控制与决策
Control and Decision

基于低密度分数的密度峰值聚类算法

陈梅, 尤远毓秀, 魏礼磊, 唐晟洲

引用本文:
陈梅, 尤远毓秀, 魏礼磊, 等. 基于低密度分数的密度峰值聚类算法[J]. 控制与决策, 2025, 40(5): 1599-1609.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2024.0963

基于低密度分数的密度峰值聚类算法

陈 梅†，尤远毓秀，魏礼磊，唐晟洲

(兰州交通大学 电子与信息工程学院，兰州 730000)

摘 要: 密度峰值聚类算法 (DPC) 可识别出任意形状的簇, 但是对于存在多密度峰值的簇, DPC 可能会识别出多个簇中心点, 导致簇划分错误. 鉴于此, 提出一种基于低密度分数的密度峰值聚类算法 (LS-DPC). 该算法首先使用低密度分数放大数据点的密度差异, 缩小整体密度差异大的相邻区域的密度差异, 使得单个簇内所有区域的密度分布均重构为单峰密度分布; 然后, 根据低密度分数自动获得子簇中心点; 接着, 得到子簇后, 根据密度相交条件对子簇进行融合, 完成聚类; 最后, 将所提出 LS-DPC 算法与k-Means、SC、DPC、DN、Extreme 以及 ICKDP 算法进行对比, 实验结果表明所提出算法在复杂数据集和 UCI 数据集上的表现优于对比算法.

关键词: 聚类；密度峰值； 多密度峰值簇；低密度分数； 子簇融合

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2024.0963

引用格式: 陈梅, 尤远毓秀, 魏礼磊, 等. 基于低密度分数的密度峰值聚类算法 [J]. 控制与决策, 2025, 40(5): 1599-1609.

A density peaks clustering algorithm based on low density score

CHEN Mei†, YOU Yuan-yu-xiu, WEI Li-lei, TANG Sheng-zhou

(School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730000, China)

Abstract: The density peaks clustering algorithm(DPC) can identify clusters with arbitrary shapes, but for clusters with multiple density peaks, the DPC may identify multiple cluster centers, leading to wrong cluster partitioning. Therefore, a density peaks clustering algorithm based on low density score(LS-DPC) is proposed. The algorithm firstly uses low density score to enlarge the density difference of data points and reduce the density difference of adjacent regions with large overall density difference, so that the density distributions of all regions in a single cluster are reconstructed into a single-peak density distribution, and then automatically obtains the center centers of sub-clusters according to the low density score. After the sub-clusters are obtained, the sub-clusters are merged according to the density intersection condition to complete the clustering. The proposed LS-DPC algorithm is compared with k-Means, SC, DPC, DN, Extreme and ICKDP. Experimental results show that the proposed algorithm outperforms the comparison algorithms on complex datasets and UCI datasets.

Keywords: clustering; density peaks; clusters with multiple density peaks; low density score; sub-clusters merging

0 引 言

聚类是一种无监督学习方法, 它可将数据分为同簇内数据点相似度较高、不同簇间数据点相似度较低的多个簇, 从而对数据中的信息进行挖掘和分析. 在真实世界中, 聚类可在没有先验知识的情况下对未知数据集进行挖掘, 有效地发现数据集上潜在的、有价值的内在结构, 因此, 聚类被广泛应用于数据分析[1]、文字提取[2-3]、行为检测[4] 和大气监督[5] 等领域.

近年来, 许多不同类型的聚类算法[6-11] 被提出. 基于划分的聚类算法k-Means[6], 它将点划分至距离最近的簇中心点, 并根据簇的质心更新簇中心点, 对球状簇的聚类效果较好; 基于图的聚类算法谱聚类 (SC)[7] 使用相似图的特征向量来降低数据维数, 在识别非线性可分数据中的簇特别有效, 但是对于大型数据集时间成本相对较高; 基于层次的聚类算法, 如 CHAMELEON[8] 和 BIRCH[9] 等, 将聚类结果转化为自上而下或自下而上划分的树形结构来划分或融合聚类结果; 基于密度的聚类算法[10-11] 在识别具有任意形状的聚类方面比其他类型的算法更有优势,

收稿日期: 2024-08-13；**录用日期:** 2024-10-24.

基金项目: 国家自然科学基金项目 (62266029)；甘肃省重点研发计划项目 (24YFGA036)；甘肃省高等学校产业支撑计划项目 (2022CYZC-36).

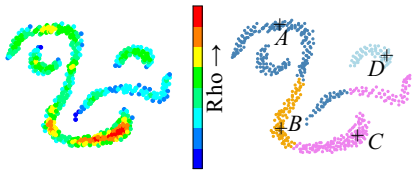
责任编委: 李少远.

†通信作者. E-mail: mei.chen.lzjtu@hotmail.com.

因此得到了广泛研究. 密度峰值聚类算法 (DPC) 是由 Rodriguez 等[12] 于 2014 年提出的优秀密度聚类算法, 因提出密度峰值思想而闻名. 该算法提出了簇中心点的密度比簇边缘区域点的密度高以及簇中心点距其他高密度点远这两个假设, 并据此来选择具有高密度和高相对距离 (点距其最近高密度点的距离) 的点作为簇中心点. 但是它选择簇中心点的方法可能会在一个含有多个密度峰值的簇中识别出多个簇中心点, 这也导致了 DPC 不能正确识别含多密度峰值的簇.

许多算法尝试通过修改密度核[13-19] 和相对距离[20-23] 的计算方法来提升 DPC 选择簇中心点的准确性, 进而提高复杂簇识别的准确率. 密度核的计算通常与数据点的近邻信息相结合, 使得数据点的密度保留邻域特点[13-17]. Liu 等[13] 提出了一种基于共享 k近邻的密度核, 陈蔚昌等[14] 提出了结合逆近邻和 k近邻的局部密度, 周玉等[15] 将k互近邻与局部核密度加权得到新的局部密度, 但是三者计算成本均较高. 吕莉等[16] 提出了结合自然近邻的密度核, 吴润秀等[17] 结合k近邻和相对密度定义了相对k近邻的局部密度, 二者均降低了簇疏密程度对簇中心的影响. 此外, Lotfi 等[18] 引入了基于模糊核的分数来识别聚类中心, Zhang 等[19] 则定义了一种能够很好地反映连通性的密度核来识别聚类中心, 但是二者需要提前知道簇的个数. 对于相对距离的改进, DPC-CE 算法[20]、MDPC算法[21] 和 ICKDP 算法[22] 均考虑了连通性来计算相对距离: DPC-CE 引入了一种基于图的连通性估计策略, 该策略同时考虑了连通性信息和空间距离; MDPC 引入了一种新的具有指数项和比例因子的流形距离; ICKDP 则提出了考虑空间位置分布信息的改进连通核距离 (ICK) 来计算相对距离. 除连通性, 还有算法考虑了密度关系来计算相对距离. Li 等[23] 将点的相对距离定义为距其最近高密度点的距离与距其最近低密度点的距离的差. 上述基于密度峰值的算法在一定程度上提高了算法对复杂簇中心点的识别能力, 但是对于包含多密度峰值的簇, 上述算法大多不能正确地识别.

包含多密度峰值的簇存在以下特点: 簇内的密度分布呈现为多个高密度区域与低密度区域相邻, 且密度峰值仅出现在这些高密度区域中. 对于包含多密度峰值簇的数据集, 基于密度峰值的聚类算法可能会在一个簇内识别多个簇中心点. 图 1 为 DB 数据集的高斯密度分布和经 DPC 聚类的结果. 如图 1 所示: 在数据集 DB 中, 反“S”型簇内存在 3 个密度相对高的区域 (呈现黄色和红色), 但是这 3 个高密度区域又被呈现绿色的低密度区域隔开. 因此,



(a) DB 数据集的密度高斯分布 (b) DB数据集经DPC聚类后的结果

图1 DB 数据集的高斯密度分布和经 DPC 聚类的结果

使用 DPC 识别 DB 数据集时, DPC 错误地在该反“S”型簇中识别了 3 个簇中心点.

