Greedy DBSCAN:一种针对多密度聚类的 DBSCAN 改进算法*

冯振华，钱雪忠，赵娜娜
（江南大学 物联网工程学院，江苏 无锡 214000）

摘　要： 针对基于密度的 DBSCAN 算法对于输入参数敏感、无法聚类多密度数据集等问题，提出了一种贪心的 DBSCAN 改进算法（greedy DBSCAN）。算法仅需输入一个参数 MinPts，采用贪心策略自适应地寻找 Eps 半径参数进行簇发现，利用相对稠密度识别和判定噪声数据，在随机寻找核对象过程中使用邻域查询方式提升算法效率，最终通过簇的合并产生最终的聚类结果。实验结果表明，改进后的算法能有效地分离噪声数据，识别多密度簇，聚类准确度较高。

关键词： 多密度；贪心策略；相对稠密度；邻域查询；噪声数据；DBSCAN 聚类
中图分类号： TP301.6　　**文献标志码：** A　　**文章编号：** 1001-3695(2016)09-2693-04
doi:10.3969/j.issn.1001-3695.2016.09.029

Greedy DBSCAN:an improved DBSCAN algorithm on multi-density clustering

Feng Zhenhua, Qian Xuezhong, Zhao Nana
(*School of Internet of Things Engineering, Jiangnan University, Wuxi Jiangsu* 214000, *China*)

Abstract: Since the DBSCAN algorithm can not handle multi-density datasets clustering problems and sensitive to input parameters, this paper proposed an improved DBSCAN algorithm based on greedy strategy, namely greedy DBSCAN. Especially, the proposed algorithm only needed to input one parameter MinPts, and it used greedy strategy to find the parameter Eps adaptively. Moreover, the proposed algorithm used the relative density to identify and determine the noise data. The proposed approach used the neighborhood query in the process of random seeking core objects to improve the efficiency. And through the combination of the clusters to generate the final clustering results. The experiment results show that Greedy DBSCAN algorithm can separate the noise data effectively and obtain higher accuracy of identifying multi-density clusters.

Key words: multi-density; greedy strategy; relative density; neighborhood query; noise data; DBSCAN clustering

0　引言

聚类是根据“同类相近，异类相远”的思想，将未知的数据集划分为若干个组或类的过程，划分的相似度描述是基于数据描述属性（对象间的距离）的取值来确定的。通过聚类分析，可以从给定的数据集中搜索出数据属性之间的所存在的有价值的联系，进而发现数据的分布模式，为数据的进一步处理奠定基础。聚类分析已经广泛地用于许多应用领域，包括商务智能、图像模式识别、Web 搜索、生物学和安全[1]。

传统的聚类方法主要有划分方法、分层方法、基于密度方法[2~4]、基于网格方法和基于模型方法。其中，基于密度方法不需要预先指定聚类的簇数，能够在含有噪声的数据集中发现任意数量和形状的簇，所以在聚类分析中有着广泛的应用。DBSCAN（density-based spatial clustering of application with noise）[5]是一种经典的基于密度的聚类算法，但该算法需要用户在没有先验知识的情况下设定参数 Eps 和 MinPts，参数对于最终的聚类结果质量影响很大，而且在密度不均匀的数据集中，由于使用全局参数而导致更差的聚类效果。

为解决上述问题，很多学者提出了不同的改进方法。针对参数敏感常用的聚类算法有 OPTICS[6] 和 DBSCANCC[7] 等，它们分别通过产生簇排序和记录簇连接信息屏蔽参数敏感，但对多密度聚类仍无能为力。已有能够处理多密度数据集的算法有 Chameleon、SNN[8] 和基于网格密度[9~12] 等，其中 Chameleon 和 SNN 算法对于噪声数据表现敏感，聚类准确率较低。文献[9~12]中基于网格密度的算法可以聚类多密度数据集，但它们在簇间临界点较多的情况下，聚类结果不够精确。文献[13,14]分别利用 Delaunay 三角网图形算法和数据流来实现多密度聚类，它们的缺点是输入参数较多，而且对参数敏感。另外，范敏等人[15]提出了一种基于区域中心点的 DBSCAN 改进算法，通过从区域密度最大的点开始扩展，直到由密度比例因子决定的边缘区域为止，该算法对于参数较为鲁棒，可以处理多密度数据集。基于此，本文提出了一种贪心的 DBSCAN 改进算法，通过贪心选择自适应地寻找半径参数进行簇发现，采用邻域查询提升算法执行效率，并利用相对稠密度识别噪声数据。最终实验结果表明改进后的算法可以有效地对多密度数据集进行聚类，屏蔽参数敏感。下面围绕所提算法的相关概念、思想、步骤，并结合实验结果进行说明。

