软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(12):5629−5648 [doi: 10.13328/j.cnki.jos.006756]

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

基于代表点与 K 近邻的密度峰值聚类算法*

张清华[1,2], 周靖鹏[1,2], 代永杨[1,2], 王国胤[1,2]

[1](旅游多源数据感知与决策技术文化和旅游部重点实验室 (重庆邮电大学), 重庆 400065)

[2](计算智能重庆市重点实验室 (重庆邮电大学), 重庆 400065)

通信作者: 张清华, E-mail: zhangqh@cqupt.edu.cn

摘　要: 密度峰值聚类 (density peaks clustering, DPC) 是一种基于密度的聚类算法, 该算法可以直观地确定类簇数量, 识别任意形状的类簇, 并且自动检测、排除异常点. 然而, DPC 仍存在些许不足: 一方面, DPC 算法仅考虑全局分布, 在类簇密度差距较大的数据集聚类效果较差; 另一方面, DPC 中点的分配策略容易导致“多米诺效应”. 为此, 基于代表点 (representative points) 与 K 近邻 (K-nearest neighbors, KNN) 提出了 RKNN-DPC 算法. 首先, 构造了 K 近邻密度, 再引入代表点刻画样本的全局分布, 提出了新的局部密度; 然后, 利用样本的 K 近邻信息, 提出一种加权的 K 近邻分配策略以缓解“多米诺效应”; 最后, 在人工数据集和真实数据集上与 5 种聚类算法进行了对比实验, 实验结果表明, 所提出的 RKNN-DPC 可以更准确地识别类簇中心并且获得更好的聚类结果.

关键词: 聚类分析; 密度峰值聚类; 代表点; K 近邻 (KNN)

中图法分类号: TP18

中文引用格式: 张清华, 周靖鹏, 代永杨, 王国胤. 基于代表点与K近邻的密度峰值聚类算法. 软件学报, 2023, 34(12): 5629–5648. http://www.jos.org.cn/1000-9825/6756.htm

英文引用格式: Zhang QH, Zhou JP, Dai YY, Wang GY. Density Peaks Clustering Algorithm Based on Representative Points and K-nearest Neighbors. Ruan Jian Xue Bao/Journal of Software, 2023, 34(12): 5629–5648 (in Chinese). http://www.jos.org.cn/1000-9825/6756.htm

Density Peaks Clustering Algorithm Based on Representative Points and K-nearest Neighbors

ZHANG Qing-Hua[1,2], ZHOU Jing-Peng[1,2], DAI Yong-Yang[1,2], WANG Guo-Yin[1,2]

[1](Key Laboratory of Tourism Multisource Data Perception and Decision, Ministry of Culture and Tourism (Chongqing University of Posts and Telecommunications), Chongqing 400065, China)

[2](Chongqing Key Laboratory of Computational Intelligence (Chongqing University of Posts and Telecommunications), Chongqing 400065, China)

Abstract: Density peaks clustering (DPC) is a density-based clustering algorithm that can intuitively determine the number of clusters, identify clusters of any shape, and automatically detect and exclude abnormal points. However, DPC still has some shortcomings: The DPC algorithm only considers the global distribution, and the clustering performance is poor for datasets with large cluster density differences. In addition, the point allocation strategy of DPC is likely to cause a Domino effect. Hence, this study proposes a DPC algorithm based on representative points and K-nearest neighbors (KNN), namely, RKNN-DPC. First, the KNN density is constructed, and the representative points are introduced to describe the global distribution of samples and propose a new local density. Then, the KNN information of samples is used to propose a weighted KNN allocation strategy to relieve the Domino effect. Finally, a comparative experiment is conducted with five clustering algorithms on artificial datasets and real datasets. The experimental results show that the RKNN-DPC algorithm can more accurately identify cluster centers and obtain better clustering results.

Key words: cluster analysis; density peaks clustering (DPC); representative point; K-nearest neighbors (KNN)

* 基金项目: 国家重点研发计划 (2020YFC2003502); 国家自然科学基金 (61876201); 重庆市自然科学基金 (cstc2019jcyj-cxttX0002, cstc2021ycjh-bgzxm0013); 重庆市教委重点合作项目 (HZ2021008)
收稿时间: 2021-12-21; 修改时间: 2022-04-18; 采用时间: 2022-06-15; jos 在线出版时间: 2023-03-08
CNKI 网络首发时间: 2023-03-10

聚类分析是一种重要的无监督学习方法[1], 根据样本间的相似性, 将数据划分为一定数量的类簇, 使得同一个类簇中的样本点间相似性较大, 不同类簇的样本间相似性较小[2]. 聚类分析作为数据挖掘和机器学习中的一个具有挑战性的问题, 它跨越了通信科学、计算机科学和生物科学, 在科学研究中起着重要的作用[3,4]. 在过去的几十年中, 诸多学者针对不同应用场景提出了许多聚类算法, 比如基于划分的聚类算法: K-means 算法[5]和 K-medoids 算法[6]等; 基于层次的聚类算法: CURE 算法[7]和 BIRCH 算法[8]等; 基于密度的聚类算法: DBSCAN 算法[9]和 OPTICS 算法[10]; 基于网格的聚类算法: STING 算法[11].

Rodriguez 等人[12]近年提出了密度峰值聚类算法 (density peaks clustering, DPC). DPC 是一种基于密度的聚类算法, 该算法首先计算样本的局部密度和相对距离, 根据样本的局部密度和相对距离构造决策图; 然后选择局部密度和相对距离均较大的点作为类簇中心; 最后, 将每个非中心点按局部密度降序分配到其高局部密度最近邻的类簇, 后文将高局部密度最近邻称为前置点. DPC 的优点包括: 算法简单高效, 不需要迭代优化目标函数, 能够识别任意形状的类簇等.

然而, DPC 存在一些不足: 1) 因为 DPC 的局部密度定义仅考虑了样本的局部分布, 所以当数据集的类簇密度差距较大时, 不能准确地表达数据样本点的密度, 从而导致选择错误的类簇中心. 2) 密度峰值聚类算法的分配策略容易导致“多米诺效应”, 即一个点分配错误, 产生连锁反应, 导致更多的点分配错误.

近年来, DPC 算法的研究主要集中于改进和扩展 DPC 算法. 关于改进 DPC 算法, 有学者将 K 近邻思想引入 DPC, 为改善 DPC 算法在多维数据集的聚类效果, Du 等人[13]提出了 DPC-KNN-PCA 算法, 将 K 近邻和主成分分析的思想引入到 DPC 中, 但算法的时间复杂度较大. Xie 等人[14]提出 FKNN-DPC 算法, 算法根据样本点到其 K 近邻的距离指数核计算局部密度. 然而, 该算法在高维数据集表现不理想, 并且未解决 DPC 算法中密度差距问题. 部分文献提出将 DPC 算法与其他模型结合. 例如: 陈玉洪等人[15]为了提高聚类效果, 减少错误分类将 ISS 模型引入 DPC 中作为检测策略, 提高了 DPC 的抗噪能力. 为了解决类簇重叠区域点聚类效果差的问题, Seyedi 等人[16]使用 K 近邻思想计算全局截断距离并基于图的标签传播以完成最终的聚类, 但该算法需要多次迭代才能获取较高的精度. Jiang 等人[17]提出了一种基于基尼系数计算截断距离并通过 KNN 找到中心点的方法, 对密度差距大的数据集有一定的聚类效果, 但在交叉缠绕数据上的准确度较差. Hou 等人[18]提出 RDDPC 算法, 提出直接下属和间接下属的概念, 以样本的直接下属数量作为样本的相对密度, 根据相对密度和相对距离选择类簇中心; 文献 [19] 中, Guan 等人提出 FHC-LDP 算法, 使用快速 KNN 的方法将算法复杂度降低至 $O(n\log(n))$, 使用从下到上的层次聚类替代 DPC 的类簇中心选择策略. 文献 [20] 使用流形距离来计算局部密度, 对于多密度的流形数据集有着较好的聚类效果, 同时构造了前置点的补偿机制, 使得算法更容易找到类簇中心. 文献 [21] 中, 孙林等人通过构造点的密度因子和距离因子改进了类簇中心的选择, 其次构造了 4 种相似度计算公式, 提出相似域及边界点概念完成最终聚类; 近年来, 研究人员将 DPC 算法应用于图像, 社区划分, 数据分析和采样等领域中, 并取得了较好的实验效果. 文献 [22] 中, Shi 等人将 DPC 算法应用到图像分割领域, 能够捕捉图像的固有结构并检测非球形团簇. 文献 [23] 中, Tu 等人在高光谱数据分析和处理中使用了 DPC 算法, 利用 DPC 算法在高光谱异常检测中的优势, 规避了影响检测性能的两个负面因素: 高斯分布的不成立假设和异常对背景统计的污染. 文献 [24] 中, Xu 等人扩展 DPC 算法应用到重叠社区检测, 通过线性拟合自动选择社区中心, 在现实网络和合成网络实验中均得到更好的结果. 文献 [25] 中, Xie 等人将 DPC 思想应用到欠采样技术中, 首先引入了 DPC 中局部密度和相对距离衡量数据中的每个实例的重要程度, 以此构造密度峰值序列, 再从序列中提取重要实例并自动确定类的最佳欠采样大小. 文献 [26] 中, Elaziz 等人将 DPC 算法应用到分割新冠肺炎的 CT 图像, 通过广义极值分布与 DPC 结合, 找到最佳的类簇中心数量, 在图像分割上取得了较好的效果.

针对 DPC 及其改进算法在确定类簇中心以及非中心点分配策略的不足, 本文提出了基于代表点和 K 近邻的密度峰值算法 (density peaks clustering algorithm based on representative points and K-nearest neighbors, RKNN-DPC). RKNN-DPC 算法的创新点包括: 1) 基于样本的 K 近邻信息提出了 K 近邻密度定义, K 近邻密度更能反映点近邻分布; 提出了代表点和代表值的定义来表示一个样本在类簇的重要程度, 不仅使样本的局部密度更层次化, 还能更准确地找到类簇中心; 为了解决类簇间密度差距的问题, 结合代表值和 K 近邻密度构造了 RKNN-DPC 算

法的局部密度. 2) 依据样本 K 近邻点的欧氏距离及所属类簇数量构造类簇权重, 确认待分配点的所属类簇; 在此基础上引入预备队列提出了改进的迭代分配策略, 针对不同的待分配点选择分配策略, 使得 RKNN-DPC 算法有效缓解了“多米诺效应”.

本文第 1 节介绍 DPC 聚类算法并分析其不足, 第 2 节详细介绍本文提出的 RKNN-DPC 算法, 第 3 节通过实验将本文 RKNN-DPC 与其他经典聚类算法进行了对比, 以验证 RKNN-DPC 的有效性, 第 4 节总结本文, 并指出了未来的研究方向.

