第九章 聚类

9.1 常用的聚类划分方式有哪些？列举代表算法。

算法名称	算法步骤
K-Means（K均值）聚类	1. 首先我们选择一些类/组，并随机初始化它们各自的中心点。中心点是与每个数据点向量长度相同的位置。这需要我们提前预知类的数量(即中心点的数量)。 2. 计算每个数据点到中心点的距离，数据点距离哪个中心点最近就划分到哪一类中。 3. 计算每一类中中心点作为新的中心点。 4. 重复以上步骤，直到每一类中心在每次迭代后变化不大为止。也可以多次随机初始化中心点，然后选择运行结果最好的一个。
均值漂移聚类	1. 确定滑动窗口半径 r，以随机选取的中心点 C 半径为 r 的圆形滑动窗口开始滑动。均值漂移类似一种爬山算法，在每一次迭代中向密度更高的区域移动，直到收敛。 2. 每一次滑动到新的区域，计算滑动窗口内的均值来作为中心点，滑动窗口内的点的数量为窗口内的密度。在每一次移动中，窗口会想密度更高的区域移动。 3. 移动窗口，计算窗口内的中心点以及窗口内的密度，知道没有方向在窗口内可以容纳更多的点，即一直移动到圆内密度不再增加为止。 4. 步骤一到三会产生很多个滑动窗口，当多个滑动窗口重叠时，保留包含最多点的窗口，然后根据数据点所在的滑动窗口进行聚类
基于密度的聚类方法(DBSCAN)	1. 首先确定半径 r 和 minPoints. 从一个没有被访问过的任意数据点开始，以这个点为中心，r 为半径的圆内包含的点的数量是否大于或等于 minPoints，如果大于或等于 minPoints 则改点被标记为 central point,反之则会被标记为 noise point。 2. 重复 1 的步骤，如果一个 noise point 存在于某个 central point 为半径的圆内，则这个点被标记为边缘点，反之仍为 noise point。重复步骤 1，直到所有的点都被访问过
用高斯混合模型（GMM）的最大期望（EM）聚类	1. 选择簇的数量（与 K-Means 类似）并随机初始化每个簇的高斯分布参数（均值和方差）。也可以先观察数据给出一个相对精确的均值和方差。 2. 给定每个簇的高斯分布，计算每个数据点属于每个簇的概率。一个点越靠近高斯分布的中心就越可能属于该簇。 3. 基于这些概率我们计算高斯分布参数使得数据点的概率最大化，可以使用数据点概率的加权来计算这些新的参数，权重就是数据点属于该簇的概率。 4. 重复迭代 2 和 3 直到在迭代中的变化不大

凝聚层次聚类	1. 首先我们将每个数据点视为一个单一的簇，然后选择一个测量两个簇之间距离的度量标准。例如我们使用 average linkage 作为标准，它将两个簇之间的距离定义为第一个簇中的数据点与第二个簇中的数据点之间的平均距离。 2. 在每次迭代中，我们将两个具有最小 average linkage 的簇合并成为一个簇。 3. 重复步骤 2 知道所有的数据点合并成一个簇，然后选择我们需要多少个簇
图团体检测 (Graph Community Detection)	1. 首先初始分配每个顶点到其自己的团体，然后计算整个网络的模块性 M。 2. 第 1 步要求每个团体对（community pair）至少被一条单边链接，如果有两个团体融合到了一起，该算法就计算由此造成的模块性改变 ΔM。 3. 第 2 步是取 ΔM 出现了最大增长的团体对，然后融合。然后为这个聚类计算新的模块性 M，并记录下来。 4. 重复第 1 步和 第 2 步——每一次都融合团体对，这样最后得到 ΔM 的最大增益，然后记录新的聚类模式及其相应的模块性分数 M。 5. 重复第 1 步和 第 2 步——每一次都融合团体对，这样最后得到 ΔM 的最大增益，然后记录新的聚类模式及其相应的模块性分数 M

9.2 Kmeans 初始类簇中心点的如何选取？

1)选择彼此距离尽可能远的 K 个点

2)先对数据用层次聚类算法或者 Canopy 算法进行聚类，得到 K 个簇之后，从每个类簇中选择一个点，该点可以是该类簇的中心点，或者是距离类簇中心点最近的那个点。

9.3 传统的凝聚层次聚类过程每步合并两个簇。这样的方法能够正确地捕获数据点集的（嵌套的）簇结构吗？如果不能，解释如何对结果进行后处理，以得到簇结构更正确的视图。

