

采用有效邻近点和适应密度的密度聚类算法

闫强强,张 敏,荀亚玲

(太原科技大学 计算机科学与技术学院,山西 太原 030024)

摘 要:密度聚类作为一类重要的聚类分析方法,具有无需预先指定类簇数,可识别任意形状聚类族等优点,但在计算密度的过程中, K 近邻或邻域半径的选取对聚类效果具有较大的影响,且当数据集中存在类簇间距相差较大的情况时,密度聚类无法自适应类簇中数据对象密度变换,导致聚类效果与实际存在较大误差。针对现有密度聚类分析存在的不足,利用有效邻近点和适应密度分布,提出了一种密度聚类分析算法。该算法首先通过相对距离确定伸缩半径,定义了数据对象的有效邻近点,并有效地克服了近邻值 K 选取对聚类效果的影响;其次,计算核心点和边界点阈值,依据有效邻近点,并确定类簇中的核心区域数据对象,有效地改善了聚类分析效率;然后,调整簇内有效距离,改善了类簇密度分布不均匀、类簇间距离过大等问题;最后,在人工和UCI数据集上验证了该算法的有效性。

关键词:密度聚类;伸缩半径;有效邻近点;适应密度分布;相对距离

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2022)09-0014-09

doi:10.3969/j.issn.1673-629X.2022.09.003

A Density Clustering Algorithm Based on Effective Neighboring Points and Adaptive Density Distribution

YAN Qiang-qiang, ZHANG Min, XUN Ya-ling

(School of Computer Science and Technology, Taiyuan University of Science and Technology,
Taiyuan 030024, China)

Abstract: As an important cluster analysis method, density clustering has the advantages of unspecified number of cluster in advance and clustering with arbitrary shapes can be discovered. However, in the process of calculating the density, there is an important influence on the clustering due to the selection of K -nearest neighboring or Eps. When cluster spacing vary a lot in the datasets, the density clustering is unable to adapt to the data object density transformation in the clusters, which leads to a large deviation between the clustering and the reality datasets. In order to overcome shortcomings of existing density cluster analysis, a density clustering algorithm is proposed by using effective neighboring points and adaptive density distribution. Firstly, the telescopic radius is determined by the relative distance, the effective neighboring points of the data object is defined, and the influence of the selection of the nearest neighbor value K on the clustering effect is overcome. Secondly, core point and boundary point threshold are calculated using the relative distance, so that core area objects in the cluster are determined according to the effective neighboring points, which effectively improves the efficiency of cluster analysis. Thirdly, uneven density distribution and large distance between clusters are improved by adjusting the effective distance within the cluster. In the end, the effectiveness of the proposed algorithm is validated on artificial and UCI datasets.

Key words: density clustering algorithm; telescopic radius; effective neighboring points; adaptive density distribution; relative distance

0 引 言

聚类分析是根据对象之间的相似性,将数据集划分为若干类簇,使类内对象具有高度相似性,而类间对象具有差异性的类簇^[1-2],并已成功应用于故障检测、异常值检测、模式识别以及信息检索等领域^[3-6]。聚类分析主要分为基于划分的聚类^[7-8]、基于层次的聚

类^[9]、基于密度的聚类^[10]和基于子空间的聚类^[11]等方法。由于基于密度的聚类分析方法具有不需要提前预估簇类数量,可以发现任意形状类簇和识别噪声点等优点,已成为聚类分析中的研究热点之一。

密度聚类是一类重要的聚类分析方法,利用数据集中数据对象的邻域密度信息,不需要预先估计类簇

收稿日期:2021-09-27

修回日期:2022-01-31

基金项目:国家青年科学基金项目(61602335);山西省自然科学基金(201901D211302)

作者简介:闫强强(1997-),男,硕士研究生,研究方向为数据挖掘及应用;通讯作者:荀亚玲(1980-),女,博士,副教授,研究方向为数据挖掘与并行计算。

数目,就可以发现任意形状类簇,但随着数据维度的增加,不同类簇之间密度的差异也随之增大,会使聚类效果变差^[12];在大多数密度聚类分析中,近邻 K 值的选取对聚类效果影响极大。该文采用邻近点信息刻画数据对象邻近点的疏密程度,提出了一种基于有效邻近点和适应密度分布的密度聚类算法,并利用数据对象邻近点的相对距离来计算伸缩半径,判断数据对象的有效邻近点,有效减少了对数据对象密度影响较小邻近点的计算,改善了类簇密度分布不均匀、类簇间距离过大导致聚类效果差的问题。

其主要贡献如下:

- (1) 利用相对距离,定义了有效邻近点;
- (2) 提出了一种调整簇内有效距离的策略;
- (3) 提出了一种基于有效近邻点和适应密度分布的密度聚类分析算法。

1 相关工作

密度聚类是依据数据对象的密度进行划分,当一个区域内数据对象的密度大于某个阈值时,将数据对象添加到相近的聚类簇中的聚类。密度聚类分析主要分为两个研究方向:DBSCAN (density-based spatial clustering of application with noise) 算法和 DPC 算法 (clustering by fast search and find of density peaks)^[13]。DBSCAN 利用数据对象的密度信息,计算核心点和边界点实现聚类任务^[14],因其具有聚类效率高,对不规则的数据集有良好的聚类效果等优点,已吸引了众多学者关注^[15]。

DBSCAN 作为一类有效的密度聚类分析方法,其典型研究成果有:周水庚等人提出了一种基于数据分区的聚类算法 PDBSCAN^[16],该算法依据数据集的分布特性,将整个数据集分为若干个区域,分别对不同区域进行聚类,但聚类效果仍受到邻域半径 Eps 的影响;Wu Y 等人提出了一种基于 LSH 的线性 DBSCAN 算法^[17],该算法构建 LSH (局部敏感哈希) 来优化最近邻域搜索过程,聚类效率得到改善,但数据集的聚类效果没有提升,并且在大型数据集上聚类效果不佳;武佳薇提出了一种邻域平衡密度聚类算法 bDBSCAN^[18],该算法利用核心对象的邻域平衡性,将平衡密度可达的对象划分到同一簇完成聚类,有效排除边界稀疏对象的噪声点,提高了聚类精度,但算法新引入两个参数,在聚类前需要对参数进行多次调整;Cassisi C 等人提出了一种增强基于密度的聚类算法 IS-DBSCAN^[19],该算法利用数据对象的反向最近邻,执行密度聚类,从而有效识别不同密度的类簇;Kumar 等人提出了一种通过 Groups 加速邻居搜索的快速 DBSCAN 聚类算法^[20],该算法有效提高了对大型数据集的计算效率,