1 DPC 聚类算法介绍与分析

1.1 DPC 聚类算法

DPC 算法基于两个假设: 类簇中心点的密度大于周围点的密度; 类簇中心点与更高密度点之间的距离相对较大. DPC 算法给每一个样本赋予局部密度和相对距离两个重要属性.

DPC 算法提供了 2 种局部密度 ρ 的计算方法, 其中公式 (1) 为截断距离法, 公式 (2) 为高斯核法.

定义 1[12]. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i,x_j\in X$, x_i 的局部密度 ρ_i 为:

$$\rho_i=\sum_{x_j\in X}\chi\left[d(x_i,x_j)-d_c\right],\ \chi(x)=\begin{cases}1 & x<0\\ 0 & x\geqslant 0\end{cases} \tag{1}$$

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

其中, $d(x_i,x_j)$ 表示 x_i 和 x_j 的欧氏距离, d_c 为截断距离.

使用截断距离计算局部密度是找以 x_i 为中心, d_c 为半径内的点的数量. 核距离法是通过 x_i 与其他点的高斯距离和来计算局部密度.文献 [12] 提出在较大数据集, 聚类使用截断距离法效果更好, 而较小数据集使用高斯核的效果更好.

定义 2[12]. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i,x_j\in X$, x_i 的相对距离 δ_i 为:

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

其中, $d(x_i,x_j)$ 表示 x_i 和 x_j 的欧氏距离. 当 x_i 的局部密度为数据集 X 中最大时, x_i 的相对距离 δ_i 为:

$$\delta_i=\max_{x_j\in X}\delta_j \tag{4}$$

相对距离 δ_i 的含义为局部密度大于 ρ_i 的点集中与 x_i 的最短距离, 对于局部密度最大的点, 相对距离则取两点的最大距离, 即认定局部密度最大的点为中心点. 一个样本有越大的局部密度和相对距离, 那么该样本是类簇中心的可能性就越大. 选取类簇中心可以通过以局部密度 ρ 为横轴, 相对距离 δ 为纵轴的决策图, 具有较小局部密度、较大的相对距离的点为异常点. 有较大局部密度、较大的相对距离的点为类簇的中心点, 因为决策图在确认类簇中心时, 没有定量分析而是通过人为确定, 由此给定了一个确认类簇中心的决策值. 决策值计算方式如定义 3.

定义 3[12]. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i\in X$, x_i 的决策值为:

$$\gamma_i=\rho_i\delta_i \tag{5}$$

其中, ρ_i 为 x_i 的局部密度, δ_i 为 x_i 的相对距离.

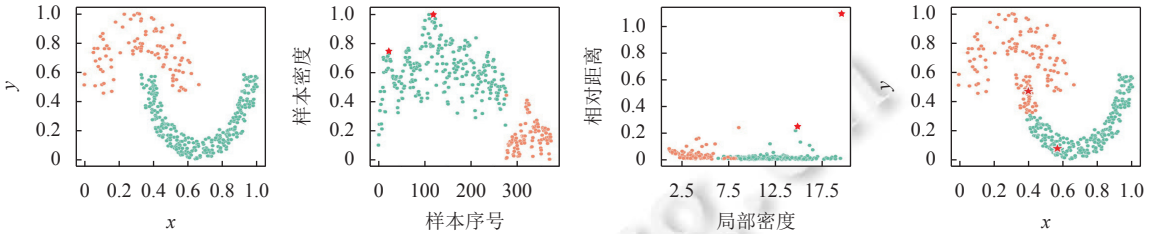
DPC 的聚类过程为: 根据定义 1 和定义 2 计算样本的局部密度 ρ 和相对距离 δ, 再根据定义 3 计算样本的决策值, 选取大的决策值为类簇中心. 找到类簇中心后, 将剩余点以局部密度由大到小的顺序分配, 按序分配到前置点的所属类簇.

1.2 DPC 分析

DPC 算法思想简单, 可以识别任意形状的类簇, 能直观找到簇数量. 但依然存在如下不足.

1) 对于类簇之间密度差距较大的数据集, DPC 算法的局部密度计算方式无法找到正确的类簇中心. 本文以 Jain 数据集[27]为例进行分析, 如图 1 所示, 其中红色五角星为 DPC 找到的类簇中心, 通过图 1(a) Jain 数据集的样

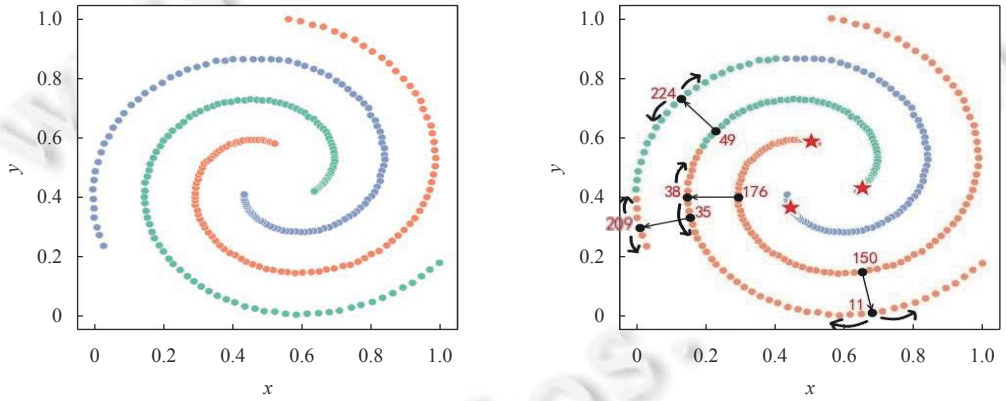
本分布可以看到绿色类簇的样本数量明显比橙色类簇样本多, 并且绿色类簇样本分布较橙色类簇样本更为紧密. 图 1(b) 为 DPC 在 $d_c = 0.05$ 时 Jain 数据集每个点的局部密度大小, 因为 DPC 计算局部密度时, 仅考虑截断距离内点的个数, 所以越密集的类簇局部密度越大, 从图 1(b) 也可以看到的绿色类簇样本的局部密度整体大于橙色类簇样本的局部密度. 图 1(c) 展示了 DPC 算法得到的决策图, 因为 DPC 选取局部密度和相对距离较大的点, 所以在决策图上选择了最偏右上的两个点作为类簇中心, 它们都属于绿色类簇. 图 1(d) 展示了 DPC 在 Jain 数据集的聚类结果, DPC 找到的两个类簇中心都在分布紧密的绿色类簇中, 导致最终的聚类效果差.



(a) Jain 数据集 (b) Jain 样本密度 (c) 决策图 (d) 聚类结果

图 1 DPC 密度差距问题

2) 若前置点的类簇中心找错, DPC 算法的分配策略可能会导致“多米诺效应”. 本文以 Spiral 数据集[28]为例进行分析. 图 2(a) 为 Spiral 数据集, 图 2(b) 是 DPC 在数据集上的聚类结果 ($d_c = 0.02$), 从图 2(b) 可以看到, DPC 在找到类簇中心后, 按局部密度由大到小将非中心点分配, 38 号点的前置点为 176 号点, 38 号点也就被分配到 176 号点所属的橘色类簇, 前置点为 38 号的点也将被分配到橘色类簇. 同样的原因 209 号点的前置点为 35 号点, 由于 35 号点的分配错误, 209 号点也分配到橙色类簇. DPC 仅考虑一个点所属类簇极容易因为一个点的分配错误导致后续点的分配错误, 这也是 DPC 算法聚类过程中点分配错误的主要原因.



(a) Spiral 数据集 (b) 聚类结果

图 2 DPC 的“多米诺效应”

2 RKNN-DPC 聚类算法

针对第 1.2 节中的分析, 本文提出了基于代表点与 K 近邻的 RKNN-DPC 算法. RKNN-DPC 聚类过程分为类簇中心的选择和非中心点的分配, 整体流程如图 3 所示. 在类簇中心的选择过程中, 基于 K 近邻的思想定义样本的 K 近邻密度, 并基于代表点的思想, 通过 K 近邻密度获取代表值构造新的局部密度, 解决了类簇密度差距问题. 对于非中心点的分配, 通过点的 K 近邻信息定义新的类簇权重公式, 进一步完成非中心点的分配, 有效缓解了“多米诺效应”.

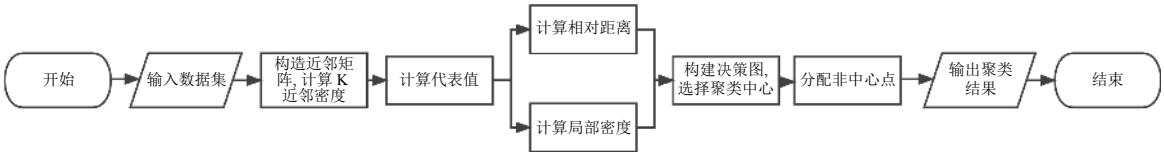


图 3 RKNN-DPC 流程图

2.1 类簇中心的选择策略

传统 DPC 算法是基于欧氏距离来计算局部密度和相对距离, 仅考虑欧氏距离忽略了每个类簇的分布情况. 为了更准确地估计样本局部密度本文引入 K 近邻和代表点思想以确认密度峰值.

定义 4. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i\in X$, x_i 的 K 近邻密度定义为:

$$\rho'(x_i)=\exp\left[-\sum_{x_j\in KNN(x_i)}d(x_i,x_j)\right] \tag{6}$$

其中, $d(\mathrm{x}_i,x_j)$ 表示点 x_i 和 x_j 之间的欧氏距离, $KNN(x_i)$ 表示点 x_i 的 K 近邻点集.

K 近邻密度是由点的 K 近邻信息计算, 而传统 DPC 考虑的是全局分布, 局部密度取决于截断距离 d_c, 并不能保证在截断距离内点的数量, 所以 K 近邻密度更能反映点的近邻分布. 但仅使用 K 近邻密度并不能解决类簇密度差距问题, 为了在每一个类簇都能找到一个密度峰值, 本文结合代表点的思想: 通过选择代表来解决随着人口的增长, 社会结构变得越来越复杂, 而引发的社会事务管理困难的问题. 代表比其他人更有权力. 权力来自他们的邻居, 反过来又作用于他们的邻居[29], 所以本文将具有局部最大密度的点作为代表点, 并从代表点来选择类簇中心.

定义 5. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i\in X$, x_i 的代表点定义为:

$$rep(x_i)=\underset{x_j\in KNN(x_i)}{\arg\max}\,\rho'(x_j) \tag{7}$$

为了体现一个点在类簇的重要性, 本文将代表点量化为代表值, 代表值的含义为点在数据集中作为其他点的代表点的次数, 代表值越大就表明该点为 K 近邻点集中越密集, 该值可以表明它在类簇的密集程度. 代表值定义如下.