为了正确地识别包含多密度峰值的簇, 本文提出基于低密度分数的密度峰值聚类算法 (LS-DPC). LS-DPC 提出低密度分数来放大数据点间的密度差异, 缩小相邻区域的密度差异, 使得包含多密度峰值簇内每个区域的密度分布均重构为单峰密度分布. 同时, 根据低密度分数自动获得子簇的簇中心点. 在子簇划分完成后, 根据密度相交条件对子簇进行融合, 从而完成聚类.

1 密度峰值聚类算法

DPC 算法根据数据点的密度分布情况提出了两个关键的假设.

假设 1 簇中心点的密度比簇边缘区域点的密度高.

假设 2 簇中心点距离其他高密度点远.

基于这两个假设, DPC 提出了关于密度和相对距离的定义.

定义 1 (密度) 基于假设 1, 数据点x_i的密度ρ_i由密度核计算, 如截断核 (1) 和高斯核 (2) 分别为

$$\rho_i = \sum_{x_j \in D} \chi(d(x_i, x_j) - d_c), \tag{1}$$

$$\rho_i = \sum_{x_j \in D} \exp\left(-\frac{d(x_i, x_j)^2}{d_c^2}\right). \tag{2}$$

其中: 若$x<0$, 则$\chi(x)=1$,否则$\chi(x)=0$; $d(x_i, x_j)$为点x_i与点x_j的距离; d_c为截断距离.

定义 2 (相对距离) 基于假设 2, DPC 定义了相对距离δ_i来表示点x_i到它最近高密度点x_j的距离, 即

$$\delta_i = \min_{x_j:\rho_j>\rho_i} \{d(x_i, x_j)\}. \tag{3}$$

根据这两个定义, DPC 的主要流程如下: 1) 计算每个点x_i的密度ρ_i. 2) 计算每个点x_i的相对距离δ_i, 并找到每个点x_i对应的最近高密度点x_j作为该点的邻居. 3) 根据决策图选择拥有高密度和高相对距离的点作为簇中心点 (或找到前k个有最大γ值的点. 其中: $\gamma = \rho \times \delta$, k为数据集簇的个数). 4) 为选出的簇中心点分配标签. 5) 从最高密度点开始, 将点x_i的

最近高密度点x_j的标签作为该点的标签.

为了便于后文表示, 本文使用的符号如表 1 所示.

表1　符号定义

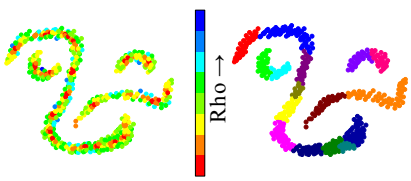
符号	定义
n	数据集中点的数量
D	数据集
x_i	数据集中第i个数据点
ρ	数据点的密度
ρ_i	第i个数据点的密度, $1 \leqslant i \leqslant n$
δ	数据点距其最近高密度点的距离
δ_i	第i个数据点距其最近高密度点的距离
$d(x_i, x_j)$	第i个数据点到第j个数据点的距离
$k\mathrm{NN}_{x_i}$	距第i个数据点最近的k个数据点的集合
score	低密度分数
score_i	第i个数据点的低密度分数
C	子簇集合
C_m	第m个子簇, $1 \leqslant m \leqslant n$
c	子簇中心点
c_m	第m个子簇的子簇中心点
$\mathrm{average}_{d_m}$	簇m的平均距离
Border_m^l	簇m对簇l的边缘点集合
Score_m^l	簇m对簇l的平均分数

2　LS-DPC 算法

2.1　算法核心思想

包含多密度峰值簇的密度分布会呈现出部分区域密度高、部分区域密度低的差异性, 如图 1(a) 中反“S”型上方相邻的呈现黄色密度分布的高密度区域和呈现绿色密度分布的低密度区域. 考虑到该特点, 所提出算法首先使用低密度分数重构密度划分子簇, 然后使用密度相交对子簇进行融合.

所提出算法首先使用与密度成反比的低密度分数放大数据点间的密度差异, 缩小相邻低密度区域与高密度区域的密度差异, 使得单个簇内每个区域的密度分布重构为反映数据点密度大小关系的单峰密度分布. 图 2 为 DB 数据集的低密度分数分布和子簇划分结果. 如图 2(a) 所示: 上述图 1(a) 中两个区域的密度分布皆重构为低密度点 (呈现黄色、绿色) 包围高密度点 (呈现红色、橙色) 的单峰密度分布, 且簇内每个区域的密度分布均呈现为单峰密度分布. 此外, 根据低密度分数的特性, 所提出算法可自动获得这些单峰的密度峰值点作为子簇中心点, 得到子簇如图 2(b) 所示.



(a) DB 数据集的低密度分数分布　(b) 根据低密度分数得到的子簇

图2　DB 数据集的低密度分数分布和子簇划分结果

获得子簇后, 对子簇进行两两比较, 若两个相邻子簇边缘处低密度分数分布如图 3 所示: 簇 1 存在点x_2, 且x_2的低密度分数大于簇 2 边缘区域的平均分数; 同理, 簇 2 存在点x_1, 且x_1的低密度分数大于簇边缘区域的平均分数. 则簇 1 与簇 2 密度相交, 融合所有密度相交的子簇完成聚类.

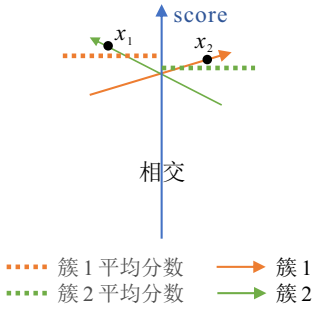


图3　两相邻子簇密度相交

2.2　根据低密度分数获得子簇

使用低密度分数重构簇的密度, 并根据低密度分数获得子簇.

定义 3 (数据点密度)　所提出算法使用的数据点x_i的密度ρ_i如下所示, 定义为数据点x_i距其$k\mathrm{NN}_{x_i}$中所有点距离的平均值的倒数:

$$\rho_i = \frac{k}{\sum\limits_{x_j \in k\mathrm{NN}_{x_i}} d(x_i, x_j)}. \tag{4}$$

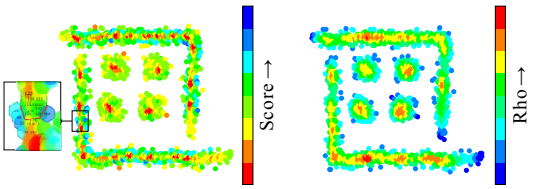
定义 4 (低密度分数)　点x_i的低密度分数score_i定义为$k\mathrm{NN}_{x_i}$内比x_i密度大的点的个数, 如下所示:

$$\mathrm{score}_i = \sum_{x_j \in k\mathrm{NN}_{x_i}} \chi(\rho_i - \rho_j). \tag{5}$$

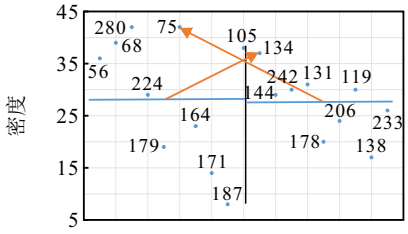
其中: 若$x<0$, 则$\chi(x)=1$; 否则, $\chi(x)=0$. 一个点的低密度分数越大, 在该点k近邻内x_i的密度相对越小; 一个点的低密度分数越小, 在该点k近邻内其密度相对越大. 此外, 当低密度分数为 0 时, 该点为近邻内密度峰值点, 记为子簇中心点.

近邻内的数据点使用原始密度计算往往只会呈现出细微差异, 而低密度分数考虑了点与k近邻内点的密度大小关系, 使得密度差异很小的两个点也可以根据低密度分数的值放大两个点的密度差异. 此外, 低密度分数的计算只考虑数据点间的密度大小关系, 并不考虑数据点本身的密度大小. 因此, 在两个相邻的整体密度差异较大的区域, 低密度分数可缩小两个区域的密度差异并使得两个区域的密度分布变为更相似的单峰密度分布. 图 4(a) 为 Ls 数据集上的低密度分数分布, 图 4(b) 为 Ls 数据集上的密度分布. 由图 4 可见: 在图 4(b) 的原始密度上, 上方的

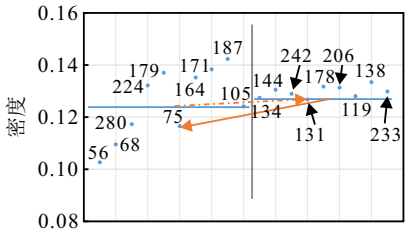
任意形状簇存在多个高密度区域 (呈现红色), 但是该簇内的高密度区域间存在分布较为平缓的低密度区域 (呈现绿色), 簇内密度分布差异较大; 而在图 4(a)展示的低密度分数的密度分布上, 每个区域的密度分布均呈现中心红色高密度区域边缘绿色低密度区域的单峰密度分布, 使得簇内的每个区域均保留相似的单峰密度分布.



