计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2024,33(8):18−29 [doi: 10.15888/j.cnki.csa.009582]

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

基于局部密度峰和标签传播的最小生成树聚类①

林钰莹[1], 侯新民[1,2,3,4]

[1](中国科学技术大学 大数据学院, 合肥 230026)
[2](中国科学技术大学 数学科学学院, 合肥 230026)
[3](中国科学院 吴文俊数学重点实验室, 合肥 230026)
[4](合肥国家实验室, 合肥 230088)
通信作者: 林钰莹, E-mail: linyuying@mail.ustc.edu.cn

摘　要: 基于最小生成树 (minimum spanning tree, MST) 的聚类算法能够识别具有任意形状的簇, 该算法在如何有效构建最小生成树和识别无效边方面存在不足, 而且易受到噪声点影响. 本文利用密度峰值聚类算法思想的优点来寻找局部密度峰, 局部密度峰在保留原始数据集分布结构的同时, 排除了噪声点, 因此, 将局部密度峰与最小生成树聚类算法相结合, 采用标签传播, 提出了基于局部密度峰和标签传播的最小生成树聚类算法 (DPMST). 该算法采用了局部密度峰之间基于共享邻的距离, 利用局部密度峰之间的邻域信息, 有效构造最小生成树和识别无效边, 使算法能够发现具有复杂结构的簇. 标签传播增强强标签, 削弱弱标签, 以细化错误的标签, 特别是对于边界点以及揭示复杂流形, 能够提高聚类结果的质量. 人工和真实数据集上的实验结果表明, 与经典聚类算法 DPC、MST、K-means、DBSCAN、AP、SC 和 BIRCH 比较, DPMST 算法表现优异.

关键词: 局部密度峰; 最小生成树; 标签传播; 聚类

引用格式: 林钰莹,侯新民.基于局部密度峰和标签传播的最小生成树聚类.计算机系统应用,2024,33(8):18–29. http://www.c-s-a.org.cn/1003-3254/9582.html

Minimum Spanning Tree Clustering Based on Local Density Peak and Label Propagation

LIN Yu-Ying[1], HOU Xin-Min[1,2,3,4]

[1](School of Data Science, University of Science and Technology of China, Hefei 230026, China)
[2](School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China)
[3](Wu Wen-tsun Key Laboratory of Mathematics, Chinese Academy of Sciences, Hefei 230026, China)
[4](Hefei National Laboratory, Hefei 230088, China)

Abstract: Clustering algorithm based on the minimum spanning tree (MST) can identify clusters with arbitrary shapes, but the algorithm has limitations in efficiently constructing a minimum spanning tree and identifying invalid edges and is easily influenced by noise points. This study proposes an MST clustering algorithm based on local density peaks and label propagation (DPMST) by combining the advantages of the density peaks clustering algorithm to find local density peaks and exclude noise points with the MST algorithm. The DPMST algorithm adopts the shared neighbors-based distance between local density peaks and uses the neighborhood information between local density peaks to efficiently construct minimum spanning trees and identify invalid edges, enabling the discovery of clusters with complex structures. Label propagation is used to enhance the strong labels and weaken the weak labels to refine wrong labels, which can improve the quality of clustering results, especially for border region points as well as revealing complex manifolds. The experimental results on several synthetic and real-world datasets show that the DPMST algorithm outperforms classical clustering algorithms DPC, MST, K-means, DBSCAN, AP, SC, and BIRCH.

Key words: local density peak; minimum spanning tree (MST); label propagation; clustering

① 基金项目: 国家自然科学基金 (12071453); 量子通信与量子计算机重大项目 (2021ZD0302902)
收稿时间: 2024-01-24; 修改时间: 2024-02-26; 采用时间: 2024-03-18; csa 在线出版时间: 2024-06-28
CNKI 网络首发时间: 2024-07-03

聚类算法[1,2]根据数据之间的相似性对数据进行分组, 主要目的是最大化属于同一簇的数据之间的相似性和属于不同簇的数据之间的不相似性. 在现实生活中, 聚类是数据挖掘[3]、图像处理[4,5]、社交网络[6]、生物信息学[7]等领域的一个活跃的研究课题, 还在医学等方面具有重要的作用[8]. 针对不同的应用, 已经提出了各种聚类算法, 如基于层次、划分、模型、网格和密度的算法, 文献[9,10]对各种聚类算法进行了详细描述.

国内外也有很多基于图的聚类算法被提出. 例如文献[11,12]通过构造 k 最近邻图, 也有算法[13,14]利用最小生成树 (MST) 进行聚类. Zahn [13]提出的基于最小生成树的聚类算法将不一致边定义为那些权重明显大于子树中邻近边的平均权重的边.

针对不同的应用场景, 已经提出了广泛的聚类算法. 然而, 由于问题的多样性和数据分布的复杂性, 这些算法并不总是令人完全满意. 基于划分的聚类算法是最基本的聚类方法, 其中 K-means 算法[15]是划分聚类算法中的经典代表. 该算法简单易用, 倾向于将数据聚类成球形, 但对于具有任意形状的簇来说显然不足够. 尽管如此, 识别具有任意形状的簇在聚类算法的应用中仍然是一项非常重要的任务. 最小生成树聚类算法不受簇边界的几何变化影响, 因此簇边界的形状对算法性能的影响很小, 这使我们能够克服经典聚类算法通常面临的问题. 基于密度的聚类算法一直是聚类算法研究的重点. DPC 算法[16]是一种非常有影响力的聚类算法, 尽管在各种数据上表现良好, 但它不能正确处理不同部分密度不同的数据集. 尽管许多聚类算法已经被提出许多年, 但对能够处理上述问题的新的聚类算法的需求仍在继续. 为了克服这些问题, 本文对 DPC 算法和 MST 算法进行了改进和扩展, 提出了能够在密度不均匀、形状复杂的数据集上表现优异的聚类算法 DPMST.

对国内外研究现状进行分析, 发现大部分聚类算法在处理具有复杂流形分布的数据集方面存在问题, 而在实际应用中, 复杂流形分布是普遍存在的一种结构, 因此复杂流形数据集的聚类分析具有重要的实际意义. 最小生成树聚类算法可以识别复杂的流形簇, 但功能较弱; 密度峰值聚类 (DPC) 算法功能强大, 但对流形结构的适用性较低.

本文针对最小生成树聚类算法和 DPC 算法存在的问题进行研究, 将密度峰与最小生成树聚类算法相结合, 用标签传播提高聚类质量, 提出了基于局部密度峰和标签传播的最小生成树聚类算法 (DPMST). 该算法主要在以下 3 个方面做了改进: 一是利用 DPC 算法思想的优点来寻找局部密度峰. 局部密度峰是具有局部最大密度的点, 它保留了原始数据集的分布结构,有助于在局部密度峰上有效构建最小生成树, 减少噪声点的影响, 从而提高算法性能; 二是采用基于共享邻的距离权值, 利用局部密度峰之间的邻域信息, 以更合适的方式表示它们之间的距离, 更有效地识别无效边; 三是引入标签传播机制, 提高聚类结果的质量.

1 相关工作

我们提出的算法是在最小生成树聚类算法和 DPC 算法的基础上改进的, 先介绍两种算法, 再进行研究分析.

1.1 最小生成树聚类算法

基于最小生成树的聚类算法流程: 首先构造数据集的无圈加权图, 再创建最小生成树, 计算所有边的均值与方差, 不断去除不一致的边, 直到满足某些终止条件, 则算法结束, 得到一组连通子图即为聚类结果.

