控制与决策
Control and Decision

基于过滤模型的聚类算法

邱保志，张瑞霖，李向丽

引用本文:
邱保志, 张瑞霖, 李向丽. 基于过滤模型的聚类算法[J]. 控制与决策, 2020, 35(5): 1091-1101.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2018.1089

第35卷 第5期 控 制 与 决 策 Vol.35 No.5
2020年 5月 Control and Decision May 2020

文章编号: 1001-0920(2020)05-1091-11 DOI: 10.13195/j.kzyjc.2018.1089

基于过滤模型的聚类算法

邱保志†, 张瑞霖, 李向丽

(郑州大学 信息工程学院，郑州 450001)

摘 要: 合理的聚类原型是正确聚类的前提. 针对现有聚类算法原型选取不合理、计算聚类个数存在偏差等问题, 提出基于过滤模型的聚类算法(CA-FM). 算法以提出的过滤模型去除干扰聚类过程的边界和噪声对象, 依据核心对象之间的近邻关系生成邻接矩阵, 通过遍历矩阵计算聚类个数; 然后, 按密度因子将数据对象排序, 从中选出聚类原型; 最后, 将其余对象按照距高密度对象的最小距离划分到相应的簇中, 形成最终聚类. 在人工合成数据集、UCI数据集以及人脸识别数据集上的实验结果验证了算法的有效性, 与同类算法相比, CA-FM算法具有较高的聚类精度.

关键词: 聚类算法；过滤模型；偏差因子；聚类原型；局部密度；密度因子

中图分类号: TP273 **文献标志码:** A

Clustering algorithm based on filter model

QIU Bao-zhi†, ZHANG Rui-lin, LI Xiang-li

(School of Information Engineering，Zhengzhou University，Zhengzhou 450001，China)

Abstract: Reasonable clustering prototype is the premise of correct clustering. Most of the existing clustering algorithms have some shortcomings such as the unreasonable selection of clustering prototypes and calculation deviation of cluster numbers. A clustering algorithm based on filter model (CA-FM) is proposed. The algorithm uses the proposed filtering model to remove the boundary and noise objects which interfere with the clustering process. The adjacency matrix is generated according to the neighbor relationships among the core objects, and the number of clusters is calculated by traversing the matrix. Then, the objects are sorted according to the density factor, and clustering prototypes are selected from them. Finally, the remaining objects are assigned into corresponding clusters according to the minimum distance from the high density objects. The effectiveness of the proposed algorithm is demonstrated by experiments on synthetic datasets, UCI datasets and Olivetti face dataset. Compared with similar algorithms, the CA-FM has a higher clustering accuracy.

Keywords: clustering algorithm；filter model；deviation factor；clustering prototype；local density；density factor

0 引 言

聚类是一种学习范式[1], 旨在发现数据的内部结构, 它在数据探索和知识发现中扮演着重要角色. 目前已提出大量的聚类算法, 并在图像分割、生物学、电子商务、互联网等领域得到广泛应用[2-7], 如均值聚类算法(K-means)[8]、模糊均值聚类算法(fuzzy C-means clustering algorithm, FCM)[9]、基于密度和噪音的空间聚类算法(density-based spatial clustering of applications with noise, DBSCAN)[10]、密度峰值聚类算法(clustering by fast search and find of density peaks, DPC)[11]、无参数拉普拉斯中心性聚类算法(parameter-free Laplacian centrality peaks clustering, LPC)[12]、优化密度峰值聚类算法(comparative density peaks clustering, CDP)[13]等. 许多算法是通过寻找聚类骨架完成聚类, 而寻找聚类骨架的关键在于确定聚类原型集. 选取正确的聚类原型可以提高聚类精度, 如何合理地确定聚类原型集已成为聚类算法亟待解决的问题.

以K-means、FCM为代表的基于划分的聚类算法将随机选取的k个对象作为聚类的初始原型, 按照相似性原则将数据对象分配给相应的原型形成一个个簇, 通过反复计算每个簇的原型和再分配, 直至目标函数收敛. 这一机制决定了这一类算法不能有效地处理非球形簇, 且聚类精度不高.

收稿日期: 2018-08-08；**修回日期:** 2018-11-27.
基金项目: 河南省基础与前沿技术研究项目(152300410191).
责任编委: 阳春华.
†通讯作者. E-mail: iebzqiu@zzu.edu.cn.