定义 6. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i\in X$, x_i 的代表值为:

$$repv(x_i)=\sum_{x_j\in X}\psi(x_j),\ \psi(x_j)=\begin{cases}1, & rep(x_j)=x_i\\ 0, & rep(x_j)\neq x_i\end{cases} \tag{8}$$

K 近邻密度反映的是点的局部信息, 其在密度差距较大的数据集并不能正确表达点的密度, 所以使用代表值来获取每一个类簇的密度峰值, 如果单独使用代表值来代替局部密度可能会出现一个类簇有相同代表值的情况, 所以本文结合代表值和 K 近邻信息定义新的局部密度, 定义如下.

定义 7. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 对于 $\forall x_i\in X$, x_i 的局部密度为:

$$\rho(x_i)=\rho'(x_i)+repv(x_i) \tag{9}$$

其中, $\rho'(x_i)$ 表示为点 x_i 的 K 近邻密度, $repv(x_i)$ 表示 x_i 的代表值.

改进的局部密度由两个部分组成, 点的 K 近邻密度取决于它与 K 近邻点集的距离, 在越密集的区域, 该点的 K 近邻密度越大, 相反越稀疏 K 近邻密度越小, 这可以表明它在一个类簇的密集程度; 代表值取决于 K 近邻密度, DPC 算法假设类簇中心点的密度大于周围点的密度, 同样本文也基于这一假设, 那么一个类簇中心的 K 近邻密度就应该在其 K 近邻范围内最大, 所以类簇中心的代表值往往是等于 K, 因为代表值是基于 K 近邻, 所以即使在类簇差距较大的数据集, 也能从密度稀疏的类簇找到代表值为 K 的点, 这也是 RKNN-DPC 能够解决密度差距的核心.

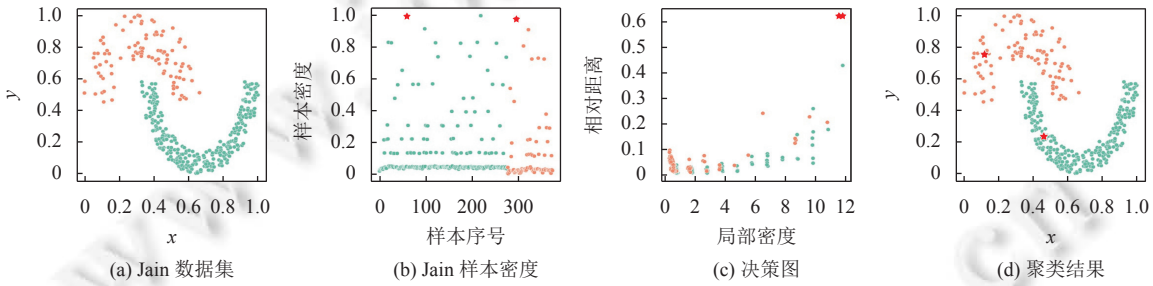
本文 RKNN-DPC 算法保留了 DPC 算法确定类簇中心的过程, 在改进的局部密度定义的基础上, RKNN-DPC 算法确定类簇中心的详细步骤如算法 1.

算法 1. RKNN-DPC 确定类簇中心.

输入: 数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 近邻数 K;
输出: 类簇中心 $C=\{c_1,c_2,\ldots,c_m\}$.

1. 数据归一化, 计算样本的距离矩阵.
2. 根据公式 (6) 计算每个样本的 K 近邻密度 $\rho'(x_i)$.
3. 根据公式 (7) 和公式 (8) 计算每个样本的代表点 $rep(x_i)$, 代表值 $repv(x_i)$.
4. 根据公式 (9) 和公式 (3) 计算每个样本的局部密度 ρ_i 和相对距离 δ_i.
5. 通过 ρ_i 和 δ_i 构建决策图或根据公式 (5) 计算 γ_i, 选取前 m 个点作为类簇中心 C.

为了说明 RKNN-DPC 选取类簇中心的有效性, 本节仍以 Jain 数据集为例进行分析, 其中红色五角星为 RKNN-DPC 找到的类簇中心, 如图 4 所示, 图 4(a) 为 Jain 数据集, 图 4(b) 为 Jain 数据集 RKNN-DPC 算法 ($K=7$) 计算的局部密度, 图 4(c) 为决策图, 图 4(d) 为聚类结果. 由图 4(b) 可以看到虽然上半类簇分布稀疏, 下半类簇分布紧密, 但代表值能确保每个类簇都能找到一个密度峰值, 相较图 1(b) RKNN-DPC 算法可以在每一个类簇中找到局部密度较大的点并且局部密度分布更有层次, 同时根据图 1(c) 与图 4(c) 决策图对比, 图 1(c) 找到的类簇中心都在分布紧密的下半类簇, 图 4(c) 因为每个类簇都有较大的局部密度, 所以找到的两个类簇中心分别在两个类簇中. 改进的局部密度定义不仅解决密度差距问题还更容易找到类簇中心. 通过图 4(d) 可看出 RKNN-DPC 算法分别在两个类簇找到中心点并且呈现了完全正确聚类结果, 与 DPC 相比, RKNN-DPC 解决了类簇间密度差距的问题.



(a) Jain 数据集 (b) Jain 样本密度 (c) 决策图 (d) 聚类结果

图 4 RKNN-DPC 密度差距问题

2.2 非中心点的分配策略

因为 DPC 算法的分配策略仅考虑前置点所属的类簇, 所以容易出现“多米诺效应”. 本文考虑多个点的类簇信息, 并基于样本的 K 近邻信息优化分配策略. 在分配点时, 首先找到点的 K 近邻分别属于的类簇, 再计算该点属于每一个类簇的权重, 并将其划分到权重最高的类簇. 类簇的权重定义如下.

定义 8. 给定数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 类簇中心集 $C=\{c_1,c_2,\ldots,c_m\}$, 对于 $\forall x_i\in X$, $\forall c\in C$, x_i 属于类簇 c 的权重为:

$$\lambda_{x_i}^{c}=\exp\left[-\sum_{x_j\in P(x_i,c)}\frac{d(x_i,x_j)}{|P(x_i,c)|}\right] \tag{10}$$

其中, 集合 $P(x_i,c)$ 表示 x_i 的 K 近邻中属于类簇 c 的点, $|P(x_i,c)|$ 表示集合 $P(x_i,c)$ 元素的个数, $d(x_i,x_j)$ 表示 x_i 和 x_j 的欧氏距离.

在分配的过程中, 点 x_i 计算每一个类簇的权重 $\{\lambda_{x_i}^{c1},\lambda_{x_i}^{c2},\ldots,\lambda_{x_i}^{cm}\}$, 并找到最大的类簇权重 $\max(\lambda_{x_i}^{c})$, 即点 x_i 的中心点为 c. 类簇权重作为非中心点所属类簇的决定性属性, 依据的是 K 近邻的距离, 本文未采用 DPC 只寻找前置点的做法, 因为 K 近邻中大多情况下已经包含前置点, 而考虑 K 近邻点集可以获取到待分配点周围的更多信息, 所以使用类簇权重能有更好的聚类效果.

基于类簇权重, 本文引入预备队列设计优化的非中心点分配策略, 分配思想为: 将离类簇中心的 K 近邻点分

配对应类簇; 然后根据预备队列的预备值 R 即判断样本 K 近邻是否有 R 个点被分配, 若符合条件则计算其类簇的权重完成分配, 若不符合则将该点放入预备队列等候分配, 直到所有数据点分配完成. 最后还有点未分配时, 则不判断条件直接计算类簇的权重. 分配策略的详细步骤如算法 2.

算法 2. 分配非中心点.

输入: 数据集 $X=\{x_1,x_2,\ldots,x_n\}$, 类簇中心 $C=\{c_1,c_2,\ldots,c_m\}$, 近邻数 K, 预备值 R;
输出: 聚类结果 S.

1. 将每个中心点 c 的 K 近邻点集的中心点置为 c, 并给定一个载入值 $L=K$;
2. **for** $x\in X_{\text{sort}}$ **do** // X_{sort} 为以局部密度由大到小排序的 X
3. **if** $num<R$ **then** // num 为 x 的 K 近邻点集已分配的数量
4. 将 x 放入预备队列 X_{ready}
5. **else**
6. 根据公式 (10) 计算 x 的最大类簇权重
7. $L=L-1$
8. **end if**
9. **if** $L=0$ **then**
10. **for** $x_r\in X_{\text{ready}}$ **do**
11. **if** $num<R$ **then**
12. **continue**
13. **else**
14. 根据公式 (10) 计算 x_r 的最大类簇权重, 将 x_r 从 X_{ready} 移除
15. $L=K$
16. **end if**
17. **end for**
18. **end if**
19. **end for**
20. **for** $x_r\in X_{\text{ready}}$ **do**
21. 根据公式 (10) 计算 x_r 的最大类簇权重, 将 x_r 从 X_{ready} 移除
22. **end for**

预备队列存放的是不具备分配条件的点, 通过预备值 R 判断点是否可以分配, 以此来提高聚类准确率. 分配策略通过迭代的方法即算法 2 的第 10–17 行, 保证在有点分配后预备队列中满足预备值 R 的点尽快分配, 可以使更多的点通过计算类簇权重进行分配.

在面对复杂的数据集下, 传统 DPC 采用的分配策略使非中心点的所属类簇完全取决于前置点, DPC 考虑的前置点可能是错误的, 从而容易导致连续性错误. 本文使用的类簇权重可以缓解这种情况, 类簇权重考虑的是待分配点的 K 近邻点集, 点集往往包含着 DPC 分配过程中的前置点, 而 DPC 分配时若分配错误前置点时没有纠错的措施, 而 RKNN-DPC 使用类簇权重在分配过程时基于欧氏距离和 K 近邻点集所属类簇, 即便前置点出现分配错误, 也可以通过其余的 K 近邻点修正. 所以本文提出的非中心点的分配策略在类簇相接较多的数据集下, 相比于 DPC 算法具有更好的聚类效果和泛化能力.

2.3 算法时间复杂度

设样本的数量为 N, RKNN-DPC 算法时间复杂度由算法 1 和算法 2 组成: 算法 1 花费的时间主要是计算样本之间的距离, 时间复杂度为 $O(N^2)$, 其次是计算局部密度 ρ、相对距离 δ 以及决策值 γ 时间复杂度均为 $O(N)$, 所以

算法 1 所需的时间复杂度为 $O(N^2+3N)$. 算法 2 第 2–15 行的最坏情况时间复杂度为 $O(N^2)$, 第 20–22 行将预备队列中剩余点进行分配时间最坏情况时间复杂度为 $O(N)$, 所以算法 2 所需的时间复杂度为 $O(N^2+N)$. 综上, RKNN-DPC 算法的时间复杂度近似于 $O(N^2)$.

