控制与决策
Control and Decision

基于双向代表点和相互K近邻的密度峰值聚类算法

任春华, 李朝荣, 余洋

引用本文:
任春华, 李朝荣, 余洋. 基于双向代表点和相互K近邻的密度峰值聚类算法[J]. 控制与决策, 2025, 40(8): 2491-2502.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2024.0959

基于双向代表点和相互 K 近邻的密度峰值聚类算法

任春华[1]，李朝荣[1†]，余 洋[2]

(1. 宜宾学院 计算机科学与技术学院，四川 宜宾 644000；2. 西南石油大学 计算机与软件学院，成都 610500)

摘 要: 密度峰值聚类算法 (DPC) 能够识别任意形状的类簇, 但存在两大明显不足: 一是在密度分布不均的数据集中不能正确发现稀疏集群的聚类中心; 二是剩余点分配策略容易引起连锁反应导致数据点归类错误. 为此, 提出一种基于双向代表点 (BRP) 和相互K近邻 (MKNN) 的密度峰值聚类算法, 称为 BRPMK-DPC. 首先, 设计一种基于正向K近邻代表点和逆向逆K近邻代表点的局部密度计算方法, 好处是可以在密度分布不均的数据集中高效识别正确的聚类中心; 其次, 提出一种相互K近邻的剩余点分配方法, 在分配过程中具有自适应性, 避免衍生类 DPC 算法采用固定K值带来的劣势; 最后, 在人工合成数据集和真实数据集上进行测试, 实验结果表明所提出的算法不仅能够高效识别密度不均集群的聚类中心, 而且在大部分数据集上的聚类性能优于其他 7 种对比算法.

关键词: 双向代表点；K近邻；逆K近邻；相互K近邻；剩余点分配；密度峰值聚类

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2024.0959

引用格式: 任春华, 李朝荣, 余洋. 基于双向代表点和相互 K 近邻的密度峰值聚类算法 [J]. 控制与决策, 2025, 40(8): 2491-2502.

Density peak clustering algorithm based on bidirectional representative points and mutual K-nearest neighbors

REN Chun-hua[1]，*LI Chao-rong*[1†]，*YU Yang*[2]

(1. School of Computer Science and Technology，Yibin University，Yibin 644000，China；2. School of Computer and Software，Southwest Petroleum University，Chengdu 610500，China)

Abstract: The density peak clustering (DPC) algorithm is capable of identifying clusters of any shape, but there are two obvious shortcomings. First, it struggles to accurately identify the cluster centers of sparse clusters in datasets with uneven density distribution. Second, its remaining points assignment strategy can easily lead to a chain reaction, resulting in incorrect data point classification. Therefore, this paper introduces a density peak clustering algorithm called BRPMK-DPC based on bidirectional representative points (BRP) and mutual K-nearest neighbors (MKNN). The algorithm initially develop a method to calculate local density using forward K-nearest neighbor representative points and backward inverse K-nearest neighbor representative points. This method efficiently identifies the correct cluster centers in datasets with uneven density distribution. Additionally, a method for assigning remaining points based on mutual K-nearest neighbors is proposed. This method offers adaptability in the assignment process, avoiding the drawback of using fixed K-values in derivative DPC algorithms. Finally, the BRPMK-DPC algorithm is tested on artificially synthesized datasets and real datasets. The experimental results demonstrate that the proposed algorithm not only efficiently identifies cluster centers with uneven density but also outperform the other seven compared algorithms on most datasets.

Keywords: bidirectional representative points； K-nearest neighbors； inverse K-nearest neighbors； mutual K-nearest neighbors；remaining points assignment；density peak clustering

0 引 言

随着数字化时代的到来, 社会数据爆发式增长. 为了更好应对日益增长的数据挑战, 数据挖掘技术已经渗透到各行各业, 帮助人们理解数据的同时并作出明智的决策. 聚类技术是数据挖掘中的一种无监督学习方法, 它主要针对大量无标签的数据, 利用

收稿日期: 2024-08-12；**录用日期:** 2025-01-09.

基金项目: 宜宾学院高层次人才启航计划项目 (2023QH02)；四川省科技计划项目 (2024ZYD0089).

责任编委: 刘德荣.

†通信作者. E-mail: lichaorong88@163.com.

本文附带电子附录文件,可登录本刊官网该文“资源附件”区自行下载阅览.

数据之间的相似性进行类别归属, 使得相同类别中的数据特征高度相似, 不同类别间的数据相似度较低[1]. 根据不同的聚类原理, 聚类算法可以分为基于划分的聚类[2]、基于层次的聚类[3]、基于密度的聚类[4]、基于网格的聚类[5]、基于模型的聚类[6] 以及基于图的聚类[7]. 不同的聚类算法已经在客户细分[8]、医疗图像[9]、产品推荐[10]、社交网络[11] 等领域取得了显著成效. 到目前为止, 产生了诸多经典的聚类算法, 如K-means[12]、DBSCAN[13]、SC[14]、EM[15] 等, 同时也衍生了大量改进或优化的聚类方法, 但它们并不能很好地处理任意形状、任意尺寸和任意密度的数据集群.

2014 年, Rodriguez 等[16] 发表了一篇关于密度峰值聚类算法 (DPC) 的论文, 成功吸引了聚类技术研究者的高度关注. DPC 是一种基于密度理论的聚类算法, 其聚类原理基于两个假设: 一是聚类中心的局部密度大于其周围样本点的局部密度, 该假设侧重于局部密度的比较; 二是不同的聚类中心相距甚远. DPC 算法仅需要一个参数, 能够快速找到聚类中心并进行样本点归类, 且在任意形状的数据集上具有较好的聚类性能.

虽然 DPC 聚类过程简单且高效, 但存在两个明显的问题: 一是在密度分布不均的数据集中不能正确发现稀疏集群的聚类中心; 二是单一的剩余点分配策略容易引起数据点归类错误的连锁反应. 针对 DPC 存在的不足, 不少研究者从局部密度和剩余点分配两个角度对 DPC 进行改进, 并成功产生一批研究成果. 其中, K近邻[17]、模糊K近邻[18]、共享K近邻[19]、分层K近邻[20]、反向K近邻[21] 的方法被用于 DPC 的聚类过程, 虽然衍生的 DPC 算法改善了聚类性能, 但仍存在一些问题: 一是稀疏集群的聚类中心有待精确识别; 二是剩余点分配过程中采用固定K值不能较好地适应样本点的局部分布.

综上, 本文提出一种基于双向代表点和相互K近邻的密度峰值聚类算法 (BRPMK-DPC). 该算法主要有两大贡献: 1) 设计一种双向代表点的局部密度计算方法, 该方法结合正向K近邻代表点和逆向逆K近邻代表点能够更好地识别流形数据集中稀疏集群的聚类中心; 2) 提出一种相互K近邻的剩余点分配方法, 该方法具有自适应性且能弥补固定K值类分配方法带来的不足, 有效改善聚类性能.

本文剩余工作安排如下: 第 1 节介绍 DPC 原始算法; 第 2 节分析 DPC 研究的相关工作; 第 3 节详细介绍本文提出的 BRPMK-DPC 算法; 第 4 节开展实验分析, 将 BRPMK-DPC 算法与其他 7 个经典算法进行对比分析; 第 5 节进行全文总结.

1 DPC 算法介绍

首先, DPC 定义了两种局部密度计算方法, 方法 1 利用式 (1) 的分段函数计算局部密度, 方法 2 利用式 (2) 的高斯核函数计算局部密度.

$$\begin{cases}\rho(x_i)=\displaystyle\sum_{x_j}\chi(d(x_i,x_j)-d_c).\\ \chi(z)=\begin{cases}1, & z<0;\\ 0, & z\geqslant 0.\end{cases}\end{cases}\tag{1}$$

$$\rho(x_i)=\sum_{x_j}\exp\Big(-\Big(\frac{d(x_i,x_j)}{d_c}\Big)^2\Big).\tag{2}$$

其中: $\rho(x_i)$是数据点x_i的局部密度; $d(x_i,x_j)$是数据点x_i与x_j的欧氏距离; d_c是 DPC 算法中唯一的距离阈值参数, 根据文献 [16] 的描述, d_c通常取 2 %; $\chi(z)$是一个分段函数, 当z小于 0 时, $\chi(z)$的值为 1, 否则$\chi(z)$取 0.