(a) LS 数据集上的低密度分数分布 (b) LS 数据集上的密度分布



(c) 两相邻子簇相交处的低密度分数分布



(d) 两相邻子簇相交处的密度分布

图4 Ls 数据集上的低密度分数分布和密度分布以及两个相邻子簇相交处点的低密度分数分布和密度分布

在获得低密度分数后, 将低密度分数值为 0 的点自动获取为子簇中心点, 并为所有子簇中心点分配单独的标签, 再从最高密度点开始将数据点的标签赋为该点最近高密度点的标签, 完成子簇划分.

2.3 根据密度相交条件融合子簇

在获得子簇后, 通过判断两个簇是否满足密度相交条件来对两个簇进行融合. 相关定义如下.

定义 5 (簇m的平均距离) 簇的平均距离定义为簇内所有点到簇中心点距离的平均值, 即

$$\text{average}_{d_m}=\frac{1}{|C_m|}\sum_{x_i\in C_m}d(x_i,c_m). \tag{6}$$

其中: C_m为子簇m, c_m为子簇m的簇中心点.

定义 6 (簇C_m对簇C_l的边缘点) 簇C_m对簇C_l的边缘点定义为簇C_m和簇C_l内两个点距离小于两个簇最小簇平均距离的点, 如下所示:

$$\begin{aligned}&\text{Border}_m^l=\\&\{x_i|d(x_i,x_j)<\min(\text{average}_{d_m},\text{average}_{d_l}),\\&\forall x_i\in C_m,x_j\in C_l\}.\end{aligned} \tag{7}$$

注意, 在簇与不同簇比较时, 簇的边缘点是存在变动的.

定义 7 (簇C_m对簇C_l的平均分数) 簇C_m对簇C_l的边缘点的低密度分数平均值定义为簇C_m对簇C_l的平均分数, 如下所示:

$$\text{Score}_m^l=\frac{1}{|\text{Border}_m^l|}\sum_{x_i\in\text{Border}_m^l}\text{score}_{x_i}. \tag{8}$$

定义 8 (密度相交) 若簇C_m与簇C_l密度相交, 则两个簇应满足下列条件: 1) $\text{Border}_m^l\neq\varnothing$且$\text{Border}_l^m\neq\varnothing$; 2) $\exists x_i\in\text{Border}_m^l$ & $\exists x_j\in\text{Border}_l^m,\text{score}_i>\text{Score}_l^m$ & $\text{score}_j>\text{Score}_m^l$. 其中: 条件 1) 表明两个子簇的边缘区域存在数据点, 保证了两个子簇距离接近; 条件 2) 表明在两个子簇边缘区域内均存在点的低密度分数大于另一个子簇的平均分数, 保证了两个子簇的边缘区域是密度相交的.

注 1 由于低密度分数与原始密度成反比, 用图表示, 原始密度呈现的相交趋势与图 3 相反, 为向下相交.

同样地, 使用图 4 中 Ls 数据集表明使用低密度分数来判断密度相交对于子簇融合的有效性. 图 4(c)为图 4(a) 框出的两个子簇的低密度分数分布, 图 4(d)为图 4(a) 框出的两个子簇的密度分布. 如图 4(d)所示: 在使用密度判断时, 上方子簇 (位于黑色竖线右侧) 中数据点的密度分布较为平缓, 而下方子簇 (位于黑色竖线左侧) 中数据点的密度有轻微的波动趋势, 且图 4(d) 中右侧簇内所有边缘点密度均大于左侧簇的平均密度, 两个橙色箭头没有向下相交, 表明两个簇的边缘区域没有密度相交, 没有满足子簇融合条件. 如图 4(c) 所示: 使用 score 进行判断时, 可以明显看出子簇内点的密度分布变大, 两簇均有满足密度相交条件的点, 满足了子簇融合的条件, 且由图 4(a) 和图 4(b) 也可以更清楚地观察到框出两个子簇的低密度分数对比原始密度有更相似的密度分布, 因此使用低密度分数时, 可使得两个子簇顺利合并.

2.4 算法流程

算法 1 LS-DPC 的流程如下.

input: 数据集$D=\{x_1,x_2,...,x_n\}$, 近邻数k;

output: 最终簇C.

step 1: 初始化.

1. $C\leftarrow\varnothing$;

2. foreach $x_i\in D$ do

由式 (4) 计算ρ_i;

step 2: 计算低密度分数并获得所有子簇.

1. $D_a \leftarrow$ 将D中数据点按照ρ升序排序;

2. foreach $x_i \in D_a$ do

由式 (5) 计算score_i;

3. 选择$\text{score}_i = 0$的数据点作为子簇中心点;

4. 为每个子簇中心点分配标签;

5. $D_d \leftarrow$ 将D中数据点按照ρ降序排序;

6. foreach $x_i \in D_d$ do

将x_i的标签赋为其最近高密度点的标签;

将有相同标签的数据点放到同一个子簇C_m中;

7. 将每个子簇C_m加入C中.

step 3: 融合子簇.

1. foreach $C_m \in C$ do

由式 (6) 计算average_{dm};

2. foreach $C_m \in C$ do

foreach $C_l \in C$ do

由式 (7) 和 (8) 分别计算Border_m^l、Border_l^m、Score_m^l 和 Score_l^m;

if C_m 和 C_l 满足定义 8 then

$C_m = C_m \bigcup C_l, C = C/C_l$

3. return C

2.5 算法时间复杂度分析

LS-DPC 的时间复杂度由以下几部分组成: 1) 计算样本间欧氏距离的时间复杂度为$O(n^2)$, 其中n为数据点个数; 2) 获取数据点的密度和低密度分数需要遍历每个点的k近邻, 在使用kd-Tree[24] 的情况下, 时间复杂度为$O(n\log n)$; 3) 分配子簇标签的时间复杂度为$O(n\log n)$; 4) 融合子簇部分的时间复杂度主要是在比较两个子簇的边缘点间的距离, 时间复杂度为$O(n^2)$. 综上, LS-DPC 与 DPC 的时间复杂度相同, 均为$O(n^2)$.

3 实验结果与分析

3.1 实验设置

为了验证算法的有效性, 将所提出算法与k-Means[6]、SC[7]、DPC[12]、DN[25]、Extreme[26] 以及 ICKDP[22] 这 6 个对比算法在 10 个复杂数据集和 8 个 UCI 数据集上进行聚类比较. 10 个复杂数据集中除 Aggregation、Asymmetric 和 R15, 均存在包含多密度峰值的簇, 且 Zigzag、Moon、Ring、DB、DB3 和 LS 还包含任意形状的簇. 数据集点数n、维度d、簇数e和各算法的参数如表 2 所示. 其中: DK 为密度核, CF 为截断核, GS 为高斯核, K为根据n得到的自由参数. 具体而言, DN、Extreme 和 LS-DPC 的参数为在给定范围内寻优所得. 这里: DN 的两个参数寻优范围为 1 ~ 50 (步长为 1), Extreme 的参数δ寻优范围为 0.1 ~ 1 000 (步长为 0.1), LS-DPC 的参数k寻优范围为数据集点数的1 % ~ 10 % (步长为 1). 其余算法均采用固定参数. 其中: k-Means 和 SC 设置e为数据集的簇个数, DPC 设置d_c为2 %, ICKDP 设置输入参数为$K/2$. 实验采用评价指标调整兰德指数 (ARI)[27]、标准化互信息 (NMI)[28] 和F_1分数 (F_1) 对聚类结果进行量化比较.

3.2 复杂数据集的实验结果与分析

为了验证算法对复杂数据集的有效性, 本实验将所提出算法在 10 个复杂数据集上与 6 个对比算法进行比较. 聚类结果量化指标如表 3 所示. 图 5 ~ 图 11 为 7 个包含多密度峰值簇的数据集.

