控制与决策
Control and Decision

基于Zipf分布的网格密度峰值聚类算法

马福民, 宫婷, 杨帆, 张腾飞

引用本文:
马福民,宫婷,杨帆,张腾飞. 基于Zipf分布的网格密度峰值聚类算法[J]. 控制与决策, 2024, 39(2): 577−587.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.0365

您可能感兴趣的其他文章

Articles you may be interested in

基于相互邻近度的密度峰值聚类算法

Density peaks clustering based on mutual neighbor degree

控制与决策. 2021, 36(3): 543−552 https://doi.org/10.13195/j.kzyjc.2019.0795

基于相异性度量选取初始聚类中心改进的K−means聚类算法

Improved K−means clustering algorithm for selecting initial clustering centers based on dissimilarity measure

控制与决策. 2021, 36(12): 3083−3090 https://doi.org/10.13195/j.kzyjc.2020.0554

基于聚类簇结构特性的自适应综合采样法在入侵检测中的应用

Toward intrusion detection via cluster structure−based adaptive synthetic sampling approach

控制与决策. 2021, 36(8): 1920−1928 https://doi.org/10.13195/j.kzyjc.2019.1672

基于优化DBSCAN聚类算法的晶圆图预处理

Wafer map preprocessing based on optimized DBSCAN clustering algorithm

控制与决策. 2021, 36(11): 2713−2721 https://doi.org/10.13195/j.kzyjc.2020.0738

一种基于相对密度和决策图的聚类算法

A novel clustering algorithm based on relative density and decision graph

控制与决策. 2018, 33(11): 1921−1930 https://doi.org/10.13195/j.kzyjc.2017.0822

基于Zipf分布的网格密度峰值聚类算法

马福民[1†], 宫 婷[1], 杨 帆[1], 张腾飞[2]

(1. 南京财经大学 信息工程学院，南京 210023；2. 南京邮电大学 自动化学院、人工智能学院，南京 210023)

摘 要: 网格密度峰值聚类在兼顾密度峰值聚类算法可识别任意形状类簇的基础上,通过数据集的网格化简化整体计算量,成为当前备受关注的聚类方法. 针对大规模数据,如何进一步区分稠密与稀疏网格,减少网格密度峰值聚类中参与计算的非空网格代表点的数量是解决“网格灾难”的关键. 结合以网格密度为变量的概率密度分布呈现出类Zipf分布的特点,提出一种基于Zipf分布的网格密度峰值聚类算法. 首先计算所有非空网格的密度并映射为Zipf分布,根据对应的Zipf分布筛选出稠密中心网格和稀疏边缘网格;然后仅对稠密中心网格进行密度峰值聚类,在自适应确定潜在聚类中心的同时减少欧氏距离的计算量,降低算法复杂度;最后通过对稀疏边缘网格的处理,进一步优化类簇边界并提高聚类精度. 人工数据集和UCI数据集下的实验结果表明,所提出算法对大规模、类簇交叉数据的聚类具有明显优势,能够在保证聚类精度的同时降低时间复杂度.

关键词: 聚类；密度峰值；网格；Zipf分布；密度阈值

中图分类号: TP18 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.0365 **开放科学(资源服务)标识码(OSID):**

引用格式: 马福民, 宫婷, 杨帆, 等. 基于Zipf分布的网格密度峰值聚类算法[J]. 控制与决策, 2024, 39(2): 577-587.

Grid density peak clustering algorithm based on Zipf distribution

MA Fu-min[1†], *GONG Ting*[1], *YANG Fan*[1], *ZHANG Teng-fei*[2]

(1. College of Information Engineering，Nanjing University of Finance and Economics，Nanjing 210023，China；2. College of Automation & College of Artificial Intelligence，Nanjing University of Posts and Telecommunications，Nanjing 210023，China)

Abstract: Grid density peak clustering, taking advantage of the strong ability that the density peak clustering algorithm can identify arbitrary shape clusters, greatly reduces the computational cost by gridding the data set. It has become a popular clustering method nowadays. However, for large-scale data, how to further distinguish the dense and sparse grids and reduce the number of non-empty grids involved in the calculation of the grid density peak clustering is the key to solve the ”grid disaster”. In view of the probability density distribution with grid density as variable shows a Zipf-like distribution, a grid density peak clustering algorithm based on Zipf distribution is proposed. Firstly, the density of all non-empty grids is calculated and mapped to Zipf distribution, so the dense center grids and sparse edge grids are filtered according to the Zipf distribution. And then, only the dense center grids are clustered with peak density, and the calculation of Euclidean distance is reduced while the cluster centers are determined heuristically, which reduces the complexity of the algorithm. Finally, the sparse edge grids are processed to further optimize the cluster boundary and improve the clustering accuracy. Experimental results on the synthetic datasets and UCI datasets show that the proposed algorithm has obvious advantages for the clustering of large-scale and cross-cluster data, and can reduce the time complexity while ensuring the clustering accuracy.

Keywords: clustering；density peaks；grid；Zipf distribution；density threshold

0 引 言

如何对大数据高效地进行自动分析和挖掘是近些年备受关注的课题[1]. 聚类分析作为一种重要的数据挖掘技术,可以将无标签数据分为不同的类簇,同一类簇的数据尽可能相似,不同类簇间的数据尽可能不同[2]. 借助聚类分析可以从数据中发现潜在的知

收稿日期: 2022-03-09；**录用日期:** 2022-10-10.

基金项目: 国家自然科学基金项目(61973151, 62073173)；江苏省自然科学基金项目(BK20191406)；江苏省重点研发计划项目(BE2021001-4)；江苏省研究生科研与实践创新计划项目(G-TXW21001)；江苏省高校青蓝工程项目.

责任编委: 胡清华.

†通讯作者. E-mail: fmmatj@126.com.

识,揭示隐藏的模式和规律[3].

现有的聚类算法大体上可分为基于划分、基于密度、基于层次、基于网格以及基于模型的算法[4]. 2014年,Rodriguez等[5]提出了基于密度峰值的聚类算法(clustering by fast search and find of density peaks, DPC),该算法思想简单,参数要求少,无需迭代即可发现任意形状的类簇[6],但算法决策图的生成依赖于距离矩阵的计算,时空复杂度较高,难以适用于大规模数据的聚类分析[7]. 另外,决策图中离散点的数目随数据量的增大而增多,当处理大规模数据或决策图复杂时,在决策图中人为选定聚类中心的难度增加,选取不当会导致类簇划分错误,影响聚类精度.

针对DPC算法人工选取聚类中心困难的缺点,王万良等[8]利用切比雪夫不等式和标准差分别确定局部密度和距离的阈值,生成决策函数,自动确定聚类中心,但当聚类中心点密度非跳跃值时,聚类中心的选取可能产生遗漏. 郭佳等[9]从决策图的数据分布特征出发,将原有的欧氏距离值用距离的比较量替代,再利用二维区间估计法实现聚类中心的自动识别,在复杂形状的数据集下有较好的聚类准确性,但影响聚类中心选取的准确性受截断距离影响较大. 文献[10]在计算数据点局部密度的同时引入图论考量数据点间的整体关系,根据图中相邻数据点的转折角度自适应选择潜在聚类中心,一定程度上提高了聚类中心选择的准确性和便利性.

