k-中心点算法

- k-means算法的改进方法——k-中心点算法
- k-中心点算法：k -means算法对于孤立点（噪声）是敏感的。为了解决这个问题，不采用簇中的平均值作为参照点，可以选用簇中位置最中心的对象，即中心点作为参照点。这样划分方法仍然是基于最小化所有对象与其参照点之间的相异度之和的原则来执行的。

k-中心点算法

- k-means算法为什么会对“噪声”敏感。还记得K-means寻找质点的过程吗?
- 对某类簇中所有的样本点维度求平均值，即获得该类簇质点的维度。当聚类的样本点中有“噪声”（离群点）时，在计算类簇质点的过程中会受到噪声异常维度的干扰，造成所得质点和实际质点位置偏差过大，从而使类簇发生“畸变”。

k-中心点算法

- Eg: 类簇C1中已经包含点A(1,1)、B(2,2)、 C(1,2)、 D(2,1)，质点为
- Centroid((1+2+1+2)/4,(1+2+2+1)/4)=centroid(1.5,1.5),
- 假设N(100,100)为异常点，当它纳入类簇C1时，计算新质点为
- Centroid((1+2+1+2+100)/5,(1+2+2+1+100)/5)=centroid(21,21),
- 此时造成了类簇C1质点的偏移，在下一轮迭代重新划分样本点的时候，将大量不属于类簇C1的样本点纳入，因此得到不准确的聚类结果。

k-中心点算法

- 为了解决该问题，K中心点算法（K-medoids）提出了新的质点选取方式，而不是简单像k-means算法采用均值计算法。
- 在K中心点算法中，每次迭代后的质点都是从聚类的样本点中选取，而选取的标准就是当该样本点成为新的质点后能提高类簇的聚类质量，使得类簇更紧凑。
- 该算法使用绝对误差标准来定义一个类簇的紧凑程度。

$$E=\sum_{1}^{k}\sum_{p\in Cj}|p-oj|$$

- p是空间中的样本点，Oj是类簇Cj的质点
- 如果某样本点成为质点后，绝对误差能小于原质点所造成的绝对误差，那么K中心点算法认为该样本点是可以取代原质点的，在一次迭代重计算类簇质点的时候，我们选择绝对误差最小的那个样本点成为新的质点。

k-中心点算法

- 如果某样本点成为质点后，绝对误差能小于原质点所造成的绝对误差，那么K中心点算法认为该样本点是可以取代原质点的，在一次迭代重计算类簇质点的时候，我们选择绝对误差最小的那个样本点成为新的质点。
- Eg：
- 样本点A –>E1=10
- 样本点B –>E2=11
- 样本点C –>E3=12
- 原质点O–>E4=13，
- 那我们选举A作为类簇的新质点。

k-中心点算法

- 与K-means算法一样，K-medoids也是采用欧几里得距离来衡量某个样本点到底是属于哪个类簇。终止条件是，当所有的类簇的质点都不在发生变化时，即认为聚类结束。
- 它最初随机选择k个对象作为中心点，该算法反复的用非代表对象（非中心点）代替代表对象，试图找出更好的中心点，即使总的代价降低，以改进聚类的质量。
- 该算法除了改善K-means的“噪声”敏感以后，其他缺点和K-means一致，并且由于采用新的质点计算规则，也使得算法的时间复杂度上升。

用于K-medoids算法的数据集

表2.6 用于K-medoids算法的数据集

空间中有5个点{A,B,C,D,E}，表中记录了各点之间的距离，实现聚类划分，设k=2，初始中心点为{A,B}。

Instance	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

用于K-medoids算法的数据集

- 假设分为2类，以A，B为中心点，初始聚类为{A,C,D}和{B,E}。
- 接下来进行交换（以非代表对象代替代表对象），需要计算TC_{AC}、TC_{AD}、TC_{AE}、TC_{BC}、TC_{BD}、TC_{BE}。
- TC_{ij}表示用非中心点j替换中心点i所产生的代价。

用于K-medoids算法的数据集

- 计算TC_{AC}：当A被C替换后，遍历所有对象，判断他们是否聚到别的类里。
  - 先看A是否变化：C成为中心点后，A离B比A离C近，故A被划分到B簇里。所产生的代价为d(A,B)-d(A,A)=1。(d(i,j)表示i划分到中心点j的距离；差值表示属于新的中心点-属于旧的中心点产生的代价。)
  - 看B是否变化：C成为中心点后，B当然离自己是最近的，不变
  - 看C是否变化：C成为中心点后，C划分到C簇里，代价为d(C,C)-d(C,A)=-2
  - 看D是否变化：C成为中心点后，D离C最近，故划分到C里，代价为d（D,C）-d（D,A）=-1；
  - 看E是否变化：C成为中心点后，E离B最近，为0
- TC_{AC}就等于上述的代价之和，为
- 1+0-2-1+0=-2。

用于K-medoids算法的数据集

- 同理需要计算TC_{AD}=-2、TC_{AE}=-1、TC_{BC}=-2、TC_{BD}=-2、TC_{BE}=-2
- 然后选取代价最小的替换，这里有多个选择，随便选择一个就行。选C的话，新的簇为{C,D}和{A,B,E}。新的簇中心为C，B，继续迭代计算直到收敛。

用于K-medoids算法的数据集

- 为了判定一个非代表对象o_{random}是否是当前一个代表对象oi的好的替代，对于每一个非中心点p，需要考虑下面4中情况：
  - 第一种情况：p当前隶属于代表对象oj（A类中心点），如果oj被orandom所代替作为代表对象，并且p离其他代表对象oi（B类的中心点）最近，则p重新分配给oi。(i!=j)
  - 第二种情况：p当前隶属于代表对象oj（A类中心点），如果oj被orandom所代替作为代表对象，并且p离orandom（新的中心点）最近，则p重新分配给orandom。(i!=j)
  - 第三种情况：p当前隶属于代表对象oi（B类中心点），如果oj被orandom所代替作为代表对象，并且p仍然离oi最近，则p不发生变化。(i!=j)
  - 第四种情况：p当前隶属于代表对象oi（B类中心点），如果oj被orandom所代替作为代表对象，并且p离orandom最近，则p重新分配给orandom。(i!=j)

用于K-medoids算法的数据集

- 最终结果得到以A，E为中心点的两个簇：{A,B,C,D}和{E}。

k-中心点算法小结

- K-mediods算法具有能够处理大型数据集，结果簇相当紧凑，并且簇与簇之间明显分明的优点，这一点和K-means算法相同。
- 同时，该算法也有K-means同样的缺点，如，必须事先确定类簇数和中心点，簇数和中心点的选择对结果影响很大；一般在获得一个局部最优的解后就停止了；对于除数值型以外的数据不适合；只适用于聚类结果为凸形的数据集等。
- 与K-means相比，K-mediods算法对于噪声不那么敏感，这样对于离群点就不会造成划分的结果偏差过大，少数数据不会造成重大影响。
- K-mediods由于上述原因被认为是对K-means的改进，但由于按照中心点选择的方式进行计算，算法的时间复杂度也比K-means上升了。