软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(4):1390−1411 [doi: 10.13328/j.cnki.jos.006462]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 *K* 近邻和优化分配策略的密度峰值聚类算法*

孙 林[1,2], 秦小营[1], 徐久成[1], 薛占熬[1]

[1](河南师范大学 计算机科学与信息工程学院, 河南 新乡 453007)

[2](教育人工智能与个性化学习河南省重点实验室, 河南 新乡 453007)

通信作者: 徐久成, E-mail: xjc@htu.edu.cn

摘 要: 密度峰值聚类(density peak clustering, DPC)是一种简单有效的聚类分析方法. 但在实际应用中, 对于簇间密度差别大或者簇中存在多密度峰的数据集, DPC 很难选择正确的簇中心; 同时, DPC 中点的分配方法存在多米诺骨牌效应. 针对这些问题, 提出一种基于 *K* 近邻(*K*-nearest neighbors, *K*NN)和优化分配策略的密度峰值聚类算法. 首先, 基于 *K*NN、点的局部密度和边界点确定候选簇中心; 定义路径距离以反映候选簇中心之间的相似度, 基于路径距离提出密度因子和距离因子来量化候选簇中心作为簇中心的可能性, 确定簇中心. 然后, 为了提升点的分配的准确性, 依据共享近邻、高密度最近邻、密度差值和 *K*NN 之间距离构建相似度, 并给出邻域、相似集和相似域等概念, 以协助点的分配; 根据相似域和边界点确定初始聚类结果, 并基于簇中心获得中间聚类结果. 最后,依据中间聚类结果和相似集, 从簇中心到簇边界将簇划分为多层, 分别设计点的分配策略; 对于具体层次中的点,基于相似域和积极域提出积极值以确定点的分配顺序, 将点分配给其积极域中占主导地位的簇, 获得最终聚类结果. 在 11 个合成数据集和 27 个真实数据集上进行仿真实验, 与最新的基于密度峰值的聚类算法作对比, 结果表明: 所提算法在纯度、*F* 度量、准确度、兰德系数、调整兰德系数和标准互信息上均表现出良好的聚类性能.

关键词: 密度峰值聚类; *K* 近邻; 簇中心; 积极值; 分配策略

中图法分类号: TP311

中文引用格式: 孙林, 秦小营, 徐久成, 薛占熬. 基于 *K* 近邻和优化分配策略的密度峰值聚类算法. 软件学报, 2022, 33(4): 1390−1411. http://www.jos.org.cn/1000-9825/6462.htm

英文引用格式: Sun L, Qin XY, Xu JC, Xue ZA. Density Peak Clustering Algorithm Based on *K*-nearest Neighbors and Optimized Allocation Strategy. Ruan Jian Xue Bao/Journal of Software, 2022, 33(4): 1390−1411 (in Chinese). http://www.jos.org.cn/1000-9825/6462.htm

Density Peak Clustering Algorithm Based on *K*-nearest Neighbors and Optimized Allocation Strategy

SUN Lin[1,2], QIN Xiao-Ying[1], XU Jiu-Cheng[1], XUE Zhan-Ao[1]

[1](College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China)

[2](Key Laboratory of Artificial Intelligence and Personalized Learning in Education of Henan Province, Xinxiang 453007, China)

Abstract: The density peak clustering (DPC) algorithm is a simple and effective clustering analysis algorithm. However, in real-world practical applications, it is difficult for DPC to select the correct cluster centers for datasets with large differences of density among clusters or multi-density peaks in clusters. Furthermore, the allocation method of points in DPC has a domino effect. To address these issues, a density peak clustering algorithm based on the *K*-nearest neighbors (*K*NN) and the optimized allocation strategy was proposed. First, the candidate cluster centers using the *K*NN, densities of points, and boundary points were determined. The path distance was defined to reflect the similarity between the candidate cluster centers, based on which, the density factor and distance factor were

* 基金项目: 国家自然科学基金(62076089, 61976082, 61772176); 河南省科技攻关项目(212102210136)
本文由“面向开放场景的鲁棒机器学习”专刊特约编辑陈恩红教授、李宇峰副教授、邹权教授推荐.
收稿时间: 2021-01-10; 修改时间: 2021-07-16; 采用时间: 2021-08-27; jos 在线出版时间: 2021-10-26

proposed to quantify the possibility of candidate cluster centers as cluster centers, and then the cluster centers were determined. Second, to improve the allocation precision of points, according to the shared nearest neighbors, high density nearest neighbor, density difference, and distance between *K*NN, the similarity measures were constructed, and then some concepts of the neighborhood, similarity set, and similarity domain were proposed to assist in the allocation of points. The initial clustering results were determined according to the similarity domains and boundary points, and then the intermediate clustering results were achieved based on the cluster centers. Finally, according to the intermediate clustering results and similarity set, the clusters were divided into multiple layers from the cluster centers to the cluster boundaries, for which the allocation strategies of points were designed, respectively. To determine the allocation order of points in the specific layer, the positive value was presented based on the similarity domain and positive domain. The point was allocated to the dominant cluster in its positive domain. Thus, the final clustering results were obtained. The experimental results on 11 synthetic datasets and 27 real datasets demonstrate that the proposed algorithm has sound clustering performance in metrics of the purity, *F*-measure, accuracy, Rand index, adjusted Rand index, and normalized mutual information when compared with the state-of-the-art DPC algorithms.
Key words: density peak clustering; *K*-nearest neighbors (*K*NN); cluster center; positive value; allocation strategy

随着机器学习模型不断付诸开放应用场景, 其往往面临诸多不适[1]. 聚类是机器学习领域中一种重要的无监督学习方法, 其鲁棒性的提升, 是机器学习方法亟待解决的问题. 聚类算法是处理大数据的关键技术之一[2,3], 目前可以粗略地分为 5 类: 划分、层次、网格、模型和密度[4]. 基于划分的聚类算法需要用户指定簇的个数, 适用于凸状簇, 对异常点和噪音敏感[5]. 层次聚类算法有较高的计算复杂度, 对数据输入顺序敏感[4]. 基于网格的聚类算法遇到数据分布具有局部形状和局部密度的情况时, 性能会受到单元大小、边界的影响[6]. 基于模型的聚类算法的准确性取决于实验概率表达数据的能力[7]. 基于密度的聚类算法探测稠密区域, 将高密度区域中的点识别为簇, 将低密度区域中的点识别为异常点和噪音[8]. 因而,基于密度的聚类算法可以识别任意形状簇, 并且能够有效处理含有噪音的生物数据、多媒体数据、图像文本数据等. 但是, 密度聚类算法也面临数据集密度可变和参数敏感等问题. 在真实世界中, 大多数数据集由非球形簇或者非椭圆形簇组成, 例如一些以流形分布的人脸图像、手写数字图像以及超矩形分布的生物数据等. 事实上, 基于点的局部密度的聚类算法很容易识别任意形状和任意尺寸的簇, 而且可以有效过滤掉噪音数据[7]. 近年来, 密度聚类算法吸引了越来越多学者们的关注. 基于上述发现, 本文聚焦于基于密度策略的聚类算法, 以解决复杂数据尤其是非球形数据的分类问题.

2014 年, Rodriguez 和 Laio[7]提出了一种密度峰值聚类算法(clustering by fast search and find of density peaks, DPC), 当前已经涌现出了大量的 DPC 变体算法[9,10]. 总体来讲, DPC 能够有效探测任意形状的簇, 使用简单, 具有非常好的鲁棒性, 但是也存在一些缺点: 对于簇间密度差别大或者簇中存在多密度峰的数据集, DPC 难以为数据集选择到正确的簇中心; DPC 中点的分配方法存在多米诺骨牌效应, 针对该现象, DPC 目前无有效措施. 近年来, 针对 DPC 存在的问题, 诸多学者已做了大量研究工作. 针对 DPC 不能为密度可变的数据集选择正确的簇中心以及分配策略鲁棒性差等问题, Liu 等人[11]设计了相似度和两步点的分配方法, 提出了基于共享最近邻的 DPC 算法. 但该算法对噪声敏感, 计算复杂度高. 对于带有多密度峰的簇, DPC 难以确定这些密度峰是否在同一个簇中, 且若一个点被错误分类, 则比这个点密度低的一系列点也容易被错误分类. 为了解决这些问题, Jiang 等人[12]提出了基于 K 最近邻和基尼系数的自适应 DPC 算法. 但是, 该算法在高维数据上的准确性较低. 对于簇间密度差别大的数据集, DPC 使用绝对密度难以选出正确的簇中心. 受此启发, Hou 等人[13]构建了基于相对密度关系的簇中心识别标准增强 DPC 算法. 为了解决 DPC 不能识别多密度峰的簇以及密度相对较低的簇的问题, Wang 等人[14]基于层次方法设计了一种多中心的 DPC 算法. 事实上, DPC 的分配策略容易引起连锁反应, 即: 若一个点被分配到错误的簇中, 就很可能导致更多的点被分配到错误的簇中. 于是, Seyedi 等人[15]提出了一种动态的基于图标签传播的 DPC 算法. 但是, 该算法在聚类高维数据时存在诸多限制. 若 DPC 使用了不合适的截断距离, 则会导致簇中心的错误选择. 在某些情况下, 即使为截断距离设置正确的值, 也难以在决策图中确定簇中心. 为了解决这个问题, Liu 等人[16]提出了一种基于 *K*NN 和合并策略的自适应的 DPC 算法. 但是, 该算法在密度可变的数据集上性能较差. 由此, 针对 DPC 算法和现有改进算法在簇中心的确定和点的分配方面的不足, 为了有效确定簇中心并提升点分配的正确率, 本文提出一种基于 K

近邻和优化分配策略的密度峰值聚类算法(density peak clustering algorithm based on *K*-nearest neighbors and optimized allocation strategy, DPCKS). DPCKS 算法的主要创新点包括:

(1) 基于 *K*NN、点的局部密度和边界点给出候选簇中心, 排除大量非簇中心的同时, 保留局部密度相对较低的簇中心; 提出路径距离、密度因子和距离因子等概念, 确定一个候选簇中心作为簇中心的信任度, 依据信任度确定簇中心;

(2) 依据共享近邻、高密度最近邻、密度差值和 *K*NN 之间的距离, 提出 4 种相似度; 给出邻域、相似集和相似域等概念协助点的分配; 依据相似域和边界点确定初始聚类结果, 结合簇中心确定中间聚类结果; 由此结合相似集, 将簇划分为多层, 针对不同的层分别设计点分配策略, 确定点具体的分配顺序, 将待分配点分配到其积极域中占主导地位的簇.

1 DPC 算法

DPC 是一种粒计算模型[9], 不需要迭代过程和较多参数, 在没有先验知识下, 可通过决策图快速定位簇中心. 假设数据集 $X=\{x_1,\ldots,x_i,\ldots,x_n\}$, $x_i=[x_{i1},x_{i2},\ldots,x_{im}]$, n 表示点数量, m 是数据维度, 则任意点 x_i 的局部密度为

$$\rho_i=\sum_{i\neq j}\chi(d_{ij}-d_c) \tag{1}$$

其中, $d_{ij}=\sqrt{\sum_{g=1}^{m}(x_{ig}-x_{jg})^2}$ 表示点 x_i 和点 x_j 之间的欧式距离, d_c 是截断距离. 若 $d_{ij}-d_c<0$, 则 $\chi(d_{ij}-d_c)=1$; 否则, $\chi(d_{ij}-d_c)=0$. 一般情况下, 当点的邻居数量平均值占点总量的 1%–2%时, 就可以确定 d_c. 另外, 当数据集规模较小时, 则用高斯核函数定义任意点 x_i 的局部密度为

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

点 x_i 的高密度最近邻距离是指 x_i 到所有比其局部密度大的点的距离的最小值[17], 其公式为

$$\delta_i=\begin{cases}\min(d_{ij}),\ \rho_j>\rho_i\\ \max(\delta_j),\ i\neq j,\ \text{point } x_i \text{ with highest density}\end{cases} \tag{3}$$

DPC 算法主要步骤为: 首先, 计算所有点的局部密度和高密度最近邻距离; 其次, 以局部密度和高密度最近邻距离去构建决策图, 依据此图将具有较大局部密度和高密度最近邻距离的点选作簇中心; 然后, 将剩余点分配到其高密度最近邻所在的簇; 最后, 若一个点到其他簇任意点的距离小于 d_c, 则该点是边界点. 以 ρ_b 表示边界点中局部密度最大点的密度值, 那么对于任意簇, 局部密度大于 ρ_b 的点被称作簇核, 局部密度小于或等于 ρ_b 的点被称作簇晕.

2 改进的密度峰值聚类算法

2.1 簇中心确定

对于簇之间密度差别较大的数据集, DPC 算法易忽略密度相对较小的簇中心; 对于簇中存在多个密度峰的数据集, DPC 算法容易为一个簇选取多个簇中心. 为解决这两个问题, 本文提出新的方法确定簇中心. 为降低聚类算法的计算复杂度, 在簇中心确定之前, 从数据集中排除两类不可能成为簇中心的点: 第 1 类是边界点; 第 2 类是不满足局部密度大于或等于其 *K*NN 中所有点的点. 排除这两类点后, 剩余的点就是候选簇中心.

定义 1. 候选簇中心被描述为

$$f_1(x_i)=\begin{cases}1,\ \forall x_j\in K\mathrm{NN}(x_i),\ \rho(x_i)\geqslant\rho(x_j),\ x_i\notin bp\\ 0,\ \text{otherwise}\end{cases} \tag{4}$$

其中, $f_1(x_i)=1$ 表示 x_i 是候选簇中心; $f_1(x_i)=0$ 表示 x_i 不是候选簇中心; $K\mathrm{NN}(x_i)$ 表示 x_i 的 k 个邻居构成的集合; $\rho(x_i)$ 和 $\rho(x_j)$ 分别表示 x_i 和 x_j 的局部密度, 其计算公式参考文献[18]的公式(3); bp 表示边界点的集合.