但聚类效果受参数 Eps 和 $MinPts$ 影响较大;于彦伟等人提出一种面对位置大数据的快速密度聚类算法 CB-SCAN^[21],该算法通过确定核心点和密度相连关系,减少了高密度区域的距离计算,提高了聚类效果;Lv Y 等人提出一种新的基于影响空间和边界点检测的密度聚类算法 (ISB-DBSCAN)^[22],该算法改进局部敏感哈希方法,实现了最近邻的快速查找,有效实现区分边界对象和噪声对象;Bryant 等人提出一种基于密度的反向最近邻密度聚类算法 (RNN-DBSCAN)^[23],该算法仅使用参数 K 实现对不同密度类簇的区分,但是与 IS-DBSCAN、ISB-DBSCAN 算法相同,聚类效果受近邻值 K 影响较大;Gholizadeh 等人提出一种改进的大数据密度聚类算法 K-DBSCAN^[24],该算法通过对数据集进行划分聚类,降低 DBSCAN 执行的计算负担,使算法可以有效地对大数据集进行聚类分析,但此算法的聚类精度没有发生改变。

但上述算法对存在密度有较大差异的数据集进行聚类分析时,参数邻域半径 Eps 和最小包含点 $MinPts$ 无法自动适应数据对象密度变换,影响聚类分析效果;且在 K 近邻与密度聚类相结合的算法中,由于近邻数 K 影响数据对象的密度,导致聚类效果对 K 的抗干扰性较差。

针对上述不足,利用相对距离计算数据对象的有效邻近点,有效解决聚类效果受近邻 K 影响的情况,并通过调整簇内有效距离的策略来适应数据对象密度的变换。

2 密度聚类与核心点

密度聚类分析具有抗噪性强、效率高,以及可发现任意形状类簇等优点,已得到广泛应用。参照文献 [14],相关概念和公式描述如下:

对于给定任意数据集 $DB = \{x_1, x_2, \dots, x_n\}$, 其中, n 表示数据对象的数量。

以数据对象 x_i 为中心, Eps 为半径的区域内包含的数据对象,定义为 Eps 邻域 $N_{eps}(x_i)$, 如公式 (1) 所示:

$$N_{eps}(x_i) = \{x_j \mid \text{dist}(x_i, x_j) < eps\} \quad (1)$$

其中, $\text{dist}(x_i, x_j)$ 为数据对象 x_i 和 x_j 之间的欧氏距离, x_j 为数据集中的数据对象。

当数据对象 x_i 在邻域半径 Eps 内的密度值 Num 满足公式 (2) 时,则称数据对象 x_i 为核心点。其中 Num 为数据对象 x_i 在 Eps 邻域内数据对象的个数:

$$Num \geq MinPts \quad (2)$$

当数据对象 x_i 的密度值 Num 不满足公式 (2),但其邻域内至少有一核心点时,则称数据对象 x_i 为边界点。

3 基于有效邻近点和适应密度分布的密度聚类

3.1 有效邻近点及相关定义

对于任意数据对象 $x_i \in DB$, x_i 的邻近点集 $N(x_i)$ 可定义如下:

$$N(x_i) = \{data_1, data_2, \dots\} \quad (3)$$

其中, $data_1, data_2, \dots$ 分别为数据对象 x_i 的最邻近点, 次邻近点, \dots , 其欧氏距离分别表示为 d_1, d_2, \dots , 且按从小到大排序。

数据对象 x_i 的邻近点与最邻近点相对距离 σ_{21} , 可定义如下:

$$\sigma_{j1} = d(x_i, data_j) / d(x_i, data_1) \quad (4)$$

其中, $j=2, 3, \dots, n$; x_i 为数据对象, $data_j$ 为数据对象 x_i 的第 j 个邻近点, $data_1$ 为数据对象 x_i 的最邻近点。

在密度聚类的过程中, 通常选取邻域半径 Eps 或数据对象 K 个近邻点的方式计算数据对象的密度。在图 1 所示数据集中, 类簇中数据对象的邻近点距离存在较大差异。Eps 选取定长的方式计算数据对象密度, 但 Eps 无法适应同时存在稀疏类簇和稠密类簇数据集中数据对象近邻点间距的变化, 在图 1 中选取两个数据对象, 无论 Eps 取何值, 均无法同时计算数据对象的密度; 近邻值 K 给定确定近邻点个数的方式计算数据对象密度, 但并非所有近邻点都对数据对象密度有较大的影响, 当选取 5 作为近邻值 K 对密度进行密度分析时, 稀疏区域中的数据对象有 3 个近邻点在其他类簇中, 且邻近点距数据对象越远, 对密度的影响越小。利用 Eps 或近邻值 K 均无法准确对类簇中密度差异较大的数据对象进行计算。

因此, 可采用伸缩半径适应在同一类簇和不同类簇中数据对象的密度, 伸缩半径依据数据对象邻近点特性的计算, 无需人为指定或通过数据集判断, 具备伸缩性, 可有效避免 Eps 或近邻值 K 无法同时适应密度差异较大数据集的情况。

为判断数据对象的伸缩半径, 可利用相对距离判断数据对象邻近点的距离关系, 相对距离是不同数据对象相较于参考点的绝对距离, 不因数据对象邻近点距离差异而改变。在不同数据对象的邻近点中, 对密度的影响随着距离的增大而降低, 其中, 最邻近点、次邻近点对数据对象密度影响最大。为判断对数据对象密度影响最大邻近点的相对距离, 选取最邻近点作为参考点, 计算数据对象次邻近点与最邻近点的相对距离 σ_{21} 。图 2 展示了图 1 所示数据集中相对距离 σ_{21} 的分布。由图 2 可知, 尽管不同类簇中数据对象邻近点距离有较大差异, 但其相对距离 σ_{21} 存在较高的相似度, 在邻近点距离分布较大数据对象中相对距离 σ_{21}

