doi:10.19665/j.issn1001-2400.20250801

融合类簇生长及边界分配策略的密度峰值聚类

陈素根[1,2],赵志忠[1]

(1. 安庆师范大学 数理学院,安徽 安庆 246133;
2. 安徽省大别山区域复杂生态系统建模、仿真与控制重点实验室,安徽 安庆 246133)

摘要: 针对密度峰值聚类算法在变密度数据集上聚类效果不佳且样本分配过程会产生“多米诺骨牌”现象等问题,提出一种融合类簇生长及边界分配策略的密度峰值聚类算法。该算法利用局部 k 近邻信息计算样本密度和相对距离,进而得到样本决策值。基于样本间距离、密度和近邻关系定义样本间吸引度和生长半径,结合决策值依次选取类簇中心,提出类簇生长策略。该生长策略从每个类簇中心出发,利用吸引度和生长半径不断生长当前类簇以获得初始聚类结果。在此基础上,利用已分配类簇和未分配样本间近邻和距离信息定义邻接度,提出边界分配策略。该分配策略由邻接度将每个未分配样本划分到最合适的类簇中,不断更新已分配和未分配样本集,直到所有样本分配完成获得最终聚类结果。在 16 个人工数据集和 10 个 UCI 数据集上与 7 个算法的比较实验结果表明,所提算法在大部分数据集上的调整兰德系数、标准化互信息和调整互信息聚类指标均优于对比算法。同时,统计检验结果表明所提算法与对比算法在统计学上均有显著性差异,具有较好的聚类效果。

关键词: 模式识别;聚类分析;密度峰值;类簇生长;边界分配

中图分类号:TP391 **文献标识码:**A **文章编号:**1001-2400(2025)05-143-19

Density peak clustering combining cluster growth and boundary assignment strategy

CHEN Sugen[1,2], *ZHAO Zhizhong*[1]

(1. School of Mathematics and Physics, Anqing Normal University, Anqing 246133, China
2. Key Laboratory of Modeling, Simulation and Control of Complex Ecosystem in Dabie Mountains of Anhui Higher Education Institutes, Anqing 246133, China)

Abstract: Aiming at the problems that the density peak clustering algorithm has a poor clustering effect on variable density datasets and that the "domino" phenomenon will occur in the sample assignment process, a density peak clustering algorithm combining cluster growth and boundary assignment strategy is proposed. The algorithm uses the local k-nearest neighbor information to calculate the sample density and relative distance, and then obtains the sample decision value. Based on the distance, density and neighbor relationship between samples, the attraction degree and growth radius are defined. Combined with the decision value, the cluster centers are selected in turn, and the cluster growth strategy is proposed. Starting from each cluster center, this strategy grows the current cluster by using the attraction degree and the growth radius to obtain the initial clustering result, on the basis of which the adjacency degree is defined by using the nearest neighbor and distance information between the assigned clusters and the unassigned

收稿日期:2024-09-23 **网络出版时间:**2025-08-19
基金项目:国家自然科学基金(61702012);安徽省高校科研自然科学研究重点项目(2024AH051095)
作者简介:陈素根(1982—),男,教授,E-mail: chensugen@126.com
赵志忠(1999—),男,安庆师范大学硕士研究生,E-mail: zhaozz1999@126.com
网络出版地址:https://doi.org/10.19665/j.issn1001-2400.20250801

samples, and the boundary assignment strategy is proposed. The assignment strategy divides each unassigned sample into the most appropriate cluster by the adjacency degree, and updates the assigned and unassigned sample sets continuously until all the samples are assigned to obtain the final clustering result. Compared with 7 algorithms on 16 synthetic datasets and 10 UCI datasets, experimental results show that the proposed algorithm is superior to the comparison algorithms in adjusted rand index, normalized mutual information and adjusted mutual information on most datasets. At the same time, the statistical test results show that the proposed algorithm and the comparison algorithm have statistically significant differences. The proposed algorithm has a better clustering effect.

Key Words: pattern recognition; cluster analysis; density peak; cluster growth; boundary assignment

1 引 言

聚类分析是机器学习和模式识别领域的重要研究问题,它旨在将给定数据中的样本划入不同类簇,使得同一类簇中的样本尽可能相似,而不同类簇中的样本具有较大差异[1]。鉴于聚类分析的快速发展,已经涌现出各种类型的聚类算法,它们在生产生活的各个方面具有广泛的应用价值[2-6]。

2014 年,Science 期刊上发表了一篇名为基于快速搜索和发现密度峰值的聚类算法的研究成果,简称密度峰值聚类(Density Peak Clustering, DPC)[7]。它具有聚类过程无需迭代、算法执行过程简单高效和可以识别任意形状的类簇等优点,得到了国内外学者的广泛关注和研究[8-9]。随着研究的深入,DPC 算法的问题逐渐显现[10],如:该算法在变密度数据集上聚类效果不佳,容易将类簇中心全部定位在高密度类簇中;分配策略容易出现因先分配样本出错导致后续样本分配均发生错误的"多米诺骨牌"现象。

为了解决 DPC 算法存在的问题,众多学者提出了多种改进策略。在提高变密度数据集聚类性能方面,DING 等[11]提出了基于方差的 DPC 算法,该算法将样本的方差与密度计算相融合解决原始 DPC 算法无法识别低密度簇的问题。YU 等[12]提出了基于加权局部密度序列和最近邻分配的 DPC 算法 (Density Peaks Clustering based on weighted local density Sequence and nearest neighbor Assignment, DPCSA),该算法将 k 近邻内外样本与密度计算相结合实现类簇划分。LI 等[13]提出基于树结构的 DPC 算法,该算法将数据集划分成多个子树,在每个子树中获得类簇中心以发现低密度类簇。ZANG 等[14]提出了基于上级节点和模糊关联准则的 DPC 算法,该算法利用模糊关联准则构造连通子图以获得变密度数据集中各个类簇的结构;HASSAN 等[15]提出了基于样本流行度的峰值聚类算法(Popularity Peak Clustering, PPC),该算法基于样本的近邻信息定义样本的流行度,使用流行度代替密度,以找到低密度类簇的中心。在改进样本分配策略方面,GUO 等[16]提出了具有连通性估计的 DPC 算法(Density Peak Clustering with Connectivity Estimation, DPC-CE),该算法利用样本构建近邻图,使用图的连通性进行样本分配,一定程度上避免了"多米诺骨牌"现象的发生。GUAN 等[17]提出了基于关联度转移方法的局部密度峰快速分层聚类算法(Fast Hierarchical Clustering of Local Density Peaks via an association degree transfer method, FHC-LDP),该算法根据局部密度峰将数据集划分成不同小簇,再使用层次聚类聚合小簇,避免样本的错误分配。HOU 等[18]提出了结合 DBSCAN 算法的 DPC 算法(DBSCAN-DPC),该算法首先从类簇中心出发,使用 DBSCAN 算法生成初始簇,然后根据相邻样本间的关系对初始簇进行扩展得到最终类簇。GUO 等[19]提出了基于局部中心和改进连通核的 DPC 算法(Density Peak clustering by local centers and Improved Connectivity Kernel, ICKDP),该算法在排除无关点影响下选择局部类簇中心进行聚类,并利用改进的连通核计算局部中心间的连通距离,提高算法在特定数据集上的样本分配准确性。尽管上述改进算法在一定程度上解决了 DPC 算法存在的问题,但这些算法没能充分考虑样本间的密度信息和相互关系,且多数改进算法的样本分配策略仅从单一近邻方面进行考虑,在一些结构相对复杂或类簇间存在相互交叉区域的数据集上往往不能取得较好的聚类结果。

基于上述分析,为进一步提高 DPC 算法的性能,提出融合类簇生长及边界分配策略的密度峰值聚类算法(Density Peak Clustering combining Cluster growth and Boundary assignment strategy, CBDPC)。CBDPC 算法使用 k 近邻定义样本密度和相对距离,进而获得样本决策值,以充分反映样本所处区域的局部

结构。利用决策值依次选择类簇中心,并基于样本间近邻、密度和距离关系不断从类簇中心出发扩展当前类簇,使得类簇可以按照条件生长。该生长策略可避免在聚类过程中无法找到低密度类簇,准确获得类簇的初始结构信息。针对生长后剩余的未分配样本,定义它们与已分配类簇间的邻接度,据此将它们依次分配到最合适的类簇中,不断更新已分配样本和未分配样本集,使得后续样本可使用周围多个近邻样本的最新类簇信息进行分配,直到未分配样本集为空,完成聚类过程。该分配策略可避免发生"多米诺骨牌"现象,提高样本分配准确性。大量数据集上实验结果表明,CBDPC 算法与 DPC 及其改进算法相比,性能更加优越且在统计学上具有显著性差异。

2　DPC 算法

2.1　DPC 算法的聚类流程

DPC 算法基于类簇中心被低密度样本包围和类簇中心之间相距较远这两点假设,首先计算样本密度和相对距离,再结合密度与相对距离选择各个类簇的中心,最后将剩余样本分配到它的高密度最近邻所在类簇中。基于上述聚类过程,DPC 算法首先计算数据集 D 中任意样本 x_i 的密度如下:

$$\rho_i=\sum_{\boldsymbol{x}_j\neq\boldsymbol{x}_i}\chi(d_{ij}-d_c),\chi(x)=\begin{cases}1,x<0\\0,x\geqslant 0\end{cases}, \tag{1}$$

$$\rho_i=\sum_{\boldsymbol{x}_j\neq\boldsymbol{x}_i}\exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right), \tag{2}$$

其中,d_{ij} 为样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 之间的欧氏距离,d_c 为截断距离,这是算法需要考虑数据集的全局信息而给定的参数。一般地,对于样本数较多的数据集,使用式(1)计算样本密度,反之使用式(2)计算样本密度。

其次,利用样本的密度与距离关系获得样本 $\boldsymbol{x}_i$ 的相对距离:

$$\delta_i=\begin{cases}\min\limits_{\boldsymbol{x}_j:\rho_j>\rho_i}d_{ij},\rho_i\neq\max(\rho)\\\max\limits_{\boldsymbol{x}_j\neq\boldsymbol{x}_i}d_{ij},\rho_i=\max(\rho)\end{cases}, \tag{3}$$

其中,ρ 为所有样本密度构成的集合。

当获得每个样本的密度和相对距离后,计算样本 $\boldsymbol{x}_i$ 的决策值:

$$\gamma_i=\rho_i\delta_i\quad。 \tag{4}$$

把决策值按照大小降序排列,一次性选择排序靠前的样本作为各个类簇的中心,并赋予不同的标签。最后,将剩余样本归为它们的高密度最近邻所属的类簇,完成聚类过程。

2.2　DPC 算法的问题分析

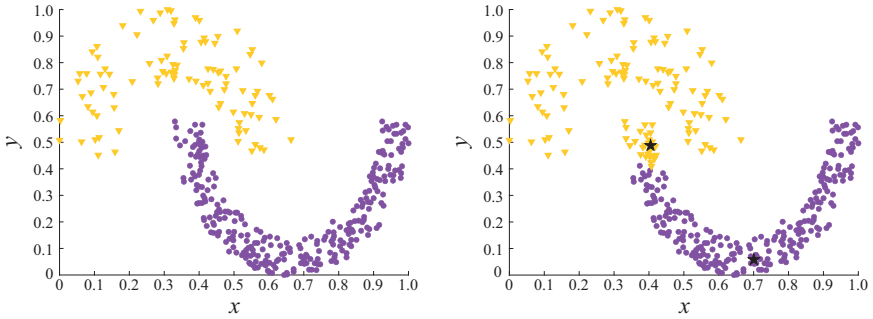
DPC 虽然具有一些优点,但它也同样存在一些问题。DPC 算法的问题主要集中于以下两个方面:

(1) DPC 算法无法在变密度数据集的低密度类簇上发现类簇中心,容易将类簇中心全部定位在高密度类簇上,导致算法在变密度数据集上聚类效果不佳。以 Jain 数据集[20]为例,叙述 DPC 算法的上述问题。绘制 Jain 数据集的真实分布图和 DPC 算法在该数据集上发现的类簇中心如图 1 所示,图中不同形状代表不同类簇,黑色五角星为类簇中心。

从图 1(a)可以发现,Jain 数据集包含的两个类簇密度分布情况不同,位于上方的流形类簇密度要明显低于下方流形类簇的密度,其是一个典型的变密度数据集。从图 1(b)可以发现,DPC 算法在下方的高密度类簇中选取到了两个类簇中心,无法在上方的低密类簇中选取到类簇中心,这是由于 DPC 算法的密度定义方式和一次性选取全部中心的类簇中心选择策略只从全局角度进行考虑,忽略了样本的局部信息和样本间的近邻、密度和距离关系,使其无法分辨变密度数据集不同密度类簇之间的结构差异,从而在该数据集上获得了较差的聚类结果。

(2) DPC 算法容易出现一个样本分配错误而后续样本均被错误分配的"多米诺骨牌"现象,导致聚类效果不佳。以 Pathbased 数据集[20]为例,叙述 DPC 算法"多米诺骨牌"现象的发生机制。绘制 Pathbased 数据集的真实分布图和 DPC 算法在该数据集上的聚类结果如图 2 所示,图中不同形状代表不同类簇,黑色五角

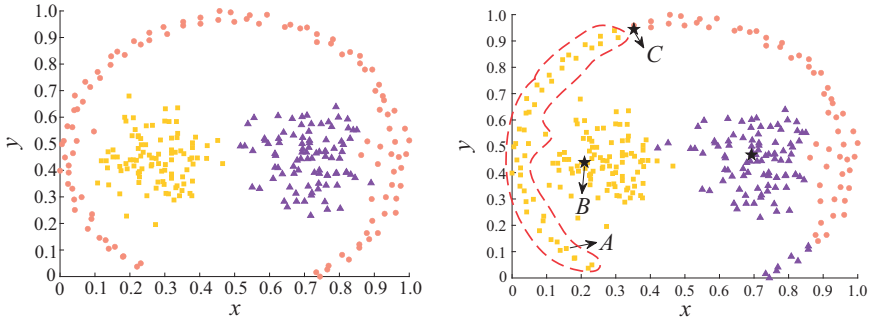
星为类簇中心。



(a) Jain数据集 (b) DPC算法找到的中心

图 1 以 Jain 数据集叙述 DPC 算法的问题

从图 2(a)可以发现,Pathbased 数据集拥有 3 个类簇,最外层的流形类簇包含中间两个球形类簇,其是一个类簇之间相互交叉的数据集。图 2(b)为 DPC 算法在该数据集上所得聚类结果,从图 2(b)可知,DPC 算法在该数据集的 3 个类簇上都得到了正确的类簇中心,但将部分样本错误分配。具体地,样本 A 是虚线圈中的样本中密度最大的样本,样本 C 和样本 A 同属于最外层的流形类簇,样本 B 与样本 A 属于不同类簇,且样本 B 和 C 的密度都大于 A 的密度。由于样本 A 和 B 之间的距离小于样本 A 和 C 之间的距离,按照 DPC 算法仅从单一近邻角度进行考虑的分配策略,样本 A 被错分到样本 B 所在类簇,进而导致虚线中的样本都被错分,这就是“多米诺骨牌”现象。



(a) Pathbased数据集 (b) DPC算法“多米诺骨牌”现象

图 2 以 Pathbased 数据集叙述 DPC 算法的问题

3 CBDPC 算法

DPC 算法无法有效聚类变密度数据集,且在样本分配过程中易发生“多米诺骨牌”现象,针对上述问题,提出 CBDPC 算法。以包含 3 个类簇的变密度数据集 D3cno123[20] 为例,图 3 给出了 CBDPC 算法框架示意图。

3.1 类簇生长

对于数据集 D 中的任意样本 $\boldsymbol{x}_i$,利用 k 近邻计算它的密度为

$$\rho_i = \frac{k}{\sum\limits_{\boldsymbol{x}_j \in N_k(\boldsymbol{x}_i)} d_{ij}} \quad , \tag{5}$$

其中,$N_k(\boldsymbol{x}_i)=\{\boldsymbol{x}_j \in D \mid d_{ij} \leqslant d_{ik}\}$ 为样本 $\boldsymbol{x}_i$ 的 k 近邻集合,d_{ik} 为 $\boldsymbol{x}_i$ 与离其第 k 远样本之间的距离。

利用式(5)计算数据集 D 中每个样本的密度,可以统一原始 DPC 算法的密度定义准则,使得算法执行过程无需考虑样本规模。同时,将样本的密度计算范围控制在 k 近邻内,较原始 DPC 算法的密度相比,能准确捕获样本的局部信息,以有效发现变密度数据集蕴含的局部结构。

当获得每个样本的 k 近邻密度后,基于样本间的近邻、密度和距离关系定义样本间的吸引度,利用吸引度来实现类簇生长。为此,首先给出样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 之间的共享近邻:

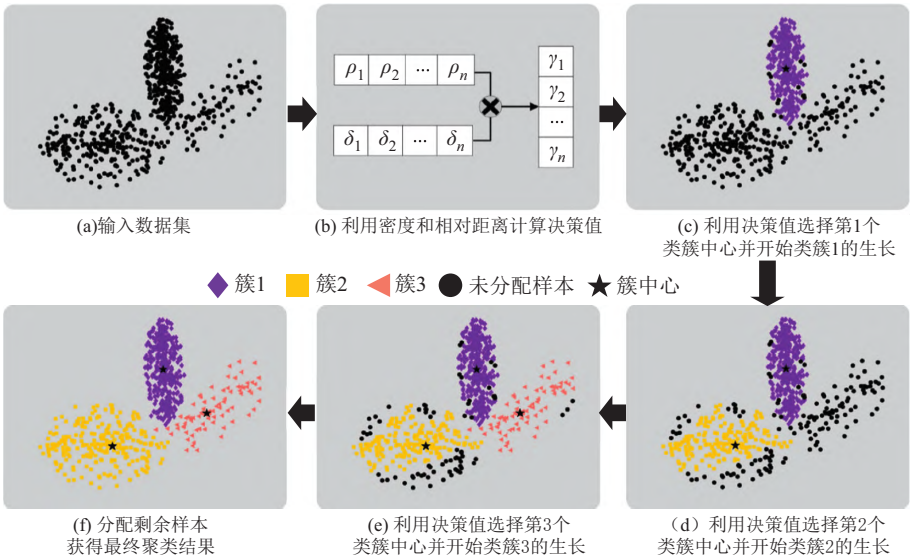


图 3　CBDPC 算法框架图

$$S_{NN}(\boldsymbol{x}_i,\boldsymbol{x}_j)=N_k(\boldsymbol{x}_i)\cap N_k(\boldsymbol{x}_j)\quad 。\tag{6}$$

样本间的共享近邻包含了样本之间的近邻关系,若样本之间的共享近邻样本数越多,样本同属于一个类簇的概率也就越大。

在获得了样本间的共享近邻之后,计算样本之间的密度关系值:

$$r(\boldsymbol{x}_i,\boldsymbol{x}_j)=\frac{2(\rho_i\rho_j)^{1/2}}{\rho_i+\rho_j}\quad 。\tag{7}$$

同属于一个类簇的样本,应该具有相同的密度分布情况,它们的密度大小比较相近。若样本间的密度越接近,由式(7)所定义的密度关系值也就越大,更能体现它们同属一类的程度。

基于样本间的近邻、密度和距离关系定义样本间的吸引度如下:

$$A(\boldsymbol{x}_i,\boldsymbol{x}_j)=\frac{|S_{NN}(\boldsymbol{x}_i,\boldsymbol{x}_j)|+r(\boldsymbol{x}_i,\boldsymbol{x}_j)}{d_{ij}}\quad ,\tag{8}$$

其中,$|S_{NN}(\boldsymbol{x}_i,\boldsymbol{x}_j)|$ 为样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 之间的共享近邻样本数。由样本间的近邻、密度和距离关系可知,样本间吸引度越大,它们同属一类的概率也就越大。

CBDPC 算法不再如 DPC 算法一次性选择所有的类簇中心,而是使用式(3)和式(4)计算每个样本的相对距离和决策值后,依次选择决策值最大的样本作为中心进行类簇生长。后续实验表明,该方法可结合样本局部 k 近邻信息来避免在变密度数据集的高密度类簇中得到多个中心而无法获得低密度类簇中心的问题,且可生长到期望的类簇数目。具体地,CBDPC 算法从每个类簇中心出发开始一个类簇的生长,在类簇生长过程中根据当前样本 $\boldsymbol{x}_w$ 与类簇中心 $\boldsymbol{x}_m$ 之间的密度关系定义 $\boldsymbol{x}_w$ 的生长半径如下:

$$d_r(\boldsymbol{x}_w)=\frac{\rho_w}{\rho_m}d_{wk}\quad ,\tag{9}$$

其中,d_{wk} 为 $\boldsymbol{x}_w$ 与离其第 k 远样本之间的距离。

为直观展现生长半径的优势,图 4 给出了在同一个类簇中处于不同位置的样本,其生长半径内所包含的样本情况。

通过图 4 可以发现,样本 A 位于类簇的内部区域,其密度高于位于边界区域的样本 B。当近邻数 k 为 4 时,通过式(9)计算它们的生长半径,获得生长半径内的样本集合后,可知样本 A 的生长半径内有 4 个样本,而样本 B 的生长半径内只有 1 个样本。上述现象说明式(9)可根据样本间密度关系获得生长半径,使得高密度样本的生长半径内可以包含更多样本,而位于类簇边界区域的低密度样本,其生长半径内包含较少的样本,以适应样本所属区域的局部结构,避免在类簇生长过程中类簇交叉区域或在真实分布中属于其它类簇内的样本对当前聚类结果的影响。

在获得了 $\boldsymbol{x}_w$ 的生长半径和生长半径内的样本集 M 后，对于任意 $\boldsymbol{x}_t \in M$，若 $\boldsymbol{x}_t$ 满足如下吸引度条件，则将其纳入当前生长的类簇中：

$$A(\boldsymbol{x}_w, \boldsymbol{x}_t) > \alpha_1 A_m(\boldsymbol{x}_w, M) \quad , \tag{10}$$

