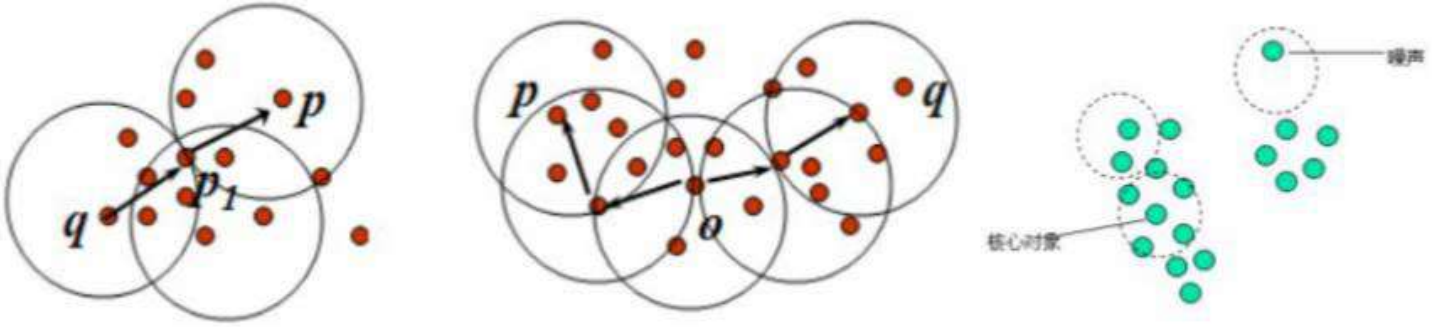
DBSCAN算法的若干概念

- 对象的ε-邻域：给定对象在半径ε内的区域。
- 核心对象：对于给定的数目m，如果一个对象的ε-邻域至少包含m个对象，则称该对象为核心对象。
- 直接密度可达：给定一个对象集合D，如果p是在q的ε-邻域内，而q是一个核心对象，我们说对象p从对象q出发是直接密度可达的。
- 如图ε=1cm，m=5，q是一个核心对象，从对象q出发到对象p是直接密度可达的。



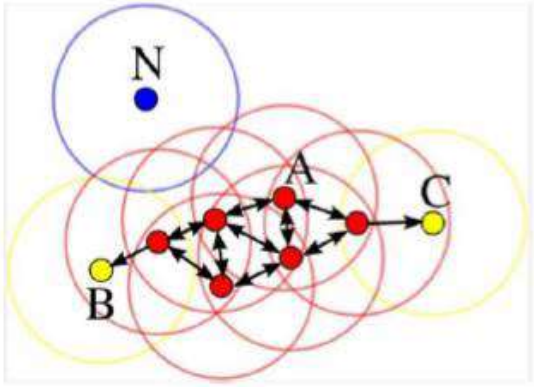
DBSCAN算法的若干概念

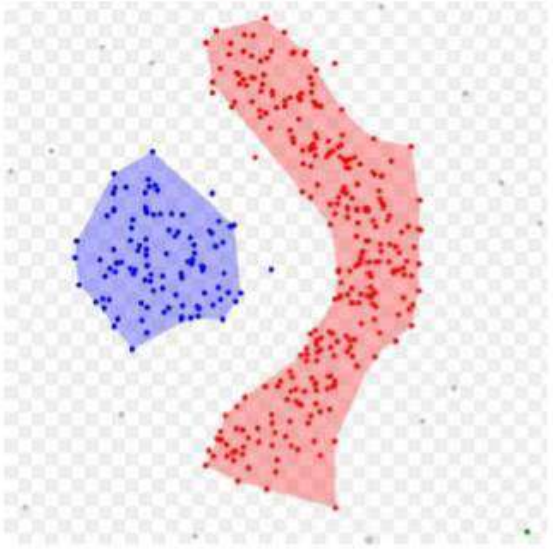
- 密度可达：如果存在一个对象链$p_1p_2...p_n$，$p_1=q$，$p_n=p$，对$p_i \in D$，$(1 \le i \le n)$，p_{i+1}是从p_i关于ε和m直接密度可达的，则对象p是从对象q关于ε和m密度可达的。
- 密度相连：如果对象集合D中存在一个对象o，使得对象p和q是从o关于ε和m密度可达的，那么对象p和q是关于ε和m密度相连的。
- 簇：一个基于密度的簇是最大的密度相连对象的集合。
- 噪声：不包含在任何簇中的对象称为噪声。



DBSCAN算法

- DBSCAN算法流程：
  - 如果一个点p的ε-邻域包含多于m个对象，则创建一个p作为核心对象的新簇；
  - 寻找并合并核心对象直接密度可达的对象；
  - 没有新点可以更新簇时，算法结束。
- 有上述算法可知：
  - 每个簇至少包含一个核心对象；
  - 非核心对象可以是簇的一部分，构成了簇的边缘(edge)；
  - 包含过少对象的簇被认为是噪声。





