侵蚀聚类

杜明晶[1,2],吴福玉[1,2],李宇蕊[1,2],董永权[1,2]
(1. 江苏师范大学计算机与科学技术学院,江苏徐州 221116;2. 江苏师范大学江苏省高校教育智能技术重点实验室,江苏徐州 221116)

摘 要: 基于密度的聚类是一种经典的聚类分析方法,它能够在不指定类簇数目的情况下发现非球形类簇. 但真实复杂数据集中存在类簇边界模糊、数据密度不均、数据分布复杂等问题. 当前,能够同时应对这三种问题的研究工作相对较少. 对此,本文从自然世界的侵蚀现象中汲取灵感,提出侵蚀聚类(Erosion Clustering,EC)算法. 本算法引入动态密度估计方法和侵蚀策略,逐层识别和剔除位于类簇边界上的数据,进而发现各个类簇潜在的核心区域;采用基于互可达图的聚类方法实现核心区域的聚类;设计基于局部密度峰值的分配方式完成边界数据的划分. 在12个基准数据集上的实验结果表明,EC算法的聚类性能比7种对比算法分别在修正兰德指标、修正互信息、F_1分数上平均提高了96%、53%和36%.

关键词: 密度聚类;聚类分析;密度估计;局部密度峰值;互k近邻;侵蚀策略
基金项目: 国家自然科学基金(No.62006104,No.61872168)
中图分类号: TP181 **文献标识码:** A **文章编号:** 0372-2112(2024)10-3459-13
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20231146

Erosion Clustering

DU Ming-jing[1,2], WU Fu-yu[1,2], LI Yu-rui[1,2], DONG Yong-quan[1,2]
(1. *School of Computer Science and Technology*, *Jiangsu Normal University*, *Xuzhou*, *Jiangsu* 221116, *China*;
2. *Jiangsu Key Laboratory of Educational Intelligent Technology*, *Jiangsu Normal University*, *Xuzhou*, *Jiangsu* 221116, *China*)

Abstract: Density-based clustering is a classical algorithm in cluster analysis, which can find non-spherical clusters without specifying the number of clusters in advance. In the real-world scene, there are still some issues, including unclear boundaries between clusters, varying densities of data, and complex cluster shapes. Most existing density-based clustering algorithms do not tackle these problems in a unified way. We counter this difficulty by taking inspiration from the natural erosion phenomenon to present erosion clustering (EC). Firstly, the proposed dynamic density evaluation method is integrated into the erosion strategy, which identifies and removes the data on the cluster boundary layer by layer, revealing the cores of the latent clusters. After that, a mutual-reachability-graph-based clustering is used to group the core data. Finally, the allocation strategy based on the local density peak is designed to associate the eroded data to different clusters. The experimental results on 12 benchmark datasets demonstrate that the clustering performance of the proposed EC algrithm is improved by an average of 96%, 53%, and 36% in the adjusted Rand index, adjusted mutual information, and F_1 score, respectively, comparing with the other seven algrithms.

Key words: density-based clustering; cluster analysis; density estimation; local density peak; mutual k-nearest neighbor; erosion strategy
Foundation Item(s): National Natural Science Foundation of China (No.62006104, No.61872168)

1 引言

聚类是数据挖掘和机器学习领域中最基础的任务之一,它已得到了非常广泛的研究和应用. 作为一种经典的无监督学习方法,聚类能够在没有数据标签的情况下,发现数据之间的潜在关系,并试图把相似的数据分为一组,把不相似的数据划分至不同组中. 近六十年里,研究者们提出了诸多经典的聚类算法,如:K-Means[1]、高斯混合模型(Gaussian Mixture Model,GMM)[2]、谱聚类(Spectral Clustering,SC)[3]、支持向量聚类(Support Vector Clustering,SVC)[4,5]、稀疏子空间聚类(Sparse Subspace Clustering,SSC)[6~8]等. 上述算法需要提前知晓数据真实的类簇数目,

收稿日期:2023-12-08;修回日期:2024-05-31;责任编辑:宋甲英

而在实际应用中,用户很难提前给定类簇的数目,这极大地阻碍了这些聚类算法的推广.

与上述算法不同,基于密度的聚类将类簇视作被低密度区域分隔的高密度连通区域.该类算法能够检测任意形状的类簇,并且不需要预先指定类簇数量.DBSCAN(Density-Based Spatial Clustering of Applications with Noise)[9]和Mean Shift[10]是这类算法的早期代表.密度峰聚类(Density Peaks clustering,DP)[11]融合了DBSCAN和Mean Shift算法的优点.上述算法虽然已取得了不错的聚类效果,但它们及其衍生算法[12~14]依然存在诸多不足:(1)不擅长处理密度分布不均的数据;(2)难以区分存在边界模糊的类簇.

针对上述问题,本文提出了侵蚀聚类(Erosion Clustering,EC)算法.该工作的主要贡献:提出了一种基于密度的聚类算法,能够处理具有密度不均、边界模糊、形状任意的类簇;(2)定义了基于激活互近邻的动态密度评估方法,能够更真实地反映数据的分布情况;(3)开发了基于互可达图的核心数据聚类方式,能够更精确地区分各种分布的核心区域;(4)设计了基于局部密度峰的边界数据分配方式,能够更容易地处理边界模糊的类簇.

2 相关工作

DBSCAN[9]作为经典的密度聚类算法,能够在不提前给定类簇数目的情况下发现任意形状的类簇,但该算法对参数的设置比较敏感.针对该问题,OPTICS(Ordering Points To Identify the Clustering Structure)[15]、HDBSCAN(Hierarchical Density-Based Spatial Clustering of Applications with Noise)[16]和DBADV(Density-Based clustering algorithm for Adaptive Density Variation)[17]给出了不同的解决思路.但是,这些算法对于模糊边界区域的处理效果并不理想.密度峰聚类[11]在这方面具有天然的优势.为进一步提高DP聚类在复杂分布上的聚类效果,研究者们展开了大量的研究[18],如DPC-KNN(Density Peaks Clustering based on K-Nearest Neighbors)[19]、DPC-DBFN(Density Peaks Clustering based on Density Backbone and Fuzzy Neighborhood)[20]、DPC-CE(Density Peak Clustering with Connectivity Estimation)[21]、McDPC(Multi-center Density Peak Clustering)[22]、MDPC+(Main Density Peak Clustering)[23]和DEMOS(DEnsity MOuntains Separation clustering)[24]等.然而,当数据存在多种密度分布,或者数据分布极其复杂时,上述算法并不总能取得令人满意的结果.

针对密度峰聚类在密度不均匀数据上表现欠佳的问题,一些算法提出了改进的策略[25~27].但是,该类算法大多采用距离缩放和类簇合并等策略,这限制了其处理模糊边界问题的能力.

与本文提出的侵蚀聚类算法最为相关的有两种聚类算法,分别是LGD(Local Gap Density)聚类[28]和BP(Border Peeling)聚类[29].所提算法与这两种算法在密度估计、核心数据聚类和边界数据分配三个方面具有本质区别.

3 侵蚀聚类

本文受侵蚀现象的启发.如果将密集区域假设为一个个的岛屿,而将稀疏区域假设为海水.经过海水不断侵蚀,岛屿的边缘区域可能会逐渐沉入海中,岛屿间的空隙逐渐增大,这使得人们清晰地分辨出各个岛屿真实的轮廓.本算法模仿此自然现象,逐层地侵蚀位于密集区域(或类簇)边界上的数据,发现各个类簇潜在的核心区域.然后采用基于互可达图的聚类方法实现核心区域划分.最后采用一种基于局部密度峰的分配方式完成已侵蚀数据的划分.

侵蚀聚类的整体流程如图1所示.EC算法首先执行侵蚀步骤,该步骤包含两个核心模块:密度的动态评估(详见3.2节)与边界数据的逐层识别(详见3.3节).蓝色框内的曲面图和等高线图直观展示了数据密度的分布情况,左侧子图为第一层侵蚀的密度分布示意图,右侧则对应第二层.绿色框内的散点图则展示了边界数据的逐层识别过程,蓝色点代表正常数据,紫色点代表当前层新识别的边界数据,而空心点表示先前已识别的边界数据.左侧、中间和右侧子图分别展示了第一层、第二层和最终的侵蚀状态.随后,EC算法基于互可达图进行核心数据的聚类操作(详见3.4节).红色框内的散点图清晰呈现了这一过程,四种颜色的实心点构成了四个类簇的核心区域,空心点则代表尚未分配的边界数据.最后,EC算法采用基于局部密度峰的方法处理边界数据(详见3.5节).紫色框内的散点图展示了边界数据的分配过程,四种颜色的实心点依旧代表各自的类簇,紫色实心点表示当前层上待分配的边界数据,黑色空心点依然表示未分配的数据.边界分配过程会回溯各层侵蚀的结果,因此,其侵蚀状态与上方子图保持一致.

3.1 符号与定义

令$X=\{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n\}$表示包含$n$个数据的数据集,其中$\boldsymbol{x}_i=(x_{i,1},\cdots,x_{i,m})$表示第$i$个数据,$m$为属性个数.