多分布于 1.0、1.1、1.2, 不因数据对象邻近点距离而改变, 可以较好表示对数据对象密度影响最大邻近点的相对距离。

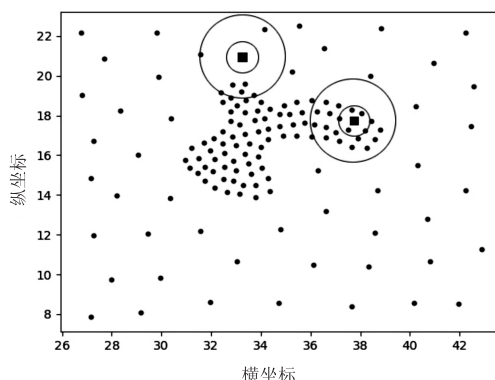


图 1 二维数据集

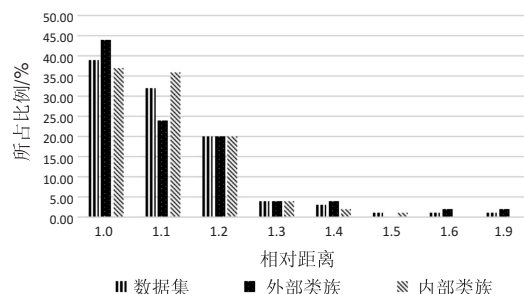


图 2 相对距离分布

因最邻近点对数据对象密度影响最大, 将最邻近点距离作为伸缩半径的基长, 与相对距离 σ_{21} 相结合即为数据对象的伸缩半径, 数据对象邻近点的距离随着最邻近点距离而改变, 使伸缩半径具有伸缩性, 可以适应不同数据对象邻近点间距离的改变。利用伸缩半径, 克服无法识别数据集中存在密度分布不均匀数据对象的情况。最邻近点为对密度影响最大的邻近点, 以最邻近点的距离为基长, 在伸缩半径范围内的邻近点与最邻近点对数据对象密度的影响相近, 即伸缩半径范围内邻近点均对数据对象密度有较大影响, 将伸缩半径范围内的邻近点定义为数据对象 x_i 有效邻近点 $ENN(x_i)$ 。

取 σ_{21} 小数点后一位, 以最邻近点的距离为基长, 当 σ_{21} 为 1.0, 对数据对象密度影响较大邻近点的距离仅为最邻近点距离, 不能有效判断邻近点对数据对象密度影响较大的邻近点, 故不考虑 σ_{21} 为 1.0 的相对距离。为统一判断数据集中数据对象的有效邻近点, 选取多次重复出现 σ_{21} 的中位数作为判断有效邻近点标准 ϑ , 选取 ϑ 可以有效体现数据集中对多数数据对象密度影响较大邻近点的相对距离, 可以有效适应不同类簇中对数据对象密度影响较大邻近点距离的差异, 有效邻近点 $ENN(x_i)$ 定义如下:

$$ENN(x_i) = \{data_j \mid d_j < \vartheta \times d_1\} \quad (5)$$

其中, d_1 为与 x_i 最近的欧氏距离值, d_j 为 x_i 第 j 个最近的欧氏距离值, $data_j$ 为 x_i 的第 j 个邻近点。

位于类簇中心的数据对象,有效邻近点较多,位于类簇边缘的数据对象,有效邻近点较少,利用 ENN(x_i) 的个数判断数据集中的核心点与边界点,当数据对象 x_i 的有效邻近点满足公式(6)时,即可判定数据对象为核心点,核心点定义如下:

$$|ENN(x_i)| \geq \text{corethreshold} \quad (6)$$

其中, corethreshold 为核心点阈值。

当数据对象 x_i 的有效邻近点满足公式(7)时,即可判定数据对象为边界点,边界点定义如下:

$$|ENN(x_i)| \geq \text{borderthreshold} \quad (7)$$

其中, $|ENN(x_i)|$ 为数据对象 x_i 的有效邻近点个数, borderthreshold 为边界点阈值,其计算方法由算法 CalculateIndex 给出。

数据对象 x_i 的有效距离 $\text{EDist}(x_i)$ 为 x_i 有效邻近点的平均距离,可定义如下:

$$\text{EDist}(x_i) = \frac{1}{|ENN(x_i)|} \sum_{j \in ENN} d(x_i, data_j) \quad (8)$$

其中, $|ENN(x_i)|$ 为数据对象 x_i 有效邻近点的个数, $data_j$ 为 x_i 的第 j 有效邻近点。

类簇中数据对象的密度仍存在差异,为适应类簇中数据对象密度变换,通过调整簇内有效距离的策略对有效距离进行调整。 $\text{EDist}(x_i)$ 为 x_i 有效邻近点的平均距离,其中有效邻近点均为对 x_i 密度影响较大的邻近点,若其邻近点在 $\text{EDist}(x_i)$ 条件下仍存在有效邻近点,则说明其有效邻近点与数据对象 x_i 的有效邻近点相似。依次遍历 x_i 邻近点中存在有效邻近点的数据对象,分别计算其有效邻近点,将其存放于集合 C 中,簇内有效距离 C_Dist 调整如式(9)所示:

$$C_Dist = \frac{\sum_{i=0}^{|C|} \text{EDist}_i}{|C|} \quad (9)$$

其中, $|C|$ 为存在有效邻近点数据对象的个数,其有效邻近点的判定为:

$$ENN(x_i) = \{data_j \mid d_j < \partial \times \text{EDist}(x_i)\}$$

在对簇进行扩展时,类簇中数据对象 x_i 的有效邻近点的判定如式(10)所示:

$$\text{CENN}(x_i) = \{data_j \mid d_j < \partial \times C_Dist\} \quad (10)$$

其中, C_Dist 为簇内有效距离, $data_j$ 为 x_i 的第 j 个有效邻近点, d_j 为与 x_i 第 j 个最近的欧氏距离值。

利用簇内有效距离判断数据对象的有效邻近点并对其扩展过程中会出现将单个类簇划分为多个小类簇的情况。对此,算法引入簇对象阈值 α 对类簇划分进行约束,当数据对象扩展完成后,若扩展的类簇在数据集中所占比例低于 α ,则不判定类簇的划分。 α 是根据