3 实验与分析

为了验证 RKNN-DPC 算法的聚类效果, 实验使用人工数据集和真实数据集来测试其性能, 并以 DPC、FKNN-DPC、RDDPC、DBSCAN、K-means 这 5 种聚类方法作为对照组来对比实验结果. 表 1 和表 2 给出了实验中使用的人工数据集和真实数据集, 人工数据集包含了符合高斯分布的数据集、流形数据集和类簇高度重叠的数据集, 真实数据集是 UCI 的常用聚类数据集和图像数据集. 评估聚类算法的性能采用了常用的聚类评估指标: AMI[30]、ARI[31]、NMI[32]这 3 个指标来评估聚类效果, 3 个指标的上限都为 1, 指标大小与聚类效果成正相关. 实验环境为: Intel(R) Core(TM) i7-7700、8 GB 内存, PyCharm 2021.1.2.

表 1 人工数据集

数据集	文献	样本量	属性数量	类簇数量
Spiral	[28]	312	2	3
Jain	[27]	373	2	2
Zelnik3	[32]	266	2	3
Four-lines	[32]	512	2	4
DIM512	[33]	1 024	512	16
Aggregation	[34]	788	2	7
Flame	[35]	240	2	2
D31	[36]	3 100	2	31
Mouse	[37]	490	2	3
Xclara	[38]	3 000	2	3

表 2 真实数据集

数据集	文献	样本量	属性数量	类簇数量
Iris	[39]	150	4	3
Ionosphere	[39]	351	34	2
Wine	[39]	178	13	3
WDBC	[39]	569	30	2
Seeds	[39]	210	7	3
Dermatology	[39]	366	34	6
Parkinsons	[39]	195	23	2
Libras movement	[39]	360	91	15
Yeast	[39]	1 484	8	10
Banknote	[39]	1 372	4	2
Zoo	[39]	101	17	7
MFCCs	[39]	7 195	22	10
Vihecle	[39]	846	18	4
Abalone	[39]	4 177	8	3
Pima	[39]	768	8	2
Thyroid	[39]	215	6	3
Olivetti faces	[40]	400	64×64	40
COIL20	[16]	1 440	32×32	20

3.1 预处理与算法参数设置

为了消除指标之间的量纲对实验影响, 需要对数据标准化处理. 本文采用最大最小归一化, 最大最小归一化的计算公式如下:

$$x' = \frac{x - min}{max - min} \tag{11}$$

其中 x 表示单个数据的取值, min 是数据所在列的最小值, max 是数据所在列的最大值.

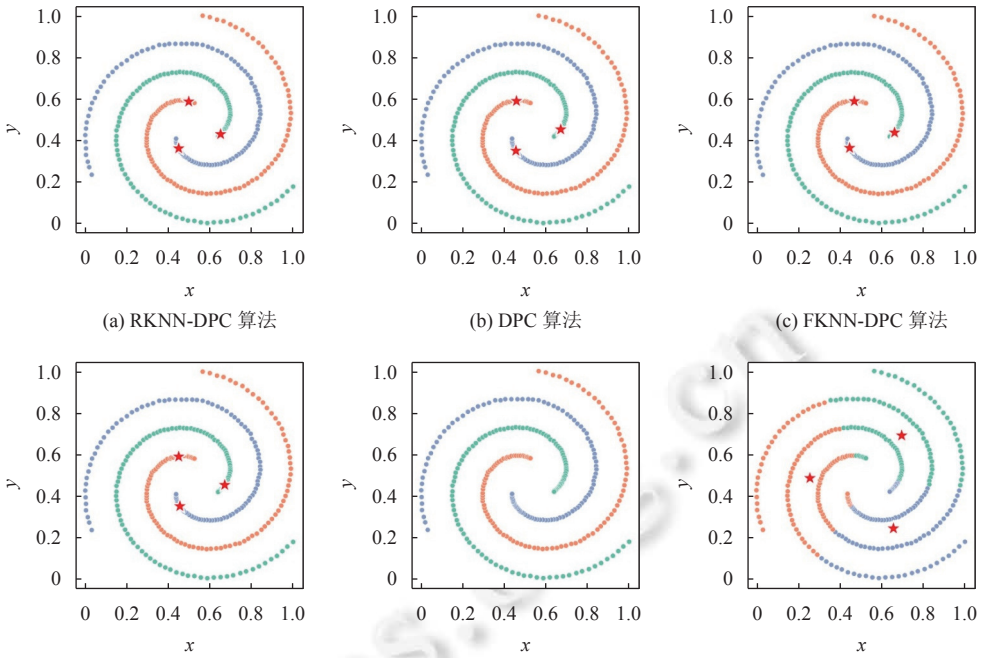
为了保证对比实验的有效性, RKNN-DPC、DPC、FKNN-DPC、RDDPC、DBSCAN、K-means 这 5 种聚类算法的聚类结果均在测试多组参数值后取最优结果, 其中 RKNN-DPC 的 K 近邻数取 [5, 51), 步长为 1, 预备值取 [2, K], 步长为 1; DPC 的截断距离 d_c 取 [0.05, 1), 步长为 0.01; FKNN-DPC 的 K 近邻数取 [5, 51), 步长为 1; RDDPC 的 K 近邻数 K 取 [5, 51), 步长为 1; DBSCAN 中两个参数设置半径和密度阈值取值范围分别为 (0, 1] 和 [1, 40], 步长分别为 0.01 和 1. 由于 K-means 算法随机选择类簇中心的特点, K-means 的实验结果为在每个数据集上运行 30 次的指标平均值.

3.2 人工数据集

本文选择了若干个常用的人工数据集, 比较各种聚类算法的聚类性能. 这些数据集的类簇数量不同, 样本不同, 可以模拟不同的情况. 表 3 列出了 6 种聚类算法在若干数据集上聚类指标 AMI、ARI、NMI, 最优结果以粗体显示, avg.表示 6 种算法同一数据集的指标平均值. Arg-表示算法在数据集上最优结果的参数. 图 5–图 14 为不同聚类算法在人工数据集上的聚类结果, 图中不同颜色的点被分为不同的类簇, 图中的红星表示类簇中心, 红叉表示算法分配错误的点, 灰色十字为 DBSCAN 算法确定的噪声点, 由于密度差距较大数据集的结果图可读性较差, 为了更好地分析分配错误点的原因, 实验中图 5–图 8 并未将分配错误的点标为红叉.

表 3 人工数据集实验结果

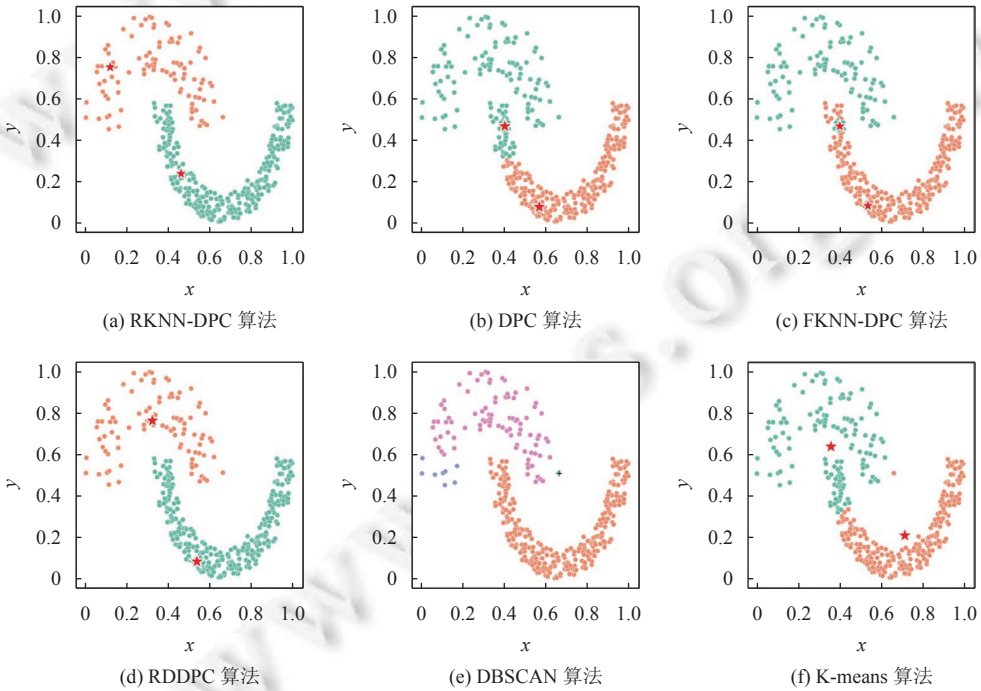
数据集	指标	RKNN-DPC	DPC	FKNN-DPC	RDDPC	DBSCAN	K-means	avg.
Spiral	AMI	**1**	**1**	**1**	**1**	**1**	−0.0054	0.8324
	ARI	**1**	**1**	**1**	**1**	**1**	−0.0058	0.8323
	NMI	**1**	**1**	**1**	**1**	**1**	0.0005	0.8334
	Arg-	2/8	0.06	5	5	0.04/2	3	—
Jain	AMI	**1**	0.7414	0.7895	**1**	0.9341	0.043	0.7513
	ARI	**1**	0.8224	0.8597	**1**	0.976	0.0021	0.77
	NMI	**1**	0.7421	0.79	**1**	0.9345	0.1824	0.7748
	Arg-	2/7	0.31	13	12	0.08/2	2	—
Zelnik3	AMI	**1**	0.7869	**1**	0.826	**1**	0.5843	0.8662
	ARI	**1**	0.721	**1**	0.7918	**1**	0.4874	0.8333
	NMI	**1**	0.7884	**1**	0.8272	**1**	0.5873	0.8671
	Arg-	8/11	0.06	7	13	0.07/11	3	—
Four-lines	AMI	**1**	**1**	**1**	0.8693	**1**	0.6849	0.9257
	ARI	**1**	**1**	**1**	0.8174	**1**	0.5206	0.8896
	NMI	**1**	**1**	**1**	0.8693	**1**	0.6849	0.9257
	Arg-	2/11	0.06	9	48	0.05/5	4	—
DIM512	AMI	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	ARI	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	NMI	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	Arg-	2/6	0.05	7	6	0.41/2	16	—
Aggregation	AMI	**1**	0.9923	0.9956	0.9923	0.9828	0.8307	0.9656
	ARI	**1**	0.9956	0.9978	0.9956	0.9881	0.721	0.9496
	NMI	**1**	0.9924	0.9956	0.9924	0.9828	0.833	0.966
	Arg-	2/17	0.09	9	26	0.06/11	7	—
Flame	AMI	0.9633	**1**	0.8752	**1**	0.8894	0.3969	0.8541
	ARI	0.9832	**1**	0.8748	**1**	0.9494	0.4534	0.8768
	NMI	0.9634	**1**	0.9338	**1**	0.8901	0.3987	0.8643
	Arg-	2/17	0.12	6	5	0.1/9	2	—
D31	AMI	0.9526	0.9547	0.9559	**0.9582**	0.9233	0.9496	0.949
	ARI	0.9311	0.9345	0.935	**0.9408**	0.8851	0.9058	0.922
	NMI	0.9547	0.9568	0.9579	**0.9601**	0.9269	0.952	0.9514
	Arg-	2/16	0.05	11	33	0.03/23	31	—
Mouse	AMI	0.9737	0.8626	**0.9768**	0.9246	0.8021	0.5947	0.8557
	ARI	**0.9868**	0.8728	0.9801	0.9414	0.7591	0.5185	0.8431
	NMI	**0.9738**	0.8632	0.9638	0.9249	0.8032	0.5963	0.8542
	Arg-	10/18	0.08	13	7	0.07/9	3	—
Xclara	AMI	**1**	0.9887	0.9951	0.9955	0.9736	0.9872	0.99
	ARI	**1**	0.9939	0.998	0.9979	0.9872	0.9928	0.9949
	NMI	**1**	0.9887	0.9951	0.9955	0.9736	0.9872	0.99
	Arg-	2/10	0.05	18	9	0.07/28	3	—