针对不同的数据集, DPC 提供了两种局部密度的计算方法, 这也导致另外一个问题: 不同局部密度计算方法可能适用于不同的数据集, 采用何种方法来计算将增加额外的工作量.

其次, DPC 定义了另外一个关键变量: 相对距离$\delta(x_i)$. 它是数据点x_i到具有更高密度数据点x_j的最短距离, 计算方法如下所示:

$$\delta(x_i)=\min_{x_j:\rho(x_j)>\rho(x_i)}(d(x_i,x_j)).\tag{3}$$

当数据点x_i具有最大的局部密度时, DPC 认为该点可能是一个峰值点, 其相对密度被设定为最大值, 计算方法如下所示:

$$\delta(x_i)=\max_{x_j}(d(x_i,x_j)).\tag{4}$$

当每个数据点的局部密度和相对距离计算完毕后, DPC 根据下式将候选的聚类中心通过决策图筛选出来:

$$\gamma(x_i)=\delta(x_i)*\rho(x_i).\tag{5}$$

候选聚类中心的特点是数据点同时具有最大的局部密度和相对距离.

最后, DPC 算法从候选聚类中心集中筛选出实际聚类中心, 依次将剩余点分配给最近且密度较高的聚类中心.

2 相关工作

自密度峰值聚类算法被提出以来, 广大研究者针对 DPC 的不足开展了大量研究, 主要集中在局部密度的计算、剩余点的分配策略、多峰值问题和聚类中心选择几个方面.

在局部密度计算方面, 首先是 DPC-KNN 算法[17],

该算法主要采用K近邻的方法重新定义了一种新的局部密度, 考虑了数据点之间的分布差异, 避免了DPC 算法不好选取截断距离参数的缺陷. DPC-KNN 相比 DPC 提高了聚类性能, 但面对密度分布不均的数据集时无法准确获取真实的聚类中心. 之后, Xie 等[18] 提出了一种经典的 FKNN-DPC 算法, 该算法采用K近邻方法与模糊集理论设计了新的局部密度; Liu 等[19] 提出了一种 SNN-DPC 算法, 该算法基于最近邻与共享近邻重新定义了局部密度, 能够更好地适配样本点的局部环境, 但聚类过程中与 FKNN-DPC 一样, 都需要采用固定的K近邻参数. 面向密度分布不均的数据集, 陈蔚昌等[22] 提出一种近邻优化密度峰值聚类算法, 该算法结合逆近邻和K近邻定义新的局部密度, 改善了稀疏样本的局部密度, 能够比较准确地找到类簇中心. 此外, 大量衍生算法都注重局部密度的优化, CDP[23]、REDPC[24]、IDDC[25]、DPC-FWSN[26]、SFKNN-DPC[27]、ANN-DPC[28] 算法分别采用比较密度、残差、相对密度、最近邻模糊核函数、标准差加权距离、自适应近邻的方法来改进局部密度.

在剩余点分配策略方面, FKNN-DPC 算法[18] 提出了一种两阶段剩余点分配策略, 然而该分配策略每次都采用固定K值, 没有考虑样本点的局部分布. 为了应对密度不平衡的样本集, 一种基于相对密度的聚类算法 (IDDC) 应运而生, 该算法从聚类的角度寻找未分配的点, 同时设计了一种新的分配策略, 但 IDDC 需要 2 个参数. 针对不平衡的数据集, Zhao 等[26] 提出了一种新的 DPC-FWSN 聚类算法, 设计了加权共享邻居相似度分配策略. 实验表明, DPC-FWSN 能有效处理密度分布不均匀的数据集, 不足之处是仍采用手动设置近邻参数k. 在 FKNN-DPC 的基础上, Xie 等[27] 进一步提出了 SFKNN-DPC, 考虑了每个特征对数据点之间距离的贡献, 同时设计了一种分而治之的分配策略, 具有更好的鲁棒性. 对于 DPC 无法找到稀疏集群聚类中心的问题, 一种自适应最近邻密度聚类算法 (ANN-DPC)[28] 被提出来, 该算法采用自适应近邻算法并结合广度优先搜索和模糊加权自适应近邻算法设计了新的分配策略. 虽然 ANN-DPC 性能优异, 但仍需要提前指定聚类数目. 赵嘉等[29] 针对 DPC 对密集程度不一的数据处理效果不佳的问题, 提出了一种相互邻近度的密度峰值聚类算法, 设计了数据全局和局部特征的样本相互邻近度的度量准则, 并提出了一种新的样本分配策略, 但参数k仍需要人为指定.

在处理多峰值问题方面, Ren 等[20] 提出了一种分层K近邻和子簇合并的密度峰值聚类算法 (LKSM_DPC), 该算法的核心是设计一种基于共享近邻和万有引力的子簇合并策略. 虽然该算法能有效处理一个集群中的多个峰值问题, 但在子簇合并时选择多少个子簇需要经过大量调试, 增加了时间开销. 针对 DPC 无法有效处理多峰值的问题, Xu 等[30] 提出了 FDPC 算法, 该算法在找到初始聚类中心后利用支持向量机计算聚类之间的反馈值, 并根据反馈值进行聚类. 针对 DPC 无法区分重叠集群的痛点, Parmar 等[24] 设计了一种 REDPC 算法, 采用残差计算局部密度并识别低密度样本点. 虽然该方法能够生成更有利聚类的决策图, 但聚类过程参数多且自主性较差. 陈梅等[31] 针对 DPC 可能存在的多密度峰值问题, 提出了一种基于低密度分数的密度峰值聚类算法 (LS-DPC), 该算法能够有效获得所有子簇的簇中心点, 但聚类效率有限.

在聚类中心选择方面, 针对 DPC 依赖与截断距离参数的问题, 通过引入自然近邻的思想形成了一种 NaNDP 算法[32], 该算法不需要额外的参数且能通过搜索聚类中点的自然邻域, 从聚类中心进行扩展, 最后定义扩展规则来确定聚类的边界. 唯一的不足是 NaNDP 算法仍然采用 DPC 的分配策略, 对于边界点的划分不太理想. 为了解决 DPC 针对部分数据集无法正确选择聚类中心的问题, ADPC-KNN[33] 设计了一种新的自动选择聚类中心方法, 利用簇密度可达的思想改善了算法的聚类性能, 但是方法中的唯一参数需要手动人工设置. 一种基于系统密度的锚点聚类方法 (APC)[34] 在 2020 年被提出, 该算法利用锚点为中心得到中间聚类并自动选择合适的聚类策略, 实验结果表明 APC 算法在大多数情况下具有较好的聚类性能, 但 4 个自定义参数产生了较大的时间开销. GADPC[35] 基于转弯角度和图连通性自动选择聚类中心, 虽然该算法在处理如 Jain 和 Spiral 不同密度的数据集时更加有效, 但却忽略了同 DPC 选取截断距离参数的问题.

3　BRPMK-DPC 算法

本节主要介绍基于双向代表点和相互K近邻的密度峰值聚类算法 (BRPMK-DPC). BRPMK-DPC 算法除了能够识别任意形状、任意密度和任意尺寸的数据集以外, 还具有两大贡献: 1) 设计了一种双向代表点的局部密度, 有利于选择正确的聚类中心; 2) 提出了一种相互K近邻的剩余点分配方法, 可以有效避免剩余点分配错误从而引发多米诺效应. 具体技术细节在后文中进行详细介绍.

3.1 双向代表点的局部密度

DPC 算法在计算局部密度时有一个关键参数: 截断距离d_c, 但针对不同数据集时很难获取最优值. KNN-DPC 采用了K近邻的方法计算局部密度, 虽然避免了d_c的抉择, 但K近邻没有考虑样本点周围的分布情况, 固定的K值也容易导致聚类中心的计算错误. 因此, 本节设计一种新的局部密度, 称为双向代表点的局部密度.

首先通过下式计算样本点x_i的K近邻集合$\mathrm{KNN}(x_i)$, 并按距离进行升序排序:

$$\mathrm{KNN}(x_i)=\{x_j\in D|d(x_i,x_k)-d(x_i,x_j)\geqslant 0\}. \tag{6}$$

其中: D表示样本数据集, $d(x_i,x_k)$表示样本点x_i与第k个近邻的距离.

接着, 定义x_i的逆K近邻集合$\mathrm{RKNN}(x_i)$如下:

$$\mathrm{RKNN}(x_i)=\{x_j\in D|x_i\in \mathrm{KNN}(x_j)\}. \tag{7}$$

$\mathrm{RKNN}(x_i)$反应了样本点x_i在数据集D中的被影响情况: 若x_i处于高密度区域时, 通常会被更多的样本点包围; 反之若处于稀疏区域则逆近邻数量较少. 逆K近邻集合的大小更容易反馈样本点所处的局部分布情况.