如图 5 所示: 只有 LS-DPC 可以识别出 Zigzag 数据集的 3 个簇, 其他算法均将 Zigzag 数据集中的“Z”型簇划分为多个簇. 图 6 为 7 种算法在 Moon 数据集上的聚类结果. 由图 6 可见: 只有 LS-DPC 和 DN 将两个月牙形簇识别出来, 其余算法均将一个或全部簇划分为两段. 图 7 为 DB 数据集经 7 种算法聚类的结果. 其中: 只有 LS-DPC 将所有簇识别正确; 在其余算法中, SC 将所有簇识别为一个簇, k-Means、DPC、DN、Extreme 和 ICKDP 将反“S”型的簇和右下方的簇错误地划分为多个簇. 对于 DB3 数据集, 如图 8 所示: LS-DPC 和 DN 可以识别出主要的簇, k-Means、DPC 和 ICKDP 将“Y”型簇划分为多个簇, SC 几乎将所有点划分至一个簇, 而 Extreme 则将上方两个簇和下方两个簇分别合成为一个簇. 如图 9 所示: LS-DPC 和 DPC 可将 Line 数据集中的簇全部识别正确, SC 算法将所有簇划分为一个簇, 其余算法均将某一个或多个簇分别划分为多个簇. 同样地, 只有 LS-DPC 能够正确识别 Ls 数据集, 如图 10 所示: SC 仍然将几乎所有点划分为一个簇, 其余算法将包含多密度峰值的左下簇和右上簇划分为多个簇. 图 11 为 Ring 数据集的聚类结果. 其中: 只有 LS-DPC 正确地识别了所有簇; 在其余算法中, k-Means、SC 和 DPC 分别将内圈和外圈的两个簇划分为两段, 剩余算法 DN、Extreme 和 ICKDP 则将外圈的簇错误地划分为多个簇.

图 5 ~ 图 11 和表 3 表明了 LS-DPC 算法可以识别包含多密度峰值的簇, 此外, 对于任意形状的簇, 如 Zigzag、Moon、DB、DB3、Ls 和 Ring, LS-DPC 也能够较好地识别. 相比之下, k-Means 算法不能较好地识别出这 7 个包含多密度峰值簇的数据集, 且只在 R15 和 Asymmetric 这两个简单的球状簇数据集上表现良好. SC 同样只能识别出 Aggregation 和 R15

表2 数据集信息和算法参数

datasets	n	d	e	k-Means	SC	DPC	DN	Extreme	ICKDP	LS-DPC
				e	e	$d_c=2\,\%$	m/n	δ/DK	$K/2$	k
Zigzag	1 002	2	3	3	3	−	22/18	0.6/GS	−	76
Moon	1 000	2	2	2	2	−	16/2	0.6/CF	−	53
DB	629	2	4	4	4	−	15/2	46.0/CF	−	25
DB3	582	2	5	5	5	−	28/4	140.0/CF	−	36
Line	1 267	2	4	4	4	−	26/2	41.0/CF	−	39
Ls	1 725	2	6	6	6	−	13/2	41.0/CF	−	49
Ring	1 000	2	2	2	2	−	3/7	0.6/GS	−	71
Aggregation	788	2	7	7	7	−	18/2	4.8/CF	−	32
R15	600	2	15	15	15	−	2/2	0.8/CF	−	25
Asymmetric	1 000	2	5	5	5	−	3/2	120.0/CF	−	18
Glass	214	9	6	6	6	−	6/3	1.0/CF	−	12
Leaf	340	14	30	30	30	−	4/13	0.2/CF	−	6
Shuttle	2 175	9	5	5	5	−	8/10	24.2/CF	−	28
Wdbc	569	30	2	2	2	−	13/2	50.8/CF	−	21
SPECTF	214	44	2	2	2	−	3/27	4.9/CF	−	17
Libra	360	90	15	15	15	−	11/9	57.8/CF	−	7
COIL20	1 440	1 024	20	20	20	−	19/6	17.8/CF	−	7
Orl_vector10	100	10 304	10	10	10	−	4/2	1.1/CF	−	10

表3 7 个算法在 10 个复杂数据集上的聚类结果评价指标

datasets		k-Means	SC	DPC	DN	Extreme	ICKDP	LS-DPC
Zigzag	ARI	0.366 2	0.319 9	0.266 3	0.445 5	*0.545 6*	0.296 7	**1.000 0**
	NMI	0.517 6	0.486 8	0.455 2	0.641 9	*0.759 9*	0.468 9	**1.000 0**
	F1	0.712 6	0.666 3	0.584 5	0.605 3	*0.756 7*	0.558 8	**1.000 0**
Moon	ARI	0.245 3	0.293 1	0.295 4	**1.000 0**	*0.519 5*	0.295 4	**1.000 0**
	NMI	0.185 6	0.223 9	0.376 4	**1.000 0**	*0.712 4*	0.347 1	**1.000 0**
	F1	0.748 0	*0.771 0*	0.759 5	**1.000 0**	0.750 0	0.680 5	**1.000 0**
DB	ARI	0.284 8	0.008 9	0.127 9	*0.351 2*	0.304 6	0.220 5	**1.000 0**
	NMI	0.440 2	0.013 8	0.391 9	0.479 1	*0.695 5*	0.473 0	**1.000 0**
	F1	0.579 4	0.518 3	0.470 3	0.629 6	*0.662 9*	0.560 2	**1.000 0**
DB3	ARI	0.480 4	0.003 8	0.324 9	*0.963 4*	0.527 4	0.433 5	**0.997 9**
	NMI	0.684 4	0.009 4	0.572 6	*0.927 0*	0.646 4	0.600 5	**0.972 4**
	F1	0.745 4	0.410 0	0.546 0	*0.960 4*	0.591 2	0587 6	**0.983 5**
Line	ARI	0.598 2	−0.000 2	**1.000 0**	0.743 9	*0.945 0*	0.743 5	**1.000 0**
	NMI	0.682 7	0.002 8	**1.000 0**	0.830 1	*0.957 2*	0.804 9	**1.000 0**
	F1	0.723 9	0.199 5	**1.000 0**	0.731 8	*0.977 5*	0.819 4	**1.000 0**
Ls	ARI	0.343 3	−0.002 5	0.454 8	*0.560 6*	0.427 8	0.518 2	**1.000 0**
	NMI	0.557 7	0.004 2	0.675 8	*0.709 9*	0.709 7	0.654 1	**1.000 0**
	F1	0.502 9	0.226 5	*0.688 2*	0.619 3	0.668 9	0.639 0	**1.000 0**
Ring	ARI	0.000 6	−0.001 0	0.003 7	0.442 4	*0.604 0*	0.001 7	**1.000 0**
	NMI	0.001 2	0.000 0	0.003 7	0.484 2	*0.682 9*	0.002 0	**1.000 0**
	F1	0.265 2	0.502 5	0.524 4	*0.765 6*	0.712 6	0.500 8	**1.000 0**
Aggregation	ARI	0.767 4	0.989 8	*0.997 8*	**1.000 0**	*0.997 8*	0.992 0	**1.000 0**
	NMI	0.884 2	0.985 1	*0.995 7*	**1.000 0**	*0.995 7*	0.986 8	**1.000 0**
	F1	0.817 2	0.995 0	*0.998 7*	**1.000 0**	*0.998 7*	0.993 7	**1.000 0**
R15	ARI	0.906 0	**0.992 8**	**0.992 8**	**0.992 8**	*0.989 2*	**0.992 8**	**0.992 8**
	NMI	0.968 4	**0.994 2**	**0.994 2**	**0.994 2**	*0.991 3*	**0.994 2**	**0.994 2**
	F1	0.885 6	**0.996 7**	**0.996 7**	**0.996 7**	*0.995 0*	0.993 2	**0.996 7**
Asymmetric	ARI	0.972 7	0.000 0	**0.985 2**	**0.985 2**	**0.985 2**	*0.974 9*	**0.985 2**
	NMI	0.966 5	0.007 9	**0.980 8**	**0.980 8**	**0.980 8**	*0.968 5*	**0.980 8**
	F1	*0.989 0*	0.082 3	**0.994 0**	**0.994 0**	**0.994 0**	0.979 9	**0.994 0**