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

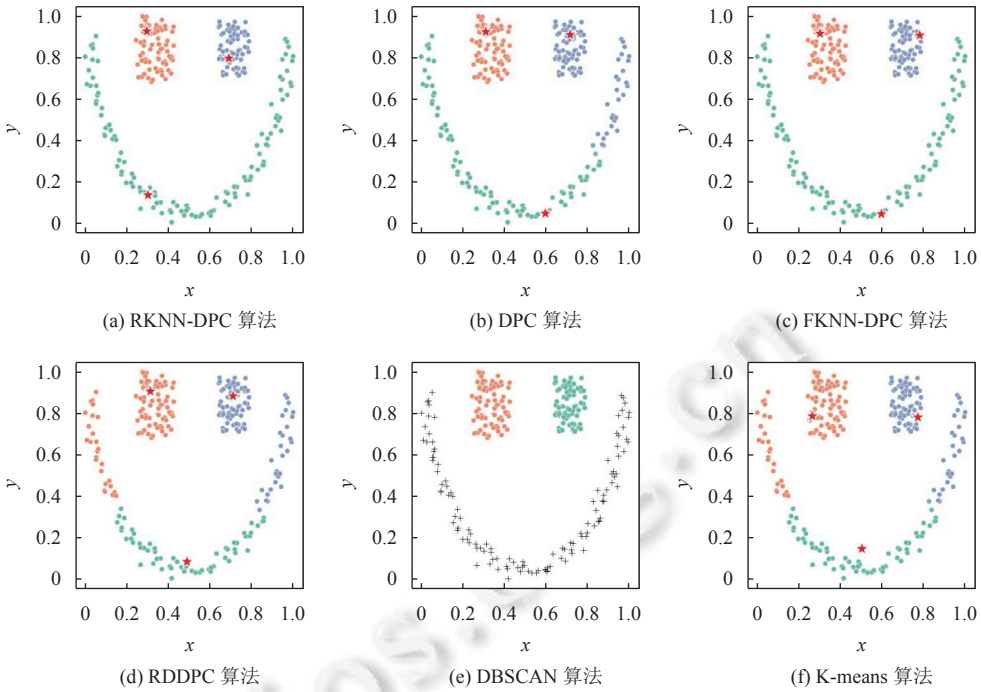
图 5 Spiral 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

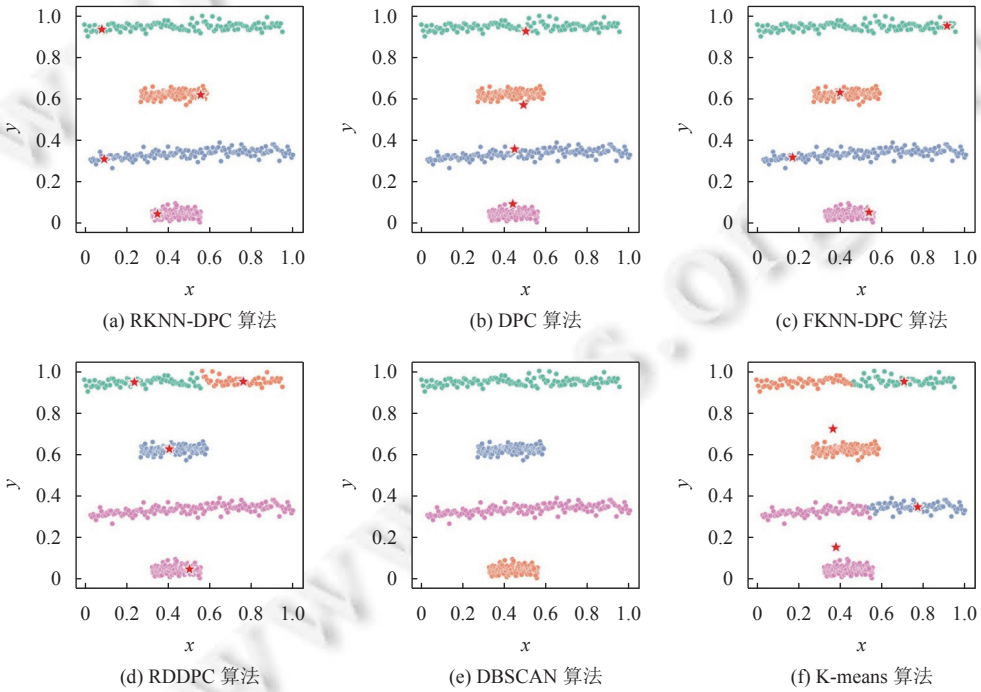
图 6 Jain 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

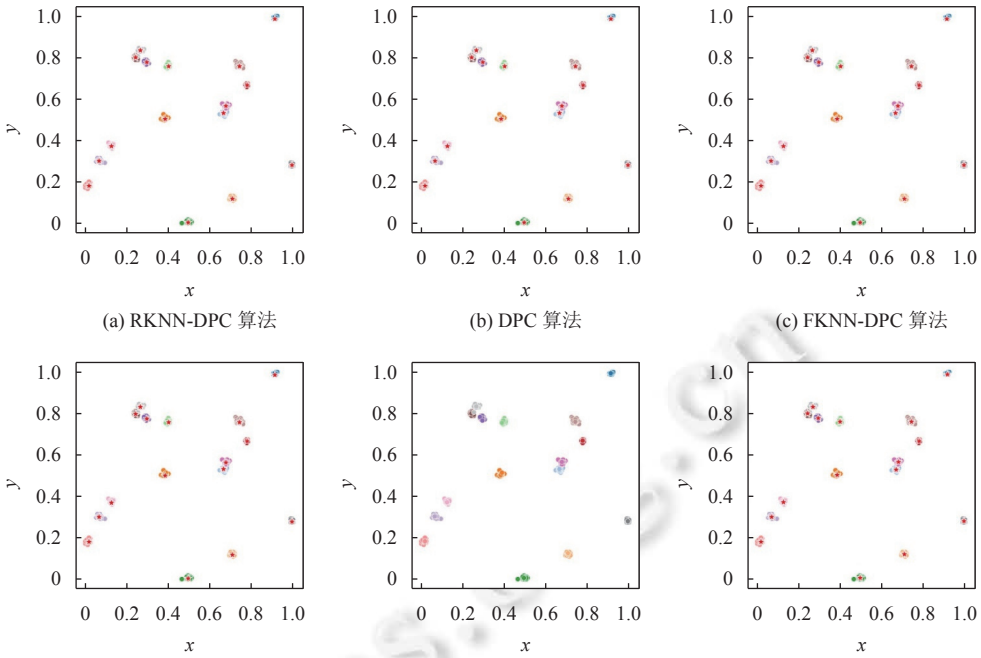
图 7 Zelnik3 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

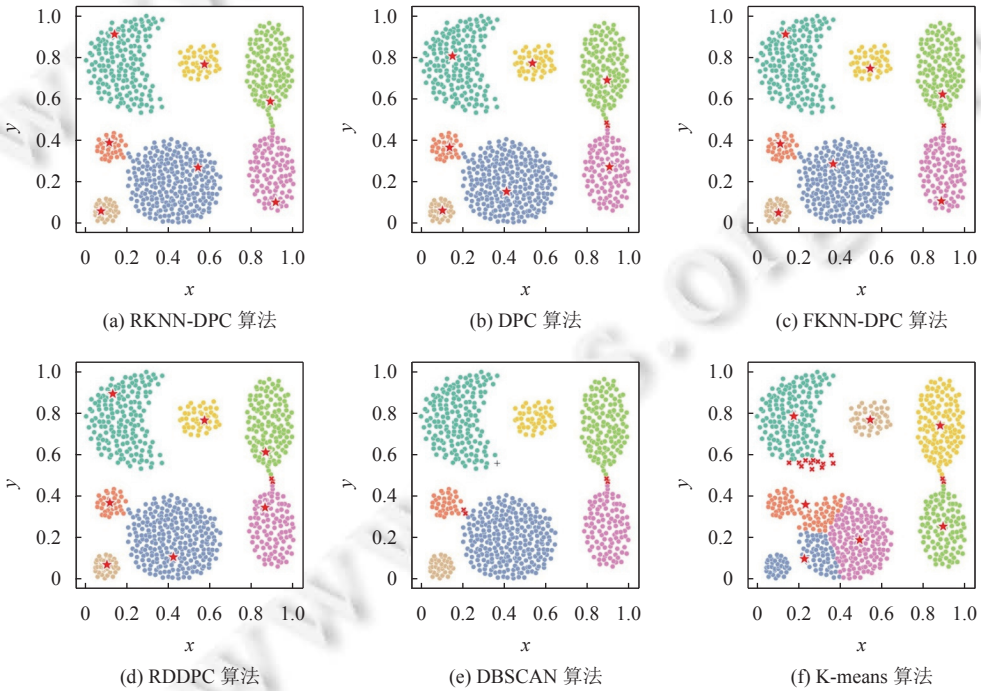
图 8 Four-lines 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

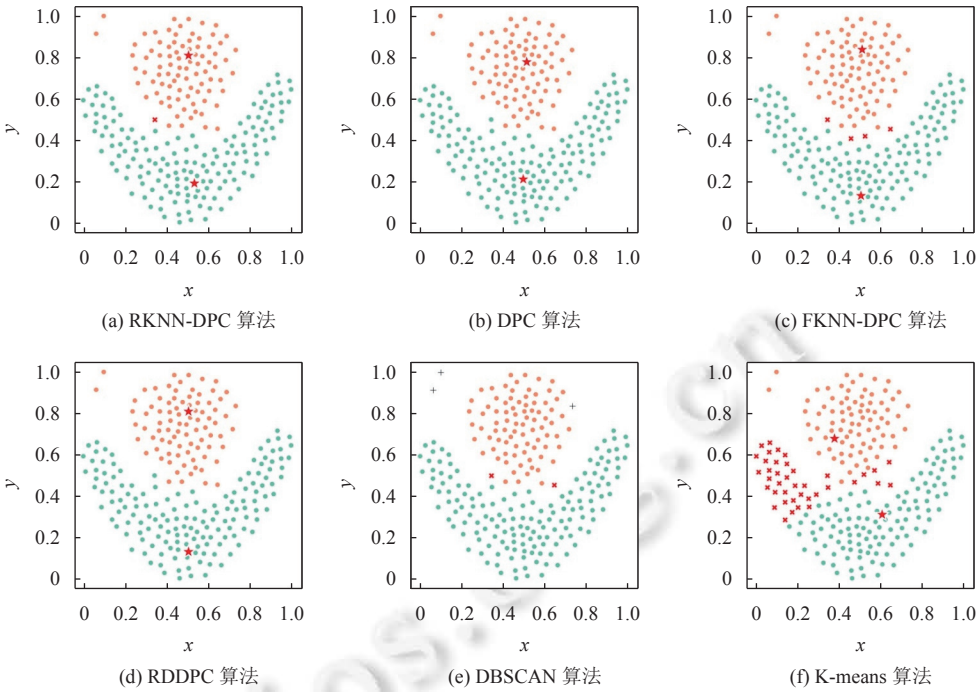
图 9 DIM512 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

