文章编号:1671-9352(2025)01-0029-16 DOI:10.6040/j.issn.1671-9352.7.2023.4130

基于自然邻居搜索优化策略的密度峰值聚类算法

张春昊[1],解滨[1,2,3*],徐童童[1],张喜梅[1]
(1.河北师范大学计算机与网络空间安全学院, 河北 石家庄 050024; 2.供应链大数据分析与数据安全河北省工程研究中心(河北师范大学), 河北 石家庄 050024; 3.河北省网络与信息安全重点实验室(河北师范大学), 河北 石家庄 050024)

摘要:结合自然邻居搜索算法改进了密度峰值聚类(clustering by fast search and find of density peaks, CFSFDP)算法存在的一系列问题,提出基于自然邻居搜索优化策略的密度峰值聚类(density peak clustering algorithm optimized by natural neighbor search, NaN-CFSFDP)算法。基于自然邻居搜索算法提出了一种离群样本的检测方法,针对 CFSFDP 算法中截断距离 d_c 人工准确取值较难的问题,结合自然邻居搜索算法改进了 d_c 的计算方式,实现了 d_c 的自动取值。重新设计并统一了 CFSFDP 算法的样本密度度量规则,使其更关注每个样本的局部信息。由于数据集中因类簇间的密度差异大,密度峰值点集中于稠密簇使得簇丢失,因此提出样本共享自然邻居和类簇共享自然邻居的概念,构造新的类簇融合算法。合成数据集和真实数据集上的实验结果表明,在大多数情况下,NaN-CFSFDP 算法在聚类性能上优于或至少与比较方法相当,且与CFSFDP算法及其改进算法相比参数更少。

关键词:密度峰值;自然邻居;聚类;类簇融合;离群样本;截断距离
中图分类号:TP391; TP181 **文献标志码**:A
引用格式:张春昊,解滨,徐童童,等.基于自然邻居搜索优化策略的密度峰值聚类算法[J].山东大学学报(理学版),2025,60(1):29-44.

Density peak clustering algorithm optimized by natural neighbor search

ZHANG Chunhao[1], XIE Bin[1,2,3*], XU Tongtong[1], ZHANG Ximei[1]
(1. College of Computer and Cyber Security, Hebei Normal University, Shijiazhuang 050024, Hebei, China; 2. Hebei Provincial Engineering Research Center for Supply Chain Big Data Analytics & Data Security, Hebei Normal University, Shijiazhuang 050024, Hebei, China; 3. Hebei Provincial Key Laboratory of Network & Information Security, Hebei Normal University, Shijiazhuang 050024, Hebei, China)

Abstract: We combine the natural neighbor search algorithm to improve a series of problems of the density peaks clustering (CFSFDP) algorithm, and propose the NaN-CFSFDP algorithm. First, an outlier samples detection method is proposed based on the natural neighbor search algorithm. Then, for the problem that the truncation distance d_c is difficult to be taken accurately manually in the CFSFDP algorithm, the calculation of d_c is improved in combination with the natural neighbor search algorithm, and the automatic taking of d_c is realized. The metric rule of the sample density of the CFSFDP algorithm is redesigned and unified to make it pay more attention to the local information of each sample. Finally, to address the problem that the density peak points in the dataset may be concentrated in dense clusters due to the large density difference between clusters, which leads to cluster loss, the concepts of shared natural neighbors for samples and shared natural neighbors for clusters are proposed to construct a new cluster fusion algorithm. Experimental results on synthetic and real datasets show that the algorithm outperforms or is at least comparable to the comparative method in terms of clustering performance in most cases and has fewer parameters compared to CFSFDP algorithm and its improvements.

Key words: density peaks; natural neighbor; clustering; cluster fusion; outlier samples; truncation distance

---

收稿日期:2023-05-06; **网络出版时间**:2023-11-08 17:29:10
基金项目:国家自然科学基金资助项目(62076088);中央科研院所基本科研业务费资助项目(SK202324); 河北师范大学技术创新基金资助项目(L2020K09)
第一作者:张春昊(1996—),男,硕士研究生,主要研究方向为机器学习与数据挖掘. E-mail: zhangchunhao_hebtu@ 163.com
***通信作者**:解滨(1976—),男,教授,博士,主要研究方向为机器学习、粒计算与智能数据分析. E-mail: xiebin_hebtu@ 126.com

0 引言

在数据挖掘领域中，以聚类为代表的无监督学习方法由于其训练数据集无须提前标注，仅通过比较样本之间的相似程度即可对样本进行分组，具有更好的泛化性，因此，受到越来越多研究者的关注。从广义上讲，聚类的应用包括搜索引擎、社交网络、图像分割和脱氧核糖核酸（deoxyribonucleic acid，DNA）分析等[1]。

研究者依据不同的思想提出了多种聚类算法，大致可分为基于分区、基于网格、基于密度和基于层次的聚类算法[2]。其中以 k-means[3] 和模糊 c 均值[4]（fuzzy c-means，FCM）为代表的基于分区的聚类算法运行速度较快，k-means 聚类算法对样本进行硬划分，FCM 算法通过迭代使目标函数最小化，将隶属度矩阵去模糊化得到样本属于每一个簇的确切划分；但是，由于初始类簇中心的位置随机选取，因此算法的运行结果不稳定，且该类算法对于非球形类簇的聚类效果不理想。基于网格的聚类算法通过设置不同的网格粒度减小算法的时间复杂度，但是如何设置合适的网格粒度是研究者讨论的热点内容[5]。层次聚类算法[6]分为凝聚式层次聚类（agglomerative nesting，AGNES）算法和分裂式层次聚类（divisive analysis，DIANA）算法，目的是通过迭代合并或拆分聚类来将数据集分解为各个类簇，但该类算法受样本数据的体量影响过大，当样本数量较多或维度较高时，该算法的时间成本也较高，且容易聚类成链状。以密度聚类（a density-based algorithm for discovering clusters in large spatial databases with noise，DBSCAN）算法[7]和密度峰值聚类（clustering by fast search and find of density peaks，CFSFDP）算法[8]为代表的基于密度的聚类算法因能够识别任意形状的类簇而受到广泛的关注，但是 DBSCAN 算法的最小样本数和邻域半径无法自适应选择，对于不同的数据集须要人工设定不同的参数值，给算法应用带来一定困难。CFSFDP 算法的稳定性易受类簇密度分布情况以及样本密度度量方式的影响，比如针对数据集中不同类簇的密度分布不均匀的情况，会出现密度大的类簇包含多个密度峰值，而密度小的类簇很少甚至没有密度峰值，因此选出来的类簇中心均在高密度簇中，导致后续剩余样本的分配会发生连带错误。另外，CFSFDP 算法对于不同规模的数据集须要采用不同的样本密度度量方式，但对于数据集规模的衡量并没有合适的界定规则[9]。

针对上述问题，研究者提出了多种改进的 CFSFDP 算法。Xie 等[10]提出了一种结合模糊 K 近邻的密度峰值聚类算法（robust clustering by detecting density peaks and assigning points based on fuzzy weighted K nearest neighbors，FKNN-DPC），该算法通过结合样本 K 近邻的信息统一了样本密度的度量方式，并提出了 2 种分配策略来检测数据集的真实分布，聚类性能较 CFSFDP 算法有所提升，但是 K 值是人工预先设定的，密度峰值通过分析决策图人工寻找。如何自动选择 K 值并自动发现数据集的密度峰值仍须要进一步研究。Jiang 等[11]针对 CFSFDP 算法在处理一些非球形数据集容易引发“多米诺骨牌”效应的缺陷，提出了将 K 近邻的思想融入到距离计算和分配过程中的密度峰值聚类算法（a novel density peaks clustering algorithm based on k nearest neighbors for improving assignment process，DPC-KNN），避免了非聚类中心点的错误分配，但并未解决 K 值自动选择及截断距离 d_c 和 K 之间的关系问题。Zhang 等[12]基于共享 K 近邻设计了一种冗余高密度核心区域的自动融合机制改进的密度峰值聚类算法（adaptive density-based clustering algorithm with shared k nearest neighbor conflict game，DC-SKCG），降低了算法对参数的敏感性，但同时又引入了新的参数，且提高了算法的复杂程度。Liu 等[13]提出了一种自适应聚类算法（adaptive density peak clustering based on K nearest neighbors with aggregating strategy，ADPC-KNN），该算法引入 KNN 的概念来计算全局参数 d_c 和每个点的局部密度 ρ_i，最后聚集密度可达的簇，但如何实现 K 值的自动选择仍有待研究。Bai 等[14]提出了一种涉及较少距离计算的加速算法（fast density clustering strategies based on the k-means algorithm，CFSFDP+A），可以获得与原算法相同的聚类结果，且提高了算法的运行速度，但算法的密度度量不统一、分配方式易发生连带错误等问题仍然存在。Liu 等[15]提出了一种基于共享近邻的密度峰值聚类算法（shared-nearest-neighbor-based clustering by fast search and find of density peaks，SNN-DPC），该算法基于共享近邻提高了 CFSFDP 算法在多尺度、交叉缠绕以及变化密度数据集上的优越性能，但无法人为避免选择聚类中心的主观性。张新元等[16]在 SNN-DPC 基础上引入放大因子重新定义样本局部密度，提出了共享 K 近邻和多分配策略的密度峰值聚类算法（sharing k nearest neighbors and multiple assignment policies density peaks clustering algorithm，SKM-DPC），改进了分配策略，聚类性能较 SNN-DPC 有所提升，但还无法避免人为选择

