计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(2):195−205 [doi: 10.15888/j.cnki.csa.009770] [CSTR: 32024.14.csa.009770]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

自适应稀疏感知密度峰值聚类算法①

李欣娅, 何星星, 任芮彬

(西南交通大学 数学学院, 成都 611756)
通信作者: 何星星, E-mail: x.he@swjtu.edu.cn

摘　要: 密度峰值聚类 (density peaks clustering, DPC) 算法通过考虑局部密度和相对距离来识别簇中心以实现聚类. 然而, 该算法在处理密度分布不均匀和类簇大小不平衡的数据时容易忽视低密度区域的类簇中心, 需要人为设定类簇数量, 并且其分配策略中一个数据点分配错误会导致后续点的错误分配. 为了解决上述问题, 本文提出一种自适应稀疏感知密度峰值聚类算法. 首先, 引入模糊点概念以降低对子簇合并过程的影响; 其次, 利用减法聚类方法识别低密度区域的中心; 然后, 根据新的局部密度和反向最近邻数来识别噪声并更新子簇中心; 最后, 给出改进的全局交叠度, 结合全局可分度指导子簇融合, 并在这些度量下自动确定聚类结果. 实验结果表明, 在合成数据集和 UCI 数据集上, 与 DPC 及其改进算法相比, 本文提出的算法能够更好地识别稀疏簇、减少非中心分配带来的连锁反应, 自动确定最优类簇数目并获得更加准确的聚类结果.

关键词: 聚类分析; 密度峰值; 减法聚类方法; 反向近邻; 子簇融合

引用格式: 李欣娅,何星星,任芮彬.自适应稀疏感知密度峰值聚类算法.计算机系统应用,2025,34(2):195–205. http://www.c-s-a.org.cn/1003-3254/9770.html

Adaptive Sparse-aware Density Peaks Clustering Algorithm

LI Xin-Ya, HE Xing-Xing, REN Rui-Bin

(School of Mathematics, Southwest Jiaotong University, Chengdu 611756, China)

Abstract: The density peaks clustering (DPC) algorithm achieves clustering by identifying cluster centers based on local density and relative distance. However, it tends to overlook cluster centers in low-density regions for data with uneven density distribution and unbalanced cluster sizes. Therefore, the number of clusters needs to be set artificially. Besides, if a data point allocation occurs to be wrong in the whole strategy, it will lead to incorrect allocation of subsequent points. To address these issues, this study proposes an adaptive sparse-aware density peaks clustering algorithm. Firstly, fuzzy points are introduced to minimize their impact on the subcluster merging process. Secondly, the subtractive clustering method is used to identify the low-density regions' center. Then, noise is identified and subcluster centers are updated based on new local density and reverse nearest neighbor. Finally, a redefined global overlap metric combined with global separability guides subcluster merging while automatically determining clustering results using these metrics. Experimental results demonstrate that compared to DPC and its improved algorithms, the proposed algorithm effectively identifies sparse clusters in both synthetic and UCI datasets while reducing chain reactions caused by non-center assignments. Also, the proposed algorithm can automatically determine the optimal clustering number, ultimately yielding more accurate clustering results.

Key words: cluster analysis; density peak; subtractive clustering method; reverse nearest neighbor; subcluster fusion

① 基金项目: 中央高校基本科研业务费专项资金 (2682024ZTPY041); 四川省科技计划 (2023YFH0066); 成都市科技项目 (2023-RK00-00080-ZF)
收稿时间: 2024-07-18; 修改时间: 2024-08-13; 采用时间: 2024-09-03; csa 在线出版时间: 2024-12-16
CNKI 网络首发时间: 2024-12-17

聚类[1]是多元统计分析、数据挖掘和机器学习中的一个核心研究技术, 其核心在于依据特定的相似性度量准则, 对数据集进行无监督分类, 旨在实现类内数据点高相似度而类间数据点低相似度的目标. 该技术在图像分割[2]、模式识别[3]及生物医学[4]等多个领域展现出广泛的应用前景.

聚类算法的研究主要有两个方面: 一是与深度学习结合的神经网络类算法, 凭借神经网络的特征学习和表征能力优化聚类效果. 其中, 多视图聚类[5]融合多源数据视角, 增强聚类的鲁棒性与全面性; 深度编码聚类[6]则是运用编码-解码框架, 自动学习数据的高层次抽象表征用以聚类. 二是以传统机器学习算法为核心的统计学习方法, 此类算法多聚焦于数据的内在属性与结构特征. 如 K-means[7]通过迭代优化分区, 实现数据的紧凑簇划分; BIRCH[8]利用树状结构高效处理大规模数据集; DBSCAN[9]则对密度变化高度敏感, 能够识别任意形状的簇; Spectral clustering[10]借助图谱理论, 通过相似度矩阵的特征分解发现数据中的簇结构; CLIQUE[11]则是在高维空间中以网格划分为基础, 识别密集区域形成簇. 这些算法被广泛应用于各个领域, 并受到许多学者的深入研究与探索.

在聚类算法领域, 基于密度的聚类算法因其非迭代性、良好的可解释性和直观性而广受关注. 其中, 密度峰值聚类 (density peaks clustering, DPC)[12]是由 Rodriguez 等提出的一种新兴的密度聚类算法. 该算法核心在于构建决策图来识别簇中心, 而簇中心根据局部密度高于其邻近点以及远离其他的高密度点两个显著特征决定. 因此 DPC 算法不仅无需迭代, 还具备识别边界点和噪声点 (密度小、相对距离大) 的能力. 然而, DPC 算法也存在局限性. 首要挑战在于需要预先设定聚类数目, 这一步骤往往依赖于主观判断或额外分析. 其次, DPC 在数据点分配中可能遭遇“多米诺骨牌效应”, 即一旦某个数据点被错误分配, 可能引发一系列连锁反应, 影响整体聚类质量. 尤为关键的是, 面对密度分布不均和簇大小不平衡的数据集, DPC 的聚类效果往往难以达到预期, 限制了其在更广泛场景下的应用. 针对 DPC 算法存在的问题, 学者们分别提出了一系列改进与优化策略.

针对 DPC 需要手动设定类簇个数的问题, Yan 等提出的 ADPC 算法[13]采用二维高斯核函数进行离群点检测, 自动确定聚类质心, 从而避免了手动设定聚类的数量. Wang 等提出的基于锚点的密度聚类算法[14], 结合 DPC 和 DBSCAN 的关键机制, 通过对底层数据集的自主密度分析, 实现了更高效且准确的聚类. Tong 等提出的 DPADN 算法[15], 首先利用决策图的特性进行预聚类, 然后结合尺度空间理论进行聚类, 自动生成聚类结果, 无需人工设定参数. 为了避免人工预先指定参数, Wang 等提出了一种伪标签引导的密度峰值聚类方法[16], 首先基于共现理论生成伪标签, 然后使用互信息最大化方法来获得聚类结果. UIFDBC[17]是 Chowdhury 等提出的一种无用户输入且不依赖于任何特定的聚类有效性指标的聚类方法, 可以检测任意形状的类簇.