定义 2. 假设 x_q 是 x_p 的 KNN 中的点, x_r 是 x_q 的 KNN 中的点, x_p 到 x_q 的距离相对于 x_q 到 x_r 的距离越大, x_p 就越有可能是边界点. 若以边界度来表示一个点是边界点的可能性, 则一个点的边界度越大, 它就越有可能是边界点. 一个点的边界度的计算公式表示为

$$bunDgr(x_p)=\frac{1}{|KNN(x_p)|}\sum_{x_q\in KNN(x_p)}\frac{rd(x_q)}{rd(x_p)} \tag{5}$$

其中, $rd(x_p)=\dfrac{|KNN(x_p)|}{\sum d_{pq}}$.

依据文献[4], 设置边界点的数量为 $0.2n$, 从数据集中选取边界度最大的前 $0.2n$ 个点加入公式(4)的 bp 中.

定义 3. 若一共有 o 个候选簇中心, 则以 HC 表示所有候选簇中心构成的集合为

$$HC=\{hc_1,\ldots,hc_i,\ldots,hc_o\} \tag{6}$$

为了从候选簇中心中选出簇中心, 基于候选簇中心之间的相似度计算一个候选簇中心作为簇中心的信任度. 密度聚类算法假设簇由一群密集点组成, 且这些密集点被稀疏区域分割[6]. 因此, 对于任意两个候选簇中心 hc_i 和 hc_j, 若它们之间存在稀疏区域, 说明它们不属于同一个簇, 具有较低的相似度; 否则, 具有较高的相似度. 为此, 构建一种路径距离, 放大低密度区域路段的距离并缩小高密度区域路段的距离. 那么, hc_i 和 hc_j 之间的路径距离越小, 它们之间存在稀疏区域的可能性就越低, 它们的相似度就越高. 于是, 基于相互最近邻构建图的邻接矩阵, 设置边界点与任意点之间构成的边的权重为无穷大, 对于任意两个候选簇中心 hc_i 和 hc_j, 结合邻接矩阵和 Dijkstra 算法[19]查找它们之间的最短路径 $P_{ij}=\{p_1,...,p_g,p_{g+1},...,p_t\}$.

定义 4. 任意两个候选簇中心 hc_i 和 hc_j 之间的路径距离可以表示为

$$\gamma_1(hc_i,hc_j)=\begin{cases}\sum_{g=1}^{t-1}\mathrm{e}^{d(p_g,p_{g+1})}, & d(p_g,p_{g+1})>cons\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

$$\gamma_2(hc_i,hc_j)=\begin{cases}\sum_{g=1}^{t-1}\sqrt{d(p_g,p_{g+1})}, & d(p_g,p_{g+1})\leqslant cons\\ 0, & \text{otherwise}\end{cases} \tag{8}$$

$$\gamma(hc_i,hc_j)=\gamma_1(hc_i,hc_j)+\gamma_2(hc_i,hc_j) \tag{9}$$

其中, γ_1 是指数函数放大低密度区域路段上的欧式距离, γ_2 是幂函数缩小高密度区域路段上的欧式距离, $d(p_g,p_{g+1})$是最短路径中相邻两点间的欧式距离, $cons=\dfrac{1}{t-1}\sum_{g=1}^{t-1}\mathrm{e}^{d(p_g,p_{g+1})}$ 表示最短路径中所有相邻点之间的欧式距离的平均值. 若一段路径上的欧式距离大于 $cons$, 说明该路径在低密度区域; 否则在高密度区域.

DPC 算法强调, 只有具备相对较高的局部密度和高密度最近邻距离的点才是簇中心[7].

本文以密度因子和距离因子分别代替局部密度和高密度最近邻距离, 选取密度因子和距离因子都较大的候选簇中心作为簇中心.

定义 5. 一个候选簇中心的密度因子被描述为

$$r_1(hc_i)=\mathrm{e}^{f_2(hc_i)} \tag{10}$$

下面分 3 种情况来讨论函数 $f_2(hc_i)$的取值.

(1) 当 hc_i 与任何候选簇中心之间都不存在最短路径时, hc_i 极有可能是簇中心, 设置 $f_2(hc_i)=1$;

(2) 当 hc_i 只与 hc_j 存在最短路径时, 则局部密度大的候选簇中心更有可能是簇中心. 于是, 可以设置为

$$f_2(hc_i)=\begin{cases}1, & \rho(hc_i)>\rho(hc_j)\\ 0.5, & \text{otherwise}\end{cases};$$

(3) 当 hc_i 与 u 个候选簇中心之间存在最短路径时, 假设 $u_1,\ldots,u_g,\ldots,u_u$ 是与 hc_i 存在最短路径的 u 个点, 且它们到 hc_i 的路径距离依次变远. 令 $U=\{u_1,\ldots,u_g,\ldots,u_u\}$.

实例: 为解释公式(10), 以 u_1 为例, 若 u_1 的局部密度小于 hc_i, 则 u_1 支持 hc_i 为簇中心; 否则, 反对 hc_i 为

簇中心. 若 u_1 和 u_2 都支持 hc_i 为候选簇中心, 则到 hc_i 的路径距离越小, 对 hc_i 为簇中心的支持度就越大. 如下过程计算 U 中点对 hc_i 为簇中心的支持度.

(1) 为 U 中每一个点都分配一个大于 0 的数值, u 个点的数值和为 1. 若一个点支持 hc_i 为簇中心, 则它的数值是它对 hc_i 为簇中心的支持度; 若一个点不支持 hc_i 为簇中心, 则它的数值自己保留;

(2) 对于 U 中的任意点, 到 hc_i 的路径距离越小, 为其分配的数值就会越大;

(3) 依据等差数列确定上述数值的具体值. 以 S_x, a_x, a_1, d 和 x 分别表示等差数列的前 x 项和、通项公式、首项、公差和项数. 以 $a_x>0$ 表示为 U 中第 x 个点分配的数值. 以 S_x 表示 x 个点的数值和, 则 $S_x=1$. 以 a_1 表示为 u_1 分配的数值, 因为 u_1 到 hc_i 的路径距离最小, 所以 a_1 必须最大, 即 a_x 为递减的等差数列, 所以 $d<0$. 由此, $\begin{cases} a_x = a_1 + (x-1)d > 0 \\ d < 0 \\ S_x = xa_1 + \dfrac{x(x-1)d}{2} = 1 \end{cases}$. 根据 $S_x=1$, 则 $xa_1 + \dfrac{x(x-1)d}{2} = 1$, $a_1 + \dfrac{(x-1)d}{2} = \dfrac{1}{x}$ 和 $a_1 = \dfrac{1}{x} - \dfrac{(x-1)d}{2}$. 由此可知, $a_x = \dfrac{1}{x} + \dfrac{(x-1)d}{2}$. 因为 $a_x>0$, 所以 $d > -\dfrac{2}{x(x-1)}$. 综上所述, $-\dfrac{2}{x(x-1)} < d < 0$. 令 $d_1 = -\dfrac{2}{x(x-1)}$, 设置 $d = d_1 + \dfrac{0-d_1}{2} = \dfrac{d_1}{2}$.

因为 U 中有 u 个点, 所以等差数列的项数为 u. 计算 a_1 和 d, 得出等差数列所有项的值. 假设 $a_1,\ldots,a_g,\ldots,a_u$ 表示为 u 个点分配的数值, pr_g 表示 U 中第 g 个点对 hc_i 为簇中心的支持度, 则 $pr_g = \begin{cases} a_g, & \rho(hc_i) > \rho(u_g) \\ 0, & \text{otherwise} \end{cases}$, U 中所有点对 hc_i 为簇中心的支持度为 $f_2(hc_i) = \sum_{g=1}^{u} pr_g$.

因为 $f_2(hc_i)$的最大值为 1, 所以密度因子的最大值为 e.

类比 DPC 算法, 本文假设 hc_i 到密度因子比其大且距离(指路径距离)其最近的点(指候选簇中心)的距离(指路径距离)越大, hc_i 就越有可能是簇中心. 于是, 假设 $\max(\rho(HC))$表示所有候选簇中心中密度的最大值, 对于任意一个候选簇中心 hc_i, 从路径距离分 3 种情况判断其作为簇中心的可能性.

(1) 当 $r_1(hc_i)<e$ 时, $s_1(hc_i)$表示 HC 中密度因子大于 hc_i 的所有点, hc_j 表示 $s_1(hc_i)$中与 hc_i 路径距离最小的点. 设置 $f_3(hc_i)=\gamma(hc_i,hc_j)$;

(2) 当 $r_1(hc_i)=e$ 且$\rho(hc_i)\neq\max(\rho(HC))$时, 以 s_2 表示所有密度因子为 e 的候选簇中心构成的集合. 从 s_2 中找到密度比 hc_i 大、且到 hc_i 路径距离最小的点 hc_g, 设置 $f_3(hc_i)=\gamma(hc_i,hc_g)$;

(3) 当 $r_1(hc_i)=e$ 且$\rho(x_i)=\max(\rho(HC))$时, $\max(f_3(x))$表示 $f_3(x)$在其余候选簇中心上最大值, 设置:

$$f_3(hc_i)=\max(f_3(x)).$$

对于 HC 中任意的候选簇中心 hc_k, 若 $f_3(hc_k)$为无穷大, 则将 $f_3(hc_k)$替换为函数 $f_3(x)$中除了无穷大之外的最大值.

定义 6. 一个候选簇中心的距离因子被描述为

$$r_2(hc_i) = f_3(hc_i)\mathrm{e}^{\frac{1}{k}\sum_{x_i\in KNN(hc_i)} d(x_i,hc_i)} \tag{11}$$

其中, $\dfrac{1}{k}\sum_{x_i\in KNN(hc_i)} d(x_i,hc_i)$ 是 hc_i 到其 KNN 的距离平均值. 综合考虑 r_1 和 r_2, 构建候选簇中心作为簇中心的信任度.

定义 7. 任意候选簇中心为簇中心的信任度描述为

$$cr(hc_i)=r_1(hc_i)r_2(hc_i) \tag{12}$$

由定义 7 表明: hc_i 的信任度越大, hc_i 就越有可能是簇中心.

2.2 点的相似度

为进一步排除候选簇中心以及为之后分配点的工作做准备, 构造 4 种相似度, 并提出邻域、相似集和相似域等概念. 对于任意两个点 x_i 和 x_j, 假设 $snn'(x_i,x_j)$表示 x_i 和 x_j 的共享近邻构成的集合; x_r 代表 $snn'(x_i,x_j)$中任意一个点; d_r 表示 d_{ir} 和 d_{jr} 中的最大值; $|snn'(x_i,x_j)|$越大, x_i 和 x_j 的相似度就越高. 当 $d_{ij}>d_r$ 时, x_i 和 x_j 的距离相对较远, 从 $snn'(x_i,x_j)$中删除 x_r 以降低 x_i 和 x_j 的相似度. 以 $snn(x_i,x_j)$表示从 $snn'(x_i,x_j)$中删除满足上述条件的点之后的集合. 依据文献[11], 若 x_i 和 x_j 共享近邻的数量大于 $\frac{k}{2}$, 则认为 x_i 和 x_j 具有较高的相似度.

定义 8. 基于共享近邻定义相似度 sml_1 为

$$sml_1(x_i,x_j)=\begin{cases}1, & |snn(x_i,x_j)|>t_1\\ 0, & \text{otherwise}\end{cases} \tag{13}$$

其中, $t_1=round\left(\frac{k}{2}\right)$, $round(\cdot)$表示上取整处理.

对于任意点 x_i, 假设 x_j 是 x_i 的高密度最近邻. 若 $x_j\in KNN(x_i)$, 则 x_j 是 x_i 的合格的高密最近邻点, 记作 $hdn(x_i)=x_j$. 当 $hdn(x_i)=x_j$ 时, x_i 和 x_j 具有较高的相似度.

定义 9. 基于高密度最近邻定义相似度 sml_2 为

$$sml_2(x_i,x_j)=\begin{cases}1, & hdn(x_i)=x_j, \text{ or } hdn(x_j)=x_i\\ 0, & \text{otherwise}\end{cases} \tag{14}$$

若存在点的序列 $x_1,\ldots,x_g,\ldots,x_i$ 满足 $hdn(x_1)=x_2$, $hdn(x_{g1})=x_g$ 和 $hdn(x_g)=x_{g+1}$, 即后面点是前面点的合格的高密度最近邻, 则认为在 x_i 之前的所有点都是 x_i 的后续点, 以 sub_i 表示 x_i 的所有后续点构成的集合.

假设 $\varphi_i'=\{\varphi_{i1}',\varphi_{i2}',\ldots,\varphi_{ik}'\}$ 表示 $KNN(x_i)$中的点与 x_i 的密度差值的绝对值, 将 φ_i' 中的点按照升序排列的结果为 $\varphi_i=\{\varphi_{i1},\varphi_{i2},\ldots,\varphi_{ik}\}$. 假设$\varepsilon_i$ 表示φ_i 中的第 t_1 个值.