以DBSCAN为代表的基于密度的聚类算法将核心点作为聚类原型,寻找与聚类原型密度可达的对象,形成聚类. 它可以发现任意形状的聚类,对噪声具有很好的鲁棒性,但由于算法采用了固定大小的邻域计算密度,使得这一类算法的聚类结果对输入参数敏感,且不能有效处理高维和多密度数据集.

DPC、LPC、CDP等算法通过计算对象密度,选取决策图中密度峰值对象作为聚类原型;依据距峰值最小距离原则,将其余数据对象划分到相应聚类. 虽然算法结构简单,易于理解,但DPC、CDP算法的密度度量方式依赖于手动输入的截断距离参数,不合理的参数设置会引起对象划分错误的连锁反应. LPC算法借鉴谱聚类思想,将数据集视为无向图,采用拉普拉斯中心性[14]表征数据对象的密度,由于LPC算法在提取每一维度的特征值形成拉普拉斯矩阵的计算量较大,对于高维数据集,算法运行时间会指数倍增加,无法适应高维数据聚类要求.

为了解决上述问题,本文以提出的偏差因子建立非核心对象过滤模型,用于过滤掉那些影响聚类原型选取的噪声和聚类边界对象;然后基于提出的原型选取机制自动确定聚类原型;最后,将其余对象分配到各个原型所属的簇中,形成聚类. 本文的创新点如下: 1) 建立非核心对象过滤模型; 2) 提出一种局部密度计算方法; 3) 提出一种聚类原型自动选取的机制.

1 基于过滤模型的聚类算法

1.1 相关定义

数据采样可分为静态采样和动态采样. 超球采样、立方采样和网格采样[6]属于静态采样,采样思想是以固定大小的邻域或超立方体中包含的数据对象个数来衡量一个对象的密度[15]. 若对象分布不均或对于分布稀疏的高维空间,难以设置合适的邻域半径或边长,造成对象的局部分布特征度量不准确,导致密度度量失衡.

相对于静态采样,动态采样可以更好地反映对象的分布特征,k近邻采样是动态采样,它提取k个最近的数据对象形成动态采样空间,可以更好地表征数据的局部密度分布. 设数据集D含有m个属性,k近邻的定义如下.

定义1 设$x \in D$, x的k近邻[6]是距离x最近的k个对象的集合,用$\mathrm{knn}(x)$表示,即

$$\begin{aligned}\mathrm{knn}(x) = \{&x_1, x_2, \ldots, x_k | x_i \in D,\ 1 \leqslant i \leqslant k,\\ &\forall y \in D - \{x_1, x_2, \ldots, x_k\},\\ &\mathrm{dist}(x, y) \geqslant \mathrm{dist}(x_i, x)\}.\end{aligned} \tag{1}$$

定义2 设$x \in D$, x的k近邻距离[6]$\mathrm{knn_}d(x)$定义为x到k近邻对象的距离之和,即

$$\mathrm{knn\_}d(x) = \sum_{y \in \mathrm{knn}(x)} \mathrm{dist}(x, y). \tag{2}$$

其中: dist为欧氏距离, $\mathrm{knn_}d(x)$反映了x周围的分布情况,其值越小,说明x周围分布越稠密,反之越稀疏,但对象的k近邻距离对参数k较为敏感,若k值选取过小,则采样不充分,无法表征对象的真实分布; k值选取过大,任意两个对象的k近邻距离几乎一致. 为了合理地度量对象的周围分布,本文提出k近邻共享度的定义.

定义3 设$x \in D$,对象x的k近邻共享度是对象x的k近邻与每个近邻对象的k近邻拥有相同对象个数之和,记为$\mathrm{knns}(x)$,即

$$\mathrm{knns}(x) = \sum_{y \in \mathrm{knn}(x)} |\mathrm{knn}(x) \bigcap \mathrm{knn}(y)|, \tag{3}$$

其中$\mathrm{knns}(x)$的取值范围为$[0, (k-1)^2]$. $\mathrm{knns}(x)$值越大,说明x周围数据对象分布越稠密; $\mathrm{knns}(n)$值越小,说明x的近邻与x的分布差异越大.

核心对象周围分布较为稠密,其k近邻距离较小, k近邻共享度较大;而非核心对象周围分布较为稀疏,其k近邻共享度较小. 为了增大核心对象与非核心对象的差异,结合k近邻距离与k近邻共享度,本文给出了局部密度的定义.

定义4 设$x \in D$,对象x的局部密度$\mathrm{den}(x)$定义如下:

$$\mathrm{den}(x) = \frac{\mathrm{knns}(x)}{\mathrm{knn\_}d(x)}. \tag{4}$$

DPC算法认为聚类中心具有较高的密度且距高密度对象的最小距离较远[11]. 为了方便聚类中心的选取,这里使用密度因子的概念放大聚类中心与其他对象的特征差异.