聚类中心的主观性。Zhang 等[17]针对 CFSFDP 算法存在样本密度度量方式不统一,在剩余样本分配上易发生多米诺骨牌效应以及容易错选密度峰值作为聚类中心的现象,提出了一种基于反向最近邻优化的快速搜索和发现密度峰值聚类算法(reverse-nearest-neighbor-based clustering by fast search and find of density peaks, RNN-CFSFDP),但该算法中反向最近邻居的 K 值须要事先给定,在面对不同数据集时仍然不能自适应地选择。徐童童等[18]认为 CFSFDP 算法通过截断距离定义局部密度未考虑样本的空间分布特征,因此提出一种自适应聚类中心策略优化的密度峰值聚类算法(density peak clustering algorithm optimized by adaptive clustering centers strategy, SNN-ADPC),然而,SNN-ADPC 算法样本 K 近邻中的参数 K 是提前给定的,如何实现 K 值自适应将在未来工作中进一步研究。张春昊等[19]为了提高算法的泛化能力,设计了一种自适应近邻优化的密度峰值聚类算法和改进的 FCM 算法,融合二者优势,提出了一种结合自适应近邻与密度峰值的基于信息熵加权的模糊聚类算法(weighted fuzzy clustering algorithm combining adaptive nearest neighbors and density peaks, ANNDP-WFCM),但是 FCM 算法通过迭代寻优,容易陷入局部最优。

在现有的研究基础上,为了进一步解决 CFSFDP 算法存在的问题,首先基于自然邻居搜索算法提出了一种离群样本的检测方法;然后,针对因固定扫描半径而出现假峰现象的问题[20],提出结合样本自然特征值的截断距离 d_c 的计算方式,实现 d_c 的动态化,使其更关注每个样本的局部信息;再次重新设计并统一CFSFDP 算法在不同规模数据集上局部密度的度量方式;最后,针对数据集中每个类簇的密度不同,在人工选取类簇中心的时候,可能导致稀疏簇没有选到密度峰值,造成簇丢失,从而严重影响聚类效果的问题,提出样本共享自然邻居和类簇共享自然邻居的概念,并设计不同类簇之间相似度的度量方式,提出一种类簇融合的算法。

1　相关工作

1.1　自然邻居

K 近邻(k nearest neighbor, KNN)和反向 K 近邻[21](reverse k nearest neighbor, RKNN)是许多算法的基础,但它们都容易受到参数 K 值的影响。与 KNN 和 RKNN 相比,自然邻居[22-25](natural neighbor, NaN)是一种自适应邻域,可以根据样本所处位置的密度情况,自适应地选择近邻数量,且无须人工预先设定 K,能够更好地反映数据的局部特征。

自然邻居是关于邻居的一种新的刻画方式。自然邻居算法的灵感来自于人类社会的友谊,当且仅当 2 个人相互友好,才能称 2 个人之间存在友谊关系。当仅有一方对另一方友好时,2 人之间关系只能称之为单边友谊。将友谊和友好两个概念抽象为样本之间的关系,如果 2 个样本互为对方的近邻样本,则称 2 个样本构成自然邻居,仅有一个样本是对方的近邻样本则不构成自然邻居,其中的 K 通过自适应得到。

一个人真正的朋友的数量应该是把他(她)当作朋友并同时被他(她)视为朋友的人的数量。如果每个人都至少有一个朋友,社会就形成了一个基于朋友关系的相对稳定结构,称为自然稳定结构。对于数据集而言,如果样本 $\boldsymbol{x}$ 视样本 $\boldsymbol{y}$ 为邻居,同时样本 $\boldsymbol{y}$ 视样本 $\boldsymbol{x}$ 为邻居,则样本 $\boldsymbol{y}$ 是样本 $\boldsymbol{x}$ 的自然邻居之一。位于稀疏区域的样本有更少的自然邻居,而位于密集区域的样本则有更多的自然邻居,因此,样本自然邻居数量可以更好地反映样本局部的密度。

若数据集为 $\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$,则样本 $\boldsymbol{x}_i$ 的前 r 个近邻的集合定义为

$$\boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_i)}=\bigcup_{n=1}^{r}\{F(\boldsymbol{x}_i,n)\}, \tag{1}$$

式中,$F(\boldsymbol{x}_i,n)$ 表示返回样本 $\boldsymbol{x}_i$ 的第 n 个最近邻的搜索函数,$\boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_i)}$ 是 $\boldsymbol{X}$ 的子集。

定义 1(稳定搜索状态)　只有满足公式(2)情况下,自然邻居搜索过程才能达到稳定的搜索状态,

$$(\forall \boldsymbol{x}_i)(\exists \boldsymbol{x}_j)(r\in N)\wedge(\boldsymbol{x}_i\neq\boldsymbol{x}_j)\rightarrow(\boldsymbol{x}_i\in\boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_j)})\wedge(\boldsymbol{x}_j\in\boldsymbol{X}_{\mathrm{KNN}r(\boldsymbol{x}_i)}), \tag{2}$$

其中,r 为搜索轮数。

定义 2(自然邻居特征值)　当算法达到稳定搜索状态时,自然邻居特征值 λ 等于搜索轮数 r,即

$$\lambda\triangleq r_{r\in N}\{r\mid(\forall \boldsymbol{x}_i)(\exists \boldsymbol{x}_j)(r\in N)\wedge(\boldsymbol{x}_i\neq\boldsymbol{x}_j)\rightarrow(\boldsymbol{x}_i\in\boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_j)})\wedge(\boldsymbol{x}_j\in\boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_i)})\}, \tag{3}$$

定义 3(自然邻居)　样本 $\boldsymbol{x}_i$ 的自然邻居定义为

$$\boldsymbol{x}_j \in \boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_i)} \Leftrightarrow (\boldsymbol{x}_i \in \boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_j)}) \wedge (\boldsymbol{x}_j \in \boldsymbol{X}_{\mathrm{KNN}_r(\boldsymbol{x}_i)}), \tag{4}$$

其中 $\boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_i)}$ 为样本 $\boldsymbol{x}_i$ 的自然邻居集合。

1.2 CFSFDP 算法

CFSFDP 算法依赖于 2 个重要的假设[26]:(1)类簇中心的样本密度大于围绕它的邻居样本的密度;(2)不同类簇中心之间的距离相对较远。

在 CFSFDP 算法中将样本 $\boldsymbol{x}_i$ 的局部密度 ρ_i 定义为分布在其截断距离范围内的样本个数,即

$$\rho_i = \sum_j \chi(d_{ij} - d_{\mathrm{c}}), \tag{5}$$

$$\chi(d_{ij} - d_{\mathrm{c}}) = \begin{cases} 1, & d_{ij} < d_{\mathrm{c}}, \\ 0, & \text{其他}, \end{cases} \tag{6}$$

式中,d_{ij}是样本 $\boldsymbol{x}_i$ 和样本 $\boldsymbol{x}_j$ 之间的欧氏距离,d_{c} 是用户根据具体情况须要输入的参数,根据 CFSFDP 算法[8],通常使截断距离 d_{c} 内样本邻居的平均数量为数据集总数的 1%~2%。

此外,在实际应用中,通常采用高斯核函数来计算样本密度,主要解决算法处理小样本数据时密度变化不明显的缺陷,即

$$\rho_i = \sum_{j \neq i} \exp\left(-\left(\frac{d_{ij}}{d_{\mathrm{c}}}\right)^2\right)。 \tag{7}$$

样本 i 的相对距离 δ_i 为

$$\delta_i = \min_{j:\rho_j > \rho_i} \{d_{ij}\}。 \tag{8}$$

如果样本具有最大密度,则其相对距离 δ_i 定义为与数据集中最远的样本之间的距离为

$$\delta_i = \max\{d_{ij}\}。 \tag{9}$$

以 ρ_i 为横坐标,δ_i 为纵坐标绘制决策图,从决策图中选出 ρ_i 和 δ_i 都相对较大的点作为密度峰值点。如果决策图不明显,可求出每个样本的 $\gamma_i = \rho_i \delta_i$,将其升序或者降序排列作为纵坐标绘制决策图。选出 γ_i 相对较大的点作为类簇中心,最后将除密度峰值点之外的剩余样本分配给密度比它大的最近邻样本所在的簇。

此外,与其他基于密度的聚类方法类似,CFSFDP 算法还将样本分为边界点、核心点和噪声点 3 类。将每簇中与其他簇中的样本的距离小于截断距离 d_{c} 的样本构成的区域,定义为该簇的边界区域。每个簇的边界区域都是属于该簇的一组样本,将类簇边界区域内样本密度最高值记为 ρ_{b},将该类簇中满足样本密度 $\rho_i \geqslant \rho_{\mathrm{b}}$ 的点定义为核心点,其他点(包括边界区域中的点)则被视为噪声。

2 相关算法的缺陷分析

2.1 自然邻居搜索算法分析

在自然邻居搜索算法中提到了噪声样本的判定规则[22],只有当除噪声样本外的所有样本都达到稳定的搜索状态时,仍没有自然邻居的样本才属于噪声,即数据集中没有自然邻居的样本,在连续$\sqrt{\lambda}$次搜索后仍没有变化,则将这些没有自然邻居的样本判定为噪声样本。该噪声样本判定规则对于若干个相互距离较近但整体距离类簇较远的噪声样本无法检测出来。另外,在判断噪声点的过程中,作者并未给出以连续$\sqrt{\lambda}$轮迭代作为其判断标准的理由。

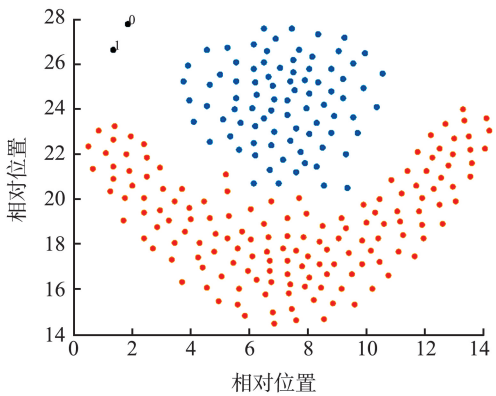


图 1 Flame 数据集的可视化

Fig.1 Visualization of Flame dataset

如图 1 所示,该图为合成数据集 Flame[27] 的样本分布情况,在自然邻居搜索算法的第一轮搜索中,样本 0 和样本 1 便会互相成为对方的自然邻居,而一旦 2 个样本成为自然邻居,这种状态一直持续到算法结束都不会改变,因此导致 2 个最明显的离群样本无法检测出来。结合了自然邻居搜索算法在自适应搜索邻域方面的优势,重新定义了离群样本的判定规则,去除明显的离群样本,然后再