最近几年, 在基于 MST 的聚类算法中, Zhou 等提出 AMST 算法[17], 通过使用新的有效性指标来定义任意形状的簇. 在他们的工作中, 有效性指标被用来确定最佳的簇数量并评估最终的簇. 类似的, Halim 等使用新的内部有效性指标[18], 用于改进生成的簇. Saar 等提出的算法[19]的目标是克服去除不一致的边和处理异常值这两个问题, 他们的主要贡献是使用一种被称为临界距离的方法来定义最优簇. Zhong 等[20]结合了 K-means 算法和 MST 算法, 首先利用 K-means 算法将数据集划分为 k 个分区, 然后用最小生成树来构建簇.

理想状态是簇间分离良好, 不一致的边就是最长的边, 但是事实上噪声点的出现使得去掉最长边未必能得到正确的聚类结果. 因此, 在最小生成树聚类中如何有效构建最小生成树、有效识别无效边和降低噪声点的干扰是研究中面临的主要问题.

1.2 密度峰值聚类算法

密度峰值聚类 (DPC) 算法要画出关于局部密度和相对距离的决策图, 找出局部密度和相对距离都较大的数据点, 从而很容易得到簇中心.

DPC 算法至提出后一直流行, 近年来, 许多学者都在努力改进 DPC. Li 等提出的 KS-FDPC 算法[21]采用

了分区合并策略, 通过将数据划分为多个子簇, 可以降低高密度点对后续分配点的影响范围. Zhao 等提出 DPC-FWSN 算法[22]将 k 最近邻与模糊邻域相结合, 得到一个最近邻模糊核函数以此重新定义局部密度. DPC-PPNNN 算法[23]通过引入自然最近邻域和概率传播的思想, 实现了非参数聚类过程.

DPC 算法在许多数据集上表现出色, 具有显著的优点. 然而, 该算法也存在一些局限性. 首先, 其聚类过程相对简单, 主要是通过最近距离原则将每个点分配到相应的簇中心所在的簇, 这使得它在处理复杂数据结构时可能不适用. 其次, DPC 算法仅适用于每个簇都有密度峰且各簇之间密度差异较小的简单数据集. 当数据集具有流形结构时, DPC 算法的适用性会大大降低.

我们尝试利用 DPC 算法的优点和最小生成树聚类算法来设计一种新的聚类算法. 尽管 DPC 算法在全局聚类上可能不够可靠, 但它能够在局部数据子集中实现高纯度聚类. 因此, 本文将使用局部密度峰来改进最小生成树聚类算法.

1.3 分析

最小生成树聚类算法可以识别形状不规则的簇, 问题在于如何有效构建最小生成树、识别无效边和降低噪声点干扰.

密度峰值聚类算法原理简单, 功能强大, 但对复杂数据结构的适用性较低.

我们结合两种算法的优点, 针对上述问题, 做出以下改进.

一是利用 DPC 算法思想的优点来寻找局部密度峰. DPC 算法可以在局部数据子集中实现高纯度. 局部密度峰包含了一定的局部信息, 保留了原始数据集的分布结构. 我们提出用家族树来捕获数据中的潜在结构, 这有助于反映数据中最近邻之间的密度优势. 通过将数据集划分为多个家族树, 可以更好地获得复杂的数据结构. 每个家族树都包含一个局部密度峰, 在此基础上可以有效构造最小生成树, 从而减少噪声点干扰, 提高算法性能.

二是采用基于共享邻的距离权值. 基于共享邻的距离权值充分利用邻域信息, 能够缩小在密集区域的局部密度峰间的距离, 扩大由稀疏区域连接的局部密度峰间的距离, 从而更准确地表示局部密度峰间的距离. 当使用基于共享邻的距离构造最小生成树时, 以一种更合适的方式表示了局部密度峰之间的距离, 更有效地识别无效边.

三是标签传播. 标签传播通过转移矩阵将簇主干的强标签传播到其余点, 并根据得到的标签矩阵重新分配标签. 标签传播可以增强强标签, 削弱弱标签, 以细化错误的标签, 这种改进策略可以提高聚类结果的质量.

2 改进的算法

本节将详细介绍我们提出的算法, 即基于局部密度峰和标签传播的最小生成树聚类算法 (DPMST).

我们将介绍一些重要的概念, 再基于这些概念描述基于局部密度峰和标签传播的最小生成树聚类算法 (DPMST) 的详细流程, 该算法包括 3 个阶段: 识别局部密度峰, 构造最小生成树, 以及标签传播.

2.1 识别局部密度峰

局部密度峰是在局部密度最大的数据点, 识别它要先计算局部密度, 再构造家族树 (family tree), 数据集将被划分为多个家族树, 每个家族树都有一个根节点 (*root*), 即局部密度峰.

局部密度的计算通过下面几个定义说明.

定义 1 (k 最近邻). k 最近邻是距离数据点最近的 k 个邻居的集合. 给定数据点 x_i, $x_i \in X$, 本文使用 $NN_k(x_i)$ 来表示数据点 x_i 在数据集 X 中的 k 最近邻.

定义 2 (相互最近邻 *MNN*). 若数据点 p 属于数据点 q 的 k 最近邻并且数据点 q 属于数据点 p 的 k 最近邻, 那它们互为相互最近邻 (mutual nearest neighbor):

$$MNN(p) = \{q | p \in NN_k(q) \text{ 且 } q \in NN_k(p)\} \tag{1}$$

定义 3 (局部密度函数). 给定数据点 x_i, $x_i \in X$, 局部密度的定义是相互最近邻的个数:

$$\rho(x_i) = |MNN(x_i)| \tag{2}$$

通过局部密度, 再构造家族树. 家族树的定义如下.

定义 4 (家族树). 构造家族树的思想是在 k 个最近邻的意义上找到每个数据点的父节点 $P(x_i)$, 并将没有父节点的数据点识别为树的根节点 (*root*), 即为局部密度峰. 如果点 p 的 k 最近邻中存在密度较高的点, 则选择最近的点 q 作为 p 的父点 (如果有多个这样的点 q, 选择第一个满足条件的); 否则, p 没有父点, p 被定为局部密度峰. 局部密度峰, 也就是家族树的根节点 (*root*) 可以代表所在的家族树上的数据点, 因此又叫代表点.

$$P(x_i)=\begin{cases}\underset{x_j\in higher(x_i)}{\text{argmin}}\ dist_{ij}, & \text{if } higher(x_i)\neq\varnothing\\ \text{none}, & \text{otherwise}\end{cases} \tag{3}$$

$$higher(x_i)=\left\{x_j|x_j\in NN_k(x_i),\rho\left(x_j\right)>\rho(x_i)\right\} \tag{4}$$

其中, $dist$ 是一个 $n\times k$ 矩阵, $dist_{ij}$ 是 x_i 与其第 j 个近邻之间的欧氏距离.

数据集构造家族树的思想是在 k 个最近邻的意义上找到每个数据点的父节点, 并将没有父节点的数据点识别为树的根节点 (*root*), 即局部密度峰. 数据集将被划分为多个家族树, 每个家族树都有一个局部密度峰, 在局部密度峰上构造最小生成树, 降低了噪声点的干扰, 提高了算法的性能.

2.2 构造最小生成树

局部密度峰在数据集上并非均匀分布, 这是由数据集的分布情况决定的. 因此, 使用欧氏距离来衡量局部密度峰之间的距离是不合适的. 我们采用基于共享邻的距离[24,25]权值 (*SW*) 来度量局部密度峰间的距离. 首先需要介绍一些与局部密度峰相关的概念.