定义 10. 基于密度差值定义相似度 sml_3 为

$$sml_3(x_i,x_j)=\begin{cases}1, & x_j\in KNN(x_i),\ |\rho(x_i)-\rho(x_j)|\leqslant\varepsilon_i\\ 0, & \text{otherwise}\end{cases} \tag{15}$$

若 x_i 和 x_j 在同一簇中且距离很小, 则 $KNN(x_i)$与 $KNN(x_j)$距离也很小. $x_1,x_2,\ldots,x_k$ 是 $KNN(x_i)$的 k 个点. $\psi_i'=\{\psi_{i1}',\psi_{i2}',\ldots,\psi_{ik}'\}$ 表示 $KNN(x_i)$ 与 $KNN(x_1),\ldots,KNN(x_k)$ 之间的距离, 将 ψ_i' 中的点升序排列为 $\psi_i=\{\psi_{i1},\psi_{i2},\ldots,\psi_{ik}\}$. 假设$\eta_i$ 表示ψ_i 中的第 t_1 个值.

定义 11. 基于 KNN 之间距离定义相似度 sml_4 为

$$sml_4(x_i,x_j)=\begin{cases}1, & x_j\in KNN(x_i),\ \dfrac{\sum\limits_{x_m\in KNN(x_i),x_n\in KNN(x_j)}d_{mn}}{k^2}\leqslant\eta_i\\ 0, & \text{otherwise}\end{cases} \tag{16}$$

依据公式(13)–公式(16)的 4 种相似度, 构造如下邻域、相似集和相似域等概念来协助点的分配.

定义 12. 对于任意点 x_i, 其邻域 N_i, W_i, R_i 和 F_i 分别表示为

$$N_i=\{x_j|sml_1(x_i,x_j)=1\} \tag{17}$$

$$W_i=\{x_j|sml_2(x_i,x_j)=1\} \tag{18}$$

$$R_i=\{x_j|sml_3(x_i,x_j)=1\} \tag{19}$$

$$F_i=\{x_j|sml_4(x_i,x_j)=1\} \tag{20}$$

假设在 N_i, W_i, R_i 和 F_i 这 4 个邻域中, $y_i(x_j)=1$ 表示只有 1 个邻域包含 x_j; $y_i(x_j)=2$ 表示只有 2 个邻域包含 x_j; $y_i(x_j)=3$ 表示只有 3 个邻域包含 x_j; $y_i(x_j)=4$ 表示 4 个邻域全都包含 x_j, 则 x_i 的相似集如下所示.

定义 13. 对于任意点 x_i, 其相似集 sN_i, sW_i, sR_i 和 sF_i 分别表示为

$$sN_i=\{x_j|y_i(x_j)=1\} \tag{21}$$

$$sW_i=\{x_j|y_i(x_j)=2\} \tag{22}$$

$$sR_i=\{x_j|y_i(x_j)=3\} \tag{23}$$

$$sF_i=\{x_j|y_i(x_j)=4\} \tag{24}$$

由定义 13 可知: 从 sN_i, sW_i, sR_i 到 sF_i, 这些相似集中的点与 x_i 的相似度依次变高.

定义 14. 对于任意点 x_i, 其相似域 dR_i, dW_i 和 dN_i 分别表示为

$$dR_i=sF_i\cup sR_i \tag{25}$$

$$dW_i=dR_i\cup sW_i \tag{26}$$

$$dN_i=dW_i\cup sN_i \tag{27}$$

对于任意点 x_i, pW_i 表示从 dW_i 中去掉边界点后的集合; 若 $x_j\in pW_i$ 且 $x_i\notin pW_j$, 则 x_i 与 x_j 的相似度较低, 从 pW_i 中删除 x_j. 假设 pW_i 中的点与 x_i 在同一簇, 确定初始聚类结果 pcr. 由于边界点的类别相对难以确定, 所以从 dW_i 中去掉边界点. 对于任意候选簇中心, 若其后续点数量少于 k+1, 则从 HC 中删除该候选簇中心; 在 pcr 中, 若多个候选簇中心属于同一簇, 则保留 cr 值最大的候选簇中心, 删除其余候选簇中心. 假设删除一部分候选簇中心后, HC 中还有 θ 个候选簇中心, 按照 cr 值降序排列后的候选簇中心为 $cc=\{cc_1,\ldots,cc_i,\ldots,cc_\theta\}$.

2.3 点的分配

由于 DPC 的分配策略容易产生多米诺骨牌效应, 于是优化分配策略, 具体包括:

(1) 根据 pcr 和簇中心确定中间聚类结果

假设点的簇标签表示点的类别, $lab(X)$表示数据集 X 中所有点的簇标签, $lab(s)$表示集合 s 中所有点的簇标签, $lab(x_i)$是点 x_i 的簇标签. 若 s 中簇标签为 l 的点的数量最多, 则称这些点构成的集合为 s 的主导簇, l 为 s 的主导类别, 记作 $label(s)=l$.

mid 表示中间聚类结果, $C=\{c_1,\ldots,c_i,\ldots,c_c\}$表示一组簇中心, sF_{ci} 和 sR_{ci} 表示簇中心 c_i 的相似集. 在 pcr 中, 若一个簇包含簇中心, 则将其加入 mid 中. 对于任意簇中心 c_i, 若其不在 pcr 的任意簇中, 则 $t_2=sF_{ci}\cup sR_{ci}$ 且 $t_3=t_2\cup c_i$, 将 t_3 作为一个簇加入 mid 中. 即: 对 C 中的任意簇中心 c_i, 将与其相似度较高的点合并构建一个簇. $mid=\{p_1,\ldots,p_i,\ldots,p_c\}$, 针对 mid 中的任意簇 p_i, 设置 p_i 中所有点的簇标签为 i, 即 $lab(p_i)=i$. 以 ass 表示已分配点的集合, rmn 表示未分配点的集合. 在下文, 只要有新的点被确定簇标签, 就更新 mid, ass 和 rmn. 关于此, 不再赘述.

(2) 根据中间聚类结果和积极域确定最终聚类结果

对于任意点 x_i, 假设 dR_i', dW_i' 和 dN_i' 分别表示 dR_i, dW_i 和 dN_i 中已经分配过的点构成的集合, 分别称它们为 x_i 的积极域 R, W 和 N, 统称 dR_i', dW_i' 和 dN_i' 为 x_i 的积极域.

定义 15. 对于任意点 x_i, 针对 dR_i', dW_i' 和 dN_i' 这 3 个集合中的任意一个集合, 其积极值, 即 vR_i, vW_i 和 vN_i 分别表示为

$$vR_i=\frac{|dR_i'|}{|dR_i|} \tag{28}$$

$$vW_i=\frac{|dW_i'|}{|dW_i|} \tag{29}$$

$$vN_i=\frac{|dN_i'|}{|dN_i|} \tag{30}$$

其中, 以$|\cdot|$表示集合中成员的数量.

由定义 15 可知: vR_i 表示在 x_i 的相似域 dR_i 中, 已经分配的点的数量占所有点的数量的百分比. vR_i 越大, 就越容易将 x_i 分配到正确的簇. vW_i 和 vN_i 有类似的含义.

若一个点被分配到错误的簇, 将会误导一系列点被分配到错误的簇. 一般情况下, 靠近簇中心的点与远离簇中心的点相比, 前者容易被分配到正确的簇. 于是, 将簇从簇中心到簇边界划分为多层, 先分配靠近簇中心层次中的点. 本文设计的点分配方法中, 不同层次点分配策略大同小异. 针对具体层次中的点, 若其积极值足够大, 就设置其簇标签为其积极域的主导类别.

接下来, 为了更容易理解本文设计的点分配方法, 将其具体过程描述如下.

(1) 对于 *ass* 中的任意点 x_i, 从 sF_i 和 sR_i 中查找未被分配的点, 加入 S 中(S 中的点即为待确定类别的点, 下文也是同样的). 假设 S 中所有点积极域的类别为积极域 R, 阈值ε=0.5, 则 S 中点的分配过程如下.

① 针对 S 中任意点 x_j, 计算其积极值 vR_j. 查找最大的积极值及其对应的点, 假设为 vR_m 和 x_m. 若 $vR_m>\varepsilon$, 转至步骤②; 否则, 结束点分配;

② 令 $lab(x_m)=label(dR_i')$, 其中, $label(dR_i')$ 表示 dR_i' 的主导类别. 从 S 中删除 x_m. 若$|S|>0$, 转至步骤①; 否则, 结束点分配.

令 $S'=S$, 删除 S 中所有的点. 步骤(2)–步骤(5)与上述点分配过程类似. 下面通过算法伪代码来描述上述 S 中点的分配过程, 详细步骤如下:

算法 1. 分配 S 中的点.

输入: 待分配点集合 S、积极域类别和参数ε;

输出: *ass*, *rmn*, *mid*, $lab(X)$和 S'.

Step 1. **While** $|S|>0$ **do**

Step 2. 计算 S 中所有点的积极值, 查找最大的积极值及其对应的点, 假设为 t 和 x_m.

Step 3. **If** $t>\varepsilon$ **then** 假设 l_1 为 x_m 的积极域的主导类别, 令 $lab(x_m)=l_1$. 从 S 中删除 x_m, 更新 *ass*, *rmn*, *mid* 和 $lab(X)$

Step 4. **Else Break** /*即不再处理 S 中剩余的点, 退出 While 循环语句*/

Step 5. **End If**

Step 6. **End While**

Step 7. 令 $S'=S$, 删除 S 中所有的点

Step 8. **Return** *ass*, *rmn*, *mid*, $lab(X)$和 S'

(2) 对于 *ass* 中任意点 x_i, 从 sW_i 中查找未被分配的点, 并入 S 中. 令 $S=S\cup S'$, 即将 S'中的点也加入 S 中. 设置 S 中所有点的积极域的类别为积极域 W, ε=0.5. 调用算法 1 分配 S 中的点. 每次调用算法 1 时, 都会返回 S 中未被成功分配的点构成的集合(即 S'). 关于此, 下文不再说明.

(3) 相对来讲, S'中的点更靠近簇中心. 所以降低点的分配条件, 即令ε=0, 尽量将 S'中的点全部分配完, 再去分配其他的点. 令 $S=S'$, 设置 S 中所有点的积极域的类别为积极域 W, 调用算法 1 分配 S 中的点.

(4) 对于 *ass* 中的任意点 x_i, 从 sN_i 中查找未被分配的点, 并入 S 中. 令 $S=S\cup S'$. 假设 S 中所有点的积极域的类别为积极域 W, 令ε=0.5. 调用算法 1 分配 S 中的点.

(5) 因为一些点靠近簇边界, 与 *ass* 中的点联系不紧密, 导致到现在也没被分配. 于是令 S=*rmn*, 假设 S 中所有点的积极域的类别为积极域 W, 令ε=0.5. 调用算法 1 分配 S 中的点.

(6) 在步骤(5)中, 因为ε=0.5 导致一些点不满足分配条件, 所以可能还有点未被分配.

在此步骤, 将 *rmn* 中所有的点全部分配完毕. 对于 *rmn* 中的任意点 x_i, 当其不满足被分配的条件时, 通过如下方式务必将其分配到一个簇: 降低分配一个点的条件, 将ε=0.5 改为ε=0; 更改一个点的分配依据, 将积极域 W 改为积极域 N. 下面通过算法伪代码来描述 *rmn* 中的点的分配过程, 详细步骤如下:

算法 2. 分配 *rmn* 中的点.

输入: 待分配点集合 *rmn*;

输出: *mid* 和 $lab(X)$.

Step 1. **While** $|rmn|>0$ **do**

Step 2. 对于 *rmn* 中任意点 x_i, 计算积极值 vW_i, 查找最大积极值及其对应的点, 设为 vW_m和 x_m

Step 3. **If** $vW_m\geqslant 0.5$ **then** 令 $lab(x_m)=label(dW_m')$, 从 *rmn* 中删除 x_m

Step 4. **Else If** $0<vW_m<0.5$ **then** 令 $lab(x_m)=label(dN_m')$, 从 *rmn* 中删除 x_m

Step 5. **Else** 对于 *rmn* 中任意点 x_j, 计算其积极值 vN_j, 查找最大积极值及其对应的点, 设为 vN_n和 x_n

Step 6. **If** $vN_n > 0$ **then** 令 $lab(x_n) = label(dN'_n)$, 从 *rmn* 中删除 x_n

Step 7. **Else** 从 *rmn* 中查找局部密度最大的点 x_o, 以 x_o 作为一个新的簇, 从 *rmn* 中删除 x_o

Step 8. **End If**

Step 9. **End If**

Step 10. 更新 *ass*, *mid* 和 *lab*(*X*)

Step 11. **End While**

Step 12. **Return** *mid* 和 *lab*(*X*)

(7) 对于 *mid* 中的任意簇 p_i, 若$|p_i|<11$, 则 p_i 为异常簇, p_i 中的点为异常点. 此步骤查找并重新分配 *mid* 中所有的异常点. 假设 x_i 为任意一个异常点, 其分配过程为: 以 S_1 和 S_2 分别表示从 dW_i 和 dN_i 中去掉异常点后的集合. 若$|S_1|>0$, 令 $lab(x_i)=label(S_1)$; 若$|S_1|=0$, $|S_2|>0$, 令 $lab(x_i)=label(S_2)$; 若$|S_1|=0$, $|S_2|=0$, 则不重新分配 x_i.

以上步骤完成了点的分配, 每一步代表一个分配策略, 经过层层分配, 尽量确保每个点都分配到与它最相似的点所在的簇. 其中, 步骤(1)、步骤(2)、步骤(4)和步骤(6)是主要步骤, 它们之间的区别在于点分配对象不同、点分配依据不同以及点分配条件不同; 其余 3 个步骤仅仅是针对少数点设计的分配策略, 与步骤(2)的设计基本相同. 至此, 形成最终的聚类结果.

2.4 DPCKS算法描述

算法 3. DPCKS 算法.

输入: 数据集 *X* 和邻居数量 *k*;

输出: *lab*(*X*).