为使DPC算法能够适用于大规模数据的聚类分析,不少学者提出了很多改进算法,其中通过网格技术实现大数据的快速聚类是一种降低密度峰值聚类算法计算复杂度的有效方法[11-14]. Wu等[13]将网格引入密度峰值聚类算法中,每一个非空网格中的数据点用一个节点来替代,减少了大数据应用场景下DPC算法的欧氏距离计算量. 王飞等[14]在网格聚类的基础上设计一种针对大规模数据的抽样策略,有效提升了大数据聚类算法的时间效率,但抽样算法如何最大限度地保留原数据集的分布信息是尚未解决的难题. 文献[15]从提高网格算法聚类精度的角度出发对数据空间循环划分,并通过定义网格间的相似性对剩余网格进行分配,该算法能有效识别噪声数据且类簇边缘更加平滑,但剩余网格归类的步骤较为繁琐,随着数据量的增加,算法总体时间复杂度依然较高. 文献[16]提出一种基于网格划分的密度峰值聚类算法(improved density peaks clustering algorithm based on grid, GDPC)对网格化后的网格代表点进行密度峰值聚类,然后将低于噪声密度阈值的网格数据点全部剔除,极大地减少了内存和计算的开销,但当非空网格个数急剧增加时,算法时间复杂度急剧上升,算法性能降低. 此外,算法对噪声数据较为敏感,当网格数据稀疏时,易将稀疏网格中的数据错误归为噪声点而导致聚类准确率降低. 文献[17]根据数据的不均匀分布,提出一种基于网格筛选的密度峰值聚类算法(density peaks clustering algorithm based on grid screening, SDPC),通过引入网格化方法去除对聚类中心选取无影响的部分密度稀疏的点,然后使用DPC算法中决策图的方法选取聚类中心,一定程度上降低了算法的计算复杂度.

网格聚类算法的时间复杂度独立于数据点数目,仅与网格划分的个数有关,计算效率较高,但是随着数据量的增加和数据维度的提升,非空网格的数量也将增加,数据点间的欧氏距离矩阵计算时间复杂度依然较高,而当非空网格数接近于数据点总个数n时,网格技术失效[18]. 因此,如何进一步区分稠密与稀疏网格,减少参与计算的非空网格代表点数量是解决“网格灾难”的关键. Zipf分布[19]是一个典型的统计型经验规律,经多个数据集测试分析发现,将数据集网格化后,以网格密度为变量的概率密度分布曲线呈现出典型的类Zipf分布,即大量网格的密度很低,极少数网格的密度较高. 利用网格密度的Zipf分布,可以有效区分稠密与稀疏网格,能够在不影响聚类中心选取的情况下有效地筛选掉密度稀疏的网格. 鉴于此,本文提出一种基于Zipf分布的网格密度峰值聚类算法(grid density peak clustering algorithm based on Zipf distribution, ZGDPC). 首先,计算所有非空网格的密度并映射为Zipf分布,根据对应的Zipf分布筛选出稠密中心网格,仅计算稠密中心网格的密度和相对距离,从而大幅减少欧氏距离的计算量,降低计算复杂度;然后,根据潜在聚类中心判定函数选取出潜在的聚类中心,并从中启发式地确定聚类中心,减小聚类中心选择的误差;最后,通过对边缘网格的处理提高聚类精度.

1 相关知识

1.1 密度峰值聚类算法

DPC算法基于这样一种假设:对于一个数据集,密度较大的聚类中心点被低局部密度的数据点包围,且这些聚类中心点与其他高局部密度点的距离均比较大. 对于任意数据点x_i,DPC需要计算x_i的局部密度ρ_i以及x_i与具有更高局部密度的最近点距离δ_i,通过这两个值构建决策图,选取决策图中ρ_i和δ_i相对较大的点作为聚类中心,将其他数据点归类并对边界点

进行处理.

定义1 (局部密度)[5]　数据点x_i的局部密度ρ_i为

$$\rho_i = \sum_j \chi(d_{ij} - d_c). \tag{1}$$

其中: $d_{ij} = \mathrm{dist}(x_i, x_j)$为数据点$x_i$与$x_j$的距离; d_c为截断距离,取值为所有数据点间欧氏距离升序排序最小1%～2%距离中的最大值. 函数$\chi(x)$定义如下:

$$\chi(x) = \begin{cases} 1, & x < 0; \\ 0, & x > 0. \end{cases} \tag{2}$$

当数据点为连续时,局部密度ρ_i为

$$\rho_i = \sum_j \mathrm{e}^{-\left(\frac{d_{ij}}{d_c}\right)^2}. \tag{3}$$

定义2 (相对距离)[5]　相对距离δ_i是指数据点x_i到比自身局部密度高且与自身最近的那个点之间的距离,即

$$\delta_i = \min_{j:\rho_j > \rho_i} d_{ij}. \tag{4}$$

若数据点x_i是密度最高的点,则δ_i取所有相对距离中的最大值,即$\delta_i = \max_j (d_{ij})$.

定义3 (边界点)[5]　如果一个已分配至类簇的数据点与其他类中的点的距离小于截断距离d_c,则该点为边界点.

DPC算法根据以上定义,通过构造δ_i相对于d_c的决策图选取聚类中心,对其余数据点进行分配并将噪声点剔除,即可快速得到最终聚类结果. 密度峰值聚类算法的具体步骤如下所示.

算法1　DPC算法.

step 1: 计算两个任意数据点间的距离,对所有距离升序排序后确定截断距离d_c;

step 2: 依据截断距离d_c,由式(1)和(2)计算出任意数据点的密度ρ_i;

step 3: 对于任意数据点,由式(3)计算出其相对距离δ_i;

step 4: 以ρ_i为横轴、δ_i为纵轴,画出对应的聚类决策图;

step 5: 利用得到的决策图将ρ_i和δ_i均相对较高的数据点标记为聚类中心;

step 6: 将剩余的每个数据点分配到与其距离最近且密度比其大的数据点所在的簇;

step 7: 将密度最大的边界点的密度值设为噪声密度阈值,将所有密度低于噪声密度阈值的数据点设为噪声点.

1.2　Zipf分布

Zipf定律[19](Zipf's law)由美国语言学家Zipf提出,他在1932年研究英文单词的出现频率时发现,如果将单词频率按从高到低的次序排列,则每个单词出现频率与其排名存在一定的反比关系. 排名第i位的单词出现频率P_i可表示为

$$P_i = \frac{C}{i^\alpha}. \tag{5}$$

其中: C为一个常数, α为系数. 当α趋近于1时,排名第i位的单词出现概率为C/i. 国内外学者将Zipf定律广泛应用于语言学、情报学、信息科学等领域,为相关领域的知识决策提供了新模型. 令随机变量X为服从参数α和n的Zipf分布,则X的概率质量函数(probability mass function, PMF)是离散随机变量x在各特定取值上的概率,表示为

$$f(x) = \frac{1}{x^\alpha C},\ x = 1, 2, \ldots, n. \tag{6}$$

令式(6)中分母的常数C为$H_{n,\alpha} = C = \sum_{i=1}^{n} \left(\frac{1}{i}\right)^\alpha$,则$X$的累积分布函数为

$$F(x) = P(X \leqslant x) = \frac{H_{x,\alpha}}{H_{n,\alpha}},$$
$$x = 1, 2, \ldots, n. \tag{7}$$

X的总体均值为

$$E(X) = \frac{H_{n,\alpha-1}}{H_{n,\alpha}}. \tag{8}$$

2　基于Zipf分布的网格密度峰值聚类

DPC算法的核心步骤依赖于距离矩阵的计算,当数据量增大时,时间和空间复杂度呈指数型增长. 网格密度峰值聚类算法通过数据集网格化,可将每一个网格中的所有数据点看作一个整体进行简化计算,但随着数据量的增大,网格数量依然会大幅增加,一种可行的方法是进一步对网格进行区分筛选,仅对高密度中心网格进行密度峰值聚类,而低密度网格作为边界网格不会影响聚类中心的选取,只在高密度数据点划分结束后逐一归类. 然而,网格密度阈值过大会导致低密度网格数量较多,后续低密度网格的聚类计算量增大;网格密度阈值过小则会导致参与计算的网格数量减少的比例较低,无法有效解决大数据集聚类. 通过引入Zipf分布,能够合理选取网格密度阈值,在不影响高密度数据点聚类的前提下减少参与计算的网格数,进而降低算法的时间复杂度.