定义 5 (局部密度峰的成员 *MLDP*). 局部密度峰, 也就是家族树的根节点 (*root*) 可以代表所在的家族树上的数据点, *MLDP* 是局部密度峰所在家族树上的数据点的集合. 对于局部密度峰 p, 它的成员是被 p 代表的数据点:

$$MLDP(p)=\{a\in X\mid root(a)=p\} \tag{5}$$

定义 6 (局部密度峰的邻域 *NLDP*). 对于一个局部密度峰 h, 它的邻域是 h 所有成员的 k 最近邻的并集:

$$NLDP(h)=\bigcup_{a\in MLDP(h)}NN_k(a) \tag{6}$$

定义 7 (局部密度峰之间的共享邻 *SLDP*). 两个局部密度峰 p 和 q 的共享邻是它们邻域的交集:

$$SLDP(p,q)=NLDP(p)\cap NLDP(q) \tag{7}$$

基于共享邻的距离权值的定义如下.

定义 8 (基于共享邻的距离权值 *SW*). 对于局部密度峰 p 和 q, 它们的基于共享邻的距离权值计算为:

$$SW(p,q)=\begin{cases}\dfrac{d(p,q)}{|SLDP(p,q)|\times\sum\limits_{o\in SLDP(p,q)}\rho(o)}, & \text{if } |SLDP(p,q)|\neq 0\\ \max d\times(1+d(p,q)), & \text{otherwise}\end{cases} \tag{8}$$

其中, $d(p,q)$ 是局部密度峰 p 和 q 之间的欧氏距离, $\max d$ 为所有局部密度峰之间欧氏距离的最大值.

基于共享邻的距离权值 *SW* 的定义可以看出, 在密集区域紧密相连的局部密度峰的 *SLDP* 集合中会有许多数据点, 局部密度之和也很大, 所以分母很大, 能够缩短由密集区域紧密相连的局部密度峰间的距离; 在稀疏区域, 分隔的局部密度峰的 *SLDP* 集合中会有很少的数据点, 局部密度之和也很小, 所以分母很小, 能够放大被稀疏区域分隔的局部密度峰间的距离.

利用局部密度峰之间的邻域信息, 能够以一种更合适的方式表示了局部密度峰间的距离. 从下面的例子中可以清楚地看出这一点.

从图 1 中可以看出, 点 P 与点 H 的距离较近, 但并不在同一个簇中, 点 H 与点 O 距离较远, 但在同一簇中. 在数据经过归一化处理之后, 计算欧氏距离和基于共享邻的距离权值 *SW*.

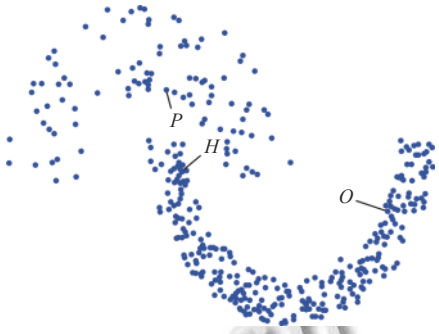

图 1 示例

从表 1 可以直观地看出, 点 P 与点 H 的距离被放大了, 点 H 与点 O 的距离被缩小了. 在欧氏距离中, 点 H 与点 O 距离大约是点 P 与点 H 距离的两倍, 而在 *SW* 距离中, 点 H 与点 O 距离大约是点 P 与点 H 距离的二分之一. *SW* 距离可以缩短由密集区域紧密相连的局部密度峰间的距离, 放大被稀疏区域分隔的局部密度峰间的距离, 以一种更合适的方式表示了局部密度峰间的距离, 用这个距离权值能更有效地构造最小生成树, 识别无效边.

表 1 距离对比

类型	(P, H)	(H, O)
欧氏距离	0.9896	2.0582
*SW*距离	2.2992	1.1251

通过 *Prim* 算法用 *SW* 距离权值来构造最小生成树. 然后, 反复删除权值最大边, 每删除一次, 簇的数目增加一个 (若目标簇数 c, 需要删除 c–1 条边), 得到给

定数目的簇. 举例如图 2, 假设目标簇数为 3, 该最小生成树需要删除 2 条边, 按权值需要删除 BC 边和 CF 边, 删除后可以看出 BE 在同一簇, AC 在同一簇, DF 在同一簇, 这样就分成了 3 个簇.

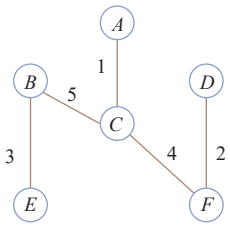

图 2 最小生成树例子

数据点根据所在的簇分配相应的标签, 所有的数据点都有一个标签. 每个局部密度峰和其邻居被定义为一个簇的主干部分, 主干的标签是强标签, 确定之后保持不变, 某些位于边界和重叠区域的数据的标签是弱标签, 之后可能会再变动标签.

2.3 标签传播

这一步的目的是使用强标签的簇主干为某些弱标签的数据点重新分配标签. 标签传播这一步要先把局部密度峰的确定性标签分配给其邻居, 每个局部密度峰和其邻居被定义为一个簇的主干部分, 主干的标签是强标签, 确定之后保持不变, 然后簇主干部分的标签通过转移矩阵传播到其余点, 根据得到的标签矩阵为某些位于边界和重叠区域的数据重新分配标签. 标签传播增强强标签, 削弱弱标签, 以细化错误的标签, 提高聚类质量. 具体流程如下.

首先, 计算转移矩阵, 我们定义了一个加权图 $G=(V, E, W)$, 其中顶点 V 对应数据集 X, 权值为:

$$W_{i,j}=\exp\left(\frac{d\left(x_i,x_j\right)^2}{\sigma_{i,j}^2}\right) \tag{9}$$

其中, $\sigma_{i,j}$表示x_i和x_j的 k 近邻之间的平均距离.

上述的权值是使用一个全连通的图来定义的. 一般来说, 全连通图并不继承数据点的自然结构, 使用全连通图也会增加计算的复杂度. 因此, 一个更好的方法是连接一个基于 k 最近邻的图, 其中每个节点只连接到它的邻居, 节省计算成本.

基于 k 最近邻的权值:

$$W_{i,j}^{(NN_k)}=\begin{cases}W_{i,j}, & \text{if } x_j\in NN_k(x_i)\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

然后通过对权值进行归一化, 也就是权值除以行和, 定义一个转移矩阵:

$$P_{i,j}=\frac{W_{i,j}^{(NN_k)}}{\sum_{h\in V}W_{i,h}^{(NN_k)}} \tag{11}$$

将数据分为有标记点和未标记点. 已标记的数据点指的是定义为簇主干的数据点, 也就是局部密度峰和其邻居, 其余的数据点被认为是未标记的. 请注意, 每个数据点都与一个标签相关联. 初始标签矩阵定义为:

$$Y_0=\left[Y^{(l)};Y^{(u)}\right]\in R^{n\times c} \tag{12}$$

其中, $Y^{(l)}$为标记数据点的标签矩阵, $Y^{(u)}$为未标记数据点的标签矩阵. 如果x_i属于第 a 个簇, 则$Y_{i,a}^{(l)}=1$, 否则将其设置为 0. n 是数据集 X 的数据点数目, c 是簇个数. 初始标签矩阵中有标记的数据点是局部密度峰与其邻居, 也就是簇的主干部分.

在每次迭代中, 未标记的点更新其标签如下:

$$Y_{t+1}=F_t*Y_t \tag{13}$$

$$F_0=P,\ F_{t+1}=\begin{cases}P, & \text{if } t=0\\ P\left(F_t+Y_tY_t^{\mathrm{T}}\right)P^{\mathrm{T}}, & \text{if } t>0\end{cases} \tag{14}$$

直到标签矩阵不再发生变化, 迭代停止. 在每次迭代中, 那些初始标记的数据点可能会被分配不同的标签. 因此, 它们的标签需要重置如下:

$$Y_{t+1}^{(l)}=Y_0^{(l)} \tag{15}$$

通过上述过程, 簇主干部分的强标签通过转移矩阵传播到其余点, 我们根据最终得到的标签矩阵为某些弱标签的数据点重新分配新标签, 得到最终的聚类结果.