对于具有L层的侵蚀过程而言,$X^{(l)}$和$X_E^{(l)}$分别表示第l层迭代时待计算数据和新侵蚀的数据,亦可称之为第l层迭代的候选核心数据(或激活数据)和新识别的边界数据.需要注意的是,是通过算法在每次迭代时计算得出的.而更高层(第l+1层)中的候选核心数据(待计算数据)由前一层(第l层)中的候选核心数据(待计算数据)减去前一层(第l层)中新识别的边界数据

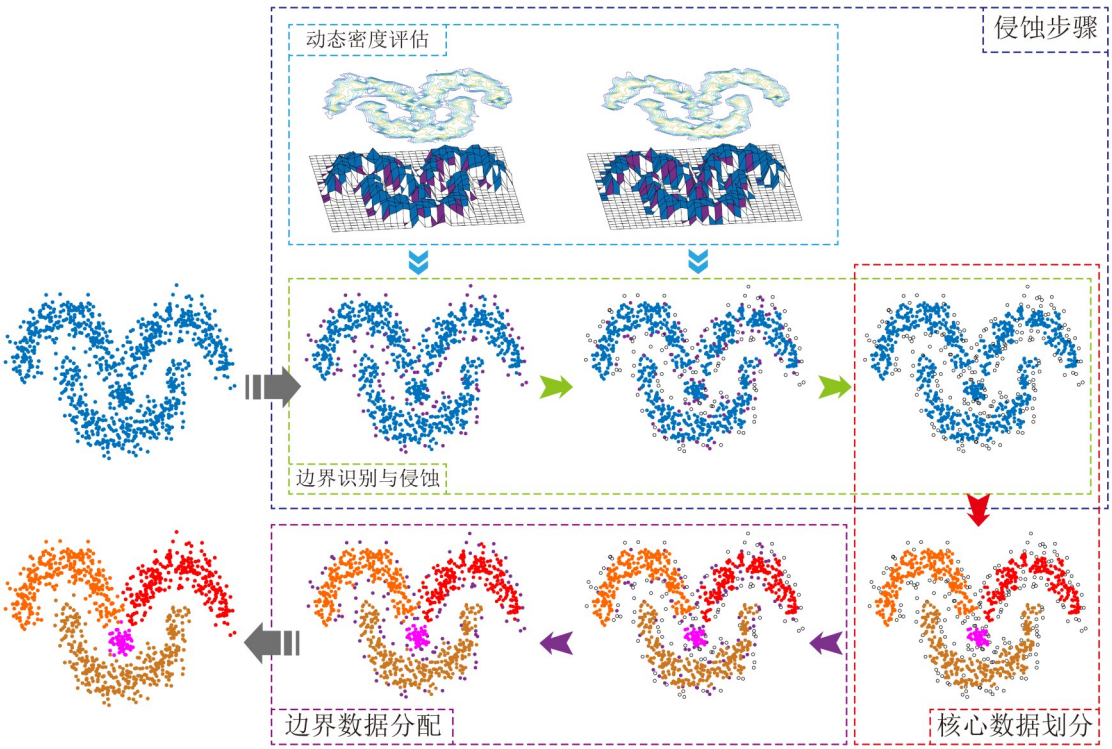


图1　侵蚀聚类整体结构

(新侵蚀数据)的差集构成,即$X^{(l+1)}=X^{(l)}\backslash X_E^{(l)}$. 令$X_B^{(l)}$表示第$l$层迭代时已被侵蚀的数据(已识别的边界数据),即第$l$层迭代的已侵蚀数据:

$$X_B^{(l)}=\begin{cases}X_E^{(1)}\cup\cdots\cup X_E^{(l-1)}, & \text{if } l>1\\ \varnothing, & \text{if } l=1\end{cases} \tag{1}$$

根据上述定义,很容易得出$X^{(l)}\cup X_B^{(l)}=X$且$X^{(l)}\cap X_B^{(l)}=\varnothing$. 在第一次迭代时(即$l$=1),存在$X^{(1)}=X$且$X_B^{(1)}=\varnothing$. 侵蚀过程结束时(即$l$=$L$+1),当前的候选核心数据被确立为最终的核心数据,当前已识别边界数据则构成最终的边界数据. 具体而言,$X^{(L+1)}$和$X_B^{(L+1)}$分别代表数据集的核心数据和边界数据.

令$\|\boldsymbol{x}_i-\boldsymbol{x}_j\|$表示数据$\boldsymbol{x}_i$和$\boldsymbol{x}_j$之间的欧氏距离. $N_k(\boldsymbol{x}_i)$表示数据$\boldsymbol{x}_i$的第k个近邻.

定义 1(核心距离)　在给定近邻数目k的条件下,数据$\boldsymbol{x}_i$的核心距离为该数据到其第k个近邻的距离,记作$d_{\text{core}}(\boldsymbol{x}_i)$. 即,$d_{\text{core}}(\boldsymbol{x}_i)=\|\boldsymbol{x}_i-N_k(\boldsymbol{x}_i)\|$.

数据集中所有数据核心距离构成的集合称作核心距离集合,记作D_{core}. 即,$D_{\text{core}}=\{d_{\text{core}}(\boldsymbol{x}_i)|\boldsymbol{x}_i\in X\}$.

$\text{kNN}(\boldsymbol{x}_i)$表示数据$\boldsymbol{x}_i$的$k$近邻集合,$\text{mkNN}(\boldsymbol{x}_i)$表示数据$\boldsymbol{x}_i$的互$k$近邻集合.

在第l层中,数据$\boldsymbol{x}_i$的k近邻中激活数据(即待计算数据)集合称作$\boldsymbol{x}_i$在第l层上的激活近邻集合,记作$\text{kNN}^{(l)}(\boldsymbol{x}_i)$. 对于每一个数据$\boldsymbol{x}_j\in\text{kNN}^{(l)}(\boldsymbol{x}_i)$,有$\boldsymbol{x}_j\in X^{(l)}\cap\text{kNN}(\boldsymbol{x}_i)$. 注意,集合$\text{kNN}^{(l)}(\boldsymbol{x}_i)$的数据数目小于等于$k$.

同理,在第l层中,数据$\boldsymbol{x}_i$的互k近邻中待计算数据集合称作$\boldsymbol{x}_i$在第l层上激活互近邻集合,记作$\text{mkNN}^{(l)}(\boldsymbol{x}_i)$. 对于每一个数据$\boldsymbol{x}_j\in\text{mkNN}^{(l)}(\boldsymbol{x}_i)$,有$\boldsymbol{x}_j\in X^{(l)}\cap\text{mkNN}(\boldsymbol{x}_i)$.

定义 2(边界k近邻)　在第l层上,待计算数据$\boldsymbol{x}_i$在已侵蚀数据中的k近邻集合称作$\boldsymbol{x}_i$的边界k近邻集合,用$\text{kNN}_B^{(l)}(\boldsymbol{x}_i)$表示. 其任意边界$k$近邻$\boldsymbol{x}_j$必然属于当前的已侵蚀数据集合,即$\boldsymbol{x}_j\in X_B^{(l)}$.

定义 3(核心k近邻)　在第l层上,已侵蚀数据$\boldsymbol{x}_i$在候选核心数据中的k近邻集合称作$\boldsymbol{x}_i$的核心k近邻集合,用$\text{kNN}_C^{(l)}(\boldsymbol{x}_i)$表示. 其任意核心$k$近邻$\boldsymbol{x}_j$必然属于当前的候选核心数据集合,即$\boldsymbol{x}_j\in X^{(l)}$.

定义 4(局部密度峰)　若数据$\boldsymbol{x}_j$是已侵蚀数据$\boldsymbol{x}_i$的核心k近邻中密度最大的数据,则$\boldsymbol{x}_j$为$\boldsymbol{x}_i$的局部密度峰,记作$p(\boldsymbol{x}_i)=\boldsymbol{x}_j$.

注意,当一个数据$\boldsymbol{x}_i$的核心k近邻中密度最高的数据有多个时,则定义与其距离最近的数据为真正的局部密度峰.

3.2　基于侵蚀互近邻的动态密度评估方法

侵蚀聚类依据数据的密度决定每层侵蚀哪些数据.

因此,密度估计方法对侵蚀算法的表现具有决定性的影响. 本小节介绍本算法所采用的侵蚀密度估计方法.

核密度估计是一种常用的密度估计方法,数据$\boldsymbol{x}_i$的核密度聚类估计可以由下列公式表示:

$$\rho_i=\sum_{\boldsymbol{x}_j\in V}\kappa\left(\frac{\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|}{h}\right) \tag{2}$$

其中,V表示有关数据$\boldsymbol{x}_i$的集合,$\kappa(\cdot)$为核函数,h表示缩放因子.