2.1　网格划分

定义4 (网格边长)[18]　设数据集X维度为d,包含n个数据样本, $X = \{x_1, x_2, \ldots, x_n\}$,其中$x_i = (x_i{}^1, x_i{}^2, \ldots, x_i{}^d)$, $x_i{}^j$为第i个数据第j维的数据值. 令第j维数据的最小值为l_j,最大值为h_j,网格边长为

$$s = \lambda \left\langle \prod_{j=1}^{d} (h_j - l_j)/n \right\rangle^{\frac{1}{d}}, \tag{9}$$

其中λ为控制参数,用来动态调整网格大小,λ过大网格划分粒度较粗,聚类质量降低,λ过小网格划分粒度较细,算法运行时间成本显著提升,为使网格大小适中,通常取$\lambda \in [0.5, 2]$[17].

定义5(网格特征向量) 每一个网格G表示为

$$G = (\rho, c, t, \text{cen}, \text{num}). \tag{10}$$

其中:ρ为网格密度,即落入网格中的数据点个数;c为网格所属的类簇编号,初始值为-1;t为网格类型,$t = 1$时网格为中心网格,$t = 2$时网格为边缘网格,t的初始值为0;cen为网格质心,num用于记录网格中所有数据点的序号,第i个网格的质心cen_i为

$$\text{cen}_i = \frac{1}{\rho_i} \sum_{j \in \text{num}_i} x_j. \tag{11}$$

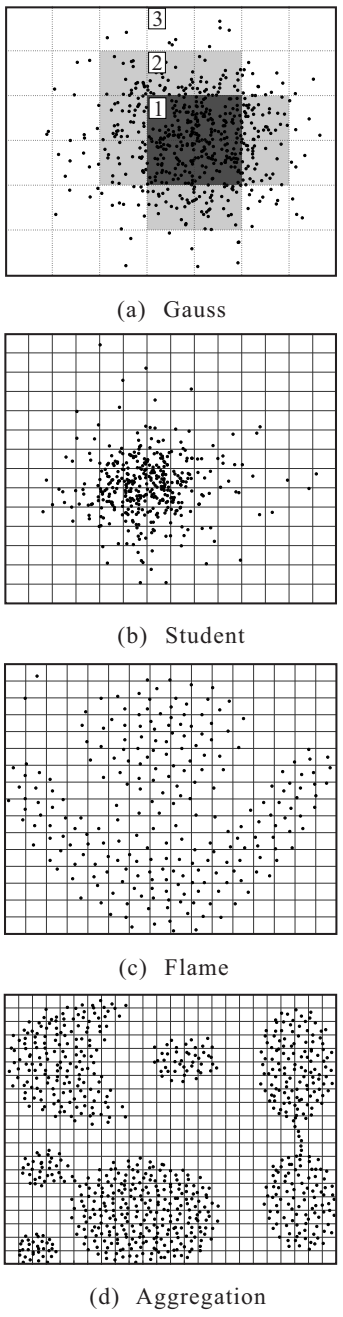
定义6(邻居网格)[18] 若网格g_j与网格g_k有公共边或公共顶点,则称网格g_j与g_k互为邻居网格.

对原始数据集进行预处理,使得第j维的任一数据值介于0与$h_j - l_j$之间,以简化后续计算. 根据网格边长s,将预处理后的数据集所在的数据空间等长划分,形成$\prod_{j=1}^{d} \frac{h_j - l_j}{s}$个互不相交的超矩形空间,并计算网格特征向量.

2.2 基于Zipf分布的网格密度阈值的确定

传统的网格密度峰值聚类算法需要计算所有非空网格两两之间的欧氏距离,当数据规模增大时,网格数量增多,距离矩阵的计算时间复杂度增加. 一种可行的方法是从全局的层面考虑聚类中心点与边界数据点的互斥关系,通过设定网格密度阈值,去除不会成为聚类中心且不影响聚类中心选取的低密度边缘网格,仅对高于密度阈值的中心网格进行密度峰值聚类.

图1为4种不同数据集的网格划分示意图. 图1(a)数据集服从均值为0、协方差为$[1, 0; 0, 1]$的二维高斯分布;图1(b)数据集服从自由度为3的T分布;图1(c)和(d)分别为经典的Flame和Aggregation数据集. 以图1(a)为例,对该数据集进行网格划分后形成31个非空网格,以网格1、2、3为代表的黑色、灰色、白色区域中网格的密度依次降低,但网格数量依次增加. 如果直接对图中所有网格进行密度峰值聚类,则欧氏距离的矩阵大小为31×31. 根据聚类分析的原理,黑色区域中的高密度网格成为聚类中心的概率更大,边界白色区域中的网格更有可能为噪声网格或边界网格. 因此,当删除边缘网格中的低密度点时(如网格3),对决策图寻找聚类中心几乎不产生影响,且计算欧氏距离时,距离矩阵由31×31降为13×13,计算复杂度大幅降低.



(a) Gauss

(b) Student

(c) Flame

(d) Aggregation

图1 不同数据集网格划分示意图

对图1(a)所示的数据集进行更为细致的网格划分,假设取$\lambda = 1.1$,划分后网格密度ρ的值域为(0,10]. 将ρ的值域等分为10个互不相交的子集$(0, 1], (1, 2], \ldots, (9, 10]$,统计每个密度子集范围内的网格数量,并计算每个密度子集出现的频率. 以密度子集为变量,将其出现的频率按从高到低进行排序. 如图2(a)所示,网格密度子集出现频率排名第1的序号为1,排名第10的序号为10,以此类推. 根据图2(a)网格密度的分布情况,计算非空网格密度子集的个数n_s及各密度子集的频率,将最小及次小密度子集序号x及频率$f(x)$代入式(6),得到该网格密度分布的Zipf函数,如图3(a)所示. 同样地,对图1(b)~(d)三种数据集进行网格划分后,其对应的网格密度子集频率排名

如图2(b)～(d)所示,相应网格密度的Zipf分布如图3(b)～(d)所示. 通过图2与图3的对比可以看出,不同数据集进行划分后的网格密度均呈现出类Zipf分布.

现有网格密度峰值聚类算法中一般使用平均密度$E(\rho)$作为密度阈值,当低密度网格过多(或过少)时,将会使平均密度阈值较低导致参与计算的网格数量增加(或平均密度阈值较高而错误地将小类簇的聚类中心归为边缘网格). 参数α为Zipf分布的系数,当$\alpha > 1$时,Zipf分布为长尾分布,密度低的网格数量占比较高;当$\alpha < 1$时,Zipf分布为短尾分布,密度高的网格数量占比较高,可以较为直观地体现网格密度的分布情况. 因此,本文将网格密度阈值ρ'表示为

$$\rho' = E(\rho)\alpha. \tag{12}$$

网格密度阈值将非空网格划分为中心网格和边缘网格,密度大于ρ'的网格为中心网格,密度小于ρ'的网格为边缘网格.