收稿日期： 2015-04-27；**修回日期：** 2015-06-05　　**基金项目：** 国家自然科学基金资助项目(61103129,61202312)；江苏省科技支撑计划资助项目(BE2009009)

作者简介： 冯振华(1990-)，男，河南安阳人，硕士，主要研究方向为数据挖掘(fzhoo7@163.com)；钱雪忠(1967-)，男，副教授，主要研究方向为数据库技术、数据挖掘、网络安全等；赵娜娜(1990-)，女，硕士，主要研究方向为数据挖掘.

1 DBSCAN 算法

1.1 DBSCAN 相关概念

DBSCAN 需要用户设定两个输入参数:ε(半径参数,与文中 Eps 参数含义一致)和 MinPts(邻域密度阈值)。确定 MinPts 的值比较容易:在数据对象不多的情况下,MinPts 在二维空间聚类中一般取 4[16];另外取数据集的$\lfloor m/25 \rfloor$也是一种有效的方式[17](其中 m 是数据样本总数,$\lfloor \cdot \rfloor$表示向下取整)。DBSCAN 中的概念主要有:

定义 1 核对象 一个对象 p 的 ε-近邻至少包含一定数目($\geq$MinPts)的对象,则对象 p 称为核对象。

定义 2 直接密度可达 在对象集 D 中,若对象 q 为另一个对象 p 的 ε-近邻且 p 为核对象,则对象 q 从对象 p 直接密度可达。

定义 3 密度可达 在对象集 D 中,若存在一个点链 p_1、p_2、…、p_n,对于 $p_i \in D(1 \leqslant i \leqslant n)$,且 p_{i+1}是从 p_i 的直接密度可达,则点 p_n 从 p_1 密度可达。

定义 4 密度连接 若存在对象 o,使得对象 p 和 q 都从 o 密度可达,则对象 p 和 q 密度相连。

通过以上描述,基于密度的聚类就是一组密度连接的对象,以实现最大化的密度可达。不包含在任何聚类中的对象就是噪声数据。

1.2 DBSCAN 算法的局限性

DBSCAN 算法对于参数十分敏感,而且不能对密度不均匀的数据集进行聚类。如图 1 所示。

图 1 中存在两个密度不同的簇和两个噪声点:高密度簇 C_1 和低密度簇 C_2,噪声点 o_1 和 o_2。如果选取 C_1 中对象间的距离作为 ε 半径参数进行 DBSCAN 聚类,聚类结果只能得到簇 C_1,其余的对象将被认为是噪声数据;相反地,如果选取一个较大的半径参数 ε,则两个簇均可被发现,但是离群的噪声对象 o_1 被错分到簇 C_1 中。原因是 DBSCAN 算法使用一组全局参数去聚类两个不同密度的簇。

2 Greedy DBSCAN 算法

本文算法借鉴贪心算法的思路,即从问题的某一个初始解出发一步一步地进行,每一步都要保证获得局部最优解。贪心策略的实质是通过问题的一系列局部最优解来逼近整体最优解。Greedy DBSCAN 算法主要步骤如下:

a)利用贪心策略自适应寻找合适的 ε 半径参数,并发现簇;

b)在簇周围进行邻域查询;

c)噪声点检测,对不合理的簇执行“回滚”操作;

d)合并含有公共点的簇。

重复以上步骤,直到数据集中所有点被标记为“已处理”。

2.1 簇的发现与合并

算法首先假设数据集中所有的点都是核对象,即每个点都可以在其 ε-近邻内找到($\geq$MinPts)个点。簇的发现与合并具体过程如下:

a)自适应寻找 ε 半径参数

随机选取一点 p,依据贪心策略的选择,寻找距离点 p 最近的 k 个(k = MinPts - 1)点使 p 成为核对象。对集合 $D = \{d(p,p_1),d(p,p_2),d(p,p_3),\cdots,d(p,p_k),\cdots\}$中的元素升序排列(其中 $d(p,p_k)$表示点 p 与距离它最近的第 k 个点的标准欧几里德距离),令 $\varepsilon = d(p,p_k)$。

b)圆形簇的形成

得到半径参数后 ε,形成以核对象 p 为圆心、ε 为半径的圆形簇。

c)簇的合并

对含有公共点的圆形簇进行合并,成为最终的聚类结果簇。

如图 2 中共有四个直接密度可达的圆形簇,它们的半径参数各不相同,其中 C_3 和 C_4 即可进行簇的合并。改进的算法通过贪心策略为所有的直接密度可达簇找到其半径参数值 ε,每一个直接密度可达的圆形簇即为一个局部最优解,最终通过簇的合并(即密度连接),而达到整体最优解。

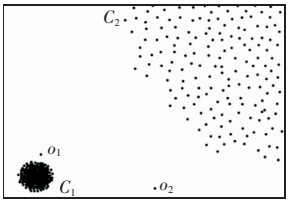


图 1 密度不均匀的数据集

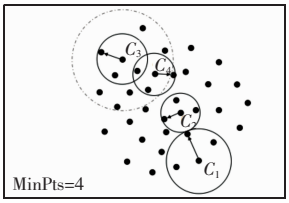


图 2 簇的发现与合并

2.2 邻域查询

为了减少算法盲目随机寻找核对象的耗时操作,提升算法执行效率,引入了一种简单高效的邻域查询方式。在图 2 中,在簇 C_3 适当邻域范围内(本文邻域取值为 ε)寻找未被标记为“已聚类”的点进行核对象扩展,发现簇 C_4。以此类推,直到邻域范围内不能发现合理的直接密度可达簇为止。

2.3 噪声点识别

如果算法随机选取的初始点是一个离群的噪声点或是与高密度簇临界的低密度簇中的点,它同样能够找到距离其最近的 k 个点并形成直接密度可达簇,但这个簇在逻辑上是错误的,影响了最终的聚类效果。为了解决这个问题,本文将噪声点检测识别放在了直接密度可达簇形成之后,利用相对稠密度进行检测:

a)采用点间的平均距离作为相对稠密度衡量标准

平均距离计算式如下:

$$\text{distance} = \frac{1}{k}\sum_{p_i \in C} d(p_i, p_{k\_\text{nearest}(i)}) \tag{1}$$

其中:$p_{k_\text{nearest}(i)}$表示与点 p_i 最近的 k 个点,$d(\cdot)$表示点间的标准欧几里德距离。式(1)中平均距离越小,说明点周围密度越高,反之则密度越低。

b)噪声点的判定

假设当离群的噪声点 p 通过半径参数 ε 达到阈值要求成为核对象后,利用式(1)计算点 p 邻域范围内其余点间的距离平均值 $\text{distance}_{\text{avg}}$。由于噪声点在搜索直接密度可达簇时,将距离其最近的高密度簇中的边缘点归入了自己的簇中,但噪声点的邻域是密度相对低的区域,所以半径参数 ε 必将远远大于簇内其余点间的平均距离。换言之,当点 p 满足式(2)时即为离群的噪声点:

$$\text{distance}_{\text{avg}} << \varepsilon \tag{2}$$

当随机选取的初始点 p 满足式(2)则说明形成的以 p 为核对象的直接密度可达簇是错误的、不合逻辑的,需对其进行

“回滚”操作,即将该簇内的点及簇相关信息全部撤销。如图 3 中点 p 的半径参数 ε 远远大于簇内其余点间平均距离 $distance_{avg}$,说明点 p 为离群的噪声点。