由于基于k近邻的柯西核密度估计方法具有良好的平滑性且易于实现,本文采用该种形式的核密度估计方法. 其数学形式如下:

$$\rho_i=\sum_{\boldsymbol{x}_j\in \mathrm{kNN}(\boldsymbol{x}_i)}\left(\frac{\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|^2}{h^2}+1\right)^{-1} \tag{3}$$

但是该方法存在两个问题:(1)缩放因子难以调节;(2)位置信息无法得到充分利用. 在缩放因子的设置问题上本文采用一种自适应的方案,对每个数据设置不同的h值. 本文定义每个数据$\boldsymbol{x}_j$的缩放因子h_j为其与第k个近邻之间的距离,即$h_j=\|N_k(\boldsymbol{x}_j)-\boldsymbol{x}_j\|$. 此方法不仅简单且易于实现,更关键的是,它能自适应数据的分布特性. 具体而言,在密集区域,数据的第k个近邻距离较近,导致h_j值较小,进而缩小了核函数的影响范围,有效避免了过度平滑的现象. 相反,在稀疏区域,数据的第k个近邻距离较远,使得h_j值较大,因此核函数能够覆盖更广泛的区域,从而成功捕捉到潜在的稀疏结构.

不同于大部分的密度评估方法,本文结合侵蚀策略设计了一种动态密度评估方法. 本评估方法受到这一现象的启发,大大降低了受侵蚀周围区域的数据密度,从而提高了该类区域进一步受到侵蚀的可能性. 这使得整个侵蚀过程更符合侵蚀规律,并实现了由外向内逐层侵蚀的效果.

侵蚀密度估计函数能够动态定义数据在各层上的密度值:

$$\rho_i^{(l)}=\sum_{\boldsymbol{x}_j\in \mathrm{mkNN}^{(l)}(\boldsymbol{x}_i)}\left(\frac{\left\|\boldsymbol{x}_i-\boldsymbol{x}_j\right\|^2}{\left\|N_k(\boldsymbol{x}_j)-\boldsymbol{x}_j\right\|^2}+1\right)^{-1} \tag{4}$$

其中$\mathrm{mkNN}^{(l)}(\boldsymbol{x}_i)$表示数据$\boldsymbol{x}_i$在第$l$层上的激活互近邻(见定义1). 侵蚀的密度估计函数考虑了数据侵蚀和空间位置变化对密度的影响.

当数据$\boldsymbol{x}_i$的互k近邻在更高层上遭受侵蚀时,$\boldsymbol{x}_i$在更高层上的密度将会低于其在当前层上的密度. 以图2为例,给出侵蚀前后的密度估计. 如图2(a)所示,红色实心点为待计算密度的数据$\boldsymbol{x}_1$,绿色实心点构成了数据$\boldsymbol{x}_1$的互k近邻(k=5)集合$\{\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4,\boldsymbol{x}_5,\boldsymbol{x}_6\}$. 图中括号内的信息表示各点的坐标. 依据式(4),数据$\boldsymbol{x}_1$的密度为

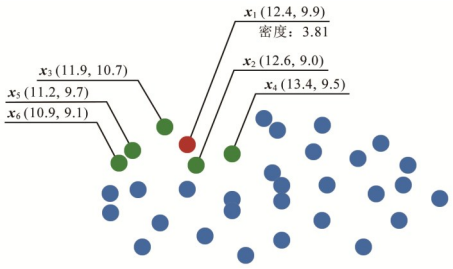
$$\begin{aligned}\rho_1^{(1)}&=\sum_{\boldsymbol{x}_j\in \mathrm{mkNN}^{(1)}(\boldsymbol{x}_1)}\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_j\right\|^2}{\left\|N_k(\boldsymbol{x}_j)-\boldsymbol{x}_j\right\|^2}+1\right)^{-1}\\&=\sum_{\boldsymbol{x}_j\in\{\boldsymbol{x}_2,\boldsymbol{x}_3,\boldsymbol{x}_4,\boldsymbol{x}_5,\boldsymbol{x}_6\}}\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_j\right\|^2}{\left\|N_k(\boldsymbol{x}_j)-\boldsymbol{x}_j\right\|^2}+1\right)^{-1}\\&=\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_2\right\|^2}{\left\|N_k(\boldsymbol{x}_2)-\boldsymbol{x}_2\right\|^2}+1\right)^{-1}+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_3\right\|^2}{\left\|N_k(\boldsymbol{x}_3)-\boldsymbol{x}_3\right\|^2}+1\right)^{-1}\\&\quad+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_4\right\|^2}{\left\|N_k(\boldsymbol{x}_4)-\boldsymbol{x}_4\right\|^2}+1\right)^{-1}\\&\quad+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_5\right\|^2}{\left\|N_k(\boldsymbol{x}_5)-\boldsymbol{x}_5\right\|^2}+1\right)^{-1}+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_6\right\|^2}{\left\|N_k(\boldsymbol{x}_6)-\boldsymbol{x}_6\right\|^2}+1\right)^{-1}\\&\approx\left(\frac{0.922^2}{1.703^2}+1\right)^{-1}+\left(\frac{0.943^2}{1.921^2}+1\right)^{-1}\\&\quad+\left(\frac{0.943^2}{1.921^2}+1\right)^{-1}+\left(\frac{1.077^2}{1.746^2}+1\right)^{-1}\\&\quad+\left(\frac{1.217^2}{1.803^2}+1\right)^{-1}+\left(\frac{1.700^2}{1.703^2}+1\right)^{-1}\\&\approx 3.491\end{aligned}$$

如图2(b)所示,$\boldsymbol{x}_1$的互k近邻$\boldsymbol{x}_3$在该轮被侵蚀,用绿色空心点表示此变化. 因此,在该层中,数据$\boldsymbol{x}_1$的互k近邻集合更新为$\{\boldsymbol{x}_2,\boldsymbol{x}_4,\boldsymbol{x}_5,\boldsymbol{x}_6\}$. 其密度为

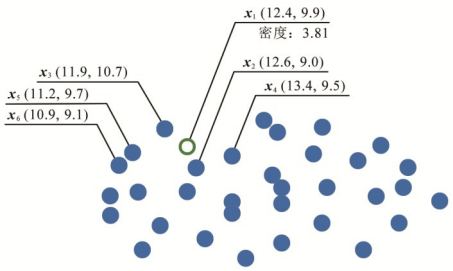
$$\begin{aligned}\rho_1^{(2)}&=\sum_{\boldsymbol{x}_j\in \mathrm{mkNN}^{(2)}(\boldsymbol{x}_1)}\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_j\right\|^2}{\left\|N_k(\boldsymbol{x}_j)-\boldsymbol{x}_j\right\|^2}+1\right)^{-1}\\&=\sum_{\boldsymbol{x}_j\in\{\boldsymbol{x}_2,\boldsymbol{x}_4,\boldsymbol{x}_5,\boldsymbol{x}_6\}}\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_j\right\|^2}{\left\|N_k(\boldsymbol{x}_j)-\boldsymbol{x}_j\right\|^2}+1\right)^{-1}\\&=\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_2\right\|^2}{\left\|N_k(\boldsymbol{x}_2)-\boldsymbol{x}_2\right\|^2}+1\right)^{-1}+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_4\right\|^2}{\left\|N_k(\boldsymbol{x}_4)-\boldsymbol{x}_4\right\|^2}+1\right)^{-1}\\&\quad+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_5\right\|^2}{\left\|N_k(\boldsymbol{x}_5)-\boldsymbol{x}_5\right\|^2}+1\right)^{-1}+\left(\frac{\left\|\boldsymbol{x}_1-\boldsymbol{x}_6\right\|^2}{\left\|N_k(\boldsymbol{x}_6)-\boldsymbol{x}_6\right\|^2}+1\right)^{-1}\\&\approx\left(\frac{0.922^2}{1.703^2}+1\right)^{-1}+\left(\frac{1.077^2}{1.746^2}+1\right)^{-1}+\left(\frac{1.217^2}{1.803^2}+1\right)^{-1}\\&\quad+\left(\frac{1.700^2}{1.703^2}+1\right)^{-1}\approx 2.686\end{aligned}$$

由于数据$\boldsymbol{x}_1$的互k近邻$\boldsymbol{x}_3$在更高层(l=2)中被侵蚀,其密度值出现$\rho_1^{(1)}>\rho_1^{(2)}$的情况,表明在更高层中该数

据的密度有所下降. 据此可推断,在侵蚀聚类算法中,所有数据的密度均满足$\rho_i^{(l)} \geqslant \rho_i^{(l+1)}$.



(*a*) 低层未被侵蚀时的情况



(*b*) 高层被侵蚀后的情况

图2　侵蚀前后的密度

侵蚀密度估计的一个显著特性是,当数据周围区域发生侵蚀时,该数据的密度会降低,进而提高其被进一步侵蚀的可能性,从而保证了侵蚀过程的连续性.

3.3　侵蚀策略

相较于类簇的核心区域,类簇边界区域更容易分类错误,尤其是在类簇边界区域模糊不清难以界定的情况下. 因此,相比于类簇的边界数据,类簇的核心数据彼此相距较远且更容易被合理的聚类.

传统的方法采用一种全局阈值的方式定义核心数据,但这类方式不适用于具有不同密度的数据集合. 与传统方法不同,侵蚀策略迭代剔除类簇的边界,逐步地发现类簇的核心数据. 单层的侵蚀过程首先通过式(4)来估计该层上每个数据的密度,然后按照密度从小到大的顺序进行排序,接着将其中一部分密度较小的数据划分为边界数据并执行侵蚀操作. 类似于BP聚类中的剥离过程,本策略将密度小于某阈值的数据定义为当前层新侵蚀的数据,即$X_E^{(l)}=\{\boldsymbol{x}_i|\rho_i^{(l)} \leqslant \rho_c^{(l)}\}$,其中$\rho_c^{(l)}$表示该层数据密度第10个百分位数. 换言之,单次侵蚀会将密度值在前10%的数据设置为新侵蚀数据. 而剩余的数据会作为下层数据继续参与计算,即$X^{(l+1)}=X^{(l)} \backslash X_E^{(l)}$. 这意味着密度较高的90%的数据会保留下来,并参与到下一层的计算中. 在该步骤中,将单次侵蚀比例γ设置为10%是一种经验性的设定. 正如参数分析小节(第4.5节)所示,当侵蚀比例在[7%,20%]范围内时,算法能够维持较高的聚类质量.