Step 1. 归一化数据, 使用欧式距离计算距离矩阵 *z*, 根据 *z* 确定 *K*NN

Step 2. 计算所有点的局部密度, 并查找高密度最近邻

Step 3. 根据 *z* 和 *K*NN 确定边界点, 并使用公式(4)确定候选簇中心

Step 4. 构建邻接矩阵, 并获取候选簇中心之间的最短路径

Step 5. 使用公式(9)–公式(12)分别计算路径距离、密度因子、距离因子和信任度

Step 6. 使用公式(13)–公式(16)分别计算 4 种相似度, 查找后续点, 构造任意点的邻域、相似集和相似域

Step 7. 获取初始聚类结果 *pcr*

Step 8. 由后续点和 *pcr* 排除一部分候选簇中心, 选定一组簇中心 *C*. 由 *pcr* 和簇中心确定中间聚类结果 *mid*

Step 9. 调用算法 1 和算法 2, 分配所有未确定簇标签的点, 更新 *lab*(*X*)

Step 10. **Return** *lab*(*X*)

假设调用算法 1 之前 *S* 中有 v_1 个点, 调用算法 2 之前 *rmn* 中有 v_2 个点, 簇中心数量为 *c*, *pcr* 中有 v_3 个簇, 则算法 3 的时间复杂度分析如下.

- Step 1 的复杂度为 $O(n^2)$;
- Step 2 计算局部密度和查找高密度最近邻的复杂度小于 $O(n)+O(n^2)$;
- Step 3 确定边界点和候选簇中心的复杂度分别为 $2O(n)$和 $0.8O(n)$;
- Step 4 确定最短路径的复杂度为 $3O(n)+o^2O(n^2)$;
- Step 5 的复杂度小于 $3O(o^2)$;
- Step 6 计算 4 种相似度与查找后续点的复杂度小于$(k^2+k+3)O(n)+O(n^2)+O(o)$, 查找所有点邻域、相似集和相似域的复杂度为 $O(n)$, 所以 Step 6 总的复杂度小于$(k^2+k+4)O(n)+O(n^2)+O(o)$;
- Step 7 获取初始聚类结果的复杂度为 $2O(n)$;
- Step 8 的复杂度是 $O(v_3)+O(c)$;
- Step 9 由于点分配方法调用 5 次算法 1 和 1 次算法 2, 故 Step 1–Step 6 的复杂度小于 $5O(v_1^2)+2O(v_2^2)$;

假设共有 v_4 个异常点, 分配异常点的复杂度为 O(v_4), 则 Step 9 的复杂度为 $5O(v_1^2)+2O(v_2^2)+O(v_4)$.

综上所述, 算法 3 总的时间复杂度小于$(3+o^2)O(n^2)+(k^2+k+12.8)O(n)+2O(v_2^2)+5O(v_1^2)+3O(o^2)+O(o)+O(v_3)+O(v_4)+O(c)$, 近似为 $O(n^2)$, 分别低于文献[11,20]中算法的时间复杂度, 接近于文献[12−16,21−26]中算法的时间复杂度.

3 实验结果与分析

3.1 实验准备

本节采用不同规模、不同维度的数据集来验证 DPCKS 算法的有效性, 具体信息见表 1. 实验环境配置为 Windows 7, 32GB RAM, Intel Core i7-8700@3.20GHz CPU 以及 Matlab R2020a. 评价指标包括纯度(purity)[20]、F 度量(F-measure, FM)[27]、准确度(accuracy, Acc)[12]、兰德系数(Rand index, RI)[28]、调整兰德系数(adjusted Rand index, ARI)[28]和标准互信息(normalized mutual information, NMI)[29]. 数据预处理包括降维和归一化操作, 通过主成分分析[30]删除累计贡献率(以 R_c 表示)大于一定百分比的特征. 文献[11]将 R_c 设置为 90%. 但在数据集 Sonar 和 Mfeat 上, 若 R_c=90%, 选择的主成分数量少. 于是, 对于 Sonar 数据集, 设置 R_c=93.0286%, 保留对原始变量贡献率大于 1%的主成分选择前 13 个主成分; 对于 Mfeat 数据集, 设置 R_c=99.2023%, 保留对原始变量贡献率大于 0.1%的主成分选择前 11 个主成分. 此外, 在 Yale 数据集上, R_c=90%时聚类效果差, 于是设置 R_c=85%. 表 2 展示降维时不同数据集的 R_c. 对数据集进行归一化处理使用的公式为

$$x'_{ij}=\frac{x_{ij}-\min(x_j)}{\max(x_j)-\min(x_j)} \tag{31}$$

其中, x_{ij} 是点 x_i 在第 j 个特征上的取值, min(x_j)和 max(x_j)分别表示在数据集 X 上第 j 个特征的最小值和最大值. 依据文献[11], 设置 $3\leqslant k\leqslant 25$, 于是, 簇中心有 23 种取值. 若 DPCKS 的聚类效果较差, 则 k 值不变, 在剩余 22 组簇中心上寻优. 对于 Waveform, Waveform(noise)和 Wifilocalizatition 这 3 个数据集, $3\leqslant k\leqslant 30$ 时聚类效果较好.

表 1 38 个数据集的信息描述

序号	数据集	实例数	特征数	簇数	序号	数据集	实例数	特征数	簇数
1	Dim128	1 024	128	16	20	WDBC	569	30	2
2	Dim512	1 024	512	16	21	Seeds	210	7	3
3	Zelnik3	266	2	3	22	Diabetes	768	8	2
4	Compound	399	2	6	23	Breast	699	9	2
5	Four-lines	512	2	4	24	Pima	768	8	2
6	Spiral	312	2	3	25	Mfeat	2 000	649	10
7	Jain	373	2	2	26	Control	600	60	6
8	Flame	240	2	2	27	Waveform	5 000	21	3
9	Pathbased	300	2	3	28	Waveform(noise)	5 000	40	3
10	Aggregation	788	2	7	29	Banknotes	1 372	4	2
11	R15	600	2	15	30	Parkinsons	197	23	2
12	Olivetti faces	400	92×112	40	31	Dermatology	366	33	6
13	USPS	9 298	256	10	32	Wifilocalization	2 000	7	4
14	Yale	165	64×64	15	33	Libras movement	360	90	15
15	Coil20	1440	1024	20	34	Vehicle	846	18	4
16	Wine	178	13	3	35	Ionosphere	351	34	2
17	Ecoil	336	8	8	36	Wholesale	440	6	3
18	Glass	214	10	6	37	Thyroid	215	5	3
19	Sonar	208	60	2	38	Iris	150	4	3

表 2 8 个数据集的维度约简

数据集	R_c (%)	数据集	R_c (%)	数据集	R_c (%)	数据集	R_c (%)
Ionosphere	90	USPS	90	Coil20	90	Sonar	93.0286
Control	90	Libras movement	90	Yale	85	Mfeat	99.2023

3.2 合成数据集上的实验结果

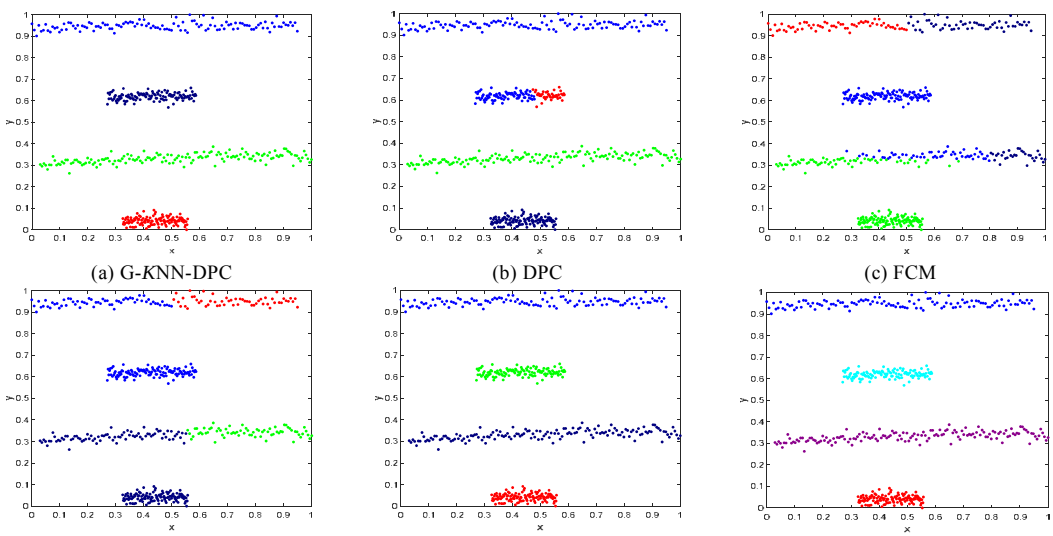
本节在 6 个合成数据集上评价 DPCKS 的聚类性能. 在评价指标 Acc, NMI 和 RI 上, 将 DPCKS 与 6 个先进的聚类算法作做实验对比, 包括 G-*K*NN-DPC (adaptive DPC based on *K*NN and Gini coefficient)[12], DPC[7], FCM (fuzzy c-means clustering)[31], *K*-Means[5], KDPC(DPC based on kernel)[26]和 DBSCAN (density-based algorithm for discovering clusters in large spatial databases with noise)[32]. 在 Four-lines, Jain, Spiral, Aggregation, Zelnik3 和 R15 这 6 个数据集上, DPCKS 的最优参数 *k* 分别设置为 10, 10, 6, 14, 11 和 22. G-*K*NN-DPC 和 KDPC 的实验参数见文献[12]. FCM 和 *K*-Means 的输入参数均为数据集中簇的数量. DPC 和 DBSCAN 的最优参数见表 3. 表 3 中, t_5 表示所有点的邻居数量的平均值占数据点总量的百分比, *MinPts* 表示点的个数, *eps* 表示邻域半径. 表 4 展示了实验的对比结果. 图 1–图 6 呈现了 6 个算法在 6 个合成数据集上的聚类效果.

表 3 DPC 和 DBSCAN 在 6 个合成维数据集上的最优参数

数据集	Four-lines	Jain	Spiral	Aggregation	Zelnik3	R15
t_5 (%)	1.4	0.3	1.7	3.1	4.2	0.3
MinPts/*eps*	12/0.0558	2/0.0744	2/0.0387	6/0.0411	6/0.0819	12/0.0378

表 4 7 种算法在 6 个合成数据集上的聚类结果

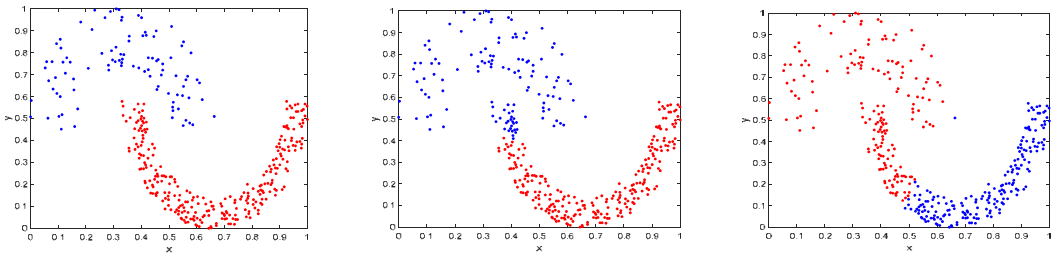
算法	Four-lines			Jain			Spiral		
	Acc	NMI	RI	Acc	NMI	RI	Acc	NMI	RI
G-*K*NN-DPC	**1**	**1**	**1**	**1**	**1**	**1**	0.994	0.971	0.991
DPC	0.841 8	0.844 1	0.901 5	0.924 9	0.653 1	0.860 8	**1**	**1**	**1**
FCM	0.660 2	0.616 4	0.794 5	0.774 8	0.357 1	0.650 1	0.339 7	0.000 2	0.554 1
K-Means	0.718 8	0.673 5	0.796 5	0.785 5	0.370 6	0.662 1	0.346 2	0.000 5	0.554
KDPC	0.699	0.635	0.743	**1**	0.994	0.999	**1**	**1**	**1**
DBSCAN	**1**	**1**	**1**	0.975 9	0.930 5	0.988 5	**1**	**1**	**1**
DPCKS	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Average	0.845 7	0.824 1	0.890 8	0.923	0.757 9	0.880 1	0.811 4	0.710 2	0.871 3
算法	Zelnik3			R15			Aggregation		
	Acc	NMI	RI	Acc	NMI	RI	Acc	NMI	RI
G-*K*NN-DPC	**1**	**1**	0.837	**1**	**1**	**1**	**1**	**1**	**1**
DPC	0.939 8	0.845 8	0.920 6	**1**	**1**	**1**	0.998 7	0.995 7	0.999 3
FCM	0.751 9	0.544 1	0.734 1	0.995	0.993 4	0.998 7	0.795 7	0.857	0.922 1
K-Means	0.744 4	0.537 2	0.729	0.895	0.954 6	0.985 9	0.753 8	0.845 4	0.908 2
KDPC	0.711	0.507	0.707	0.997	0.989	0.986	0.741	0.874	0.901
DBSCAN	**1**	**1**	**1**	0.99	0.987 6	0.998 2	0.988 6	0.974 9	0.994 6
DPCKS	**1**	**1**	**1**	0.996 7	0.995 1	0.999 1	0.998 7	0.995 8	0.999
Average	0.878 2	0.776 3	0.846 8	0.982	0.988 5	0.995 4	0.896 6	0.934 7	0.960 6