针对 DPC 的分配容错性差和在密度不均匀数据集上聚类效果不理想的问题, Lotfi 等提出的 DPC-DBFN 算法[18], 利用模糊核来提高聚类可分性, 减少异常值的影响, 并通过基于密度的 KNN 图标记主干信息, 有效防止了分配策略的连锁反应, 使其更适用于不同形状和密度的数据聚类. Guan 等提出的 FHC-LDP 算法[19]只考虑邻居之间的关联来避免原始算法的分配中的问题, 并通过自下而上的层次聚类快速且自动地生成类簇. Wang 等提出的多中心密度峰值聚类 McDPC 算法[20], 解决了 DPC 在处理具有多个密度峰和低密度类簇时的不足, 通过决策图划分、局部密度、相对距离和设定阈值, 该算法识别微簇并依据密度水平进行聚合.

从以往研究中不难发现, 不同研究对于 DPC 算法所面临的挑战各有侧重, 然而如何更全面应对类簇数量的自动确定难题、分配连锁错误、密度分布不均和簇大小不平衡等挑战, 将是提升 DPC 算法的性能和普适性点的关键所在. 鉴于此, 本文在剖析 DPC 算法既有的优点和局限性基础上, 提出了一种综合性的解决框架, 即自适应稀疏感知密度峰值聚类算法 (adaptive sparse-aware density peaks clustering, ASADPC). ASADPC 算法的工作主要包括: 1) 引入减法聚类方法自适应初定子簇中心, 结合新定义的局部密度与反向近邻优化中心, 精准应对密度不均以及簇大小不平衡挑战; 2) 定义综合度量自动确定类簇数, 优化合并顺序, 解决手动设定难题; 3) 两阶段噪声与模糊点识别机制, 减少分配错误, 提升聚类准确性与鲁棒性.

1 相关理论基础

1.1 密度峰值聚类

密度峰值聚类[12]的两个前提是: 聚类中心点被低

密度点包围、不同中心点之间相距较远. 由此 DPC 关键在于计算局部密度和相对距离来构建决策图, 从决策图中识别中心, 同时可以识别边界点及噪声点. 假设 D 维空间中数据集为$\{x_1,x_2,\cdots,x_n\}\subset R^D$, 对于局部密度$\rho$, DPC 计算方式包括使用截断距离和高斯核函数两种方法.

$$\rho_i=\sum_{j=1}^{n}\chi(d_{ij}-d_c),\ \chi(x)=\begin{cases}1, & x<0\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

$$\rho_i=\sum_{j=1,j\neq i}^{n}\mathrm{e}^{-\left(\frac{d_{ij}}{d_c}\right)^2} \tag{2}$$

其中, d_{ij}是数据点 i 和数据点 j 之间的欧氏距离. d_c 是截断距离, 其选取原则是其使每个数据点的平均近邻个数约为数据点总数的 2% 左右 (用 pct 表示该比例). 相对距离是指数据点到局部密度比它高且最近的点的距离, 计算公式如下:

$$\delta_i=\begin{cases}\min(d_{ij}), & \text{if } \rho_i\neq\rho_{\max}\ \&\ \rho_i<\rho_j\\ \max(d_{ij}), & \text{otherwise}\end{cases} \tag{3}$$

在获得所有数据点的局部密度和相对距离后, DPC 算法以ρ和δ作为坐标轴绘制决策图, 选取ρ和δ值都相对较大的点作为中心点. 当数据规模较大时, 选择决策值γ大的数据点作为类簇中心, γ的公式为:

$$\gamma_i=\rho_i\cdot\delta_i \tag{4}$$

在选出类簇中心之后, 赋予中心点不同的标签, 对于非中心点进行分配, 具体是将其分配到密度比它高且最近的点所在的簇, 若该点尚且无归属的簇, 则用同样的方式迭代寻找完成分配, 最终得到聚类结果.

然而, DPC 在密度分布不均匀、不平衡数据集上聚类效果不理想. 图 1(a) 是 Banana 数据集, 它由两个密差异较大的簇组成. 由于密度差异较大, DPC 计算局部密度只考虑了截断距离内的点贡献, 然而稀疏区域的截断距离内数据点密度小, 导致稀疏簇区域中心无法被识别出来 (如图 1 (b)).

1.2 减法聚类方法

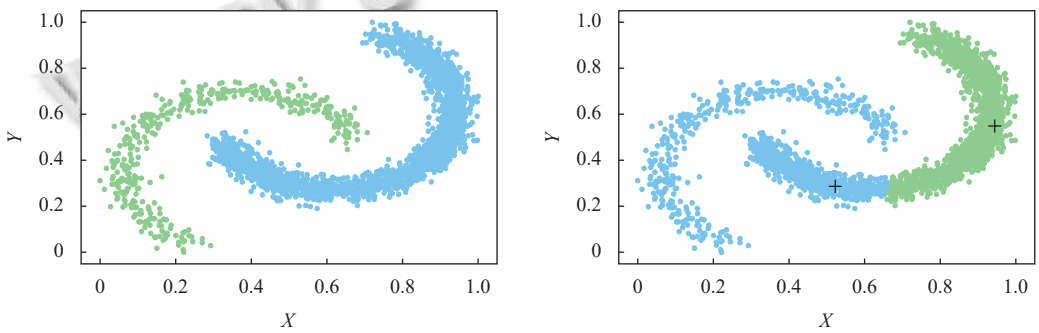
减法聚类方法[21]是一种基于密度的确定类别数的方法, 能够快速且独立地找到足够多的聚类中心. 设 D 维空间的 n 个数据点为$\{x_1,x_2,\cdots,x_n\}$, 不失一般性, 假设数据已归一化. 数据点x_i的密度指标计算如下:

$$D_i=\sum_{j=1}^{n}\exp\left(-\frac{\|x_i-x_j\|^2}{(r_a/2)^2}\right) \tag{5}$$

其中, $\|\cdot\|$是l_2范数, $r_a>0$. 从式 (5) 可知, 半径r_a定义了x_i的一个邻域, 若该范围内的样本点越多, 则x_i的密度越高, 半径以外的点对该点的密度贡献甚微. 在得到每个样本点的密度指标后, 选取具有最大值的样本点作为第 1 个聚类中心, 记为x_{c1}, 其密度指标为D_{c1}. 接下来通过如下公式对其余样本点的密度指标进行修正:

$$D_i=D_i-D_{c1}\sum_{j=1}^{n}\exp\left(-\frac{\|x_i-x_{c1}\|^2}{(r_b/2)^2}\right) \tag{6}$$

其中, r_b是一个正数, 通常$r_b>r_a$, 用于定义密度指标显著减小的范围. 显然, 靠近x_{c1}的样本点的密度指标减小得更显著, 这样使其成为下一个中心的可能性降低, 避免相距过近的样本点同时被选中. 修正后选取密度指标最大的样本点作为下一个聚类中心x_{c2}, 继续修正剩余样本点的密度指标, 重复该过程, 直至找到足够的聚类中心, 也可以设定阈值 (通常可设置为 0. 05) 自动确定类簇个数.