其中，$A_m(\boldsymbol{x}_w, M)$ 代表 $\boldsymbol{x}_w$ 和生长半径内样本的平均吸引度，α_1 为生长系数。通过大量数据集实验可知，当 α_1 为 0.8 时，算法可取得较好的聚类结果。式(10)利用样本间的吸引度进一步降低聚类过程中其它类簇内样本的影响。

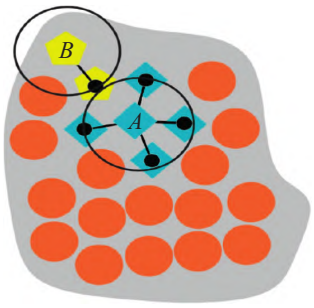


图 4 生长半径示意图

循环上述步骤直到遍历到期望数目的类簇中心，结束类簇生长过程，获得初始聚类结果。该生长过程使用生长半径和吸引度发掘样本间的近邻、密度和距离关系，获得不同密度类簇正确的局部结构信息，提升聚类的准确性。综上，算法 1 给出了类簇生长过程的伪代码。

算法 1 类簇生长。

输入：数据集 D，决策值，近邻数 k，类簇数目 c，样本密度 ρ，吸引度 A

输出：初始标签 L

① $i=0$// i 是当前的类簇数目

② $L=$zeros(1, n)//初始化样本标签为 0，n 为样本数

③ While $i \leqslant$c do

④ 获得决策值最高的样本 $\boldsymbol{x}_m$ 作为类簇中心

⑤ $L(\boldsymbol{x}_m)=i+1$;//将样本 $\boldsymbol{x}_m$ 的标签标记为 i

⑥ 初始化队列 Q 为空集，将 $\boldsymbol{x}_m$ 添加到队列 Q 中

⑦ While Q 非空 do

⑧ 选择 Q 中密度最大的样本 $\boldsymbol{x}_w$

⑨ 利用式(9)计算 $\boldsymbol{x}_w$ 的生长半径 $d_r(\boldsymbol{x}_w)$

⑩ 获得 $\boldsymbol{x}_w$ 生长半径内的样本集合 M

⑪ For M 中每个标签为 0 的样本 $\boldsymbol{x}_t$ do

⑫ If $\boldsymbol{x}_t$ 满足式(10) then

⑬ $L(\boldsymbol{x}_t)=L(\boldsymbol{x}_m)$;//将样本 $\boldsymbol{x}_t$ 的标签标记为 i

⑭ 在 Q 中添加 $\boldsymbol{x}_t$

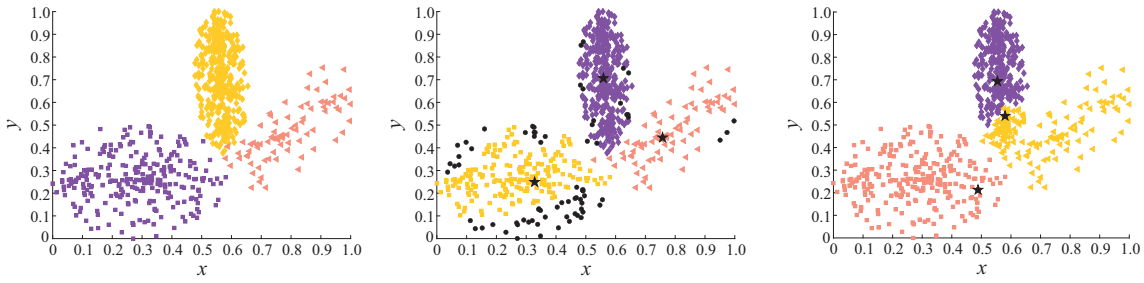
⑮ End if

⑯ End for

⑰ 在 Q 中删除 $\boldsymbol{x}_w$

⑱ End while

⑲ End while



(a) D3cno123数据集 (b) 类簇生长 (c) DPC算法

图 5 类簇生长及 DPC 算法在 D3cno123 数据集上的聚类结果

图 5 给出了类簇生长过程及原始 DPC 算法在 D3cno123 数据集上获得的聚类结果，图中不同形状代表

不同的类簇,圆点表示类簇生长后剩余未分配的样本。

结合图 3 和图 5 可以看出,CBDPC 算法的类簇生长策略按照顺序准确识别了该数据集的 3 个类簇,获得了较为完整的初始聚类结果,而 DPC 算法在上方的高密度类簇中找到了 2 个类簇中心,未能发现右侧低密度类簇的中心,获得了较差的聚类结果。上述结果说明,类簇生长策略可结合样本 k 近邻信息依次得到变密度数据集所有类簇准确的中心,并利用样本间的密度近邻关系获得不同密度类簇正确的初始结构,而 DPC 算法仅使用样本全局结构信息来识别类簇中心,未考虑样本间密度差异,所以聚类结果不理想。

3.2　边界分配

在类簇生长过程结束以后,仍可能有一些未分配的样本,这些样本通常位于类簇边界或类簇之间的交叉区域,称之为边界样本。实际上,这些边界样本的正确分配对提升聚类的性能至关重要。为了将上述边界样本分配到最合适的类簇中,基于样本与初始聚类结果中已分配类簇(类簇生长过程获得的类簇)间的距离和近邻关系,定义它们之间的邻接度,基于该邻接度依次分配边界样本。

具体地,对于数据集 D 中任意一个未分配的边界样本 $\boldsymbol{x}$,定义它的分配近邻数为

$$f=R_{\mathrm{ou}}(\alpha_2 k)\quad,\tag{11}$$

其中,$R_{\mathrm{ou}}(\cdot)$ 表示对数据四舍五入,α_2 为近邻系数。通过大量数据集实验可知,当 α_2 为 1.5 时,算法可取得较好的聚类结果。使用式(11)适当扩大边界样本的近邻数,以获得更全面的类簇信息。

在已分配的类簇中获得 $\boldsymbol{x}$ 的分配近邻 $N_f(\boldsymbol{x})$,若它包含 p 个类簇,并且第 $t(t=1,2,\cdots,p)$ 个类簇 C_t 包含 $\lambda_t(\lambda_1+\lambda_2+\cdots+\lambda_p=f)$ 个 $N_f(\boldsymbol{x})$ 中的样本,则 $\boldsymbol{x}$ 与 C_t 之间的邻接度定义为

$$S(\boldsymbol{x},C_t)=\frac{\sum\limits_{\boldsymbol{x}_j\in C_t\wedge\boldsymbol{x}_j\in N_f(\boldsymbol{x})}\exp\left(-\dfrac{d_{ij}}{k+|S_{\mathrm{NN}}(\boldsymbol{x},\boldsymbol{x}_j)|}\right)}{\lambda_t}\quad。\tag{12}$$

式(12)从共享近邻和距离角度刻画了未分配样本 $\boldsymbol{x}$ 和已分配类簇 C_t 之间的相互关系,由此定义二者的邻接度来对该关系进行度量。若样本 $\boldsymbol{x}$ 与其分配近邻中属于类簇 C_t 的样本间距离越小,并且 $\boldsymbol{x}$ 与上述样本间的共享近邻样本数越多,则样本 $\boldsymbol{x}$ 和类簇 C_t 之间的邻接度就越大。具体而言,式(12)的分子部分为样本 $\boldsymbol{x}$ 和其分配近邻中属于类簇 C_t 的样本之间总的关系值,除以分母 λ_t(λ_t 为 $\boldsymbol{x}$ 的分配近邻中属于类簇 C_t 的样本数)得到一种平均关系值。基于该平均关系值来刻画样本 $\boldsymbol{x}$ 和类簇 C_t 之间的邻接度,可以更好地利用样本 $\boldsymbol{x}$ 周围的局部信息,提高样本分配的准确性。

对于每一个未分配的边界样本 $\boldsymbol{x}$,当得到它与已分配类簇间的邻接度后,依据如下公式将其分配到最合适的类簇中:

$$L(\boldsymbol{x})=L(C_v),v=\arg\max_t\{S(\boldsymbol{x},C_t)\}\quad。\tag{13}$$

利用图 6 展示式(12)所提邻接度的优势。图 6 中,在近邻数 k 为 4,分配近邻数 f 为 6 的情况下,样本 1 是位于类簇 C_1 和 C_2 间的待分配低密度样本,在真实类簇分布中属于 C_1。若采用原始 DPC 算法仅从单个近邻角度进行考虑的分配策略,由于样本 1 和更高密度样本 3 之间的距离 d_{13} 小于样本 1 和更高密度样本 2 之间的距离 d_{12},样本 1 会被错误分配给 C_2;若采用式(12)计算它与周围已分配类簇 C_1 和 C_2 间的邻接度 $S(1,C_1)$ 和 $S(1,C_2)$,由于该邻接度计算方式充分考虑了样本间的距离和样本周围多个近邻间关系,所以可得 $S(1,C_1)$ 大于 $S(1,C_2)$,则样本 1 被正确分配到 C_1 中,提高了聚类准确率。

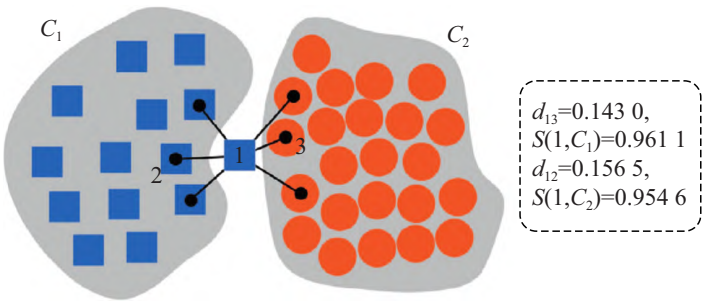


图 6　利用邻接度进行样本分配

在实际的边界分配过程中，CBDPC 算法按照密度从大到小的顺序分配边界样本。当每一个边界样本分配完毕后，不断动态更新已分配样本和未分配样本集合，使后续的边界样本分配过程可以使用待分配样本周围多个近邻样本所包含的最新类簇信息，以获得样本间更加完善的距离和近邻关系，避免“多米诺骨牌”现象的发生，提升边界分配的准确性。综上，算法 2 给出了边界分配过程的伪代码。