其次, 为了更好地在密度分布不均的数据集中选取聚类中心, 本节设计双向代表点的局部密度计算方法, 代表点的思想来自我国社会结构中的人大代表, 代表往往在一个人群中比其他人更有权力, 权力来自周围邻居, 反过来服务他们的邻居. 单代表点的作用比较局限[36], 因此本文提出双向代表点的理念, 包括正向K近邻代表点和逆向逆K近邻代表点.

对于任意数据点x_i, 它的正向K近邻代表点$\mathrm{RPK}(x_i)$定义为

$$\mathrm{RPK}(x_i)=\underset{x_j\in \mathrm{KNN}(x_i)}{\arg}(\max(\rho(x_j))). \tag{8}$$

为了体现代表点的重要性, 本文将x_i的正向K近邻代表点量化为正向K近邻代表值, 该值表示数据点x_i在数据集中作为其他数据点的K近邻代表点次数. 若正向K近邻代表值越大, 则表明该点为K近邻点集中越密集, 同时也表明该点在集群中的密集程度.

对于任意数据点x_i, 它的正向K近邻代表值$\mathrm{RPKV}(x_i)$定义为

$$\begin{cases}\mathrm{RPKV}(x_i)=\displaystyle\sum_{x_j\in D}\Phi(x_j).\\ \Phi(x_j)=\begin{cases}0,\ \mathrm{RPK}(x_j)\neq x_i;\\ 1,\ \mathrm{RPK}(x_j)=x_i.\end{cases}\end{cases} \tag{9}$$

式 (9) 主要由其余点$(x_j\in D)$的K近邻代表点是否为x_i决定, 也就是通过$\Phi(x_j)$计算得到. 若$\mathrm{RPK}(x_j)$为x_i, 则$\Phi(x_j)=1$, 反之$\Phi(x_j)=0$.

同理, 本文定义数据点x_i的逆向逆K近邻代表点$\mathrm{RPRK}(x_i)$和逆向逆K近邻代表值$\mathrm{RPRKV}(x_i)$, 分别如下所示:

$$\mathrm{RPRK}(x_i)=\underset{x_j\in \mathrm{RKNN}(x_i)}{\arg}(\max(\rho(x_j))). \tag{10}$$

$$\begin{cases}\mathrm{RPRKV}(x_i)=\displaystyle\sum_{x_j\in D}\Phi(x_j).\\ \Phi(x_j)=\begin{cases}0,\ \mathrm{RPRK}(x_j)\neq x_i;\\ 1,\ \mathrm{RPRK}(x_j)=x_i.\end{cases}\end{cases} \tag{11}$$

式 (10) 需要先结合式 (7) 计算出每个数据点x_i的逆K近邻点集, 再选择逆K近邻点集中具有最大局部密度的数据点作为逆向逆K近邻代表点$\mathrm{RPRK}(x_i)$.

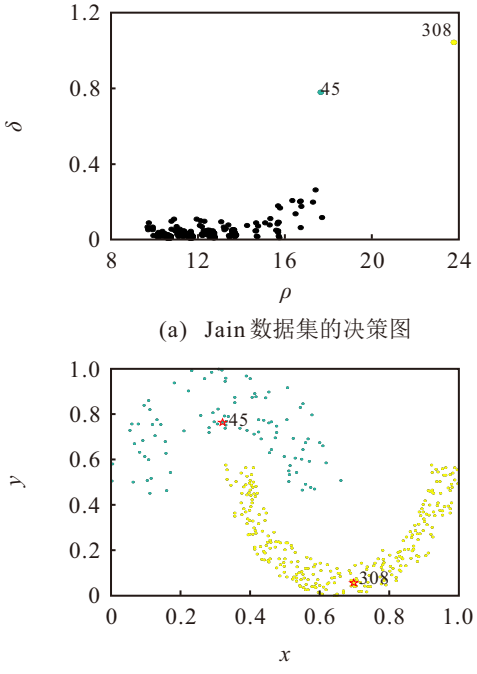
$\mathrm{RPRKV}(x_i)$表示x_i的逆向逆K近邻代表值, 该值也是x_i作为其他数据点的逆K近邻代表点次数, 该值越大表明x_i为逆K近邻点集中越密集, 也反应了该点在集群中的局部密集程度.

式 (11) 主要由其余点$(x_j\in D)$的逆K近邻代表点是否为x_i决定, 即通过$\Phi(x_j)$计算得到. 若$\mathrm{RPRK}(x_j)$为x_i, 则$\Phi(x_j)=1$, 反之$\Phi(x_j)=0$.

最后, 基于双向代表点的局部密度计算方法为

$$\rho(x_i)=\mathrm{RPKV}(x_i)+\mathrm{RPRKV}(x_i)+\sum_{x_j\in \mathrm{KNN}(x_i)}\exp(-d(x_i,x_j)). \tag{12}$$

双向代表点的局部密度计算方法更容易在密度



(a) Jain 数据集的决策图

(b) Jain 数据集的聚类结果

图1 在 Jain 数据集上的聚类中心示例

分布不均的数据集中捕获真实的聚类中心, 如图 1 在典型密度分布不均的数据集 Jain 中, 双向代表点局部密度在决策图 1(a) 中起到了关键作用, 能够正确获取一个稀疏集群的聚类中心 (图 1(b) 中编号 45) 和一个稠密集群的聚类中心 (图 1(b) 中编号 308).

3.2　相互K近邻的剩余点分配方法

DPC 在剩余点分配时如果一个样本点分配错误则容易引起连锁反应, 主要原因是 DPC 的分配策略是优先将剩余点分配到高密度集群中, 从而导致原本属于稀疏集群的样本点被错误分配到高密度集群中. 受 FKNN-DPC 启发, 本文设计了一种相互K近邻的剩余点分配方法, 主要考虑到每个样本点的相互K近邻数量不固定, 更适合匹配样本点的局部分布. 该方法与 FKNN-DPC 不同, FKNN-DPC 在分配过程中每次都采用固定K值进行分配.

首先, 本节定义了数据点x_i的相互K近邻集合$\mathrm{MKNN}(x_i)$, 它主要利用了K近邻与逆K近邻的概念, $\mathrm{MKNN}(x_i)$反应了数据点x_i在正向K近邻与逆向K近邻的双向局部关系. 若数据点x_i的正向K近邻集合$\mathrm{KNN}(x_i)$与逆向K近邻集合$\mathrm{RKNN}(x_i)$的交集不为空, 则x_i的相互K近邻集合$\mathrm{MKNN}(x_i)$等于$\mathrm{KNN}(x_i)\bigcap\mathrm{RKNN}(x_i)$. 但是, 极个别数据点的正向$K$近邻集合与逆向$K$近邻集合的交集存在为空的情况, 本文认为这些个别点的相互K近邻集合近似于它的K近邻集合, 计算方法如下所示:

$$\mathrm{MKNN}(x_i)=\begin{cases}\mathrm{KNN}(x_i)\bigcap\mathrm{RKNN}(x_i),\\ \quad \{\mathrm{KNN}(x_i)\bigcap\mathrm{RKNN}(x_i)\}\neq\varnothing;\\ \mathrm{KNN}(x_i),\ \{\mathrm{KNN}(x_i)\bigcap\mathrm{RKNN}(x_i)\}=\varnothing.\end{cases}\tag{13}$$

通过实验分析, 可以发现每个样本点的相互K近邻数量都不太相同, 这样带来的好处是采用相互K近邻来描述样本点的局部情况比固定K近邻的方法更合理, 尤其是区分低密度区域, 更有利于剩余点分配, 避免在分配过程中采用固定K值带来的劣势.

本文设计的相互K近邻的剩余点分配方法主要包括两个算法, 算法 1 利用样本点的相互K近邻与广度优先搜索进行初次分配, 广度优先搜索的基本原理是从聚类中心开始, 逐步将相邻点分配到该类簇, 优先处理距离较近的点; 算法 2 是在算法 1 的基础上结合相互K近邻计算未分配点的隶属度, 将属于某簇的高概率样本点分配到最合适的簇中, 若还剩有未分配的点则根据K近邻进行类别归属划分. 该分配方法最突出的特点是能根据样本点的局部情况进行自适应分配, 具有较强的鲁棒性.