对剩余样本进行聚类,最后对去除的离群样本再次分配,将最终仍未分配的样本判定为噪声样本。

2.2　CFSFDP 算法分析

CFSFDP 算法在截断距离 d_c 的选择和决策图中密度峰值点数量的选取均受到人工取值的影响。在数据规模较大的数据集中采用截断核函数度量样本密度[8],在数据规模较小的数据集中采用高斯核函数度量样本密度,但是在研究过程中如何衡量数据集的规模方面没有统一化的标准。在面临真实问题时,研究者该选择何种样本密度的度量方式成为难题。另外,无论选择何种密度度量方式,均须要用到截断距离 d_c,而截断距离 d_c 须要人为设定,当截断距离 d_c 内样本邻居数为样本总数的 1%~2%基本能取得更好的效果,但该截断方式缺乏理论证明,且在很多数据集上并不能取得很好的结果。图 2 展示了在 Flame 数据集上采用 2 种截断函数和不同的截断距离的聚类表现。其中,最好的聚类结果,出现在选取 d_c=3%的时候,超出了原文中给出的取值范围。可以看出,不同的截断函数和不同的截断距离都会导致不同的聚类结果,因此截断函数和截断距离的选择尤为重要。针对这一系列问题,结合自然邻居搜索算法重新定义了样本局部密度的度量规则,无须考虑数据集的规模和截断距离 d_c 选择的问题,根据样本自身局部邻域的密度,自适应地选择截断距离。

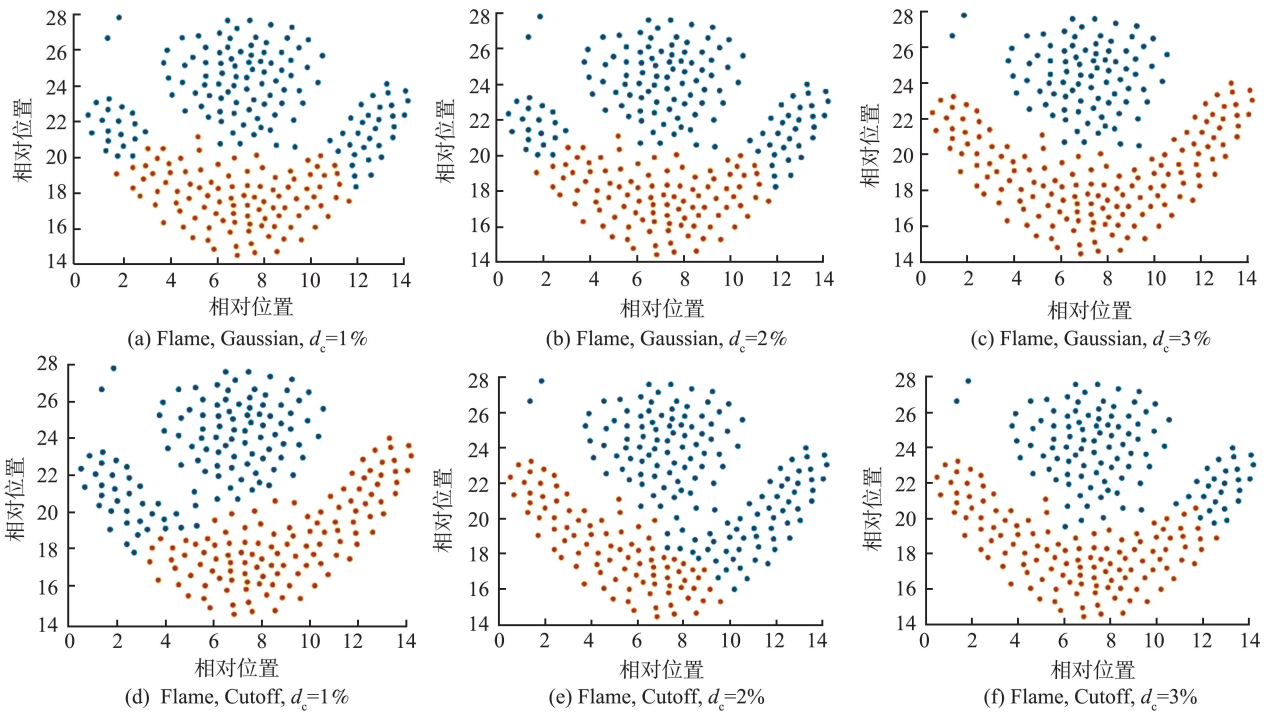


图 2　采用两种截断方法和不同截断距离在 Flame 数据集上的聚类结果比较

Fig.2　Comparison of clustering results using two cutoff methods and different cutoff distances on Flame dataset

虽然 CFSFDP 算法的决策图为类簇中心的选择提供了很好的启发式方法;但由于有些数据集密度峰值点和次密度峰值点区分不明显以及同一数据集的不同类簇的稠密情况不一致等原因,因此仍存在难以选择或者选择错误的情况。其主要原因在于,受数据集中类簇的形状和密度分布的影响,数据集中存在多个密度更高、相对距离更大的样本,导致类簇中心点无法从决策图中直观地被选择出来。另外,受不同类簇间密度差异变化的影响,低密度簇类簇中心和高密度簇类簇中心之间的密度和相对距离存在显著差异,导致选出来的类簇中心可能全部位于高密度簇,而低密度簇没有发现类簇中心。

如图 3 所示,选取 Aggregation 数据集[28]和 Jain 数据集[29],根据簇心选择原则,从决策图中人工选择相对较大的点作为类簇中心。Aggregation 数据集是由 7 个类簇构成的数据集,仅仅从它的决策图中很难通过观察决策图直接选出 7 个密度峰值点,可能会被误选为 3 个或者 8 个。Jain 数据集由密度分布不均匀的两簇构成,从决策图中虽然可以很明显地选择出 2 个密度峰值点,但是这 2 个密度峰值点都位于密度相对较高的簇中,因此导致聚类效果不理想。

针对这一系列问题,本文重新规定了从决策图选取密度峰值点的规则,提高了 CFSFDP 算法根据决策图

选择密度峰值的鲁棒性。另外,结合自然邻居的概念,本文定义了样本共享自然邻居和类簇共享自然邻居,并设计了类簇间的相似度,通过将多余的相似类簇进行融合的办法有效解决了 CFSFDP 算法存在的密度峰值多选或者密度峰值集中在高密度簇的问题。

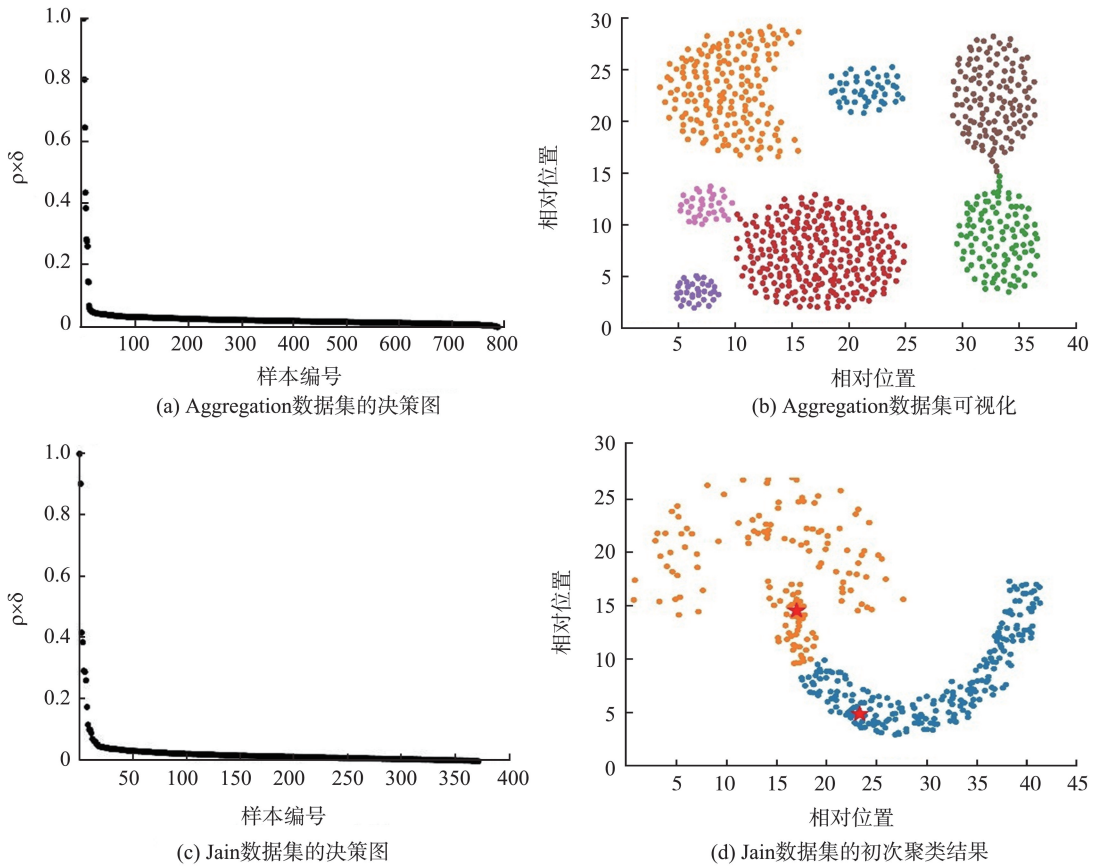


图 3 Aggregation、Jain 数据集的聚类结果

Fig.3 Clustering results for Aggregation and Jain datasets

3 CFSFDP 算法优化

3.1 结合自然邻居搜索算法的噪声点去除

首先将自然邻居搜索算法中提到的噪声点检测方法命名为单离群点检测方法。在此基础上,定义了多离群点的检测方法,即

$$\lambda_{\max(\boldsymbol{x}_i)}=\max_{j\in \boldsymbol{X}_{\lambda\_\mathrm{NN}(\boldsymbol{x}_i)}}\{d_{ij}\}, \tag{10}$$

$$\mu=\frac{1}{N}\sum_{i=1}^{N}\lambda_{\max(\boldsymbol{x}_i)}, \tag{11}$$

$$\boldsymbol{X}_{\mathrm{Multiple\_Outlier}}=\{\boldsymbol{x}_i\mid\lambda_{\max(\boldsymbol{x}_i)}>2\mu\}, \tag{12}$$

$$\boldsymbol{X}_{\mathrm{Outlier}}=\boldsymbol{X}_{\mathrm{Single\_Outlier}}\cup\boldsymbol{X}_{\mathrm{Multiple\_Outlier}}, \tag{13}$$

