0.1 BIRCH 聚类算法

BIRCH（Balanced Iterative Reducing and Clustering using Hierarchies）是一种适用于大规模数据集的聚类算法。BIRCH 算法通过构建和维护一个簇特征树（Clustering Feature Tree, CF Tree）来有效地对大规模数据进行聚类。CF Tree 是一种高度压缩的树状数据结构，能够通过增量式学习动态地调整簇。

BIRCH 算法的核心在于簇特征（Clustering Feature, CF）的计算和使用。簇特征 CF 是一个三元组 $(N, \vec{LS}, SS)$，用于有效地描述簇中的数据点。具体计算如下：

$$\text{CF} = (N, \vec{LS}, SS)$$

其中，N 是簇中的点数，$\vec{LS}$ 是簇内所有数据点的线性和：

$$\vec{LS} = \sum_{i=1}^{N} \vec{x}_i$$

SS 是簇内所有数据点的平方和：

$$SS = \sum_{i=1}^{N} \|\vec{x}_i\|^2$$

通过簇特征，簇的质心和半径可以计算为：

$$\vec{C} = \frac{\vec{LS}}{N}, \quad \text{Radius} = \sqrt{\frac{SS}{N} - \left(\frac{\vec{LS}}{N}\right)^2}$$

两个簇 CF_1 和 CF_2 之间的距离可以通过以下公式计算：

$$D(CF_1, CF_2) = \sqrt{\frac{N_1 \cdot N_2}{N_1 + N_2}} \cdot \|\vec{C}_1 - \vec{C}_2\|$$

BIRCH 算法的主要步骤如下：

- 构建 CF Tree：通过遍历数据集，BIRCH 算法将每个数据点插入到 CF Tree 中，并根据簇特征进行调整。
- 聚类阶段：在 CF Tree 构建完成后，BIRCH 可以使用凝聚层次聚类或其他算法对叶节点进行进一步的聚类，以生成最终的簇。

BIRCH 的优点在于它能够有效地处理大规模数据，并且能够在内存受限的情况下进行聚类。缺点是对于非球形簇的识别能力有限，并且可能对簇的初始构建顺序较为敏感。

BIRCH 算法的时间复杂度通常为 $O(n)$，其中 n 为样本数量，适合大规模数据集的聚类任务。