层次聚类方法

- 聚类方法概述
- 划分聚类方法
- 层次聚类方法
- 密度聚类
- 谱聚类

复习1：实对称阵的特征值是实数

- 首先 $A\bar{x} = \bar{A}\bar{x} = \overline{Ax} = \overline{\lambda x} = \bar{\lambda}\bar{x}$
- 因为 $\bar{x}^T(Ax) = \bar{x}^T(Ax) = \bar{x}^T\lambda x = \lambda\bar{x}^T x$

  $\bar{x}^T(Ax) = (\bar{x}^T A^T)x = (A\bar{x})^T x = (\bar{\lambda}\bar{x})^T x = \bar{\lambda}\bar{x}^T x$
- 从而 $\lambda\bar{x}^T x = \bar{\lambda}\bar{x}^T x \Rightarrow (\lambda - \bar{\lambda})\bar{x}^T x = 0$
- 而 $\bar{x}^T x = \sum_{i=1}^{n} \bar{x}_i x_i = \sum_{i=1}^{n} |x_i|^2 \neq 0$
- 所以 $\lambda - \bar{\lambda} = 0 \Rightarrow \lambda = \bar{\lambda}$

实对称阵不同特征值的特征向量正交

- 令实对称矩阵为A，其两个不同的特征值λ1λ2对应的特征向量分别是$\mu_1\mu_2$；
  - $\lambda_1\lambda_2$ $\mu_1\mu_2$都是实数或是实向量。

$$
\begin{cases} A\mu_1 = \lambda_1\mu_1 \\ A\mu_2 = \lambda_2\mu_2 \end{cases} \Rightarrow \mu_1^T A\mu_2 = \mu_1^T \lambda_2\mu_2
$$

$$
\Rightarrow \left(A^T\mu_1\right)^T\mu_2 = \lambda_2\mu_1^T\mu_2 \Rightarrow \left(A\mu_1\right)^T\mu_2 = \lambda_2\mu_1^T\mu_2
$$

$$
\Rightarrow \left(\lambda_1\mu_1\right)^T\mu_2 = \lambda_2\mu_1^T\mu_2
$$

$$
\Rightarrow \lambda_1\mu_1^T\mu_2 = \lambda_2\mu_1^T\mu_2
$$

$$
\xrightarrow{\lambda_1 \neq \lambda_2} \mu_1^T\mu_2 = 0
$$

谱和谱聚类

- 方阵作为线性算子，它的所有特征值的全体统称方阵的谱。
  - 方阵的谱半径为最大的特征值
  - 矩阵A的谱半径：(ATA)的最大特征值
- 谱聚类是一种基于图论的聚类方法，通过对样本数据的拉普拉斯矩阵的特征向量进行聚类，从而达到对样本数据聚类的目的。

谱分析的整体过程

- 给定一组数据$X_1, X_2, \ldots X_n$，记任意两个点之间的相似度（“距离”的减函数）为$S_{ij} = <Xi, Xj>$，形成相似度图(similarity graph)：$G=(V,E)$。如果x_i和x_j之间的相似度S_{ij}大于一定的阈值，那么，两个点是连接的，权值记做S_{ij}。
- 接下来，可以用相似度图来解决样本数据的聚类问题：找到图的一个划分，形成若干个组(Group)，使得不同组之间有较低的权值，组内有较高的权值。

若干概念

- 无向图G=(V,E)
- 邻接矩阵 $W = (w_{ij})_{i,j=1,\ldots,n}$
- 顶点的度di　　度矩阵D (对角阵)

$$d_i = \sum_{j=1}^{n} w_{ij}$$

若干概念若干概念

- 子图A的指示向量

$$\mathbb{1}_A = (f_1, \ldots, f_n)' \in \mathbb{R}^n$$
$$f_i = 1 \text{ if } v_i \in A$$
$$f_i = 0 \text{ otherwise}$$