3.4　基于互可达图的核心数据聚类

侵蚀过程结束后,剩余数据所在区域之间可能存在更大的间隔,更利于聚类操作. 本文构建了一种基于互可达关系的近邻图,用以实现核心数据的聚类. 在给出具体的聚类步骤前,先给出一些基础概念.

定义5(ε半径)　对于第l层的候选核心数据$\boldsymbol{x}_i$(即,$\boldsymbol{x}_i \in X^{(l)}$),其$\varepsilon$半径定义为

$$\varepsilon(\boldsymbol{x}_i)=\frac{1}{k}\sum_{\boldsymbol{x}_j \in \mathrm{kNN}_B^{(l)}(\boldsymbol{x}_i)} \delta(\boldsymbol{x}_j) \tag{5}$$

其中,$\mathrm{kNN}_B^{(l)}(\boldsymbol{x}_i)$表示数据$\boldsymbol{x}_i$的边界$k$近邻(见定义2),$\delta(\boldsymbol{x}_j)$表示已侵蚀数据$\boldsymbol{x}_j$的连接距离(见定义8).

本文仿照HDBSCAN中的概念,定义了一种互可达阈值.

定义6(互可达阈值)　对于最终核心数据$\boldsymbol{x}_i$和$\boldsymbol{x}_j$的互可达阈值定义为

$$d_{\mathrm{mr}}(\boldsymbol{x}_i,\boldsymbol{x}_j)=\max\left\{\min\{\varepsilon(\boldsymbol{x}_i),\lambda\},\min\{\varepsilon(\boldsymbol{x}_j),\lambda\}\right\} \tag{6}$$

其中,$\lambda=\mu(D_{\mathrm{core}})+\sigma(D_{\mathrm{core}})$(见定义1). $\mu(\cdot)$和$\sigma(\cdot)$分别表示集合的均值和标准差. 其中,均值代表了数据间距离的平均水平,标准差代表了数据间距离的波动情况. 通过将均值加上标准差,可以将阈值定义在距离波动范围内的较高水平.

定义7(互可达图)　互可达图$\boldsymbol{G}$是一种以核心数据$X^{(L+1)}$为顶点的无向图,其中当两个顶点$\boldsymbol{x}_i,\boldsymbol{x}_j \in X^{(L+1)}$间的距离小于等于其间的互可达阈值($\|\boldsymbol{x}_i-\boldsymbol{x}_j\| \leqslant d_{\mathrm{mr}}(\boldsymbol{x}_i,\boldsymbol{x}_j)$)时,这两个顶点间存在一条边.

互可达图$\boldsymbol{G}$的一个连通分量视为核心数据$X^{(L+1)}$的一个类簇.

3.5　基于局部密度峰的边界数据分配方式

核心数据聚类结束后,算法将回溯每层的侵蚀结果,逐层实施侵蚀数据的分配任务. 该分配的指导原则是将已侵蚀数据分配到内层中某个数据所在类簇. 即,该分配是将已侵蚀数据与内层中的某个数据建立连接的过程,这种逐层的连接关系必然使得每个侵蚀数据均会与某个核心数据建立关系. 这种连接关系意味着侵蚀数据与相连接的核心数据隶属于同一类簇. 因此,该方式能够实现边界数据的分配.

最简单的分配方式是将已侵蚀数据与其最近的内层数据建立连接. 但是,该种分配方式仅考虑了距离因素,并未考虑密度因素.

本文提出的分配方式其核心思想:朝着局部密度增长的方向建立连接. 具体而言,将已侵蚀数据与其核心k近邻中密度最大的数据建立连接,即与其局部密度峰(见定义4)建立连接.

定义8(连接距离)　若数据$\boldsymbol{x}_i$为一个已建立连接

的已侵蚀数据，则称该数据到其局部密度峰的距离为其连接距离，记作 $\delta(\boldsymbol{x}_i)=\left\|\boldsymbol{x}_i-p(\boldsymbol{x}_i)\right\|$.

3.6 算法流程

综合上述主要阶段的介绍，侵蚀聚类的算法流程如算法1所示.

算法1 侵蚀聚类算法

输入:数据集 X, 近邻数目 k, 侵蚀层数 L

输出:聚类结果 $\boldsymbol{y}$

1. 计算数据间的距离
2. 计算互近邻关系
3. while $l\leqslant L$
4. 依据式(4)计算数据密度
5. 执行侵蚀操作 $X_E^{(l)}=\{\boldsymbol{x}_i|\rho_i^{(l)}\leqslant\rho_c^{(l)}\}$
6. 更新 $X^{(l+1)}\leftarrow X^{(l)}\backslash X_E^{(l)}$
7. end while
8. 依据互可达图 $\boldsymbol{G}$ 划分最终核心数据 $X^{(L+1)}$
9. 执行边界数据分配

3.7 时间复杂度

本小节分析侵蚀聚类算法的时间复杂度. 步骤①的时间复杂度是 $O(n^2)$，其中 n 表示数据数目. 步骤②的复杂度取决于搜索树的复杂度，假设采用KD树等索引结构执行搜索，则复杂度是 $O(n\log n)$. 步骤③~⑦表示侵蚀阶段以及边界数据分配阶段，其时间复杂度是 $O(Ln\log n)$，其中 L 为侵蚀层数. 步骤⑧的时间复杂度为 $O(kn_c^2)$，其中 n_c 表示最终核心数据 $X^{(L+1)}$ 的数目，且有 $n_c<n$. 步骤⑨的时间复杂度也为 $O(Ln\log n)$. 因此，侵蚀聚类算法总的时间复杂度为 $O(n^2)$.

4 实验分析

4.1 参数设置

数据集：本文选取了12个数据集用于验证侵蚀聚类算法的有效性. 这些数据集分为两个部分：6个人工数据集和6个真实数据集. 人工数据集包括Square、Jain、SF、Handl、T8和Shape5. 真实数据集包括Zoo、Iris、Dermatology、WDBC、Penbased和HTRU2. 数据集的详情见表1，其中，n 表示数据个数，m 表示属性个数，c 表示类个数.

对比算法：本文实验选7种聚类算法进行比较，包括：DBSCAN[9]、HDBSCAN[16]、DP[11]、DPC-CE[21]、DEMOS[24]、LGD[28]和BP[29]. 这些算法都无须提前给定类簇数目. 对于DP、DPC-CE、DEMOS和LGD算法，用户需要在决策图上手动地选择类簇中心. 为了公平比较，本文实验借鉴DP-MD-FN[30]算法中所使用的自动确定类簇中心的策略，通过引入调节参数 α 自动确定类簇数目. 本算法以及所有对比算法的参数设置如表2所示.

表1 数据集的统计信息

数据集	$n/m/c$	数据集	$n/m/c$
Square	1 100/2/2	Jain	373/2/2
SF	4096/2/4	Handl	715/2/3
T8	7677/2/8	Shape5	3 150/2/5
Zoo	101/16/7	Iris	150/4/3
Dermatology	366/34/6	WDBC	569/30/2
Penbased	10 992/16/10	HTRU2	17 898/8/2

表2 参数设置

算法	参数设置
EC	k: [5:1:50]; L: [2:12]
DBSCAN	ε: [0.1:0.25:5.1]; k: [5:1:50]
HDBSCAN	k: [5:1:50]; m_{clSize}: [5:5:20]
DP	d_c: [2%:1%:10%]; α: [0.5:0.5:2.5]
DPC-CE	d_c: [2%:1%:10%]; α: [0.5:0.5:2.5]
DEMOS	k: [5:1:50]; α: [0.5:0.5:2.5]
LGD	k: [5:1:50]; τ: [0.5:0.02:0.6]; α: [0.5:0.5:2.5]
BP	k: [5:1:50]

评价指标：本文采用三种经典的外部聚类评价指标，分别是修正兰德指标(ARI)、修正互信息(AMI)、F_1. 指标的值越大，表示聚类效果越好. F_1 值的范围在[0, 1]之间；ARI和AMI的范围为[−1, 1].

实验环境：设备配置为Intel(R) Core(TM) i9-10900X @3.70 GHz CPU，64 GB内存，Windows 10操作系统；编程环境为Matlab 2021a.

4.2 人工数据集上的实验结果

为更直观地展示各算法的聚类性能，本小节使用6个二维人工数据集展开测试. Square和Jain数据集中各个类簇具有不同的密度；SF和Handl数据集中各类簇间具有重叠区域，边界难以区分；T8数据集具有较为复杂的数据分布；Shape5数据集兼具边界模糊、密度不同、分布不规则的特性. 图3展示了8种算法在6个人工数据集上的聚类结果. 图中使用不同的颜色标识不同类簇的数据，其中黑色圆圈表示噪声数据.