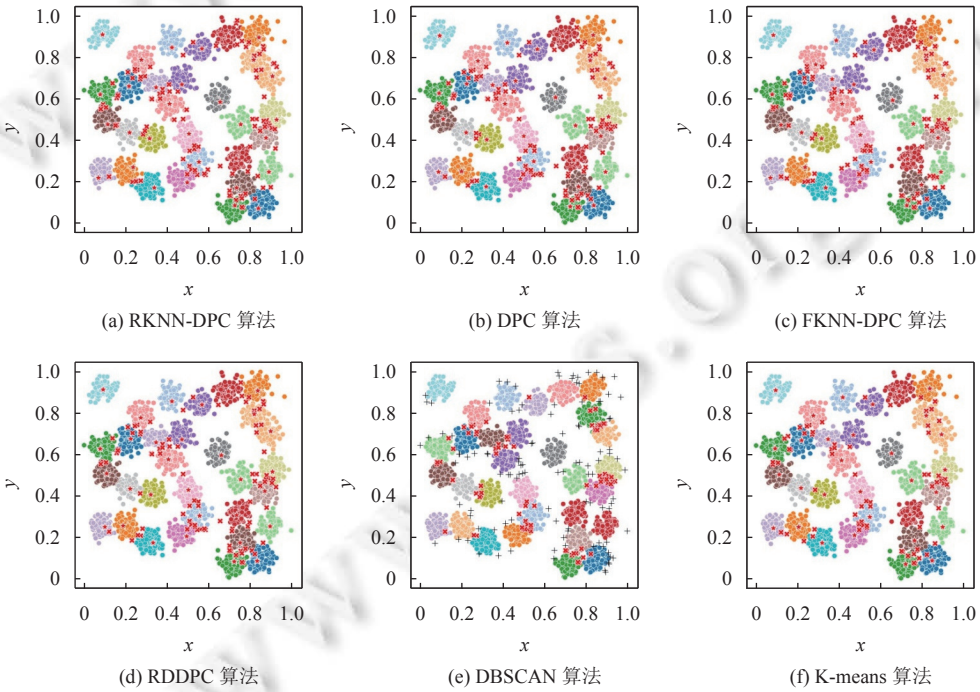
图 10 Aggregation 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

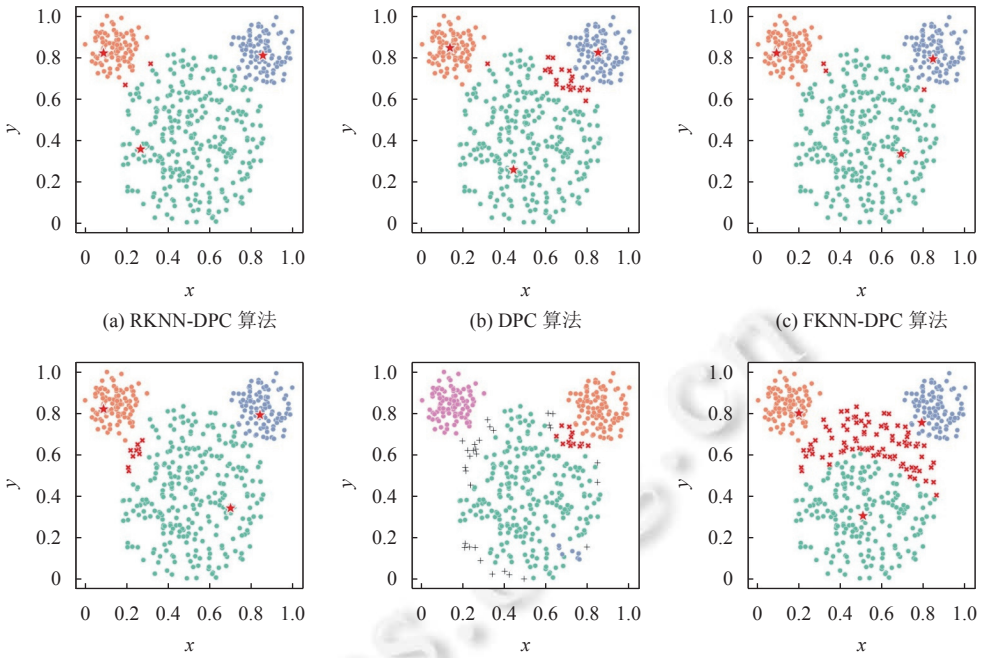
图 11 Flame 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

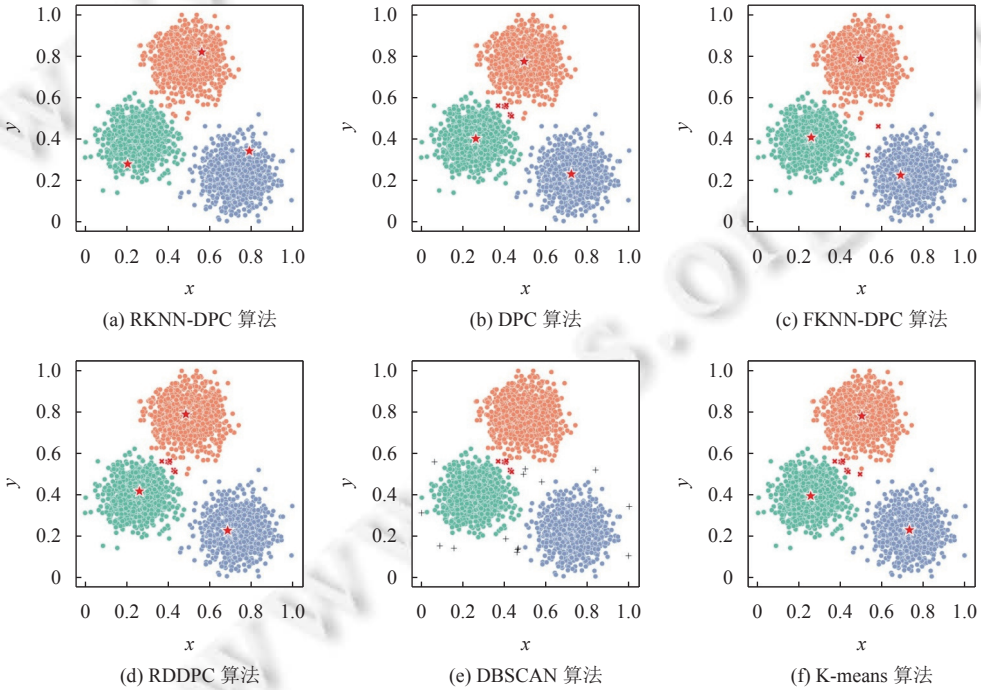
图 12 D31 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

图 13 Mouse 数据集上的聚类结果



(a) RKNN-DPC 算法 (b) DPC 算法 (c) FKNN-DPC 算法

(d) RDDPC 算法 (e) DBSCAN 算法 (f) K-means 算法

图 14 Xclara 的聚类结果

首先分析 Spiral、Jain、Zelnik3 这 3 个流形数据集, 其中 Jain、Zelnik3 的类簇密度有差距. 从图 5 Spiral 数据集的聚类结果可以看到 RKNN-DPC、DPC、FKNN-DPC、RDDPC、DBSCAN 的处理效果很好, 都获得正确的聚类结果, 但是本文提出的 RKNN-DPC 算法选择的类簇中心相较于传统 DPC 更加靠近类簇的末端, 使它指定不正确的类簇中心的概率更低. 而 K-means 算法的效果很差, 原因是 K-means 的分配策略无法处理不符合高斯分布的数据集; Jain 数据集作为分布清晰, 但类簇差距较大的数据集, 上部分的类簇密度明显小于下部分的类簇密度, 如图 6 所示, RKNN-DPC 与 RDDPC 对 Jain 数据集的聚类效果比较好. 而传统 DPC 和 FKNN-DPC 找到的密度中心都在下部分密度更高的类簇, 从而导致聚类效果不理想. 而 DBSCAN 算法对下部分密度较高类簇, 聚类效果较好, 但是将上部分簇分成了两个簇. 同样 K-means 对于这种流形数据集的效果不理想, 原因在于它的分配策略处理不了这类型数据集; 如图 7 所示, Zelnik3 数据集由一个较稀疏的环形类簇和两个分布在左上、右上较密集的类簇, RKNN-DPC、FKNN-DPC 和 DBSCAN 能够完全正确的聚类. DBSCAN 是将环形类簇当作噪音点处理. 在 DPC 算法的聚类过程中, 环形类簇的右上部分被分配到了右上的蓝色类簇. RDDPC 和 K-means 都将环形类簇的左上、右上部分分配到离它们较近的密集类簇.

接下来讨论 Four-lines、DIM512、Aggregation、Flame 数据集, 4 个数据集的分布较为清晰, 所以大多聚类算法可以得到较好的聚类效果. 图 8 可知 Four-lines 数据集分布清晰但仍有不均匀情况, RDDPC 并没有将第 3 个类簇识别, 而 K-means 则是将分布较广的类簇聚类错误; 由图 9 可知 6 种聚类算法都在 DIM512 数据集上识别类簇形状并且聚类完全正确; 其次图 10 是 Aggregation 数据集的聚类结果, 聚类结果显示 DPC、FKNN-DPC、RDDPC 算法对该数据集可以正确检测聚类数据集中的中心和识别每个类簇的形状, 但仍有少数的类簇相接的部分点存在分配错误. RKNN-DPC 更准确地分配了这些点, 由表 3 可以看出, RKNN-DPC 是唯一能完全正确聚类的算法. 对于 DBSCAN 算法虽然有点被标记为噪声, 但聚类的类簇形状基本正确. K-means 算法虽然找到了正确的类簇数量, 但是将一个簇分为了 3 个簇, 并在一个簇里选择了 3 个类簇中心; 如图 11 所示, 除了 K-means 的聚类结果较差以外, 其余聚类算法在 Flame 数据集上均可以找到类簇的中心, 识别类簇形状.