类簇在数据集中数据对象所占比例决定。

3.2 基于有效邻近点和适应密度分布的密度聚类算法

综上所述,基于有效邻近点和适应密度分布的密度聚类分析基本思想为:构建 KD 树,利用公式(4),选取判断有效邻近点的标准 ∂ ,通过公式(5)判断数据对象的有效邻近点;利用公式(6)、(7)完成对数据集中核心点和边界点的判断;利用公式(9)调整簇内有效距离,适应类簇扩展过程中数据对象的密度变换。其伪代码描述如下:

算法:ENND(A density clustering algorithm based on effective nearest neighbor distribution)

输入:数据集 DB,簇对象阈值 α

输出:聚类结果 clustering

1. 利用 KD 树,计算数据对象 x_i 最邻近的距离,从小到大对数据集重新排序

2. $\forall x \in D: x = \text{Unassign} //$ 添加未聚类标签

3. CalculateIndex($D, \partial, \text{corethreshold}, \text{borderthreshold}$)

4. for $i = 1$ to $|D|$ {

5. if $x_i = \text{Unassign}$ Then

6. ExpandCluster($\partial, x_i, \text{corethreshold}, \text{borderthreshold}, \text{cluster}$)

7. if the number of cluster $> \alpha * |DB|$ Then {

8. cluster = assign // 保存类簇 cluster

9. else { cluster = assigned // 标记未聚类数据对象

10. AssignData(assigned, cluster, clustering)

11. return clustering

12. end ENND

CalculateIndex 算法利用数据对象的相对距离计算数据集的 ∂ 、核心点阈值 corethreshold 以及边界点阈值 borderthreshold 。

算法 1: CalculateIndex($D, \partial, \text{corethreshold}, \text{borderthreshold}$)

输入:数据集 D

输出: ∂ , 核心区阈值 corethreshold , 边界区阈值 borderthreshold

1. for $i = 1$ to $|D|$ {

2. calculate σ for $x_i // (\sigma$ 取小数点后一位)

3. calculate the number of $\{\sigma_1 = \sigma_{21}\} (k = 1, 2, \dots) = \text{core}$

4. calculate the number of {only bigger than border $\sigma_{21}\} (k = 1, 2, \dots)$

5. calculate the median of the two numbers that σ_{21} most frequently for ∂

6. $\text{borderthreshold} = \text{the count max of border}$

7. $\text{corethreshold} = \text{count max of core and } \text{corethreshold} \geq \text{borderthreshold}$

8. if corethreshold isn't exist then $\text{corethreshold} = \text{borderthreshold}$

9. return $\partial, \text{borderthreshold}, \text{corethreshold}$

ExpandCluster 算法利用 CalculateIndex 算法中 ∂ 、corethreshold、borderthreshold, 判断核心点和边界点, 将未聚类的数据对象扩展为类簇。

算法 2: ExpandCluster(x_i , ∂ , corethreshold, borderthreshold, cluster)

输入: 数据对象 x_i , ∂ , 核心区阈值 corethreshold, 边界区阈值 borderthreshold

输出: 类簇 cluster

```

1. calculate EDist(  $x_i$  )//计算数据对象  $x_i$  有效距离
2. initialize empty list for stack and C
3. put  $x_i$  in stack
4. while stack is not empty {
5.  $x_j = \text{stack.pop}()$ 
6. calculate ENN(  $x_i$  ) ENN(  $x_i$  ) = { dataj |  $d_j < \partial \times \text{EDist}(x_i)$  }
7. if { ENN(  $x_j$  ) exist ENN } Then { put ENN(  $x_j$  ) in stack and | C | }
8. calculate C_Dist in | C | //利用集合 C 中数据对象计算簇内有效距离
9. for  $k = 1$  to | C | {
10. count = the number of CENN(  $x_k$  )
11. if count  $\geq$  corethreshold Then { put  $x_k$  into corearea }
12. else { if count  $\geq$  borderthreshold Then { put  $x_k$  into borderarea }
    }
13. clust = corearea + borderarea //将 corearea 和 borderarea 的数据对象聚和为类簇 cluster
14. return cluster

```

AssignData 算法对未聚类成功的数据对象进行二次分配, 依次判断数据对象邻近点的归属, 将数据对象划分给邻近点归属的类簇。

算法 3: AssignData(assigned, cluster, clustering)

输入: 未聚类数据对象 assigned, 类簇 cluster

输出: 类簇 clustering

```

1. while True
2. for  $j = 1$  to | assigned | {
3. for  $i = 1$  to  $k$  (  $k = 1$  ) {
4. if {
5. assigned[  $j$  ] the datai is in cluster
6. assigned[  $j$  ] = cluster( datai )//若数据对象 assigned[  $j$  ] 的第  $i$  邻近点被分配, 则  $x_j$  划分到第  $i$  邻近点的类簇中
    } }
7. if exist assigned {  $k = k + 1$  continue }
8. else break
    }
9. 将所有簇类合并为聚类结果 clustering
10. return clustering

```

算法复杂性分析: 在 ENND 算法中, 计算数据集的邻近点, 其时间复杂度为 $O(n \log N)$, 计算核心点和边界点阈值的时间复杂度为 $O(n)$; 对数据集中所有数

据对象进行扩展, 其时间复杂度为 $O(n^2)$; 对未分配的数据对象进行二次分配, 其时间复杂度为 $O(n)$ 。因此, ENND 算法的时间复杂度为 $O(kn \log N + n + n^2 + n) \approx O(n^2)$ 。由于在数据对象的扩展过程中, 只针对数据对象的有效邻近点进行扩展, 时间复杂度应远小于 $O(n^2)$ 。

4 实验结果与分析

实验环境: Windows OS10 系统, Intel (R) Core (TM) i5CPU, 16 GB 动态随机存取存储器和 1 TB 主内存。实验数据来源于人工数据集 (<https://cs.joensuu.fi/sipu/datasets/>) 和 UCI 数据库 (<http://archive.ics.uci.edu/ml/index.php>), 其详细描述见表 1 和表 2。聚类效果评价指标: 调整兰德指数 (ARI)、归一化信息 (NMI)、Fowlkes-Mallows (F-M)、精度 (ACC)^[25]。采用 K-means^[26]、DBSCAN^[12]、OPTICS^[27]、RNNDSCAN^[23]、DADC^[28] 作为 ENND 的实验对比算法。类簇占比均采用四舍五入。