式中,λ 为每个数据集的自然特征值,在不同数据集中通过自然邻居搜索算法自适应得到,为一个固定值,$\lambda_{\max(\boldsymbol{x}_i)}$ 为样本 $\boldsymbol{x}_i$ 和第 λ 个近邻样本(即最远的近邻)的距离,$\boldsymbol{X}_{\lambda\_\mathrm{NN}(\boldsymbol{x}_i)}$ 为样本 $\boldsymbol{x}_i$ 的 λ 个近邻的集合。$\boldsymbol{X}_{\mathrm{Multiple\_Outlier}}$ 为多离群点的集合,$\boldsymbol{X}_{\mathrm{Outlier}}$ 为单离群点和多离群点的并集,代表最终的离群点的集合。由于每个样本的 λ 个近邻在自然邻居的搜索过程中已经得到,因此该寻找噪声点的方法并未增加算法的时间复杂度量级,且 λ 通过自适应得到未引入任何参数,也不须要任何人为干预。

在求解离群点集合的过程中,首先结合样本自身的自然邻居分布情况,通过式(10)—(11)自适应地计算每一个样本的邻域范围,然后通过式(12)统计超出该邻域范围内的样本并视其为多离群点集合,最后通

过式(13)整合最终的离群点集合为单离群点集合和多离群点集合的并集。

3.2　结合自然邻居搜索算法的局部密度度量和离群样本再分配

将自然邻居搜索算法和 CFSFDP 算法结合,重新定义截断距离 d_c,根据自然邻居搜索算法,每个数据集的自然特征值不同,因此重新定义的 d_c 可以根据不同数据集,以及样本局部的密度自动调整截断距离 d_c,弱化了人工干预。另外,本节统一了样本局部密度的度量方式,对于任何数据集均采用同一种局部密度的度量方式,提高了 CFSFDP 算法的泛化性、实用性和可操作性。d_c 定义为

$$d_c = \mu + \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\lambda_{\max(\boldsymbol{x}_i)} - \mu)^2}, \tag{14}$$

$$\rho_i = \sum_{j \in \boldsymbol{X}_{\lambda\_NN(\boldsymbol{x}_i)}} \exp\left(-\frac{d_{ij}^2}{d_c^2}\right), \tag{15}$$

其中,μ 为样本 $\boldsymbol{x}_i$ 和第 λ 个近邻(即最远的近邻)样本的距离的平均值,由式(11)定义,N 为数据集中的样本数量。d_c 为样本的截断距离,定义为 $\lambda_{\max(\boldsymbol{x}_i)}$ 的均值和标准差的和,可以根据样本的形状、分布等自动选择,无须进行参数选择和人工干预。

计算出每个样本的 ρ_i 和 δ_i 之后,绘制 γ_i 决策图,在决策图中选择密度峰值点时,不会出现选错、选多的情况,提高了人工选取密度峰值点过程的鲁棒性,具有更强的可操作性。然后按照 CFSFDP 算法的分配方式,将除离群样本和密度峰值点之外的剩余样本进行分配。最后,将离群样本分配给最近邻样本所在的簇,直到离群样本的数量不再变化,则停止分配。剩下的样本被彻底判定为噪声点,不再进行分配。

3.3　结合自然邻居的多簇融合

结合自然邻居,本节提出了共享自然邻居的概念,根据共享自然邻居定义不同类簇的相似度。依次合并相似度高的类簇。

定义 4(样本共享自然邻居集)　对于任意 2 个样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$,若存在样本 $\boldsymbol{x}_k$, $\boldsymbol{x}_k \neq \boldsymbol{x}_i$ 且 $\boldsymbol{x}_k \neq \boldsymbol{x}_j$,使得 $\boldsymbol{x}_k \in \boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_i)}$ 且 $\boldsymbol{x}_k \in \boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_j)}$,那么称样本 $\boldsymbol{x}_k$ 构成的集合 $\boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{x}_i,\boldsymbol{x}_j)}$ 为样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的共享自然邻居集,即

$$\boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{x}_i,\boldsymbol{x}_j)} = \{\boldsymbol{x}_k \mid \boldsymbol{x}_k \in \boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_i)} \text{ 且 } \boldsymbol{x}_k \in \boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_j)}\}, \tag{16}$$

式中,$\boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_i)}$ 为样本 i 的自然邻居,$\boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_j)}$ 为样本 j 的自然邻居。

样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的共享自然邻居集 $\boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{x}_i,\boldsymbol{x}_j)}$ 具有如下特性:如果样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的共享自然邻居数量较少,意味着样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的距离较远且相似度较低。如果样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的共享自然邻居数量较多,意味着样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的距离较近且相似度较高。样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的共享自然邻居数量和样本所处位置的密度有关,因此,在相同条件下,如果样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 所处位置较为稠密,样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 相似度较高,反之,样本 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 相似度较低。

定义 5(类簇共享自然邻居集)　对于任意 2 个类簇 $\boldsymbol{C}_m$ 和 $\boldsymbol{C}_n$,其中 $\boldsymbol{x}_i \in \boldsymbol{C}_m$, $\boldsymbol{x}_j \in \boldsymbol{C}_n$,若存在 $\boldsymbol{x}_k \in \boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{x}_i,\boldsymbol{x}_j)}$,则称 $\boldsymbol{x}_k$ 构成的集合 $\boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{C}_m,\boldsymbol{C}_n)}$ 为类簇 $\boldsymbol{C}_m$ 和 $\boldsymbol{C}_n$ 的类簇共享自然邻居集,即

$$\boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{C}_m,\boldsymbol{C}_n)} = \{\boldsymbol{x}_k \mid \boldsymbol{x}_k \in \boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{x}_i,\boldsymbol{x}_j)},\ \boldsymbol{x}_i \in \boldsymbol{C}_m,\ \boldsymbol{x}_j \in \boldsymbol{C}_n\}, \tag{17}$$

类簇 $\boldsymbol{C}_m$ 和 $\boldsymbol{C}_n$ 的类簇共享自然邻居集 $\boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{C}_m,\boldsymbol{C}_n)}$ 具有以下特性:如果类簇 $\boldsymbol{C}_m$ 和 $\boldsymbol{C}_n$ 的共享自然邻居数量较少,意味着类簇 $\boldsymbol{C}_m$ 和 $\boldsymbol{C}_n$ 的距离较远且相似度较低。如果类簇 $\boldsymbol{C}_m$ 和 $\boldsymbol{C}_n$ 的共享自然邻居数量较多,意味着类簇 $\boldsymbol{C}_m$ 和 C_n 的距离较近且相似度较高。

定义 6(类簇相似度)　2 个类簇的共享自然邻居集的个数为

$$\mathrm{Sim}(\boldsymbol{C}_m,\boldsymbol{C}_n) = |\boldsymbol{X}_{\mathrm{SNaN}(\boldsymbol{C}_m,\boldsymbol{C}_n)}|, \tag{18}$$

2 个类簇的共享自然邻居集中样本个数反映着 2 个类簇的相似程度,如果 2 个类簇本属于同一类簇则其距离更近,共享自然邻居集的个数更多。

算法 1　类簇融合算法。

输入　初始聚类结果 $\boldsymbol{L} = \{\boldsymbol{C}_j\}_{j=1}^{l}$, $l \geqslant k$,实际类簇数 k。

输出　最终聚类结果 $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1}^{k}$。

(1)　根据公式(8)计算任意两簇的相似度,得到相似度矩阵 $\boldsymbol{S}_{l\times l} = \mathrm{Sim}(\boldsymbol{C}_i,\boldsymbol{C}_j)$;

(2)　while $l>k$

(3)　　合并相似度最高的两簇 $\boldsymbol{C}_i$ 和 $\boldsymbol{C}_j$ 为簇 $\boldsymbol{C}_{i,j}$,$i<j$,令 i 为该簇的簇标记;