算法 2 边界分配。

输入：数据集 D，初始标签 L，近邻数 k

输出：最终标签 L

① 选取所有标签为零的样本构成未分配边界样本集合 B

② 选取所有标签非零的样本构成已分配样本集合 Y

③ While B 非空 do

④ 选择 B 中密度最大的样本 $\boldsymbol{x}$

⑤ 在 Y 中获得 $\boldsymbol{x}$ 的分配近邻

⑥ 利用式(12)计算 $\boldsymbol{x}$ 与分配近邻所含类簇间的邻接度

⑦ 利用式(13)获得邻接度最大的类簇 C_v

⑧ $L(\boldsymbol{x})=L(C_v)$ //更新 $\boldsymbol{x}$ 的标签

⑨ 在 Y 中添加 $\boldsymbol{x}$

⑩ 在 B 中删除 $\boldsymbol{x}$

⑪ End while

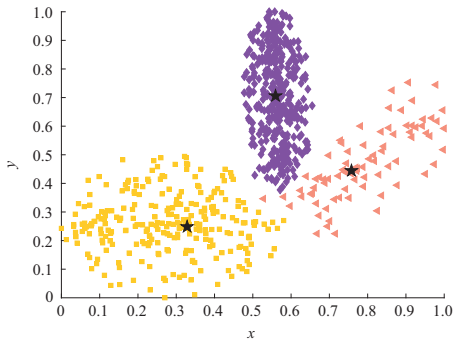


图 7 边界分配策略在 D3cno123 数据集上所得最终结果

图 7 给出了边界分配策略在 D3cno123 数据集上获得的最终聚类结果。从图 7 可以看出，CBDPC 算法的边界分配策略将图 5(b)中类簇生长过程后所有未分配的边界样本都较准确地分配到了最合适的类簇中，取得了与图 5(a)中 D3cno123 数据集的真实分布十分接近的结果。反观图 5(c)中原始 DPC 算法的聚类结果，由于识别到了错误的类簇中心，且仅从单一近邻角度进行样本分配，无法准确反映样本所属区域的局部结构，导致在中间和右侧的类簇之间发生了“多米诺骨牌”现象，其最终聚类的结果劣于所提 CBDPC 算法。

3.3 算法聚类流程及时间复杂度分析

算法 1 和算法 2 分别给出了类簇生长和边界分配策略，综合这两个算法步骤来实现 CBDPC 算法的完整聚类过程，具体算法步骤见算法 3。

算法 3 CBDPC 算法。

输入：数据集 D，近邻数 k，类簇数目 c

输出：最终标签 L

① 对数据集 D 归一化

② 利用式(5)计算样本密度

③ 利用式(3)计算相对距离

④ 利用式(4)计算样本决策值

⑤ 利用式(8)计算样本间的吸引度

⑥ 利用算法 1 获得初始标签

⑦ 利用算法 2 获得最终标签 L

假设数据集 D 中的样本数目为 n，对数据归一化并计算样本密度的时间复杂度为 $O(n^2)$；计算样本相对距离和决策值的时间复杂度为 $O(n^2)$；计算样本间吸引度的时间复杂度为 $O(n^2)$；利用算法 1 获得类簇的初始标签的时间复杂度为 $O(n)$；利用算法 2 分配边界样本的时间复杂度为 $O(n)$。综上所述，CBDPC 总的时间复杂度为 $O(n^2)$。

4　实验结果及分析

4.1　实验设置

为了验证所提 CBDPC 算法的聚类效果,在 16 个人工数据集[20]和 10 个 UCI 数据集[21]上进行了实验,各数据集的详细信息分别见表 1 和表 2。其中,人工数据集包含变密度和类簇交叉等多种复杂类型,UCI 数据集来自生产生活的多方面,使用它们进行实验可全面展示算法性能。

表 1　人工数据集

数据集	样本数	数据维度	类簇数	数据类型	数据集	样本数	数据维度	类簇数	数据类型
2d4c2	863	2	4	变密度	Xclara	3 000	2	3	类簇交叉
Ls	1 725	2	6	流形结构	DS850	850	2	5	变密度、类簇交叉
Donut2	1 000	2	2	流形结构	D3cno123	715	2	3	变密度、类簇交叉
Complex9	3 031	2	9	流形结构	Compound	399	2	6	变密度、类簇交叉
DB	629	2	4	流形结构	Pathbased	300	2	3	变密度、类簇交叉
R15	600	2	15	类簇交叉	Complex8	2 551	2	8	变密度、流形结构
D13	588	2	13	流形结构	Jain	373	2	2	变密度、流形结构
S1	5 000	2	15	类簇交叉	Disk6000n	6 000	2	2	流形结构

表 2　UCI 数据集

数据集	样本数	数据维度	类簇数	应用领域	数据集	样本数	数据维度	类簇数	应用领域
Libras	360	90	15	运动识别	DNA	2 000	180	3	医学
Wine	178	13	3	农业物理学	Heart	303	13	2	医学
Sonar	208	60	2	物理学与化学	Seeds	210	7	3	农业
Pima	768	8	2	医学	Thyroid	215	5	3	医学
Haberman	306	3	2	医学	Vote	435	16	2	社会科学

同时,为了客观评估所提 CBDPC 算法性能优劣,与 DPC[7]、DPCSA[12]、PPC[15]、DPC-CE[16]、FHC-LDP[17]、DBSCAN-DPC[18]和 ICKDP[19]算法相比较,其中 DPC 为原始算法,其余 6 个算法均是 DPC 的改进算法,DPCSA 和 PPC 算法重在提升原始 DPC 算法在变密度数据集上的聚类性能,DPC-CE、FHC-LDP、DBSCAN-DPC 和 ICKDP 算法重在解决原始 DPC 算法的样本分配问题,避免“多米诺骨牌”现象的发生。

聚类的评价指标[22]采用标准化互信息(Normalized Mutual Information, NMI)、调整兰德系数(Adjusted Rand Index, ARI)和调整互信息(Adjusted Mutual Information, AMI),各聚类指标的取值范围在 $-1\sim1$ 之间,值越接近 1 说明算法的聚类效果越优秀。各指标的计算公式如下:

$$\mathrm{NMI}=\frac{2I(T,L)}{H(T)+H(L)}\quad,\tag{14}$$

$$\mathrm{ARI}=\frac{R(T,L)-E(R(T,L))}{\max(R(T,L))-E(R(T,L))}\quad,\tag{15}$$

$$\mathrm{AMI}=\frac{I(T,L)-E(I(T,L))}{\max(H(T),H(L))-E(I(T,L))}\quad,\tag{16}$$

其中,T 为数据集的真实标签,L 为聚类所得标签,$I(T,L)$ 为 T 和 L 间的互信息,$H(T)$ 和 $H(L)$ 为 T 和 L 间熵,$R(T,L)$ 为 T 和 L 间的兰德系数,$E(\cdot)$ 为求期望,$\max(\cdot)$ 为取最大值。

公平起见,除了 DPCSA 和 DPC-CE 算法不需参数外,其余均利用网格搜索方法进行参数调优,在一定范围内选择使算法聚类效果最好的参数值。具体地,CBDPC 的参数 k 在[5,60]内取值,步长为 1;FHC-LDP 算法的参数 k 在[1,50]内取值,步长为 1;ICKDP 算法的参数 k 根据数据集的规模在一定范围内取值,步长为 1;PPC 算法的参数 k 在[1,50]内取值,步长为 1;DBSCAN-DPC 算法的参数 ε 和 k 分别在[0.025,0.040]和[4,20]内取值,步长分别为 0.001 和 1;DPC 算法的参数 d_c 在[0.05,2]内取值,步长为 0.01。所有算法的实验环境为 Windows11 64 位操作系统、i7-12700H 处理器、16 GB 内存和 MATLAB R2022b 软件。

4.2 人工数据集上的聚类结果及分析

表 3 给出了所提 CBDPC 算法及 7 种比较算法在 16 个人工数据集上的聚类结果。表 3 中,“Arg-”为算法获得最优结果时的参数值,“-”表明算法无需输入参数,“/”前的数值为 DBSCAN-DPC 算法的参数 k,“/”后数值为参数 ε,各个数据集上的第 1 和第 2 指标值分别使用黑体和下划线进行标注。表 3 中的数据集包含变密度、流形结构和类簇间相互交叉等多种复杂类型,在这些数据集上进行实验可充分验证算法在复杂结构数据集上的聚类性能。

表 3 人工数据集上的聚类结果

算法	Jain				DS850			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**1.000 0**	**1.000 0**	**1.000 0**	25	**1.000 0**	**1.000 0**	**1.000 0**	5
DPCSA	0.233 0	0.042 2	0.216 7		**1.000 0**	**1.000 0**	**1.000 0**	
FHC-LDP	**1.000 0**	**1.000 0**	**1.000 0**	18	0.995 4	0.996 6	0.995 3	43
ICKDP	**1.000 0**	**1.000 0**	**1.000 0**	6	0.988 9	0.991 6	0.988 6	9
DBSCAN-DPC	**1.000 0**	**1.000 0**	**1.000 0**	20/0.040	0.973 8	0.974 0	0.958 4	20/0.040
PPC	0.645 6	0.705 5	0.610 3	3	0.991 7	0.993 3	0.991 7	17
DPC	0.653 1	0.714 6	0.618 3	0.27	0.983 4	0.986 1	0.983 0	0.27
DPC-CE	0.554 7	0.585 3	0.515 2		0.975 2	0.979 7	0.974 7	
算法	Xclara				D3cno123			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**1.000 0**	**1.000 0**	**1.000 0**	10	**0.972 4**	**0.988 5**	**0.971 8**	33
DPCSA	0.992 0	0.995 9	0.992 0		0.721 5	0.668 8	0.677 8	
FHC-LDP	0.995 6	0.998 0	0.995 6	9	0.910 3	0.939 1	0.892 0	19
ICKDP	0.988 8	0.993 9	0.988 8	17	0.751 5	0.689 5	0.703 6	6
DBSCAN-DPC	0.988 8	0.993 9	0.988 8	20/0.040	0.962 1	0.980 5	0.957 3	20/0.025
PPC	0.997 6	0.999 0	0.999 7	16	0.946 8	0.969 9	0.938 7	3
DPC	0.995 6	0.998 0	0.995 6	1.14	0.783 4	0.754 1	0.739 8	0.09
DPC-CE	0.992 0	0.995 9	0.992 0		0.751 5	0.689 5	0.703 6	
算法	2d4c2				Ls			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**1.000 0**	**1.000 0**	**1.000 0**	14	**1.000 0**	**1.000 0**	**1.000 0**	29
DPCSA	0.993 0	0.995 9	0.992 8		0.759 7	0.628 2	0.744 9	
FHC-LDP	0.993 0	0.995 9	0.992 8	14	**1.000 0**	**1.000 0**	**1.000 0**	45
ICKDP	0.993 0	0.995 9	0.992 8	12	**1.000 0**	**1.000 0**	**1.000 0**	14
DBSCAN-DPC	0.987 4	0.991 8	0.987 0	20/0.040	**1.000 0**	**1.000 0**	**1.000 0**	20/0.040
PPC	**1.000 0**	**1.000 0**	**1.000 0**	33	0.785 7	0.646 3	0.748 6	6
DPC	0.932 1	0.955 5	0.914 0	1.22	0.749 1	0.619 2	0.688 0	0.36
DPC-CE	0.875 6	0.864 7	0.824 2		0.730 5	0.626 6	0.700 2	
算法	Donut2				D13			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**0.973 5**	**0.988 0**	**0.973 5**	7	**0.980 0**	**0.970 5**	**0.974 0**	14
DPCSA	0.966 4	0.984 0	0.966 3		0.852 5	0.577 9	0.801 0	
FHC-LDP	0.966 4	0.984 0	0.966 3	40	0.972 3	0.941 3	0.967 3	6
ICKDP	0.966 4	0.984 0	0.966 3	12	0.958 8	0.933 3	0.949 5	5
DBSCAN-DPC	0.966 4	0.984 0	0.966 3	20/0.030	0.972 6	0.949 0	0.943 4	11/0.034
PPC	0.225 4	0.122 0	0.185 6	50	0.918 2	0.805 2	0.806 1	19
DPC	0.266 2	0.177 5	0.231 3	0.28	0.933 0	0.819 2	0.919 2	1.99
DPC-CE	0.228 7	0.101 9	0.181 5		0.899 5	0.720 6	0.889 8	