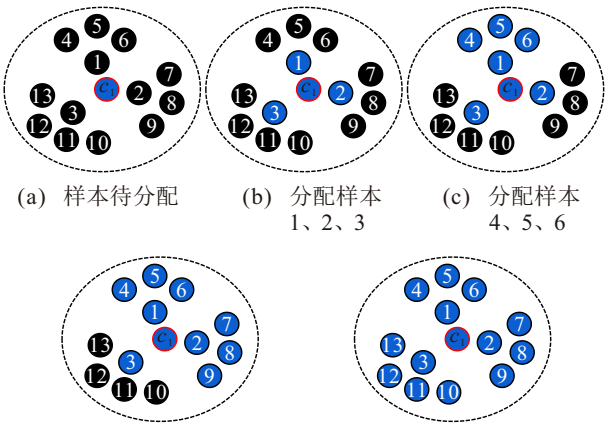
算法1　基于相互K近邻与广度优先搜索的初次分配算法.

输入: 聚类中心集合C, 相互K近邻集合$\mathrm{MKNN}(x_q)$, 距离矩阵Dist;

输出: 数据点的类簇标签.

1. for each $c_i\in C$ do
2. 设置$\mathrm{MKNN}(c_i)$的类簇标签为lab_{c_i}, 将$\mathrm{MKNN}(c_i)$进入队列Que;
3. while Que $\neq\varnothing$ do
4. 取Que的对头元素x_q;
5. for each $x_p\in\mathrm{MKNN}(x_q)$ do
6. if $\mathrm{lab}_{x_p}==0$ and $d(x_p,x_q)\leqslant\dfrac{\sum\limits_{x_j\in\mathrm{MKNN}(x_q)}d(x_q,x_j)}{|\mathrm{MKNN}(x_p)|}$ then
7. 输出x_p的类簇标签为lab_{c_i};
8. 将x_p进入队列Que;
9. end
10. end
11. 移除Que的对头元素;
12. end
13. end

为了进一步说明算法 1 的分配原理, 给出图 2 所示的样本分配示意图. 利用广度优先搜索的关键是需要用一个队列. 假设该数据集的聚类中心是c_1, 样本点 1、2、3 是c_1的相互K近邻, 样本点 4、5、6 是样本点 1 的相互K近邻, 样本点 7、8、9 是样本点 2 的相互K近邻, 样本点 10、11、12、13 是样本点 3 的相互K近邻. 样本点的详细分配步骤如下: 1) 初始待分配状态如图 2(a) 所示, 样本呈蓝色表示已分配, 样本为黑色表示未分配; 2) 如图 2(b) 所示, 由于样本点 1、2、3 属于c_1的相互K近邻, 样本点 1、2、



(a) 样本待分配　(b) 分配样本 1、2、3　(c) 分配样本 4、5、6

(d) 分配样本7、8、9　(e) 分配样本10、11、12、13

图2　算法 1 的分配示意图

3 被标记为已分配, 并进入队列; 3) 如图 2(c) 所示, 取队头元素样本点 1, 将满足条件的样本点 4、5、6 设置为已分配, 并入队列的同时出队样本点 1; 4) 同理如图 2(d) 所示, 取队头元素样本点 2, 将满足条件的样本点 7、8、9 设置为已分配, 并入队列的同时出队样本点 2; 5) 如图 2(e) 所示, 继续取队头元素样本点 3, 将满足条件的样本点 10、11、12、13 设置为已分配, 并入队列的同时出队样本点 3. 由于所有样本点已分配, 依次移除队列中的元素后算法 1 结束.

其次, 定义数据点x_i与x_j的相似度$S(x_i,x_j)$, 两数据点之间的距离越小, 表明两数据点越相似, $S(x_i,x_j)$越大. 计算方法如下:

$$S(x_i,x_j)=\frac{1}{d(x_i,x_j)^2+1}. \tag{14}$$

为计算数据点x_i属于集群c_y $(y=1,2,\dots,m)$的隶属度$p(x_i,c_y)$, 考虑数据点x_i的相互K近邻对$p(x_i,c_y)$的贡献, 即计算x_i的相互K近邻集合中属于c_y的概率. 该值不仅考虑了数据点之间的相似度, 同时也考虑了数据点的相互K近邻分布, 更有利于将剩余点分配到最合适的类簇中. $p(x_i,c_y)$的计算方法为

$$p(x_i,c_y)=\operatorname{avg}\Big(\sum_{x_j\in \mathrm{MKNN}(x_i),\mathrm{lab}_{x_j}=c_y} S(x_i,x_j)\Big). \tag{15}$$

其中: $\mathrm{lab}_{x_j}=c_y$表示数据点x_j的类别属于c_y; $\operatorname{avg}(\cdot)$是一个平均值函数, 用于计算$x_i$的相互$K$近邻集合中属于同一类簇的平均概率.

假设分配算法 1 执行完毕后还有剩余点集D_t未分配, 则接下来采用算法 2 进行隶属度的计算并进行样本点的再次分配.

算法2 相互K近邻的隶属度分配算法.

输入: 剩余未分配点集D_t, $\mathrm{MKNN}(x_q)$, C;

输出: 数据点的类簇标签.

1. for each $x_i\in D_t$ do
2. 计算剩余点的隶属度$p(x_i,c_y)$;
3. 构造剩余点的隶属度矩阵$P[t][|C|]=\{p(x_i,c_y)|x_i\in D_t\}$;
4. 构造剩余点的最大隶属度矩阵$\mathrm{MaxP}[t][2]=\{\max(p(x_i,c_y)),\mathrm{maxc}(p(x_i,c_y))\}$;
5. end
6. 从$\mathrm{MaxP}[t][1]$中选择最大隶属度的点x_p;
7. if $P[x_p][c_y]\neq 0$ then
8. flag=1;
9. end
10. while flag $\neq 0$ do
11. 设置并输出x_p的类标签$\mathrm{MaxP}[x_p][2]$;
12. 更新x_p的隶属度矩阵P和MaxP;
13. for each $x_q\in \mathrm{MKNN}(x_p)$ do
14. 用$p(x_q,c_y)=p(x_q,c_y)+S(x_p,x_q)$更新$P$和MaxP;
15. end
16. 继续从$\mathrm{MaxP}[t][1]$中选择最大隶属度的点x_p;
17. if $P[x_p][c_y]==0$ then
18. flag $=0$;
19. end
20. end
21. while $x_j\in D_t$ and $\mathrm{lab}_{x_j}==0$ do
22. 利用K近邻原理设置并输出x_j的类标签;
23. end

为了进一步说明算法 2 中的隶属度计算, 以图 3 为例. 假设数据集有两个类簇, 分别为蓝色簇 (类中心c_1) 和绿色簇 (类中心c_2), 该数据集当前已执行完毕算法 1, 现在还剩有样本点 9 待分配, 样本点 9 的相互K近邻集合为$\{7,8,10,11\}$, 样本点 9 与相互K近邻的距离分别为 0.2、0.3、0.4、0.5. 接下来采用算法 2 对样本点进行隶属度计算, 利用式 (14) 可以计算出$S(9,7)=0.92$, $S(9,8)=0.96$, $S(9,10)=0.86$, $S(9,11)=0.8$; 利用式 (15) 计算样本点 9 的隶属度为$p(9,c_1)=0.94$, $p(9,c_2)=0.83$. 因此, 根据算法 2 中隶属度最大原理, 可将样本点 9 分配到聚类中心为c_1的簇中.

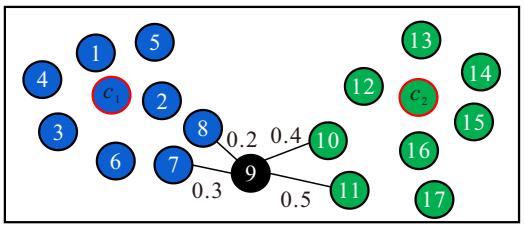


图3 算法 2 的隶属度计算示意图

3.3 BRPMK-DPC 算法

本节给出 BRPMK-DPC 算法的完整流程, 具体细节见算法 3.

算法3 BRPMK-DPC.

输入: 数据集$D=\{x_1,\dots,x_i,\dots,x_n\}$, 最近邻参数$k$;

输出: 数据点的类簇标签.