图3第一行给出了Square数据集上的聚类结果. 该数据集包含两个方形类簇，其中上方稀疏类簇的分布并不均匀. 本文算法、DBSCAN、DPC-CE、DEMOS和LGD给出了正确结果；HDBSCAN仅识别密集区域数据，而将稀疏区域的数据视为噪声；其余算法将上方密度不均匀类簇划分为多个更小的类簇.

图3第二行展示了Jain数据集上的聚类结果. 该数据集包含两个月牙形类簇，其中下方类簇的密度要明显高于上方类簇的密度. 本算法和DEMOS能够正确识别这两个类簇；DBSCAN将上方稀疏的类簇错误识别为噪声数据；HDBSCAN只标注了部分密集区域上的数据，将大部分数据错误识别为噪声；LGD给出了正确的类簇数目，但依然存在误划分的情况；其余算法均出现

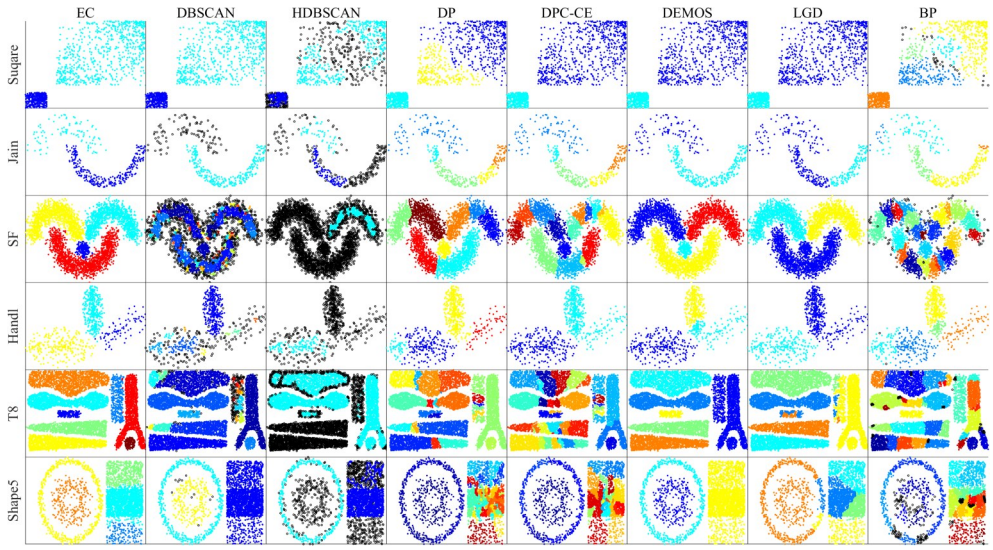


图3　人工数据集聚类结果

了过度划分的问题.

前两个数据集的聚类结果显示,BP聚类在密度不均匀的数据上容易产生过度划分的问题;而LGD聚类也仅在一个数据集(Square)上有着不错表现.DBSCAN、HDBSCAN和DP等传统聚类算法同样难以胜任多密度数据的聚类任务.DPC-CE作为DP聚类的改进算法,重新定义了局部密度,但仍未能提高其在该问题上的处理能力.与DEMOS相同,本聚类算法在上述两组多密度数据集上均取得了最佳的聚类结果.这是因为本文使用了区域自适应策略来定义核心数据间的互可达阈值,从而使得侵蚀聚类算法可以自适应不同数据密度,并能够胜任该种聚类任务.

图3的第三行展示了SF数据集的聚类结果.该数据集包含三个月牙形类簇和一个球形类簇.四个类簇彼此相距较近,尤其是下方的球形类簇与月牙形类簇间具有比较模糊的边界.仅有本文算法和DEMOS获得不错的聚类效果;LGD将下方的两个类簇归为一类;HDBSCAN将大部分数据识别成为噪声数据;其余算法将三个月牙形类簇过渡分割成多个子类簇.

图3的第四行展示了Handl数据集的聚类结果.该数据集包含三个密度略有不同的凸型类簇,它们间存在区域重叠现象.DPC-CE和LGD都将上方的类簇和右侧的类簇聚为了一类;DBSCAN和HDBSCAN都将部分数据错归为噪声;DEMOS将上方类簇中的一部分数据误分配给了右侧类簇;而BP产生了过度划分的问题.在所有算法中侵蚀聚类产生了最好的聚类结果.

从上述两组数据集的聚类结果可以看出,本算法在处理类簇重叠问题上表现明显优于其他算法.这得益于算法所采用的动态密度估计方法和侵蚀策略,能够有效地检测处于重叠区域的类簇边界,并描绘出类簇的核心骨架.

T8数据集的聚类结果如图3第五行所示.该数据集由八个不规则的类簇构成.侵蚀聚类算法准确地发现了所有类簇;其他聚类算法产生了过分划分或错误合并的情况.本算法运用了基于图的聚类思想,即采用了基于互可达图的核心数据聚类策略.这种策略使得本算法在聚类任意形状的类簇任务上展现出与DBSCAN相媲美的性能.

图3第六行给出了Shape5上的聚类结果.该数据集有两个同心圆类簇和三个彼此邻近、密度不同的方形类簇共同构成.仅有侵蚀聚类算法能够识别各个类簇的轮廓.该数据集上的表现说明,侵蚀聚类能够处理同时具有密度不均、边界模糊、形状复杂等特性的数据.

表3展示了在6个人工数据集上8种算法的各项聚类评价指标.表中粗体指标值表示各个数据集的对应指标下最高得分.列"c"表示该算法产生的类簇数目,该项粗体意味着该聚类产生的类簇数目与真实的类数目相同.如表3所示,侵蚀聚类算法比其他7种算法分别在ARI、NMI和F_1上平均提升105%、65%和39%.

4.3　真实数据集上的实验结果

本小节展示真实数据集上的聚类结果,并详尽分析各算法聚类性能.表4展示了6个真实数据集上所有算法的ARI、AMI和F_1.表中粗体数值表示各个数据集的对应指标下最高得分.ARI和AMI的值为0通常是由于算法产生的聚类结果仅包含1个类簇.请注意,表4中存在类簇数目为1(即,$c=1$),但ARI和AMI的值并非为0的情况,如DBSCAN、HDBSCAN和BP等算法在WDBC数据集上的表现.这是因为在这种情况下,相关算法错误地将正常数据识别为噪声数据.为获得更公正的评价结果,在聚类评估指标的计算中,噪声数据会

表3 算法在人工数据集上的表现

算法	ARI	AMI	F_1	c	参数	ARI	AMI	F_1	c	参数
	Square					Jain				
EC	**1**	**1**	**1**	**2**	16/2	**1**	**1**	**1**	**2**	16/2
DBSCAN	**1**	**1**	**1**	**2**	0.1/5	**1**	**1**	**1**	1	3.1/28
HDBSCAN	0.528	0.602	0.856	**2**	5/5	0.209	0.37	0.671	**2**	5/5
DP	0.633	0.781	0.889	3	3/1	0.334	0.615	0.824	5	6/2.5
DPC-CE	**1**	**1**	**1**	**2**	10/2.5	0.303	0.606	0.783	5	2/2.5
DEMOS	**1**	**1**	**1**	**2**	30/2.5	**1**	**1**	**1**	**2**	9/2
LGD	**1**	**1**	**1**	**2**	13/0.6/1	0.028	0.222	0.608	**2**	5/0.6/1.5
BP	0.494	0.672	0.836	5	50/3	0.478	0.675	0.856	5	13/3
	SF					Handl				
EC	**0.977**	**0.951**	**0.985**	**4**	16/3	**0.988**	**0.974**	**0.994**	**3**	12/2
DBSCAN	0.135	0.482	0.476	77	0.1/5	0.72	0.664	0.655	9	0.1/5
HDBSCAN	0.127	0.304	0.481	1	21/20	0.25	0.357	0.581	1	9/15
DP	0.553	0.754	0.76	8	4/2.5	0.844	0.828	0.916	4	5/1.5
DPC-CE	0.346	0.66	0.627	15	6/2.5	0.762	0.817	0.734	2	8/2.5
DEMOS	0.976	0.95	0.983	**4**	17/2.5	0.754	0.782	0.849	**3**	9/1.5
LGD	0.894	0.897	0.783	3	25/0.6/1	0.762	0.818	0.734	2	28/0.58/1
BP	0.201	0.6	0.516	24	30/3	0.659	0.784	0.878	5	30/3
	T8					Shape5				
EC	**0.999**	**0.997**	**0.999**	**8**	24/2	**0.906**	**0.907**	**0.957**	**5**	22/3
DBSCAN	0.934	0.925	0.916	23	0.1/10	0.474	0.719	0.666	3	0.6/5
HDBSCAN	0.26	0.44	0.343	1	50/20	0.644	0.651	0.626	2	5/20
DP	0.645	0.818	0.731	25	7/2.5	0.122	0.543	0.494	75	10/1
DPC-CE	0.421	0.734	0.633	41	9/2.5	0.177	0.523	0.604	30	9/1
DEMOS	0.908	0.946	0.79	6	35/2.5	0.475	0.726	0.671	3	25/2.5
LGD	0.88	0.895	0.662	5	10/0.6/1.5	0.463	0.609	0.728	**5**	5/0.6/1.5
BP	0.39	0.751	0.708	31	46/3	0.219	0.603	0.64	20	25/3

被单独视为一个类簇参与计算.