(a) Banana数据分布情况　　(b) DPC聚类结果

图 1　DPC 算法在 Banana 数据集上的聚类效果

减法聚类方法将所有数据点视为潜在中心, 在更新过程中, 每当一个数据点被选定为聚类中心后, 其领域半径r_a内的数据点的密度减少量会根据预设的衰减规则进行递减. 这一过程不仅确保了高密度区域中心的有

效识别, 还通过逐步“减法”操作, 有效地降低了高密度区域对周围低密度区域数据点的“遮蔽”, 从而该方法能够“穿透”高密度区, 触及并选取到密度稀疏区域的中心. 此外, 减法聚类方法常被用于通信信号调制识别中类别数确定, 在聚类领域的应用多用作模糊聚类算法初始中心的确定, 鲜有将经典的减法聚类方法与新兴的密度聚类算法 DPC 结合应用, 是一种创新的尝试.

因此, 本文考虑将减法聚类方法引入到 DPC 算法中, 用以解决 DPC 在密度分布不均匀、不平衡数据集上准确识别中心的问题.

2 自适应稀疏感知密度峰值聚类算法

2.1 子簇形成

本节将介绍本文所提出的 ASADPC 算法中的子簇形成过程以及相关重要概念.

定义 1. 模糊点[22]. 数据集中的数据点满足$\overline{Dist_i} > mean(\overline{Dist_i}) + 3std(\overline{Dist_i})$的点.

其中, $\overline{Dist_i}$为数据点x_i与其k_1个近邻的距离的均值, $mean(\cdot)$和$std(\cdot)$表示对应的均值和标准差.

定义 2. 反向 k 最近邻[23]. 对于数据点x_i的反向 k 最近邻集合满足:

1) $RNN_k(x_i) \subseteq X\backslash\{x_i\}$;

2) 如果$x_i \in KNN(x_j)$, 那么$x_j \in RNN_k(x_i)$.

在本文算法中选取 k=1, 即反向最近邻.

定义 3. 局部密度. 局部密度定义如下:

$$\rho_i = \sum_{j=1,j\neq i}^{n} e^{-\left(\frac{d_{ij}}{r_a}\right)^2} \tag{7}$$

这里截断距离定义为减法聚类方法中的半径r_a.

定义 4. 噪声点[24]. 记每个数据点的反向近邻数量为Rnn_i, 那么噪声点满足以下条件:

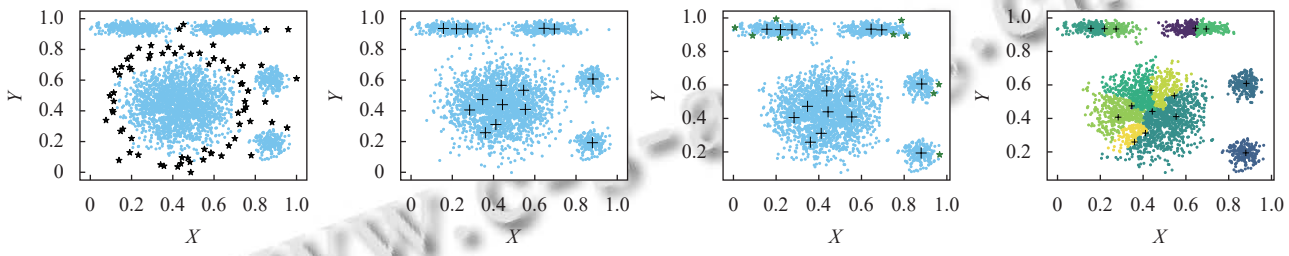
$$\rho_i < mean(\rho) - std(\rho) \tag{8}$$

$$\delta_i < mean(\delta) - std(\delta) \tag{9}$$

$$Rnn_i < mean(Rnn) - std(Rnn) \tag{10}$$

其中, ρ_i和δ_i是x_i的局部密度和相对距离, $mean(\cdot)$和$std(\cdot)$表示对应的均值和标准差.

图 2 是在 Ids2 数据集上展示本文算法的子簇形成过程. 图 2(a) 中根据定义 1 识别出的模糊点; 图 2(b) 是根据减法聚类方法确定的初始类簇中心 (用“+”标识), 由图可知, 不论大簇还是小簇都能识别出中心点. 图 2(c) 是根据式 (7)–式 (10) 进一步识别噪声点, 同时对初始中心点集合进行更新, 即若噪声点出现在该集合中, 则将此中心点从集合中删除. 最后, 每个中心代表一个子簇, 数据点依据密度指标值降序排列后, 根据其最近邻关系 (无论是直接的中心点还是通过传递关系连接的子簇成员) 被分配到相应的子簇中. 按照上述分配策略形成子簇, 得到图 2(d).



(a) 识别模糊点 (b) 确定初始类簇中心 (c) 识别噪声点 (d) 形成子簇

图 2 Ids2 数据集上的子簇形成

2.2 子簇融合及最终簇数的确定

在剩余数据点分配之后得到若干子簇, 对于子簇集合$\{C_1, C_2, \cdots, C_{\hat{k}}\}$, 本算法基于分组合并[25]的策略在合并子簇的同时决定最终的簇数, 其核心目标是优先合并实际上属于同一类别的子簇.

定义 5. 子簇间的距离[26]. 子簇间的距离是指两个子簇的数据点对之间的最小距离. 即:

$$d(C_i, C_j) = \min_{m\in C_i, n\in C_j}\{d(m,n)\} \tag{11}$$

其中, $d(m,n)$是子簇C_i中数据点 m 和子簇C_j中数据点 n 之间的欧氏距离.

定义 6. 子簇分离度量. 两个子簇C_i和C_j之间的分离度量定义如下:

$$S(C_i, C_j) = \frac{\min(|C_i|, |C_j|)}{2\sum_{m\in C_i, n\in C_j} I(d(m,n) < r_a) + \xi} + d(C_i, C_j) + d(v_i, v_j) \tag{12}$$

其中, $|\cdot|$是集合元素数量, v_i和v_j是子簇中心, $I(d(m,n)<r_a)$是示性函数. 这里的ξ是一个尺度参数, 取 0.001, 避免分母为取 0.

对于子簇$\{C_1,C_2,\cdots,C_{\hat{K}}\}$, 初始子簇各自为一组, 得到分组集合$G^{\hat{K}}=\{G_1,G_2,\cdots,G_{\hat{K}}\}$, 此时有$K=\hat{K}$组. 然后, 根据上述式 (12) 计算两两子簇之间的分离度, 每一次选择分离度最小的两个子簇所在的组进行合并操作.

定义 7. 全局交叠度. $K=\hat{K}-1$组时的全局交叠度的定义如下:

$$ol_{\hat{K}-1}=\min_{1\leqslant i,j\leqslant\hat{K}}\min_{C_p\in G_i,C_q\in G_j}S(C_p,C_q) \tag{13}$$

然而, 若实际属于不同类别的两个组被合并时, 那么它们之间的分离度应该远远高于合并属于同类别的组时的分离度. 因此, 除了式 (13), 还需计算每个组内数据点的近邻中有多少属于其他组的数据点来衡量组间分离程度.