(a) G-*K*NN-DPC (b) DPC (c) FCM

(d) *K*-Means (e) DBSCAN (f) DPCKS

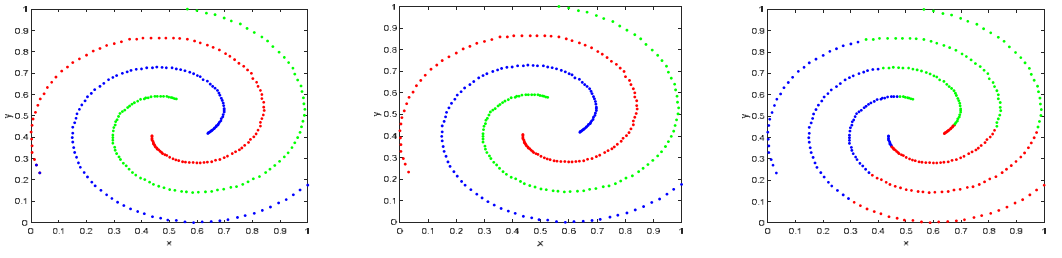
图 1 6 种算法在 Four-lines 数据集上的聚类结果



(a) G-*K*NN-DPC (b) DPC (c) FCM

(d) *K*-Means (e) DBSCAN (f) DPCKS

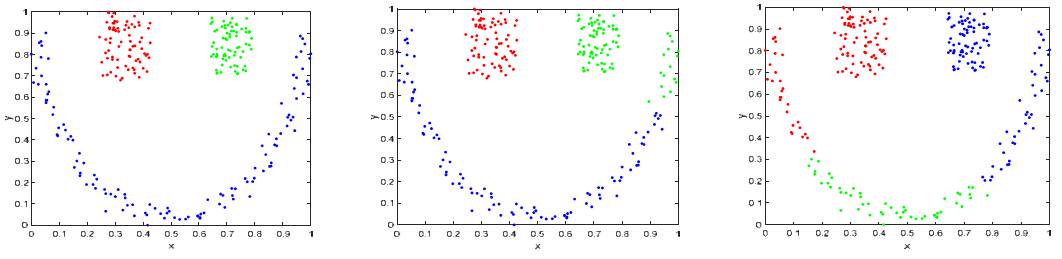
图 2 6 种算法在 Jain 数据集上的聚类结果



(a) G-*K*NN-DPC (b) DPC (c) FCM

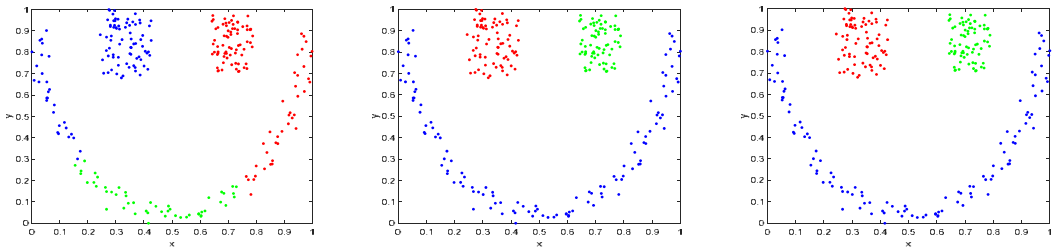
(d) *K*-Means (e) DBSCAN (f) DPCKS

图 3 6 种算法在 Spiral 数据集上的聚类结果



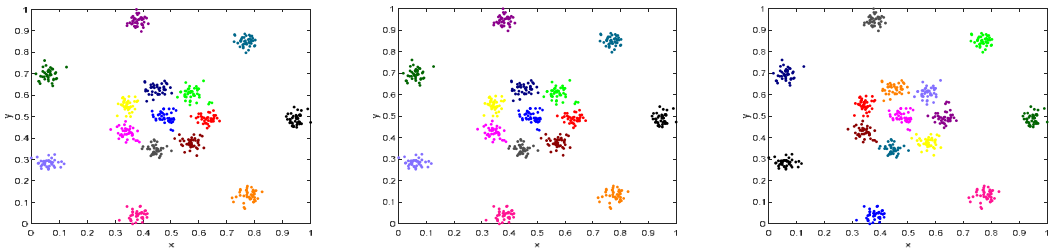
(a) G-*K*NN-DPC (b) DPC (c) FCM

图 4 6 种算法在 Zelnik3 数据集上的聚类结果

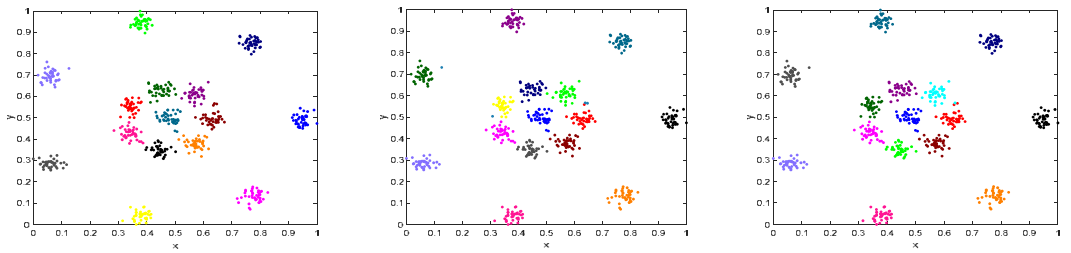


(d) *K*-Means (e) DBSCAN (f) DPCKS

图 4 6 种算法在 Zelnik3 数据集上的聚类结果(续)

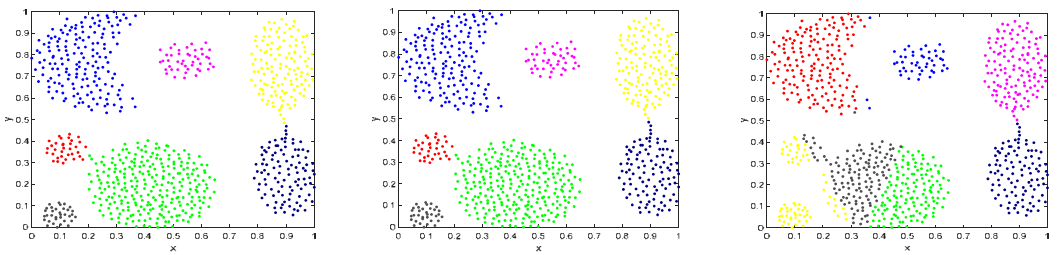


(a) G-*K*NN-DPC (b) DPC (c) FCM

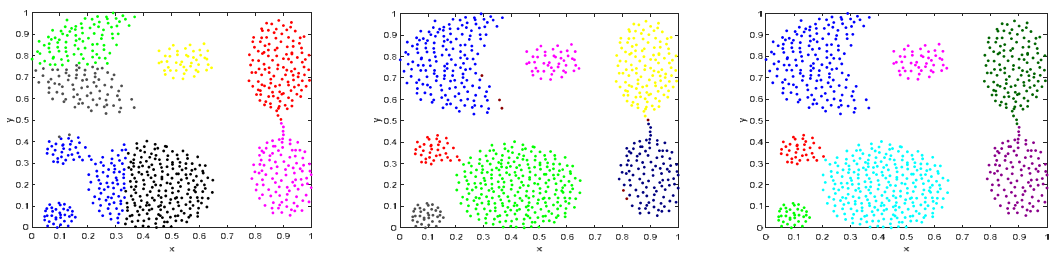


(d) *K*-Means (e) DBSCAN (f) DPCKS

图 5 6 种算法在 R15 数据集上的聚类结果



(a) G-*K*NN-DPC (b) DPC (c) FCM



(d) *K*-Means (e) DBSCAN (f) DPCKS

图 6 6 种算法在 Aggregation 数据集上的聚类结果

从表 4 可以看出: 在 Four-lines, Jain, Spiral 和 Zelnik3 这 4 个数据集上, DPCKS 呈现了完全正确的聚类结果; 在 R15 和 Aggregation 这 2 个数据集上, DPCKS 的聚类结果接近于完全正确. G-*K*NN-DPC 的聚类性能与 DPCKS 不相上下, 其他 5 个聚类算法相比于 DPCKS 和 G-*K*NN-DPC 效果次之. 图 2 的 Jain 数据集簇间密度差别大, DPC 没有选出局部密度相对较小的簇中心. 图 3 的 Spiral 和图 4 的 Zelnik3 是形状相对复杂的 2 个数据集. 图 5 的 R15 数据集簇间联系时而紧密时而松散. 图 6 中, Aggregation 数据集簇间存在异常点干扰点的分配. 对于上述 4 个不同类型的数据集, DPCKS 都呈现了良好的聚类结果, 其余聚类算法效果稍差.

3.3 UCI数据集上的实验结果

本节在 Acc, NMI 和 RI 上评价 DPCKS 的聚类性能. 在 6 个 UCI 数据集上, 将 DPCKS 与 G-*K*NN-DPC[12], DPC[7], FCM[31], *K*-Means[5], KDPC[26] 和 DBSCAN[32] 作实验比较. 在 Iris, Seeds, Breast, Banknotes, Wifilocalization 和 Wholesale 这 6 个数据集上, DPCKS 的最优参数 k 分别设置为 11, 14, 10, 18, 29 和 10, 其他算法的实验参数参照文献[12]. 表 5 展示了这 7 种算法在 UCI 数据集上的聚类结果. 从表 5 可以看出: 对于 Iris, Seeds, Breast 和 Banknotes 中的任意数据集, DPCKS 的 3 个指标值都是最优的. 对于数据集 Wifilocalization 和 Wholesale, DPCKS 仅在部分指标上的聚类性能最优. 总体而言, DPCKS 的聚类效果最好, 其余算法次之.

表 5 7 种算法在 6 个 UCI 数据集上的聚类结果

算法	Iris			Seeds			Breast		
	Acc	NMI	RI	Acc	NMI	RI	Acc	NMI	RI
G-*K*NN-DPC	0.727	0.467	0.82	0.866	0.443	0.767	0.711	0.355	0.589
DPC	0.64	0.443	0.783	0.227	0.125	0.559	0.424	0.292	0.599
FCM	0.527	0.369	0.78	0.539	0.075	0.502	0.609	0.303	0.524
K-Means	0.547	0.401	0.717	0.6	0.441	0.519	0.612	0.157	0.525
KDPC	0.567	0.316	0.78	0.746	0.151	0.697	0.414	0.027	0.503
DBSCAN	0.333	0.292	0.329	0.314	0.132	0.72	0.491	0.036	0.732
DPCKS	**0.973 3**	**0.914 4**	**0.965 6**	**0.914 3**	**0.724 2**	**0.895 6**	**0.97**	**0.817 7**	**0.941 6**
Average	0.616 3	0.457 5	0.739 2	0.600 9	0.298 7	0.665 7	0.604 4	0.284	0.630 5
算法	Banknotes			Wifilocalization			Wholesale		
	Acc	NMI	RI	Acc	NMI	RI	Acc	NMI	RI
G-*K*NN-DPC	0.732	0.214	0.607	**0.97**	0.003 3	**0.96**	0.561	0.021	**0.553**
DPC	0.674	0.108	0.559	0.761	0.003 2	0.905	0.514	0.014	0.489
FCM	0.677	0.206	0.562	0.962	0.003 3	0.489	0.545	0.001	0.498
K-Means	0.671	0.255	0.557	0.957	0.003 2	0.498	0.532	0.002	0.538
KDPC	0.626	0.24	0.53	0.747	NaN	0.837	0.382	0.004	0.439
DBSCAN	0.555	0.056	0.511	0.25	NaN	0.282	0.532	0.017	0.507
DPCKS	**0.990 5**	**0.927 9**	**0.981 2**	0.749	**0.840 2**	0.871 7	**0.681 8**	**0.030 6**	0.528 3
Average	0.703 6	0.286 7	0.615 3	0.770 9	0.282 2	0.691 8	0.535 4	0.012 8	0.507 5

3.4 图像数据集上的实验结果

本节在 Purity, NMI, RI 和 ARI 上测试 DPCKS 的聚类性能. 在 4 个图像数据集 Coil20[15], Olivetti faces[11], Yale[20]和 USPS[15]上, 将 DPCKS 与如下 5 种算法做对比: DPC-*K*NN[33], IDPC[34], F*K*NN-DPC (robust clustering by detecting density peaks and assigning points based on fuzzy weighted *K*NN)[21], DPC-DBFN(DPC based on density backbone and fuzzy neighborhood)[20]和 DPC-DBFN-E (DPC based on density backbone and fuzzy neighborhood hybridized with soft exponential kernel)[20]. DPCKS 在上述 4 个图像数据集上的最优参数 k 分别设置为 7, 4, 6 和 12, 其他算法的参数见文献[20]. 表 6 展示了 6 种算法在 4 个图像数据集上的聚类结果. 从表 6 可知: 大部分情况下, DPCKS 的聚类结果是最优的.