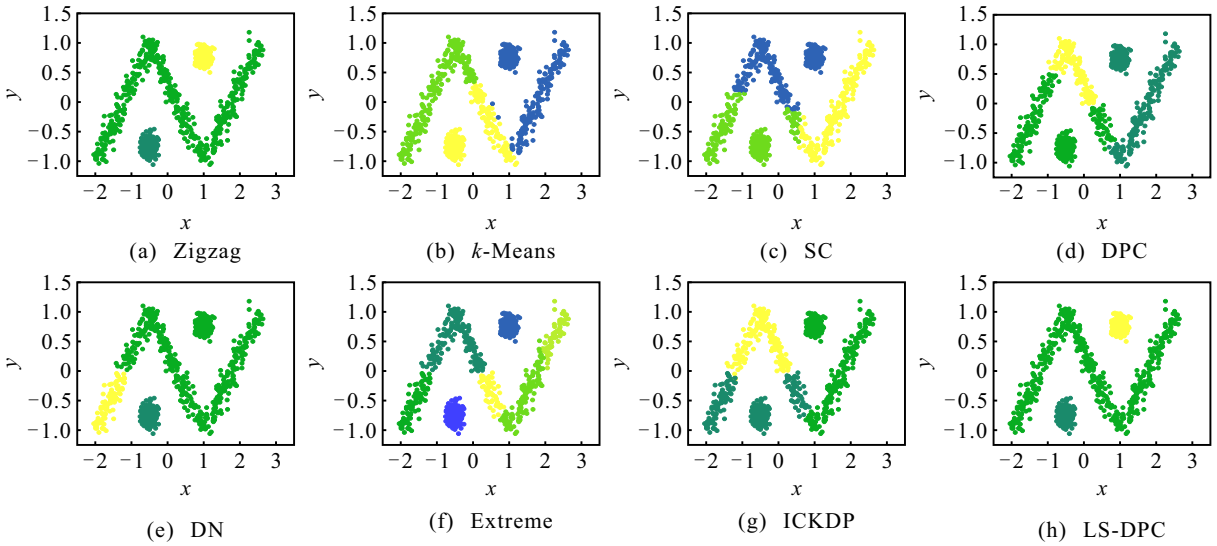


图5　7 种算法在 Zigzag 数据集上的聚类结果

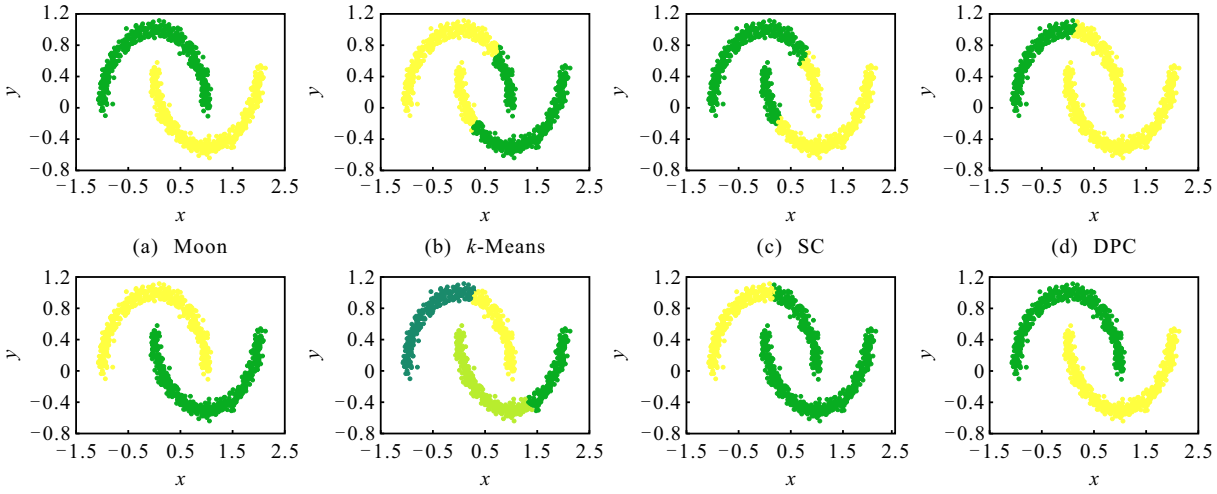


图6　7 种算法在 Moon 数据集上的聚类结果

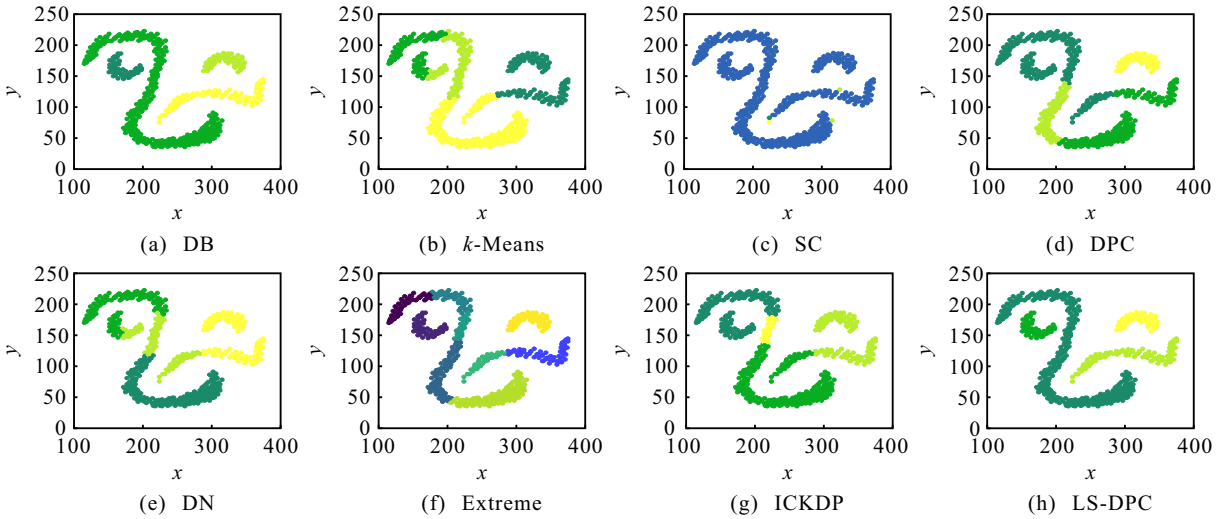


图7　7 种算法在 DB 数据集上的聚类结果

这两个数据集. 在包含多密度峰值簇的数据集中, DPC 能够识别出簇中心点间相对距离明显的 Line 数据集, 但是在其他数据集中, DPC 错误地将包含多密度峰值的单个簇划分为多个簇. DN 数据集可将 Aggregation、Moon、DB3、R15 和 Asymmetric 中的簇识别出来, 但是对于 Zigzag、DB、Ls 和 Ring 这种包含任意形状簇的复杂数据集的聚类效果较差. Extreme 和 ICKDP 只能识别出 Aggregation、R15 和