1) 传统方法每次合并两个最相似的簇，直到最后只剩下一个簇结束。显然传统方法并不能正确的捕获正确的嵌套簇结构，该方法不考虑嵌套结构

2) 为了得到正确的簇结构，可以采用树结构，即每次合并过程记录该簇的父簇和子簇信息，合并到最后就可以得到一个正确的树形嵌套簇结构。

9.4 使用下表中的相似度矩阵进行单链和全链层次聚类。绘制树状图显示结果。树状图应当清楚地显示合并的次序。

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

相似度与距离成反比，相似度越大，则距离越近

- **单链(d_min，最大相似度)：定义簇的邻近度为不同两个簇的两个最近的点之间的距离**
  1. 先找出距离最近的两个点，P2 和 P5，similarity = 0.98，得到簇{P2,P5}
  2. 接着找{{P1},{P2,P5},{P3},{P4}}中最近两个点，{P2,P5}和 P3 最近，得到簇{P2,P3,P5}:
     dist(1, {2,5}) = min(dist(1,2),dist(1,5)) = max(sim(1,2),sim(1,5)) = max(0.1,0.35) = 0.35
     dist(1,3) = 0.41
     dist(1,4) = 0.55

     dist({2,5},3) = min(dist(2,3),dist(5,3)) = max(sim(2,3),sim(5,3)) = max(0.64,0.85) = 0.85
     dist({2,5},4) = max(0.47,0.76) = 0.76
     dist(3,4) = 0.44
  3. 不断执行上述步骤，结果如下图所示：

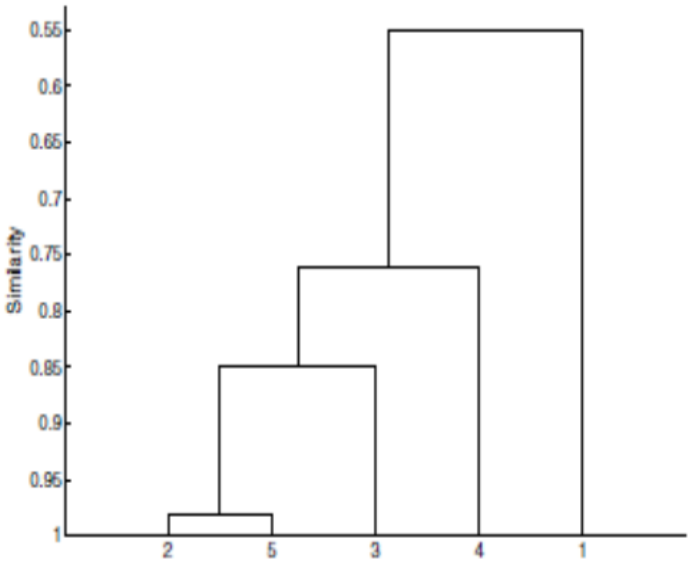

图 1：单链树状图结果

- **全链（d_max）：定义簇的邻近度为不同两个簇的两个最远的点之间的距离**
  1. 先找出距离最近的两个点，P2 和 P5，similarity = 0.98，得到簇{P2,P5}
  2. 接着找{{P1},{P2,P5},{P3},{P4}}中最近两个点，{P2,P5}和 P3 最近，得到簇{P2,P3,P5}:
     dist(1,{2,5}) = max(dist(1,2),dist(1,5)) = min(sim(1,2),sim(1,5)) = min(0.1,0.35) = 0.1
     dist(1,3) = 0.41

dist(1,4) = 0.55
dist({2,5},3) = max(dist(2,3),dist(5,3)) = min(sim(2,3),sim(5,3)) = min(0.64,0.85) = 0.64
dist({2,5},4) = min(0.47,0.76) = 0.47
dist(3,4) = 0.44

3. 接着找{{P1},{P2,P3,P5},{P4}}中最近两个点，P1 和 P4 最近，得到簇{P1,P4}：
   dist(1,{2,3,5}) = max(dist(1,2),dist(1,3),dist(1,5)) = min(0.1,0.41,0.35) = 0.1
   dist(1,4) = 0.55
   dist({2,3,5},4) = min(0.47,0.44,0.76) = 0.44

4. 最后将{P2,P3,P5}和{P1,P4}聚成一类，结果如下图所示

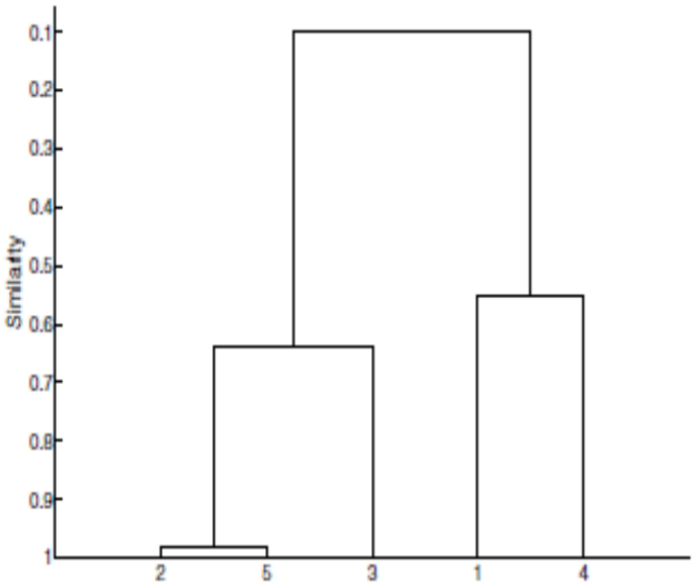

图 2：全链树状图结果