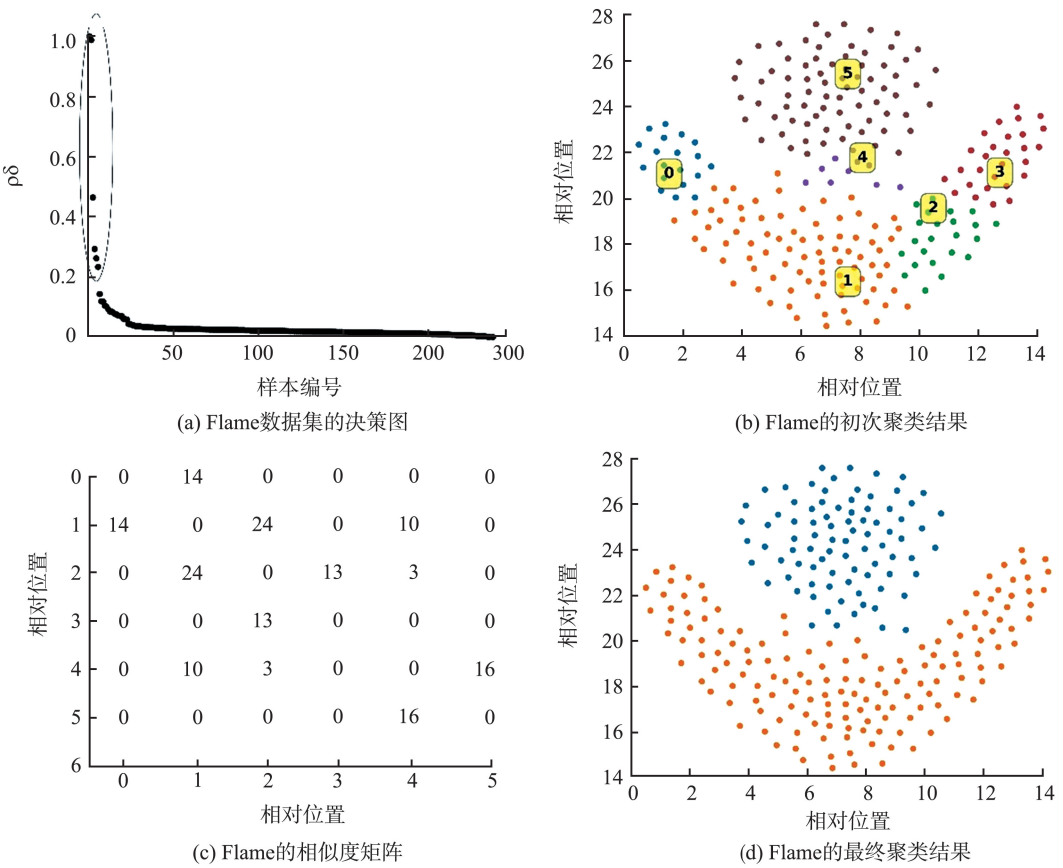
(4) 更新剩余簇 $\boldsymbol{C}_m \in \boldsymbol{L}/\{\boldsymbol{C}_i, \boldsymbol{C}_j\}$ 和簇 $\boldsymbol{C}_{i,j}$ 的相似度为 $\mathrm{Sim}(\boldsymbol{C}_m, \boldsymbol{C}_{i,j}) = \max(\mathrm{Sim}(\boldsymbol{C}_m, \boldsymbol{C}_i), \mathrm{Sim}(\boldsymbol{C}_m, \boldsymbol{C}_j))$;

(5) end while

(6) 更新簇标记为 $1 \sim k$。

3.4 NaN-CFSFDP 算法流程及复杂度分析

NaN-CFSFDP 算法流程如算法 2 所示。图 4 展示了数据集 Flame 的聚类流程,如图 4(a)所示选取密度峰值点,如图 4(b)所示生成聚类结果,如图 4(c)所示计算相似度矩阵,如图 4(d)所示生成最终聚类结果。



(a) Flame数据集的决策图 (b) Flame的初次聚类结果

(c) Flame的相似度矩阵 (d) Flame的最终聚类结果

图 4 Flame 数据集的聚类过程

Fig.4 Clustering process for Flame dataset

算法 2 NaN-CFSFDP 算法。

输入 类簇个数 k

输出 类别标签 $\boldsymbol{C} = \{\boldsymbol{C}_j\}_{j=1}^{k}$。

(1) 根据自然邻居搜索算法,返回数据集的 λ,每个样本 $\boldsymbol{x}_i$ 的 $\boldsymbol{X}_{\mathrm{NaNE\_NN}(\boldsymbol{x}_i)}$ 和 $\boldsymbol{X}_{\mathrm{NaN}(\boldsymbol{x}_i)}$ 以及 $\boldsymbol{X}_{\mathrm{Single\_Outlier}}$;

(2) 根据式(10)—(13),返回数据集的 $\boldsymbol{X}_{\mathrm{Outlier}}$;

(3) 根据式(14)计算截断距离 d_c;

(4) 根据式(15)计算每个样本的 ρ_i;

(5) 根据式(8)(9)计算每个样本的 δ_i;

(6) 计算每个样本的 $\gamma_i = \rho_i \delta_i$,降序排列,并绘制 γ_i 决策图;

(7) 选取密度峰值点;

(8) 将剩余样本分配给密度比它大的最近邻样本所在的簇;

(9) 将离群样本分配给最近邻样本所在的簇,直到离群样本数量不再变化,剩余样本为噪声点;

(10) 根据算法 1 依次合并相似度最高的两簇,直到类簇的数量为 k。

假设数据集的样本数量为 N,自然特征值为 λ,离群样本数量为 M,每个样本的自然邻居数量为 γ。

NaN-CFSFDP 算法的时间复杂度分析过程如下:(1) 步骤(1)调用自然邻居搜索算法,时间复杂度为 $O(N*\log N)$。(2) 步骤(2)须要判断每个样本是否为离群点,时间复杂度为 $O(N)$。(3) 步骤(3)计算该数据集的自适应截断距离 d_c 须要考虑全部样本,时间复杂度为 $O(N)$。(4) 步骤(4)中使用 λ 近邻样本计算每个样本的局部密度,时间复杂度 $O(\lambda*N)$,其中,$2\leqslant\lambda<N$,通过调用自然邻居搜索算法自适应得到,针对不同数据集略有差异,一般为 6 或者 7,在高维数据集或者不规则数据集,$20\leqslant\lambda<30$,因此 $\lambda\ll N$。(5) 步骤(5)计算每个样本的相对距离 δ_i,须要计算任意 2 个样本之间的距离,因此时间复杂度为 $O(N^2)$。(6) 步骤(6)—(8)分配剩余样本的时间复杂度为 $O(N)$。(7) 步骤(9)分配离群样本的过程中,由于每个样本的最近邻样本在步骤(1)已经得到,而且离群样本的数量远小于样本总数,因此该过程时间复杂度为 $O(M)$($M\ll N$)。(8) 步骤(10)合并相似类簇的过程中,须要判断任意 2 个样本是否构成自然邻居关系,因此时间复杂度为 $O(\gamma*N^2)$, $1\leqslant\gamma\leqslant\lambda$。综上所述,NaN-CFSFDP 算法最终的时间复杂度为 $O(\gamma*N^2)$。

在传统 CFSFDP 算法中,因为计算任意样本的局部密度 ρ_i 时,须要考虑其他所有样本,所以时间复杂度为 $O(N^2)$。计算任意 2 个样本的相对距离 δ_i 时间复杂度为 $O(N^2)$。最后,分配剩余样本的时间复杂度约为 $O(N)$,传统 CFSFDP 算法最终的时间复杂程度为 $O(N^2)$。NaN-CFSFDP 算法与 CFSFDP 算法的时间复杂度量级相同。

4　实验结果与分析

在如表 1 所示的 18 个数据集上将 NaN-CFSFDP 算法与 k-means[3]、FCM[4]、AGNES[6]、CFSFDP[8]、DPC-KNN[11]、SNN-DPC[15]和 SKM-DPC[16]算法进行了比较,采用 FM 指数[12](Fowlkes-Mallows index, FMI)、准确率(accuracy, Acc)、调整互信息[10](adjusted mutual information, AMI)、标准互信息[30](normalized mutual information, NMI)以及调整兰德指数[31](adjusted Rand index, ARI)评价指标验证了提出算法的有效性,5 个评价指标分别用 D_{FMI}、D_{Acc}、D_{AMI}、D_{NMI} 和 D_{ARI} 来表示。所提到的算法的实验环境为 Windows 10 64bit 操作系统,PyCharm Community 2020.3.2,12 GB 内存,Intel(R) Core(TM) i5-4210H CPU @ 2.90 GHz。

4.1　数据集

实验采用的数据集选自具有不同类簇数、不同规模、不同形状、不同密度的合成数据集和真实数据集。如加州大学欧文分校机器学习(University of California Irvine Machine Learning Repository, UCI)数据集,Flame 数据集是由 2 个类簇构成的半包围结构的数据集,其中一个类簇被另一个类簇紧紧包围,且 2 个类簇之间紧密相连。Aggregation 数据集由分布较为均匀的 7 个类簇构成,其中有 2 对类簇有轻微连接。Spiral 数据集由 3 个类簇构成,每个类簇都是环形。Jain 数据集为 2 个月牙形类簇交替连接,2 个类簇密度差距较大。R15、S2、D31 数据集属于样本数和类簇数都较多的数据集。对比实验所采用的数据集细节和来源如表 1 所示。

表 1　本文对比实验数据集

Table 1　Comparative experiment datasets in this paper

名称	样本数	特征数	类簇数	数据来源
Flame	240	2	2	文献[27]
Aggregation	788	2	7	文献[28]
Spiral	312	2	3	文献[32]
Jain	373	2	2	文献[29]
R15	600	2	15	文献[33]
D31	3 100	2	31	文献[33]
4k2_far	400	2	4	文献[12]
G2_2_30	2 048	2	2	文献[12]
S2	5 000	2	15	文献[10]
Compound	399	2	6	文献[34]
Unbalance	6 500	2	8	文献[35]
Iris	150	3	3	UCI 数据集

续表

名称	样本数	特征数	类簇数	数据来源
Leuk72_3k	72	39	3	文献[12]
Haberman	306	3	2	UCI 数据集
Libras_movement	360	90	15	UCI 数据集
Ecoli	336	7	8	UCI 数据集
Spect	187	22	2	UCI 数据集
Bupa	345	6	2	UCI 数据集