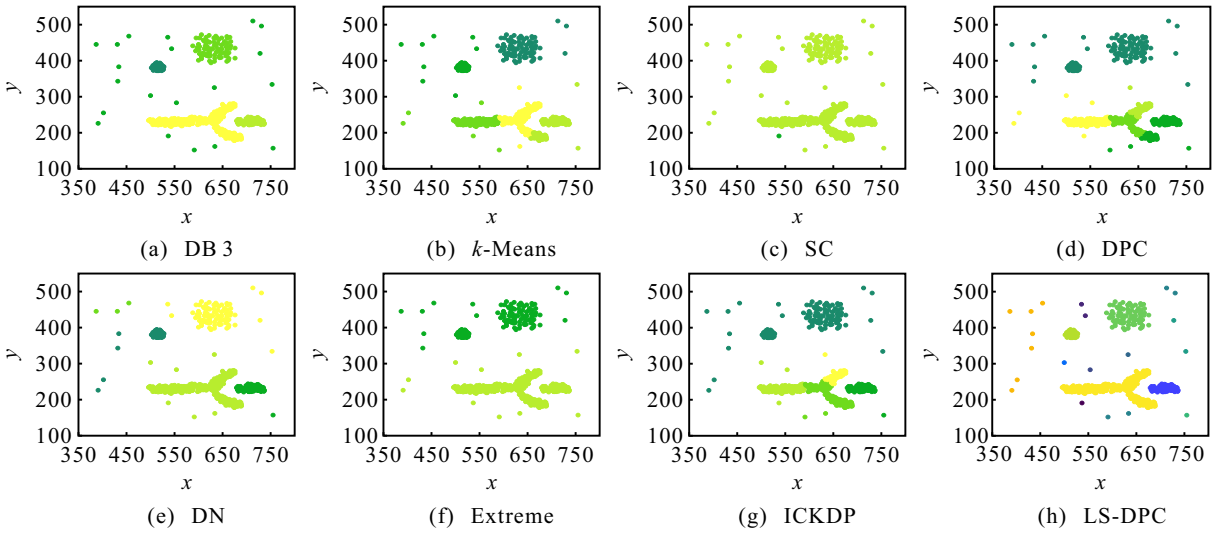


图8 7 种算法在 DB3 数据集上的聚类结果

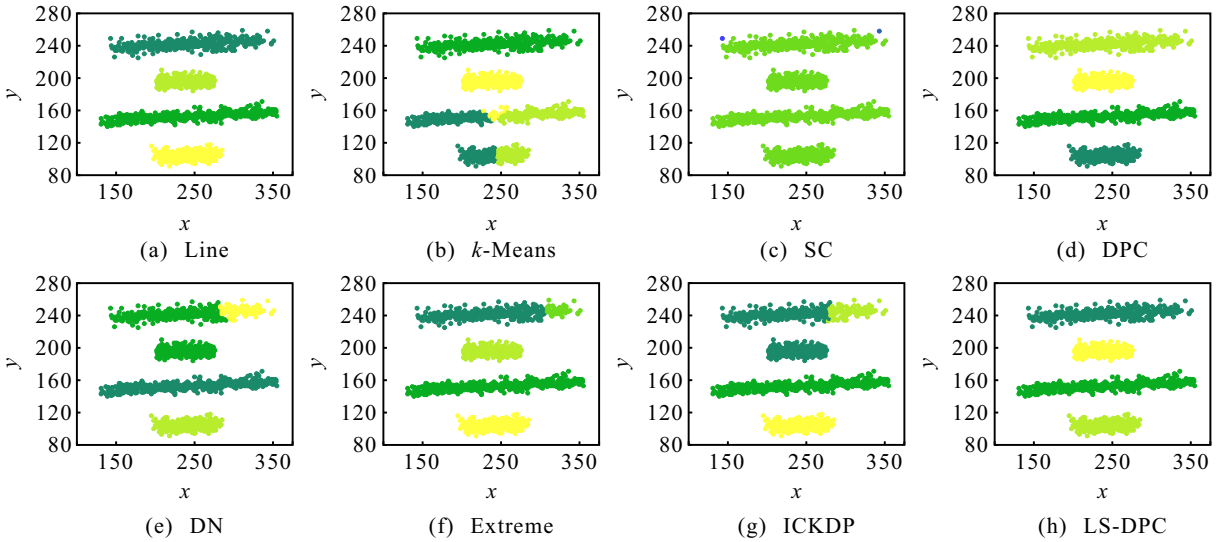


图9 7 种算法在 Line 数据集上的聚类结果

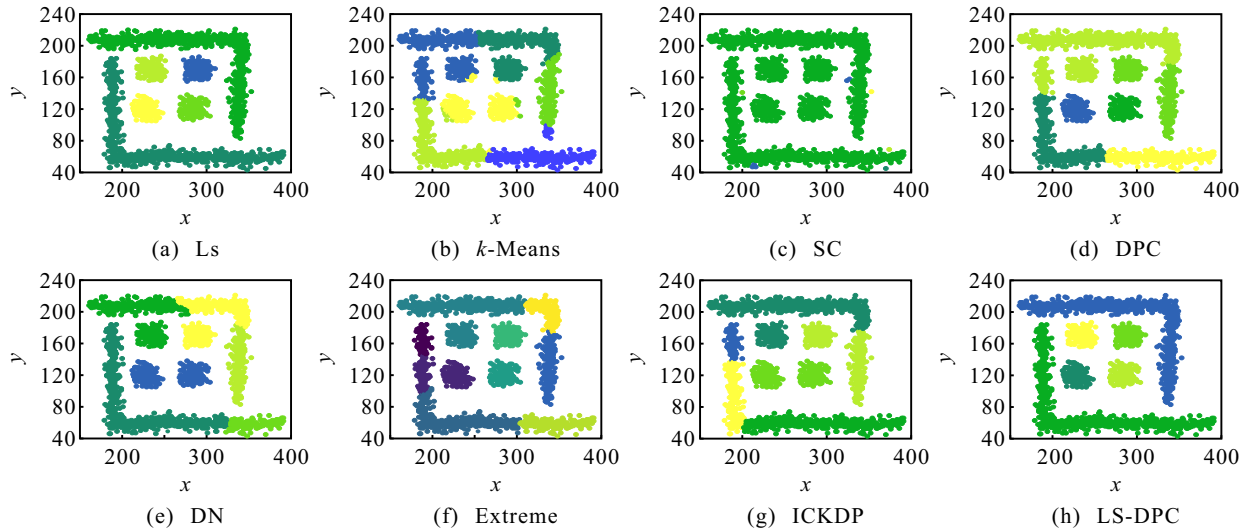


图10 7 种算法在 Ls 数据集上的聚类结果

Asymmetric 中的簇. 由表 3 可见, LS-DPC 在 10 个复杂数据集上均获得了最高的 ARI、NMI 和F_1.

综上所述, LS-DPC 在处理包含多密度峰值簇以及任意形状簇等复杂数据集上表现较好.

3.3 UCI 数据集的实验结果与分析

为了进一步验证算法的有效性, 将所提出算法与对比算法在真实 UCI 数据集上进行聚类对比. 如表 4 所示: LS-DPC 在所有 UCI 数据集中的 ARI 指

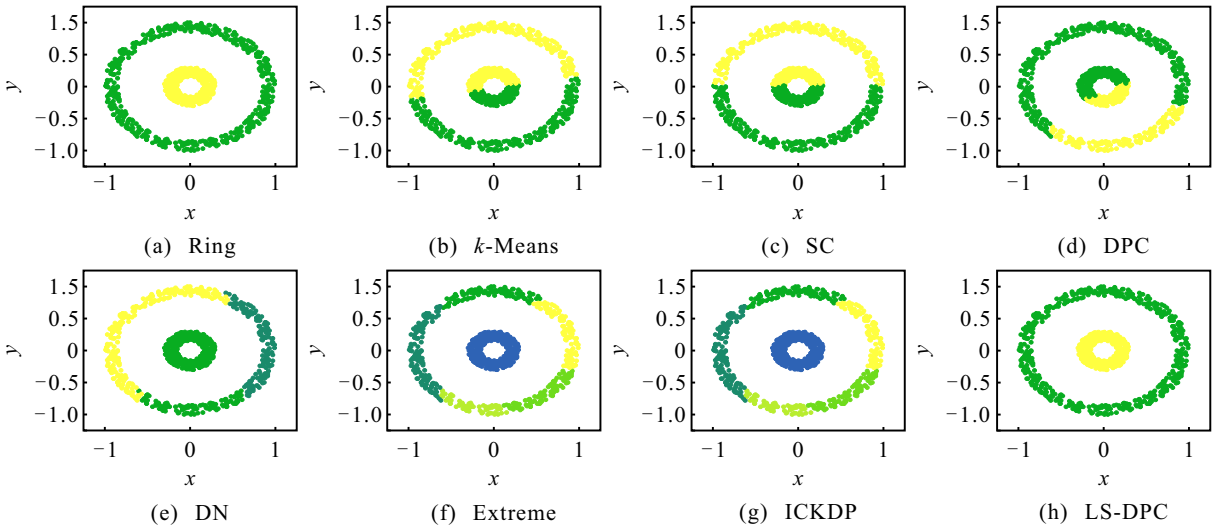


图11　7 种算法在 Ring 数据集上的聚类结果

标均排名最高, 在 Glass 和 Leaf 数据集中的 NMI 指标和F_1分数也能排名第 2, 仅在 SPECTF 数据集上的F_1分数排名第 3. 综上, 表明所提出算法对于 UCI 数据集也有较好的聚类效果.