大多数标签分配是单步操作的, 因此如果某个数据点被错误地分配了标签, 由于缺乏细化策略, 这个错误会传播到其他节点. 为了解决这个问题, 我们提出通过迭代过程更新标签. 数据点的标签根据标签传播后的标签矩阵进行更新, 给某些位于边界和重叠区域的数据重新分配真标签. 这种策略将数据的局部结构与标签空间相关性相结合, 特别是对边界点以及流形区域, 可以提高聚类结果的质量.

以上就是基于局部密度峰和标签传播的最小生成树聚类算法的整个流程.

基于局部密度峰和标签传播的最小生成树聚类 (DPMST) 的具体步骤如算法 1 所示.

算法 1. 基于局部密度峰和标签传播的最小生成树聚类

输入: 数据集$X=\{x_1,x_2,\cdots,x_n\}$, 最近邻数量 k, 簇个数 c.
输出: 聚类标签$Y=\{y_1,y_2,\cdots,y_n\}$.

for each $x_i \in X$ **do**
  Calculate $MNN(x_i)$ (见定义 2)
  Calculate $\rho(x_i)$ (见定义 3)
end for
$root \leftarrow \varnothing$
for each $x_i \in X$ **do**
  Compute $P(x_i)$ (见定义 4)
  If $P(x_i)$=none **then**
    $root \leftarrow root \cup \{x_i\}$
end for
root 即局部密度峰
for each 局部密度峰 p **do**
  Compute $MLDP(p)$ (见定义 5)
  Compute $NLDP(p)$ (见定义 6)
end for
for each pair of 局部密度峰 p and q **do**
  Compute $SLDP(p, q)$ (见定义 7)
  Compute $SW(p, q)$ (见定义 8)
end for
edge←构造最小生成树 (局部密度峰, *SW*)
k=1
while $k<c$ **do**
  Cut the longest edge
  $k=k+1$
end while
Assign labels according to the cluster in which they are located
Assign the label of cluster centers to their nearest neighbors to form cluster backbones
Compute transition matrix P (见式 (11))
for 迭代次数 t=1 to T **do**
  Compute label matrix Y_t (见式 (13)–式 (15))
end for
Reassign new labels to certain data points according to the resulting label matrix

2.4 DPMST 算法分析

DPMST 算法第 1 步识别局部密度峰: 计算局部密度, 为数据集构造家族树, 在 k 个最近邻的意义上找到每个数据点的父节点, 并将没有父节点的数据点识别为树的根节点 (*root*), 即局部密度峰. 局部密度峰包含了一定的局部信息, 保留了原始数据集的分布结构. 用家族树来捕获数据中的潜在结构, 可以反映数据中最近邻之间的密度优势. 数据集被划分为多个家族树, 有助于获得复杂的数据结构. 一个数据集将被划分为多个家族树, 每个家族树都有一个局部密度峰, 在局部密度峰上能有效构造最小生成树, 降低噪声点的干扰, 提高算法的性能.

DPMST 算法第 2 步构造最小生成树: 计算局部密度峰之间 *SW* 距离权值, 使用 *SW* 距离权值来构造最小生成树, 反复删除最长边, 直到找到给定数量的簇, 根据所在的簇分配标签. 使用基于共享邻的距离构造最小生成树, 能够更有效地识别无效边.

DPMST 算法第 3 步标签传播: 把局部密度峰的确定性标签分配给其邻居, 即簇的主干部分, 计算转移矩阵, 然后簇主干部分的强标签通过转移矩阵传播到其余点, 数据点的标签根据标签传播后的标签矩阵进行更新. 这种策略可以提高聚类结果的质量.

3 实验

在本节中, 我们将详细介绍并分析实验结果, 以证明基于局部密度峰和标签传播的最小生成树聚类算法 (DPMST) 的有效性.

3.1 实验指标和设置

我们选择了 7 种经典的聚类算法与 DPMST 算法比较, 包括 DPC[16]、MST[26]、K-means[15]、DBSCAN[27]、AP[28]、谱聚类 (SC)[29]和 BIRCH[30]. 为了更客观地反映各种算法的实际结果, 我们进行参数调优, 从而确保比较的是它们的最佳性能.

为验证我们的 DPMST 算法在各种数据集上的有效性, 我们选取了 3 种评价指标, 评价指标均保留 4 位小数. 本文采用了调整兰德指数 (ARI)、标准化互信息 (NMI) 和 Fowlkes-Mallows 指数 (FMI) 对聚类算法的有效性进行评估, 3 种评价指标的满分为 1, 指标值与聚类效果成正比.

3.2 实验结果

在本节中, 我们选取几个广泛应用于检验聚类算法性能的人工数据集进行分析. 这些数据集具有点、簇分布与数目上的差异, 能够对比各种聚类算法在不同场景的效果. 我们选择的人工和真实数据集的情况如表 2 所示.

真实和人工数据集源于 UCI network data repository 和 GitHub 中. 真实数据集大多来源于生物和医学领域.

8 个人工数据集上的聚类评价指标得分在表 3 显示, 得分最高的评价指标用粗体标记, 可以看到, 在大多数情况下, 我们的算法 DPMST 获得的得分最高.

8 个人工数据集都是便于可视化的 2 维数据, 因此

本文对 8 个人工数据集都以图表的形式展示了聚类结果, 绘制图所使用软件是 PyCharm.

表 2 实验数据集

数据集分类	数据集名称	样本量	属性	簇
人工数据集	spiral	312	2	3
	jain	373	2	2
	2d-4c-no9	876	2	4
	Aggregation	788	2	7
	Compound	399	2	6
	zelink1	299	2	3
	Half kernel	1 000	2	2
	Moon	514	2	4
真实数据集	glass	214	9	7
	Thy	215	5	3
	Wine	178	13	3
	ecoli	336	7	8
	zoo	101	17	7
	dermatology	366	34	6
	divorce	170	54	2
	Breast	699	10	2

如图 3 所示, spiral 是一个具有 3 个簇且低密度的数据集, 数据点构成了彼此交叉缠绕的螺旋形曲线.可以看到我们的算法 DPMST、MST、DBSCAN 和谱聚类算法在此数据集上表现良好, 而 K-means 算法仅考虑数据点之间的距离特性, 无法处理环绕的流线形数据, 效果最差.

jain 是一个具有 2 个簇的数据集, 图 4 中数据点形成上下两环形, 上面的簇密度低, 下面的簇密度高. 可以看到只有我们的算法 DPMST 能够正确识别所有簇, 密度聚类算法性能不佳的原因是两个簇的密度差异太大. 而 K-means 不适合用于具有流线形状的数据集.

在图 5 中, 我们可以看到 2d-4c-no9 数据集有 4 个簇, 其中一个簇有非常高的密度. 我们的算法 DPMST 和 DPC 算法较为成功地检测到了所有的簇, 聚类效果最佳. K-means 和 BIRCH 效果稍微差一些.

我们的算法 DPMST 和其他算法在 Aggregation 上的表现如图 6 所示. 可以看出, 我们的算法可以正确地检测到该数据集, K-means 不能这样做, 因对 K-means 来说 Aggregation 数据集有些复杂.

图 7 显示了 Compound 数据集上的结果. 我们的算法与 MST 算法各具优势, 可以正确地检测大部分簇, 还可以识别噪声点. DBSCAN 擅长识别噪声, 但它也将许多数据点错误地归类为噪声. DPC 和 K-means 不能识别有噪声的簇.