2.3　潜在聚类中心判定

根据中心网格的质心计算网格的相对距离,通过潜在聚类中心判定函数得到潜在聚类中心,以缩小聚类中心范围;然后结合网格密度,给出潜在聚类中心的密度-相对距离决策图,从中启发式地选取聚类中心,减小聚类中心选取的误差.

定义7(相对距离)　网格g_j与网格g_k间的距离计算公式为

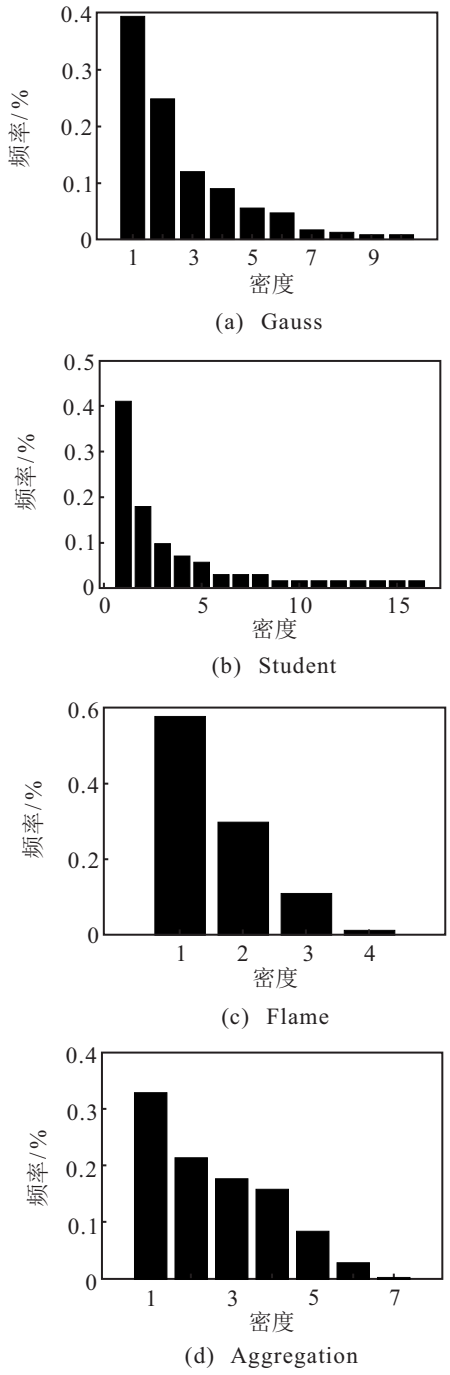


图2　不同数据集网格密度子集频率排名

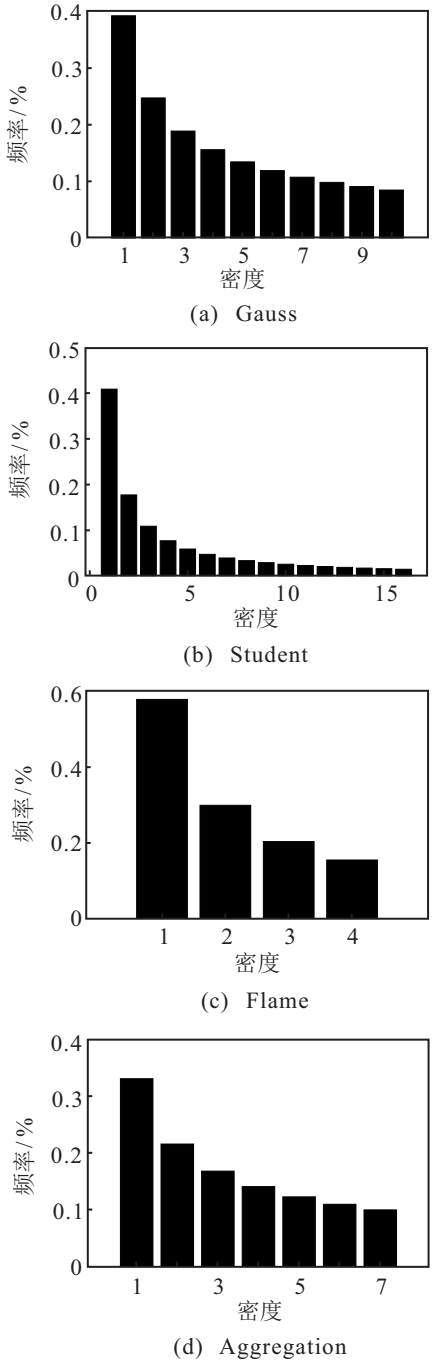


图3　不同数据集的网格密度Zipf分布

$$\text{dist}_{j,k} = \text{dist}(\text{cen}_j, \text{cen}_k). \tag{13}$$

其中: cen_j 和 cen_k 分别为网格 g_j 和 g_k 的质心, dist为欧氏距离. 若网格 g_k 的密度高于网格 g_j, 且在所有比 g_j 密度高的网格中, g_k 与 g_j 的距离最近, 则称 g_k 与 g_j 的距离为网格 g_j 的相对距离, 记为 δ_j.

定义8 (潜在聚类中心判定函数) 聚类中心需要满足两个条件: 1) 密度较高; 2) 与其他密度高的点距离较远. 根据这两个条件以及Zipf分布的特征, 给出一种自适应的潜在聚类中心判定函数, 即

$$\delta_i - E(\delta) \geqslant 0; \tag{14}$$

$$\begin{cases} F(\rho_i) \geqslant 0.8\alpha,\ E(\rho) - \text{median}(\rho) \geqslant \xi; \\ \rho_i \geqslant E(\rho),\ E(\rho) - \text{median}(\rho) < \xi. \end{cases} \tag{15}$$

其中: δ_i 为第 i 个潜在聚类中心网格的相对距离, $E(\delta)$ 为所有相对距离的均值, ρ_i 为第 i 个潜在聚类中心网格的密度, $\text{median}(\rho)$ 为所有密度的中值, ξ 用于平衡实际分布情况与所计算的Zipf函数的误差. 式(14)和(15)的判定方法分别满足聚类中心的相对距离较远、密度较高, 因此满足聚类中心判定函数的网格质心点为潜在聚类中心.

2.4 边缘网格处理

为了提高聚类精度和网格边缘的平滑度, 对边缘网格逐一处理. 利用网格的特性对边缘网格集合 G_e 中的每一个边缘网格 g_e 找到其有类标的邻居网格集合 G_{ne}, 若 G_{ne} 中网格全部属于类簇 c, 则将 g_e 归为类 c, 否则将边缘网格 g_e 中的所有数据点添加到边缘数据点集合 D 中, 最后将 D 中数据点归为其最近邻数据点所在的类簇.