定义 8. 全局可分度[27]. K 组时的全局分离程度的定义如下:

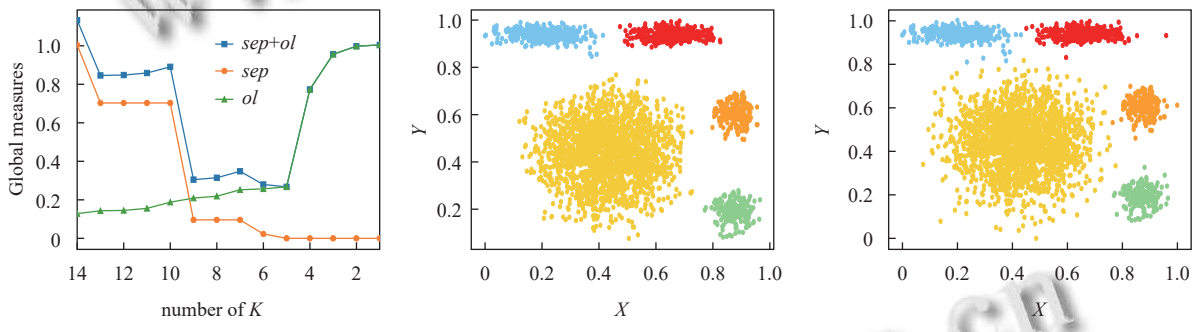
$$sep_K=\max_{i=1,2,\cdots,K}\frac{1}{|G_i|}\sum_{j\in G_i}\frac{n_j}{k_2} \tag{14}$$

其中, k_2是用户输入的近邻数目, $|G_i|$是G_i中数据点的数量, n_j是数据点j的近邻不属于G_i的数量.

结合全局交叠度与全局可分度, 最终聚类簇数的选择如下:

$$K^*=\arg\min_{1<K\leqslant\hat{K}-1}\left(\frac{sep_K}{\max_{K'}\{sep_{K'}\}}+\frac{ol_K}{\max_{K'}\{ol_{K'}\}}\right) \tag{15}$$

图 3(a)、(b) 是 Ids2 数据集得到子簇后进行子簇融合的图例, 并根据内部验证聚类度量选择最佳融合结果, 最后将模糊点和噪声点分配到离自己最近的数据点所在的类簇, 得到最终的聚类结果 (图 3(c)).



(a) 类簇数目的确定 (b) 融合结果 (c) 最终聚类结果

图 3 Ids2 数据集上的子簇的融合及最终簇数的确定

2.3 算法步骤

基于第 2.1 和 2.2 节的定义及说明, 本文的 ASADPC 算法主要分为子簇形成、子簇融合两个阶段: 第 1 阶段, 对完成模糊点筛除的数据集采取减法聚类方法预选子簇中心, 结合原始 DPC 与反向近邻完成噪声识别和中心集合更新, 最后分配非中心点完成预聚类形成子簇; 第 2 阶段, 定义子簇分离度量指导子簇合并, 结合全局交叠度与全局可分度在合并的过程中自动选择簇数, 同时得到最终的聚类结果. 在本节分别给出两个阶段的算法伪代码, 如算法 1 和算法 2.

算法 1. 子簇形成

输入: 数据集 $X=\{x_1,x_2,\cdots,x_n\}$, 参数r_a, r_b, k_1.
输出: 子簇$SC=\{C_1,C_2,\cdots,C_{\hat{K}}\}$, 中心$SV=\{v_1,v_2,\cdots,v_{\hat{K}}\}$, 噪声点集合 NP.

1. 数据归一化;
2. **for** i=1 to n **do**
3. **if** x_i满足$\overline{Dist_i}>mean(\overline{Dist_i})+3std(\overline{Dist_i})$ **then**
将x_i从X中移除, 添加到 NP;
4. **end if**
5. $i=i+1$;
6. **end for**
7. **for** i=1 to n **do**
8. 通过式 (1)、式 (2) 得到初始 SV, 密度指标D_i;
9. 根据定义 2, 得到反向最近邻数Rnn_i;
10. 根据定义 3, 计算ρ_i, 由式 (3) 计算δ_i;
11. **if** x_i满足定义 4 **then**
将x_i从X中移除, 添加到 NP;
12. **end if**
13. $i=i+1$;
14. **end for**
15. **for** i=1 to $\hat{K}$ **do**
16. **if** v_i 满足定义 4 **then**
从$\{v_1, v_2, \cdots, v_{\hat{K}}\}$中移除 v_i, 更新 SV;

17. **end if**
18. **end for**
19. 中心点$\{v_1, v_2, \cdots, v_{\hat{k}}\}$对应子簇$\{C_1, C_2, \cdots, C_{\hat{k}}\}$;
20. 将数据点按 D_i 降序排列;
21. **for** i=1 to n **do**
22. **if** $x_i \notin \{C_1, C_2, \cdots, C_{\hat{K}}\}$ **then**
23. **if** x_i 的最近邻$NN(x_i)=v_j \in SV$ **then**
24. 将 x_i 分配到v_j所在的簇;
25. **else**
26. x_i 跟随最近邻点分配到最近邻点所在的簇;
27. $i=i+1$
28. **end for**
29. **Return** $SC=\{C_1, C_2, \cdots, C_{\hat{K}}\}$, $SV=\{v_1, v_2, \cdots, v_{\hat{K}}\}$, NP.

算法 2. 子簇融合及簇数确定

输入: $SC=\{C_1, C_2, \cdots, C_{\hat{K}}\}$, $SV=\{v_1, v_2, \cdots, v_{\hat{K}}\}$, NP, 参数k_2.
输出: 聚类结果G_{k^*} 簇数K^*.

1. **for** i=1 to $\hat{K}$ **do**
2. **for** $j=i+1$ to $\hat{K}$ **do**
通过式 (12) 计算$S(C_i, C_j)$;
3. **end for**
4. **end for**
5. **for** K=1 to $\hat{K}$ **do**
6. $G_K \leftarrow \{C_K\}$;
7. **end for**
8. 记$G_{K^*} \leftarrow \{G_1, \cdots, G_{\hat{K}}\}$
9. **for** $K=\hat{K}$ to 2 **do**
10. 根据定义 7, $ol_{K-1}=\min\limits_{1\leqslant i,j\leqslant \hat{K}} \min\limits_{C_p\in G_i, C_q\in G_j} S(C_p, C_q)$;
11. 合并: $G^{K-1}=G^K \backslash \{G_i, G_j\} \cup \{G_i \cup G_j\}$;
12. 根据定义 8, 计算sep_{K-1};
13. **end for**
14. 通过式 (15) 选择最佳聚类簇数: $K^*=\arg\min\limits_{1\leqslant K\leqslant \hat{K}-1}\left(\frac{sep_K}{\max_{K'}\{sep_{K'}\}}+\frac{ol_K}{\max_{K'}\{ol_{K'}\}}\right)$, 并得到对应 G_{k^*};
15. 将集合 NP 中的点分配到最近的簇中, 得到最终 G_{k^*};
16. **Return** 聚类结果G_{K^*}, 簇数 K^*.