DBSCAN和HDBSCAN的聚类结果同样并不理想.这两种基于密度的聚类算法并未考虑类簇重叠问题.而在真实数据中,尤其是高维复杂数据中,类簇重叠的现象普遍存在.此外,当数据集中存在密度变化较大的区域时,这两种算法可能会出现误识别噪声的问题.以WDBC数据集为例,两种算法都将该数据集划分为一类,并将正常的数据识别为噪声.

DP算法的聚类表现明显优于DBSCAN和HDBSCAN.这主要是因为相对于前两种算法,DP聚类对参数选择的敏感性较低,因此其聚类结果更加稳定.此外,DP聚类算法采用基于密度的数据分配能够获得更准确的聚类结果.

DPC-CE和DEMOS是改进的DP聚类算法,它们仅在WDBC和HTRU2数据集上相对于DP算法略有改善.该情况的主要原因可能是本文未采用人为圈选类簇中心的方式,即未在决策图上手动选定数据集的密度峰.这种人为选择方式不利于与其他聚类算法进行公平比较.为此,本文采用了一种自动确定密度峰的策略.然而,该策略可能无法充分展现这些算法的性能.

与本文算法类似,LGD和BP算法也是两种面向边界分割的聚类算法.从表4的聚类结果可以看出,相比于这两种算法,EC算法也有着显著提升.

如表4所示,侵蚀聚类算法比其他7种算法分别在ARI、NMI和F_1上平均提升87%、41%和33%.其中,部分数据集表现出较为明显的优势.总的来说,这些聚类结果再次验证了本算法提出的密度评估方法、核心数据聚类和边界数据分配方式的有效性.

4.4 参数分析

本节测试不同参数取值对侵蚀聚类性能的影响.侵蚀聚类算法存在两个关键参数:近邻数目k和侵蚀层数L.本文将这两个参数的取值范围分别设置为$k\in[5,50]$和$L\in[2,12]$.图4展示了在四组不同数据集上(包括两组人工数据集和两组真实数据集),采用不同近邻数目和不同侵蚀层数时的ARI变化.图4展示的结果显示,本文算法的聚类效果未出现明显的异常变化,并且性能的变化呈现一定的规律.

表4　算法在真实数据集上的表现

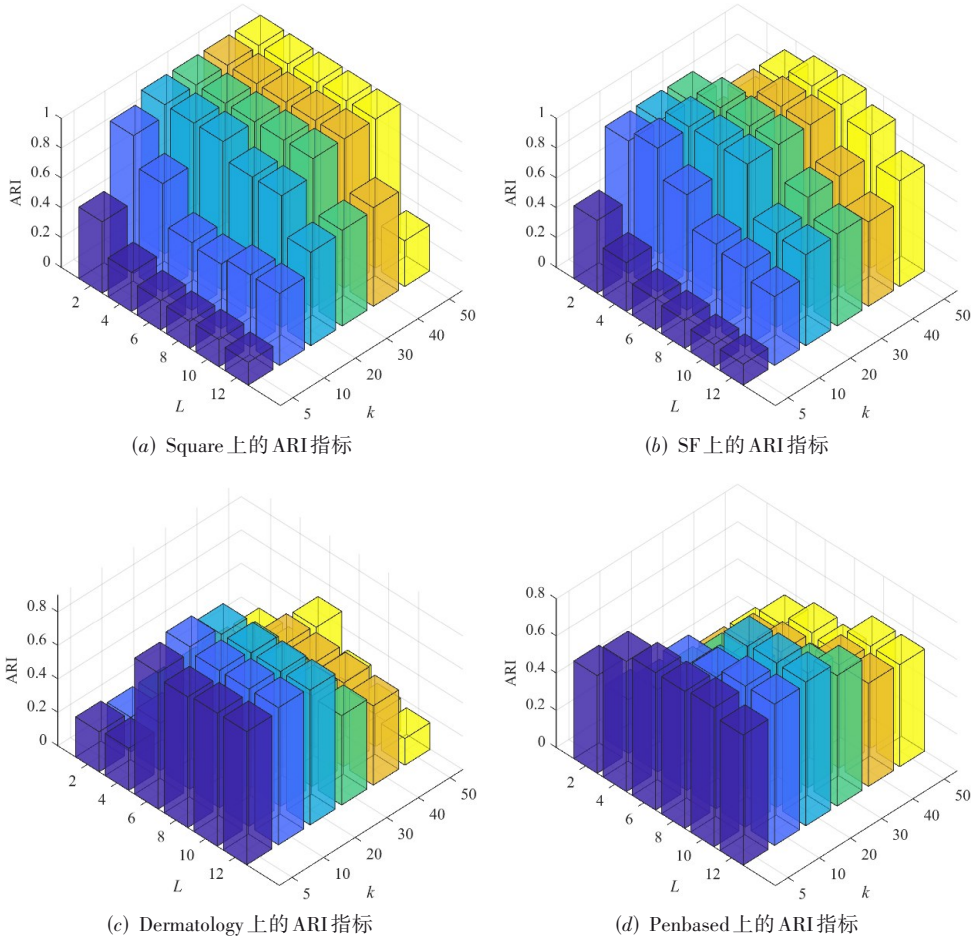
算法	ARI	AMI	F_1	c	参数	ARI	AMI	F_1	c	参数
	Zoo					Iris				
EC	**0.954**	**0.908**	**0.85**	6	10/2	**0.904**	**0.879**	**0.967**	**3**	7/9
DBSCAN	0.918	0.864	0.798	5	1.35/5	0.568	0.759	0.778	2	0.35/21
HDBSCAN	0.365	0.538	0.495	2	5/15	0.001	0.017	0.497	1	25/20
DP	0.745	0.775	0.542	3	8/2.5	0.72	0.781	0.883	**3**	3/1
DPC-CE	0.716	0.769	0.513	3	9/2.5	0.568	0.759	0.777	2	10/2.5
DEMOS	0.16	0.311	0.492	12	5/0.5	0.568	0.759	0.777	2	9/1
LGD	0.813	0.782	0.633	4	5/0.58/1	0	0	0.167	1	28/0.6/1
BP	0.9	0.866	0.732	4	9/3	0.566	0.741	0.776	2	30/3
	Dermatology					WDBC				
EC	**0.852**	**0.918**	**0.884**	5	8/6	**0.792**	**0.702**	**0.94**	**2**	5/10
DBSCAN	0.432	0.651	0.578	3	1.35/25	0.309	0.331	0.778	1	0.35/21
HDBSCAN	0.173	0.319	0.405	1	9/20	0.206	0.161	0.721	1	5/20
DP	0.839	0.841	**0.884**	10	5/2	0.08	0.267	0.381	170	10/0.5
DPC-CE	0.837	0.798	0.803	11	5/2	0.111	0.29	0.459	41	9/1
DEMOS	0.468	0.726	0.674	5	5/1.5	0	0	0.385	1	17/1
LGD	0.769	0.867	0.748	4	21/0.54/1	0.713	0.602	0.918	**2**	21/0.6/2
BP	0.026	0.021	0.647	1	30/3	0.437	0.736	0.621	23	9/3
	Penbased					HTRU2				
EC	**0.776**	**0.847**	**0.851**	28	15/10	**0.809**	**0.668**	**0.916**	**2**	36/12
DBSCAN	0.522	0.733	0.767	11	0.35/35	0.513	0.298	0.764	**2**	0.1/41
HDBSCAN	0.112	0.386	0.38	4	10/5	0.485	0.309	0.774	1	23/10
DP	0.509	0.73	0.64	174	7/2.5	0.02	0.174	0.496	41	4/2.5
DPC-CE	0.588	0.739	0.705	71	3/2.5	0.02	0.171	0.575	63	10/2.5
DEMOS	0.728	0.832	0.824	13	23/1	0.447	0.385	0.859	3	41/0.5
LGD	0.75	0.835	0.823	16	46/0.6/1	0	0	0.476	1	50/0.6/2.5
BP	0.73	0.818	0.817	19	46/3	0.397	0.365	0.687	5	50/3

图4(a)和图4(b)分别展示了在人工数据集Square和SF上的性能变化. 从两幅子图中可以发现,侵蚀聚类在它们的上方区域和中间偏上区域分别达到了ARI的峰值. 首先观察近邻数目k对算法性能的影响. 在两个数据集上,随着k的增加,算法性能整体呈现增长趋势,并逐渐趋于稳定. 其次观察侵蚀层数L对算法性能的影响. 在Square数据集上,随着侵蚀层数L的增加,算法的性能整体呈现下降趋势. 而在SF数据集上,随着L的增加,算法性能先上升后下降. 这两种不同的变化趋势可能源于这两个数据集具有不同的分布特征. 在Square数据集中,由于两个类簇之间的间隔较大,因此在进行聚类时不需要进行过多的侵蚀操作(即L较低时),就能够获得不错的聚类效果;然而,随着侵蚀层数的增加,可能会错误地剔除核心数据,导致过度划分问题,从而严重影响聚类性能. 而SF数据集存在类簇重叠现象. 因此,随着侵蚀迭代次数的增加,类簇间隔变得更加清晰,从而提高了聚类的表现. 然而,当侵蚀层数过高时,也会出现过度划分的问题.