算法2 边缘网格处理算法.

step 1: 从边缘网格集合 G_e 中任取一个未标记的边缘网格 g_e, 若所有网格均被标记, 则转至step 4.

step 2: 若 g_e 的邻居网格均为空网格, 则令 g_e 为噪声网格, g_e 中的所有数据点为噪声点; 否则判断其邻居网格中是否有中心网格, 如果有中心网格则将 g_e 从集合 G_e 中删除.

step 3: 若这些中心网格均属于同一类簇, 则将边缘网格 g_e 归为中心网格所属类簇; 否则将 g_e 中的所有数据点加入边缘数据点集合 D 中, 转至step 1.

step 4: 当 G_e 不为空时, 从 G_e 中任取一个网格 g_e, 将 g_e 从 G_e 中删除; 否则转至step 6.

step 5: 若 g_e 对应的邻居网格集合 G_{ne} 中所有边缘网格均属于同一类簇 c, 则将 g_e 归为类 c; 否则将 g_e 中的所有数据点加入集合 D 中.

step 6: 计算 D 中所有数据点 x_{g_e} 与 G_{ne} 所有网格数据点的欧氏距离, 并将 x_{g_e} 归为其距离最近的数据点所在类簇.

2.5 算法设计

2.5.1 算法步骤

基于Zipf分布的网格密度峰值聚类算法步骤如下.

算法3 ZGDPC算法.

输入: 数据集 $X = \{x_1, x_2, \ldots, x_n\}$, 数据维数 d, 网格控制参数 λ;

输出: 聚类结果.

step 1: 将数据集标准化, 并通过式(9)计算网格边长 s, 根据定义4对数据空间进行网格划分.

step 2: 将数据点映射到网格空间中, 并根据定义5计算网格的网格特征向量 G.

step 3: 计算最小非空网格密度子集的个数 n_s 及各密度子集的频率, 将最小及次小密度子集中值 x 及频率 $f(x)$ 代入式(6)中, 得到该网格密度分布的Zipf参数 α.

step 4: 由式(12)计算密度阈值 ρ', 依次比较所有网格密度 ρ, 若 ρ 大于 ρ', 则将该网格加入中心网格集合 G_c; 否则加入边缘网格集合 G_e.

step 5: 由定义7计算中心网格的质心相对距离 δ, 根据相对距离 δ 和网格密度 ρ 做出决策图, 并通过式(14)和(15)选择潜在聚类中心; 然后启发式确定聚类中心, 更新聚类中心网格所属类簇 c.

step 6: 将 G_c 中的中心网格按密度 ρ 降序排序, 若 G_c 不为空则从 G_c 中按序取出一个网格 g_i, 并将网格 g_i 从 G_c 中删除, 转至step 7; 否则转至step 8.

step 7: 若 c_i 不为0, 则转至step 6; 否则, 若中心网格 g_i 与中心网格 g_j 的距离 $\text{dist}(i,j) == \delta_i$, 则将网格 g_i 归为网格 g_j 所在的类簇, 即 $c_i = c_j$, 转至step 6.

step 8: 调用算法2对边缘网格进行归类.

step 9: 返回最终聚类结果.

2.5.2 算法时间复杂度分析

若数据集中数据点个数为 n, 维度为 d, 划分后的非空网格数为 m, 中心网格数为 w, 则对数据集进行网格划分并更新网格特征向量的算法复杂度为 $O(nd)$. 计算密度阈值筛选出中心网格后, 对中心网格进行密度峰值聚类分为两部分: 首先计算中心网格相对距离的时间复杂度 $O(w^2)$, 选取出聚类中心后, 将非聚类中心的中心网格进行归类的时间复杂度为 $O(w^2)$; 边缘网格的归类处理需要找到所有边缘网格的邻居网格, 其时间复杂度为 $O(2d(m-w))$. 因此, ZGDPC算法的整体时间复杂度为

$$T = O(nd) + 2 \times O(w^2) + O(2d(m-w)). \tag{16}$$

相对而言, DPC算法的时间消耗主要用于距离矩阵的计算, 其时间复杂度为 $O(n^2)$; SDPC算法过滤部分

“稀疏”数据点,仅计算“密集”数据点的距离矩阵,其时间复杂度为$O((n-n_1)^2)$(其中n_1为“稀疏”数据点的个数,$n_1<n$);GDPC算法直接计算所有网格间的欧氏距离,时间复杂度为$O(m^2)$. 当n足够大时,$w<m\ll n-n_1<n$且$2d(m-w)\ll n^2$,此时$O(nd)+2\times O(w^2)+O(2d(m-w))<O(m^2)\ll O((n-n_1)^2)<O(n^2)$. 可见,所设计的ZGDPC算法总的时间复杂度小于GDPC算法,且远小于DPC算法和SDPC算法.

3　实验及结果分析

为验证算法的聚类性能,将本文算法在人工模拟数据集和UCI数据集上进行测试,并与密度峰值聚类算法[5](DPC)、基于网格筛选的密度峰值聚类算法[17](SDPC)、基于网格划分的密度峰值聚类算法[16](GDPC)在聚类精度和效率上进行对比分析. 所有实验在Intel(R) Core(TM) i7处理器、8 G内存、Windows 10操作系统和Matlab 2018b的环境下进行.

3.1　人工数据集实验分析

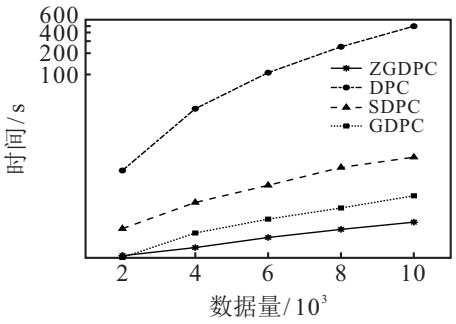
3.1.1　算法运行时间分析

由于网格划分的最坏情况是数据分布较为均匀且非空网格数较多(如Birch1[20]数据集的分布情况),本组实验利用随机选择函数从Birch1数据集中随机选择7组数据量递增的二维人工数据集I,用于比较DPC、SDPC、GDPC以及ZGDPC算法的运行效率. 为确保4种算法的可比性,以下实验均在网格划分参数$\lambda=1$、类簇数$K=10$的条件下进行,DPC、SDPC和GDPC算法参数d_c为2%. 4种算法在2 k～30 k数据规模下的运行时间如表1所示.

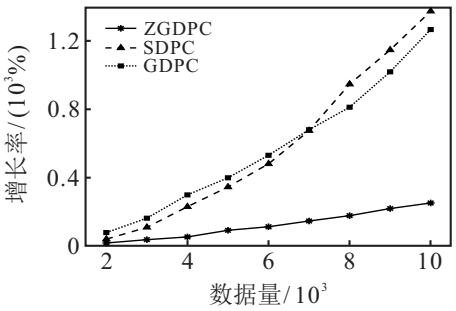
表1　不同规模人工数据集I的运行时间比较　单位: s

数据规模	DPC	SDPC	GDPC	ZGDPC
2 000	4.097	0.594	0.228	0.242
4 000	31.863	1.418	0.514	0.317
6 000	105.584	2.505	0.815	0.443
8 000	249.478	4.528	1.180	0.580
10 000	493.564	6.379	1.770	0.737
20 000	无法计算	198.337	6.625	1.023
30 000	无法计算	无法计算	214.536	1.754

图4直观地给出了4种算法的运行时间对比结果. 由表1和图4(a)可见,数据规模在10 k以下时,SDPC、GDPC以及ZGDPC算法的运行时间较短,DPC算法的运行时间呈指数型上升. 当数据规模到达20 k和30 k时,DPC和SDPC算法相继提示内存溢出无法计算,而GDPC和ZGDPC算法则不受影响. 随着数据规模的增大,网格数量增多,DPC、SDPC和GDPC算法的运行时间快速上升,而所提出ZGDPC算法较其他3种算法的速度提升比率越来越大. 图4(b)详细地展示了SDPC、GDPC以及ZGDPC算法的时间增长速率,可以看出,SDPC算法随着数据量的增加,算法的运行时间呈指数型增长,而本文算法的时间增长速率与数据量成线性关系,且优于GDPC算法.