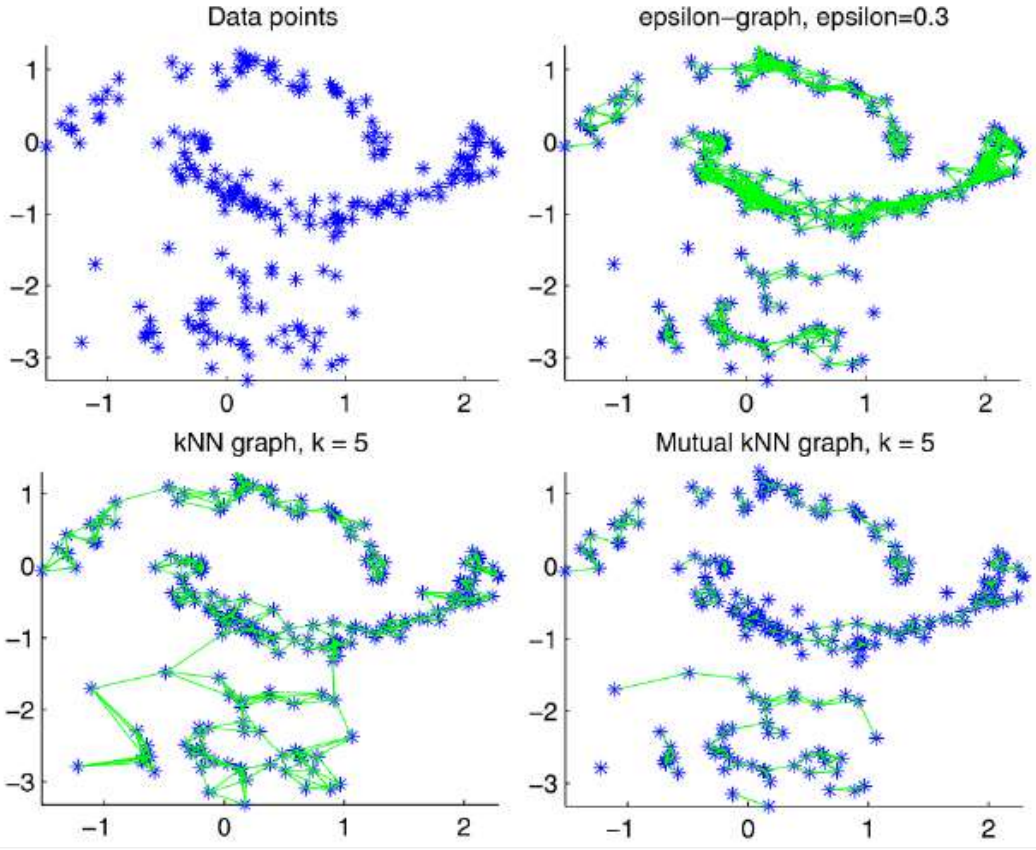
- A和B是图G的不相交子图，则定义子图的连接权：

$$W(A, B) := \sum_{i \in A, j \in B} w_{ij}$$

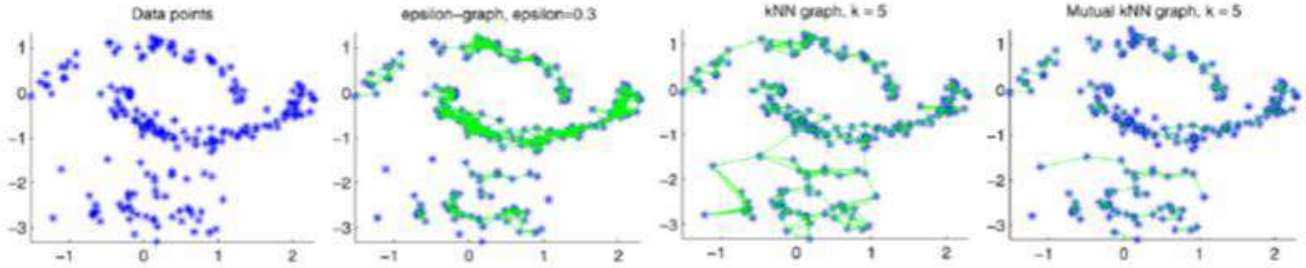
相似度图G的建立方法

- 全连接图：距离越大，相似度越小
  - 高斯相似度 $s(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right)$
- ε近邻图
  - 给定参数ε/如何选择ε？
    - 图G的权值的均值
    - 图G的最小生成树的最大边
- k近邻图(k-nearest neighbor graph)
  - 若vi的k最近邻包含vj，vj的k最近邻不一定包含vi：有向图
  - 忽略方向的图，往往简称“k近邻图”
  - 两者都满足才连接的图，称作“互k近邻图(mutual)”

相似度图G的举例


Data points
epsilon−graph, epsilon=0.3
kNN graph, k = 5
Mutual kNN graph, k = 5

权值比较



- ε近邻图：ε=0.3，“月牙部分”非常紧的连接了，但“高斯部分”很多都没连接。当数据有不同的“密度”时，往往发生这种问题。
- k近邻图：可以解决数据存在不同密度时有些无法连接的问题，甚至低密度的“高斯部分”与高密度的“月牙部分”也能够连接。同时，虽然两个“月牙部分”的距离比较近，但k近邻还可以把它们区分开。
- 互k近邻图：它趋向于连接相同密度的部分，而不连接不同密度的部分。这种性质介于ε近邻图和k近邻图之间。如果需要聚类不同的密度，这个性质非常有用。
- 全连接图：使用高斯相似度函数可以很好的建立权值矩阵。但缺点是建立的矩阵不是稀疏的。
- 总结：首先尝试使用k近邻图。

