计算几何实验报告

全连接聚类的快速算法的实现

（Fast Algorithms for Complete Linkage Clustering）

吴鲲 （994845）& 张敏（994988） 2000/1/15

[算法的主要思想]

1． 简介：

通常计算全连接聚类的法都需要立方级的时间复杂度。还有的算法使用优先级队列，能使时间复杂度降到 $O(n^2 \log n)$. Defays 提出了一种使用线性空间的方法，使时间复杂度降到了 $O(n^2)$。但是这种方法的输出依赖与插入的具体顺序，因此只是一种近似的分类。即使使用并行算法，仍然不能比 $O(n^2)$ 更小。

这里我们假设在平面上有 n 个不同的点，点之间的距离用欧氏距离来衡量。则在本算法中，全连接聚类的复杂度可达 $O(n\log^2 n)$,及线性空间。

2． 基本定义：

(1)全连接距离：$d(c,c')=\max\{|xy|: x\in c \text{ and } y\in c'\}$

(2)直径：某类中包含所有点的最小范围的直径。

(3)2-cluster：若 S 的一个子集 D 是一个 2-cluster，如果 D 中的任何点到 S-D 中的点的距离大于 2*Diam（D）或 D=S。

3．算法原理：

我们可以求得 D 中的任意点到 S-D 中的任意点之间的距离。有这样的定理：对于任意的 2-聚类子集 D，且 D 有不少于 2 个孩子结点。设 d 为 D 的两个不同的孩子结点之间的最小距离，d’为最大距离，则 $d'<2^{2m-3}d$. [*]

算法这样描述：

仅当两个类 C 和 C’是局部最近点时，我们把这两个类合并。这样做的主要原因是因为在某一个聚集的类附近，只有常数个类。

[算法描述]

对每一个 2-cluster 分别按步骤处理:

我们易知任何的 2-cluster 在一个全连接分类中也是一个聚集的类。因此我们可以按照后序遍历 2-cluster 树的方法来计算一个全连接分类。因此我们只要考虑下面的情况：给定一个 2-cluster 集 D ，根据全连接的方法来把 D 的所有孩子进行聚类。方法如下：

以步骤 $p_0,p_1,.....$来对 D 的孩子聚类。设 LF 是 D 的所有不同孩子中最接近的一对的距离。

第 p_i 步的目标是把所有满足下面的条件两类聚集到一起：

$L_i \leq d(c,c')<2L_i$. 其中 d(c,c')为上文的定义。L_i 等于 2^iL。在第 p_i 步中对

应于距离 L_i 。

现在考虑 D 的一个孩子 D'。L'为 D'中的任何点到 S- D'的任何点之间的最小距离。显然，当全连接距离< L'时，没有其他的类可以与 D'相聚。因此这时 D'可以忽略。直到全连接的距离>= L'。因此如果

$L_i \leqslant L' < 2L_i$ ， 则 D'才在第 p_i 步中考虑。

这样，我们把 D 的每一个孩子在某一步中考虑，也就是说 D 的每一个孩子一直在闲置，直到它到达了要考虑它的那一步为止。根据[*]式，设计每个孩子属于哪一步所需的时间为线性的。

[算法实现]

用图 G 来描述状态 Pi：

（1） G 的顶点对应 L 中的类。L 包括：Pi-1 中剩下的类和本来就属于状态 Pi 中的类。

（2） G 的边：两类距离<$2L_i$ 的才有边。边长=d(c,c').

以一个空的堆栈开始，重复下列步骤，直到 G 没有边：

Do:

（1）从堆栈中 pop 出边 e .（若堆栈为空，则 e=任意边）

（2）e'为与 e 相邻的最短边。（若 e 无邻边，则 goto 4）.

（3）If e'<e ,then push e. Goto 1

（4）合并 e 连接的两类：

do

a. G 中加新的顶点 Ce，

b. 任意（c，c'）的边，c'为 e 的末点。if d(c,c') <$2L_i$ 且边不存在，则加（C,Ce）入 G.

c. If 堆栈非空，则 pop 并丢弃栈顶的边。

d. 删除 e 的末结点，及 G 中所有与 e 相邻的边。

Until G 中无边。

最后的聚类结果，用图形的直观形式表示出来：用不同的颜色标注各点。同一中颜色表示的点为已经聚在同一类中。

[数据结构]

考虑含有 n 个结点的点集。

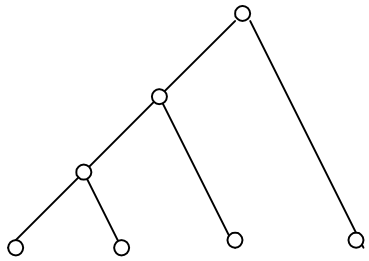
采用树形结构。根结点为一个空的结点，无实际意义。初始时，认为每个点是一类。共有 n 类。它们是树的叶子结点。以后每聚一类，则生成上一层的新的结点（表示新的类）。直到最后，只剩下一类（即所有的点都聚为同一类）； 或聚类结束（所有剩下的类之间的距离都足够远）。

最后，根结点之下的那一层结点，就是聚类的最终结果——剩下的类。

[时间复杂度分析]

考虑树形结构，归并的最差情况（即 n 个点的归并次数最多的情况）：

（这样的情况下，归并层数为 n-1）。如下图所示：



但是这种情况下，基于上文的描述，某个与其他点的距离为 2^iL 的结点，要到第 I 步才进行归并。因此，这种情况下，每层只考虑两个结点即可。因此，层数增加造成的时间复杂度，在每层处理结点的减少上得到了补偿。

一般来说，层数为 logN。 判断每个结点在第几步处理，所用时间复杂度为 o(n). 每次归并处理的结点数近似为 logN。 因此总的时间复杂度可以为 $nlog^2n$. （因为缺少一些补充的计算几何知识，所以这里只做近似估计）。

[算法的改进]

研究文章中所描述的算法，发现聚类的结果虽然与点插入的顺序无关，但是取决于初始时，两个点的最近距离。如果所有点中，只有两个点，其距离（L）非常小，而其他点虽然也表现了一定的聚集性，但是相互之间的距离大于 2L， 则聚类的结果是只有这两个点聚到了一起，而其他点却依然各自孤立为一类。因此这里作了如下的改进：

聚类结束后，如果想要在进一步再次聚类，则以刚才聚类的结果为初始条件（即把每个类看作一个单位，作为树最底层的结点），再次进行聚类。如果用户对聚类结果仍然不满意，则可以继续进行。直到满足用户要求或聚为一类为止。

[总结算法的优点]

1.时间复杂度低于以前任何一种提出的算法.

2.空间复杂度可以在线性空间上实现。这里采用树的结构，简单易行。（实现时，每次只保存当前层结点。因此任何时刻保存的结点数不会超过 n+1，且其中有一个是根结点。）

3.算法实现时，用户只要输入各点的图形。程序一次性计算两点之间的距离，作为全连接图的基本信息。每次聚类，可以继承上一次聚类的结点各种信息，不用做重复运算，也不必计算距离。

[算法原文出处]

D. Krznaric and C.Levcopoulos, Fast Algorithms for Complete Linkage Clustering, Discrete&Computational Geometry, 19: 131-145 ,1998.