(a)　算法运行时间对比图



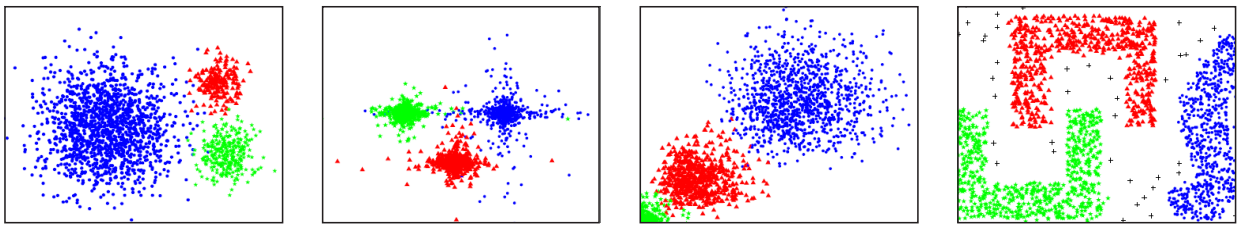
(b)　算法增长速率走势图

图4　不同算法运行时间对比

3.1.2　算法准确率分析

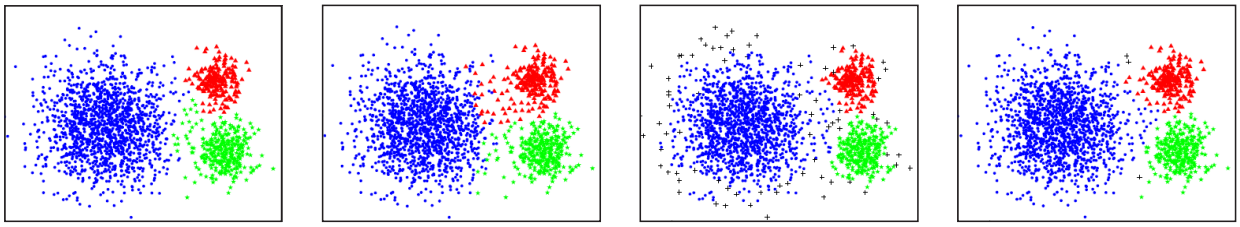
本节实验使用4组具有不同分布特征的人工数据集II,每组数据集均包含2 000个数据点,并均由3个类簇组成. 其中: GaussData数据集符合高斯分布,其类簇边缘交叉且类簇大小不均衡; TData数据集由3个自由度分别为1、2、3的T分布数据点集构成,自由度越高,数据集中趋势越明显; ChiData数据集符合卡方分布,随着自由度的增加,类簇分散趋势更加明显; UnifData数据集包含3个符合连续均匀分布的数据点集及40个噪声数据点,3个类簇分别具有不同形状且不存在数据点交叉.

4组不同分布的人工数据集II如图5所示,实验时每组数据集中不同类簇的数据分别采用不同符号表示,其中噪声点采用加号表示. DPC、SDPC、GDPC和ZGDPC四种算法在4组不同分布数据集上的聚类效果对比如图6～图9以及表2所示.



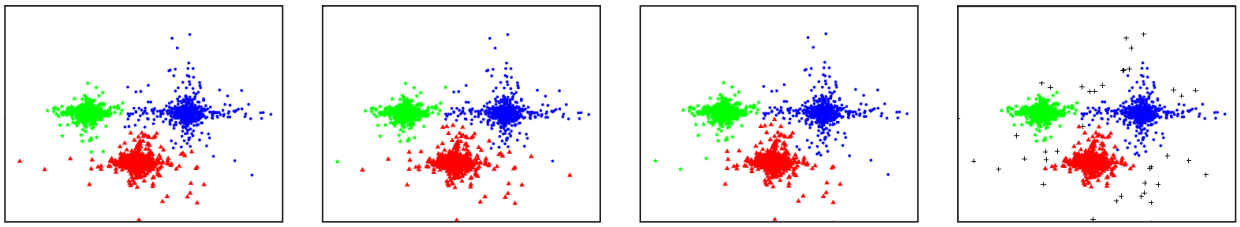
(a) GaussData (b) TData (c) ChiData (d) UnifData

图 5 不同分布人工数据集 II 分布



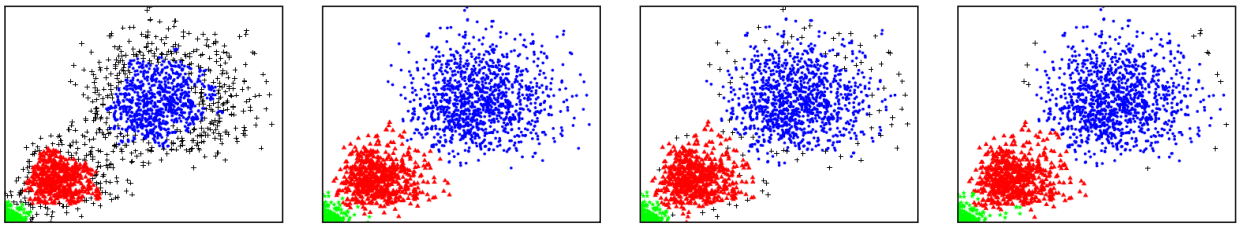
(a) DPC (b) SDPC (c) GDPC (d) ZGDPC

图 6 GaussData 数据集上不同算法聚类效果



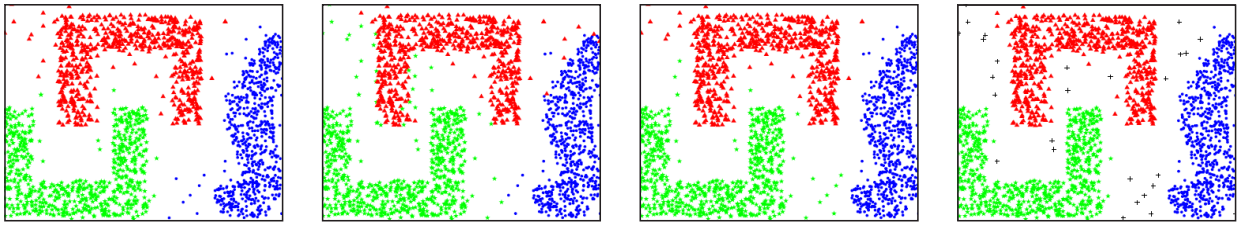
(a) DPC (b) SDPC (c) GDPC (d) ZGDPC

图 7 TData 数据集上不同算法聚类效果



(a) DPC (b) SDPC (c) GDPC (d) ZGDPC

图 8 ChiData 数据集上不同算法聚类效果



(a) DPC (b) SDPC (c) GDPC (d) ZGDPC

图 9 UnifData 数据集上不同算法聚类效果

表 2 不同分布人工数据集 II 上各算法聚类准确率比较

数据集	聚类准确率对比			
	DPC	SDPC	GDPC	ZGDPC
GaussData	0.986 0	0.975 0	0.916 5	**0.990 0**
TData	0.979 5	0.978 5	**0.980 5**	**0.969 0**
ChiData	0.734 5	**0.988 5**	0.949 0	**0.980 0**
UnifData	0.980 0	0.969 5	0.980 0	**0.994 5**