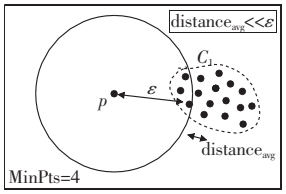


图 3　噪声点作为随机初始点

2.4　Greedy DBSCAN 伪代码

Greedy DBSCAN 算法的伪代码如下所示。其中,D 表示聚类的数据集,O_i 表示数据集 D 中的点对象。

```
Function GreedyDBSCAN(int MinPts){
k = MinPts - 1
Mark all points in D as unprocessed
while there is a unprocessed point{
    Randomly select point p as a initial object
    Mark p as processed
    //计算半径 eps
    Search for the k points closest to p
    Put the distance-values into array k_nearest[ ]
    Sort array k_nearest[ ]//对 k 个邻近距离排序
    eps = k_nearest[k - 1]
    Expand cluster C around p in epslion eps
    //判断噪声数据
    if(avg_{i∈Neighbors(p),i≠p}(P_C(i) - P(k_nearest(i)))) < <eps){
      Rollback(C)//回滚操作
      Mark p as a non-core object}
      else{
      Mark points in cluster C as is Classed(processed)
      Mark p as a core object
      //核对象邻域查询
      Search new cluster in C's neighborhood area}}
//合并含有相同公共点的簇(“密度连接”)
if(C_i contain C_j's object){
    Merge(C_i, C_j)}
//输出噪声数据
for all O_i ∈ D
    if O_i is not isClassed
    Then print O_i is an Outlier
End
```

3　实验分析

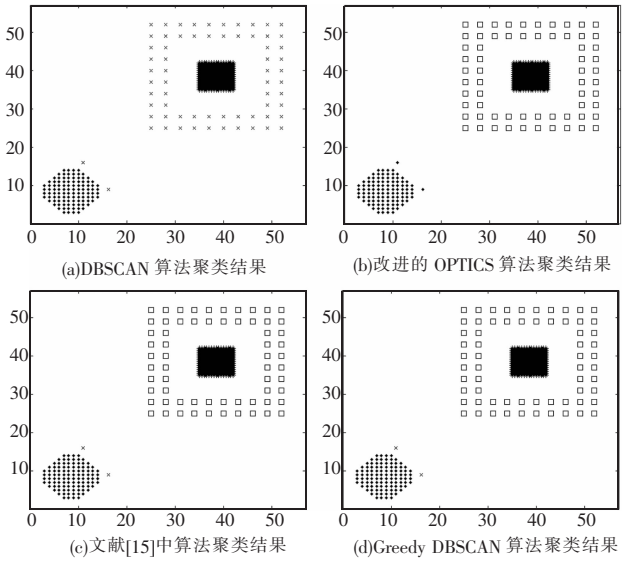
本文提出的 Greedy DBSCAN 算法通过 Java 语言在 Eclipse 环境中实现,MATLAB 工具处理实验结果。实验的环境是:CPU 为 Intel Centrino 2.2 GHz;内存为 2 GB;OS 为 Windows 7。实验在各个算法所需参数均等、公平的情况下,通过在四个不同类型和规模的多密度数据集上分别与 DBSCAN 算法、改进的 OPTICS 算法[6]、文献[15]中的算法进行聚类效果比较,最后对本文所提算法性能进行评估分析。

3.1　实验结果分析

第一组实验采用类似于图 1 中的含有噪声点的人工模拟构造的多密度数据集,由于该数据集样本点个数较少,MinPts 参数统一取 4。实验结果如图 4 所示。

在图 4 的实验结果中以黑色的“×”表示噪声点,图 4(a)(b)中的方法均不能在不同密度的簇中准确识别噪声点。图 4(c)(d)中的方法均正确识别了密度高低不同的三个簇,并且准确识别了距离高密度簇较近的两个噪声点。