4.2 评价指标

D_{FMI}是对聚类结果和真实值计算得到的召回率和精确率进行几何平均的结果,取值范围为 $0\leqslant D_{\mathrm{FMI}}\leqslant 1$,越接近 1 越好。$D_{\mathrm{AMI}}$和 D_{NMI}是对互信息(mutual information, MI)的一种改进,$-1\leqslant D_{\mathrm{AMI}}\leqslant 1$, $0\leqslant D_{\mathrm{NMI}}\leqslant 1$, D_{NMI}取值越大说明算法的聚类结果越好。D_{ARI}的前身是兰德系数(Rand index, RI), $-1\leqslant D_{\mathrm{ARI}}\leqslant 1$,$D_{\mathrm{ARI}}$越接近 1,表明算法的聚类质量越好。

4.3 实验结果分析

表 2 展示了 *k*-means、FCM、AGNES、CFSFDP、DPC-KNN、SNN-DPC 和 SKM-DPC 算法同 NaN-CFSFDP 算法在不同评价指标上的比较结果。表 2 实验结果表明,NaN-CFSFDP 算法在 10 个数据集上所有评价指标均取得了最好的结果,在 Leuk72_3k、Spect 和 Bupa 数据集上超过半数的指标取得了最好的结果。在 Aggregation和 Compound 数据集上仅次于最好的结果。在 D31 数据集上 5 个指标低于 *k*-means 算法,主要原因在于 D31 数据集由球形类簇构成,使得 *k*-means 算法能够更好地发挥其优势。综合在合成数据集和真实数据集上的实验结果,在大多数情况下,NaN-CFSFDP 算法在聚类性能上优于或至少与比较方法相当,且与 CFSFDP 算法及其改进算法相比须要更少的参数。

表 2 不同算法在不同数据集上的比较

Table 2 Comparison of different algorithms on different datasets

数据集	算法	评价指标				
		D_{FMI}	D_{Acc}	D_{AMI}	D_{NMI}	D_{ARI}
Flame	*k*-means	0.736 4	0.837 5	0.396 9	0.398 8	0.453 4
	FCM	0.753 0	0.850 0	0.440 3	0.442 0	0.488 0
	AGNES	0.731 1	0.833 3	0.481 4	0.483 1	0.442 2
	CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	DPC-KNN	0.795 2	0.783 0	0.880 7	0.882 4	0.739 0
	SNN-DPC	0.949 4	0.966 7	0.805 3	0.806 7	0.890 4
	SKM-DPC	0.939 6	0.954 2	0.799 6	0.801 9	0.872 9
	NaN-CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
Aggregation	*k*-means	0.815 8	0.784 3	0.877 6	0.879 2	0.762 2
	FCM	0.691 7	0.633 2	0.759 8	0.762 9	0.611 3
	AGNES	0.994 9	0.996 2	0.989 4	0.989 6	0.993 5
	CFSFDP	0.770 1	0.751 3	0.873 6	0.875 4	0.708 4
	DPC-KNN	0.823 7	0.850 3	0.902 6	0.903 9	0.776 6
	SNN-DPC	0.968 1	0.978 4	0.954 8	0.955 5	0.959 4
	SKM-DPC	**0.996 6**	**0.997 5**	**0.992 3**	**0.992 4**	**0.995 6**
	NaN-CFSFDP	0.994 9	0.996 2	0.989 4	0.989 6	0.993 5
Spiral	*k*-means	0.327 9	0.342 9	−0.005 2	0.000 7	−0.005 7
	FCM	0.327 2	0.339 7	−0.005 7	0.000 2	−0.006 2
	AGNES	0.349 9	0.381 4	0.071 3	0.013 0	0.004 6
	CFSFDP	0.903 8	0.948 7	0.864 1	0.864 9	0.855 5

续表

数据集	算法	评价指标				
		D_{FMI}	D_{Acc}	D_{AMI}	D_{NMI}	D_{ARI}
Spiral	DPC-KNN	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	SNN-DPC	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	SKM-DPC	0.776 2	0.676 3	0.736 0	0.736 9	0.577 1
	NaN-CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
Jain	*k*-means	0.700 5	0.785 5	0.367 7	0.369 0	0.324 1
	FCM	0.689 4	0.774 8	0.354 1	0.355 5	0.300 4
	AGNES	0.921 8	0.946 4	0.695 6	0.696 4	0.779 2
	CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	DPC-KNN	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	SNN-DPC	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	SKM-DPC	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	NaN-CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
R15	*k*-means	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
	FCM	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
	AGNES	0.990 0	0.995 0	0.991 6	0.992 2	0.989 3
	CFSFDP	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
	DPC-KNN	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
	SNN-DPC	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
	SKM-DPC	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
	NaN-CFSFDP	**0.993 2**	**0.996 7**	**0.993 8**	**0.994 2**	**0.992 8**
D31	*k*-means	**0.955 0**	**0.977 1**	**0.966 0**	**0.967 5**	**0.953 5**
	FCM	0.830 4	0.846 8	0.919 6	0.923 4	0.823 6
	AGNES	0.926 2	0.961 9	0.949 5	0.951 9	0.923 8
	CFSFDP	0.937 8	0.968 1	0.954 8	0.956 9	0.9358
	DPC-KNN	0.939 2	0.968 7	0.956 7	0.958 7	0.9372
	SNN-DPC	0.952 5	0.975 8	0.964 2	0.965 9	0.9509
	SKM-DPC	0.953 8	0.976 5	0.965 1	0.966 7	0.9523
	NaN-CFSFDP	0.942 8	0.970 6	0.958 2	0.960 2	0.9409
4k2_far	*k*-means	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	FCM	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	AGNES	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	DPC-KNN	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	SNN-DPC	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	SKM-DPC	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	NaN-CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
G2_2_30	*k*-means	0.981 6	0.990 7	0.924 5	0.924 5	0.963 2
	FCM	0.982 6	0.991 2	0.927 6	0.927 7	0.965 1
	AGNES	0.974 9	0.987 3	0.911 4	0.911 4	0.949 8
	CFSFDP	0.985 4	0.992 7	0.937 5	0.937 5	0.970 9
	DPC-KNN	0.986 4	0.993 2	0.941 1	0.941 1	0.972 8
	SNN-DPC	0.975 4	0.986 8	0.904 5	0.904 5	0.950 8
	SKM-DPC	0.893 7	0.934 6	0.709 6	0.709 8	0.790 8
	NaN-CFSFDP	**0.988 9**	**0.994 4**	**0.951 0**	**0.951 1**	**0.977 8**
S2	*k*-means	0.941 2	0.969 6	0.945 5	0.945 9	0.937 1
	FCM	0.941 9	0.970 0	0.945 7	0.946 1	0.937 8
	AGNES	0.918 4	0.957 4	0.930 2	0.930 8	0.912 6

续表 2

数据集	算法	评价指标				
		D_{FMI}	D_{Acc}	D_{AMI}	D_{NMI}	D_{ARI}
S2	CFSFDP	0.935 2	0.966 2	0.940 9	0.941 4	0.930 6
	DPC-KNN	0.943 1	0.970 6	0.946 2	0.946 6	0.939 1
	SNN-DPC	0.924 5	0.960 8	0.930 9	0.931 4	0.919 1
	SKM-DPC	0.655 6	0.701 8	0.742 2	0.744 2	0.617 6
	NaN-CFSFDP	**0.947 5**	**0.973 0**	**0.954 3**	**0.954 7**	**0.943 8**
Compound	*k*-means	0.642 2	0.656 6	0.713 5	0.719 2	0.537 9
	FCM	0.640 4	0.656 6	0.704 4	0.710 3	0.535 7
	AGNES	0.861 6	0.862 2	0.831 4	0.835 3	0.803 0
	CFSFDP	0.625 1	0.631 6	0.753 9	0.758 9	0.511 6
	DPC-KNN	0.647 3	0.644 1	0.730 8	0.736 2	0.543 5
	SNN-DPC	**0.895 9**	0.869 7	**0.897 1**	**0.899 1**	**0.850 4**
	SKM-DPC	0.845 4	0.832 1	0.822 6	0.829 6	0.769 8
	NaN-CFSFDP	0.891 2	**0.890 1**	0.877 6	0.879 5	0.836 6
Unbalance	*k*-means	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	FCM	0.912 0	0.832 8	0.909 5	0.909 7	0.876 1
	AGNES	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
	CFSFDP	**1.000 0**	0.999 8	0.999 4	0.999 4	**1.000 0**
	DPC-KNN	**1.000 0**	**1.000 0**	0.998 8	0.998 8	**1.000 0**
	SNN-DPC	0.989 7	0.991 5	0.971 0	0.971 0	0.985 6
	SKM-DPC	0.308 5	0.245 4	0.164 8	0.166 7	0.106 4
	NaN-CFSFDP	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**	**1.000 0**
Iris	*k*-means	0.820 8	0.893 3	0.755 1	0.758 2	0.730 2
	FCM	0.819 7	0.893 3	0.746 5	0.749 6	0.729 4
	AGNES	0.840 7	0.906 7	0.803 2	0.805 7	0.759 2
	CFSFDP	0.590 5	0.730 3	0.450 8	0.456 6	0.372 8
	DPC-KNN	0.840 7	0.906 7	0.803 2	0.805 7	0.759 2
	SNN-DPC	**0.947 9**	**0.973 3**	**0.913 3**	**0.914 4**	**0.922 2**
	SKM-DPC	0.935 6	0.966 7	0.898 4	0.899 7	0.903 9
	NaN-CFSFDP	0.841 0	0.905 4	0.802 2	0.804 7	0.759 0
Leuk72_3k	*k*-means	0.919 7	**0.958 3**	**0.855 8**	**0.859 6**	0.880 3
	FCM	0.919 7	**0.958 3**	**0.855 8**	**0.859 6**	0.880 3
	AGNES	0.919 9	**0.958 3**	0.838 1	0.842 4	0.880 5
	CFSFDP	**0.920 5**	**0.958 3**	0.855 5	0.859 3	**0.880 9**
	DPC-KNN	**0.920 5**	**0.958 3**	0.855 5	0.859 3	**0.880 9**
	SNN-DPC	**0.920 5**	**0.958 3**	0.855 5	0.859 3	**0.880 9**
	SKM-DPC	0.820 2	0.888 9	0.695 4	0.703 7	0.726 2
	NaN-CFSFDP	**0.920 5**	**0.958 3**	0.855 5	0.859 3	**0.880 9**
Haberman	*k*-means	0.550 7	0.519 6	−0.001 7	0.000 9	−0.001 1
	FCM	0.550 6	0.509 8	−0.002 6	0.000 0	−0.002 8
	AGNES	0.749 6	0.735 3	**0.017 5**	**0.021 6**	**0.064 3**
	CFSFDP	0.560 9	0.539 2	−0.002 1	0.000 6	−0.006 7
	DPC-KNN	0.599 0	0.562 1	0.016 5	0.019 4	−0.043 6
	SNN-DPC	**0.780 7**	0.735 3	0.000 0	0.000 0	0.000 0
	SKM-DPC	**0.780 7**	0.735 3	0.000 0	0.000 0	0.000 0
	NaN-CFSFDP	0.778 5	**0.739 9**	−0.006 2	0.000 5	0.003 7
Libras_movement	*k*-means	0.352 0	0.477 8	0.532 5	0.588 3	0.303 3
	FCM	0.210 5	0.130 6	0.063 8	0.092 5	0.025 2