算法	Compound				Pathbased			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**1.000 0**	**1.000 0**	**1.000 0**	10	**0.912 5**	**0.929 2**	**0.910 1**	26
DPCSA	0.843 9	0.828 4	0.839 2		0.731 1	0.613 3	0.707 3	
FHC-LDP	0.855 3	0.848 3	0.850 1	5	0.714 4	0.562 9	0.604 2	8
ICKDP	0.897 6	0.843 4	0.841 6	5	0.511 8	0.386 0	0.449 1	5
DBSCAN-DPC	0.909 8	0.855 3	0.877 2	12/0.040	0.772 8	0.775 2	0.770 5	13/0.028
PPC	0.758 6	0.777 4	0.715 2	1	0.651 9	0.607 6	0.639 2	11
DPC	0.737 3	0.546 1	0.696 8	0.76	0.618 5	0.563 4	0.600 1	0.11
DPC-CE	0.681 9	0.461 0	0.636 6		0.555 1	0.459 8	0.473 0	

算法	Complex9				DB			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**1.000 0**	**1.000 0**	**1.000 0**	35	**1.000 0**	**1.000 0**	**1.000 0**	32
DPCSA	0.711 9	0.422 1	0.682 1		0.453 4	0.109 6	0.413 1	
FHC-LDP	**1.000 0**	**1.000 0**	**1.000 0**	15	**1.000 0**	**1.000 0**	**1.000 0**	18
ICKDP	0.940 5	0.868 7	0.917 4	12	**1.000 0**	**1.000 0**	**1.000 0**	7
DBSCAN-DPC	**1.000 0**	**1.000 0**	**1.000 0**	20/0.040	**1.000 0**	**1.000 0**	**1.000 0**	20/0.040
PPC	0.729 7	0.544 2	0.708 5	22	0.612 7	0.429 0	0.556 1	14
DPC	0.745 4	0.609 4	0.740 2	2.00	0.512 0	0.325 1	0.459 6	0.83
DPC-CE	0.711 6	0.448 4	0.541 5		0.717 6	0.469 9	0.644 3	

算法	Complex8				R15			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**0.997 6**	**0.998 7**	**0.997 3**	32	**0.997 1**	**0.996 4**	**0.996 9**	26
DPCSA	0.758 7	0.531 1	0.742 4		0.989 3	0.985 7	0.988 5	
FHC-LDP	0.950 1	0.927 4	0.939 6	8	0.994 2	0.992 8	0.993 8	37
ICKDP	0.933 8	0.920 9	0.923 7	10	0.994 2	0.992 8	0.993 8	10
DBSCAN-DPC	0.954 6	0.941 2	0.925 7	15/0.029	0.994 2	0.992 8	0.993 8	19/0.040
PPC	0.734 8	0.588 7	0.704 7	48	0.994 2	0.992 8	0.993 8	23
DPC	0.794 6	0.703 6	0.772 1	1.72	0.994 2	0.992 8	0.993 8	1.74
DPC-CE	0.727 6	0.576 6	0.713 3		0.994 2	0.992 8	0.993 8	

算法	S1				Disk6000n			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	0.989 3	0.989 3	0.989 2	51	**1.000 0**	**1.000 0**	**1.000 0**	43
DPCSA	0.988 0	0.988 4	0.986 9		0.238 8	0.092 9	0.241 8	
FHC-LDP	0.988 4	0.988 1	0.988 2	23	**1.000 0**	**1.000 0**	**1.000 0**	7
ICKDP	0.988 4	0.988 1	0.988 2	11	0.260 0	0.132 5	0.263 4	22
DBSCAN-DPC	0.988 0	0.988 4	0.986 9	20/0.040	0.931 6	0.969 4	0.931 6	19/0.025
PPC	0.988 7	0.988 4	0.988 6	4	0.168 6	0.252 5	0.169 9	2
DPC	**0.989 6**	**0.989 7**	**0.989 5**	1.58	0.227 3	0.312 9	0.229 0	0.50
DPC-CE	0.988 4	0.988 1	0.988 2		0.123 9	0.041 0	0.124 7	

由表 3 知,CBDPC 算法除了在 S1 数据集以外,在其余 15 个人工数据集上均取得了最优值,在 S1 数据集上的各个聚类指标也处于第 2 的位置,充分说明了 CBDPC 算法不但在变密度数据集上聚类效果优越,可避免样本分配时发生“多米诺骨牌”现象,而且在各种复杂结构类型数据集上也有比较好的性能。虽然 DBSCAN-DPC、FHC-LDP 和 ICKDP 等算法在一些变密度(Jain 和 2d4c2 数据集)和流形数据集(Ls 和 DB 数据集)上,能够取得和 CBDPC 算法相同的聚类结果,但这些算法未充分考虑样本的局部结构信息和样本间的密度关系,导致它们在类簇间存在相互交叉区域的数据集(Xclara、Pathbased 和 Compound 数据集)上未能取得较好的聚类结果。另外,原始 DPC 算法在除 S1 数据集以外的其余 15 个人工数据集上的聚类结果

都比较差，说明它无法有效聚类结构复杂的数据集，CBDPC 算法在所有数据集上 NMI、ARI 和 AMI 的平均值较原始 DPC 算法分别提升了 0.244 2、0.299 6 和 0.265 2。

为了直观展现 8 种算法的聚类效果，图 8～图 11 给出了各算法在 D3cno123、Jain、Compound 和 Pathbased 数据集上的聚类效果图。图中不同的形状代表不同的类簇，五角星表示类簇中心。图 8 为各算法在 D3cno123 数据集上的聚类效果图。该数据集包含 3 个类簇且它们之间相互交叉，易引发“多米诺骨牌”现象。由图 8 可知，CBDPC 算法成功识别到该数据集不同密度的类簇结构，且在边界分配时结合多个近邻样本间的信息成功避免了“多米诺骨牌”现象的发生。DBSCAN-DPC 算法凭借 DBSCAN 算法发现初始类簇的优势，也获得了较优的聚类效果，但将类簇间的部分交叉样本分配错误。PPC 和 FHC-LDP 算法虽然能识别到右侧的低密度类簇，但它们错误地将该类簇的部分样本划分到高密度类簇中，引发了“多米诺骨牌”现象。其余算法没有充分考虑样本间的相互关系，无法准确获得类簇的局部结构信息，导致无法成功识别该数据集的低密度类簇，均取得了较差的效果。

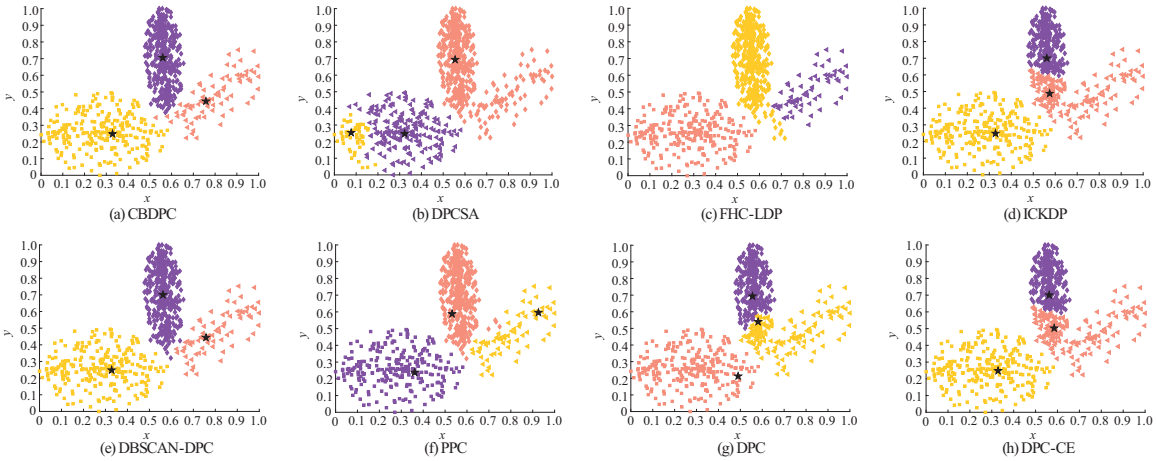


图 8 算法在 D3cno123 数据集上的聚类效果

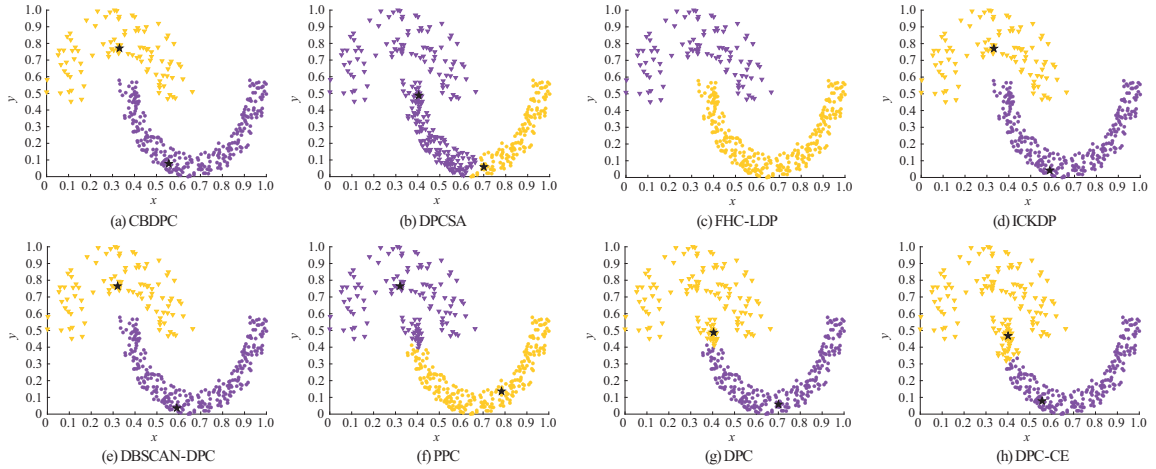


图 9 算法在 Jain 数据集上的聚类效果

图 9 为各算法在 Jain 数据集上的聚类效果图。该数据集是包含两个类簇的变密度数据集，其类簇又属于流形结构。由图 9 可知，CBDPC 算法的类簇生长策略可以准确识别两个类簇的结构，获得了完全正确的聚类结果。FHC-LDP、ICKDP 和 DBSCAN-DPC 算法在该数据集上也取得了与 CBDPC 算法相同的聚类结果，而 DPCSA、DPC 和 DPC-CE 算法没有充分考虑类簇间的结构差异，且它们定义的样本密度易受不相关样本的干扰，导致这些算法在该数据集上没有得到正确的类簇中心，聚类效果较差。虽然 PPC 算法在两个类簇上都能找到正确的类簇中心，但它的样本分配策略仅从单一近邻的角度进行考虑，未充分考虑待分配样本周围多个近邻样本的局部信息，导致将下方类簇的部分样本错误分配给了上方类簇，没有得到较优的聚类结果。