表 1 人工数据集

数据集	维度	簇数	样本数	类簇占比
Flame	2	3	240	36%; 64%
Jain	2	2	373	27%; 73%
Pathbased	2	3	300	37%; 32%; 30%
R15	2	15	600	6.67% 15 个

表 2 UCI 数据集

数据集	维度	簇数	样本数	类簇占比
data_banknote	5	2	1 372	55%; 45%
ionosphere	34	2	350	64%; 36%
iris	4	3	147	33%; 34%; 33%
libras	90	15	360	7.3% 5 个; 6.4% 10 个
optdigit	64	10	1 797	10% 10 个
seeds	4	3	210	33.3% 3 个
segmentation	19	7	2 100	14.2% 7 个
wine	13	3	178	33%; 40%; 27%

设数据集 DB 含有 n 个数据对象, m 个属性维度, 对于任意数据对象 x, y , 采用欧氏距离来表示数据对象 x 和 y 之间的距离。在对数据集进行处理时, 删除相同数据对象。

4.1 簇对象阈值对聚类效果的影响

为验证簇对象阈值 α 对聚类效果的影响, 分别选取 2%, 5%, 10%, 15%, 20% 作为 UCI 数据集的 α , 将调整兰德指数、归一化信息、Fowlkes-Mallows 和精度作为评价指标, 其实验结果如表 3 所示。

由表 3 可以看到, ENND 聚类效果与类簇在数据集中占比及数据集的分布特征密切相关。当类簇占比

表3 α 对聚类效果的影响

数据集	α 阈值	ARI	AMI	F-M	ACC	类簇数	数据集	α 阈值	ARI	AMI	F-M	ACC	类簇数
data_banknote	2% (30)	0.191	0.386	0.439	0.023	18	optdigit	2% (36)	0.389	0.674	0.463	0.043	27
	5% (68)	0.283	0.473	0.534	0.052	9		5% (90)	0.596	0.741	0.635	0.006	12
	10% (135)	0.247	0.390	0.533	0.116	5		10% (180)	0.396	0.637	0.544	0.098	4
	15% (202)	0.656	0.633	0.829	0.905	2		15% (270)	0.359	0.572	0.508	0.011	4
	20% (270)	0.656	0.633	0.829	0.905	2		20% (360)	0.317	0.567	0.503	0	3
ionosphere	2% (7)	0.049	0.213	0.239	0.023	25	seeds	2% (4)	0.131	0.390	0.309	0.095	25
	5% (18)	0.080	0.230	0.311	0.057	13		5% (10)	0.225	0.471	0.407	0.086	11
	10% (35)	0.176	0.307	0.451	0.12	7		10% (21)	0.411	0.557	0.578	0.095	5
	15% (53)	0.146	0.234	0.448	0.189	5		15% (32)	0.707	0.665	0.804	0.590	3
	20% (70)	0.209	0.299	0.516	0.254	4		20% (42)	0.707	0.665	0.804	0.590	3
iris	2% (3)	0.195	0.479	0.388	0.068	20	segmentation	2% (42)	0.256	0.523	0.354	0.056	24
	5% (7)	0.425	0.603	0.593	0.068	10		5% (104)	0.358	0.551	0.443	0.256	14
	10% (15)	0.882	0.867	0.921	0.959	3		10% (209)	0.563	0.641	0.622	0.184	8
	15% (22)	0.882	0.867	0.921	0.959	3		15% (312)	0.495	0.595	0.593	0.035	5
	20% (29)	0.882	0.867	0.921	0.959	3		20% (417)	0.259	0.378	0.452	0.129	3
Libras	2% (7)	0.245	0.487	0.291	0.063	22	wine	2% (4)	0.116	0.275	0.287	0.034	21
	5% (17)	0.321	0.538	0.400	0.145	9		5% (9)	0.187	0.305	0.372	0.067	11
	10% (33)	0.223	0.474	0.363	0.039	5		10% (18)	0.262	0.387	0.457	0.169	6
	15% (50)	0.119	0.326	0.303	0.036	3		15% (27)	0.439	0.424	0.629	0.039	3
	20% (66)	0.067	0.197	0.272	0.39	2		20% (36)	0.435	0.415	0.625	0.045	3

分布均匀时(例如:iris、optdigit、seeds 数据集), α 值越大, 聚类效果越好, 其主要原因是当 α 值增加, ENND 利用有效邻近点对簇内有效距离的调整, 有效地适应于类簇中数据对象的密度变换, 从而判断类簇中的核心点和边界点; 当类簇占比分布不均匀时(例如: data_banknote、ionosphere、Libras、wine 数据集), α 值越小, 聚类效果也随之变差, 其主要原因是较小的簇内有效距离仅能识别类簇中部分核心点与边界点, α 减少会导致核心点与边界点满足聚类条件(α), 从而将单个类簇划分为多个小类簇, 聚类效果变差; 当 α 值大于最小类簇占比时(例如 segmentation 数据集), 聚类效果变差, 其主要原因是 α 大于最小类簇占比时, 类簇中由核心区域内的数据对象扩展的类簇无法满足聚类条件, 而对边界区域内的数据对象扩展时, 得到较大的簇内有效距离, 会将非本类簇中的数据对象错误识别, 导致聚类效果变差。

综上所述, 对于多数 UCI 数据集, 当簇对象阈值 α 未大于最小类簇占比时, 聚类效果随 α 的增大而增加, 在接近最小簇占比及大于最小簇占比时, 聚类效果变差。

4.2 聚类簇可视化

为实验验证 ENND 对不同密度、不同形状类簇的聚类效果, 采用了如表 1 所示的 4 个人工数据集。由 4.1 节可知, 当簇对象阈值 α 小于最小类簇占比时, 聚类效果较好, flame、jain、pathbased 均为类簇占比不均匀数据集, 选取 α 为 20%; R15 类簇占比分布均匀数据集, 分别选取 α 为 5%。为得到其他算法最好的聚类

效果, 依据文献[25]中调整参数方法对 K-means、DBSCAN、OPTICS 参数选取, 针对 RNN-DBSCAN、DADC, 采用改变近邻 K 取值的方式选取参数。其实验结果见图 3~图 6。