3 实验分析

为了验证 ASADPC 算法的聚类有效性, 分别在 8 个合成数据集和 7 个 UCI 真实数据集上进行了实验. 本文涉及的对比算法有: McDPC[20]、DPC-DBFN[18]、DPCSA[28]、DPC-CE[29]、K-means[7]、DPC[12]算法. 其中 K-means 和 DPC 算法是参照原论文 Python 编程实现, DPC-DBFN、DPCSA、DPC-CE、McDPC 算法代码由原作者提供.

关于实验中参数, 选取本实验效果最好的参数值. 具体地. 本文提出 ASADPC 算法的参数k_1和k_2在 2–50 范围内搜索, 步长为 1, r_a和r_b在 0.1–2 范围内搜索, 步长为 0. 01. McDPC 用于获得相应结果的参数值 γ、θ、λ和 pct 遵循原论文的介绍设定. DPC-DBFN 的参数k在 2–50 范围内搜索, 步长为 1. DPC 的参数d_c在[0.1, 0.2, 0.5, 1.0, 1.5, 2.0]中取值, DPC-CE 使用原论文中建议的参数值. 其中, K-means、DPCSA 和 DPC 需要给定类簇个数.

本文选用 3 个常用指标来评估聚类性能: 调整兰德系数 (adjusted Rand index, ARI)[30]、归一化互信息 (normalized mutual information, NMI)[31]和聚类准确率 (accuracy, ACC)[32]. ARI 的取值范围在[−1, 1], 值越接近 1, 表示聚类结果与真实类别越相似. NMI 和 ACC 的取值范围均为[0, 1], NMI 值越高, 说明聚类结果与真实标签共享的信息越多, 聚类效果更佳; ACC 值越高, 则表明聚类结果与真实标签匹配的样本比例越高.

3.1 数据集介绍

本文实验采用了合成数据集与 UCI 标准数据集共 15 个 (如表 1 和表 2), 用来全面验证算法性能. 在合成数据集中, 特别选取了如 Circle、Happy、Fourlines、Ids2、Banana 等二维数据集, 这些数据集不仅便于可视化分析, 还具备密度不均匀、类簇大小不平衡的特性, 部分如 Circle 和 Pathbased 数据集还展现了流形结构. 此外, 为了进一步评估算法在更广泛场景下的适用性, 还选取了 8 个特征维度在 5–30 之间的 UCI 二分类或多分类数据集, 这些真实世界数据集虽难以直观展示密度分布, 但通过样本数量可观察到不同程度的类别不平衡现象, 这样的数据集选择旨在全面评估算法在处理复杂、非均匀及不平衡数据方面的性能.

表 1 合成数据集

数据集	样本数	特征数	各簇样本数
Circle	299	2	61, 139, 99
Fourlines	512	2	150, 117, 123, 122
Gaussian	2 000	2	61, 1 212, 606, 121
Happy	266	2	118, 75, 73
Ids2	3 200	2	2 000, 200, 200, 400, 400
Banana	2 400	2	2 000, 400
Lithuanian	2 400	2	2 000, 400
Pathbased	300	2	110, 97, 93

3.2 合成数据集实验分析

表 3 列出了 7 种算法在合成数据集上的调整兰德系数 (ARI)、归一化互信息 (NMI) 和聚类准确率 (ACC), 最佳结果加粗显示. 这 3 项指标值越高, 算法性能越好. 实验通过调优参数得到实验的最佳结果. 本文提出的 ASADPC 算法在合成数据集上表现优异, 在 Circle、

Happy、Banana 和 Lithuanian 等密度分布不均的数据集上, 这 3 项指标均达到 1.00. 对于类簇大小不平衡的 Ids2 和 Gaussian 数据集, ASADPC 算法也优于其他算法. 在 Fourlines 数据集上, ASADPC 算法与 McDPC、DPC-CE 算法都能够正确聚类, 而 K-means、DPCSA 和 DPC 算法表现较差. 综上所述, ASADPC 算法在密度不均匀、簇大小不平衡的数据集上聚类性能较好.

表 2 UCI 数据集

数据集	样本数	特征数	各簇样本数
Pageblocks	5357	10	4913, 329, 115
Parkinsons	195	22	147, 48
Robotnavigation	5456	4	2205, 2097, 826, 328
Thyroid	215	5	150, 35, 30
Wine	178	13	71, 59, 48
Wdbc	569	30	357, 212
Yeast	1484	8	463, 429, 244, 163, 51, 44, 35, 30, 20, 5

表 3 合成数据集上的聚类性能比较

数据集	指标	ASADPC (ours)	McDPC	DPC-DBFN	DPCSA	DPC-CE	K-means	DPC
Circle	ARI	**1.0000**	**1.0000**	0.4606	**1.0000**	**1.0000**	0.0469	0.2141
	NMI	**1.0000**	**1.0000**	0.6489	**1.0000**	**1.0000**	0.1572	0.3450
	ACC	**1.0000**	**1.0000**	0.7199	**1.0000**	**1.0000**	0.4549	0.5686
Fourlines	ARI	**1.0000**	**1.0000**	0.8311	0.6360	**1.0000**	0.4940	0.4192
	NMI	**1.0000**	**1.0000**	0.8756	0.8037	**1.0000**	0.6672	0.5864
	ACC	**1.0000**	**1.0000**	0.9333	0.6600	**1.0000**	0.7070	0.6406
Gaussian	ARI	**0.9817**	0.9768	0.8573	0.9779	0.9661	0.6464	0.9805
	NMI	**0.9533**	0.9437	0.7784	0.9460	0.9188	0.7064	0.9502
	ACC	**0.9923**	0.9910	0.9430	0.9910	0.9832	0.8240	0.9920
Happy	ARI	**1.0000**	**1.0000**	0.5221	**1.0000**	**1.0000**	0.4810	0.6017
	NMI	**1.0000**	**1.0000**	0.6094	**1.0000**	**1.0000**	0.5832	0.6791
	ACC	**1.0000**	**1.0000**	0.8158	**1.0000**	**1.0000**	0.7932	0.8496
Ids2	ARI	**0.9886**	0.9877	0.9714	0.9858	0.9852	0.5274	0.9858
	NMI	**0.9778**	0.9765	0.9514	0.9740	0.9721	0.7432	0.9740
	ACC	**0.9959**	0.9956	0.9860	0.9950	0.9928	0.6231	0.9950
Banana	ARI	**1.0000**	0.1519	0.2264	–0.0990	–0.0767	0.0833	0.1735
	NMI	**1.0000**	0.2532	0.2879	0.0555	0.1213	0.1062	0.2647
	ACC	**1.0000**	0.6958	0.6604	0.7183	0.5034	0.6458	0.7100
Lithuanian	ARI	**1.0000**	**1.0000**	0.0487	0.348	–0.0711	0.0003	0.0401
	NMI	**1.0000**	**1.0000**	0.0395	0.3627	0.1257	0.0002	0.0063
	ACC	**1.0000**	**1.0000**	0.8333	0.8042	0.5018	0.5146	0.6804
Pathbased	ARI	**1.0000**	**1.0000**	0.7363	0.6133	0.4738	0.4613	0.4679
	NMI	**1.0000**	**1.0000**	0.5016	0.7307	0.5656	0.5458	0.5519
	ACC	**1.0000**	**1.0000**	0.7013	0.8233	0.7394	0.7433	0.7467

此外, 在图 4、图 5 中展示了一些实验中的合成数据集的聚类结果, 图中的不同颜色代表不同的类簇. 其中, DPC-DBFN、DPCSA、DPC-CE 和 DPC 算法的聚类中心用“+”号表示.