图 10 为各算法在 Compound 数据集上的聚类效果图。该数据集既包含变密度类簇,类簇间又存在相互交叉的区域,是一个典型的复杂结构数据集。由图 10 可知,CBDPC 算法在该数据集上获得了完全正确的聚类结果,原因是 CBDPC 算法通过样本间的近邻、密度和距离关系提出类簇生长策略,以准确获得各个类簇的基本结构信息;基于边界分配策略将类簇边界或交叉区域的样本成功分配,取得了较好的聚类结果,体现了 CBDPC 算法在各种类型数据集上均具有良好的聚类性能。其余算法在该数据集上的聚类效果均比较差,PPC 和 DPC-CE 算法无法有效聚类左上角的相交类簇,DBSCAN-DPC、FHC-LDP、ICKDP 和 DPC 算法无法将右侧高低密度相互包含的类簇分离,说明这些算法在聚类过程中未充分考虑样本间的密度因素,无法排除类簇间交叉样本的影响,导致它们只能聚类一些结构比较简单的数据集。

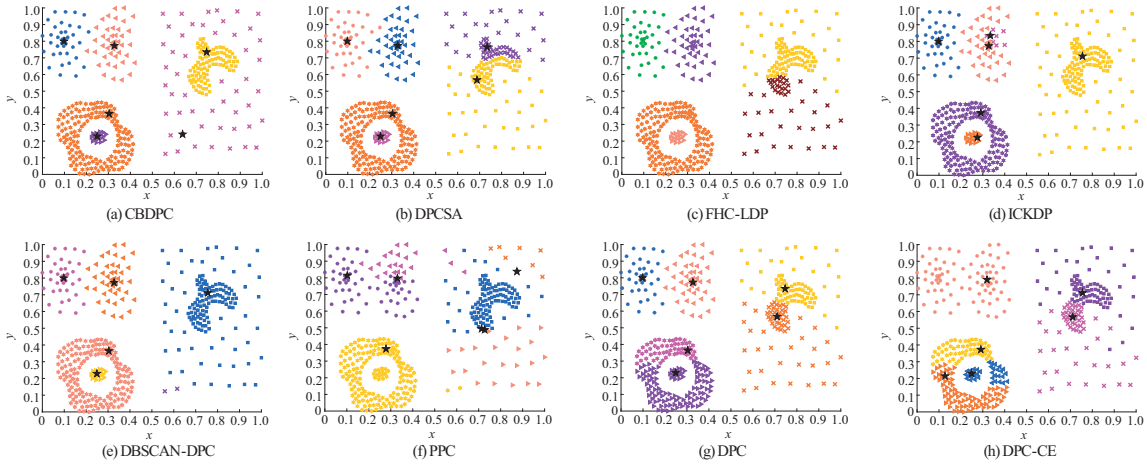


图 10　算法在 Compound 数据集上的聚类效果

图 11 为各算法在 Pathbased 数据集上的聚类结果图。该数据集包含 3 个类簇,类簇间密度差异较大且存在相互交叉区域,最外层类簇包含流形结构。利用该数据集可进一步展现算法聚类性能。由图 11 可知,CBDPC 算法取得了最优的聚类结果,获得了各个类簇较为正确的结构信息,且在类簇交叉区域样本分配时充分考虑样本周围多个近邻的信息,避免了“多米诺骨牌”现象的发生。其余对比算法虽然能准确找到各个类簇的中心,但它们无法正确处理类簇交叉区域的样本,并将外层流形类簇中的样本错误分配给了内部类簇,这使得聚类效果不够理想。

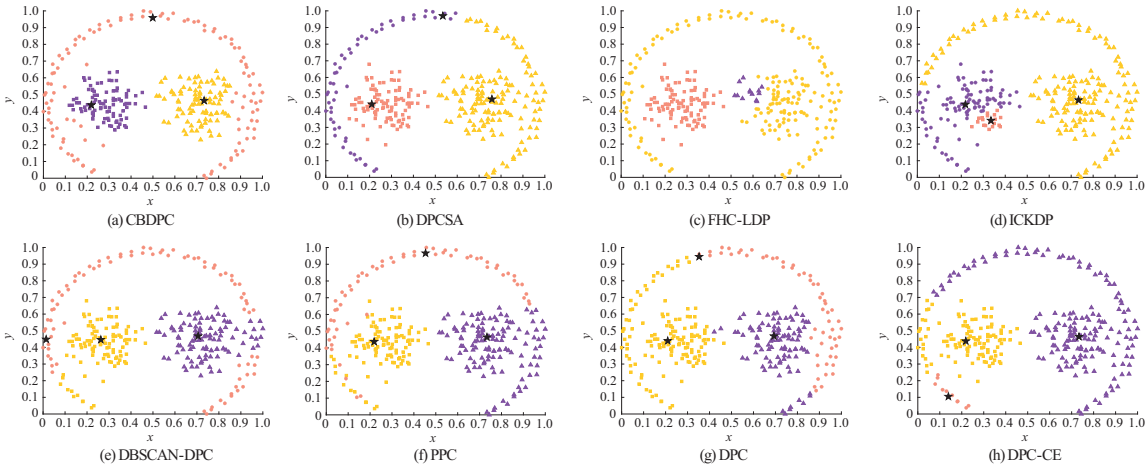


图 11　算法在 Pathbased 数据集上的聚类效果

4.3　UCI 数据集上的聚类结果及分析

UCI 数据集通常有更高的维度和更复杂的类簇结构,有利于验证算法对真实数据的处理能力。表 4 给出了 CBDPC 算法及 7 种比较算法在 UCI 数据集上的聚类结果。观察表 4 可知,CBDPC 算法除了在 Libras 和 Vote 数据集获得了第 2 的聚类效果外,在其余数据集上均获得了最优聚类结果。在 Sonar 数据集上,ICKDP 算法在 NMI 上取得了和 CBDPC 算法相同的结果,但在 ARI 和 AMI 上比 CBDPC 算法差。另外,

CBDPC 算法在所有数据集上 NMI、ARI 和 AMI 的平均值较原始 DPC 算法分别提升了 0.098 4、0.132 5 和 0.098 8。总之，CBDPC 算法考虑样本间的近邻信息和密度关系，使它在 UCI 数据集上也表现出卓越的聚类性能。

表 4 UCI 数据集上的聚类结果

算法	Vote				Libras			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	0.494 3	0.571 0	0.484 8	53	0.667 5	0.398 1	0.601 8	8
DPCSA	0.486 7	0.536 8	0.476 6		0.559 5	0.279 9	0.424 3	
FHC-LDP	0.449 9	0.536 8	0.417 5	14	0.630 0	0.337 7	0.556 1	8
ICKDP	**0.575 7**	**0.694 9**	**0.574 9**	5	0.602 3	0.309 5	0.524 3	10
DBSCAN-DPC	0.480 7	0.568 0	0.438 3	20/0.040	**0.671 3**	0.396 4	0.590 4	6/0.037
PPC	0.448 7	0.550 0	0.441 8	32	0.662 7	**0.407 9**	**0.602 4**	5
DPC	0.469 4	0.536 7	0.459 8	0.23	0.635 5	0.350 0	0.566 8	0.24
DPC-CE	0.456 0	0.510 1	0.446 4		0.590 1	0.321 5	0.514 7	
算法	Wine				Sonar			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**0.841 7**	**0.853 7**	**0.835 7**	15	**0.096 8**	**0.129 3**	**0.093 5**	13
DPCSA	0.564 7	0.505 4	0.548 6		0.062 8	0.063 0	0.050 9	
FHC-LDP	0.743 5	0.726 9	0.739 1	20	0.069 6	0.073 3	0.065 1	5
ICKDP	0.743 5	0.726 9	0.739 1	8	**0.096 8**	-0.002 3	0.051 2	8
DBSCAN-DPC	0.623 3	0.497 6	0.462 1	6/0.036	0.096 3	0.075 0	0.062 7	13/0.036
PPC	0.825 2	0.836 8	0.819 1	7	0.058 2	0.068 0	0.054 7	13
DPC	0.710 4	0.672 4	0.706 5	1.99	0.066 2	0.084 4	0.062 8	0.08
DPC-CE	0.591 1	0.536 2	0.584 1		0.038 0	0.019 0	0.031 0	
算法	Pima				Haberman			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**0.084 8**	**0.123 2**	**0.081 5**	57	**0.037 2**	**0.111 2**	**0.029 6**	31
DPCSA	0.005 2	0.014 3	0.001 7		0.001 6	-0.010 5	-0.001 0	
FHC-LDP	0.005 2	0.014 3	0.001 7	21	0.022 0	0.045 2	0.008 3	33
ICKDP	0.005 2	0.014 3	0.001 7	11	0.000 3	-0.000 4	-0.002 1	9
DBSCAN-DPC	0.032 7	0.086 7	0.022 8	12/0.030	0.033 5	0.029 3	0.019 9	7/0.0340
PPC	0.052 4	0.096 1	0.051 0	5	0.011 8	0.036 2	0.008 6	16
DPC	0.010 4	0.034 1	0.006 7	0.18	0.004 6	0.032 0	0.001 4	0.5
DPC-CE	0.031 8	0.067 7	0.023 2		0.002 0	0.015 8	-0.001 7	
算法	DNA				Heart			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**0.117 5**	**0.092 7**	**0.117 5**	56	**0.235 0**	**0.308 8**	**0.232 9**	55
DPCSA	0.051 8	0.028 4	0.051 8		0.123 5	0.098 1	0.097 4	
FHC-LDP	0.050 0	0.040 1	0.050 0	32	0.175 4	0.207 4	0.163 1	15
ICKDP	0.023 6	0.014 2	0.023 6	15	0.125 9	0.120 8	0.106 2	9
DBSCAN-DPC	0.088 9	0.065 0	0.088 9	4/0.0280	0.144 6	0.149 3	0.102 9	7/0.040
PPC	0.035 9	0.053 5	0.033 1	1	0.192 0	0.245 7	0.184 9	6
DPC	0.047 1	0.077 7	0.043 3	0.06	0.134 6	0.178 4	0.129 4	0.17
DPC-CE	0.009 5	0.028 9	0.009 5		0.154 2	0.172 3	0.139 4	
算法	Seeds				Thyroid			
	NMI	ARI	AMI	Arg-	NMI	ARI	AMI	Arg-
CBDPC	**0.788 9**	**0.814 2**	**0.784 8**	15	**0.711 4**	**0.798 9**	**0.684 9**	59
DPCSA	0.715 1	0.723 6	0.706 4		0.351 7	0.318 5	0.231 4	