在图 3 中, Flame 数据集是由两个形状不规则的类簇构成, 除 K-means 算法外, 其他 5 种算法均可正

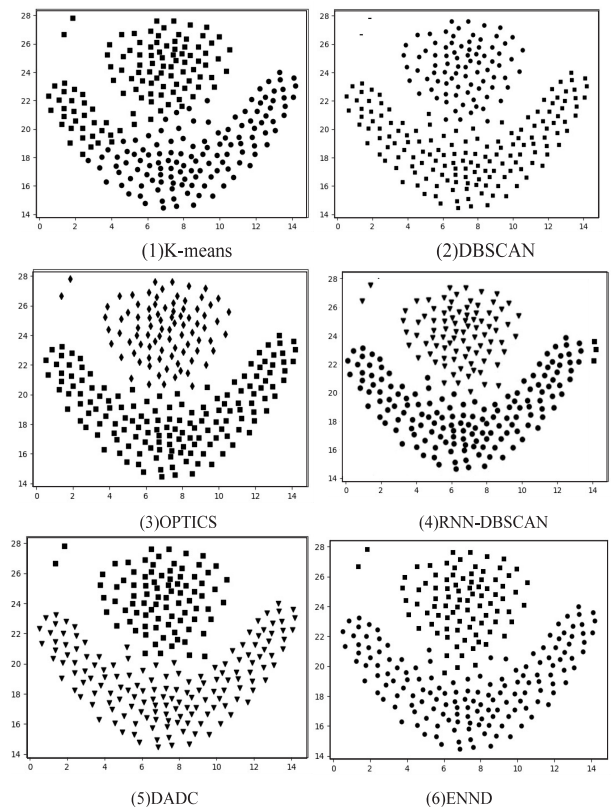


图3 6种算法在 FLame 数据集的聚类结果

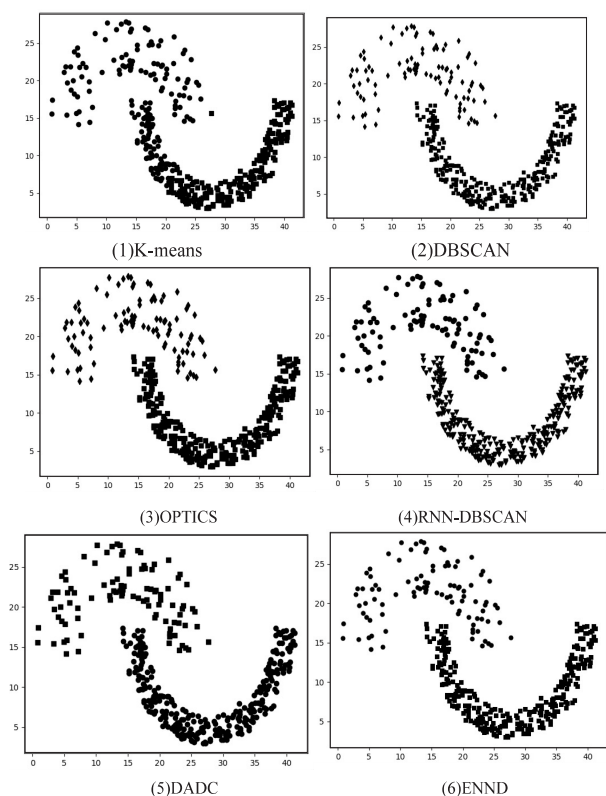


图 4 6 种算法在 Jain 数据集的聚类结果

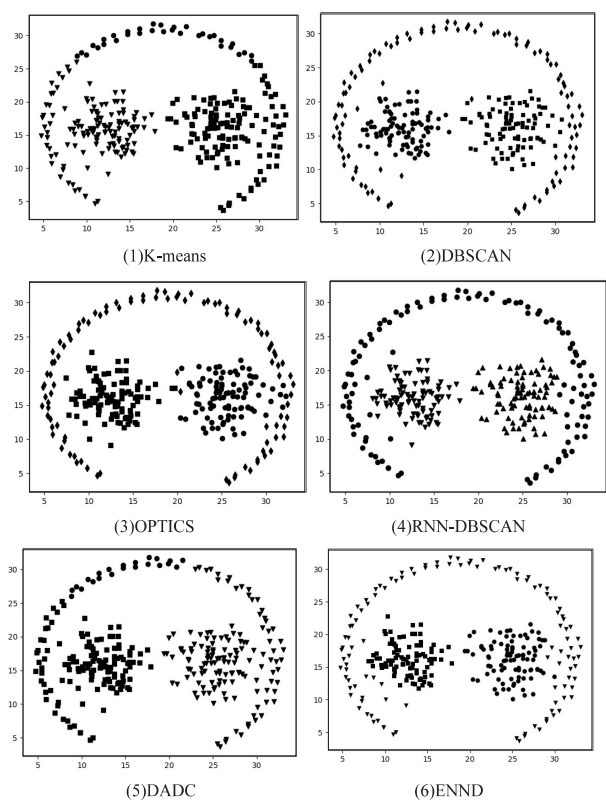


图 5 6 种算法在 Pathbased 数据集的聚类结果

确识别类簇。在图 4 中, Jain 数据集是由两个密度差异较大双月形状类簇构成, K-means 算法错误地将类簇分为两类, 其他聚类算法可以很好地识别密度分布不均匀的类簇。在图 5 中, Pathbased 数据集中相邻的类簇间存在交叉数据对象, K-means 算法无法识别

非凸状的数据集, DADC 算法由于聚类融合度 (CFD), 无法正确合并类簇, 其他 4 种算法均可以实现对数据集正确的聚类分析。在图 6 中, R15 数据集类簇具有凸状, 6 种算法均可正确识别类簇。