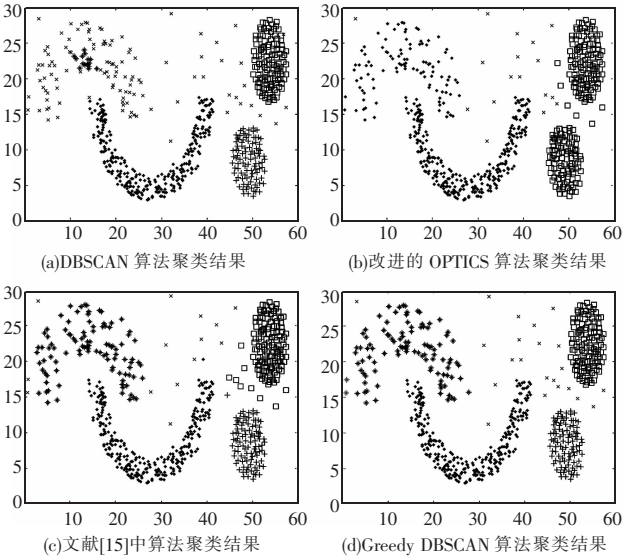
为了验证算法在更复杂的多密度数据集上的表现,第二组实验在 Shape Sets[18] 数据集中随机选取两组合并成为一个更复杂的多密度数据集上进行。合并后的数据集共有 624 个数据点,包含了四个形状、密度各不相同的簇和若干噪声点。不同算法的聚类结果如图 5 所示。



(a)DBSCAN 算法聚类结果　(b)改进的 OPTICS 算法聚类结果

(c)文献[15]中算法聚类结果　(d)Greedy DBSCAN 算法聚类结果

图 4　实验一聚类结果



(a)DBSCAN 算法聚类结果　(b)改进的 OPTICS 算法聚类结果

(c)文献[15]中算法聚类结果　(d)Greedy DBSCAN 算法聚类结果

图 5　实验二聚类结果

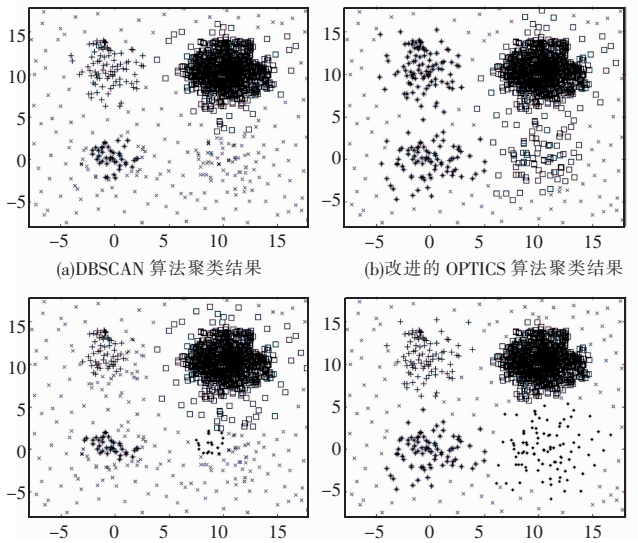
上述实验结果中,图 5(a)的低密度簇中的绝大部分对象都被处理为噪声点(噪声点用黑色“×”表示),图 5(b)中只识别出两个簇,而且将噪声数据归入了簇中,图 5(c)中算法虽然识别出了 4 个簇,但对于边界数据处理较为粗糙,影响了最终的聚类结果。而 Greedy DSCAN 算法成功识别出了 4 个密度、形状各不相同的簇,噪声数据检测也十分准确。

实验三针对高密度簇与低密度相邻、并且有临界点干扰的情况,使用加入了噪声的 Sizes5[19] 数据集。MinPts 参数取样本点个数的⌊1/25⌋,取值为 38,聚类结果如图 6 所示。

图 6 中用黑色“×”表示噪声点,DBSCAN 算法只识别了三个簇,其中一个低密度簇全部处理为了噪声数据;改进的 OPTICS 算法识别了两个簇,但其将与高密度簇相邻的低密度簇错误地并入了同一簇中;文献[15]中算法聚类结果得到了正确簇数,但是错误地将边缘噪声归入了高密度簇中,同时对于