由聚类结果可以看出, DPC 算法能较好地将球形簇正确聚类, 但当类簇交叉区域的数据点密度较高或不同类簇间的相对距离较近时, 算法可能将较低密度数据点错误归为噪声数据, 或将多个类簇错误地归为同一类簇, 如图 6(a) 中大量边缘数据点被错误地归为噪声点; SDPC 算法在 DPC 算法的基础上, 通过

计算边缘数据点与所有聚类中心的欧氏距离,将边缘数据点归为与其距离最近的聚类中心所属的类簇,当类簇大小不均衡时,极容易将大类簇的边缘数据点错误归为小类簇,如图6(b)中类簇1边缘的点被错误归为类簇2和类簇3,图9(b)中将类簇1和类簇2中低密度数据点错误聚类;而GDPC算法将密度低于噪声阈值的网格中的数据点全部丢弃,导致类簇边缘不平滑且聚类正确率下降,如图6(c)和图8(c)中3个类簇的部分数据点由于所在网格密度较低,被归为噪声点而导致正确率下降. 本文所提出的ZGDPC算法将边缘网格归为其邻居网格所在类簇,并对类簇交叉部分的网格中的数据点逐一进行处理,能够有效地区分噪声点,在多数数据集上均有着理想的聚类准确率,仅在TData数据集上,因类簇边缘数据点分布稀疏且边缘网格距其他网格较远,会有稀疏边缘网格被归为噪声网格.

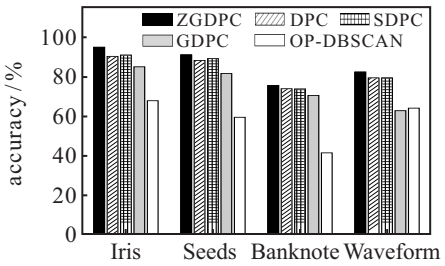
综合来看,ZGDPC算法在不同分布的数据集下均有较好的聚类效果,该算法在数据点归类时综合考虑了边界数据点的归属,多数情况下能够正确地区分噪声点,得到合理的聚类结果和较高的聚类精度,仅在少数情况下将与邻居网格距离相对较远的稀疏网格归为噪声网格,聚类精度略低于其他3种算法. 对比之下,SDPC算法无法识别噪声数据,鲁棒性较低;DPC和GDPC算法受边缘网格密度及边缘网格间的相对距离影响较大,往往无法正确区分噪声点和低密度边界点.

表3　各数据集信息及5种聚类算法在UCI数据集上的性能对比

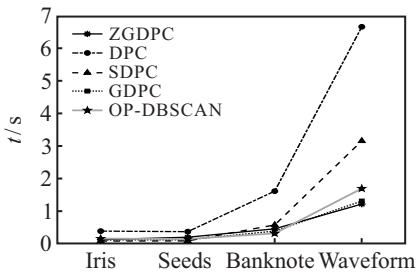
数据集	数据量	维数	簇数	算法	accuracy	NMI	time/s
Iris	150	4	3	ZGDPC	**0.953 3**	**0.857 2**	**0.123**
				DPC	0.906 7	0.805 7	0.387
				SDPC	0.913 3	0.813 8	**0.080**
				GDPC	0.853 3	0.727 7	**0.115**
				OP-DBSCAN	0.680 0	0.422 2	0.157
Seeds	210	7	3	ZGDPC	**0.914 3**	**0.738 1**	**0.199**
				DPC	0.885 7	0.698 2	0.370
				SDPC	0.895 2	0.674 2	**0.085**
				GDPC	0.819 0	0.578 5	**0.120**
				OP-DBSCAN	0.595 2	0.513 6	**0.145**
Banknote	1 372	4	2	ZGDPC	**0.758 0**	**0.357 4**	**0.458**
				DPC	0.741 3	0.346 4	1.618
				SDPC	0.740 5	0.340 5	0.578
				GDPC	0.707 0	0.303 2	**0.388**
				OP-DBSCAN	0.413 3	0.530 4	**0.317**
Waveform	5 000	21	3	ZGDPC	**0.826 4**	**0.344 4**	**1.220**
				DPC	0.797 0	**0.362 8**	6.679
				SDPC	0.797 2	**0.365 8**	3.165
				GDPC	0.629 0	0.285 6	1.314
				OP-DBSCAN	0.642 0	0.031 0	**1.699**
Pen Digits	10 992	16	10	ZGDPC	**0.622 7**	**0.625 1**	**0.500**
				DPC	无法计算	无法计算	无法计算
				SDPC	0.460 2	**0.655 8**	55.944
				GDPC	0.295 9	0.328 6	1.663
				OP-DBSCAN	0.331 3	0.373 3	19.700
Skin	51 444	3	2	ZGDPC	**0.556 9**	**0.186 9**	**0.160**
				DPC	无法计算	无法计算	无法计算
				SDPC	无法计算	无法计算	无法计算
				GDPC	0.292 9	0.173 5	0.419
				OP-DBSCAN	无法计算	无法计算	无法计算

3.2 UCI数据集实验分析

本组实验采用UCI数据集中数据量依次递增的Iris、Seeds、Banknote、Waveform、Pen Digits和Skin六个高维数据集验证本文算法的聚类性能,其中Waveform和Pen Digits是维度较高的大数据集. 各数据集信息及5种聚类算法在UCI数据集上的性能对比如表3所示(为便于对比,对上述每个数据集均进行10次实验,最后结果取10次实验的平均值). 考虑到DPC算法不适用于大数据集,本节进一步增加基于高效密度计算的大数据集快速DBSCAN聚类算法[21](a fast DBSCAN algorithm for big data based on operational dataset, OP-DBSCAN)进行对比实验. 图10分别给出了5种不同算法聚类精度和运行时间的直观对比.



(a) 聚类精度对比图



(b) 运行时间对比图

图 10 UCI数据集上不同算法的聚类指标

由图10(a)和表3可见,所提出ZGDPC算法由于对类簇交叉网格中的数据点逐一进行处理,算法准确率高于其他4种算法. DPC算法和SDPC算法因对所有数据点逐一归类,两者的聚类准确率与本文算法相差不大;而GDPC算法仅对网格进行处理,当网格划分较大或网格中数据点分布较为稀疏时,会错误地将小密度网格归为噪声点,导致聚类质量降低;OP-DBSCAN算法依据密度较高核心数据点的半径参数生成操作数据集,并对操作数据集进行DBSCAN聚类,对数据点密度较敏感,因此在高维数据集下聚类准确度较低. 当数据量较大时,DPC、 SDPC及OP-DBSCAN算法相继提示内存溢出而无法计算,GDPC和ZGDPC算法则不受影响,且ZGDPC算法的聚类精度略高于GDPC算法.

由图10(b)可知,在低维小数据集上,5种算法的运行效率相当;但随着数据规模的增大,ZGDPC算法和GDPC算法明显优于DPC算法、SDPC算法及OP-DBSCAN算法. 因GDPC算法对网格进行计算,ZGDPC算法仅对中心网格进行计算,当网格密度大于1时,网格数量远少于数据点的数目,时间复杂度大幅降低;OP-DBSCAN算法使用操作数据集OP和潜在数据集概念来修剪数据空间并减少用于查找邻居的计算,但当数据量较大时,OP对提升算法运行速度的影响较小,运行时间急剧升高;DPC算法和SDPC算法对所有数据点进行计算,时间复杂度较高;ZGDPC算法在处理边缘网格时,需要搜索每一个边缘网格的邻居网格,因此随着网格维度的增加,聚类效率略低于GDPC算法,但在聚类精度上仍具有明显优势. 综合考虑聚类精度和运行时间,ZGDPC算法在大规模、类簇交叉数据集上优势明显.