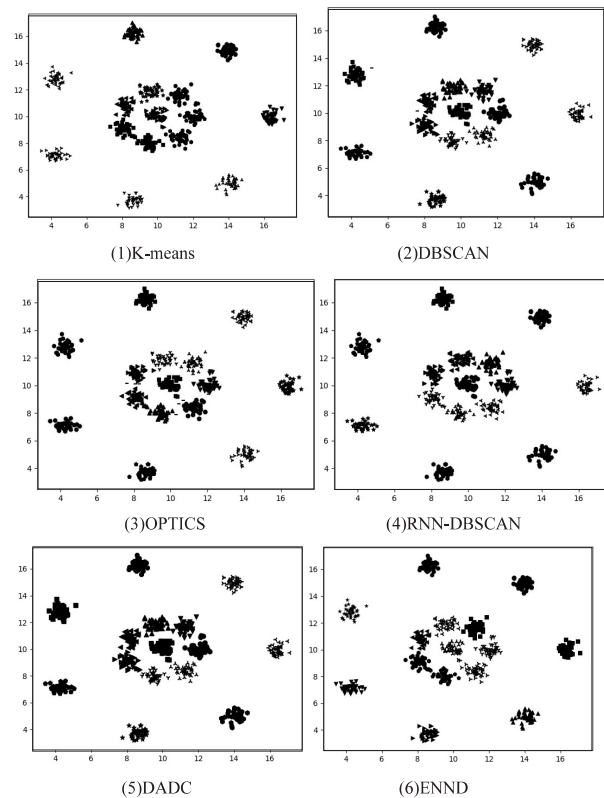


图 6 6 种算法在 R15 数据集的聚类结果

综上所述, ENND 的聚类结果优于 K-means、DADC, 与 DBSCAN、OPTICS、RNN-DBSCAN 的聚类结果基本相同, 能够有效地实现对不同密度、不同形状类簇的聚类分析, 其主要原因是 ENND 算法通过有效邻近点的判断, 无需考虑数据对象邻近点间距离的差异。

4.3 聚类性能分析

为实验验证 ENND 聚类性能, 采用调整兰德指数、归一化信息、Fowlkes-Mallows 和精度指标, 以及如表 2 所示的 UCI 数据集, 依据表 3 中的聚类效果, 选取 ENND 中的 α 阈值, 依据 4.2 节中选取参数方式, 设置对比算法的参数阈值, 其实验结果如表 4 所示。

从表 4 中可以看出, ENND 的聚类效果在多数情况下, 优于 5 种对比算法。其主要原因是: 在高维数据集中, 数据对象密度差异较大, 参数邻域半径无法适用于整体数据对象, 因此 DBSCAN、OPTICS 的聚类效果较差; RNN-DBSCAN 算法通过引入逆 K 近邻, 消除了邻域半径对数据对象密度的影响, 但忽视了数据对象邻近点间距离, 因此高维数据集的聚类分析效果较差; DADC 算法利用域自适应密度寻找密度峰值点, 但在高维数据中, 类簇间存在较多交叉点, 导致在类簇融合过程中, 聚类融合度 (CFD) 会将部分类簇错误合并,

使聚类结果比 ENND 较差;ENND 算法在类簇扩展过程中,利用有效邻近点调整簇内有效距离,使得在高维数据聚类分析中,有效识别类簇中存在密度差异较大的类簇,因此,ENND 相较于其他算法,对高维数据具

有良好的聚类分析效果,但 ENND 对 ionosphere 进行聚类时,因类簇间交叉点较多及邻近点间距离相差较小,最小簇阈值 48% 聚类效果最好。

表4 6种算法在UCI数据集上的聚类结果

数据集	算法	ARI	AMI	F-M	ACC	参数取值
data_banknote	k-means	0.047	0.030	0.551	0.610	k = 2
	DBSCAN	0.729	0.709	0.855	0.436	eps = 1.910/sample = 7
	OPTICS	0.356	0.419	0.615	0.017	sample = 22
	RNN	0.527	0.555	0.728	0	k = 19
	DADC	0.196	0.143	0.594	0.174	k = 5
	ENND	0.656	0.633	0.829	0.905	$\alpha = 15\%$ (202)
ionosphere	k-means	0.176	0.131	0.605	0.711	k = 2
	DBSCAN	0.691	0.590	0.865	0.631	sample = 1.72/ eps = 3
	OPTICS	0.239	0.187	0.635	0.463	sample = 49
	RNN	0.105	0.080	0.598	0.217	k = 7
	DADC	0.044	0.072	0.551	0	k = 20
	ENND	0.324	0.294	0.675	0.786	$\alpha = 48\%$ (167)
iris	k-means	0.737	0.759	0.825	0.898	k = 3
	DBSCAN	0.561	0.728	0.768	0.667	sample = 1.63/ eps = 5
	OPTICS	0.589	0.713	0.743	0.463	sample = 15
	RNN	0.709	0.743	0.799	0	k = 6
	DADC	0.767	0.808	0.846	0.912	k = 20
	ENND	0.882	0.867	0.921	0.959	$\alpha = 15\%$ (22)
Libras	k-means	0.333	0.554	0.381	0.097	k = 15
	DBSCAN	0.240	0.460	0.288	0.009	sample = 0.77/ eps = 1
	OPTICS	0.069	0.394	0.206	0.027	sample = 4
	RNN	0.336	0.565	0.378	0.082	k = 4
	DADC	0.366	0.594	0.421	0.136	k = 15
	ENND	0.360	0.575	0.414	0.027	$\alpha = 4\%$ (13)
optdigit	k-means	0.670	0.742	0.704	0.006	k = 10
	DBSCAN	0.661	0.793	0.695	0.350	sample = 21.0/eps = 4
	OPTICS	0.116	0.451	0.362	0.098	sample = 6
	RNN	0.747	0.821	0.775	0	k = 5
	DADC	0.620	0.781	0.671	0.014	k = 5
	ENND	0.709	0.801	0.739	0.165	$\alpha = 8\%$ (146)
seeds	k-means	0.717	0.592	0.811	0.052	3
	DBSCAN	0.499	0.522	0.649	0.2	sample = 1.2/eps = 21.00
	OPTICS	0.465	0.542	0.657	0	sample = 28
	RNN	0.607	0.648	0.729	0.252	k = 7
	DADC	0.563	0.623	0.711	0	k = 10
	ENND	0.707	0.665	0.803	0.590	$\alpha = 23\%$ (48)
segmentation	k-means	0.343	0.512	0.468	0.114	k = 7
	DBSCAN	0.397	0.620	0.528	0.458	sample = 25.95/eps = 12
	OPTICS	0.113	0.387	0.422	0.024	sample = 30
	RNN	0.503	0.648	0.572	0	k = 10
	DADC	0.375	0.506	0.459	0.027	k = 5
	ENND	0.563	0.641	0.622	0.184	$\alpha = 10\%$ (209)
wine	k-means	0.371	0.423	0.584	0.370	k = 3
	DBSCAN	0.402	0.413	0.670	0	sample = 75.95/ eps = 23
	OPTICS	0.377	0.404	0.660	0	sample = 30
	RNN	0.409	0.388	0.607	0.489	k = 28
	DADC	0.166	0.297	0.349	0.152	k = 5
	ENND	0.411	0.438	0.590	0.051	$\alpha = 13\%$ (23)

