基于密度的聚类算法 — DBSCAN

**一  什么是基于密度的聚类算法**

由于层次聚类算法和划分式聚类算往往只能发现凸形的聚类簇。为了弥补这一缺陷，发现各种任意形状的聚类簇，开发出基于密度的聚类算法。这类算法认为，在整个样本空间点中，各目标类簇是由一群的稠密样本点组成的，而这些稠密样本点被低密度区域（噪声）分割，而算法的目的就是要过滤低密度区域，发现稠密样本点。

**二  DBSCAN（Density-based Spatial Clustering of Applications with Noise）**

是一种基于高密度联通区域的聚类算法，它将类簇定义为高密度相连点的最大集合。它本身对噪声不敏感，并且能发现任意形状的类簇。

DBSCAN中的的几个定义：

Ε领域：给定对象半径为Ε内的区域称为该对象的**Ε领域**

核心对象：如果给定对象Ε领域内的样本点数大于等于MinPts，则称该对象为**核心对象**。

直接密度可达：对于样本集合D，如果样本点q在p的Ε领域内，并且p为核心对象，那么对象q从对象p**直接密度可达**。

密度可达：对于样本集合D，给定一串样本点p1,p2….pn，p= p1,q= pn,假如对象pi从pi-1直接密度可达，那么对象q从对象p**密度可达**。

密度相连：对于样本集合D中的任意一点O，如果存在对象p到对象o密度可达，并且对象q到对象o密度可达，那么对象q到对象p**密度相连**。

可以发现，密度可达是直接密度可达的传递闭包，并且这种关系是非对称的。密度相连是对称关系。DBSCAN目的是找到密度相连对象的最大集合。

Eg: 假设半径Ε=3，MinPts=3，点p的E领域中有点{m,p,p1,p2,o}, 点m的E领域中有点{m,q,p,m1,m2},点q的E领域中有点{q,m},点o的E领域中有点{o,p,s},点s的E领域中有点{o,s,s1}.

那么核心对象有p,m,o,s(q不是核心对象，因为它对应的E领域中点数量等于2，小于MinPts=3)；

点m从点p直接密度可达，因为m在p的E领域内，并且p为核心对象；

点q从点p密度可达，因为点q从点m直接密度可达，并且点m从点p直接密度可达；

点q到点s密度相连，因为点q从点p密度可达，并且s从点p密度可达。

**三  算法描述**

|  |
| --- |
| 算法：DBSCAN  输入：E — 半径        MinPts — 给定点在E领域内成为核心对象的最小领域点数        D — 集合  输出：目标类簇集合  方法：repeat  1)       判断输入点是否为核心对象  2)       找出核心对象的E领域中的所有直接密度可达点        util 所有输入点都判断完毕        repeat           针对所有核心对象的E领域所有直接密度可达点找到最大密度相连对象集合，           中间涉及到一些密度可达对象的合并。        Util 所有核心对象的E领域都遍历完毕 |

**四  算法实现**

package com.dbscan;  
  
public class DataPoint {  
    private String dataPointName; // 样本点名  
    private double dimensioin[]; // 样本点的维度  
    private boolean isKey; //是否是核心对象  
  
    public DataPoint(){  
  
    }  
  
    public DataPoint(double[] dimensioin,String dataPointName,boolean isKey){  
         this.dataPointName=dataPointName;  
         this.dimensioin=dimensioin;  
         this.isKey=isKey;  
    }

**}**

**------------**

package com.dbscan;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
  
public class Cluster {  
    private List<DataPoint> dataPoints = new ArrayList<DataPoint>(); // 类簇中的样本点  
    private String clusterName; //簇名  
  
    public List<DataPoint> getDataPoints() {  
        return dataPoints;  
    }  
  
    public void setDataPoints(List<DataPoint> dataPoints) {  
        this.dataPoints = dataPoints;  
    }  
  
    public String getClusterName() {  
        return clusterName;  
    }  
  
    public void setClusterName(String clusterName) {  
        this.clusterName = clusterName;  
    }  
  
}

**------------**

package com.dbscan;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ClusterAnalysis {  
  
     
    public List<Cluster> doDbscanAnalysis(List<DataPoint> dataPoints,  
            double radius, int ObjectNum) {  
         List<Cluster> clusterList=new ArrayList<Cluster>();  
         for(int i=0; i<dataPoints.size();i++){  
             DataPoint dp=dataPoints.get(i);  
             List<DataPoint> arrivableObjects=isKeyAndReturnObjects(dp,dataPoints,radius,ObjectNum);  
             if(arrivableObjects!=null){  
                  Cluster tempCluster=new Cluster();  
                  tempCluster.setClusterName("Cluster "+i);  
                  tempCluster.setDataPoints(arrivableObjects);  
                  clusterList.add(tempCluster);  
             }  
         }  
  
         for(int i=0;i<clusterList.size();i++){  
             for(int j=0;j<clusterList.size();j++){  
                  if(i!=j){  
                      Cluster clusterA=clusterList.get(i);  
                      Cluster clusterB=clusterList.get(j);  
  