表 3 人工数据集上聚类指标对比

数据集	指标	DPMST (ours)	DPC	MST	K-means	DBSCAN	AP	SC	BIRCH
spiral	ARI	**1.0000**	0.8293	**1.0000**	0.0049	**1.0000**	0.1516	**1.0000**	0.0204
	NMI	**1.0000**	0.8167	**1.0000**	0.0013	**1.0000**	0.4348	**1.0000**	0.0291
	FMI	**1.0000**	0.8860	**1.0000**	0.3285	**1.0000**	0.3325	**1.0000**	0.3526
jain	ARI	**1.0000**	0.7055	0.8731	0.2831	0.9053	0.1250	0.9800	0.5071
	NMI	**1.0000**	0.6447	0.6130	0.3458	0.7563	0.3911	0.9750	0.5003
	FMI	**1.0000**	0.8779	0.9482	0.6814	0.9619	0.3953	0.9980	0.7869
2d-4c-no9	ARI	**0.9959**	**0.9959**	0.3922	0.9692	0.3972	0.3127	0.3972	0.9918
	NMI	**0.9930**	**0.9930**	0.6494	0.9512	0.6666	0.5946	0.6666	0.9874
	FMI	**0.9974**	**0.9974**	0.7194	0.9806	0.7242	0.5140	0.7242	0.9948
Aggregation	ARI	**0.9949**	0.7361	0.8089	0.7071	0.7338	0.3915	0.9898	0.8230
	NMI	**0.9915**	0.8817	0.8894	0.8264	0.8359	0.7545	0.9851	0.9225
	FMI	**0.9960**	0.7928	0.8652	0.7694	0.8189	0.5385	0.9920	0.8632
Compound	ARI	0.9247	0.6171	**0.9427**	0.4044	0.8713	0.3965	0.5282	0.8194
	NMI	**0.9228**	0.8028	0.8383	0.6266	0.8930	0.6936	0.7622	0.8386
	FMI	0.9445	0.7087	**0.9570**	0.5338	0.9078	0.5351	0.6378	0.8709
zelink1	ARI	**1.0000**	0.2141	0.6500	0.0524	0.5300	0.2835	0.0555	0.3028
	NMI	**1.0000**	0.3450	0.4600	0.1601	0.4600	0.5691	0.1637	0.4332
	FMI	**1.0000**	0.4923	0.6044	0.4065	0.6044	0.4874	0.4045	0.6343
Half kernel	ARI	**1.0000**	0.0954	0.9841	0.1006	0.7416	0.0900	0.9500	0.0399
	NMI	**1.0000**	0.1144	0.9632	0.4003	0.7769	0.3647	0.9875	0.0316
	FMI	**1.0000**	0.6068	0.9920	0.5038	0.8610	0.2998	0.9900	0.5304
Moon	ARI	**1.0000**	0.3577	0.6322	0.2354	0.4500	0.2597	0.2098	0.3268
	NMI	**1.0000**	0.5149	0.8271	0.3533	0.3800	0.6259	0.3221	0.3859
	FMI	**1.0000**	0.5329	0.7744	0.4282	0.5056	0.4367	0.4105	0.5857