低密度簇的识别也不够完整;Greedy DBSCAN 算法准确识别出了四个相邻的密度不同的簇,且仅吸收了少量的噪声数据到簇中。

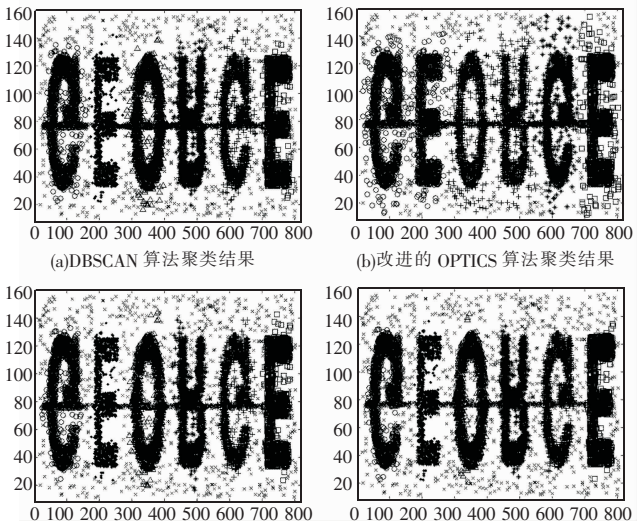
第四组仿真实验在来源于文献[20]中的 Chameleon 数据集上进行,实验结果如图 7 所示。



(a)DBSCAN 算法聚类结果 (b)改进的 OPTICS 算法聚类结果

(c)文献[15]中算法聚类结果 (d)Greedy DBSCAN 算法聚类结果

图 6 含有噪声的 Sizes5 数据集实验结果



(a)DBSCAN 算法聚类结果 (b)改进的 OPTICS 算法聚类结果

(c)文献[15]中算法聚类结果 (d)Greedy DBSCAN 算法聚类结果

图 7 Chameleon 数据集实验结果

Chameleon 数据集共有 8 000 个数据对象,包含六个字母形状的簇。图 7(a)中识别了六个簇,但是在簇与簇间连接处的噪声数据没有被识别;图 7(b)只识别出了四个簇,而且吸收了数量相当多的噪声数据到簇中;图 7(c)中基本识别出了六个字母形状的簇,但是对于簇周边的噪声及簇间干扰未能准确识别;图 7(d)中对于处在六个簇间的"横线"噪声数据干扰进行了准确的识别,仅吸收了少量噪声数据到簇中。

3.2 性能分析

通过上述四组仿真实验结果,可以发现文献[15]中的方法和本文提出的 Greedy DBSCAN 算法都具备处理多密度数据集的能力,但是当数据集中噪声数据较多或簇间存在明显噪声干扰的情况下,本文所提算法的聚类结果准确率要明显优于文献[15]中的方法。下面在 Chameleon 数据集中随机抽取 10 组不同规模大小的数据集,对应地将四个算法分别在不同规模大小的数据集上独立运行 20 次,分别计算平均运行时间,并将其图表化,如图 8 所示。

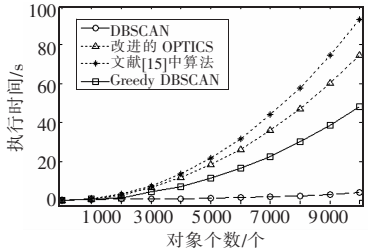


图 8 四个算法的运行时间对比图

从图 8 中可以看出,四个算法的运行时间都随着数据集规模的扩大而增长,但 DBSCAN 算法效率最高,但它对于含有噪声的多密度数据集无能为力。OPTICS 算法需要为数据集中的每一个对象寻找核心距离和可达距离,而且在迭代查找可达对象时,需要对种子对象依可达对象距离进行排序[7];文献[15]中的算法首先需要在数据集中找到区域中心点(区域密度最大的点)依次作为候选核心点,而这个过程需要遍历数据集中所有的数据,耗时严重,并且在簇的扩展时需计算其 k 个最邻近;Greedy DBSCAN 算法按照贪心策略,只需为每一个圆形簇找到其邻域半径,同时算法在寻找核对象时利用邻域查询提高效率,而且在数据集中圆形簇的数量远远小于点的个数,因此算法运行效率要优于改进的 OPTICS 算法和文献[15]中的算法。

4 结束语

本文提出的 Greedy DBSCAN 算法继承了基于密度聚类算法的优势,可以识别任意形状和数量的簇,对于含有噪声点的多密度数据集有着很好的聚类效果,仅需用户输入一个参数,而且在算法运行效率上要优于改进的 OPTICS 算法和文献[15]中的算法,具有较高的执行效率。但当面对海量数据集时算法的处理时间仍有待提高,下一步研究重点是如何结合启发式搜索策略和网格技术来进一步提升算法效率。

参考文献:

[1] Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 范明, 孟小峰, 译. 3 版. 北京:机械工业出版社,2012:288-322.

