基于距离的层次聚类算法：

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies)

遍历一次数据集，构造一个聚类特征树，类似于B+树，每个节点包含若干个聚类特征，非叶子节点有子节点，叶子节点之间相互连接。

聚类特征(cluster feature,CF)：三元组<N,LS,SS>，其中N代表了这个CF中拥有的样本点的数量，这个好理解；LS代表了这个CF中拥有的样本点各特征维度的和向量，SS代表了这个CF中拥有的样本点各特征维度的平方和。

CF满足相加性，在CF Tree中的某一个节点的某一个CF中，有下面5个样本

(3,4), (2,6), (4,5), (4,7), (3,8)。则它对应的N=5，

LS=(3+2+4+4+3,4+6+5+7+8)=(16,30),

SS=(32+22+42+42+32+42+62+52+72+82)=(54+190)=244



可调参数B和L，每个内部节点最多CF个数和每个叶子节点最多CF个数

（一）构造CF Tree

总结下CF Tree的插入：

　　1. 从根节点向下寻找和新样本距离最近的叶子节点和叶子节点里最近的CF节点

　　2. 如果新样本加入后，这个CF节点对应的超球体半径仍然满足小于阈值T，则更新路径上所有的CF三元组，插入结束。否则转入3.

　　3. 如果当前叶子节点的CF节点个数小于阈值L，则创建一个新的CF节点，放入新样本，将新的CF节点放入这个叶子节点，更新路径上所有的CF三元组，插入结束。否则转入4。

　　4.将当前叶子节点划分为两个新叶子节点，选择旧叶子节点中所有CF元组里超球体距离最远的两个CF元组，分布作为两个新叶子节点的第一个CF节点。将其他元组和新样本元组按照距离远近原则放入对应的叶子节点。依次向上检查父节点是否也要分裂，如果需要按和叶子节点分裂方式相同。

（二）上面讲了半天的CF Tree，终于我们可以步入正题BIRCH算法，其实将所有的训练集样本建立了CF Tree，一个基本的BIRCH算法就完成了，对应的输出就是若干个CF节点，每个节点里的样本点就是一个聚类的簇。也就是说BIRCH算法的主要过程，就是建立CF Tree的过程。

　　当然，真实的BIRCH算法除了建立CF Tree来聚类，其实还有一些可选的算法步骤的，现在我们就来看看 BIRCH算法的流程。

1） 将所有的样本依次读入，在内存中建立一颗CF Tree, 建立的方法参考上一节。

2）（可选）将第一步建立的CF Tree进行筛选，去除一些异常CF节点，这些节点一般里面的样本点很少。对于一些超球体距离非常近的元组进行合并

　　3）（可选）利用其它的一些聚类算法比如K-Means对所有的CF元组进行聚类，得到一颗比较好的CF Tree.这一步的主要目的是消除由于样本读入顺序导致的不合理的树结构，以及一些由于节点CF个数限制导致的树结构分裂。

　　4）（可选）利用第三步生成的CF Tree的所有CF节点的质心，作为初始质心点，对所有的样本点按距离远近进行聚类。这样进一步减少了由于CF Tree的一些限制导致的聚类不合理的情况。

　　从上面可以看出，BIRCH算法的关键就是步骤1，也就是CF Tree的生成，其他步骤都是为了优化最后的聚类结果。

参考文档<https://www.cnblogs.com/pinard/p/6179132.html>