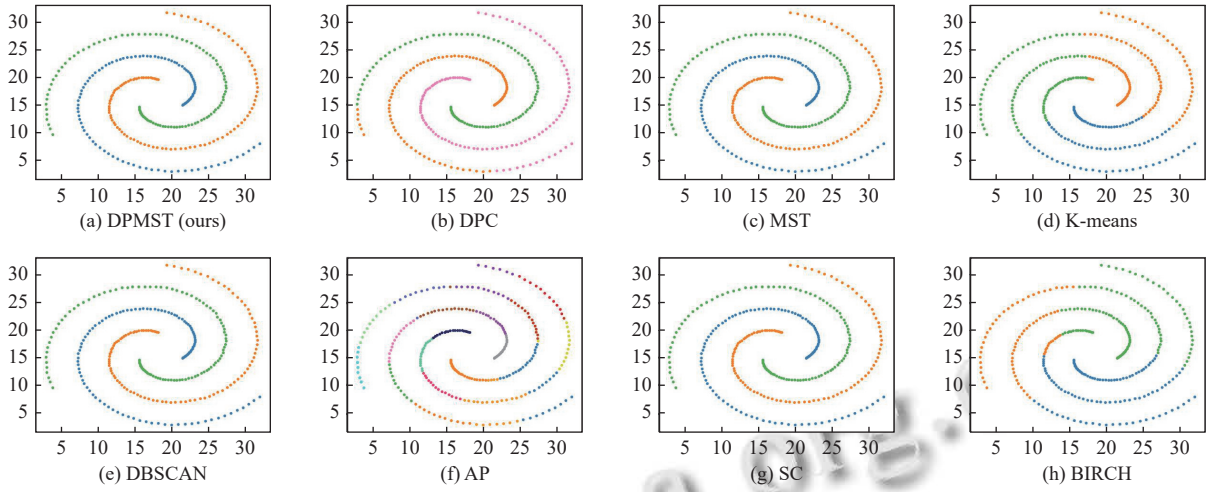

图 3 spiral 数据集聚类结果

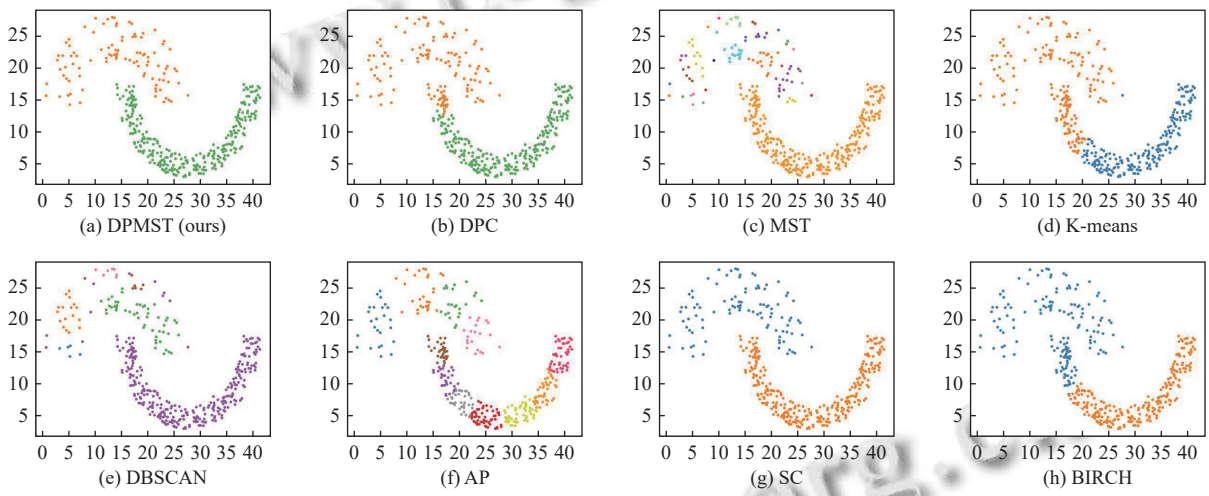

图 4 jain 数据集聚类结果

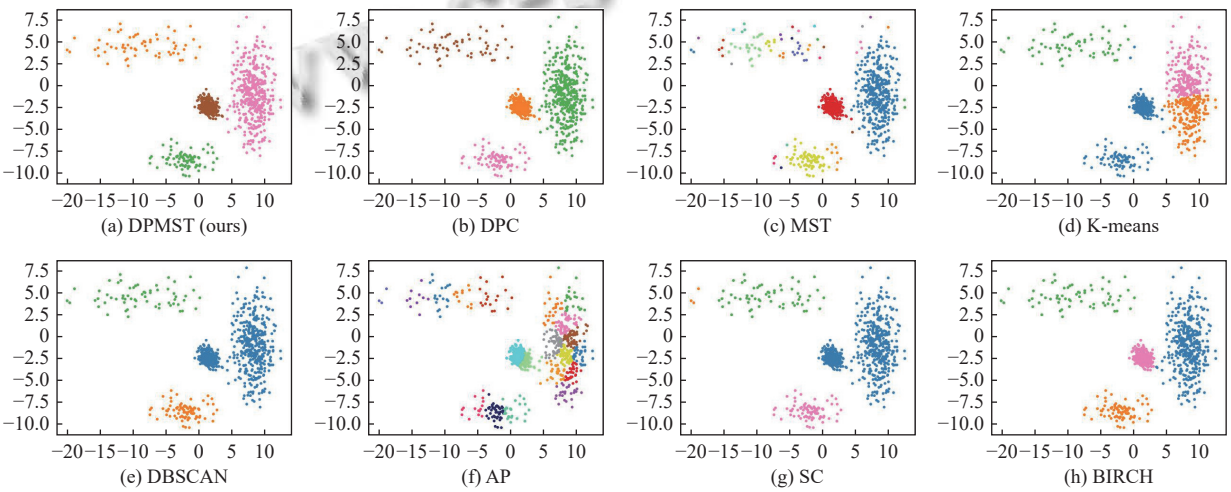

图 5 2d-4c-no9 数据集聚类结果

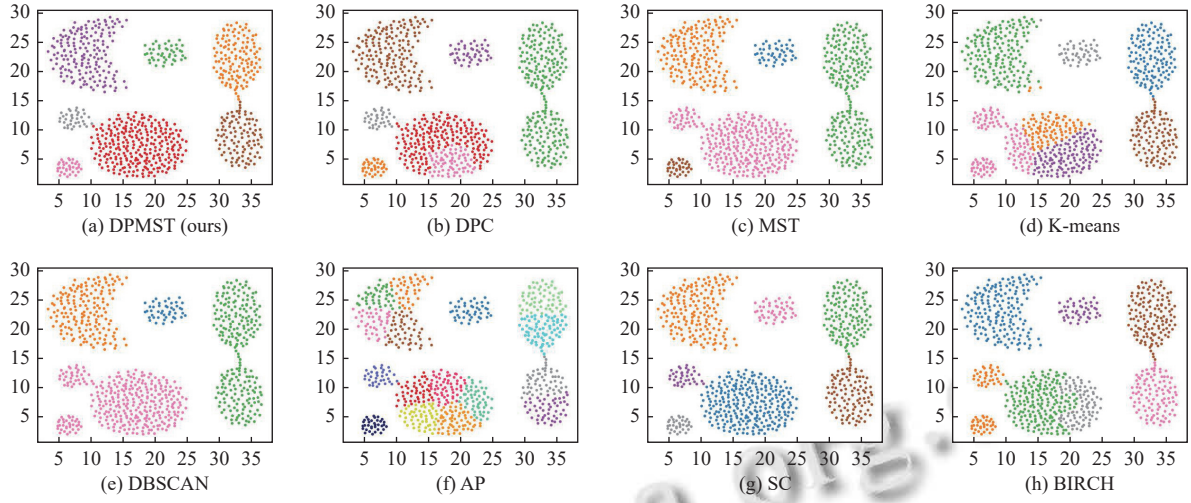

图 6 Aggregation 数据集聚类结果

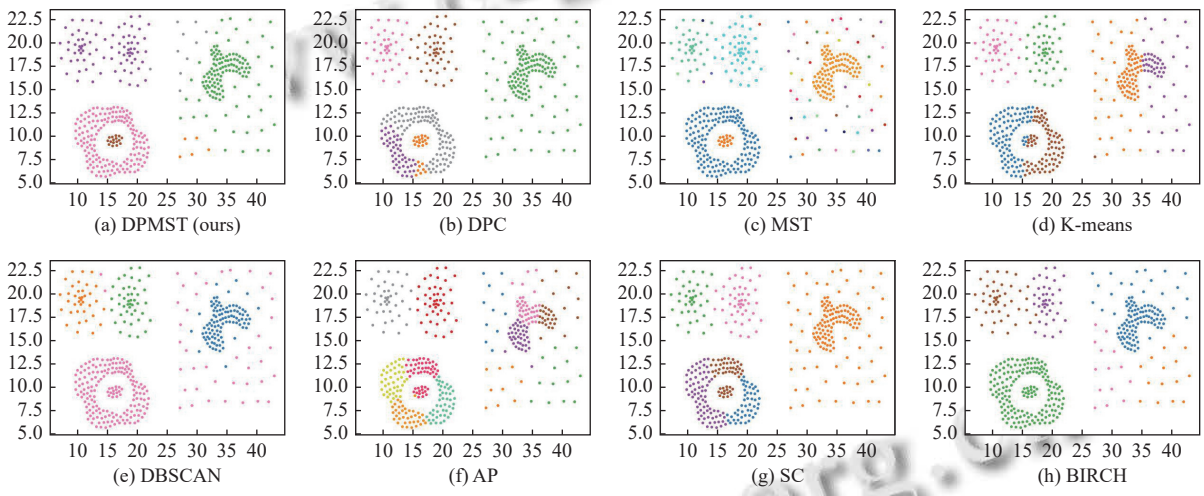

图 7 Compound 数据集聚类结果

在图 8 中, 我们可以看到 zelink1 数据集有 3 个簇, 形状像同心圆, 最中间的圆心密度高, 其余两簇密度逐渐降低. 我们的算法 DPMST 成功地检测到了所有的簇, 聚类效果最佳.

在图 9 中, Half kernel 是一个具有两个簇且高密度的数据集, 数据点形成两个弧形簇. 在使用 8 种聚类算法对它们进行聚类时, 我们的算法 DPMST 在 Half kernel 数据集上具有最佳的聚类结果, 其次是 MST 算法.

如图 10 所示, Moon 数据集有 4 个簇, 数据点形成 4 个弧形簇. 我们的算法 DPMST 可以正确识别所有簇, 而其余密度聚类算法将个别点识别为了离群点, 因此没有获得较好表现.

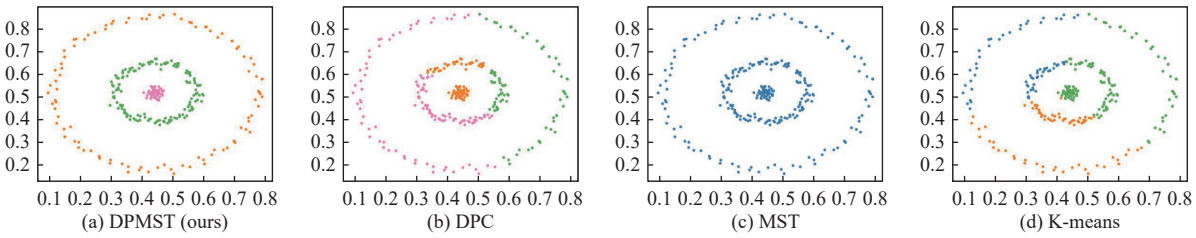

图 8 zelink1 数据集聚类结果

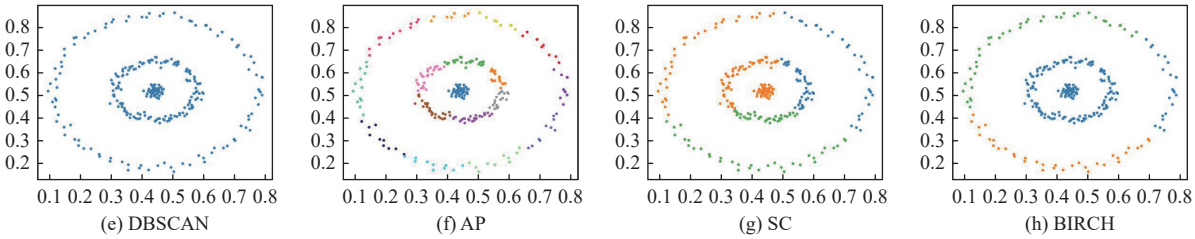

图 8 zelink1 数据集聚类结果 (续)