表 6 6 种算法在 4 个图像数据上的聚类结果

算法	Coil20				Olivetti faces			
	Purity	ARI	RI	NMI	Purity	ARI	RI	NMI
DPC-*K*NN	0.808 3	0.192 1	0.854 7	0.479 5	0.54	0.286 5	0.962 9	0.678 4
IDPC	0.622 9	0.172 1	0.847 5	0.437 9	0.422 5	0.140 9	0.942 9	0.558 6
F*K*NN-DPC	0.799 3	0.371 8	0.928 1	0.664 1	0.632 5	0.375 3	0.968 3	0.736 9
DPC-DBFN	**0.923 6**	0.237 4	0.868	0.573 2	0.742 5	0.313 8	0.964 7	0.697 1

表 6 6 种算法在 4 个图像数据上的聚类结果(续)

算法	Coil20				Olivetti faces			
	Purity	ARI	RI	NMI	Purity	ARI	RI	NMI
DPC-DBFN-E	0.783 3	0.536 1	0.947 2	0.761 6	0.617 5	0.415 3	0.971 1	0.758 8
DPCKS	0.878 5	**0.8**	**0.980 3**	**0.916 4**	**0.805**	**0.672 9**	**0.984 5**	**0.893**
Average	0.802 7	0.384 9	0.904 3	0.638 8	0.626 7	0.367 5	0.965 7	0.720 5
算法	Yale				USPS			
	Purity	ARI	RI	NMI	Purity	ARI	RI	NMI
DPC-*K*NN	0.454 5	0.141 9	0.893 2	0.422 9	0.695 7	0.425 3	0.875 3	0.506 6
IDPC	0.442 4	0.072 9	0.821 4	0.297 6	0.565 9	0.243 4	0.843 5	0.354 2
F*K*NN-DPC	0.630 3	0.181 7	0.884	0.438 8	0.809	0.428 6	0.887 2	0.568 2
DPC-DBFN	0.557 6	0.192 1	0.880 7	0.461 9	0.947 1	0.368 8	0.854 1	0.569 2
DPC-DBFN-E	**0.721 2**	0.254 1	0.905 1	0.514 3	**0.971 5**	0.418 9	0.841 7	0.588 7
DPCKS	0.575 8	**0.327 7**	**0.908 1**	**0.606 2**	0.821 4	**0.785 2**	**0.955 2**	**0.831 2**
Average	0.563 6	0.195 1	0.882 1	0.457	0.801 8	0.445	0.876 2	0.569 7

3.5 与DPC变体算法的对比结果

本节在 Acc, ARI, FM, NMI 和 RI 上评价 DPCKS 的聚类性能. 从第 3.2 节和第 3.3 节中获取 Jain, Spiral, Aggregation, R15, Iris, Seeds 和 Breast 这 7 个数据集的最优参数, 其余数据集的最优参数见表 7. 在这些常用聚类数据集上, 将 DPCKS 与如下聚类算法作对比: DPCRD (DPC based on relative density)[13], *K*-Means[5], DBSCAN[32], NCut (normalized cuts)[22], AP (affinity propagation)[35], DSet (dominant sets and pairwise clustering)[23], SPRG (robust affinity graphs for spectral clustering)[24], SNN-DPC (shared-nearest-neighbor-based DPC)[11], CFSFDP-HD (clustering by fast search and find of density peaks via heat diffusion)[25], ADPC-*K*NN (adaptive DPC based on *K*NN with aggregating strategy)[16], McDPC (multi-center DPC)[14], DPC[7], DPC-DLP (dynamic graph-based label propagation for DPC)[15], DPC-*K*NN (DPC based on *K*NN and principal component analysis)[33], IDPC (improved DPC)[34], FCM[31]和 DLORE-DP (dense members of local cores-based density peaks clustering)[36]. 表 8–表 11 分别呈现了对比算法的实验结果. 表 8–表 11 中的实验参数分别参照文献[13–15,36]. 表 8 中, "–" 表示由于计算机的内存限制不能获取聚类结果[13].

表 7 DPCKS 在 22 个数据集上的最优参数

数据集	*k*	数据集	*k*	数据集	*k*	数据集	*k*	数据集	*k*	数据集	*k*
WDBC	4	Vehicle	5	Pathbased	11	Dim512	3	Diabetes	9	Ionosphere	16
Mfeat	9	Coil20	7	Libras movement	5	Wine	14	Parkinsons	9	Waveform	24
Control	7	Flame	7	Dermatology	16	USPS	12	Sonar	7		
Thyroid	6	Ecoil	11	Wavefrom(noise)	25	Pima	9	Compound	10		

表 8 11 种算法在 20 个数据集上的聚类结果

算法	Jain		R15		Pathbased		Spiral		Dim512	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
K-Means	0.36	0.66	0.9	0.97	0.55	0.75	0	0.56	0.9	0.93
DBSCAN	0.85	0.97	0.88	0.91	0.71	0.85	**1**	**1**	0.7	0.86
NCut	0.22	0.52	0.99	0.99	0.88	0.95	0.23	0.64	0.99	0.99
AP	0.51	0.5	0.99	0.99	0.6	0.72	0.53	0.7	**1**	**1**
DSet	0.52	0.54	0.97	0.99	0.76	0.89	0.56	0.7	**1**	**1**
SPRG	0.41	0.74	0.96	0.99	0.24	0.65	0	0.56	**1**	**1**
ADPC-*K*NN	0.56	0.55	0.99	0.99	0.7	0.83	**1**	**1**	**1**	**1**
CFSFDP-HD	0.6	0.83	0.99	0.99	0.52	0.73	**1**	**1**	**1**	**1**
SNN-DPC	**1**	**1**	0.99	0.99	0.9	0.97	**1**	**1**	**1**	**1**
DPCRD	**1**	**1**	0.99	0.99	0.71	0.8	**1**	**1**	**1**	**1**
DPCKS	**1**	**1**	**0.9951**	**0.9967**	**0.9119**	**0.98**	**1**	**1**	**1**	**1**
Average	0.639 1	0.755 5	0.967 7	0.981 5	0.680 2	0.829 1	0.665 5	0.832 7	0.962 7	0.98
算法	Wine		Dim128		Waveform		Seeds		Wavefrom (noise)	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
K-Means	0.84	0.94	0.92	0.96	0.36	0.67	0.67	0.87	0.36	0.67
DBSCAN	0.51	0.72	**1**	**1**	0.09	0.42	0.56	0.78	0.07	0.4
NCut	0.46	0.77	0.99	0.99	0.01	0.56	0.64	0.83	0.01	0.56
AP	0.75	0.87	**1**	**1**	0.43	0.72	0.69	0.87	0.36	**0.71**

表 8 11 种算法在 20 个数据集上的聚类结果(续)

算法	Wine		Dim128		Waveform		Seeds		Wavefrom (noise)	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
DSet	0.77	0.91	**1**	**1**	0.38	0.7	0.6	0.79	0.34	0.69
SPRG	0.9	0.97	**1**	**1**	–	–	0.66	0.87	–	–
ADPC-*K*NN	0.73	0.88	**1**	**1**	0.38	0.69	0.69	0.88	0.2	0.61
CFSFDP-HD	0.6	0.75	**1**	**1**	–	–	0.56	0.76	–	–
SNN-DPC	0.88	0.95	**1**	**1**	0.4	**0.74**	**0.75**	0.91	0.38	0.69
DPCRD	0.74	0.88	**1**	**1**	0.38	0.7	0.74 2	0.9	0.33	0.66
DPCKS	**0.902**	**0.977 5**	**1**	**1**	**0.460 8**	0.599	0.724 2	**0.914 3**	**0.395 3**	0.527
Average	0.734 7	0.874 3	0.991 8	0.995 5	0.321 2	0.644 3	0.662 2	0.852 2	0.271 7	0.613
算法	WDBC		Glass		Breast		Aggregation		Dermatology	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
K-Means	0.62	0.87	0.33	0.67	0.74	0.92	0.85	0.92	0.8	0.88
DBSCAN	0.07	0.57	0.43	0.61	0.71	0.88	0.95	0.97	0.56	0.73
NCut	0.38	0.75	0.23	0.6	0.19	0.52	0.98	0.99	0.1	0.69
AP	0.67	0.89	0.44	**0.75**	0.72	0.92	0.87	0.93	0.83	0.92
DSet	0.48	0.72	0.44	**0.75**	0.7	0.89	0.92	0.98	0.77	0.91
SPRG	0.56	0.84	0.37	0.74	0.77	0.93	0.69	0.84	0.88	0.91
ADPC-*K*NN	0.54	0.82	0.36	0.53	0.64	0.87	**1**	**1**	0.84	0.91
CFSFDP-HD	0.05	0.52	0.23	0.67	0.51	0.81	0.87	0.9	0.57	0.78
SNN-DPC	**0.76**	0.93	0.4	0.6	**0.84**	0.95	0.96	0.99	0.92	0.95
DPCRD	0.75	0.92	0.46	0.63	0.78	0.94	**1**	**1**	**0.94**	**0.98**
DPCKS	0.725 1	**0.952 5**	**0.488 3**	0.514	0.817 7	**0.97**	0.995 8	0.998 7	0.935 2	0.961 7
Average	0.509 6	0.798 4	0.379 8	0.642 2	0.674 3	0.872 7	0.916 9	0.956 2	0.740 5	0.874 7
算法	Iris		Thyroid		Compound		Libras movement		Flame	
	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc	NMI	Acc
K-Means	0.72	0.86	0.6	0.82	0.7	0.83	0.58	0.91	0.43	0.73
DBSCAN	0.76	0.78	0.47	0.8	**0.93**	**0.98**	0.42	0.53	0.9	0.98
NCut	0.61	0.81	0.36	0.63	0.76	0.83	0.43	0.87	0.93	0.98
AP	0.78	0.89	0.51	0.59	0.81	0.89	0.64	0.93	0.58	0.71
DSet	0.76	0.78	0.6	0.82	0.82	0.91	**0.68**	**0.94**	0.6	0.72
SPRG	0.75	0.89	**0.81**	**0.94**	0.61	0.82	0.63	0.92	0.03	0.51
ADPC-*K*NN	0.78	0.87	0.46	0.77	0.87	0.91	**0.68**	0.92	**1**	**1**
CFSFDP-HD	0.78	0.87	0.02	0.5	0.76	0.83	0.5	0.85	**1**	**1**
SNN-DPC	0.91	0.97	0.68	0.89	0.85	0.94	0.66	0.91	0.9	0.98
DPCRD	**0.95**	**0.98**	0.45	0.74	0.87	0.92	0.65	0.89	**1**	**1**
DPCKS	0.914 4	0.973 3	0.698 2	0.930 2	0.899 4	0.914 8	0.656 1	0.455 6	**1**	**1**
Average	0.792 2	0.879 4	0.514 4	0.766 4	0.807 2	0.888 6	0.593 3	0.829 6	0.76	0.87

从表 8 可以看出: 在 Jain, R15, Pathbased, Spiral, Dim512, Wine 和 Dim128 这 7 个数据集上, DPCKS 在所有指标上是最优的; 对于 Waveform, Seeds, Wavefrom (noise), WDBC, Glass 和 Breast 这 6 个数据集, DPCKS 只在 1 个指标上取得最优值; 对于 Aggregation, Dermatology 和 Iris 这 3 个数据集, 在 Acc 和 NMI 上, DPCKS 略次于最优聚类算法, 但优于其他算法; 对于 Thyroid, Compound 和 Libras movement 这 3 个数据集, DPCKS 的 NMI 和 Acc 与 DPCRD 不相上下; 在 Flame 数据集上, 在 NMI 和 Acc 上, DPCKS 表现最优. 综合分析表 9–表 11 可知: 在大部分数据集上, DPCKS 均呈现较好的聚类性能.

表 9 4 种算法在 6 个数据集上的聚类结果

算法	Pima				Ecoil				Sonar			
	FM	ARI	RI	NMI	FM	ARI	RI	NMI	FM	ARI	RI	NMI
DBSCAN	**0.74**	0	0.55	0	0.56	0.41	0.81	0.51	**0.71**	0	0.5	0
DPC	**0.74**	0	0.55	0	0.79	0.68	0.86	0.64	0.66	0	0.5	0
McDPC	**0.74**	0	0.55	0	**0.83**	**0.75**	**0.9**	**0.71**	**0.71**	0	0.5	0
DPCKS	0.616 2	**0.175 4**	**0.589 5**	**0.080 9**	0.787 9	0.704 1	0.879 6	0.648 9	0.606 4	**0.175 1**	**0.587 5**	**0.14**
Average	0.709 1	0.043 9	0.559 9	0.020 2	0.742	0.636	0.862 4	0.627 2	0.671 6	0.043 8	0.521 9	0.035
算法	Iris				Seeds				Wine			
	FM	ARI	RI	NMI	FM	ARI	RI	NMI	FM	ARI	RI	NMI
DBSCAN	0.77	0.57	0.78	0.73	0.57	0	0.33	0	0.58	0	0.34	0
DPC	0.67	0.43	0.43	0.63	0.8	0.7	0.87	0.7	0.6	0.39	0.72	0.43
McDPC	0.92	0.89	0.95	0.87	0.8	0.7	0.87	0.7	0.6	0.39	0.72	0.43
DPCKS	**0.947 9**	**0.922 2**	**0.965 6**	**0.914 4**	**0.842 1**	**0.764 1**	**0.895 6**	**0.724 2**	**0.955 3**	**0.932 4**	**0.969 8**	**0.902**
Average	0.827	0.703 1	0.781 4	0.786 1	0.753	0.541	0.741 4	0.531 1	0.683 8	0.428 1	0.687 5	0.440 5

表 10　5 种算法在 10 个数据集上的聚类结果

算法	Coil20		Sonar		Mfeat		Vehicle		Parkinsons	
	RI	ARI	RI	ARI	RI	ARI	RI	ARI	RI	ARI
DPC-DLP	0.850 7	0.230 3	0.564 7	0.129 3	0.786 7	0.350 7	0.680 7	0.180 4	0.703 9	0.276 6
DPC-*K*NN	0.854 7	0.192 1	0.522 2	0.044 3	0.837 7	0.286	0.657 9	0.116	0.592 9	0.171 3
IDPC	0.847 5	0.172 1	0.534	0.068	0.750 7	0.251 7	0.657 3	0.122 6	0.648 7	0.224 8
FCM	0.646 5	0.090 8	0.503 2	0.006 4	0.887 7	0.418 1	0.653 2	0.076 2	0.626 9	0
DPCKS	**0.980 3**	**0.8**	**0.587 5**	**0.175 1**	**0.971 9**	**0.843 3**	**0.680 9**	**0.253 3**	**0.750 9**	**0.431 5**
Average	0.835 9	0.297 1	0.542 3	0.084 6	0.846 9	0.43	0.666	0.149 7	0.664 7	0.220 8
算法	WDBC		Thyroid		USPS		Ecoil		Diabetes	
	RI	ARI	RI	ARI	RI	ARI	RI	ARI	RI	ARI
DPC-DLP	0.827 9	0.652 6	0.701 1	0.38	0.835	0.428 7	0.877	**0.714 4**	**0.596 3**	**0.184 4**
DPC-*K*NN	0.650 7	0.300 3	0.606 4	0.207 7	0.873 7	0.425 3	0.868 1	0.692 5	0.583	0.165 9
IDPC	0.652 6	0.283	0.606 4	0.207 7	0.843 5	0.243 4	0.868 1	0.692 5	0.583	0.165 9
FCM	0.747 8	0.486 2	0.790 7	0.579	0.558 5	0.118 3	0.819 9	0.505 9	0.549 8	0.080 4
DPCKS	**0.909 4**	**0.817 8**	**0.884 2**	**0.765 5**	**0.955 2**	**0.785 2**	**0.879 6**	0.704 1	0.584 7	0.162 5
Average	0.757 7	0.508	0.717 8	0.428	0.808 1	0.383	0.862 5	0.661 9	0.579 4	0.151 8