定义5 设$x \in D$,数据对象x的密度因子$R(x)$定义为

$$R(x) = \mathrm{den}(x)\delta(x). \tag{5}$$

其中: $\delta(x)$表示x与高密度数据对象之间的最小距离,即

$$\delta(x) = \begin{cases}\min\{\mathrm{dist}(x, y) | \mathrm{den}(x) < \mathrm{den}(y), y \in D\},\\ \qquad \exists y \in D,\ \mathrm{den}(x) < \mathrm{den}(y);\\ \max\{\mathrm{dist}(x, y) | y \in D\},\ \text{otherwise}.\end{cases} \tag{6}$$

一个对象的密度因子越大,表明对象成为聚类中心的可能性越大. 对象按其密度因子降序排序后,聚类中心一定位于序列的前半部分.

1.2 核心对象的获取

在基于密度的聚类中,数据对象通常划分为核心对象与非核心对象[16],核心对象构成聚类的骨架,而非核心对象 (如噪音、边界) 可能会干扰聚类过程[17]. 为此,首先建立非核心对象过滤模型(FM)去除边界和噪声对象,得到聚类的核心对象集,然后在核心对象集上获得聚类原型.

定义6 设$x \in D, y \in \mathrm{knn}(x)$,对象$x$的knn质心knn_centroid$(x)$定义为

$$\begin{aligned}&\mathrm{knn\_centroid}(x) = \\ &\Big(\sum_{y\in\mathrm{knn}(x)}\frac{y_1}{k}, \sum_{y\in\mathrm{knn}(x)}\frac{y_2}{k}, \ldots, \\ &\sum_{y\in\mathrm{knn}(x)}\frac{y_i}{k}, \ldots, \sum_{y\in\mathrm{knn}(x)}\frac{y_m}{k}\Big),\end{aligned} \tag{7}$$

其中y_i表示对象y在第i维上的取值,即$y = (y_1, y_2, \ldots, y_i, \ldots, y_m)$.

对象x的knn质心反映了x周围分布情况. 如果x是核心对象,则其k近邻相对均匀地分布在x的周围;若x是非核心对象,则其k近邻分布具有较大的偏向性,其k近邻聚集在某一方向. 为了反映这一特征,本文将对象x与knn质心之间的距离定义为x的偏差因子,依据偏差因子过滤非核心对象.

定义7 设$x \in D$, x的偏差因子$d_f(x)$定义为x到其k近邻质心的距离,即

$$d_f(x) = \mathrm{dist}(\mathrm{knn\_centroid}(x), x). \tag{8}$$

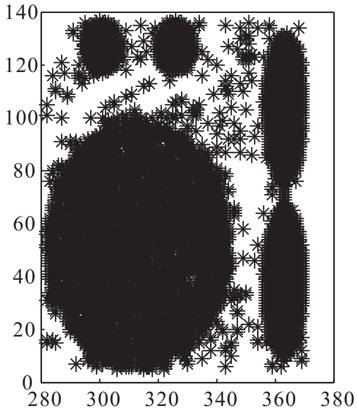
核心对象的周围分布较为均匀,其偏差因子较小;非核心对象周围分布具有较强的偏向性且稀疏,其偏差因子较大. 将数据对象按偏差因子降序排列,得到序列d_f_desc,则聚类的非核心对象和核心对象分别位于该序列的前半部分和后半部分.

设α为过滤因子(偏差因子降序排列后的百分位数). 对于数据集D,过滤模型FM将其分为非核心对象集non_core_set与核心对象集core_set,定义如下:

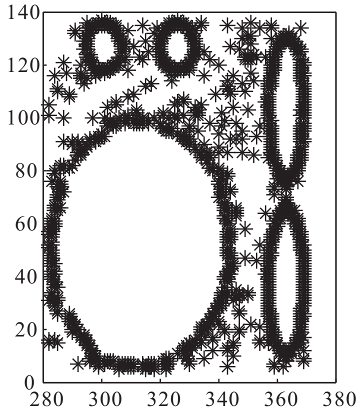
$$\begin{cases}\mathrm{core\_set} = \\ \{x|x\in D,\ \mathrm{and}\ d_f(x)\leqslant d_f\mathrm{\_desc}(\lceil|D|*\alpha\rceil)\}; \\ \mathrm{non\_core\_set} = \\ \{x|x\in D\ \mathrm{and}\ d_f(x) > d_f\mathrm{\_desc}(\lceil|D|*\alpha\rceil)\}.\end{cases} \tag{9}$$

其中: d_{f_desc}为偏差因子的降序排序,⌈ ⌉表示向上取整.