                      List<DataPoint> dpsA=clusterA.getDataPoints();  
                      List<DataPoint> dpsB=clusterB.getDataPoints();  
  
                      boolean flag=mergeList(dpsA,dpsB);  
                      if(flag){  
                          clusterList.set(j, new Cluster());  
                      }  
                  }  
             }  
         }  
  
         return clusterList;  
    }  
  
     
  
    public void displayCluster(List<Cluster> clusterList){  
        if(clusterList!=null){  
            for(Cluster tempCluster:clusterList){  
                if(tempCluster.getDataPoints()!=null&&tempCluster.getDataPoints().size()>0){  
                    System.out.println("----------"+tempCluster.getClusterName()+"----------");  
                    for(DataPoint dp:tempCluster.getDataPoints()){  
                        System.out.println(dp.getDataPointName());  
                    }  
                }  
            }  
        }  
    }  
  
     
    private double getDistance(DataPoint dp1,DataPoint dp2){  
        double distance=0.0;  
        double[] dim1=dp1.getDimensioin();  
        double[] dim2=dp2.getDimensioin();  
        if(dim1.length==dim2.length){  
            for(int i=0;i<dim1.length;i++){  
                double temp=Math.pow((dim1[i]-dim2[i]), 2);  
                distance=distance+temp;  
            }  
            distance=Math.pow(distance, 0.5);  
            return distance;  
        }  
        return distance;  
    }  
  
     
   private List<DataPoint> isKeyAndReturnObjects(DataPoint dataPoint,List<DataPoint> dataPoints,double radius,int ObjectNum){  
       List<DataPoint> arrivableObjects=new ArrayList<DataPoint>(); //用来存储所有直接密度可达对象  
  
       for(DataPoint dp:dataPoints){  
          double distance=getDistance(dataPoint,dp);  
          if(distance<=radius){  
              arrivableObjects.add(dp);  
          }  
       }  
  
       if(arrivableObjects.size()>=ObjectNum){  
           dataPoint.setKey(true);  
           return arrivableObjects;  
       }  
  
       return null;  
   }  
  
    
   private boolean isContain(DataPoint dp,List<DataPoint> dps){  
      boolean flag=false;  
      String name=dp.getDataPointName().trim();  
      for(DataPoint tempDp:dps){  
         String tempName=tempDp.getDataPointName().trim();  
         if(name.equals(tempName)){  
             flag=true;  
             break;  
         }  
      }  
  
      return flag;  
   }  
  
    
   private boolean mergeList(List<DataPoint> dps1,List<DataPoint> dps2){  
       boolean flag=false;  
  
       if(dps1==null||dps2==null||dps1.size()==0||dps2.size()==0){  
           return flag;  
       }  
  
       for(DataPoint dp:dps2){  
          if(dp.isKey()&&isContain(dp,dps1)){  
             flag=true;  
             break;  
          }  
       }  
  
       if(flag){  
           for(DataPoint dp:dps2){  
              if(!isContain(dp,dps1)){  
                  DataPoint tempDp=new DataPoint(dp.getDimensioin(),dp.getDataPointName(),dp.isKey());  
                  dps1.add(tempDp);  
              }  
           }  
       }  
  
  
       return flag;  
   }  
  
   public static void main(String[] args){  
       ArrayList<DataPoint> dpoints = new ArrayList<DataPoint>();  
        
       double[] a={2,3};  
       double[] b={2,4};  
       double[] c={1,4};  
       double[] d={1,3};  
       double[] e={2,2};  
       double[] f={3,2};  
  
       double[] g={8,7};  
       double[] h={8,6};  
       double[] i={7,7};  
       double[] j={7,6};  
       double[] k={8,5};  
  
       double[] l={100,2};//孤立点  
  
  
       double[] m={8,20};  
       double[] n={8,19};  
       double[] o={7,18};  
       double[] p={7,17};  
       double[] q={8,21};  
  
       dpoints.add(new DataPoint(a,"a",false));  
       dpoints.add(new DataPoint(b,"b",false));  
       dpoints.add(new DataPoint(c,"c",false));  
       dpoints.add(new DataPoint(d,"d",false));  
       dpoints.add(new DataPoint(e,"e",false));  
       dpoints.add(new DataPoint(f,"f",false));  
  
       dpoints.add(new DataPoint(g,"g",false));  
       dpoints.add(new DataPoint(h,"h",false));  
       dpoints.add(new DataPoint(i,"i",false));  
       dpoints.add(new DataPoint(j,"j",false));  
       dpoints.add(new DataPoint(k,"k",false));  
  
       dpoints.add(new DataPoint(l,"l",false));  
  
       dpoints.add(new DataPoint(m,"m",false));  
       dpoints.add(new DataPoint(n,"n",false));  
       dpoints.add(new DataPoint(o,"o",false));  
       dpoints.add(new DataPoint(p,"p",false));  
       dpoints.add(new DataPoint(q,"q",false));  
  
       ClusterAnalysis ca=new ClusterAnalysis();  
       List<Cluster> clusterList=ca.doDbscanAnalysis(dpoints, 2, 4);  
       ca.displayCluster(clusterList);  
  
   }  
}  
  
     
}