表 11　4 种算法在 6 个数据集上的聚类结果

算法	Iris		Wine		Seeds	
	ACC	NMI	ACC	NMI	ACC	NMI
DPC	0.91	0.81	**0.98**	**0.91**	0.89	0.7
SNN-DPC	0.95	0.84	0.71	0.63	0.78	0.63
DLORE-DP	0.91	0.81	0.97	0.88	0.84	0.63
DPCKS	**0.973 3**	**0.914 4**	0.977 5	0.902	**0.914 3**	**0.724 2**
Average	0.935 8	0.843 6	0.909 4	0.830 5	0.856 1	0.671 1
算法	Ionosphere		WDBC		Control	
	ACC	NMI	ACC	NMI	ACC	NMI
DPC	0.68	0.09	0.8	0.35	0.56	0.75
SNN-DPC	0.56	0.07	0.94	0.68	0.44	0.67
DLORE-DP	0.71	0.28	0.86	0.48	0.58	0.81
DPCKS	**0.857 5**	**0.415 7**	**0.952 5**	**0.725 1**	**0.933 3**	**0.965 4**
Average	0.701 9	0.213 9	0.888 1	0.558 8	0.628 3	0.798 9

3.6 消融实验

为了进一步验证本文所提两个创新点的有效性, 以 DPC+CS 表示 DPC 与本文簇中心选取策略融合的聚类算法, 以 DPC+AS 表示 DPC 与本文点分配策略融合的聚类算法. 实验对比结果见表 12: 对于 WDBC, Ionosphere, Compound, Pathbased, Wine, Breast, Libras movement 和 Ecoil 这 8 个数据集, DPC 参数分别设置为 0.49, 0.22, 3.73, 3.63, 1.99, 0.01, 0.23 和 0.19. DPCKS 参数如第 3.3 节和表 7 所示. 从表 12 可以看出: 对于 WDBC, Libras movement 和 Ecoil 这 3 个数据集, 在 6 个评价指标上, DPC+CS 的聚类效果均优于 DPC; 在 Compound 数据集上, 对于 Acc, ARI 和 FM, DPC+CS 的聚类效果优于 DPC; 对于 Ionosphere, Pathbased 和 Wine 这 3 个数据集, 在 6 个评价指标上, DPC+CS 的聚类效果与 DPC 一样; 对于 Breast 数据集, DPC 的聚类效果优于 DPC+CS; 针对 Breast 数据集, DPC+CS 为每一个簇选择一个簇中心, 而 DPC 没有, 但 DPC+CS 的聚类性能次于 DPC, 这与 Breast 数据集空间分布结构有关. 综上所述, DPC+CS 的聚类性能优于 DPC, 这证明了本文簇中心选择方法的有效性. 对于 Ionosphere, Compound, Pathbased, Wine, Breast, Libras movement 和 Ecoil 这 7 个数据集, 在所有指标值上, DPC+AS 均高于 DPC; 对于 WDBC 数据集, 在大多数评价指标上, DPC+AS 优于 DPC. 综上所述, DPC+AS 的聚类效果优于 DPC, 这证明本文点分配策略是有效的. 从表 12 可知: 在 WDBC, Ionosphere, Compound, Pathbased, Wine 和 Breast 这 6 个数据集上, DPCKS 的 6 个指标值均最优; 在 Libras movement 数据集上, DPCKS 除了 Acc 之外的 5 个指标值均是最优的; 在 Ecoil 数据集上, 除了 NMI 之外的 5 个指标, DPCKS 均表现最优. 总体来讲, DPCKS 优于 DPC+CS, DPC+AS 和 DPC, 该消融实验充分验证了本文算法的有效性.

表 12 DPCKS 在 8 个数据集上的消融实验结果

数据集	算法	Acc	ARI	FM	NMI	Purity	RI
WDBC	DPC	0.623 9	−0.002 8	0.691 6	0.013 2	0.627 4	0.529 9
	DPC+CS	0.847 1	0.470 5	0.781 6	0.465 2	0.847 1	0.740 5
	DPC+AS	0.660 8	0.095 9	0.585 6	0.175 1	0.660 8	0.550 9
	DPCKS	**0.952 5**	**0.817 8**	**0.916 2**	**0.725 1**	**0.952 5**	**0.909 4**
Ionosphere	DPC	0.746 4	0.235 7	0.649 1	0.152 9	0.746 4	0.620 4
	DPC+CS	0.746 4	0.235 7	0.649 1	0.152 9	0.746 4	0.620 4
	DPC+AS	**0.857 5**	**0.500 5**	**0.789 1**	**0.415 7**	**0.857 5**	**0.755**
	DPCKS	**0.857 5**	**0.500 5**	**0.789 1**	**0.415 7**	**0.857 5**	**0.755**
Compound	DPC	0.696 7	0.591	0.686 6	0.797 5	0.844 6	0.853 1
	DPC+CS	0.706 8	0.594 6	0.692 2	0.789 8	0.842 1	0.851 7
	DPC+AS	0.869 7	0.797 2	0.853 2	0.846 4	0.869 7	0.917 8
	DPCKS	**0.914 8**	**0.880 3**	**0.911 5**	**0.899 4**	**0.914 8**	**0.953 6**
Pathbased	DPC	0.753 3	0.471 7	0.663 6	0.555 2	0.753 3	0.753 6
	DPC+CS	0.753 3	0.471 7	0.663 6	0.555 2	0.753 3	0.753 6
	DPC+AS	**0.98**	**0.94**	**0.96**	**0.911 9**	**0.98**	**0.973 4**
	DPCKS	**0.98**	**0.94**	**0.96**	**0.911 9**	**0.98**	**0.973 4**
Wine	DPC	0.882	0.672 4	0.783 4	0.710 4	0.882	0.853 2
	DPC+CS	0.882	0.672 4	0.783 4	0.710 4	0.882	0.853 2
	DPC+AS	0.971 9	0.915	0.943 6	0.882 8	0.971 9	0.962
	DPCKS	**0.977 5**	**0.932 4**	**0.955 3**	**0.902**	**0.977 5**	**0.969 8**
Breast	DPC	0.759 7	0.225 2	0.727 6	0.228 5	0.759 7	0.634 3
	DPC+CS	0.681	0.049 4	0.708 3	0.084 9	0.681	0.564 9
	DPC+AS	0.957 1	0.833 8	0.925 2	0.728 9	0.957 1	0.917 7
	DPCKS	**0.97**	**0.882 6**	**0.945 9**	**0.817 7**	**0.97**	**0.941 6**
Libras movement	DPC	0.441 7	0.342 5	0.393 9	0.625 6	0.505 6	0.900 5
	DPC+CS	**0.469 4**	0.365 9	0.413 3	0.636 8	0.505 6	0.909 7
	DPC+AS	0.452 8	0.357 8	0.402 8	0.648	0.527 8	0.915 5
	DPCKS	0.455 6	**0.375 1**	**0.418 6**	**0.656 1**	**0.530 6**	**0.918 4**
Ecoil	DPC	0.544 6	0.436 5	0.555 9	0.589 3	0.779 8	0.802 6
	DPC+CS	0.592 3	0.460 8	0.584 3	0.600 6	0.806 5	0.805
	DPC+AS	0.761 9	0.680 7	0.776 5	**0.655 9**	0.761 9	0.864 2
	DPCKS	**0.809 5**	**0.704 1**	**0.787 9**	0.648 9	**0.809 5**	**0.879 6**

3.7 讨 论

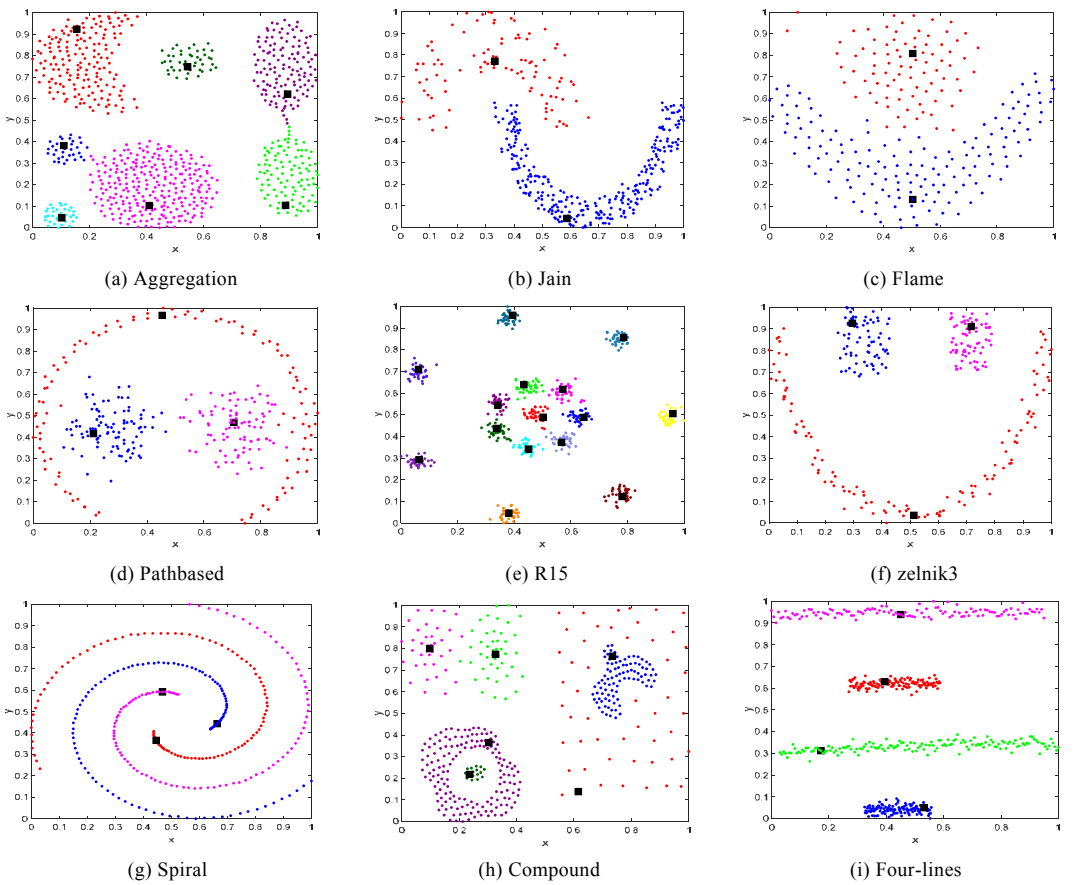
下面对本文的两个创新点(一是改进 DPC 的簇中心选取策略, 二是优化 DPC 的分配策略)进行讨论.

(1) 为了进一步讨论本文改进 DPC 簇中心选取策略的有效性, 在簇中心选择的结果上, 对 DPCKS 和 DPC 做对比分析. 这里, 仅考虑算法是否能够找到正确的簇中心. 下面给出 2 种算法选择簇中心的必要说明: 在 DPCKS 中, 设置 $3\leqslant k\leqslant 25$, 以 1 为步长寻找簇中心. 在 DPC 中, 依据文献[7]和第 3.2 节推荐 t_5 的值, 设置 $0\leqslant t_5\leqslant 5$, 以 0.01 为步长寻找簇中心, 采用高斯核选取局部密度与高密度最近邻距离乘积较大的前 an 个点作为簇中心, 其中, an 表示数据集中簇的数量. 表 13 呈现了 DPCKS 和 DPC 簇中心选择的对比情况. 其中: 若算法能够在数据集上找到正确的簇中心, 则设置为 1; 否则无值. 从表 13 可知: DPCKS 为 26 个数据集选择了正确的簇中心, DPC 仅为 18 个数据集选择了正确的簇中心. 由此说明, DPCKS 选择簇中心的策略是有效的.

为了更直观地呈现簇中心的位置, 图 7 展示了 DPCKS 在 9 个 2 维数据集上选取的簇中心. 其中, 不同颜色表示不同的簇, 黑色正方形表示簇中心. 从图 7 中可以看出: DPCKS 为所有数据集中的所有簇都找到了簇中心. 针对一些密度分布均匀的簇, DPCKS 选择的簇中心可能不在最优的簇中心位置. 下面以 Four-lines 数据集为例, 说明这种特殊情况. 图 8 呈现了 DPCKS 选取簇中心的过程, 其中一个数字对应一个候选簇中心, 以数字代表相应的候选簇中心. 图 8(a)展示了当 k=10 时, DPCKS 选择所有的候选簇中心, 即图中数字对应的标记点. 依据第 2.2 节, 后续点数量少于 k+1 的候选簇中心不能作为中心, 于是从图 8(a)中删除 192, 207, 329, 371 和 385 这 5 个点, 得到图 8(b); 在 pcr 中, 若多个候选簇中心属于同一簇, 则保留 cr 值最大的候选簇中心, 据此再次删除部分点, 如图 8(c)所示. 图 8(c)中 56, 166, 263 和 499 这 4 个点的 cr 值都比 148, 228, 403 和 432 这 4 个点的 cr 值大, 因而 56, 166, 263 和 499 这 4 个点被选作簇中心. 图 8(c)中 263 不在最优的位置, 而图 8(b)中 315 在最优的位置. 依据公式(12)可知, $cr(263)>cr(315)$, 所以删除了处于最优位置的 315.