FHC-LDP	0.728 6	0.763 5	0.725 6	24	0.394 0	0.370 1	0.363 3	4
ICKDP	0.741 0	0.788 0	0.738 1	10	0.431 3	0.424 0	0.319 4	9
DBSCAN-DPC	0.725 1	0.739 8	0.718 5	11/0.040	0.415 5	0.509 3	0.369 5	6/0.025
PPC	0.732 0	0.766 0	0.727 0	16	0.506 2	0.574 9	0.443 1	4
DPC	0.742 4	0.745 5	0.736 6	0.06	0.270 8	0.165 3	0.246 1	1.30
DPC-CE	0.698 9	0.741 6	0.695 2		0.173 6	0.157 4	0.151 8	

4.4　统计检验

利用 Friedman 检验和 Nemenyi 后续检验[23]验证算法之间的聚类结果是否有显著性差异。Friedman 检验作为一种无参数检验方法,它的原假设为各算法间没有显著性差异。由此,首先计算各算法在所有数据集上所得聚类指标的秩均值,值越小说明算法聚类效果越好,秩均值具体结果列于表 5 中,最小和次小秩均值分别用黑体和下划线进行标注。

表 5　算法秩均值

秩均值	NMI	ARI	AMI
CBDPC	**1.403 8**	**1.384 6**	**1.384 6**
DPCSA	6.038 5	6.384 6	6.038 5
FHC-LDP	3.807 7	3.826 9	3.692 3
ICKDP	4.519 2	5.000 0	4.692 3
DBSCAN-DPC	3.673 1	3.846 2	4.173 1
PPC	4.615 4	4.000 0	4.346 2
DPC	5.192 3	4.961 5	5.076 9
DPC-CE	6.570 0	6.596 2	6.596 2

然后,利用式(17)和式(18)分别计算各个聚类指标上的 Friedman 统计量和 F 分布值:

$$\tau_{\chi^2}=\frac{12w}{h(h+1)}\left[\sum_{i=1}^{h}z_i^2-\frac{h(h+1)^2}{4}\right] \quad , \tag{17}$$

$$\tau_F=\frac{(w-1)\tau_{\chi^2}}{w(h-1)-\tau_{\chi^2}} \quad , \tag{18}$$

其中,$h(h=8)$代表算法数量,$w(w=26)$代表数据集的数量,z_i 代表各算法在数据集上所得聚类指标的秩均值。

τ_{χ^2} 服从自由度为 $h-1=7$ 的 χ^2 分布,τ_F 服从自由度为 $(h-1,(h-1)(w-1))=(7,175)$ 的 F 分布,由临界值表可知,当显著性水平 $\alpha=0.1$ 时,F 分布的临界值为 1.751 4。对于 NMI 指标,由式(17)和式(18)可以得到其 $\tau_{\chi^2}=80.910\ 3$,$\tau_F=20.009\ 5$;对于 ARI 指标,由式(17)和式(18)可以得到其 $\tau_{\chi^2}=83.394\ 2$,$\tau_F=21.143\ 3$;对于 AMI 指标,由式(17)和式(18)可以得到其 $\tau_{\chi^2}=76.349\ 4$,$\tau_F=18.066\ 5$。由于 NMI、ARI 和 AMI 的 τ_F 均大于 1.751 4,因此拒绝 Friedman 检验的原假设,说明各个算法之间存在显著性差异。

在上述基础上,使用 Nemenyi 后续检验,验证算法两两之间是否存在显著性差异。Nemenyi 后续检验利用式(19)计算临界值域(Critical Difference, CD):

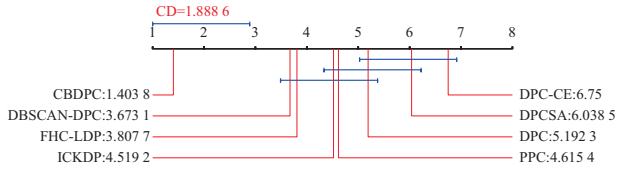
$$\mathrm{CD}=q_\alpha\left(\frac{h(h+1)}{6w}\right)^{1/2} \quad , \tag{19}$$

其中,显著水平 $\alpha=0.1$,算法数量 $h=8$,数据集数量 $w=26$,查表可知临界值 q_α 为 2.780,由此可得 CD=1.888 6。

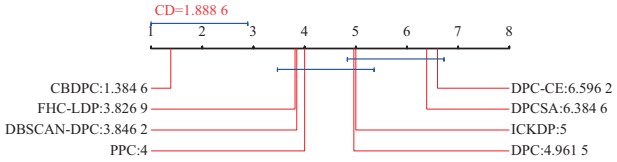
若算法两两之间的秩均值之差小于 CD 值,则说明在该聚类指标上算法间无显著性差异。图 12 给出了在各个聚类指标上的 Nemenyi 后续检验结果图。

图 12 中,若算法被线段相连,说明算法之间没有显著性差异,否则就存在显著性差异。观察表 5 和图

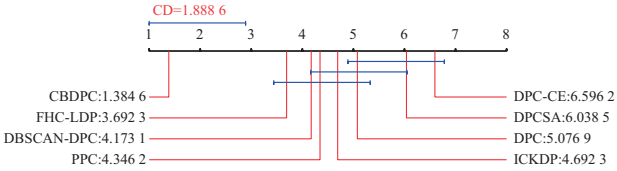
12 可知，CBDPC 算法在各个聚类指标上的秩均值均小于对比算法，且与对比算法之间没有线段相连。该结果表明，CBDPC 算法的聚类结果在 8 种算法中是最优的，且与其它对比算法两两之间存在显著性差异。通过统计检验，进一步表明了 CBDPC 算法聚类性能的优越性。



(a) NMI指标上的统计检验



(b) ARI指标上的统计检验

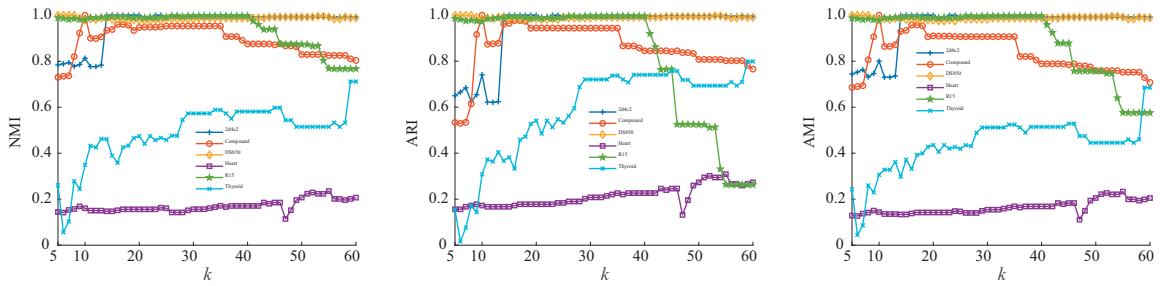


(c) AMI 指标上的统计检验

图 12 CBDPC 算法统计检验

4.5 参数分析

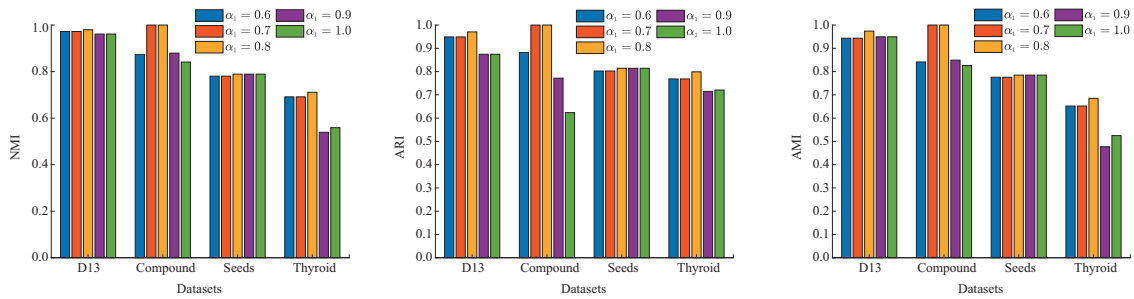
通常情况下，参数对算法的聚类性能有很大影响，首先讨论参数 k 的取值对 CBDPC 算法的影响。为了展示在不同参数取值下，CBDPC 算法聚类指标值的变化情况，选取 Thyroid、DS850、2d4c2、R15、Compound 和 Heart 数据集作为实验数据集，并在[5,60]内遍历 k 值，得到 CBDPC 算法的参数分析图，如图 13 所示。由图 13 可知，CBDPC 算法在给定范围内可以取到最优的聚类指标值，但在一些数据集上参数取值过大或者过小都会显著降低聚类性能，这说明不同的参数取值对算法的聚类性能有一定影响。



(a) NMI 指标上的参数分析 (b) ARI 指标上的参数分析 (c) AMI 指标上的参数分析

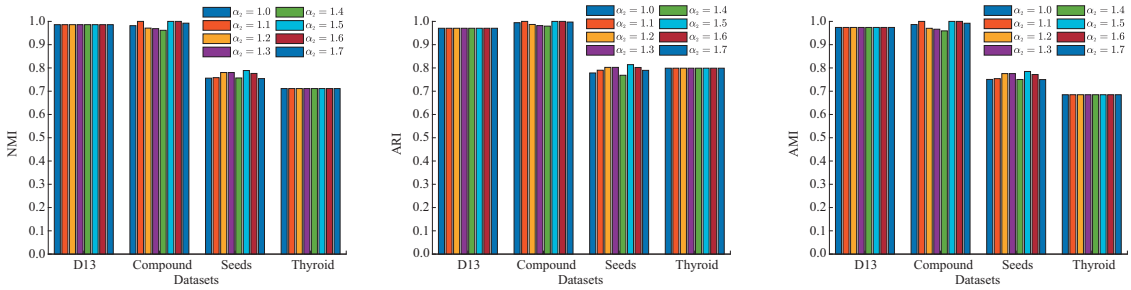
图 13 CBDPC 算法参数分析

事实上，式(10)中的系数 α_1 和式(11)中的系数 α_2 对所提算法性能也有一定影响。下面进行消融实验，验证文中设置系数 $\alpha_1=0.8$ 和 $\alpha_2=1.5$ 的有效性。消融实验选用 D13、Compound、Seeds 和 Thyroid 数据集，算法在各个数据集上的近邻值 k 均为最优配置，并使用 NMI、ARI 和 AMI 表示算法性能。首先，固定 $\alpha_2=1.5$，α_1 在[0.6,1.0]内取值，步长为 0.1，图 14 给出了不同 α_1 取值下 CBDPC 算法的聚类性能。然后，固定 $\alpha_1=0.8$，α_2 在[1.0,1.7]内取值，步长为 0.1，图 15 给出了在不同 α_2 取值下 CBDPC 算法的聚类性能。