1. 对数据集D进行标准化处理, 并计算出两数据点之间的欧氏距离矩阵;

2. 利用式(6) ~ (11)计算数据点x_i的正向K近邻代表值$\mathrm{RPKV}(x_i)$和逆向逆K近邻代表值$\mathrm{RPRKV}(x_i)$;

3. 结合式(12)计算数据点x_i的局部密度$\rho(x_i)$;
4. 利用式(3)和(4)计算数据点x_i的相对距离$\delta(x_i)$;
5. 利用式(5)构建二维决策图, 并选择合适的聚类中心;
6. 利用算法1对剩余点进行初次分配;
7. 在算法1的基础上, 采用算法2对还未分配的剩余点进行再次分配;
8. 输出每个数据点的类簇标签.

接下来评估 BRPMK-DPC 算法的时间复杂度. 假设测试数据集的数量规模为n, 设置每个数据点的最近邻数量为k. 由具体算法流程可知, BRPMK-DPC 的时间复杂度主要取决于以下几步: 1) 计算欧氏距离矩阵的时间复杂度为$O(n^2)$; 2) 计算正向K近邻代表值和逆向逆K近邻代表值的时间复杂度均为$O(kn)$; 3) 计算局部密度的时间耗费为$O(n^2+2kn)$; 4) 计算相对密度的时间复杂度为$O(n^2)$; 5) 构造决策图的时间复杂度为$O(n)$; 6) 算法 1 的核心是结合了广度优先搜索算法, 需要额外使用 1 个队列, 在最坏情况下所有数据点都进行入队操作, 算法 1 的时间复杂度为$O(n)$; 7) 在算法 2 中, 一是计算未分配点隶属度的时间复杂度为$O(kn)$, 二是更新隶属度矩阵, 在最坏情况下的时间复杂度为$O(n^2)$. 综上所述, BRPMK-DPC 算法的时间耗费与原始 DPC 算法一致, 时间复杂度为$O(n^2)$.

4　实验与分析

为了测试 BRPMK-DPC 算法的聚类性能和运行时间, 本节将 BRPMK-DPC 与经典划分聚类算法K-means[12]、典型密度聚类算法 DBSCAN[15]、原始 DPC[16] 以及衍生类 DPC 算法 (DPC-KNN[17]、FKNN-DPC[18]、DPCSA[37] 和 LF-DPC[38]) 进行比较.

4.1　实验准备

本文采用的测试数据集主要包括 10 个人工合成数据集和 11 个真实数据集 (其中包括 1 组人脸数据集). 测试数据集的详细信息见表 1 和表 2.

表1　人工合成数据集

数据集	记录数量	属性	类簇数量
Jain	373	2	2
Flame	240	2	2
Spiral	312	2	3
Pathbased	300	2	3
Compound	399	2	6
D31	3 100	2	31
Twomoons	1 502	2	2
ThreeCircle	299	2	3
Lineblob	266	2	3
Ring	1 000	2	2

表2　真实合成数据集

数据集	记录数量	属性	类簇数量
Seeds	210	7	3
Libras	360	91	15
Wine	178	13	3
Parkinsons	195	23	2
SCADI	70	206	7
Ecoli	336	8	8
Dermatology	366	33	6
Banknote	1 372	4	2
Dim1000	800	1 000	16
BinaryAlphadigs	1 404	320	36
Olivetti Faces	400	92×112	40

本文选择 3 个聚类评估指标作为算法的度量标准, 分别是调整兰德系数 (ARI)[39]、调整互信息 (AMI)[40] 和 Fowlkes-Mallows Index 指数 (FMI)[41]. 各个评估指标值越接近 1, 表示聚类性能越好.

4.2　BRPMK-DPC 在人工合成数据集上的实验

本节重点探讨 BRPMK-DPC 与其他 7 种算法在二维合成数据集上的聚类效果, 实验比较结果见表 3 (最优评估指标值已加粗).

在 Jain、Flame、Spiral、Twomoons、ThreeCircle、Lineblob 以及 Ring 等数据集测试中, BRPMK-DPC 算法均展现出卓越的聚类能力, 达到了完美聚类的效果, 其 ARI 值精准达到 1. 即便在 Pathbased 和 Compound 数据集测试中, BRPMK-DPC 未能实现完美聚类, 然而从实际聚类结果来看, 其聚类性能依旧大幅领先于其余 7 种对比算法, 优势显著. 综合各方面情况而言, 在人工合成数据集实验中, 本文所提出的算法在 9 个数据集测试中均斩获最优聚类性能. 仅在面对 D31 数据集时, BRPMK-DPC 算法的表现稍逊于 LF-DPC 和 FKNN-DPC, 即便如此, 它的聚类效果依然优秀. 因此, 本文认为基于双向代表点的局部密度计算方法以及相互K近邻的剩余点分配方法, 为 BRPMK-DPC 算法注入了强劲动力, 使其达成出众的聚类效果.

4.3　BRPMK-DPC 在真实数据集上的实验

本节主要测试 BRPMK-DPC 在真实数据集上的聚类性能. 真实数据集来自不同的研究领域, 具有不同的尺寸、维度和集群数目, 能够更好地评估所提算法的适用性. 表 4 列出了 8 个聚类算法在 10 个真实数据集上的聚类结果.

从表 4 可看出, 所提算法 BRPMK-DPC 在 8 个数据集中 (Seeds、Libras、Parkinsons、SCADI、Ecoli、Dermatology、Dim1000 和 BinaryAlphadigs) 的各项评估指标上明显优于其他 7 个算法, 在剩余 2 个数据

表3 不同算法在人工合成数据集上的聚类结果

数据集	算法	ARI	AMI	FMI	参数	数据集	算法	ARI	AMI	FMI	参数
Jain	K-means	0.5767	0.4916	0.8200	2	Flame	K-means	0.5117	0.4693	0.7643	2
	DBSCAN	0.9887	0.9691	0.9956	0.05/8		DBSCAN	0.9081	0.7570	0.9561	0.065/4
	DPC	0.6183	0.5396	0.8386	2%		DPC	**1**	**1**	**1**	5%
	DPC-KNN	0.7146	0.6183	0.8819	2%		DPC-KNN	**1**	**1**	**1**	0.50%
	FKNN-DPC	0.8224	0.7092	0.9359	43		FKNN-DPC	0.9666	0.9267	0.9845	5
	DPCSA	0.0442	0.2167	0.5924	—		DPCSA	**1**	**1**	**1**	—
	LF-DPC	0.4059	0.2936	0.8270	40		LF-DPC	**1**	**1**	**1**	2
	BRPMK-DPC	**1**	**1**	**1**	11		BRPMK-DPC	**1**	**1**	**1**	3
Spiral	K-means	−0.0061	−0.0056	0.3274	3	Pathbased	K-means	0.4613	0.5098	0.6617	3
	DBSCAN	**1**	**1**	**1**	0.04/3		DBSCAN	0.5890	0.6884	0.7317	0.065/4
	DPC	**1**	**1**	**1**	2%		DPC	0.4530	0.4997	0.6585	2%
	DPC-KNN	**1**	**1**	**1**	2%		DPC-KNN	0.4602	0.5080	0.6617	2%
	FKNN-DPC	**1**	**1**	**1**	6		FKNN-DPC	0.7323	0.7744	0.8226	8
	DPCSA	**1**	**1**	**1**	—		DPCSA	0.6133	0.7073	0.7511	—
	LF-DPC	**1**	**1**	**1**	5		LF-DPC	0.9699	0.9525	0.9799	8
	BRPMK-DPC	**1**	**1**	**1**	9		BRPMK-DPC	**0.9794**	**0.9693**	**0.9863**	10
Compound	K-means	0.5595	0.6753	0.6595	6	D31l	K-means	0.8646	0.9306	0.8693	31
	DBSCAN	0.8402	0.7839	0.8850	0.08/14		DBSCAN	0.8078	0.8895	0.8186	0.04/40
	DPC	0.5989	0.7798	0.6963	2%		DPC	0.9332	0.9539	0.9354	2%
	DPC-KNN	0.8087	0.7913	0.8661	0.50%		DPC-KNN	0.9357	0.9549	0.9378	2%
	FKNN-DPC	0.8479	0.8341	0.8941	8		FKNN-DPC	**0.9516**	**0.9653**	**0.9531**	23
	DPCSA	0.8284	0.8392	0.8707	—		DPCSA	0.9353	0.9552	0.9374	—
	LF-DPC	0.8409	0.8231	0.8891	10		LF-DPC	0.9473	0.9620	0.9490	20
	BRPMK-DPC	**0.9044**	**0.8922**	**0.9295**	14		BRPMK-DPC	0.9400	0.9576	0.9419	28
Twomoons	K-means	0.4374	0.3324	0.7387	2	ThreeCircle	K-means	0.0533	0.1560	0.4031	3
	DBSCAN	**1**	**1**	**1**	0.1/10		DBSCAN	0.8528	0.7599	0.9057	0.08/4
	DPC	0.5896	0.5524	0.8075	2%		DPC	0.5625	0.6042	0.7632	5%
	DPC-KNN	0.4921	0.4881	0.7604	2%		DPC-KNN	0.0338	0.0727	0.5809	1%
	FKNN-DPC	0.3862	0.4254	0.7103	6		FKNN-DPC	**1**	**1**	**1**	6
	DPCSA	0.2746	0.3647	0.6607	—		DPCSA	**1**	**1**	**1**	—
	LF-DPC	0.2746	0.3647	0.6607	8		DPCSA	**1**	**1**	**1**	8
	BRPMK-DPC	**1**	**1**	**1**	16		BRPMK-DPC	**1**	**1**	**1**	5
Lineblob	K-means	0.4875	0.5828	0.6650	3	Ring	K-means	0.0011	0.0008	0.5001	2
	DBSCAN	**1**	**1**	**1**	0.08/5		DBSCAN	**1**	**1**	**1**	0.08/8
	DPC	0.4875	0.5828	0.6650	2%		DPC	0.1248	0.2041	0.6473	2%
	DPC-KNN	0.3577	0.4784	0.5949	1%		DPC-KNN	0.3595	0.3954	0.7051	2%
	FKNN-DPC	**1**	**1**	**1**	6		FKNN-DPC	**1**	**1**	**1**	8
	DPCSA	**1**	**1**	**1**	—		DPCSA	**1**	**1**	**1**	—
	LF-DPC	0.7179	0.7794	0.8148	8		LF-DPC	**1**	**1**	**1**	10
	BRPMK-DPC	**1**	**1**	**1**	6		BRPMK-DPC	**1**	**1**	**1**	8