拉普拉斯矩阵及其性质

- 拉普拉斯矩阵：L = D – W

$$f^T L f = f^T D f - f^T W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j W_{ij}$$

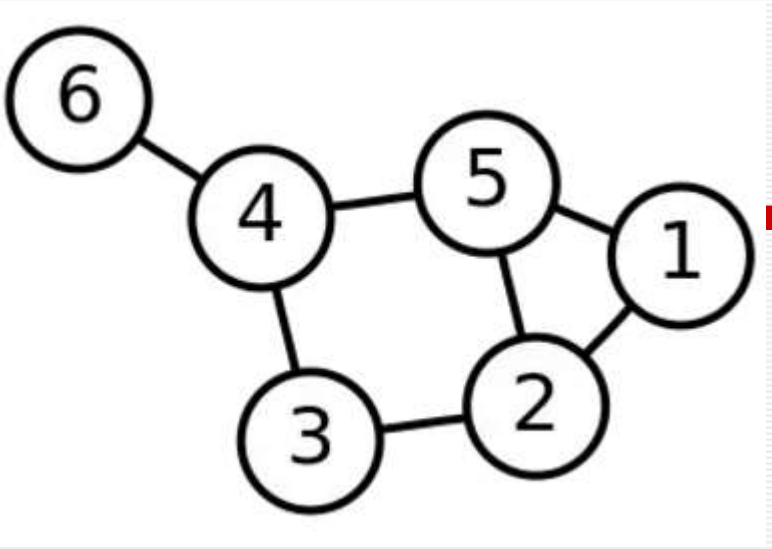
$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j W_{ij} + \sum_{j=1}^{n} d_j f_j^2 \right)$$

$$= \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left(f_i - f_j \right)^2$$

- L是对称半正定矩阵，最小特征值是0，相应的特征向量是全1向量。

拉普拉斯矩阵的定义

- 计算点之间的邻接相似度矩阵W
  - 若两个点的相似度值越大，表示这两个点越相似；
  - 同时，定义w_{ij}=0表示vi,vj两个点没有任何相似性(无穷远)
- W的第i行元素的和为vi的度。形成顶点度对角阵D
  - d_{ii}表示第i个点的度
  - 除主对角线元素，D其他位置为0
- 未正则的拉普拉斯矩阵： $L = D - W$
- 正则拉普拉斯矩阵
  - 对称拉普拉斯矩阵 $L_{sym} = D^{-\frac{1}{2}} \cdot L \cdot D^{\frac{1}{2}} = I - D^{-\frac{1}{2}} \cdot W \cdot D^{\frac{1}{2}}$
  - 随机游走拉普拉斯矩阵 $L_{rw} = D^{-1}L = I - D^{-1}W$
    - Random walk



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{pmatrix}$$

谱聚类算法：未正则拉普拉斯矩阵

- 输入：n个点{pi}，簇的数目k
  - 计算n×n的相似度矩阵W和度矩阵D；
  - 计算拉普拉斯矩阵L=D-W；
  - 计算L的前k个特征向量$U_1,U_2,...,U_k$；
  - 将k个列向量$U_1,U_2,...,U_k$组成矩阵U，$U\in R^{n\times k}$；
  - 对于i=1,2,...,n,令yi∈Rk是U的第i行的向量；
  - 使用k-means算法将点$(y_i)_{i=1,2,...,n}$聚类成簇C1,C2,...Ck；
  - 输出簇$A_1,A_2,...A_k$，其中，$Ai=\{j|y_j\in Ci\}$