图 12–图 14 分别为 D31、Mouse、Xclara 的聚类结果, 这 3 个数据集都有较多点处于类簇相接区域. D31 是一个有 3 100 个样本, 31 个簇的数据集. 图 12 展示了 6 个算法的聚类结果, 6 个算法都能找到较好的类簇中心, 但是 6 个聚类算法对于簇与簇相连接的点的处理都不太理想, 还是会有部分点分配错误. DBSCAN 算法是将簇与簇相连接的部分点判定为噪声点; Mouse 数据集是由 3 个类簇组成, 左上和右上类簇较为密集. 由图 13 可知 RKNN-DPC、DPC、FKNN-DPC、RDDPC 算法均能找到正确的类簇中心和识别出 3 个类簇, 但是, 在分配类簇边界的样本时仍然发生了错误, 而 RKNN-DPC 对于类簇的边界样本的分配效果比其他算法更好, 因此获得了最好的聚类结果. 此外 DBSCAN 算法将一些点判定为噪声点, K-means 算法的效果较差, 下方簇的样本分配错误较多; 如图 14 所示, Xclara 数据集是由 3 个密集的类簇构成, 有较多的点处在类簇相连的区域, 6 种聚类算法都有较好的聚类效果, 但是处在类簇相连区域的点容易分配错误, 只有 RKNN-DPC 能够完全正确的聚类.

由图 5–图 14 可知, RKNN-DPC、DPC、FKNN-DPC、RDDPC 对于流形数据集聚类效果较好, DBSCAN 和 K-means 对流形数据集聚类效果较差. 但是只有 RKNN-DPC 算法对密度不均匀数据集的聚类结果较理想. 从表 3 可以看到, 在多个人工数据集上, 本文提出的 RKNN-DPC 在 AMI、ARI、NMI 这 3 个指标上多数是最优的.

3.3 真实数据集

实验使用了 16 个 UCI 数据集和 2 个图像数据集来测试 RKNN-DPC 聚类算法的性能. 这些数据集在样本数、特征数和簇的数量是不同的, 表 2 提供了每一个真实数据集的具体情况. 实验分别将 RKNN-DPC、DPC、FKNN-DPC、RDDPC、DBSCAN 和 K-means 算法在 18 个真实数据集进行了聚类. 实验结果见表 4, 最优的结果以粗体显示.

从表 4 可以看出, 本文的 RKNN-DPC 在 18 个真实数据集上的聚类结果总体优于其他聚类算法. 在一些 UCI 数据集上有较高的提升, 如 Ionosphere、WDBC 数据集. RKNN-DPC 在 Ionosphere 数据集中 AMI、ARI、NMI 较

DPC 分别提高了 0.09、0.19、0.07, 在 WDBC 数据集上 AMI、ARI、NMI 分别提高了 0.20、0.28、0.20, 原因在于 DPC 的参数截断距离 d_c 对小样本数据集敏感. DPC 在 UCI 数据集平均指标 AMI、ARI 和 NMI 分别为 0.4945、0.4771、0.5048, RKNN-DPC 在 UCI 数据集平均指标 AMI、ARI 和 NMI 分别为 0.5799、0.6014、0.5864, 在 3 个指标上均有一定的提升. 在 18 个数据集中上 K-means 在 2 个数据集上取到最好的结果, 是因为 K-means 适用于球形聚类, 许多真实数据集符合高斯分布, 近似为球形, 相反人工数据集是用来测试较为特殊的情况; RKNN-DPC 算法在 Olivetti faces 和 COIL20 两个图像数据集上也有所提升, 如图 15 和图 16 表示 RKNN-DPC、DPC、FKNN-DPC、RDDPC 和 K-means 这 5 种算法选择的正确类簇中心数量, RKNN-DPC 算法在 Olivetti faces 数据集中正确的选择了 36 个类簇中心, 相比于 DPC 选择了 28 个类簇中心有较大的提升, 同样在 COIL20 数据集中, DPC 只正确的选择了 13 个类簇中心, 而 RKNN-DPC 正确的选择了 17 个, 在类簇中心的选择和聚类性能上, RKNN-DPC 相比于 DPC 有着较大的提升并且优于其他的改进算法. RKNN-DPC 算法是基于代表值选择类簇中心, 改进的局部密度定义可以将类簇中点的密度更具层次化, 所以更容易选择具有代表性的类簇中心, 其次使用基于 K 近邻类簇权重的分配策略替代传统分配策略, 考虑了样本的周围分布情况, 所以 RKNN-DPC 算法在人工数据集和真实数据集上都能取到较好的聚类效果, 也说明其具有较好的泛化能力.

表 4　真实数据集实验结果

数据集	指标	RKNN-DPC	DPC	FKNN-DPC	RDDPC	DBSCAN	K-means	avg.
Iris	AMI	**0.9167**	0.8579	0.8665	0.8815	0.728	0.7421	0.8321
	ARI	**0.9392**	0.879	0.8823	0.9008	0.7388	0.7218	0.8436
	NMI	**0.9177**	0.8598	0.8682	0.883	0.7302	0.7454	0.834
	Arg-	10/16	0.18	20	5	0.82/2	3	—
Ionosphere	AMI	**0.3581**	0.2639	0.3192	0.1299	0.2578	0.1314	0.2433
	ARI	**0.4778**	0.2885	0.3926	0.2108	0.1698	0.1761	0.2859
	NMI	**0.3596**	0.2833	0.3209	0.1319	0.2609	0.1333	0.2483
	Arg-	7/19	0.31	14	14	0.87/4	2	—
Wine	AMI	0.8336	0.7232	**0.8468**	0.7407	0.5857	0.8135	0.7572
	ARI	0.8561	0.6989	**0.8665**	0.7269	0.5291	0.8368	0.7523
	NMI	0.8354	0.7261	**0.8484**	0.7434	0.5904	0.8154	0.7598
	Arg-	16/26	0.21	37	18	0.5/21	3	—
WDBC	AMI	**0.682**	0.4855	0.6411	0.4979	0.0067	0.6225	0.4892
	ARI	**0.786**	0.5016	0.6929	0.5174	0.0048	0.7301	0.5388
	NMI	**0.6824**	0.4863	0.6416	0.4986	0.0101	0.623	0.4903
	Arg-	7/14	0.16	6	10	1.175/2	2	—
Seed	AMI	**0.7798**	0.7213	0.7409	0.7319	0.5862	0.7421	0.717
	ARI	**0.7905**	0.7447	0.7684	0.7664	0.5291	0.7454	0.724
	NMI	**0.7817**	0.7237	0.7432	0.7342	0.5911	0.7218	0.7159
	Arg-	7/9	0.08	6	5	0.24/16	3	—
Dermatology	AMI	0.8523	0.8217	0.8632	0.8559	0.644	**0.8811**	0.8197
	ARI	0.8208	0.7281	0.7941	0.7754	0.4669	**0.8836**	0.7448
	NMI	0.8555	0.8255	**0.8662**	0.8592	0.6493	0.7425	0.7997
	Arg-	6/9	0.3	7	6	1.429/20	6	—
Parkinsons	AMI	**0.4144**	0.3473	0.2989	0.2731	0.1525	0.2318	0.2863
	ARI	**0.4727**	0.391	0.3769	0.2686	0.2074	0.0519	0.2947
	NMI	**0.4178**	0.3513	0.3032	0.2761	0.1597	0.235	0.2905
	Arg-	5/7	0.07	8	19	0.34/6	2	—
Libras Movement	AMI	**0.6023**	0.5661	0.5936	0.5978	0.3977	0.5476	0.5508
	ARI	**0.3859**	0.3526	0.3391	0.3721	0.1134	0.327	0.315
	NMI	**0.6547**	0.6218	0.6472	0.6503	0.4728	0.6069	0.6089
	Arg-	2/8	0.08	8	9	0.9/2	15	—

表 4 真实数据集实验结果 (续)

数据集	指标	RKNN-DPC	DPC	FKNN-DPC	RDDPC	DBSCAN	K-means	avg.
Yeast	AMI	0.2396	0.1997	0.1352	0.2585	0.0942	**0.2673**	0.199
	ARI	**0.173**	0.0987	0.0766	0.1424	0.0354	0.1524	0.113
	NMI	0.2517	0.2116	0.1521	0.2719	0.1008	**0.2782**	0.211
	Arg-	6/14	0.06	50	8	0.14/25	10	—
Banknote	AMI	**0.902**	0.879	0.4497	0.3891	0.728	0.0178	0.5609
	ARI	**0.9501**	0.9227	0.539	0.3684	0.6343	0.024	0.573
	NMI	**0.902**	0.8791	0.45	0.3895	0.7302	0.0183	0.5615
	Arg-	2/17	0.11	20	15	0.31/5	2	—
Zoo	AMI	**0.8584**	0.5811	0.7495	0.8055	0.1361	0.5669	0.6162
	ARI	**0.9082**	0.5103	0.7165	0.8086	0	0.4479	0.5652
	NMI	**0.8732**	0.6265	0.762	0.8445	0.1873	0.5791	0.6454
	Arg-	4/6	0.9	13	6	0.01/5	7	—
MFCCs	AMI	**0.7357**	0.6097	0.6001	0.6197	0.7119	0.6851	0.6603
	ARI	**0.8772**	0.7258	0.6459	0.6966	0.828	0.5792	0.7254
	NMI	**0.7366**	0.6107	0.6012	0.6213	0.7129	0.6861	0.6614
	Arg-	2/6	0.51	11	9	0.23/35	10	—
Vihecle	AMI	**0.1964**	0.1867	0.1923	0.1833	0.1734	0.0961	0.1713
	ARI	**0.1458**	0.1286	0.1335	0.1009	0.0606	0.075	0.1074
	NMI	**0.1996**	0.1904	0.1955	0.1872	0.1787	0.0996	0.1751
	Arg-	10/33	0.33	5	6	0.25/4	4	—
Abalone	AMI	**0.2004**	0.1798	0.1534	0.1561	0.0924	0.1254	0.1512
	ARI	**0.1783**	0.1596	0.0704	0.1158	0.0938	0.0669	0.1141
	NMI	**0.2035**	0.1802	0.1549	0.1564	0.1438	0.1265	0.1608
	Arg-	10/33	0.08	9	8	0.2/2	3	—
Pima	AMI	**0.0519**	0.0358	0.0515	0.0269	0.0175	0.0507	0.0390
	ARI	**0.1093**	0.078	0.0804	0.0795	0.051	0.1023	0.0834
	NMI	**0.0529**	0.0371	0.0529	0.029	0.0187	0.0517	0.0403
	Arg-	12/21	0.12	47	2	0.3/13	2	—
Thyroid	AMI	**0.6549**	0.4545	0.4538	0.2913	0.6161	0.5908	0.5102
	ARI	0.7526	0.4257	0.3861	0.2895	**0.7666**	0.6282	0.5414
	NMI	**0.6592**	0.4635	0.462	0.3044	0.6186	0.5966	0.5173
	Arg-	2/6	0.3	22	5	0.1/8	3	—
Olivetti faces	AMI	**0.7989**	0.7513	0.7203	0.7811	0.5819	0.6174	0.7084
	ARI	**0.6732**	0.5508	0.5396	0.5855	0.1431	0.4126	0.4841
	NMI	**0.8807**	0.847	0.8313	0.866	0.7323	0.7648	0.8203
	Arg-	4/5	0.4	6	44	0.5/2	40	—
COIL20	AMI	**0.8909**	0.7928	0.8376	0.8656	0.6962	0.7842	0.8112
	ARI	**0.7634**	0.4935	0.4523	0.6576	0.2105	0.6189	0.5327
	NMI	**0.896**	0.8027	0.8457	0.8718	0.7133	0.7942	0.8206
	Arg-	14/20	0.57	10	47	0.3/3	20	—