[2] Dhami C, Bnasal M. An improvement of DBSCAN algorithm to analyze cluster for large datasets[C]//Proc of IEEE International Conference on MOOC Innovation and Technology in Education. [S. l.]: IEEE Press, 2013:42-46.

[3] Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191):1492-1496.

[4] Smiti A, Eloudi Z. Soft DBSCAN: improving DBSCAN clustering method using fuzzy set theory[C]//Proc of the 6th International Conference on Human System Interaction. [S. l.]: IEEE Press, 2013:380-385.

[5] 孙吉贵, 刘杰, 赵连宇. 聚类算法研究[J]. 软件学报, 2008, 19(1):48-61.

[6] 曾依灵, 许洪波, 白硕. 改进的 OPTICS 算法及其在文本聚类中的应用[J]. 中文信息学报, 2008, 22(1):51-55.

[7] 蔡颖琨, 谢昆青, 马修军. 屏蔽了输入参数敏感性的 DBSCAN 改进算法[J]. 北京大学学报: 自然科学版, 2004, 40(3):480-486.

[8] Antunes A, Santos M Y, Moreira A. Fast SNN-based clustering approach for large geospatial data sets[M]//Connecting a Digital Europe Through Location and Place. [S. l.]: Springer International Publishing, 2014:179-195.

(下转第 2700 页)

表 3 Wine 数据集测试结果表

序号	随机 EM	K-meansEM	DDEM
1	74.16	85.39	97.19
2	50.00	85.39	97.19
3	57.87	68.54	97.19
4	69.66	85.39	97.19
5	77.53	85.39	97.19
6	94.38	85.39	97.19
7	50.56	85.39	97.19
8	67.42	85.39	97.19
9	68.54	85.39	97.19
10	88.76	85.39	97.19
平均	69.89	83.71	97.19

表 4 Soybean 数据集测试结果表

序号	随机 EM	K-meansEM	DDEM
1	70.21	72.34	76.60
2	63.83	72.34	76.60
3	61.70	72.34	76.60
4	51.06	72.34	76.60
5	61.70	46.80	76.60
6	53.19	55.32	76.60
7	78.72	72.34	76.60
8	74.47	72.34	76.60
9	61.70	72.34	76.60
10	61.70	72.34	76.60
平均	63.83	68.08	76.60

随机 EM 算法由于随机选择初始值,最终收敛得到的聚类结果很不稳定。K-meansEM 算法先通过运行 K-means 算法,得到一个收敛后的中心作为 EM 算法的初始值,最终聚类结果相对随机 EM 算法稳定一些。但是 K-means 算法本身只能聚类球形簇,并且收敛效果取决于其随机初始中心的选择,所以通过 K-means 算法收敛取得的中心并不能保证是较优的并且不稳定。聚类过程中簇中心具有附近数据点分布较密集,并且每个簇中心相距较远的特点,DDEM 算法通过查找密度大并且与其他中心点距离远的点,保证了初始中心更接近于簇中心,从而最终结果相比于其他两个算法有更高的准确率,同时 DDEM 算法每次均可取得相同的初始中心,保证了稳定性。

4 结束语

DDEM 算法通过基于密度的方法识别噪声点,基于密度和距离的方法优化了初始值的选择。通过实验验证了 DDEM 算法可以有效识别噪声点,在保证准确率的同时提高了算法的稳定性。将改进后的算法应用于实际的工程中,并提高算法的性能将是下一步工作的重点。

参考文献:

[1] Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 范明, 孟小峰, 译. 3 版. 北京: 机械工业出版社, 2012: 288-376.

[2] Richard O, Peter E, David G. Pattern classification[M]. 李宏东, 等译. 2 版. 北京: 机械工业出版社, 2003: 415-477.

[3] Jain A K. Data clustering: 50 years beyond K-means[J]. Pattern Recognition Letters, 2010, 31(8): 651-666.