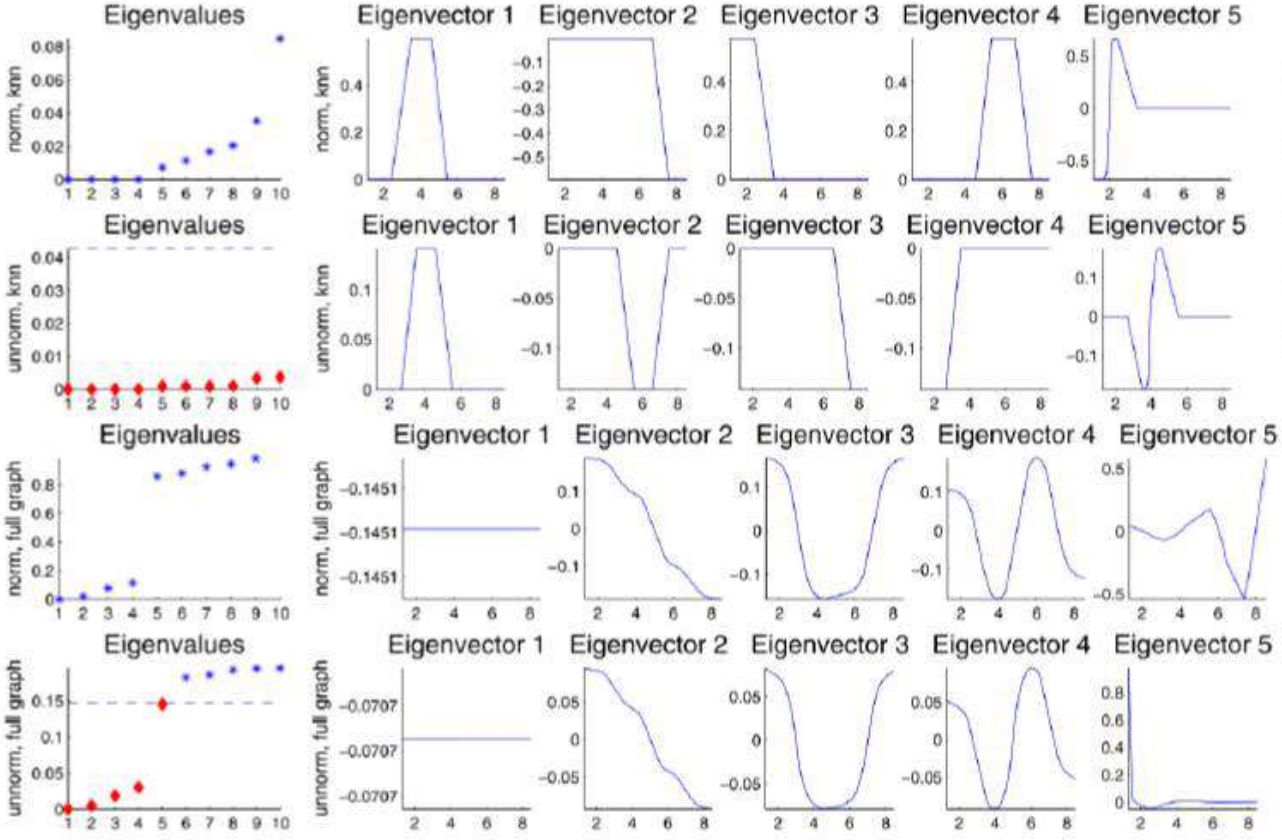
谱聚类算法：随机游走拉普拉斯矩阵

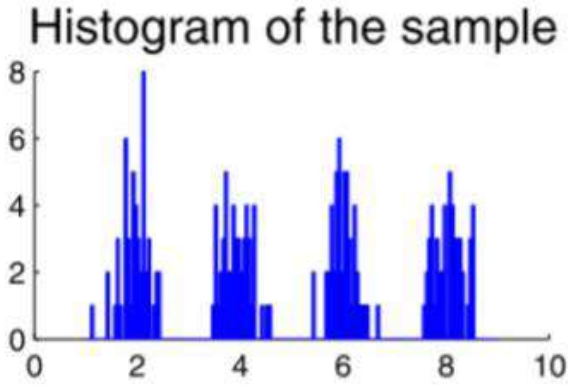
- 输入：n个点$\{P_i\}$，簇的数目k
  - 计算n×n的相似度矩阵W和度矩阵D；
  - 计算正则拉普拉斯矩阵$L_{rw}=D^{-1}(D-W)$；
  - 计算L_{rw}的前k个特征向量$u_1,u_2,\ldots,u_k$；
  - 将k个列向量$u_1,u_2,\ldots,u_k$组成矩阵U，$U\in R^{n\times k}$；
  - 对于$i=1,2,\ldots,n$,令$y_i\in R^k$是U的第i行的向量；
  - 使用k-means算法将点$(y_i)_{i=1,2,\ldots,n}$聚类成簇$C_1,C_2,\ldots C_k$；
  - 输出簇$A_1,A_2,\ldots A_k$，其中，$Ai=\{j|y_j\in Ci\}$