续表

数据集	算法	评价指标				
		D_{FMI}	D_{Acc}	D_{AMI}	D_{NMI}	D_{ARI}
Libras_movement	AGNES	0.365 4	0.447 2	0.563 4	0.615 7	0.315 4
	CFSFDP	0.308 8	0.402 8	0.488 1	0.545 9	0.237 2
	DPC-KNN	0.394 3	0.488 9	0.560 2	0.612 5	0.345 8
	SNN-DPC	0.328 1	0.438 9	0.462 1	0.510 6	0.242 8
	SKM-DPC	0.341 0	0.250 0	0.417 1	0.442 9	0.172 0
	NaN-CFSFDP	**0.442 6**	**0.513 9**	**0.600 8**	**0.647 9**	**0.395 7**
Ecoli	*k*-means	0.546 0	0.559 5	0.591 3	0.608 3	0.409 0
	FCM	0.511 8	0.497 0	0.532 2	0.551 4	0.368 2
	AGNES	0.609 2	0.636 9	0.618 8	0.634 7	0.485 8
	CFSFDP	0.478 1	0.488 1	0.486 7	0.508 4	0.314 9
	DPC-KNN	0.618 0	0.625 0	0.553 9	0.572 6	0.495 5
	SNN-DPC	0.661 9	0.723 2	0.516 9	0.534 2	0.536 1
	SKM-DPC	0.772 6	0.770 8	0.636 3	0.648 2	0.669 4
	NaN-CFSFDP	**0.817 3**	**0.782 2**	**0.680 0**	**0.684 3**	**0.721 7**
Spect	*k*-means	0.529 4	0.529 4	0.000 0	0.000 0	−0.033 1
	FCM	0.513 4	0.513 4	0.036 9	0.042 4	−0.006 7
	AGNES	0.753 1	0.738 0	0.037 2	0.044 7	−0.083 9
	CFSFDP	0.715 6	0.689 8	−0.004 5	0.002 1	−0.019 9
	DPC-KNN	0.811 1	0.807 5	0.022 0	0.031 7	−0.080 4
	SNN-DPC	0.674 2	0.566 8	0.037 7	0.046 0	**0.042 9**
	SKM-DPC	0.633 3	0.454 5	−0.002 8	0.007 5	0.006 3
	NaN-CFSFDP	**0.917 1**	**0.919 8**	**0.083 0**	**0.088 4**	0.000 0
Bupa	*k*-means	0.640 7	0.553 6	−0.002 0	0.000 9	−0.005 4
	FCM	0.610 2	0.524 6	0.004 2	0.006 8	−0.011 4
	AGNES	0.712 4	0.576 8	−0.001 1	0.004 5	−0.001 6
	CFSFDP	0.559 3	0.547 8	−0.001 9	0.000 4	0.001 3
	DPC-KNN	0.711 0	0.580 5	0.008 8	0.014 5	0.004 3
	SNN-DPC	**0.715 0**	0.420 3	0.000 0	0.000 0	0.000 0
	SKM-DPC	**0.715 0**	0.420 3	0.000 0	0.000 0	0.000 0
	NaN-CFSFDP	0.657 7	**0.617 4**	**0.028 4**	**0.031 2**	**0.039 0**

4.4　消融实验

为了进一步验证每个改进模块的有效性,本节提出了 CFSFDP 算法的 3 种变体:(1)CFSFDP_A 算法。该算法在 CFSFDP 算法的基础上结合自然邻居搜索算法补充了离群点的检测与去除。(2)CFSFDP_B 算法。通过将 CFSFDP_A 算法的样本密度计算方式改进为公式(15)自适应得到。(3)CFSFDP_C 算法。在 CFSFDP_B 算法的基础上引入了类簇融合算法。3 个变种算法的参数取相同值,由于 5 个评价指标的实验结果相似,因此以 D_{FMI}为代表展示消融实验的结果。CFSFDP 算法和 3 个种变种算法的 D_{FMI}如表 3 所示。实验结果表明,CFSFDP_A 算法和 CFSFDP 算法相比在 D_{FMI}指标上互有高低,这是由剔除了离群点导致的结果。总体来说,在 18 个数据集上有 14 个数据集在 D_{FMI}指标上高于原算法或者与其相当。CFSFDP_B 算法在 CFSFDP_A 算法的基础上通过结合自然邻居样本改进了样本局部密度的计算方式,实现了截断距离的自适应,从 D_{FMI}的指标结果来看,在 18 个数据集上有 15 个数据集相较于 CFSFDP_A 算法有了进一步提高或与其效果相当,验证了该部分内容的有效性。CFSFDP_C 算法和 CFSFDP 算法相比在 18 个数据集上均取得了最好的结果。考虑到引入类簇融合算法的目的是防止密度峰值错选并提高算法在流形数据集上的有效性,总体来看,该算法满足了目标期望。整体来说,改进的算法和 CFSFDP 算法相比具有聚类效果更佳,最重要的是须要的参数更少。

表 3 CFSFDP 算法和 3 个变种算法的 D_{FMI} 比较

Table 3 Comparison of D_{FMI} of CFSFDP algorithm and 3 variants of the algorithm

数据集	算法	D_{FMI}	数据集	算法	D_{FMI}
Flame	CFSFDP	**1.000 0**	Aggregation	CFSFDP	0.770 1
	CFSFDP_A	**1.000 0**		CFSFDP_A	0.914 7
	CFSFDP_B	**1.000 0**		CFSFDP_B	0.919 8
	CFSFDP_C	**1.000 0**		CFSFDP_C	**0.994 9**
Spiral	CFSFDP	0.903 8	Jain	CFSFDP	**1.000 0**
	CFSFDP_A	0.510 8		CFSFDP_A	0.835 5
	CFSFDP_B	0.510 8		CFSFDP_B	0.835 5
	CFSFDP_C	**1.000 0**		CFSFDP_C	**1.000 0**
R15	CFSFDP	**0.993 2**	D31	CFSFDP	0.937 8
	CFSFDP_A	**0.993 2**		CFSFDP_A	**0.942 8**
	CFSFDP_B	**0.993 2**		CFSFDP_B	**0.942 8**
	CFSFDP_C	**0.993 2**		CFSFDP_C	**0.942 8**
4k2_far	CFSFDP	**1.000 0**	G2_2_3 0	CFSFDP	0.985 4
	CFSFDP_A	**1.000 0**		CFSFDP_A	0.984 0
	CFSFDP_B	**1.000 0**		CFSFDP_B	0.986 0
	CFSFDP_C	**1.000 0**		CFSFDP_C	**0.988 9**
S2	CFSFDP	0.935 2	Compound	CFSFDP	0.625 1
	CFSFDP_A	**0.947 5**		CFSFDP_A	0.778 3
	CFSFDP_B	**0.947 5**		CFSFDP_B	0.947 5
	CFSFDP_C	**0.947 5**		CFSFDP_C	**0.891 2**
Unbalance	CFSFDP	**1.000 0**	Iris	CFSFDP	0.590 5
	CFSFDP_A	**1.000 0**		CFSFDP_A	0.677 4
	CFSFDP_B	0.778 3		CFSFDP_B	**0.841 0**
	CFSFDP_C	**1.000 0**		CFSFDP_C	**0.841 0**
Leuk72_3k	CFSFDP	**0.920 5**	Haberman	CFSFDP	0.560 9
	CFSFDP_A	**0.920 5**		CFSFDP_A	0.598 5
	CFSFDP_B	**0.920 5**		CFSFDP_B	0.761 5
	CFSFDP_C	**0.920 5**		CFSFDP_C	**0.778 5**
Libras_movement	CFSFDP	0.308 8	Ecoli	CFSFDP	0.478 1
	CFSFDP_A	0.362 2		CFSFDP_A	0.590 2
	CFSFDP_B	0.375 9		CFSFDP_B	0.585 9
	CFSFDP_C	**0.442 6**		CFSFDP_C	**0.817 3**
Spect	CFSFDP	0.715 6	Bupa	CFSFDP	0.559 3
	CFSFDP_A	0.719 0		CFSFDP_A	0.554 3
	CFSFDP_B	0.715 6		CFSFDP_B	**0.684 0**
	CFSFDP_C	**0.917 1**		CFSFDP_C	**0.684 0**

5 结论

针对 CFSFDP 算法在聚类过程中难以选择较优截断距离的问题,以及算法中提到的选择密度峰值的过程,容易造成簇多选、簇少选、簇丢失的问题,继续探索了基于局部邻居的聚类算法,并找到一种自动确定 K 值的算法,以简化算法的参数,提出了 NaN-CFSFDP 算法,有效解决了 CFSFDP 算法在聚类过程中难以选择较优截断距离的问题,以及算法中提到的选择密度峰值的过程,容易造成簇多选、簇少选、簇丢失的问题,最终实现了 K 值的自动选取。在多个选自不同形状、不同规模、不同簇数和不同密度的数据集上通过在 5 个常用的聚类评价指标上与多种聚类算法比较,验证了提出的 NaN-CFSFDP 算法的有效性。实验结果显示,

在大多数情况下,该方法在聚类性能上优于或至少与比较方法相当,且与 CFSFDP 算法及其改进算法相比须要更少的参数,但是,NaN-CFSFDP 算法可行性的论证还停留在仿真实验上,在未来的工作中会结合实际的应用背景继续优化提出的算法,增强算法的可行性和泛化能力。

参考文献:

[1] OKTAR Y, TURKAN M. A review of sparsity-based clustering methods[J]. Signal Processing, 2018, 148:20-30.