集中也取得了较好的聚类效果. 尤其是在高维数据集 Dim1000 中, 尽管 8 种聚类算法的聚类结果相差不大, 但 BRPMK-DPC 从 3 个指标来看仍然优于其他 7 种算法; 在另一组高维数据集 BinaryAlphadigs 中, 尽管所有对比算法对该数据集聚类性能有限, 但 BRPMK-DPC 的聚类结果也领先其余对比算法.

在 Wine 数据集对比测试中可以发现, LF-DPC 通过局部公平密度取得了最佳的聚类效果, 紧随其后的是 FKNN-DPC 和 BRPMK-DPC 算法. 本文算法虽然没有获得最优聚类效果, 但也领先于其他 5 种算法.

同理, 在 Banknote 数据集测试中, 没有参数的 DPCSA 算法获取到最佳的聚类效果, 其他算法均被 DPCSA 打败. 本文提出的算法仅轻微落后于 DPCSA, 所有对比结果中排名第 2, 可能造成这种结果的原因是 DPCSA 的剩余点分配策略非常适用于该数据集.

综上, 对表 4 进行总结可以发现, BRPMK-DPC 在 8/10 的数据集中 ARI、AMI 和 FMI 指标上都获取了最佳聚类性能, 虽然所有对比算法的聚类结果

表4　不同算法在真实数据集上的聚类结果

数据集	算法	ARI	AMI	FMI	参数	数据集	算法	ARI	AMI	FMI	参数
Seeds	K-means	0.7049	0.6705	0.8026	3	Libras	K-means	0.3032	0.5230	0.3528	15
	DBSCAN	0.5291	0.5302	0.6711	0.24/16		DBSCAN	0.2348	0.3598	0.2799	0.9/1
	DPC	0.7341	0.7172	0.8231	2%		DPC	0.2984	0.5138	0.3682	0.40%
	DPC-KNN	0.7448	0.7144	0.8297	1%		DPC-KNN	0.3051	0.5471	0.3666	1%
	FKNN-DPC	0.8024	0.7684	0.8680	4		FKNN-DPC	0.3211	0.5367	0.3943	10
	DPCSA	0.6873	0.6609	0.7918	—		DPCSA	0.2683	0.4939	0.3572	—
	LF-DPC	0.7777	0.7381	0.8516	8		LF-DPC	0.3437	0.5406	0.3996	5
	BRPMK-DPC	**0.8493**	**0.8042**	**0.8991**	8		BRPMK-DPC	**0.3814**	**0.5593**	**0.4483**	12
Wine	K-means	0.8471	0.8301	0.8984	3	Parkinsons	K-means	0.0520	0.2129	0.5957	2
	DBSCAN	0.5292	0.5484	0.7121	0.5/21		DBSCAN	0.0252	0.0071	0.5775	0.5/17
	DPC	0.6724	0.7065	0.7835	2%		DPC	0.2686	0.1772	0.8140	0.20%
	DPC-KNN	0.6990	0.7228	0.8006	8%		DPC-KNN	0.2686	0.1772	0.8140	2%
	FKNN-DPC	0.8819	0.8566	0.9215	8		FKNN-DPC	0.2686	0.1772	0.8140	5
	DPCSA	0.7414	0.7480	0.8283	—		DPCSA	0.2686	0.1772	0.8140	—
	LF-DPC	**0.9150**	**0.8800**	**0.9436**	7		LF-DPC	0.2686	0.1772	0.8140	6
	BRPMK-DPC	0.8685	0.8473	0.9126	8		BRPMK-DPC	**0.3632**	**0.2151**	**0.8190**	5
SCADI	K-means	0.4583	0.4910	0.5805	7	Ecoli	K-means	0.4957	0.5192	0.6174	8
	DBSCAN	—	—	—	—		DBSCAN	0.0947	0.0696	0.5203	0.2/6
	DPC	0.5618	0.4966	0.6684	2%		DPC	0.7054	0.5816	0.7983	1%
	DPC-KNN	0.5627	0.4759	0.6690	2%		DPC-KNN	0.6913	0.5817	0.7939	5%
	FKNN-DPC	0.6191	0.5319	0.7122	6		FKNN-DPC	0.5914	0.5596	0.7071	7
	DPCSA	0.5939	0.4988	0.6932	—		DPCSA	0.4883	0.4229	0.6787	—
	LF-DPC	0.6953	0.5872	0.7736	6		LF-DPC	0.7060	0.5877	0.8014	6
	BRPMK-DPC	**0.7484**	**0.6510**	**0.8105**	6		BRPMK-DPC	**0.7357**	**0.6306**	**0.8158**	8
Dermatology	K-means	0.7312	0.8741	0.7856	6	Banknote	K-means	0.0223	0.0168	0.5139	2
	DBSCAN	0.4161	0.5176	0.5293	0.98/2		DBSCAN	0.8260	0.7542	0.9099	0.1/5
	DPC	0.6622	0.7167	0.7487	0.40%		DPC	0.8008	0.7751	0.8968	1%
	DPC-KNN	0.6349	0.7731	0.7089	1%		DPC-KNN	0.3955	0.3575	0.6524	0.8%
	FKNN-DPC	0.8654	0.8741	0.8994	6		FKNN-DPC	0.7702	0.7576	0.8793	20
	DPCSA	0.6062	0.7451	0.6896	—		DPCSA	**0.9653**	**0.9359**	**0.9828**	—
	LF-DPC	0.8288	0.8345	0.8704	8		LF-DPC	0.7702	0.7576	0.8793	10
	BRPMK-DPC	**0.9506**	**0.9386**	**0.9605**	9		BRPMK-DPC	0.9368	0.8806	0.9688	14
Dim1000	K-means	0.5407	0.7413	0.5774	16	BinaryAlphadigs	K-means	0.2824	0.5151	0.3026	36
	DBSCAN	—	—	—	—		DBSCAN	—	—	—	—
	DPC	0.5920	0.7170	0.6221	3%		DPC	0.1123	0.3006	0.1829	2%
	DPC-KNN	0.5858	0.7628	0.6166	2%		DPC-KNN	0.0483	0.2398	0.1805	2%
	FKNN-DPC	0.6011	0.7481	0.6277	10		FKNN-DPC	0.1664	0.3812	0.2545	8
	DPCSA	0.5980	0.7437	0.6247	—		DPCSA	0.1098	0.3448	0.2178	—
	LF-DPC	0.5510	0.7475	0.5810	9		LF-DPC	0.1500	0.3429	0.2261	8
	BRPMK-DPC	**0.6189**	**0.7639**	**0.6430**	12		BRPMK-DPC	**0.3044**	**0.5335**	**0.3320**	6

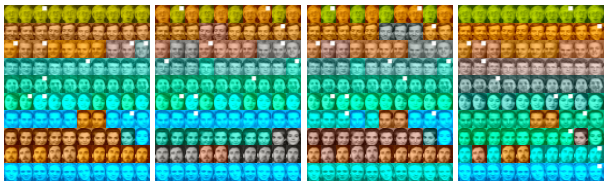
已经调优到最好, 但仍不如 BRPMK-DPC 算法, 这主要受益于所提出的双向代表点的局部密度计算方法和相互 K 近邻的剩余点分配方法. BRPMK-DPC 算法不仅能够有效处理密度分布不均的数据集, 还能进行高效的剩余点分配.