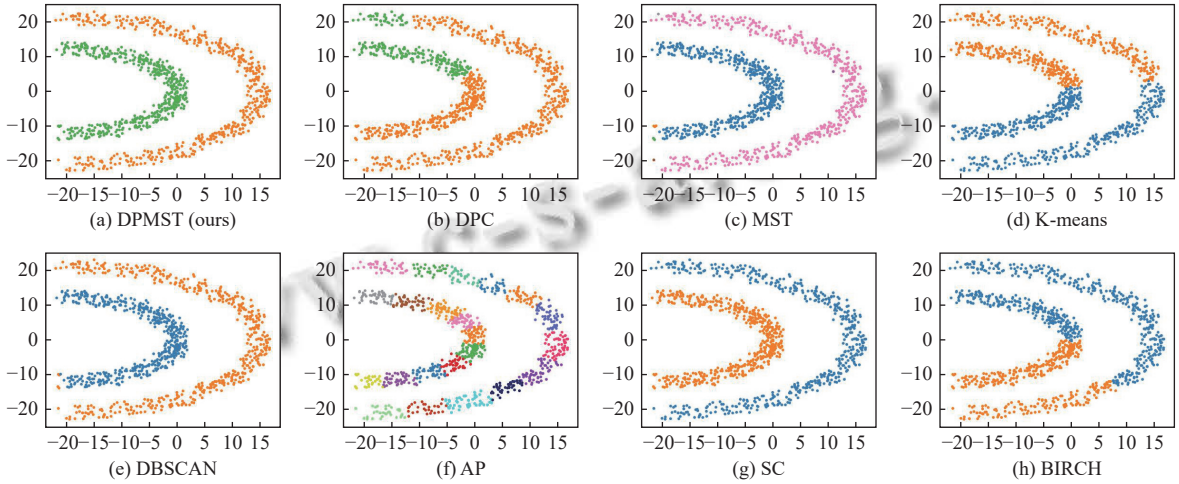

图 9 Half kernel 数据集聚类结果

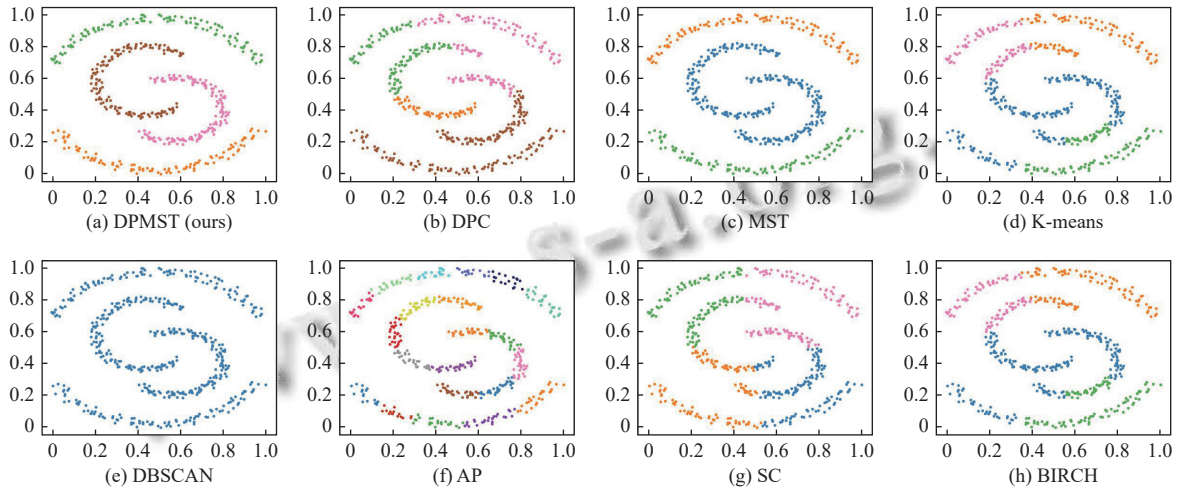

图 10 Moon 数据集聚类结果

从 8 个人工数据集的结果可以看出, 本文提出的算法具有非常好的性能, 能聚类任意形状的簇, 具有较好的鲁棒性, 通过调节参数可以对密度差异大的数据实现较好的聚类, 对流形数据也有非常好的效果.

为证明我们算法 DPMST 在实践中的有效性, 进一步选取 8 个真实数据集开展实验, 这些数据集的数量、属性以及簇数各不相同, 相较于前面的人工数据集更加复杂. 通过表 4 中的实验结果可以发现, 在密度不均匀、形状复杂的真实数据集上表现优异的聚类算法 DPMST 大部分情况下优于其他 7 种聚类算法. 因此, 我们认为 DPMST 算法具有一定的有效性和实用性, 可以在实际应用 (例如生物和医学领域) 中发挥作用.

表 4 真实数据集上聚类指标对比

数据集	指标	DPMST (ours)	DPC	MST	K-means	DBSCAN	AP	SC	BIRCH
glass	ARI	**0.2500**	0.1365	0.2342	0.2358	0.0429	0.1944	0.0673	0.2475
	NMI	0.3976	0.2829	**0.4569**	0.3973	0.0698	0.4042	0.1936	0.3686
	FMI	**0.5460**	0.3572	0.5259	0.4238	0.5012	0.3527	0.4962	0.4732
Thy	ARI	**0.7842**	0.1614	0.2410	0.1070	0.4495	0.1705	0.0313	0.6207
	NMI	**0.7080**	0.2438	0.2444	0.2034	0.4200	0.3851	0.0422	0.5821
	FMI	**0.9038**	0.5667	0.7597	0.5468	0.7362	0.4280	0.7318	0.8502
Wine	ARI	**0.8011**	0.3715	0.0146	0.3711	0.4064	0.2223	0.0004	0.3684
	NMI	**0.7957**	0.4193	0.3652	0.4288	0.4130	0.3757	0.0111	0.4161
	FMI	**0.8678**	0.5834	0.1053	0.5835	0.6626	0.4191	0.5783	0.5821
ecoli	ARI	**0.5957**	0.4540	0.0407	0.4062	0.4500	0.2693	0.4060	0.4858
	NMI	**0.6174**	0.5809	0.1556	0.6194	0.4320	0.5630	0.5536	0.6147
	FMI	**0.7077**	0.5920	0.5280	0.5434	0.5197	0.4313	0.5447	0.6092
zoo	ARI	**0.7998**	0.3176	0.7614	0.6741	0.5032	0.5842	0.6868	0.6451
	NMI	0.7805	0.5771	**0.8618**	0.7545	0.6018	0.7790	0.7983	0.7430
	FMI	**0.8453**	0.4606	0.8213	0.7464	0.6708	0.6774	0.7563	0.7233
dermatology	ARI	**0.8270**	0.0657	0.0006	0.0347	0.1952	0.1527	0.0053	0.0336
	NMI	**0.8868**	0.1868	0.4455	0.0976	0.3736	0.3706	0.0402	0.1042
	FMI	**0.8682**	0.2851	0.0194	0.2149	0.4770	0.2748	0.4269	0.2220
divorce	ARI	**0.9076**	**0.9076**	0.0049	**0.9076**	0.4956	0.3072	0.0003	**0.9076**
	NMI	**0.8621**	**0.8621**	0.2474	**0.8621**	0.5151	0.4776	0.0114	**0.8621**
	FMI	**0.9536**	**0.9536**	0.0700	**0.9536**	0.7584	0.5539	0.7010	**0.9536**
Breast	ARI	**0.8557**	0.0147	0.0001	0.0204	0.0227	0.0014	0.5400	0.4027
	NMI	**0.7773**	0.0025	0.1798	0.0041	0.0554	0.1344	0.4500	0.4037
	FMI	**0.9336**	0.5775	0.0106	0.5854	0.6944	0.0900	0.7400	0.7370