谱聚类算法：对称拉普拉斯矩阵

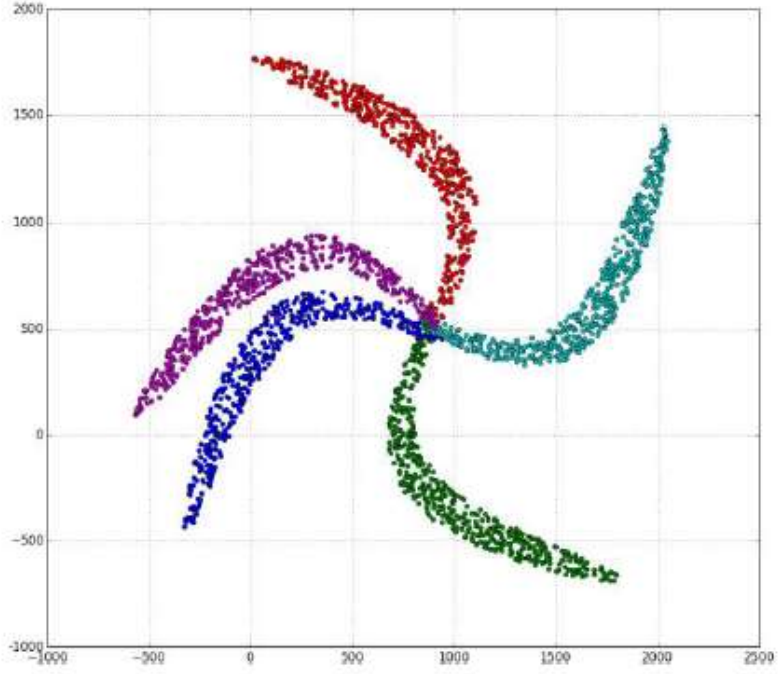
- 输入：n个点$\{p_i\}$，簇的数目k
  - 计算n×n的相似度矩阵W和度矩阵D；
  - 计算正则拉普拉斯矩阵$L_{sym}=D^{-1/2}(D-W)\ D^{-1/2}$；
  - 计算L_{sym}的前k个特征向量u1,u2,...,uk；
  - 将k个列向量$u_1,u_2,...,u_k$组成矩阵$u_1,u_2,...,u_k$；
  - 对于i=1,2,...,n,令$y_i\in R^k$是U的第i行的向量；
  - 对于i=1,2,...,n,将$y_i\in R^k$依次单位化，使得$|y_i|=1$；
  - 使用k-means算法将点$(y_i)_{i=1,2,...,n}$聚类成簇$C_1,C_2,...C_k$；
  - 输出簇$A_1,A_2,...A_k$，其中，Ai={j|$y_j\in$Ci}

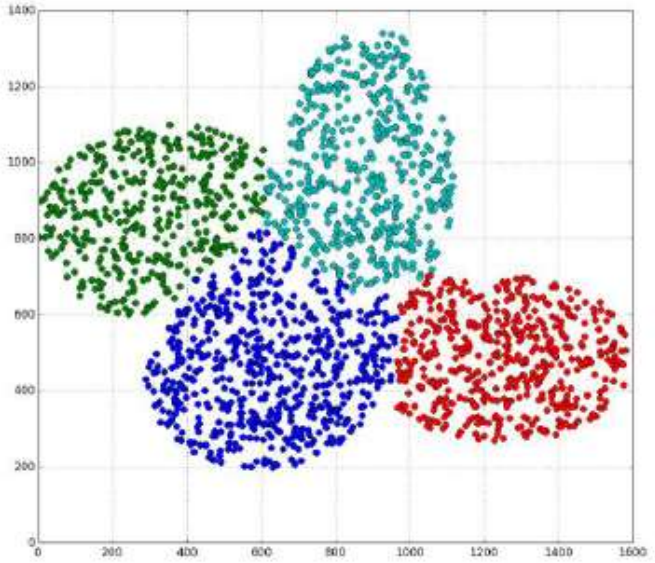
一个实例

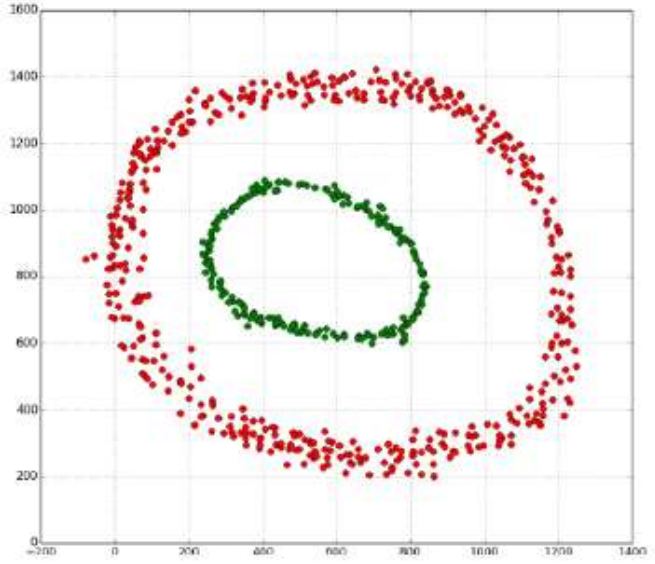

Eigenvalues
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
norm, knn
unnorm, knn
norm, full graph
unnorm, full graph