图 4 展示了 Banana 数据集上 7 种算法的实验结果图. 其中, K-means 对于非球形簇效果不佳, 而 DPC 及其变种 (DPC-DBFN、DPCSA、DPC-CE) 无法准确识别两个类簇中心, 尤其忽视了稀疏类簇的中心, 并在类簇边界上分配错误. 相比之下, ASADPC 算法通过减法聚类方法有效识别稀疏子簇中心, 并利用数据结构进行合并, 实现了正确的聚类结果. 在图 5 的 Ids2 数据集上, ASADPC 算法的聚类性能由表 3 可知, ARI、NMI 和 ACC 分别达到 98.86%、97.78%、99.59%, 均优于其他对比算法. K-means 算法由于“均匀效应”在类簇不平衡时表现不佳, 而其他算法则在类簇边界的分散点上容易出错, 本文算法通过聚类前后的模糊点与噪声点识别对数据点过滤, 有效提升了聚类性能, 进一步验证了其有效性.

3.3 UCI 数据集实验分析

本节采用 7 个来自 UCI 的数据集来验证算法的性能, 这些数据集在数据规模、特征数和类簇大小上各有差异, 用于全面评估算法的性能. 表 4 展示了 ASADPC 算法与多种对比算法的聚类性能对比. 结果显示, 本文算法在多个数据集上均取得了优秀的聚类结果.

具体而言, 在 Parkinsons、Thyroid、Wine 和 Yeast 数据集上, 本文算法在 ARI、NMI 和 ACC 这 3 项指标上均显著优于其他算法. 相较于 DPC, 在 Parkinsons 数据集上, ARI、NMI 和 ACC 分别提升了 1.32%、1.61%

和 1.95%; 在 Thyroid 数据集上, 提升幅度更是高达 36.94%、19.82% 和 27.91%; Wine 数据集上, 3 项指标较 DPC 提升了 24.25%、18.22% 和 8.99%; 在 Yeast 数据集上, 提升幅度为 16.88%、15.39% 和 10.11%. 此外, 在 Pageblocks 和 Wdbc 数据集上, 本文算法的聚类准确率 ACC 最高; 在 Pageblocks 和 Robotnavigation 数据集上, 调整兰德系数 ARI 最优; Robotnavigation 数据集上, 归一化互信息 NMI 最优.

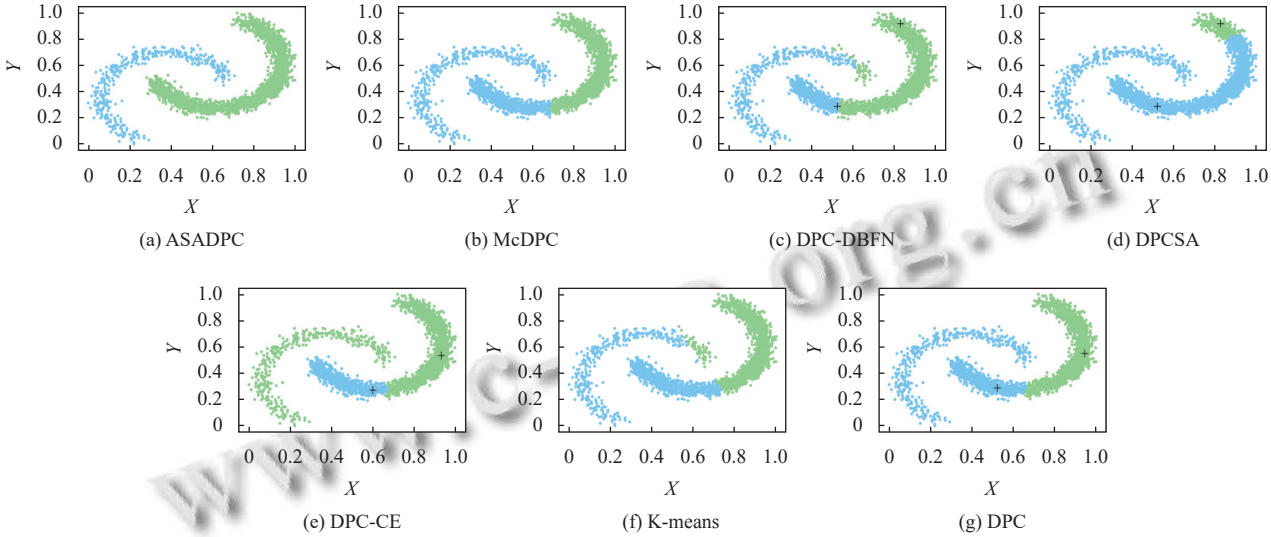


图 4 Banana 数据集上的聚类结果比较

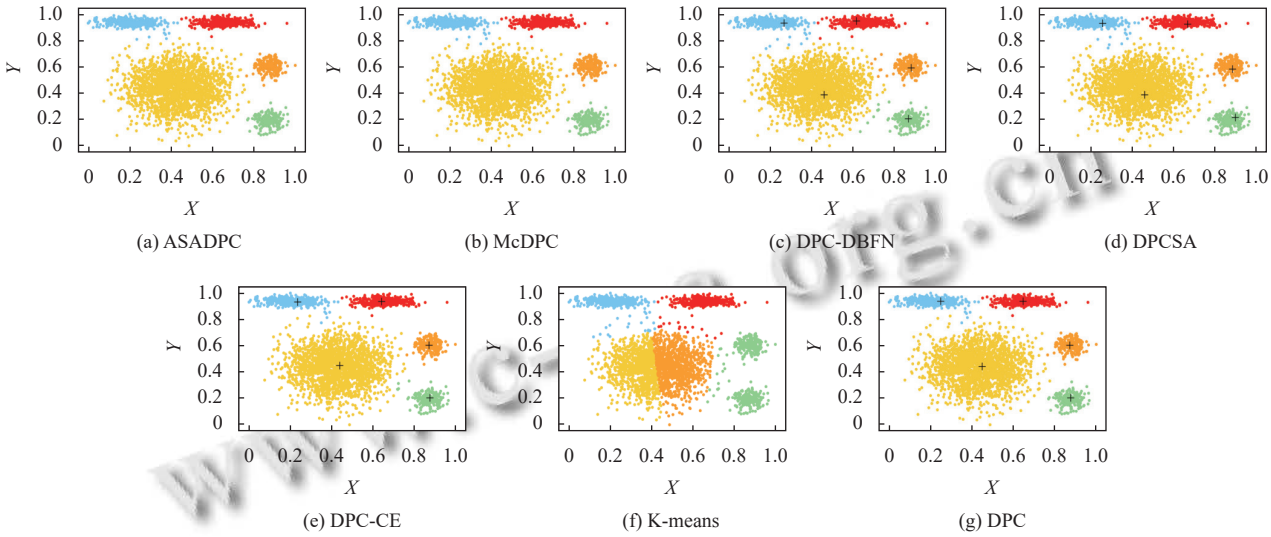


图 5 Ids2 数据集上的聚类结果比较

综上所述, 本文算法在多数真实数据集上均展现出了良好的聚类性能, 验证了其有效性和实用性.