3.4　参数k的分析

3.4.1　k对于数据点的低密度分数以及子簇划分的影响

为了验证参数k对于数据点的低密度分数以及子簇划分的影响, 本实验选取 DB 数据集分别设置$k=5, 10, 15, 20, 25$来展示不同数据点在不同k值下的低密度分数变化以及不同k值下获得子簇的个数. 图 12(a) ～ 图 12(e) 为 DB 数据集中点在不同k下的低密度分布. 其中: 标注值靠左的为 DB 数据集中第 327 号数据点的低密度分数, 标注值靠右的为 DB 数据集中第 21 号数据点的低密度分数. 图 12(f) ～ 图 12(j) 为不同k值下获得子簇划分. 由图 12 可见: 随着k值的增加, 靠右数据点的低密度分数不断增大, 同时子簇个数不断减小. 这表明随着k的增加, 数据点和

表4　7 个算法在 8 个 UCI 数据集上的聚类结果评价指标

datasets		*k*-Means	SC	DPC	DN	Extreme	ICKDP	LS-DPC
Glass	ARI	*0.281 2*	0.053 5	0.135 0	0.205 9	0.257 2	0.183 8	**0.281 9**
	NMI	0.450 8	0.170 9	0.266 8	0.361 9	**0.507 2**	0.288 8	*0.473 2*
	F1	0.380 6	0.239 0	0.416 0	0.369 8	*0.461 4*	0.451 4	**0.506 5**
Leaf	ARI	*0.352 7*	0.095 5	0.316 7	0.332 0	0.150 4	0.341 5	**0.355 7**
	NMI	0.681 5	0.496 4	0.678 0	0.691 7	**0.712 0**	0.641 5	*0.707 9*
	F1	*0.477 2*	0.165 7	0.296 4	0.436 1	0.383 3	0.365 5	**0.510 2**
Shuttle	ARI	0.315 3	−0.000 7	0.162 6	*0.703 1*	0.699 1	0.444 0	**0.893 7**
	NMI	0.312 4	0.000 3	0.138 9	*0.640 1*	0.636 3	0.418 7	**0.749 6**
	F1	0.733 2	0.689 4	0.627 9	0.889 1	*0.890 2*	0.781 8	**0.907 8**
Wdbc	ARI	0.491 4	0.000 0	0.317 7	*0.528 4*	0.517 1	0.512 4	**0.571 1**
	NMI	0.464 8	0.000 0	0.343 5	**0.493 8**	0.347 7	0.440 3	*0.478 6*
	F1	0.844 3	0.483 8	0.765 0	**0.858 6**	0.641 3	*0.795 1*	0.794 9
SPECTF	ARI	0.116 0	0.001 3	0.022 3	0.116 0	*0.171 8*	0.018 8	**0.179 8**
	NMI	*0.242 0*	0.043 6	0.038 7	*0.242 0*	0.199 3	0.023 4	**0.259 7**
	F1	**0.636 6**	0.339 0	0.339 8	**0.636 6**	0.545 8	0.515 5	*0.571 2*
Libra	ARI	0.347 9	0.279 6	0.251 2	*0.374 1*	0.331 5	0.318 2	**0.400 8**
	NMI	0.618 9	0.588 7	0.573 1	*0.662 5*	0.648 4	0.562 2	**0.701 2**
	F1	*0.468 7*	0.370 0	0.311 1	0.442 2	0.456 2	0.371 0	**0.507 0**
COIL20	ARI	0.627 1	0.466 2	0.529 7	0.374 5	*0.632 8*	0.313 0	**0.771 1**
	NMI	0.790 5	*0.846 0*	0.806 0	0.780 4	0.838 0	0.642 1	**0.918 7**
	F1	*0.656 3*	0.531 5	0.555 4	0.500 3	0.648 3	0.363 6	**0.808 4**
Orl_vector10	ARI	*0.911 4*	0.025 3	0.698 6	0.888 0	0.736 4	0.672 8	**0.957 0**
	NMI	0.943 3	0.256 4	0.870 5	*0.949 6*	0.857 6	0.805 2	**0.978 4**
	F1	*0.958 0*	0.084 7	0.735 9	0.936 6	0.789 0	0.705 2	**0.961 4**

k近邻内点的密度比较次数增加, 密度较低的点会根据低密度分数的定义获得更大的低密度分数, 且随着k的增大, 更大范围内的数据点在互相比较密度, 因此, 形成更大的单峰密度分布区域, 从而导致子簇个数减少, 减少后续子簇融合的比较次数. 但是, 由图 12(a) ~ 图 12(e) 中标注值靠左的可以发现, 对于密度较大的局部密度峰值, k的增大对其影响较小, 因此, 它的低密度分数一直保持在 0.

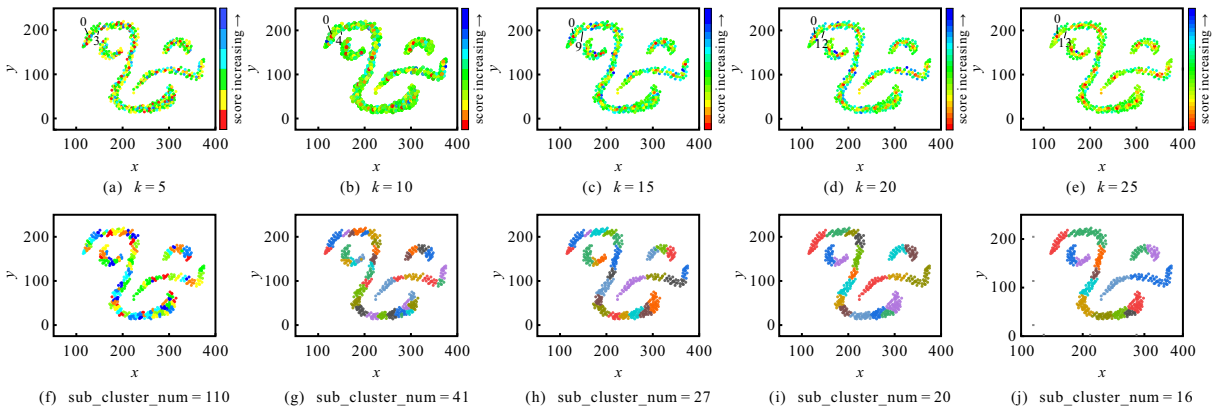


图12 不同k值下 DB 数据集的低密度分数分布和子簇划分

3.4.2 k对聚类结果的影响

为了验证 LS-DPC 算法的聚类结果对于参数k的敏感性并给出参数选取范围, 本实验选取 Zigzag、Line、Ls、R15 和 Glass 这 5 个数据集来讨论所提出算法中唯一参数k的敏感性. 本实验中k的测试范围为2 ~ 80 (k取值大于 1, 这是因为$k=1$时近邻为数据点本身), 具体结果如图 13 所示. 由图 13 可见, 随着k的增大, LS-DPC 在数据集上的聚类结果呈现逐渐升高然后平稳或逐渐升高平稳然后稳步下降. 然而, 由于不同数据集存在不同的数据点大小和不同的簇密度分布特征, k的取值会受到影响. 为了得到合适的参数k, 本实验取 ARI 在最高 ARI 的90 % (在图 13 中用横线标出) 以上连续的k取值范围进行讨论, 具体如下: 对于 Zigzag 数据集, 参数k在43 ~ 80时满足条件, 其值占 Zigzag 数据集点数的4 % ~ 8 %; 对于 Line 数据集, 参数k在50 ~ 80时满足条件, 其值占 Line 数据集点数的4 % ~ 6 % ; 对于 Ls 数据集, 参数k在49 ~ 80时满足条件, 其值占 Ls 数据集点数的3 % ~ 5 %; 对于 R15 数据集, 参数k在10 ~ 55时满足条件, 其值占 R15 数据集点数的2 % ~ 9 %; 对于 Glass 数据集, 参数k在9 ~ 18时满足条件, 其值占 Glass 数据集点数的4 % ~ 8 %. 同时, 考虑到不同数据集的数据点分布差异, 本文建议将上述选取范围的最大上限和最大下限分别扩大1 %, 按照数据集点数的1 % ~ 10 %选取k值.