图4(c)和图4(d)分别展示了在真实数据集Dermatology和Penbased上的性能变化. 从两个子图的结果可以看出,这两个数据集的ARI值都在下方偏右区域达到峰值,并且二者性能变化趋势基本相似. 近邻数目方面,随着k值的增加,获取的局部信息逐渐增多,因此聚类表现呈现上升趋势. 但是,当k取值过大时,近邻中可能会混杂其他类簇的数据,从而导致错误的侵蚀和分配操作,进而影响聚类表现. 在侵蚀层数方面,随着L的增加,算法性能呈现先上升后下降的趋势. 需要注意的是,在较低的侵蚀层数下,该算法的性能并未达到峰值. 这可能是因为这两个数据集的数据分布形式更为复杂,类簇的重叠程度可能远高于SF数据集. 在该类情况下为了检测出真实的类簇核心并实现数据的合理划分,需要进行更多的侵蚀迭代操作.

综合上述分析,对于实际应用场景,近邻数目的选择范围可为[10,30]. 而侵蚀层数的设置则要根据数据集的分布复杂程度而定:数据分布越复杂、重叠程度越高,则L取值应越大.

为了检验EC算法对单次侵蚀比例γ变化的鲁棒性,本实验在上述四组数据集上对该参数进行了评估. 实验中,γ的取值范围被设定为[4%,20%](默认为10%). 图5展示了在不同γ取值下,四组数据集上ARI的变动情况. 具体而言,在两组人工数据集上,算法的聚类效果展现出了高度的稳定性;而在两组真实数据



(a) Square上的ARI指标 (b) SF上的ARI指标

(c) Dermatology上的ARI指标 (d) Penbased上的ARI指标

图4 不同L和k下EC的性能变化

集上,随着γ值的增加,算法性能初期略有提升,随后迅速趋于稳定. 这可能是由于真实数据集相较于人工数据集具有更显著的类簇重叠问题,导致在较小比例的侵蚀下,算法难以揭示真实的类簇结构.

综上所述,本算法在γ的敏感性测试中展现出稳定的聚类效果,说明EC算法对单次侵蚀比例γ的取值并不敏感. 在实际应用中,单次侵蚀比例的选择范围可设定为[7%, 20%],亦可直接设为10%(本文默认值).

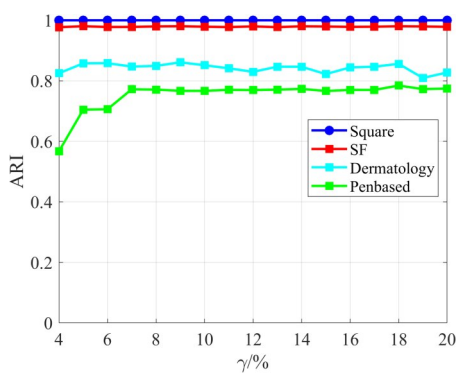


图5 不同γ值下EC的性能变化

4.5 时间对比

为了展示各个算法的运行效率,本小节设计了19个数据集,其数据量在1 000至10 000之间变化. 图6展示了这些数据集上所有算法的运行时间. 每个算法在每个数据集上进行了10次运行,图中显示了这10次结果的平均时间和标准差. 从结果可以发现,本文算法的运行时间与DBSCAN算法相当,明显快于其他7种对比算法. 在处理包含10 000个数据的数据集时,本文算法仅需约2 s,而与本算法最相似的BP和LGD算法分别需要约9 s和13 s. 正如3.7节所分析的那样,本文算法采用KD树进行近邻搜索,极大地提高了算法的运行效率.

4.6 消融实验

在本小节中,进行了一系列的消融实验来验证本算法的四个关键贡献的有效性:动态密度估计方法、侵蚀策略、基于互可达阈值的核心数据聚类方式和基于局部密度峰的边界数据分配方式. 首先,分别移除动态密度估计方法的两个重要组成部分:自适应缩放因子(Adaptive Scale Factor, ASF)和动态机制(Dynamic Mechanism, DM). 将缩放因子固定为常数,并将修改后的算法命名为EC-ASF.

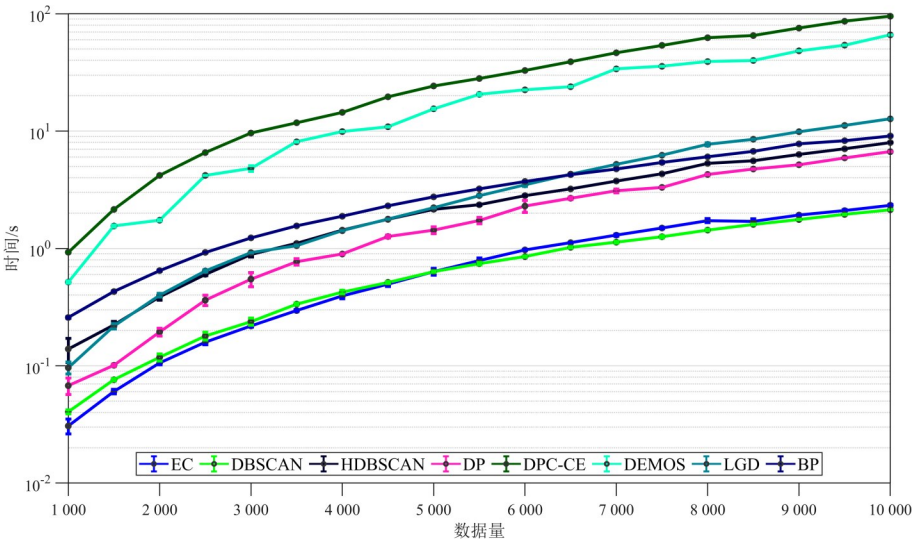


图6　运行时间的比较

移除了密度估计方法中的动态机制,改用常规的静态估计方法,即数据的密度值在侵蚀迭代过程中不再重新计算. 修改后的算法命名为EC-DM. 移除动态密度估计方法(Dynamic Density Estimation,DDE),即同时移除自适应缩放因子和动态机制. 修改后的算法命名为EC-DDE. 随后,从算法中移除侵蚀策略(Erosion Strategy,ES),并将修改后的算法命名为EC-ES. 接着,移除互可达阈值中ε半径(Epsilon Radius,ER),并将互可达阈值统一设定为λ距离. 修改后的算法命名为EC-ER. 最后,移除基于局部密度峰(Local Density Peak,LDP)的边界数据分配方式,而是采用一种最简单直观的分配方式,即直接将侵蚀数据与其最近的内层数据相连. 修改后的算法命名为EC-LDP. 所有修改后的算法在表1中的12组数据集上进行了测试,算法参数取值范围与EC的取值范围保持一致.

消融实验的详细结果见表5. 表中列出了EC算法及其各修改版本在全部12组数据集上的平均表现. 本文依然采用ARI、NMI和F_1三种指标进行评价.

表5　消融实验结果

算法	ARI	AMI	F_1
EC	**0.913**	**0.896**	**0.945**
EC-ASF	0.892	0.866	0.913
EC-DM	0.873	0.853	0.924
EC-DDF	0.790	0.789	0.851
EC-ES	0.588	0.661	0.736
EC-ER	0.828	0.826	0.895
EC-LDP	0.616	0.702	0.814

综合来看,EC-DM聚类性能下降最轻微,剩下依次为EC-ASF、EC-ER、EC-DDE和EC-LDP,而EC-ES聚类性能下降最严重. 以下将依据表格中的顺序,逐一分析每个修改后算法的性能. 消融实验验证了动态密度估计方法、侵蚀策略、基于互可达阈值的核心数据聚类方式和基于局部密度峰的边界数据分配方式均可显著提高所提算法的整体聚类性能.

4.7　噪声实验

为评估侵蚀聚类算法对噪声的鲁棒性,本节在三组包含噪声的数据集上进行了测试. 实验选取了三种不同类型的数据集:具有不同密度的Jain数据集、存在区域重叠的Handl数据集,以及类簇形状复杂的T8数据集. 本实验在这三组数据集上各自引入了占其原始数据量5%的随机噪声,并将这些新生成的数据集分别命名为Jain+、Handl+和T8+. EC算法及所有对比算法均在上述三组数据集上进行了测试,其参数设置与表3中的相应参数保持一致.

图7展示了所有算法在三组数据集上的可视化效果. 图中使用不同的颜色标识不同类簇的数据,其中黑色圆圈表示噪声数据. 此外,表6列出了8种算法的聚类评价指标. 表中粗体数值表示各数据集对应指标的最高得分. 列“c”显示了算法产生的类簇数量及是否识别出噪声,其中“+”前数字代表识别的类簇数量,“+”后数字“1”或“0”分别表示是否识别出噪声. 粗体标记表明该算法既正确识别了类簇数量,也检测到了噪声.