4 结 论

针对密度峰值聚类算法无法适用于大规模数据集的聚类,本文提出了一种基于Zipf分布的网格密度峰值聚类算法. 通过密度阈值对参与密度峰值计算的网格进行筛选,引入Zipf分布为密度阈值的筛选提供可靠决策,减少网格密度峰值聚类中欧氏距离的计算,有效降低了算法在大规模数据集下的时间复杂度. 此外,边缘网格的处理进一步提高了算法的准确率. 人工数据集和UCI数据集下的实验分析验证了所提出算法在大规模、类簇交叉数据集上的优越性. 但该算法在进行网格划分时,调整网格大小的控制参数λ仍然是参考现有经验取值,如何根据数据特征自适应地确定最佳网格划分参数,并进一步优化类簇边界的聚类效率将是下一步研究工作的重点.

参考文献(References)

[1] 陈叶旺, 申莲莲, 钟才明, 等. 密度峰值聚类算法综述[J]. 计算机研究与发展, 2020, 57(2): 378-394.
(Chen Y W, Shen L L, Zhong C M, et al. Survey on density peak clustering algorithm[J]. Journal of Computer Research and Development, 2020, 57(2): 378-394.)

[2] 朱光辉, 黄圣彬, 袁春风, 等. SCoS: 基于Spark的并行谱聚类算法设计与实现[J]. 计算机学报, 2018, 41(4): 868-885.
(Zhu G H, Huang S B, Yuan C F, et al. SCoS: The design and implementation of parallel spectral clustering algorithm based on Spark[J]. Chinese Journal of Computers, 2018, 41(4): 868-885.)

[3] 谢娟英, 高红超, 谢维信. K近邻优化的密度峰值快速搜索聚类算法[J]. 中国科学: 信息科学, 2016, 46(2): 258-280.

(Xie J Y, Gao H C, Xie W X. *K*-nearest neighbors optimized clustering algorithm by fast search and finding the density peaks of a dataset[J]. Scientia Sinica: Informationis, 2016, 46(2): 258-280.)

[4] 纪霞, 姚晟, 赵鹏. 相对邻域与剪枝策略优化的密度峰值聚类算法[J]. 自动化学报, 2020, 46(3): 562-575.
(Ji X, Yao S, Zhao P. Relative neighborhood and pruning strategy optimized density peaks clustering algorithm[J]. Acta Automatica Sinica, 2020, 46(3): 562-575.)

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

[6] Tong W N, Liu S, Gao X Z. A density-peak-based clustering algorithm of automatically determining the number of clusters[J]. Neurocomputing, 2021, 458: 655-666.

[7] 赵嘉, 姚占峰, 吕莉, 等. 基于相互邻近度的密度峰值聚类算法[J]. 控制与决策, 2021, 36(3): 543-552.
(Zhao J, Yao Z F, Lv L, et al. Density peaks clustering based on mutual neighbor degree[J]. Control and Decision, 2021, 36(3): 543-552.)

[8] 王万良, 吴菲, 吕闯. 自动确定聚类中心的快速搜索和发现密度峰值的聚类算法[J]. 模式识别与人工智能, 2019, 32(11): 1032-1041.
(Wang W L, Wu F, Lv C. Automatic determination of clustering center for clustering by fast search and find of density peaks[J]. Pattern Recognition and Artificial Intelligence, 2019, 32(11): 1032-1041.)

[9] 郭佳, 韩李涛, 孙宪龙, 等. 自动确定聚类中心的比较密度峰值聚类算法[J]. 计算机应用, 2021, 41(3): 738-744.
(Guo J, Han L T, Sun X L, et al. Comparative density peaks clustering algorithm with automatic determination of clustering center[J]. Journal of Computer Applications, 2021, 41(3): 738-744.)

[10] Xu T F, Jiang J H. A Graph Adaptive Density Peaks Clustering algorithm for automatic centroid selection and effective aggregation[J]. Expert Systems with Applications, 2022, 195: 116539.

[11] 何倩, 李双富, 黄焕, 等. 一种海量数据快速聚类算法[J]. 北京邮电大学学报, 2020, 43(3): 118-124.
(He Q, Li S F, Huang H, et al. A fast clustering algorithm for massive data[J]. Journal of Beijing University of Posts and Telecommunications, 2020, 43(3): 118-124.)

[12] 何熊熊, 管俊轶, 叶宣佐, 等. 一种基于密度和网格的簇心可确定聚类算法[J]. 控制与决策, 2017, 32(5): 913-919.
(He X X, Guan J Y, Ye X Z, et al. A density-based and grid-based cluster centers determination clustering algorithm[J]. Control and Decision, 2017, 32(5): 913-919.)

[13] Wu B, Wilamowski B M. A fast density and grid based clustering method for data with arbitrary shapes and noise[J]. IEEE Transactions on Industrial Informatics, 2017, 13(4): 1620-1628.

[14] 王飞, 王国胤, 李智星, 等. 一种基于网格的密度峰值聚类算法[J]. 小型微型计算机系统, 2017, 38(5): 1034-1038.
(Wang F, Wang G Y, Li Z X, et al. Clustering by fast search and find of density peaks based on grid[J]. Journal of Chinese Computer Systems, 2017, 38(5): 1034-1038.)

[15] Zhao J, Tang J J, Fan T H, et al. Density peaks clustering based on circular partition and grid similarity[J]. Concurrency and Computation: Practice and Experience, 2020, 32(7): 1-17.

[16] 江平平, 曾庆鹏. 一种基于网格划分的密度峰值聚类改进算法[J]. 计算机应用与软件, 2019, 36(8): 268-274.
(Jiang P P, Zeng Q P. An improved density peak clustering algorithm based on grid[J]. Computer Applications and Software, 2019, 36(8): 268-274.)

[17] 徐晓, 丁世飞, 孙统风, 等. 基于网格筛选的大规模密度峰值聚类算法[J]. 计算机研究与发展, 2018, 55(11): 2419-2429.
(Xu X, Ding S F, Sun T F, et al. Large-scale density peaks clustering algorithm based on grid screening[J]. Journal of Computer Research and Development, 2018, 55(11): 2419-2429.)

[18] Chen Y W, Tang S Y, Bouguila N, et al. A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data[J]. Pattern Recognition, 2018, 83: 375-387.

[19] Wang D, Cheng H B, Wang P, et al. Zipf's law in passwords[J]. IEEE Transactions on Information Forensics and Security, 2017, 12(11): 2776-2791.

[20] Fränti P. Efficiency of random swap clustering[J]. Journal of Big Data, 2018, 5(1): 1-29.

[21] Nagesh A, Dr, Gond S. A review on exploring clustering algorithms for partial object classification problems through spatial data analysis using grid dbscan technique[J]. Turkish Journal of Computer and Mathematics Education, 2021, 12(11): 3633-3641.

作者简介

马福民 (1979–), 女, 教授, 博士, 从事智能信息处理、智能生产系统等研究, E-mail: fmmatj@126.com;

宫婷 (1999–), 女, 硕士生, 从事大数据处理及应用的研究, E-mail: 1605342590@qq.com;

杨帆 (1990–), 男, 讲师, 硕士生导师, 从事大数据处理及应用的研究, E-mail:nufe_yf@163.com;

张腾飞 (1980–), 男, 教授, 博士, 从事智能信息处理、大数据分析等研究, E-mail: tfzhang@126.com.