Histogram of the sample

聚类效果


2000
1500
1000
500
0
−500
−1000
−1000
−500
0
500
1000
1500
2000
2500


1400
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
1200
1400
1600

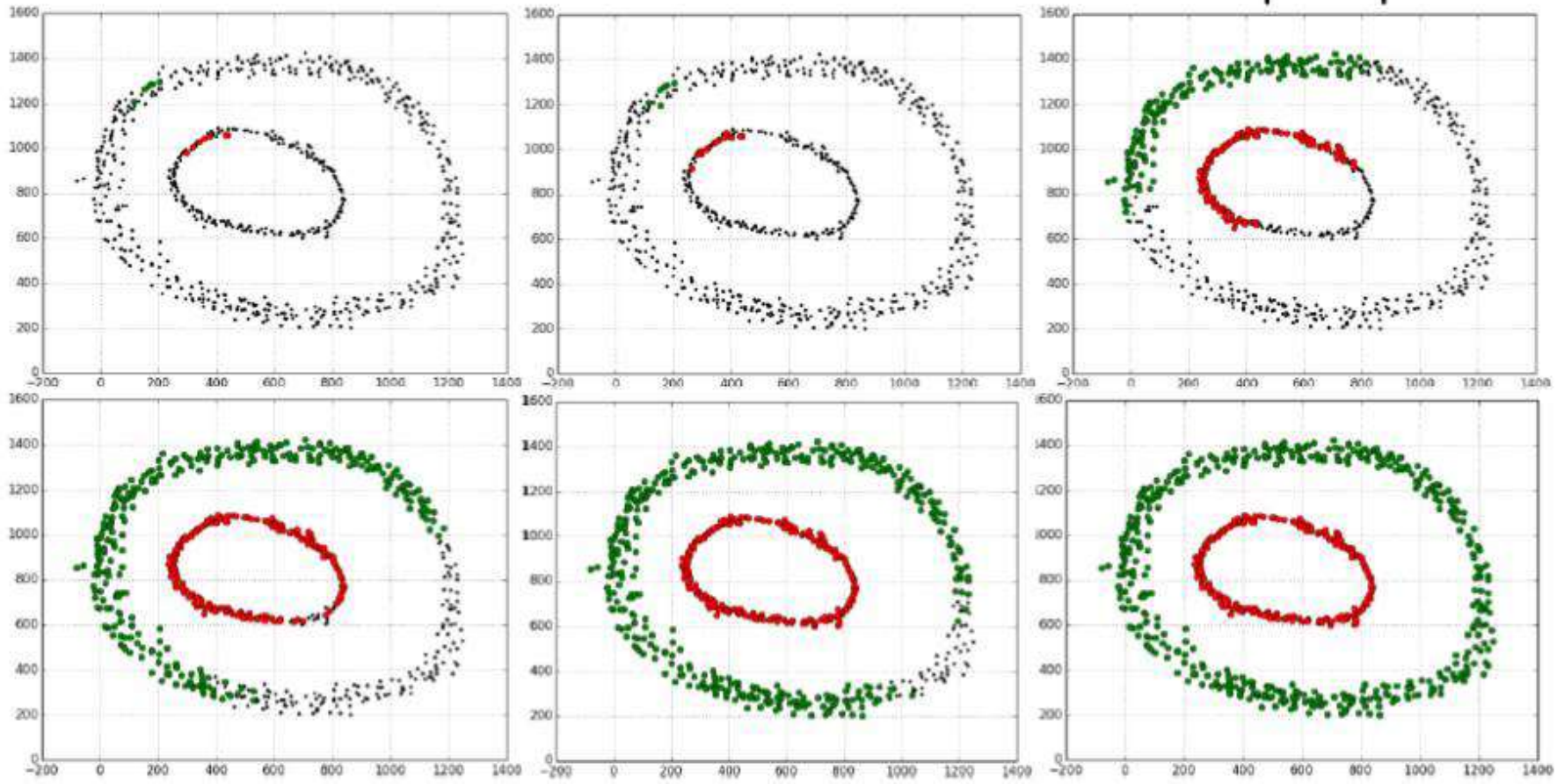

1600
1400
1200
1000
800
600
400
200
0
−200
0
200
400
600
800
1000
1200
1400

标签传递算法

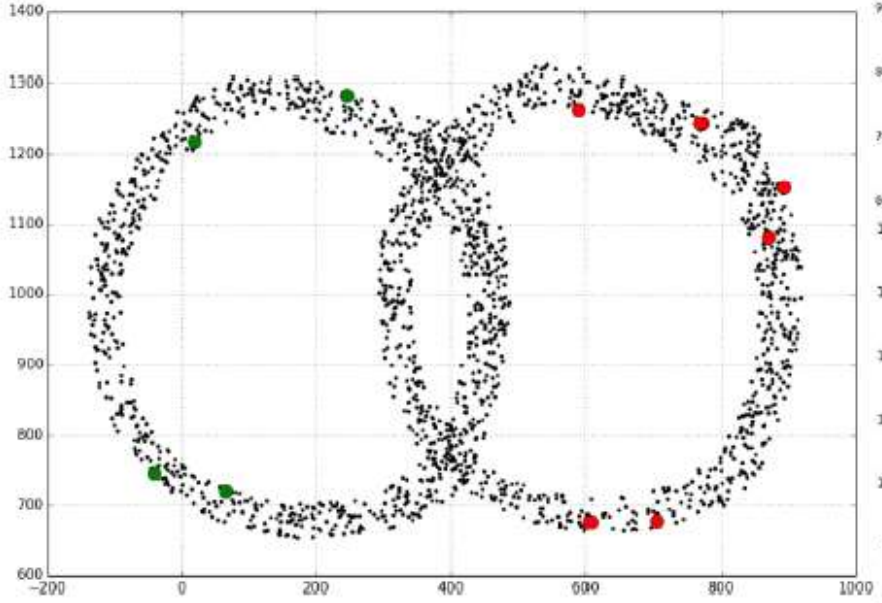
- 对于部分样本的标记给定，而大多数样本的标记未知的情形，是半监督学习问题。