在8种算法中,EC、DBSCAN、HDBSCAN及BP四种算法在三组数据集上展现了噪声识别的能力. 其余四种算法未能展现出这一能力. 具体而言,DBSCAN算法存在误将低密度类簇识别为噪声的问题,例如在处理Jain+数据集时,它错误地将上方的稀疏类簇归类为噪声. HDBSCAN算法则仅标注了部分密集区域的数据,导致大部分数据被错误地识别为噪声. BP算法尽管能够较为准确地识别噪声,但仍存在过度分割的问题. 相比之下,侵蚀聚类算法在三个数据集上均获得了最高分,并且在Jain+数据集

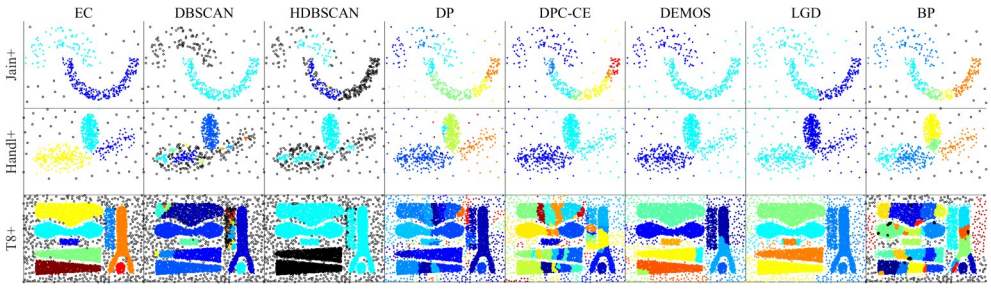


图7 具有噪声的数据集上的可视化效果

表6 算法在具有噪声的数据集上的表现

算法	ARI	AMI	F_1	c	ARI	AMI	F_1	c	ARI	AMI	F_1	c
	Jain+				Handl+				T8+			
EC	**1**	**1**	**1**	**2+1**	**0.975**	**0.951**	**0.983**	**3+1**	**0.991**	**0.981**	**0.987**	**8+1**
DBSCAN	0.951	0.906	0.731	1+1	0.690	0.639	0.546	9+1	0.928	0.919	0.885	23+1
HDBSCAN	0.168	0.344	0.491	**2+1**	0.354	0.364	0.503	1+1	0.274	0.547	0.334	1+1
DP	0.217	0.373	0.417	6+0	0.809	0.741	0.737	16+0	0.598	0.760	0.649	72+0
DPC-CE	0.231	0.495	0.523	7+0	0.692	0.724	0.562	2+0	0.380	0.677	0.565	56+0
DEMOS	0.877	0.839	0.683	2+0	0.692	0.724	0.562	2+0	0.814	0.848	0.780	10+0
LGD	0.067	0.209	0.449	2+0	0.694	0.725	0.565	2+0	0.822	0.855	0.600	5+0
BP	0.345	0.666	0.854	5+1	0.676	0.799	0.902	5+1	0.377	0.728	0.684	34+1

上取得了最佳的聚类效果. 说明其在处理噪声数据时展现了极高的鲁棒性.

5 总结

本文提出了侵蚀聚类算法. 该算法将基于激活互近邻的动态密度估计方法引入到侵蚀策略中,能有效处理类簇边界模糊的问题. 本算法设计了一种基于互可达图的核心数据聚类方式,较好地解决了数据密度不均的问题. 此外,本算法还提出了基于局部密度峰的边界数据分配策略,极大地提高了聚类的准确性. 实验数据表明,本算法的聚类性能总体优于7种典型对比算法.

参考文献

[1] HOTELLING H. A Generalized T Test and measure of multivariate dispersion[M]//Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California Press, 1951: 23-42.

[2] XU L, JORDAN M I. On convergence properties of the EM algorithm for Gaussian mixtures[J]. Neural Computation, 1996, 8(1): 129-151.

[3] SHI J B, MALIK J. Normalized cuts and image segmentation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888-905.

[4] BEN-HUR A, HORN D, SIEGELMANN H T, et al. Support vector clustering[J]. Journal of Machine Learning Research, 2001, 2(Dec): 125-137.

[5] WANG C D, LAI J H. Position regularized support vector domain description[J]. Pattern Recognition, 2013, 46(3): 875-884.

[6] PENG C, ZHANG Q, KANG Z, et al. Kernel two-dimensional ridge regression for subspace clustering[J]. Pattern Recognition, 2021, 113: 107749.

[7] PENG C, ZHANG Y Q, CHEN Y Y, et al. Log-based sparse nonnegative matrix factorization for data representation[J]. Knowledge-Based Systems, 2022, 251: 109127.

[8] PENG C, ZHANG J, CHEN Y Y, et al. Preserving bilateral view structural information for subspace clustering[J]. Knowledge-Based Systems, 2022, 258: 109915.

[9] ESTER M, H-P KRIEGEL, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining. New York: ACM, 1996: 226-231.

[10] CHENG Y Z. Mean shift, mode seeking, and clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995, 17(8): 790-799.

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

[12] CHOWDHURY S, NA H L, CORDEIRO DE AMORIM R. Feature weighting in DBSCAN using reverse nearest neighbours[J]. Pattern Recognition, 2023, 137: 109314.

[13] DING S F, DU W, XU X, et al. An improved density peaks clustering algorithm based on natural neighbor with a merging strategy[J]. Information Sciences, 2023, 624: 252-276.

[14] SUN C, DU M J, SUN J R, et al. A three-way clustering method based on improved density peaks algorithm and boundary detection graph[J]. International Journal of Approximate Reasoning, 2023, 153: 239-257.
[15] ANKERST M, BREUNIG M M, KRIEGEL H P, et al. OPTICS: Ordering points to identify the clustering structure [C]//Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data. New York: ACM, 1999: 49-60.
[16] CAMPELLO R J G B, MOULAVI D, SANDER J. Density-based clustering based on hierarchical density estimates[M]// Lecture Notes in Computer Science. Berlin: Springer, 2013: 160-172.
[17] QIAN L, PLANT C, BOHM C. Density-based clustering for adaptive density variation[C]//2021 IEEE International Conference on Data Mining (ICDM). Piscataway: IEEE, 2021: 1282-1287.
[18] 徐晓, 丁世飞, 丁玲. 密度峰值聚类算法研究进展[J]. 软件学报, 2022, 33(5): 1800-1816.
XU X, DING S F, DING L. Survey on density peaks clustering algorithm[J]. Journal of Software, 2022, 33(5): 1800-1816. (in Chinese)
[19] 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.
[20] LOTFI A, MORADI P, BEIGY H. Density peaks clustering based on density backbone and fuzzy neighborhood[J]. Pattern Recognition, 2020, 107: 107449.
[21] GUO W J, WANG W H, ZHAO S P, et al. Density Peak Clustering with connectivity estimation[J]. Knowledge-Based Systems, 2022, 243: 108501.
[22] WANG Y Z, WANG D, ZHANG X F, et al. McDPC: Multi-center density peak clustering[J]. Neural Computing and Applications, 2020, 32(17): 13465-13478.
[23] GUAN J Y, LI S, HE X X, et al. Clustering by fast detection of main density peaks within a peak digraph[J]. Information Sciences, 2023, 628: 504-521.
[24] GUAN J Y, LI S, CHEN X J, et al. DEMOS: Clustering by pruning a density-boosting cluster tree of density mounts[J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(10): 10814-10830.
[25] CHEN B, TING K M, WASHIO T, et al. Local contrast as an effective means to robust clustering against varying densities[J]. Machine Learning, 2018, 107(8): 1621-1645.
[26] CHEN J G, YU P S. A domain adaptive density clustering algorithm for data with varying density distribution[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33 (6): 2310-2321.
[27] WANG Y Z, WANG D, ZHOU Y, et al. VDPC: Variational density peak clustering algorithm[J]. Information Sciences, 2023, 621: 627-651.
[28] LI R J, YANG X F, QIN X L, et al. Local gap density for clustering high-dimensional data with varying densities[J]. Knowledge-Based Systems, 2019, 184: 104905.
[29] AVERBUCH-ELOR H, BAR N, COHEN-OR D. Border-peeling clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(7): 1791-1797.
[30] DING S, DU M, SUN T, et al. An entropy-based density peaks clustering algorithm for mixed type data employing fuzzy neighborhood[J]. Knowledge-Based Systems, 2017, 133: 294-313.

作者简介

杜明晶　男, 1989 年 2 月出生于江苏省徐州市. 现为江苏师范大学计算机科学与技术学院副教授、硕士生导师. 主要研究方向为聚类分析、时序分析等.
E-mail: dumj@jsnu.edu.cn

吴福玉　男, 2000 年 2 月出生于江苏省连云港市. 现为江苏师范大学计算机科学与技术学院硕士研究生. 主要研究方向为聚类分析、隐私计算等.
E-mail: fuyu@jsnu.edu.cn

李宇蕊　女, 1998 年 1 月出生于河北省石家庄市. 现为江苏师范大学计算机科学与技术学院硕士研究生. 主要研究方向为时序分析、聚类分析等.
E-mail: lyrchris417@gmail.com

董永权　男, 1979 年 7 月出生于江苏省宿迁市. 现为江苏师范大学计算机科学与技术学院教授、硕士生导师. 主要研究方向为机器学习、推荐系统等.
E-mail: tomdyq@163.com