4 结论

本文提出了一种聚类算法 DPMST. 算法过程有以下步骤: 首先, 计算局部密度, 为数据集构造家族树, 识别局部密度峰; 其次, 计算 *SW* 距离权值, 构造最小生成树, 反复删除最长边, 直到找到给定数量的簇, 根据所在的簇分配标签; 最后, 标签传播, 计算转移矩阵, 簇主干的标签通过转移矩阵传播到其余点, 数据点的标签根据标签传播后的标签矩阵进行更新.

本文提出的改进后的算法 DPMST, 利用 DPC 算法思想的优点来寻找局部密度峰, 在局部密度峰上能有效构造最小生成树, 降低噪声点的干扰. 该算法构造最小生成树时使用基于共享邻的距离, 其以一种更合适的方式表示了局部密度峰之间的距离, 更有效地识别无效边. 标签传播将数据的局部结构与标签空间相关性相结合, 可以提高聚类结果的质量.

人工和真实数据集的实验结果表明, DPMST 算法在密度不均匀、形状复杂的数据集上能够达到较为理想的聚类效果, 通过调节参数可以达到不同的聚类效果, 算法性能良好, 总体聚类性能优于其他对比算法, 具有一定的实用价值. 但是, DPMST 算法仍需要输入参数, 这意味着该算法不能排除人为因素的干扰, 这对该算法在实际问题中的应用带来了一定的挑战, 因此, 在后续工作中我们将考虑参数的自适应.

参考文献

1 Michalski RS, Stepp RE. Learning from observation: Conceptual clustering. In: Michalski RS, Carbonell JG, Mitchell TM, eds. Machine Learning. Berlin: Springer, 1983. 331–363. [doi: 10.1007/978-3-662-12405-5_11]

2 Berkhin P. A survey of clustering data mining techniques. In: Kogan J, Nicholas C, Teboulle M, eds. Grouping Multidimensional Data. Berlin: Springer, 2006. 25–71. [doi: 10.1007/3-540-28349-8_2]

3 Han JW, Kamber M, Pei J. 数据挖掘: 概念与技术. 范明, 孟小峰, 译. 第 3 版, 北京: 机械工业出版社, 2012.

4 杨传慧. 图像聚类及其在图像检索中的应用研究 [硕士学位论文]. 南京: 南京师范大学, 2013.

5 肖宇. 聚类分析及其在图像处理中的应用 [博士学位论文]. 北京: 北京交通大学, 2012.

6 Niu YY, Kong DT, Liu LG, *et al*. Overlapping community detection with adaptive density peaks clustering and iterative partition strategy. Expert Systems with Applications, 2023, 213: 119213. [doi: 10.1016/j.eswa.2022.119213]

7 Petegrosso R, Li ZL, Kuang R. Machine learning and statistical methods for clustering single-cell RNA-sequencing

data. Briefings in Bioinformatics, 2020, 21(4): 1209–1223. [doi: 10.1093/bib/bbz063]
8 唐东明. 聚类分析及其应用研究 [博士学位论文]. 成都: 电子科技大学, 2010.
9 孙吉贵, 刘杰, 赵连宇. 聚类算法研究. 软件学报, 2008, 19(1): 48–61. [doi: 10.3724/SP.J.1001.2008.00048]
10 Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Computing Surveys, 1999, 31(3): 264–323. [doi: 10.1145/331499.331504]
11 Karypis G, Han EH, Kumar V. Chameleon: Hierarchical clustering using dynamic modeling. Computer, 1999, 32(8): 68–75. [doi: 10.1109/2.781637]
12 Franti P, Virmajoki O, Hautamaki V. Fast agglomerative clustering using a K-nearest neighbor graph. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(11): 1875–1881. [doi: 10.1109/TPAMI.2006.227]
13 Zahn CT. Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers, 1971, C-20(1): 68–86. [doi: 10.1109/T-C.1971.223083]
14 Wang XC, Wang XL, Wilkes DM. A divide-and-conquer approach for minimum spanning tree-based clustering. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(7): 945–958. [doi: 10.1109/TKDE.2009.37]
15 Lloyd S. Least squares quantization in PCM. IEEE Transactions on Information Theory, 1982, 28(2): 129–137. [doi: 10.1109/TIT.1982.1056489]
16 Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492–1496. [doi: 10.1126/science.1242072]
17 Zhou RY, Shu LJ, Su Y. An adaptive minimum spanning tree test for detecting irregularly-shaped spatial clusters. Computational Statistics & Data Analysis, 2015, 89: 134–146. [doi: 10.1016/j.csda.2015.03.008]
18 Halim Z, Uzma. Optimizing the minimum spanning tree-based extracted clusters using evolution strategy. Cluster Computing, 2018, 21(1): 377–391. [doi: 10.1007/s10586-017-0868-6]
19 Şaar F, Topcu AE. Minimum spanning tree-based cluster analysis: A new algorithm for determining inconsistent edges. Concurrency and Computation: Practice and Experience, 2022, 34(9): e6717. [doi: 10.1002/cpe.6717]
20 Zhong CM, Malinen M, Miao DQ, *et al*. A fast minimum spanning tree algorithm based on K-means. Information Sciences, 2015, 295: 1–17. [doi: 10.1016/j.ins.2014.10.012]
21 Li C, Ding SF, Xu X, *et al*. Fast density peaks clustering algorithm based on improved mutual K-nearest-neighbor and sub-cluster merging. Information Sciences, 2023, 647: 119470. [doi: 10.1016/j.ins.2023.119470]
22 Zhao J, Wang G, Pan JS, *et al*. Density peaks clustering algorithm based on fuzzy and weighted shared neighbor for uneven density datasets. Pattern Recognition, 2023, 139: 109406. [doi: 10.1016/j.patcog.2023.109406]
23 Zuo WD, Hou XM. An improved probability propagation algorithm for density peak clustering based on natural nearest neighborhood. Array, 2022, 15: 100232. [doi: 10.1016/j.array.2022.100232]
24 程东东. 基于局部核心点的聚类算法与度量研究 [博士学位论文]. 重庆: 重庆大学, 2018.
25 靳文星, 王电钢, 张哲敏. 基于局部密度的最小生成树聚类算法及其在电力大数据的应用. 四川电力技术, 2021, 44(4): 16–19, 49. [doi: 10.16527/j.issn.1003-6954.20210404]
26 VanderPlas J. mst_clustering: Clustering via Euclidean minimum spanning trees. Journal of Open Source Software, 2016, 1(1): 12. [doi: 10.21105/joss.00012]
27 Ester M, Kriegel HP, Sander J, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland: ACM, 1996. 226–231. [doi: 10.5555/3001460.3001507]
28 Fukunaga K, Hostetler L. The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory, 1975, 21(1): 32–40. [doi: 10.1109/TIT.1975.1055330]
29 Shi JB, Malik J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888–905. [doi: 10.1109/34.868688]
30 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]

(校对责编: 孙君艳)