(a) NMI 指标上的消融实验 (b) ARI 指标上的消融实验 (c) AMI 指标上的消融实验

图 14 不同 α_1 取值下 CBDPC 算法的聚类性能



(a) NMI 指标上的消融实验　(b) ARI 指标上的消融实验　(c) AMI 指标上的消融实验

图 15　不同 α_2 取值下 CBDPC 算法的聚类性能

观察图 14 和图 15 可以发现,当系数 $\alpha_1=0.8$ 和 $\alpha_2=1.5$ 时,对应图 14 中第 3 条形图和图 15 中的第 6 条形图,算法在各个数据集上可以取得最优聚类指标值。上述现象验证了文中所选系数的有效性。

4.6　运行效率分析

除了聚类性能以外,算法运行效率也是非常重要的。为了比较各算法之间的运行效率,表 6 给出了各算法在所有数据集上的运行时间。为保持公平性,算法在每个数据集上进行实验时均采用最优参数配置,并重复实验 10 次取平均值得到算法运行时间。表 6 中,各个数据集上的最短时间和次短时间分别采用黑体和下划线表示。

表 6　算法在所有数据集上的运行时间　　单位:s

数据集	运行时间							
	CBDPC	DPCSA	FHC-LDP	ICKDP	DBSCAN-DPC	PPC	DPC	DPC-CE
2d4c2	0.264	1.826	**0.021**	0.325	0.202	0.221	0.091	18.219
Ls	0.745	4.448	0.817	**0.180**	0.339	0.244	1.168	214.018
Donut2	0.297	2.512	0.305	0.703	0.220	**0.194**	0.790	25.373
Complex9	2.150	7.712	**0.918**	3.139	1.680	1.313	1.328	3.378×10^3
DB	0.226	1.958	**0.004**	0.150	0.090	0.172	0.064	4.422
R15	0.196	2.378	**0.051**	0.230	0.104	0.210	0.075	1.814
D13	0.162	2.854	**0.056**	0.169	0.083	0.210	0.074	2.293
Xclara	2.224	5.94	**0.159**	3.103	0.562	0.277	1.414	5.950
DS850	0.250	1.785	**0.007**	0.135	0.134	0.215	0.088	13.736
D3cno123	0.211	1.164	**0.016**	0.155	0.112	0.192	0.076	8.170
Compound	0.117	1.512	**0.015**	0.084	0.053	0.275	0.074	0.949
Pathbased	0.100	0.828	**0.031**	0.065	0.075	0.196	0.092	0.572
Complex8	1.542	4.772	**0.125**	1.559	0.423	0.253	0.591	896.608
Jain	0.107	1.163	**0.002**	0.075	0.073	0.180	0.054	0.777
S1	5.130	5.041	**0.352**	4.856	4.642	0.542	1.276	1.034×10^4
Disk6000n	8.918	6.141	**0.555**	3.373	4.883	1.106	2.025	2.752×10^4
Libras	0.142	0.954	**0.005**	0.079	0.081	0.222	0.065	0.167
Wine	0.093	0.568	**0.002**	0.036	0.078	0.188	0.046	0.143
Sonar	0.132	0.846	**0.003**	0.038	0.089	0.179	0.053	0.130
Pima	0.210	0.899	0.014	**0.002**	0.216	0.199	0.059	1.784
Haberman	0.097	0.793	**0.002**	0.047	0.076	0.187	0.050	0.383
DNA	2.140	6.006	**0.115**	2.587	0.744	0.234	0.240	2.385
Heart	0.384	1.251	**0.002**	0.106	0.079	0.198	0.069	0.156
Seeds	0.095	1.605	**0.002**	0.037	0.068	0.214	0.044	0.232
Thyroid	0.103	1.344	**0.002**	0.033	0.095	0.171	0.043	0.285
Vote	0.450	0.993	**0.007**	0.056	0.104	0.195	0.043	0.152
平均时间	1.019	2.588	**0.138**	0.820	0.589	0.300	0.384	1 631.989

从表 6 可知，CBDPC 算法运行效率优于 DPCSA 和 DPC-CE 算法，与 ICKDP 算法接近，比 FHC-LDP、DBSCAN-DPC、DPC 和 PPC 算法的运行效率低。具体而言，CBDPC 算法在大部分数据集上的运行时间少于 DPCSA 和 DPC-CE 算法，这是由于 DPCSA 和 DPC-CE 算法在计算样本密度和实施样本分配时耗费了大量时间，导致它们的聚类效率不高。鉴于所对比的 FHC-LDP、ICKDP 和 PPC 算法为基于 DPC 算法的快速算法，它们具有较优的运行效率，在样本规模较大的数据集(S1 和 Disk6000n 数据集)上所提 CBDPC 算法与上述快速算法有一定差距，但在一些样本规模较小的数据集(2d4c2、Jain 和 Seeds 等数据集)上，所提算法运行效率与上述快速算法相当。总的来说，CBDPC 算法的运行效率弱于除了 DPCSA 和 DPC-CE 算法以外的其它算法，原因在于 CBDPC 算法需要计算样本间的吸引度，并综合考虑样本间的近邻、密度和距离关系进行类簇生长；在样本分配阶段，CBDPC 算法需要利用边界样本周围多个样本的信息进行分配。虽然上述流程使得 CBDPC 算法在运行效率上有所损失，但算法的聚类性能也有所提高。

5 结束语

为了解决 DPC 算法在变密度数据集等复杂结构数据集上聚类效果不佳且在样本分配过程中容易发生“多米诺骨牌”现象等问题，提出 CBDPC 算法。CBDPC 算法基于样本的 k 近邻信息定义密度和相对距离，进一步计算得到决策值，依次选取决策值最高样本作为类簇中心，并根据样本间的近邻、密度和距离关系从类簇中心出发不断生长当前类簇，直到完成所有类的类簇生长，获得初始聚类结果；针对类簇生长策略后仍未分配的边界样本，定义它们与已分配类簇间的邻接度，据此将其依次分配到最合适的类簇中，获得最终聚类结果。在人工数据集和 UCI 数据集上的实验结果表明，CBDPC 算法可以解决 DPC 及其改进算法存在的问题，在各种类型的数据集上聚类性能优越，且实验结果较对比算法在统计学上具有显著性差异。然而，CBDPC 算法需要输入事先给定参数值 k，且算法的时间复杂度较高，如何根据数据集自身信息选择最优参数，并提高算法的运行效率将是下一步的研究重点。

参考文献：

[1] 周志华. 机器学习[M]. 北京：清华大学出版社，2016：121-144.

[2] LU J Z, LIU J J, HAN X X, et al. Calibration Method of Particulate Matter Sensor Based on Density Peaks Clustering Combined with Stacking Algorithm[J]. Atmospheric Environment, 2024, 326: 120460.

[3] ZHANG Z L, YIN X L, HU W, et al. Machine Learning k-Means Clustering of Interpolative Separable Density Fitting Algorithm for Accurate and Efficient Cubic -Scaling Exact Exchange Plus Random Phase Approximation within Plane Waves[J]. Journal of Chemical Theory and Computation, 2024, 20(5): 1944-1964.

[4] BIAN J H, DAI C L, LI G H. Adaptive Channel-Weight Dual-Constrained Semi-Supervised EEG Clustering[J]. Biomedical Signal Processing and Control, 2024, 98: 106720.

[5] TRON R, ZHOU X W, ESTEVES C, et al. Fast Multi-Image Matching via Density-Based Clustering [C]//2017 IEEE International Conference on Computer Vision (ICCV). Piscataway: IEEE, 2017: 4077-4086.

[6] KHILKHAL R, ISMAEL M. Brain Tumor Segmentation Utilizing Thresholding and k-Means Clustering [C]//2022 Muthanna International Conference on Engineering Science and Technology (MICEST), Piscataway, IEEE: 2022: 43-48.

[7] RODRIGUEZ A, LAIO A. Clustering by Fast Search and Find of Density Peaks[J]. Science, 2014, 344(6191): 492-1496.

[8] WANG Y Z, QIAN J X, HASSAN M, et al. Density Peak Clustering Algorithms: A Review on the Decade 2014-2023 [J]. Expert Systems with Applications, 2024, 238: 121860.

[9] 张强，周水生，张颖. 自适应密度峰值聚类算法[J]. 西安电子科技大学学报，2024，51(2)：170-181.
ZHANG Qiang, ZHOU Shuisheng, ZHANG Ying. Adaptive Density Peak Clustering Algorithm[J]. Journal of Xidian University, 2024, 51(2): 170-181.

[10] WEI X X, PENG M S, HUANG H J, et al. An Overview on Density Peaks Clustering[J]. Neurocomputing, 2023, 554: 126633.

[11] DING S F, DU W, LI C, et al. Density Peaks Clustering Algorithm Based on Improved Similarity and Allocation Strategy[J]. International Journal of Machine Learning and Cybernetics, 2023, 14(4): 1527-1542.

[12] YU D H, LIU G J, GUO M Z, et al. Density Peaks Clustering Based on Weighted Local Density Sequence and Nearest Neighbor Assignment[J]. IEEE Access, 2019, 7: 34301-34317.

[13] LI T, LI B Y, XIN X W, et al. A Novel Tree Structure-Based Multi-Prototype Clustering Algorithm[J]. Journal of King Saud University - Computer and Information Sciences, 2024, 36(3): 102002.

[14] ZANG W K, LIU X C, MA L L, et al. Density Peaks Clustering Based on Superior Nodes and Fuzzy Correlation[J]. Information Sciences, 2024, 672: 120685.

[15] HASSAN M, NAJMEH M. AnEfficient Clustering Algorithm Based on Searching Popularity Peaks[J]. Pattern Analysis and Applications, 2024, 27(2): 67.

[16] GUO W J, WANG W H, ZHAO S P, et al. Density Peak Clustering with Connectivity Estimation[J]. Knowledge-Based Systems, 2022, 243: 108501.

[17] GUAN J Y, LI S, HE X X, et al. Fast Hierarchical Clustering of Local Density Peaks via An Association Degree Transfer Method[J]. Neurocomputing, 2021, 445: 401-418.

[18] HOU J, LIN H S, YUAN H Q, et al. Flexible Density Peak Clustering for Real-World Data[J]. Pattern Recognition, 2024, 156: 110772.

[19] GUO W J, CHEN W, LIU X G. Density Peak Clustering by Local Centers and Improved Connectivity Kernel[J]. Information Sciences, 2024, 666: 120439.

[20] MILAAN P. Clustering Datasets (2022) [EB/OL]. [2024-5-11]. https://github.com/milaan9/ClusteringDatasets.

[21] BLAKE C L, MERZ C J. UCI Repository of Machine Database (2023) [EB/OL]. [2024-6-15]. https://archive.ics.uci.edu /datasets.

[22] VINH N, EPPS J, BAILEY J. Information Theoretic Measures for Clustering Comparison: Variants, Properties, Normalization and Correction for Chance[J]. Journal of Machine Learning Research, 2010, 11: 2837-2854.

[23] DONALD W Z, BRUNO D Z. Relative Power of the Wilcoxon Test, the Friedman Test, and Repeated- Measures ANOVA on Ranks[J]. The Journal of Experimental Education, 1993, 62(1):75-86.

（编辑：牛姗姗）