- 标签传递算法(Label Propagation Algorithm,LPA)，将标记样本的标记通过一定的概率传递给未标记样本，直到最终收敛。

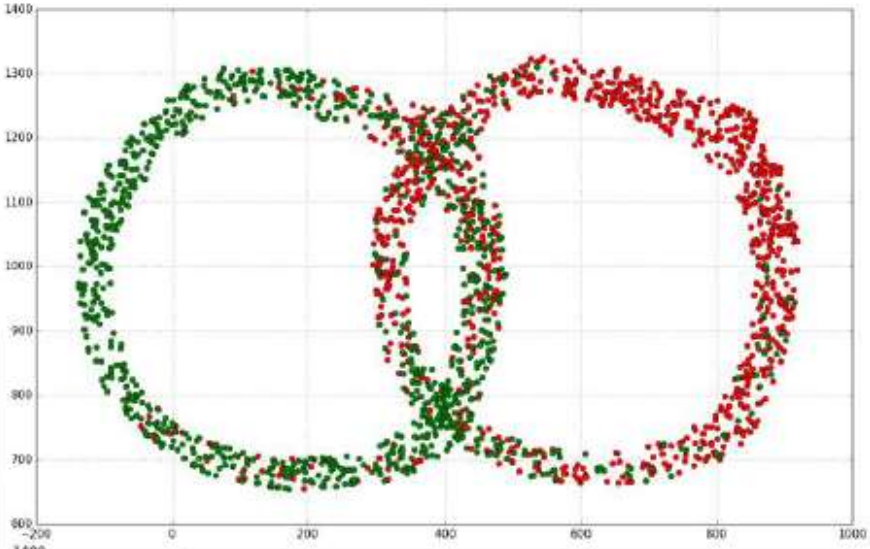
标签传递过程

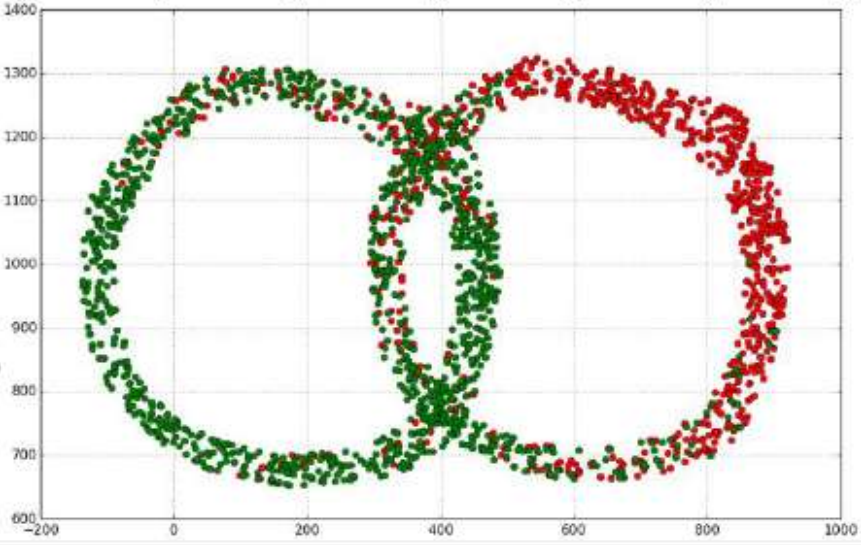
初始	1	10
20	30	40



带宽/邻域影响


1400
1300
1200
1100
1000
900
800
700
600
−200
0
200
400
600
800
1000


1400
1300
1200
1100
1000
900
800
700
600
−200
0
200
400
600
800
1000


1400
1300
1200
1100
1000
900
800
700
600
−200
0
200
400
600
800
1000

谢谢