4.4　BRPMK-DPC 在人脸数据集上的实验

为了进一步评估 BRPMK-DPC 算法的性能, 在 Olivetti Face 数据集上进行实验, 并与 DPC、FKNN-DPC 和 DPCSA 进行对比. Olivetti Face 是一个被广泛用于机器学习和聚类测试的数据集, 该数据集有 40 组共 400 幅人脸图像, 每组数据记录了同一个测试者在不同光线、表情、面部细节下的脸部特征.

为了降低计算耗费, 从该数据集中选择 10 组人脸数据进行测试, 实验结果如图 4 所示, 图中从上到下分别编号 1 ～ 10, 不同类别以不同背景色分组显示, 其中灰色背景是未识别到的人脸图像. 首先, 在聚类中心识别方面, DPC 只能够识别到 6 组人脸数据的聚类中心, 编号为 2、8、9、10 组的人脸未识别到聚类中心; FKNN-DPC 和 DPCSA 能够识别到 7 组人脸数据的聚类中心, 但编号为 8、9、10 组的人脸

未能识别到聚类中心; 本文算法能够全部识别到10组人脸数据的聚类中心. 其次, 在剩余点分配方面, DPC、FKNN-DPC 和 DPCSA 在最后 3 组人脸数据上都有明显的错误分配, 而 BRPMK-DPC 在最后 3 组人脸数据上表现高效. 最后, 从表 5 的聚类评估指标来看, DPC 的聚类效果最差, FKNN-DPC 和 DPCSA 相比原始 DPC 在聚类性能方面有微小提升, 主要受益于改进算法的局部密度和剩余点分配方法, 本文提出的 BRPMK-DPC 在各项聚类指标中明显优于其他 3 种聚类算法.



(a) DPC (b) FKNN-DPC (c) DPCSA (d) BRPMK-DPC

图4 4 种算法在人脸数据集上的聚类结果展示

表5 4 种算法在人脸数据集上的聚类评估指标

算法	ARI	AMI	FMI	参数
DPC	0.5496	0.6728	0.6119	3%
FKNN-DPC	0.6034	0.7524	0.6425	3
DPCSA	0.5610	0.6845	0.6002	—
BRPMK-DPC	**0.8109**	**0.8304**	**0.8291**	2

4.5 BRPMK-DPC 运行时间分析

由于K-means 和 DBSCAN 与 DPC 算法原理不一致, 本文主要分析 DPC、DPC-KNN、FKNN-DPC、DPCSA、LF-DPC 和 BRPMK-DPC 算法的运行时间, 运行时间是每种算法各自运行 5 次取平均值并保留 4 位小数, 表 6 和表 7 分别展示了 6 种算法在人工合成数据集和真实数据集上的运行时间. 可以发现, 无论是人工合成数据集还是真实数据集, 原始 DPC 的运行时间最短, 这主要是因为 DPC 的局部密度计算和剩余点分配步骤简单; 只优化局部密度计算的 DPC-KNN 运行时间仅次于 DPC; 本文算法 BRPMK-DPC 与 FKNN-DPC、LF-DPC 运行时间基本一致, 主要是这 3 种算法不仅改进了局部密度计算方法, 还优化了剩余点分配策略, 因此 3 种算法的运行时间稍高于 DPC 和 DPC-KNN.

在局部密度计算过程中采用正向K近邻和逆向K近邻双向操作, 因此本文对该操作进行运行时间测试, 测试结果如表 8 所示. 从测试结果来看, 虽然正向K近邻和逆向K近邻会增加 BRPMK-DPC 算法的执行时间, 但仅占总运行时间的 1% ～ 6.5%, 基本不影响整个算法的运行时间.

表6 6 种不同算法在人工合成数据集上的运行时间 单位: s

数据集	DPC	DPC-KNN	FKNN-DPC	DPCSA	LF-DPC	BRPMK-DPC
Jain	0.1163	0.1383	0.1954	0.1287	0.2577	0.2780
Flame	0.1108	0.1133	0.1975	0.1204	0.2719	0.2007
Spiral	0.1204	0.1145	0.2169	0.1297	0.2243	0.1443
Pathbased	0.1178	0.1181	0.2094	0.1235	0.2202	0.2134
Compound	0.1231	0.1260	0.2016	0.1438	0.2077	0.2642
D31	0.9486	1.1827	5.8327	1.0205	6.2774	6.0564
Twomoons	0.2371	0.2417	3.5474	0.2630	2.5862	2.6286
ThreeCircle	0.0710	0.0767	0.2247	0.0837	0.1743	0.2547
Lineblob	0.0750	0.0789	0.2086	0.0878	0.2604	0.2792
Ring	0.1180	0.1510	1.4004	0.1613	1.7398	1.8228

表7 6 种不同算法在真实数据集上的运行时间 单位: s

数据集	DPC	DPC-KNN	FKNN-DPC	DPCSA	LF-DPC	BRPMK-DPC
Seeds	0.1116	0.1137	0.2255	0.1255	0.1759	0.1790
Libras	0.1287	0.1303	0.2108	0.1445	0.3529	0.3722
Wine	0.1180	0.1193	0.1603	0.1247	0.1734	0.1614
Parkinsons	0.1072	0.1138	0.2303	0.1182	0.2123	0.2220
SCADI	0.1147	0.1155	0.1408	0.1252	0.1511	0.1322
Ecoli	0.1180	0.1270	0.2078	0.1388	0.2747	0.2145
Dermatology	0.1209	0.1298	0.2349	0.1320	0.2465	0.2956
Banknote	0.2598	0.3066	4.1260	0.3005	4.6558	4.3243
Dim1000	0.1283	0.1284	0.6115	0.1680	0.6943	0.6855
BinaryAlphadigs	0.2228	0.2904	1.2954	0.2447	1.3643	1.2808
Olivetti Faces	0.0936	0.1156	0.2244	0.1754	0.2287	0.2305

表8　正向与逆向K近邻双向操作的运行时间

数据集	BRPMK-DPC	双向操作	时间占比 / %
Jain	0.2780	0.0092	3.31
Flame	0.2007	0.0099	4.93
Spiral	0.1443	0.0082	5.68
Pathbased	0.2134	0.0083	3.89
Compound	0.2642	0.0109	4.13
D31	6.0564	0.3809	6.29
Twomoons	2.6286	0.0676	2.57
ThreeCircle	0.2547	0.0096	3.77
Lineblob	0.2792	0.0087	3.12
Ring	1.8228	0.0388	2.13
Seeds	0.1790	0.0093	5.20
Libras	0.3722	0.0112	3.01
Wine	0.1614	0.009	5.58
Parkinsons	0.2220	0.0099	4.46
SCADI	0.1322	0.0063	4.77
Ecoli	0.2145	0.0127	5.92
Dermatology	0.2956	0.0140	4.74
Banknote	4.3243	0.0700	1.62
Dim1000	0.6855	0.0249	3.63
BinaryAlphadigs	1.2808	0.0813	6.35
Olivetti Faces	0.2305	0.0091	3.95

5　结　论

本文提出了一种基于双向代表点和相互K近邻的密度峰值聚类算法 BRPMK-DPC. 该算法主要设计了一种双向代表点的局部密度计算方法, 能够在密度分布不均的集群中找到正确的聚类中心. 此外, 本文还提出了一种基于相互K近邻的剩余点分配方法, 该分配方法相比 FKNN-DPC、LF-DPC 更具自适应性. 经过大量的实验测试, BRPMK-DPC 明显优于原始 DPC 以及衍生类 DPC 算法, 同时聚类效果也超越经典的划分聚类算法K-means 和密度聚类算法 DBSCAN. 本文算法具有显著优势, 应用场景探索和继续优化算法性能是接下来的研究重点. 一方面, 考虑探索 BRPMK-DPC 算法在医学图像分割、推荐系统、客户细分等领域中的应用潜力, 尤其是在高维数据或动态数据集上的表现; 另一方面, 针对原始 DPC 及衍生类 DPC 算法在处理大规模数据集时性能不佳的问题, 如计算大规模数据集的距离矩阵比较耗时、剩余点分配策略效率不高等, 未来进一步考虑采用并行计算或分布式处理方法继续优化算法效率, 显著提高算法的实际应用能力和执行效率.

