## DBScan Clusterer Algorithm

## 概念

### **1. 欧几里德距离**

度量欧几里得空间中两点间的距离（直线）

### **2. 算法思路**

**1. 判断各个对象是否为核心对象**

即计算该点到其他所有点的距离，当距离小于给定的radius，则可达密度+1，计算完所有的点之后，判断对象的直接密度可达点>=minPts，则认为其是可达的。

**2. 合并聚类集合**

在集合i中的点k，同时存在于集合j中，那么这两个集合就可以合并成一个大集合。

### **3.代码**

计算欧几里德距离

public static double euclideanDistance(double[] p1, double[] p2)  
 throws DimensionMismatchException {  
 checkEqualLength(p1, p2);//校验数组长度是否相同  
 double sum = 0;  
 for (int i = 0; i < p1.length; i++) {  
 final double dp = p1[i] - p2[i];  
 sum += dp \* dp;  
 }  
 return Math.sqrt(sum);  
 }

获取密度可达的邻居点

// T extends Clusterable   
private List<T> getNeighbors(final T point, final Collection<T> points) {  
 final List<T> neighbors = new ArrayList<T>();  
 for (final T neighbor : points) {  
 if (point != neighbor && euclideanDistance(neighbor, point) <= eps) {  
 neighbors.add(neighbor);  
 }  
 }  
 return neighbors;  
 }  
  
public interface Clusterable {  
  
 /\*\*  
 \* Gets the n-dimensional point.  
 \*  
 \* @return the point array  
 \*/  
 double[] getPoint();  
}

聚类的合并

private Cluster<T> expandCluster(final Cluster<T> cluster,  
 final T point,  
 final List<T> neighbors,  
 final Collection<T> points,  
 final Map<Clusterable, PointStatus> visited) {  
 // 在判断邻居点个数时，没有将中心点放到聚类的集合当中，所以这里的放进去  
 cluster.addPoint(point);  
 // 标记当前点  
 visited.put(point, PointStatus.PART\_OF\_CLUSTER);  
  
 List<T> seeds = new ArrayList<T>(neighbors);  
 //用于判断边界  
 int index = 0;  
 // 遍历聚类中每个点的状态，如果为null，则没有进行聚类过  
 while (index < seeds.size()) {  
 final T current = seeds.get(index);  
 PointStatus pStatus = visited.get(current);  
 // only check non-visited points  
 if (pStatus == null) {  
 //获取邻居点  
 final List<T> currentNeighbors = getNeighbors(current, points);  
 // 符合要求的话就将两个聚类合并  
 if (currentNeighbors.size() >= minPts) {  
 seeds = merge(seeds, currentNeighbors);  
 }  
 }  
 // 到这pstaus！=null的，并且肯定是PART\_OF\_CLUSTER，如果不是就该为这个状态  
 if (pStatus != PointStatus.PART\_OF\_CLUSTER) {  
 visited.put(current, PointStatus.PART\_OF\_CLUSTER);  
 cluster.addPoint(current);  
 }  
  
 index++;  
 }  
 return cluster;  
 }  
  
 private List<T> merge(final List<T> one, final List<T> two) {  
 final Set<T> oneSet = new HashSet<T>(one);  
 for (T item : two) {  
 //出去两个集合都有的  
 if (!oneSet.contains(item)) {  
 one.add(item);  
 }  
 }  
 return one;  
 }

完整代码参考org.apache.commons.math3.ml.clustering.DBSCANClusterer

## Apache Common Math

由于之前没有接触过这个包，这里主要介绍下其提供的主要算法：

* [org.apache.commons.math4.stat](https://commons.apache.org/proper/commons-math/userguide/stat.html) - statistics, statistical tests，统计，统计测试
* [org.apache.commons.math4.analysis](https://commons.apache.org/proper/commons-math/userguide/analysis.html) - rootfinding, integration, interpolation, polynomials，寻根，积分，插值，多项式
* [org.apache.commons.math4.random](https://commons.apache.org/proper/commons-math/userguide/random.html) - random numbers, strings and data generation，随机数，字符串和数据生成
* [org.apache.commons.math4.special](https://commons.apache.org/proper/commons-math/userguide/special.html) - special functions (Gamma, Beta)，特殊功能
* [org.apache.commons.math4.linear](https://commons.apache.org/proper/commons-math/userguide/linear.html) - matrices, solving linear systems，矩阵
* [org.apache.commons.math4.util](https://commons.apache.org/proper/commons-math/userguide/utilities.html) - common math/stat functions extending java.lang.Math
* [org.apache.commons.math4.complex](https://commons.apache.org/proper/commons-math/userguide/complex.html) - complex numbers，复数
* [org.apache.commons.math4.distribution](https://commons.apache.org/proper/commons-math/userguide/distribution.html) - probability distributions，概率分布
* [org.apache.commons.math4.fraction](https://commons.apache.org/proper/commons-math/userguide/fraction.html) - rational numbers，有理数
* [org.apache.commons.math4.transform](https://commons.apache.org/proper/commons-math/userguide/transform.html) - transform methods (Fast Fourier)，快速傅立叶变换
* [org.apache.commons.math4.geometry](https://commons.apache.org/proper/commons-math/userguide/geometry.html) - geometry (Euclidean spaces and Binary Space Partitioning)，欧式几何和二分空间
* [org.apache.commons.math4.optim](https://commons.apache.org/proper/commons-math/userguide/optimization.html) - function maximization or minimization，函数的最大值和最小值
* [org.apache.commons.math4.ode](https://commons.apache.org/proper/commons-math/userguide/ode.html) - Ordinary Differential Equations integration，常用的积分微分方程
* [org.apache.commons.math4.genetics](https://commons.apache.org/proper/commons-math/userguide/genetics.html) - Genetic Algorithms，遗传算法
* [org.apache.commons.math4.fitting](https://commons.apache.org/proper/commons-math/userguide/fitting.html) - Curve Fitting，曲线拟合
* [org.apache.commons.math4.ml](https://commons.apache.org/proper/commons-math/userguide/ml.html) - Machine Learning，机器学习

参考链接：

[1. Commons-math Overview](https://commons.apache.org/proper/commons-math/userguide/overview.html)

[2. DBScan Wiki](https://zh.wikipedia.org/wiki/DBSCAN)