5 结束语

提出了一种基于有效邻近点和适应密度分布的密度聚类算法,通过相对距离计算伸缩半径,判断数据对象的有效邻近点,计算簇内有效距离来适应簇数据对象密度变化,有效解决了在高维数据集中,类簇密度差异较大导致聚类效果差以及聚类效果受近邻 K 影响较大的问题。但算法在对大数据聚类分析中,聚类效率较差。故为了适应大数据聚类分析任务,在 Spark 集群环境下,基于有效邻近点和适应密度分布的并行密度聚类分析是下一步研究工作。

参考文献:

- [1] WANG S, EICK C F. MR-SNN: design of parallel shared nearest neighbor clustering algorithm using MapReduce [C]//2017 IEEE 2nd international conference on big data analysis (ICBDA). Beijing: IEEE, 2017: 312–315.
- [2] ZHANG R, DU T, QU S, et al. Adaptive density-based clustering algorithm with shared KNN conflict game [J]. Information Sciences, 2021, 565: 344–369.
- [3] 崔一辉, 宋伟, 王占兵, 等. 一种基于格的隐私保护聚类数据挖掘方法 [J]. 软件学报, 2017, 28(9): 2293–2308.
- [4] BENYAO C, LICHENG R, JIAN Y, et al. Elevator traffic pattern recognition based on density peak clustering [C]//2018 IEEE international conference of safety produce informatization (HCSPI). Chongqing: IEEE, 2018: 587–590.
- [5] QIANG L, CHEN W, CHEN F, et al. Fault detection of motorized spindle based on clustering by density peaks [C]//2018 12th international conference on reliability, maintainability, and safety (ICRMS). Shanghai: IEEE, 2018: 292–296.
- [6] TAO Y, PI D. Unifying density-based clustering and outlier detection [C]//2009 second international workshop on knowledge discovery and data mining. Moscow: IEEE, 2009: 644–647.
- [7] DATTA S, GIANNELLA C, KARGUPTA H. Approximate distributed k-means clustering over a peer-to-peer network [J]. IEEE Transactions on Knowledge and Data Engineering, 2008, 21(10): 1372–1388.
- [8] 余冬华, 郭茂祖, 刘扬, 等. 基于距离不等式的 K-means 聚类算法 [J]. 软件学报, 2017, 28(12): 3115–3128.
- [9] RODRIGUES P P, GAMA J, PEDROSO J. Hierarchical clustering of time-series data streams [J]. IEEE Transactions on Knowledge and Data Engineering, 2008, 20(5): 615–627.
- [10] WANG X F, HUANG D S. A novel density-based clustering framework by using level set method [J]. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(11): 1515–1531.
- [11] 庞宁, 张继福, 秦啸. 一种基于多属性权重的分类数据子空间聚类算法 [J]. 自动化学报, 2018, 44(3): 517–532.
- [12] 邓定胜. 一种改进的 DBSCAN 算法在 Spark 平台上的应用 [J]. 计算机科学, 2020, 47(S02): 425–429.
- [13] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks [J]. Science, 2014, 344(6191): 1492–1496.
- [14] ESTER M, KRIEGEL H P, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise [C]//Proceedings of the 2nd international conference on knowledge discovery and data mining. Portland: AAAI Press, 1996: 226–231.
- [15] 秦佳睿, 徐蔚鸿, 马红华, 等. 自适应局部半径的 DBSCAN 聚类算法 [J]. 小型微型计算机系统, 2018, 39(10): 2186–2190.
- [16] 周水庚, 周傲英, 曹晶. 基于数据分区的 DBSCAN 算法 [J]. 计算机研究与发展, 2000, 37(10): 1153–1159.
- [17] WU Y P, GUO J J, ZHANG X J. A linear DBSCAN algorithm based on LSH [C]//2007 international conference on machine learning and cybernetics. Hong Kong: IEEE, 2007: 2608–2614.
- [18] 武佳薇, 李雄飞, 孙涛, 等. 邻域平衡密度聚类算法 [J]. 计算机研究与发展, 2010, 47(6): 1044–1052.
- [19] CASSISI C, FERRO A, GIUGNO R, et al. Enhancing density-based clustering: parameter reduction and outlier detection [J]. Information Systems, 2013, 38(3): 317–330.
- [20] KUMAR K M, REDDY A R M. A fast DBSCAN clustering algorithm by accelerating neighbor searching using groups method [J]. Pattern Recognition, 2016, 58: 39–48.
- [21] 于彦伟, 贾召飞, 曹磊, 等. 面向位置大数据的快速密度聚类算法 [J]. 软件学报, 2018, 29(8): 2470–2484.
- [22] LV Y, MA T, TANG M, et al. An efficient and scalable density-based clustering algorithm for datasets with complex structures [J]. Neurocomputing, 2016, 171: 9–22.
- [23] BRYANT A, CIOSEK K. RNN-DBSCAN: a density-based clustering algorithm using reverse nearest neighbor density estimates [J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(6): 1109–1121.
- [24] GHOLIZADEH N, SAADATFAR H, HANAFI N. K-DBSCAN: an improved DBSCAN algorithm for big data [J]. The Journal of Supercomputing, 2021, 77(6): 6214–6235.
- [25] LIU R, WANG H, YU X. Shared-nearest-neighbor-based clustering by fast search and find of density peaks [J]. Information Sciences, 2018, 450: 200–226.
- [26] MACQUEEN J. Some methods for classification and analysis of multivariate observations [C]//Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. California: University of California Press, 1967: 281–297.
- [27] ANKERST M, BREUNIG M M, KRIEGEL H P, et al. OPTICS: ordering points to identify the clustering structure [J]. ACM SIGMOD Record, 1999, 28(2): 49–60.
- [28] CHEN J, YU P S. A domain adaptive density clustering algorithm for data with varying density distribution [J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(6): 2310–2321.