[2] SAXENA A, PRASAD M, GUPTA A, et al. A review of clustering techniques and developments[J]. Neurocomputing, 2017, 267:664-681.

[3] MACQUEEN J B. Some methods for classification and analysis of multivariate observations[C]//Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California Press, 1967:281-297.

[4] BEZDEK J C, EHRLICH R, FULL W. FCM: the fuzzy *c*-means clustering algorithm[J]. Computers and Geosciences, 1984, 10(2/3):191-203.

[5] WANG Wei, YANG Jiong, MUNTZ R. STING: a statistical information grid approach to spatial data mining[C]// Proceedings of the 23rd International Conference on Very Large Data Bases. Athens: ACM, 1997:186-195.

[6] GUHA S, RASTOGI R, SHIM K. Cure: an efficient clustering algorithm for large databases[J]. Information Systems, 2001, 26(1):35-58.

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

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

[9] 谢娟英,高红超,谢维信. *K* 近邻优化的密度峰值快速搜索聚类算法[J]. 中国科学(信息科学), 2016, 46(2):258-280.
XIE Juanying, GAO Hongchao, XIE Weixin. *K*-nearest neighbors optimized clustering algorithm by fast search and finding the density peaks of a dataset[J]. Scientia Sinica(Informationis), 2016, 46(2):258-280.

[10] XIE Juanying, GAO Hongchao, XIE Weixin, 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.

[11] JIANG Jianhua, CHEN Yujun, MENG Xianqiu, et al. A novel density peaks clustering algorithm based on *k* nearest neighbors for improving assignment process[J]. Physica A, 2019, 523:702-713.

[12] ZHANG Rui, DU Tao, QU Shouning, et al. Adaptive density-based clustering algorithm with shared KNN conflict game[J]. Information Sciences, 2021, 565:344-369.

[13] LIU Yaohui, MA Zhengming, YU Fang. Adaptive density peak clustering based on *K*-nearest neighbors with aggregating strategy[J]. Knowledge-based Systems, 2017, 133:208-220.

[14] BAI Liang, CHENG Xueqi, LIANG Jiye, et al. Fast density clustering strategies based on the *k*-means algorithm[J]. Pattern Recognition, 2017, 71:375-386.

[15] LIU Rui, WANG Hong, YU Xiaomei. Shared-nearest-neighbor-based clustering by fast search and find of density peaks[J]. Information Sciences, 2018, 450:200-226.

[16] 张新元,贠卫国. 共享 *K* 近邻和多分配策略的密度峰值聚类算法[J]. 小型微型计算机系统, 2023, 44(1):75-82.
ZHANG Xinyuan, YUN Weiguo. Sharing *K*-nearest neighbors and multiple assignment policies density peaks clustering algorithm[J]. Journal of Chinese Computer Systems, 2023, 44(1):75-82.

[17] ZHANG Chunhao, XIE Bin, ZHANG Yiran. Reverse-nearest-neighbor-based clustering by fast search and find of density peaks[J]. Chinese Journal of Electronics, 2023, 32(6):1341-1354.

[18] 徐童童,解滨,张喜梅,等. 自适应聚类中心策略优化的密度峰值聚类算法[J]. 计算机工程与应用,2023,59(21):91-101.
XU Tongtong, XIE Bin, ZHANG Ximei, et al. Density peak clustering algorithm optimized by adaptive clustering centers strategy[J]. Computer Engineering and Applications, 2023, 59(21):91-101.

[19] 张春昊,解滨,张喜梅,等. 一种结合自适应近邻与密度峰值的加权模糊聚类算法[J]. 小型微型计算机系统,2023, 44(9):1974-1982.
ZHANG Chunhao, XIE Bin, ZHANG Ximei, et al. Weighted fuzzy clustering algorithm combining adaptive nearest neighbors and density peaks[J]. Journal of Chinese Computer Systems, 2023, 44(9):1974-1982.

[20] CHEN Yewang, TANG Shengyu, PEI Songwen. DHeat: a density heat-based algorithm for clustering with effective radius [J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 48(4):649-660.

[21] BRYANT A, CIOS K. RNN-DBSCAN: a density-based clustering algorithm using reverse nearest neighbor density estimates [J]. IEEE Transactions on Knowledge and Data Engineering, 2017, 30(6):1109-1121.

[22] ZHU Qingsheng, FENG Ji, HUANG Jinlong. Natural neighbor: a self-adaptive neighborhood method without parameter *K* [J]. Pattern Recognition Letters, 2016, 80:30-36.

[23] HUANG Jinlong, ZHU Qingsheng, YANG Lijun, et al. A non-parameter outlier detection algorithm based on natural neighbor[J]. Knowledge-based Systems, 2016, 92:71-77.
[24] YANG Lijun, ZHU Qingsheng, HUANG Jinlong, et al. Adaptive edited natural neighbor algorithm[J]. Neurocomputing, 2017, 230:427-433.
[25] CHENG Dongdong, ZHU Qingsheng, HUANG Jinlong, et al. Natural neighbor-based clustering algorithm with local representatives[J]. Knowledge-based Systems, 2017, 123:238-253.
[26] 陈叶旺,申莲莲,钟才明,等. 密度峰值聚类算法综述[J]. 计算机研究与发展, 2020, 57(2):378-394.
CHEN Yewang, SHEN Lianlian, ZHONG Caiming, et al. Survey on density peak clustering algorithm[J]. Journal of Computer Research and Development, 2020, 57(2):378-394.
[27] FU Limin, MEDICO E. Flame: a novel fuzzy clustering method for the analysis of DNA microarray data[J]. BMC Bioinformatics, 2007, 8(3):1-15.
[28] GIONIS A, MANNILA H, TSAPARAS P. Clustering aggregation[J]. ACM Transactions on Knowledge Discovery from Data, 2007, 1(1):1-30.
[29] JAIN A K, LAW M H C. Data clustering: a user's dilemma[C]// International Conference on Pattern Recognition & Machine Intelligence. Kolkata: Springer, 2005:1-10.
[30] DING Shifei, DU Mingjing, SUN Tongfeng, et al. An entropy-based density peaks clustering algorithm for mixed type data employing fuzzy neighborhood[J]. Knowledge-based Systems, 2017, 133:294-313.
[31] HUBERT L, ARABIE P. Comparing partitions[J]. Journal of Classification, 1985, 2:193-218.
[32] CHANG H, YEUNG D Y. Robust path-based spectral clustering[J]. Pattern Recognition, 2008, 41(1):191-203.
[33] VEENMAN C J, REINDERS M J T, BACKER E. A maximum variance cluster algorithm[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(9):1273-1280.
[34] ZAHN C T. Graph-theoretical methods for detecting and describing gestalt clusters[J]. IEEE Transactions on Computers, 1971, C-20(1):68-86.
[35] REZAE M, FRANTI P. Set matching measures for external cluster validity[J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(8):2173-2186.

(编辑:陈丽萍)

(上接第28页)

[23] RAM P, GRAY A G. Which space partitioning tree to use for search? [C]// Annual Conference on Neural Information Processing Systems. Lake Tahoe: NeurIPS, 2013:1-9.
[24] SANJOY D, YOAV F. Random projection trees and low dimensional manifolds[C]// Annual ACM symposium on Theory of Computing. Baltimore, MD: Dove Medical Press, 2008:537-546.
[25] JIANG Kun, LU Jingshu, XIA Kuiliang. A novel algorithm for imbalance data classification based on genetic algorithm improved smote[J]. Arabian Journal for Science and Engineering, 2016, 41(8):3255-3266.
[26] WEN Liuying, ZHANG Xiaomin, MIN Fan, et al. KGA: integrating KPCA and GAN for microbial data augmentation[J]. International Journal of Machine Learning and Cybernetics, 2022, 14(4):1427-1444.
[27] 王曦,温柳英,闵帆. 融合矩阵分解和代价敏感的微生物数据扩增算法[J]. 数据采集与处理, 2023, 38(2):1-12.
WANG Xi, WEN Liuying, MIN Fan. Combining matrix decomposition and cost-sensitive microbial data augmentation algorithm [J]. Journal of Data Acquisition & Processing, 2023, 38(2):1-12.
[28] BATISTA G E A P A, PRATI C R, MONARD C R. A study of the behavior of several methods for balancing machine learning training data[J]. Association for Computing Machinery, 2004, 6(1):20-29.
[29] HE H B, BAI Y, EDWARDO A, et al. ADASYN: adaptive synthetic sampling approach for imbalanced learning[C]// IEEE International Joint Conference on Neural Networks. Atlanta, GA: IEEE, 2008:1322-1328.
[30] WANG Juanjuan, XU Mantao, WANG Hui, et al. Classification of imbalanced data by using the SMOTE algorithm and locally linear embedding[C]// International Conference on Signal Processing. Guilin: IEEE, 2006:1-4.

(编辑:陈丽萍)