表 13 DPCKS 和 DPC 选择簇中心的结果

序号	数据集	DPCKS	k	DPC	t_5	序号	数据集	DPCKS	k	DPC	t_5
1	Iris	1	11	1	2.8	20	Pathbased	1	11	1	0.01
2	Jain	1	10	–	–	21	Parkinsons	1	9	–	–
3	Flame	1	7	1	0.01	22	Wifilocalization	1	22	1	0.21
4	WDBC	1	4	–	–	23	Vehicle	1	14	–	–
5	R15	1	22	1	0.02	24	Pima	1	7	–	–
6	zelnik3	1	11	1	0.01	25	Waveform(noise)	1	7	1	0.01
7	Spiral	1	6	1	0.01	26	Diabetes	1	5	–	–
8	Seeds	1	14	1	0.06	27	Libras movement	–	–	–	–
9	Wine	1	14	1	1.99	28	Olivetti faces	–	–	–	–
10	Sonar	1	7	1	0.3	29	USPS	–	–	–	–
11	Breast	1	10	–	–	30	Waveform	–	–	1	0.01
12	Dermatology	1	16	–	–	31	Wholesale	–	–	1	0.01
13	Dim512	1	3	1	0.01	32	Mfeat	–	–	–	–
14	Ionosphere	1	16	1	0.01	33	Glass	–	–	–	–
15	Aggregation	1	14	1	0.11	34	Ecoil	–	–	–	–
16	Banknotes	1	18	1	0.08	35	Dim128	–	–	1	0.08
17	Four-lines	1	10	–	–	36	Coil20	–	–	–	–
18	Thyroid	1	6	–	–	37	Control	–	–	–	–
19	Compound	1	10	–	–	38	Yale	–	–	–	–

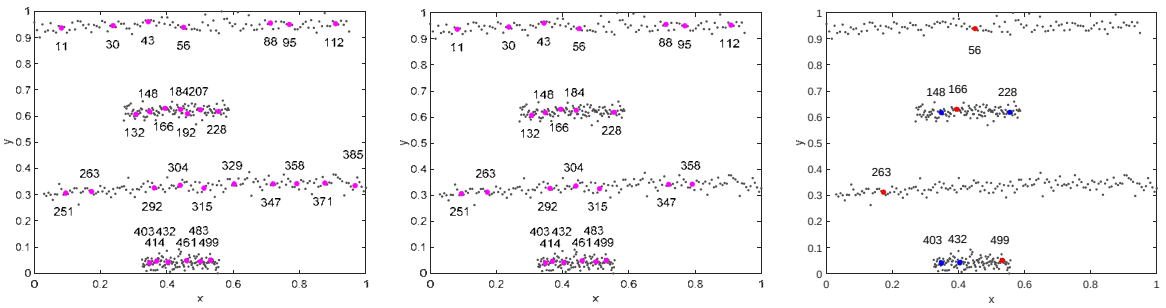


(a) Aggregation (b) Jain (c) Flame

(d) Pathbased (e) R15 (f) zelnik3

(g) Spiral (h) Compound (i) Four-lines

图 7 DPCKS 算法在 2 维数据集上选取的簇中心



(a) 所有的候选簇中心 (b) 依据后续点删除部分候选簇中心 (c) 由 *pcr* 删除部分候选簇中心

图 8 DPCKS 算法在 Four-lines 数据集上选取簇中心的过程

(2) 若聚类算法仅选择正确的簇中心, 缺乏有效的点分配方法, 其性能仍是不佳的. 对于 2 维数据集, 大部分情况下, 点和其 *K*NN 在一个簇中; 但在高维数据集上, 基于欧式距离获取的 *K*NN 无较高的可靠性. 因此, 非常有必要优化 DPC 的分配策略, 辨别一个点究竟和哪些点在同一个簇中. 由此提出相似度、积极域和积极值等概念, 确定待分配点, 并将其分配至其积极域中占主导地位的簇中, 尽量保证每个点都被分配到和其相似度最高的点所在的簇. 在表 4–表 11 中的大部分数据集上, DPCKS 的聚类效果都是最优的, 这说明点的分配策略是有效的.

4 总 结

本文对 DPC 的两个普遍问题进行改进, 提出了一种基于 K 近邻和优化分配策略的密度峰值聚类算法. 该算法包括确定簇中心、计算相似度和点的分配这 3 个主要过程: 首先排除大量非簇中心点, 提高时空效率, 构造密度因子和距离因子计算候选簇中心作为簇中心的信任度, 确定簇中心; 然后, 基于共享近邻、高密度最近邻、密度差值和 *K*NN 之间的距离构建相似度, 提出邻域、相似集和相似域等概念; 最后, 基于相似域和边界点获取初始聚类结果, 结合簇中心确定中间聚类结果, 根据中间聚类结果和积极域确定最终聚类结果. 在 38 个数据集上测试本文算法有效性, 实验结果表明, DPCKS 算法在 Purity, Acc, ARI, NMI, RI 和 FM 等评价指标上均表现出良好的聚类性能. 此外, 在未来研究工作中, 结合模糊集理论改进 DPCKS 算法, 进一步提升聚类算法的时空效率.

References:

[1] Chang B, Chen EH, Zhu FD, *et al*. Maximum a posteriori estimation for information source detection. IEEE Trans. on Systems, Man, and Cybernetics: Systems, 2020, 50(6): 2242−2256. [doi: 10.1109/TSMC.2018.2811410]

[2] Kang Z, Lin ZP, Zhu XF, *et al*. Structured graph learning for scalable subspace clustering: From single view to multiview. IEEE Trans. on Cybernetics, 2021. [doi: 10.1109/TCYB.2021.3061660]

[3] Sun L, Qin XY, Ding WP, Xu JC. Nearest neighbors-based adaptive density peaks clustering with optimized allocation strategy. Neurocomputing, 2022, 473: 159−181. [doi: 10.1016/j.neucom.2021.12.019]

[4] Sun L, Qin XY, Ding WP, *et al*. Density peaks clustering based on k-nearest neighbors and self-recommendation. Int'l Journal of Machine Learning and Cybernetics, 2021. [doi: 10.1007/s13042-021-01284-x]

[5] MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proc. of the Berkeley Symp. on Mathematical Statistics & Probability. 1965. 281−297.

[6] Zhang XC. Data Clustering. Beijing: Science Press, 2017. 68−95 (in Chinese).

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

[8] Sun L, Liu RN, Xu JC, *et al*. An adaptive density peaks clustering method with Fisher linear discriminant. IEEE Access, 2019, 7: 72936−72955. [doi: 10.1109/ACCESS.2019.2918952]

[9] Chen YW, Shen LL, Zhong CM, *et al*. Survey on density peak clustering algorithm. Journal of Computer Research and Development, 2020, 57(2): 378−394 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2020.20190104]

[10] Gu ZW, Li P, Lang X, *et al*. A controllable clustering model of the electrical load curve based on variational mode decomposition and fast search of the density peak. Power System Protection and Control, 2021, 49(8): 118−127 (in Chinese with English abstract). [doi: 10.19783/j.cnki.pspc.200713]

[11] Liu R, Wang H, Yu XM. Shared-nearest-neighbor-based clustering by fast search and find of density peaks. Information Sciences, 2018, 450: 200−226. [doi: 10.1016/j.ins.2018.03.031]

[12] Jiang D, Zang WK, Sun R, *et al*. Adaptive density peaks clustering based on *K*-nearest neighbor and Gini coefficient. IEEE Access, 2020, 8: 113900−113917. [doi: 10.1109/ACCESS.2020.3003057]

[13] Hou J, Zhang AH, Qi NM. Density peak clustering based on relative density relationship. Pattern Recognition, 2020, 108: 107554. [doi: 10.1016/j.patcog.2020.107554]

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

[15] Seyedi SA, Lot A, Moradi P, *et al*. Dynamic graph-based label propagation for density peaks clustering. Expert Systems with Applications, 2019, 115: 314−328. [doi: 10.1016/j.eswa.2018.07.075]

[16] Liu YH, Ma ZM, Yu F. Adaptive density peak clustering based on *k*-nearest neighbors with aggregating strategy. Knowledge-based Systems, 2017, 133: 208−220. [doi: 10.1016/j.knosys.2017.07.010]

[17] Ding SF, Xu X, Wang YR. Optimized density peaks clustering algorithm based on dissimilarity measure. Ruan Jian Xue Bao/ Journal of Software, 2020, 31(11): 3321−3333 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5813.htm [doi: 10.13328/j.cnki.jos.005813]

[18] Wang GT, Song QB. Automatic clustering via outward statistical testing on density metrics. IEEE Trans. on Knowledge & Data Engineering, 2016, 28(8): 1971−1985. [doi: 10.1109/TKDE.2016.2535209]

[19] Dijkstra EW. A note on two problems in connexion with graphs. Numerische Mathematik, 1959, 1(1): 269−271. [doi: 10.1016/ j.knosys.2019.06.032]

[20] 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]

[21] Xie JY, Gao HC, Xie WX, *et al*. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted *K*-nearest neighbors. Information Sciences, 2016, 354: 19−40. [doi: 10.1016/j.ins.2016.03.011]

[22] Shi JB, Malik J. Normalized cuts and image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2000, 22(8): 167−172.

[23] Pavan M, Pelillo M. Dominant sets and pairwise clustering. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2006, 29(1): 167−172. [doi: 10.1109/TPAMI.2007.250608]

[24] Zhu XT, Loy CC, Gong SG. Constructing robust affinity graphs for spectral clustering. In: Proc. of the IEEE Int'l Conf. on Computer Vision and Pattern Recognition. 2014. 1450−1457. [doi: 10.1109/CVPR.2014.188]

[25] Mehmood R, Zhang GZ, Bie RF, *et al*. Clustering by fast search and find of density peaks via heat diffusion. Neurocomputing, 2016, 208: 210−217. [doi: 10.1016/j.neucom.2016.01.102]

[26] Hou J, Cui HX. Experimental evaluation of a density kernel in clustering. In: Proc. of the Int'l Conf. on Intelligent Control and Information Processing. 2016. 5559. [doi: 10.1109/ICICIP.2016.7885876]

[27] Powers DMW. Evaluation: From precision, recall and *F*-measure to ROC, informedness, markedness and correlation. arXiv: 2010.16061, 2020.

[28] Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. Journal of Machine Learning Research, 2010, 11(95): 2837−2854.

[29] He ZY, Xu XF, Deng SC. *K*-ANMI: A mutual information based clustering algorithm for categorical data. Information Fusion, 2008, 9(2): 223−233.

[30] Hotelling H. Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology, 1933, 24(6): 417−441. [doi: 10.1037/h0071325]

[31] Bezdek JC, Ehrlich R, Full W. FCM: The fuzzy c-means clustering algorithm. Computers and Geosciences, 1984, 10: 191–203. [doi: 10.1016/0098-3004(84)90020-7]

[32] Ester M, Kriegel HP, Sander J, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining, Vol.1. 1996. 226–231.

[33] Du MJ, Ding SF, Jia HJ. Study on density peaks clustering based on *k*-nearest neighbors and principal component analysis. Knowledge-based Systems, 2016, 99: 135–145. [doi: 10.1016/j.knosys.2016.02.001]

[34] Lot A, Seyedi SA, Moradi P. An improved density peaks method for data clustering. In: Proc. of the IEEE 6th Int'l Conf. on Computer and Knowledge Engineering. 2016. 263–268.

[35] Frey BJ, Dueck D. Clustering by passing messages between data points. Science, 2007, 315(5814): 972–976. [doi: 10.1126/science.1136800]

[36] Cheng DD, Zhang SL, Huang JL. Dense members of local cores-based density peaks clustering algorithm. Knowledge-Based Systems, 2020, 193: 105454. [doi: 10.1016/j.knosys.2019.105454]

附中文参考文献:

[6] 张宪超. 数据聚类. 北京: 科学出版社, 2017. 68–95.

[9] 陈叶旺, 申莲莲, 钟才明, 王田, 陈谊, 杜吉祥. 密度峰值聚类算法综述. 计算机研究与发展, 2020, 57(2): 378–394. [doi: 10.7544/issn1000-1239.2020.20190104]

[10] 谷紫文, 李鹏, 郎恂, 喻怡轩, 沈鑫, 曹敏. 基于变分模态分解和密度峰值快速搜索的电力负荷曲线可控聚类模型. 电力系统保护与控制, 2021, 49(8): 118–127. [doi: 10.19783/j.cnki.pspc.200713]

[17] 丁世飞, 徐晓, 王艳茹. 基于不相似性度量优化的密度峰值聚类算法. 软件学报, 2020, 31(11): 3321–3333. http://www.jos.org.cn/1000-9825/5813.htm [doi: 10.13328/j.cnki.jos.005813]

孙林(1979－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为粒计算, 数据挖掘,机器学习, 生物信息学.

徐久成(1964－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为粒计算, 数据挖掘, 机器学习.

秦小营(1995－), 女, 硕士生, 主要研究领域为数据挖掘, 机器学习.

薛占熬(1963－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为人工智能基础理论, 数据挖掘.