参考文献 (References)

[1] Frey B J, Dueck D. Clustering by passing messages between data points[J]. Science, 2007, 315(5814): 972-976.

[2] Borlea I D, Precup R E, Borlea A B, et al. A unified form of fuzzy C-means and K-means algorithms and its partitional implementation[J]. Knowledge-Based Systems, 2021, 214: 106731.

[3] Murtagh F, Contreras P. Algorithms for hierarchical clustering: An overview, II[J]. WIREs Data Mining and Knowledge Discovery, 2017, 7(6): e1219.

[4] Bai L, Cheng X Q, Liang J Y, et al. Fast density clustering strategies based on the k-means algorithm[J]. Pattern Recognition, 2017, 71: 375-386.

[5] Zhao J, Tang J J, Fan T H, et al. Density peaks clustering based on circular partition and grid similarity[J]. Concurrency and Computation: Practice and Experience, 2020, 32(7): e5567.

[6] Yang M S, Chang-Chien S J, Nataliani Y. Unsupervised fuzzy model-based Gaussian clustering[J]. Information Sciences, 2019, 481: 1-23.

[7] Yin H, Benson A R, Leskovec J, et al. Local higher-order graph clustering[C]. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Halifax, 2017: 555-564.

[8] Li Y, Chu X Q, Tian D, et al. Customer segmentation using K-means clustering and the adaptive particle swarm optimization algorithm[J]. Applied Soft Computing, 2021, 113: 107924.

[9] Huang L, Ruan S, Denœux T. Application of belief functions to medical image segmentation: A review[J]. Information Fusion, 2023, 91: 737-756.

[10] Kolhe L, Jetawat A K, Khairnar V. Robust product recommendation system using modified grey wolf optimizer and quantum inspired possibilistic fuzzy C-means[J]. Cluster Computing, 2021, 24(2): 953-968.

[11] Cai Q, Gong M G, Ma L J, et al. Greedy discrete particle swarm optimization for large-scale social network clustering[J]. Information Sciences, 2015, 316: 503-516.

[12] Nie F P, Li Z H, Wang R, et al. An effective and efficient algorithm for K-means clustering with new formulation[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(4): 3433-3443.

[13] Schubert E, Sander J, Ester M, et al. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN[J]. ACM Transactions on Database Systems, 2017, 42(3): 1-21.

[14] Shi D M, Wang J, Cheng D S, et al. A global-local affinity matrix model via EigenGap for graph-based subspace clustering[J]. Pattern Recognition Letters, 2017, 89: 67-72.

[15] Asheri H, Hosseini R, Araabi B N. A new EM algorithm for flexibly tied GMMs with large number of components[J]. Pattern Recognition, 2021, 114: 107836.

[16] Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191): 1492-1496.

[17] Du M J, Ding S F, Jia H J. Study on density peaks clustering based on k-nearest neighbors and principal component analysis[J]. Knowledge-Based Systems,

2016, 99: 135-145.

[18] Xie J Y, Gao H C, Xie W X, et al. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors[J]. Information Sciences, 2016, 354: 19-40.

[19] 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.

[20] Ren C H, Sun L F, Yu Y, et al. Effective density peaks clustering algorithm based on the layered K-nearest neighbors and subcluster merging[J]. IEEE Access, 2020, 8: 123449-123468.

[21] Wu C R, Lee J, Isokawa T, et al. Efficient clustering method based on density peaks with symmetric neighborhood relationship[J]. IEEE Access, 2019, 7: 60684-60696.

[22] 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.)

[23] Li Z J, Tang Y C. Comparative density peaks clustering[J]. Expert Systems with Applications, 2018, 95: 236-247.

[24] Parmar M, Wang D, Zhang X F, et al. REDPC: A residual error-based density peak clustering algorithm[J]. Neurocomputing, 2019, 348: 82-96.

[25] Wang Y Y, Yang Y L. Relative density-based clustering algorithm for identifying diverse density clusters effectively[J]. Neural Computing and Applications, 2021, 33(16): 10141-10157.

[26] Zhao J, Wang G, Pan J S, et al. Density peaks clustering algorithm based on fuzzy and weighted shared neighbor for uneven density datasets[J]. Pattern Recognition, 2023, 139: 109406.

[27] Xie J Y, Liu X L, Wang M Z. SFKNN-DPC: Standard deviation weighted distance based density peak clustering algorithm[J]. Information Sciences, 2024, 653: 119788.

[28] Yan H, Wang M Z, Xie J Y. ANN-DPC: Density peak clustering by finding the adaptive nearest neighbors[J]. Knowledge-Based Systems, 2024, 294: 111748.

[29] 赵嘉, 姚占峰, 吕莉, 等. 基于相互邻近度的密度峰值聚类算法[J]. 控制与决策, 2021, 36(3): 543-552.
(Zhao J, Yao Z F, Lyu L, et al. Density peaks clustering based on mutual neighbor degree[J]. Control and Decision, 2021, 36(3): 543-552.)

[30] Xu X, Ding S F, Xu H, et al. A feasible density peaks clustering algorithm with a merging strategy[J]. Soft Computing, 2019, 23(13): 5171-5183.

[31] 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.
(Chen M, You Y Y X, Wei L L, et al. A density peaks clustering algorithm based on low density score[J]. Control and Decision, 2025, 40(5): 1599-1609.)

[32] Cheng D D, Zhu Q S, Huang J L, et al. Natural neighbor-based clustering algorithm with density peeks[C]. 2016 International Joint Conference on Neural Networks. Vancouver, 2016: 92-98.

[33] Liu Y H, Ma Z M, Yu F. Adaptive density peak clustering based on K-nearest neighbors with aggregating strategy[J]. Knowledge-Based Systems, 2017, 133: 208-220.

[34] Wang Y Z, Wang D, Pang W, et al. A systematic density-based clustering method using anchor points[J]. Neurocomputing, 2020, 400: 352-370.

[35] Xu T F, Jiang J H. A Graph Adaptive Density Peaks Clustering algorithm for automatic centroid selection and effective aggregation[J]. Expert Systems with Applications, 2022, 195: 116539.

[36] 张清华, 周靖鹏, 代永杨, 等. 基于代表点与K近邻的密度峰值聚类算法[J]. 软件学报, 2023, 34(12): 5629-5648.
(Zhao Q H, Zhou J P, Dai Y Y, et al. Density peaks clustering based on representative points and K-nearest neighbors[J]. Journal of Software, 2023, 34(12): 5629-5648.)

[37] Yu D H, Liu G J, Guo M Z, et al. Density peaks clustering based on weighted local density sequence and nearest neighbor assignment[J]. IEEE Access, 2019, 7: 34301-34317.

[38] Ren C H, Sun L F, Gao Y H, et al. Density peaks clustering based on local fair density and fuzzy k-nearest neighbors membership allocation strategy[J]. Journal of Intelligent & Fuzzy Systems, 2022, 43(1): 21-34.

[39] Fränti P, Rezaei M, Zhao Q P. Centroid index: Cluster level similarity measure[J]. Pattern Recognition, 2014, 47(9): 3034-3045.

[40] 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.

[41] Boudane F, Berrichi A. Gabriel graph-based connectivity and density for internal validity of clustering[J]. Progress in Artificial Intelligence, 2020, 9(3): 221-238.

作者简介

任春华 (1989−), 男, 讲师, 博士, 主要研究方向为数据挖掘、聚类算法, E-mail: 418327014@qq.com;

李朝荣 (1976−), 男, 教授, 博士, 主要研究方向为机器视觉、数据挖掘, E-mail: lichaorong88@163.com;

余洋 (1987−), 男, 讲师, 博士, 主要研究方向为数据挖掘、云平台, E-mail: yuyang@swpu.edu.cn.