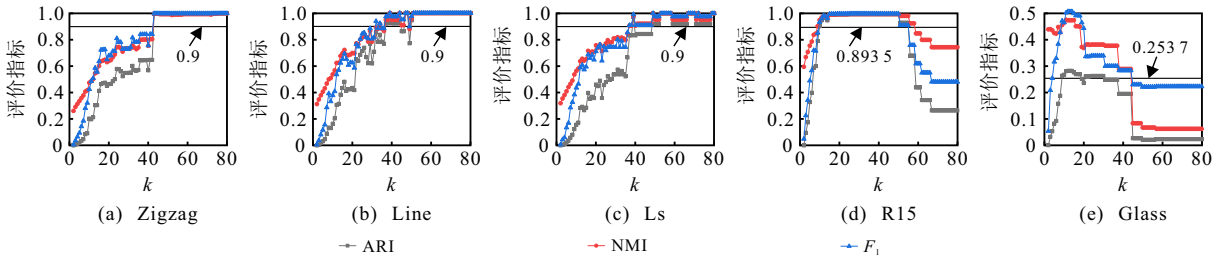


图13 不同k值下的评价指标结果

4 结 论

针对 DPC 在包含多密度峰值簇的数据集上表现较差的问题, 本文提出了基于低密度分数的密度峰值聚类算法 LS-DPC. LS-DPC 使用低密度分数将簇内每个区域的密度分布重构为单峰密度分布, 并自动获得了所有子簇的簇中心点. 根据簇中心点获得子簇后, 使用密度相交条件对子簇进行了融合. LS-DPC 不需要提前知道簇的个数, 只有一个参数k, 无其他阈值参数, 且算法对参数k鲁棒. 为了验证所提出算法的有效性, 本文将 LS-DPC 与 6 个对比算法在 10 个复杂数据集和 8 个 UCI 数据集上进行了比较. 聚类结果表明, LS-DPC 在处理包含多密度峰值簇的数据集和真实 UCI 数据集上均表现较好. 未来的研究将专注于如何提高算法效率.

参考文献 (References)

[1] Wang Y B, Saraswat S K, Komari I E. Big data analysis using a parallel ensemble clustering architecture and an unsupervised feature selection approach[J]. Journal of King Saud University-Computer and Information Sciences, 2023, 35(1): 270-282.

[2] Liang K, Liu Y, He H, et al. Characteristic analysis of

10kV bus load based on integrated clustering technology[J]. Energy Reports, 2022, 8: 413-419.
[3] McCarthy D, Apidianaki M, Erk K. Word sense clustering and clusterability[J]. Computational Linguistics, 2016, 42(2): 245-275.
[4] Munahar S, Triwiyatno A, Setiawan J D, et al. Assessment of fuel management clusters in the development of a driving behavior control system model using lookup table mapping to improve fuel savings[J]. Results in Engineering, 2023, 18: 101170.
[5] Chen M, Chen Y X, Zhu H Y, et al. Analysis of pollutants transport in heavy air pollution processes using a new complex-network-based model[J]. Atmospheric Environment, 2023, 292: 119395.
[6] MacQueen J. Some methods for classification and analysis of multivariate bservations[C]. Proceedings of the 5th Berkeley symposium on mathematical statistics and probability. Berkeley, 1967: 281-297.
[7] von Luxburg U. A tutorial on spectral clustering[J]. Statistics and Computing, 2007, 17(4): 395-416.
[8] Karypis G, Han E H, Kumar V. Chameleon: Hierarchical clustering using dynamic modeling[J]. Computer, 1999, 32(8): 68-75.
[9] Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases[J]. SIGMOD Record: ACM Special Interest Group on Management of Data, 1996, 25(2): 103-114.
[10] Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[J]. Knowledge Discovery and Data Mining, 1996, 96(34): 226-231.
[11] Chen M, Li L J, Wang B, et al. Effectively clustering by finding density backbone based-on kNN[J]. Pattern Recognition, 2016, 60: 486-498.
[12] Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191): 1492-1496.
[13] Liu R, Wang H, Yu X M. Shared-nearest-neighbor-based clustering by fast search and find of density peaks[J]. Information Sciences, 2018, 450: 200-226.
[14] 陈蔚昌, 赵嘉, 肖人彬, 等. 面向密度分布不均数据的近邻优化密度峰值聚类算法[J]. 控制与决策, 2024, 39(3): 919-928.
(Chen W C, Zhao J, Xiao R B, et al. Density peaks clustering algorithm with nearest neighbor optimization for data with uneven density distribution[J]. Control and Decision, 2024, 39(3): 919-928.)
[15] 周玉, 夏浩, 刘虹瑜, 等. 基于K互近邻与核密度估计的DPC聚类算法[J]. 北京航空航天大学学报, DOI: 10.13700/j.bh.1001-5965.2023.0342.
(Zhou Y, Xia H, Liu H Y, et al. DPC clustering algorithm based on K-reciprocal neighbors and kernel density estimation[J]. Journal of Beijing University of Aeronautics and Astronautics, DOI: 10.13700/j.bh.1001-5965.2023.0342.)
[16] 吕莉, 朱梅子, 康平, 等. 面向分布不均数据的混合近邻密度峰值聚类算法[J]. 控制理论与应用, 2024, 41(10): 1821-1830.
(Lv L, Zhu M Z, Kang P, et al. Multiplex neighbor density peaks clustering for uneven density data sets[J]. Control Theory & Applications, 2024 ,41(10): 1821-1830.)
[17] 吴润秀, 尹士豪, 赵嘉, 等. 基于相对密度估计和多簇合并的密度峰值聚类算法[J]. 控制与决策, 2023, 38(4): 1047-1055.
(Wu R X, Yin S H, Zhao J, et al. Density peaks clustering based on relative density estimating and multi cluster merging[J]. Control and Decision, 2023, 38(4): 1047-1055.)
[18] Lotfi A, Moradi P, Beigy H. Density peaks clustering based on density backbone and fuzzy neighborhood[J]. Pattern Recognition, 2020, 107: 107449.
[19] Zhang Q H, Dai Y Y, Wang G Y. Density peaks clustering based on balance density and connectivity[J]. Pattern Recognition, 2023, 134: 109052.
[20] Guo W J, Wang W H, Zhao S P, et al. Density peak clustering with connectivity estimation[J]. Knowledge-Based Systems, 2022, 243: 108501.
[21] Tao X M, Guo W J, Ren C, et al. Density peak clustering using global and local consistency adjustable manifold distance[J]. Information Sciences, 2021, 577: 769-804.
[22] Guo W J, Chen W, Liu X G. Density peak clustering by local centers and improved connectivity kernel[J]. Information Sciences, 2024, 666: 120439.
[23] Li Z J, Tang Y C. Comparative density peaks clustering[J]. Expert Systems with Applications, 2018, 95: 236-247.
[24] Gieseke F, Heinermann J, Oancea C, et al. Buffer k-d trees: Processing massive nearest neighbor queries on GPUs[C]. International Conference on Machine Learning. Beijing, 2014: 172-180.
[25] Hou J, Zhang A H. Enhancing density peak clustering via density normalization[J]. IEEE Transactions on Industrial Informatics, 2020, 16(4): 2477-2485.
[26] Wang S L, Li Q, Zhao C F, et al. Extreme clustering — A clustering method via density extreme points[J]. Information Sciences, 2021, 542: 24-39.
[27] Vinh N X, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance[J]. Journal of Machine Learning Research, 2010, 11: 2837-2854.
[28] Chen W Y, Song Y Q, Bai H J, et al. Parallel spectral clustering in distributed systems[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(3): 568-586.

作者简介

陈梅 (1973−), 女, 教授, 博士, 主要研究方向为数据挖掘、人工智能、复杂网络, E-mail: mei.chen.lzjtu@hotmail.com;

尤远毓秀 (2000−), 女, 硕士生, 主要研究方向为数据挖掘和模式识别, E-mail: 2428018162@qq.com;

魏礼磊 (2000−), 男, 硕士生, 主要研究方向为数据挖掘和密度聚类, E-mail: 846581063@qq.com;

唐晟洲 (1999−), 男, 硕士生, 主要研究方向为数据挖掘和密度聚类, E-mail: 3576226392@qq.com.