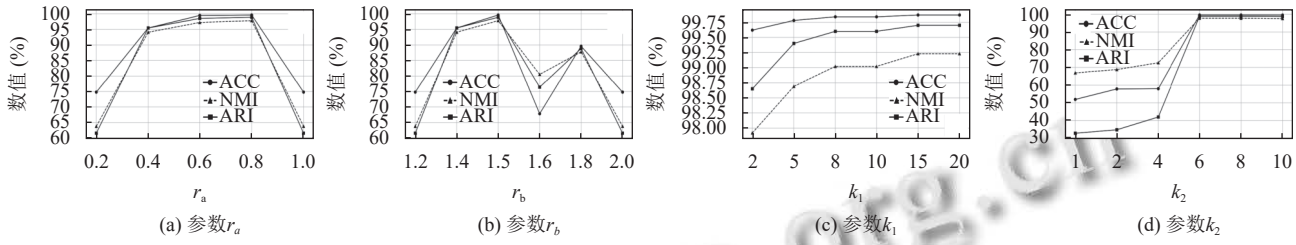
3.4 参数敏感性分析

本节对提出的 ASADPC 算法进行了参数敏感性分析. 该算法r_a、r_b、k_1和k_2, 其中r_a和r_b是第 1 阶段生成子簇中减法聚类方法中的半径, k_1是寻找模糊点时的近邻数量, 而k_2是第 2 阶段聚合中计算簇间分离程度的近邻数量. 由于 ASADPC 依赖于减法聚类方法寻找子簇初始中心, 以及基于 KNN (K-nearest neighbor) 进行聚合, 因此对r_a、r_b、k_1和k_2分别在范围 0.2–1.0、1.0–2.0、2–20、1–10 内的聚类性能进行分析. 图 6、图 7 分别展示了在数据集 Ids2、Yeasts 上 4 个参数的变化对 ASADPC 算法性能的影响. 以 Ids2 数据集为例, r_a、r_b分别在 0.6–0.8、1.4–1.5 范围内, 算法性能较

稳定, 过小或是过大的r_a、r_b都将无法准确识别出有效的初始中心, 将会影响后续聚合. 对于近邻参数k_1和k_2, 分别取值在 8–20、6–10 内, 算法性能稳定且性能较好. 实验表明, ASADPC 算法具有一定的稳定性.

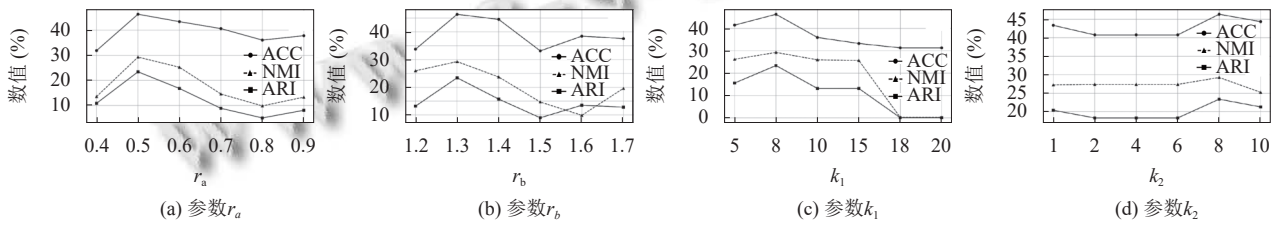
表 4 UCI 数据集聚类性能比较

数据集	指标	ASADPC (ours)	McDPC	DPC-DBFN	DPCSA	DPC-CE	K-means	DPC
Pageblocks	ARI	**0.4390**	0.0000	0.2050	0.0156	0.0197	0.0175	0.2088
	NMI	0.2857	0.0000	**0.3832**	0.0158	0.0211	0.0204	0.1315
	ACC	**0.9382**	0.9171	0.9246	0.9179	0.8471	0.8591	0.8331
Parkinsons	ARI	**0.3542**	−0.0255	0.0000	0.2686	0.0058	−0.0001	0.3410
	NMI	**0.2175**	0.0025	0.0000	0.1772	0.0005	0.00001	0.2014
	ACC	**0.8308**	0.6462	0.7538	0.8205	0.6021	0.7231	0.8113
Robotnavigation	ARI	**0.0860**	0.0753	0.0283	0.0663	0.0487	0.0725	0.0578
	NMI	**0.1981**	0.1736	0.1335	0.0938	0.1573	0.1631	0.1855
	ACC	**0.4710**	0.4850	0.4241	0.4291	0.4819	0.4078	0.4256
Thyroid	ARI	**0.5284**	0.2638	0.1617	0.0752	0.1114	0.5091	0.159
	NMI	**0.5203**	0.3007	0.243	0.0935	0.1376	0.4952	0.3221
	ACC	**0.8512**	0.5767	0.7032	0.7209	0.5794	0.8405	0.5721
Wine	ARI	**0.9149**	0.5432	0.8318	0.7414	0.2536	0.3711	0.6724
	NMI	**0.8926**	0.6812	0.7953	0.7483	0.3939	0.4288	0.7104
	ACC	**0.9719**	0.7851	0.9438	0.9101	0.6105	0.7022	0.8820
Wdbc	ARI	0.5519	**0.6734**	0.000	0.3771	0.0024	0.4914	−0.0028
	NMI	0.5107	**0.5156**	0.0000	0.3361	0.0052	0.4648	0.0048
	ACC	**0.8717**	0.8021	0.7537	0.8137	0.5326	0.8541	0.6239
Yeast	ARI	**0.2341**	0.0017	0.1143	0.0105	0.0118	0.1577	0.0210
	NMI	**0.2928**	0.0589	0.1805	0.0657	0.0713	0.2801	0.1308
	ACC	**0.4636**	0.3609	0.3477	0.3174	0.2574	0.4111	0.3073



(a) 参数r_a (b) 参数r_b (c) 参数k_1 (d) 参数k_2

图 6 Ids2 数据集上的参数



(a) 参数r_a (b) 参数r_b (c) 参数k_1 (d) 参数k_2

图 7 Yeast 数据集上的参数

4 结论与展望

针对 DPC 算法在处理密度不均与类簇不平衡数据的局限性及簇数设定难题, 本文提出了一种自适应稀疏感知密度峰值聚类算法. 该算法通过减法聚类方法初定中心, 定义局部密度并引入反向最近邻去噪, 动态更新中心并构建子簇. 随后, 基于全局交叠度与可分度融合子簇, 自动确定最优簇数, 并通过识别再分配优化模糊点与噪声点的分配. 实验验证表明, 在合成与 UCI 数据集上均展现出卓越的聚类性能.