图 15 Olivetti faces 类簇中心选择

图 16　COIL20 类簇中心选择

4 结　论

本文针对 DPC 和改进的 DPC 存在的问题, 提出了基于代表点与 K 近邻的密度峰值聚类算法. 聚类过程主要为: 选择类簇中心和分配非中心点. 对于类簇中心的选择, 基于 K 近邻思想构造 K 近邻密度, 提出代表点、代表值定义, 在此基础上构造改进的局部密度计算方式, 解决了类簇差距问题; 对于非中心点的分配, 构造类簇权重公式, 考虑每个点的类簇可能性, 找到其最佳的类簇权重以完成最终聚类, 缓解了“多米诺效应”; 最后本文在人工数据集和真实数据集上进行了实验, 实验证明 RKNN-DPC 算法可以适用于不同大小、任意形状的数据集并且优于 DPC 的改进算法和常见聚类算法. 此外, 下一步的研究工作是尝试将该算法的思想应用到社区划分中, 研究基于改进的 DPC 的社区划分方法.

References:

[1] Zhang YP, Zhou J, Deng ZH, Chung FL, Jiang YZ, Hang WL, Wang ST. Multi-view fuzzy clustering approach based on medoid invariant constraint. Ruan Jian Xue Bao/Journal of Software, 2019, 30(2): 282–301 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5625.htm [doi: 10.13328/j.cnki.jos.005625]

[2] Wang CD, Lai JH, Huang D, Zheng WS. SVStream: A support vector-based algorithm for clustering data streams. IEEE Trans. on Knowledge and Data Engineering, 2013, 25(6): 1410–1424. [doi: 10.1109/TKDE.2011.263]

[3] Xu DK, Tian YJ. A comprehensive survey of clustering algorithms. Annals of Data Science, 2015, 2(2): 165–193. [doi: 10.1007/s40745-015-0040-1]

[4] Dai YY, Zhang QH, Zhi XC. Density peaks clustering algorithm by combining relative density with nearest neighbor relationship. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2021, 33(5): 791–805 (in Chinese with English abstract). [doi: 10.3979/j.issn.1673-825X.202105220174]

[5] MacQueen JB. Some methods for classification and analysis of multivariate observations. In: Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability. Berkeley: University of California Press, 1967. 281–297.

[6] Park HS, Jun CH. A simple and fast algorithm for K-medoids clustering. Expert Systems with Applications, 2009, 36(2): 3336–3341. [doi: 10.1016/j.eswa.2008.01.039]

[7] Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases. In: Proc. of the 1998 ACM SIGMOD Int’l Conf. on Management of Data. Seattle: ACM, 1998. 73–84. [doi: 10.1145/276304.276312]

[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, Xu XW. 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. Portland: AAAI Press, 1996. 226–231.

[10] Ankerst M, Breunig MM, Kriegel HP, Sander J. OPTICS: Ordering points to identify the clustering structure. ACM SIGMOD Record, 1999, 28(2): 49–60. [doi: 10.1145/304181.304187]

[11] Wei W, Yang J, Muntz RR. STING: A statistical information grid approach to spatial data mining. In: Proc. of the 23rd Int’l Conf. on Very Large Data Bases. San Francisco: Morgan Kaufmann Publishers Inc., 1997. 186–195.

[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] 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]

[14] Xie JY, Gao HC, Xie WX, Liu XH, Grant PW. 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]

[15] Chen YH, Zhang QH, Yang J. Density peak clustering algorithm based on interval shadowed sets. Pattern Recognition and Artificial Intelligence, 2019, 32(6): 531–544 (in Chinese with English abstract). [doi: 10.16451/j.cnki.issn1003-6059.201906006]

[16] Seyedi SA, Lotfi A, Moradi P, Qader NN. 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]

[17] Jiang D, Zang WK, Sun R, Wang ZH, Liu XY. Adaptive density peaks clustering based on K-nearest neighbor and Gini coefficient. IEEE Access, 2020, 8: 113900–113917. [doi: 10.1109/ACCESS.2020.3003057]

[18] 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]

[19] Guan JY, Li S, He XX, Zhu JH, Chen JJ. 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] Tao XM, Guo WJ, Ren C, Li Q, He Q, Liu R, Zou JR. Density peak clustering using global and local consistency adjustable manifold distance. Information Sciences, 2021, 577: 769–804. [doi: 10.1016/j.ins.2021.08.036]

[21] 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 with English abstract). http://www.jos.org.cn/1000-9825/6462.htm [doi: 10.13328/j.cnki.jos.006462]

[22] Shi Y, Chen ZS, Qi ZQ, Meng F, Cui LM. A novel clustering-based image segmentation via density peaks algorithm with mid-level feature. Neural Computing and Applications, 2017, 28(S1): 29–39. [doi: 10.1007/s00521-016-2300-1]

[23] Tu B, Yang XC, Li NY, Zhou GL, He DB. Hyperspectral anomaly detection via density peak clustering. Pattern Recognition Letters, 2020, 129: 144–149.

[24] Xu ML, Li YH, Li RX, Zou FH, Gu XW. EADP: An extended adaptive density peaks clustering for overlapping community detection in social networks. Neurocomputing, 2019, 337: 287–302. [doi: 10.1016/j.neucom.2019.01.074]

[25] Xie XY, Liu HW, Zeng SZ, Lin LB, Li W. A novel progressively undersampling method based on the density peaks sequence for imbalanced data. Knowledge-based Systems, 2021, 213: 106689. [doi: 10.1016/j.knosys.2020.106689]

[26] Elaziz MA, Al-Qaness MAA, Zaid EOA, Lu SF, Lbrahim RA, Ewees AA. Automatic clustering method to segment COVID-19 CT images. PLoS One, 2021, 16(1): e0244416. [doi: 10.1371/journal.pone.0244416]

[27] Jain AK, Law MHC. Data clustering: A user's dilemma. In: Proc. of the 1st Int'l Conf. on Pattern Recognition and Machine Intelligence. Kolkata: Springer, 2005. 1–10. [doi: 10.1007/11590316_1]

[28] Chang H, Yeung DY. Robust path-based spectral clustering. Pattern Recognition, 2008, 41(1): 191–203. [doi: 10.1016/j.patcog.2007.04.010]

[29] Cheng DD, Zhu QS, Huang JL, Yang LJ, Wu QW. Natural neighbor-based clustering algorithm with local representatives. Knowledge-based Systems, 2017, 123: 238–253. [doi: 10.1016/j.knosys.2017.02.027]

[30] Vinh NX, Epps J, Bailey J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. The Journal of Machine Learning Research, 2010, 11: 2837–2854.

[31] Chen WY, Song YQ, Bai HJ, Lin CJ, Chang EY. Parallel spectral clustering in distributed systems. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2011, 33(3): 568–586. [doi: 10.1109/TPAMI.2010.88]

[32] Zelnik-Manor L, Perona P. Self-tuning spectral clustering. In: Proc. of the 17th Int'l Conf. on Neural Information Processing Systems. Vancouver British: MIT Press, 2004. 1601–1608.

[33] Franti P, Virmajoki O, Hautamaki V. Fast agglomerative clustering using a K-nearest neighbor graph. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2006, 28(11): 1875–1881. [doi: 10.1109/TPAMI.2006.227]

[34] Gionis A, Mannila H, Tsaparas P. Clustering aggregation. ACM Trans. on Knowledge Discovery from Data, 2007, 1(1): 4–es. [doi: 10.1145/1217299.1217303]

[35] Fu LM, Medico E. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. BMC Bioinformatics, 2007, 8: 3. [doi: 10.1186/1471-2105-8-3]

[36] Veenman CJ, Reinders MJT, Backer E. A maximum variance cluster algorithm. IEEE Trans. on Pattern Analysis and Machine

Intelligence, 2002, 24(9): 1273–1280. [doi: 10.1109/TPAMI.2002.1033218]

[37] Xia CY, Hsu W, Lee ML, Ooi BC. Border: Efficient computation of boundary points. IEEE Trans. on Knowledge and Data Engineering, 2006, 18(3): 289–303. [doi: 10.1109/TKDE.2006.38]

[38] Jia HJ, Ding SF, Du MJ. Self-tuning *p*-spectral clustering based on shared nearest neighbors. Cognitive Computation, 2015, 7(5): 622–632. [doi: 10.1007/s12559-015-9331-2]

[39] Bache K, Lichman M. UCI machine learning repository. 2020. http://archive.ics.uci.edu/ml

[40] Samaria FS, Harter AC. Parameterisation of a stochastic model for human face identification. In: Proc. of the 1994 IEEE Workshop on Applications of Computer Vision. Sarasota: IEEE, 1994. 138–142. [doi: 10.1109/ACV.1994.341300]

附中文参考文献:

[1] 张远鹏, 周洁, 邓赵红, 钟富礼, 蒋亦樟, 杭文龙, 王士同. 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. http://www.jos.org.cn/1000-9825/5625.htm [doi: 10.13328/j.cnki.jos.005625]

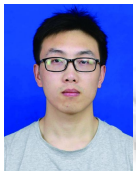
[4] 代永杨, 张清华, 支学超. 融合相对密度与近邻关系的密度峰值聚类算法. 重庆邮电大学学报(自然科学版), 2021, 33(5): 791–805. [doi: 10.3979/j.issn.1673-825X.202105220174]

[15] 陈玉洪, 张清华, 杨洁. 基于区间阴影集的密度峰值聚类算法. 模式识别与人工智能, 2019, 32(6): 531–544. [doi: 10.16451/j.cnki.issn1003-6059.201906006]

[21] 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 [doi: 10.13328/j.cnki.jos.006462]

张清华(1974－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为粗糙集, 模糊集, 粒计算, 不确定性信息处理.

代永杨(1996－), 男, 硕士生, 主要研究领域为粗糙集, 机器学习, 不确定性信息处理.



周靖鹏(1999－), 男, 硕士生, 主要研究领域为粗糙集, 机器学习, 数据挖掘.

王国胤(1970－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为粗糙集, 粒计算, 数据挖掘.