[4] Deng Hongbo, Han Jiawei. Data clustering: probabilistic models for clustering[M]. Minnesota: CRC Press, 2013: 61-81.

[5] Li Ximing, Ouyang Jihong, Zhou Xiaotang. Centroid prior topic model for multi-label classification [J]. Pattern Recognition Letters, 2015, 62(9): 8-13.

[6] Halima B, Gilles C, Adrian E, *et al*. Inference in model-based cluster analysis[J]. Statistics and Computing, 1997, 7(1): 1-10.

[7] Dempster A, Laird N, Rubin D. Maximum likelihood from incomplete data via the EM algorithm[J]. Royal Statistical Society, 1977, 39(1): 1-38.

[8] Charles B, Camille B. Model-based clustering of high-dimensional data: a review[J]. Computational Statistics and Data Analysis, 2012, 71(3): 52-78.

[9] Martin E, Hans-Peter K, Sander X, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//Proc of the 2nd International Conference on Knowledge Discovery and Data Mining. USA: AAAI, 1996: 226-231.

[10] Raymond T, Vladimir T. Distance-based outliers: algorithms and applications[J]. VLDB Journal, 2000, 8(2): 237-253.

[11] Jin Wen, Tung A, Han Jiawei, *et al*. Ranking outliers using symmetric neighborhood relationship[C]//Proc of the 10th Pacific-Asia Conference Advances in Knowledge Discovery and Data Mining. [S. l.]: Springer, 2006: 577-593.

[12] Arthur D, Vassilvitskii S. K-means ++: the advantage of careful seeding[C]//Proc of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms. New York: ACM Press, 2007: 1027-1035.

[13] Zhai Donghai, Yu Jiang. K-means text clustering algorithm based on initial cluster centers selection according maximum distance[J]. Application Research of Computers, 2014, 31(3): 713-719.

[14] Christophe B. Initializing EM using the properties of its trajectories in Gaussian mixtures[J]. Statistics and Computing, 2004, 14(3): 267-279.

[15] Chandan K, Bhanukiran V. A survey of partitional and hierarchical clustering algorithms[M]. USA: CRC Press, 2013: 88-105.

(上接第 2696 页)

[9] Chen Xiaoyun, Min Yufang, Zhao Yan, *et al*. GMDBSCAN: multi-density DBSCAN cluster based on grid[C]//Proc of IEEE International Conference on e-Business Engineering. [S. l.]: IEEE Press, 2008: 780-783.

[10] 黄红伟, 黄天民. 基于网格相对密度差的扩展聚类算法[J]. 计算机应用研究, 2014, 31(6): 1702-1705.

[11] 夏英, 李克非, 丰江帆. 基于网格梯度的多密度聚类算法[J]. 计算机应用研究, 2008, 25(11): 3278-3280.

[12] 邢长征, 温培. 基于网格密度和引力的不确定数据流聚类算法[J]. 计算机应用研究, 2015, 32(1): 98-101.

[13] Yang Xiaobing, He Lingmin, Lu Huijuan. A new method for non-spherical and multi-density clustering[C]//Proc of the 3rd International Symposium on Intelligent Information Technology Application. [S. l.]: IEEE Press, 2009: 35-38.

[14] Namadchian A, Esfandani G. DSCLU: a new data stream clustring algorithm for multi density environments[C]//Proc of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing. [S. l.]: IEEE Press, 2012: 83-88.

[15] 范敏, 李泽明, 石欣. 一种基于区域中心点的聚类算法[J]. 计算机工程与科学, 2014, 36(9): 1817-1822.

[16] 孙凌燕. 基于密度的聚类算法应用研究[D]. 太原: 中北大学, 2009.

[17] Daszykowski M, Walczak B, Massart D L. Looking for natural patterns in data: Part 1. density-based approach[J]. Chemometrics and Intelligent Laboratory Systems, 2001, 56(2): 83-92.

[18] http://cs.joensuu.fi/sipu/datasets/[EB/OL].

[19] http://www.pudn.com/downloads219/sourcecode/math/detail1030717.html[EB/OL].

[20] Karypis G, Han E H, Kumar V. Chameleon: hierarchical clustering using dynamic modeling[J]. Computer, 1999, 32(8): 68-75.