但算法存在可进一步改进的地方. 例如本算法虽不需要提前输入真实聚类簇数, 但并未真正实现无参数聚类, 可对减法聚类方法的r_a、r_b 的选取进行改进,

减少参数的输入. 另外, 如何更加准确地划分数据集的交叉重叠部分也是今后研究的难点. 再者针对高维数据聚类, 比如图像数据, 需要研究探索如何在保证算法有效性的情况下, 降低算法的计算复杂度, 以便于在更复杂形状的数据环境下更加实用.

参考文献

1 张远鹏, 周洁, 邓赵红, 等. 代表点一致性约束的多视角模糊聚类算法. 软件学报, 2019, 30(2): 282–301. [doi: 10.13328/j.cnki.jos.005625]

2 Zhang XF, Sun YJ, Liu H, *et al*. Improved clustering algorithms for image segmentation based on non-local information and back projection. Information Sciences, 2021, 550: 129–144. [doi: 10.1016/j.ins.2020.10.039]

3 Sieranoja S, Fränti P. Fast and general density peaks clustering. Pattern Recognition Letters, 2019, 128: 551–558. [doi: 10.1016/j.patrec.2019.10.019]

4 Sunori SK, Negi PB, Joshi NC, *et al*. K-means and FCM clustering of biological oxygen demand of water. Proceedings of the 2nd International Conference on Augmented Intelligence and Sustainable Systems (ICAISS). Trichy: IEEE, 2023. 1743–1748.

5 Mi Y, Ren ZW, Xu Z, *et al*. Multi-view clustering with dual tensors. Neural Computing and Applications, 2022, 34(10): 8027–8038. [doi: 10.1007/s00521-022-06927-w]

6 Mrabah N, Khan NM, Ksantini R, *et al*. Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction. Neural Networks, 2020, 130: 206–228. [doi: 10.1016/j.neunet.2020.07.005]

7 Zhao WL, Deng CH, Ngo CW. K-means: A revisit. Neurocomputing, 2018, 291: 195–206. [doi: 10.1016/j.neucom.2018.02.072]

8 Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases. ACM SIGMOD Record, 1996, 25(2): 103–114. [doi: 10.1145/235968.233324]

9 Ester M, Kriegel HP, Sander J, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland: ACM, 1996. 226–231.

10 von Luxburg U. A tutorial on spectral clustering. Statistics and Computing, 2007, 17(4): 395–416. [doi: 10.1007/s11222-007-9033-z]

11 Agrawal R, Gehrke J, Gunopulos D, *et al*. Automatic subspace clustering of high dimensional data for data mining applications. Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. Seattle: ACM, 1998. 94–105.

12 Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492–1496. [doi: 10.1126/science.1242072]

13 Yan HQ, Wang L, Lu YG. Identifying cluster centroids from decision graph automatically using a statistical outlier detection method. Neurocomputing, 2019, 329: 348–358. [doi: 10.1016/j.neucom.2018.10.067]

14 Wang YZ, Wang D, Pang W, *et al*. A systematic density-based clustering method using anchor points. Neurocomputing, 2020, 400: 352–370. [doi: 10.1016/j.neucom.2020.02.119]

15 Tong WN, Liu S, Gao XZ. A density-peak-based clustering algorithm of automatically determining the number of clusters. Neurocomputing, 2021, 458: 655–666. [doi: 10.1016/j.neucom.2020.03.125]

16 Wang YZ, Pang W, Zhou JC. An improved density peak clustering algorithm guided by pseudo labels. Knowledge-based Systems, 2022, 252: 109374. [doi: 10.1016/j.knosys.2022.109374]

17 Chowdhury HA, Bhattacharyya DK, Kalita JK. UIFDBC: Effective density based clustering to find clusters of arbitrary shapes without user input. Expert Systems with Applications, 2021, 186: 115746. [doi: 10.1016/j.eswa.2021.115746]

18 Lotfi A, Moradi P, Beigy H. Density peaks clustering based on density backbone and fuzzy neighborhood. Pattern Recognition, 2020, 107: 107449. [doi: 10.1016/j.patcog.2020.107449]

19 Guan JY, Li S, He XX, *et al*. Fast hierarchical clustering of local density peaks via an association degree transfer method. Neurocomputing, 2021, 455: 401–418. [doi: 10.1016/j.neucom.2021.05.071]

20 Wang YZ, Wang D, Zhang XF, *et al*. McDPC: Multi-center density peak clustering. Neural Computing and Applications, 2020, 32(17): 13465–13478. [doi: 10.1007/s00521-020-04754-5]

21 Chiu SL. Fuzzy model identification based on cluster estimation. Journal of Intelligent and Fuzzy Systems, 1994, 2(3): 267–278. [doi: 10.3233/IFS-1994-2306]

22 柳德龙. 基于多子类的不平衡数据聚类算法研究 [硕士学位论文]. 西安: 西安电子科技大学, 2022.

23 Cheng DD, Huang JL, Zhang SL, *et al*. Improved density peaks clustering based on shared-neighbors of local cores for

manifold data sets. IEEE Access, 2019, 7: 151339–151349. [doi: 10.1109/ACCESS.2019.2948422]

24 Tong WN, Wang YP, Liu DL. An adaptive clustering algorithm based on local-density peaks for imbalanced data without parameters. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(4): 3419–3432. [doi: 10.1109/TKDE.2021.3138962]

25 Lu Y, Cheung YM, Tang YY. Self-adaptive multiprototype-based competitive learning approach: A K-means-type algorithm for imbalanced data clustering. IEEE Transactions on Cybernetics, 2021, 51(3): 1598–1612. [doi: 10.1109/TCYB.2019.2916196]

26 Tong WN, Wang YP, Liu DL, *et al*. A multi-center clustering algorithm based on mutual nearest neighbors for arbitrarily distributed data. Integrated Computer-aided Engineering, 2022, 29(3): 259–275. [doi: 10.3233/ICA-220682]

27 Liu YC, Li ZM, Xiong H, *et al*. Understanding and enhancement of internal clustering validation measures. IEEE Transactions on Cybernetics, 2013, 43(3): 982–994. [doi: 10.1109/TSMCB.2012.2220543]

28 Yu DH, Liu GJ, Guo MZ, *et al*. Density peaks clustering based on weighted local density sequence and nearest neighbor assignment. IEEE Access, 2019, 7: 34301–34317. [doi: 10.1109/ACCESS.2019.2904254]

29 Guo WJ, Wang WH, Zhao SP, *et al*. Density peak clustering with connectivity estimation. Knowledge-based Systems, 2022, 243: 108501. [doi: 10.1016/j.knosys.2022.108501]

30 Xu W, Liu X, Gong YH. Document clustering based on non-negative matrix factorization. Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. Toronto: ACM, 2003. 267–273.

31 Yang Y, Xu D, Nie FP, *et al*. Image clustering using local discriminant models and global integration. IEEE Transactions on Image Processing, 2010, 19(10): 2761–2773. [doi: 10.1109/TIP.2010.2049235]

32 Steinley D. Properties of the Hubert-Arabie adjusted rand index. Psychological Methods, 2004, 9(3): 386–396. [doi: 10.1037/1082-989X.9.3.386]

(校对责编: 张重毅)