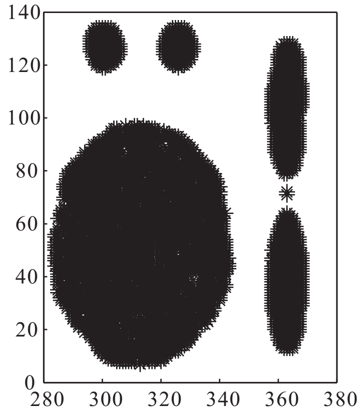
以合成数据集Syn为例,该数据集共有5 324个对象,含有5个大小不同的簇,簇周围存在噪声,垂直方向的两个椭圆簇之间存在桥接现象. 图1显示了聚类核心对象的获取过程,图1(b)表示使用过滤模型得到的边界和噪声对象. 图1(c)表示过滤后得到的核心对象.



(a) Syn数据集



(b) 非核心对象



(c) 核心对象

图1 核心点获取过程($\alpha = 19.84$)

1.3 聚类原型选取

自动且准确地获取聚类原型将提高聚类算法的适用性与效率, Chen等[18]采用证据积累的思想,迭代运行FCM算法,形成累积邻接矩阵,通过图切分的方式获取原型个数,由于FCM算法将最小误差平方和作为目标函数,导致部分隶属度矩阵存在偏差,使获取的原型个数并不稳定. Zhang等[19]将不同k值对应的最小误差平方和作为决策图,将决策图中拐点所对应的k值作为原型个数,对于非球形簇,此算法得到的聚类原型与实际原型存在一定的偏差. DC-MDACC[20]算法利用残差分析与线性回归,将未在置信区间中的对象视为聚类原型,但原型个数极易受到置信因子参数的影响. DPC与LPC算法在决策图中采用人工方式筛选聚类原型,即算法在选取过程中需要人工参与.

由于过滤了干扰聚类过程的非核心对象,核心对

象集core_set可以较好地表征数据集的分布特征. 本文利用非核心对象过滤模型获取核心对象集; 建立核心对象之间的近邻关系, 形成邻接矩阵; 采用遍历算法计算子图个数, 并结合密度因子得到聚类原型.

定义8 设x、$y \in$ core_set, 邻接矩阵Conn_M中元素定义如下:

$$\text{Conn\_}M(x,y)=\begin{cases}1,\ x\in \text{knn}(y);\\ 0,\ x\notin \text{knn}(y).\end{cases} \tag{10}$$

其中Conn_M为一个非对称的$n\times n$矩阵. 通过建立近邻关系, 将数据集的核心部分转换为独立的连通子图, 每个子图代表原始数据集中的一个聚类, 形成邻接矩阵Conn_M, 然后采用广度优先搜索的方式, 得到邻接矩阵中子图的个数I, 即为聚类原型个数.

由式(5)可知, 密度因子R越大的数据对象, 作为聚类中心的可能性越大. 依据密度因子进行降序排序后得到序列R_desc, 聚类原型一定在该序列的前半部分, 则序列R_desc的前I个数据对象即为聚类原型, 即Center_set $= R$_desc$[1:I]$.

1.4 基于过滤模型的聚类算法

基于过滤模型的聚类(clustering algorithm based on filter model, CA-FM)算法包含计算、核心点获取、原型获取和分派4个步骤. 计算步骤包含计算每个对象的局部密度、偏差因子、α距离、密度因子; 核心点获取步骤是依据过滤模型, 过滤那些影响聚类原型提取的非核心对象, 从而得到核心对象集; 原型获取步骤是根据核心集中对象的近邻关系构建邻接矩阵, 采用图搜索方式计算子图个数, 并使用密度因子获取聚类原型; 分派步骤将剩余数据对象划分至相应的簇中形成最终聚类. 详细步骤如下.

算法1 CA-FM算法.

输入: 数据集data、近邻参数k、过滤因子α;

输出: 数据集的聚类标签Label.

step 1: 计算. 根据式(4)、(5)、(6)、(8)计算数据对象的局部密度den、密度因子R、距离δ、偏差因子d_f.

step 2: 核心对象获取.

step 2.1: 依据偏差因子d_f将对象降序排序, 得到d_f_desc序列;

step 2.2: 选取d_f_desc集合中位置$\lceil |D|*\alpha \rceil$之后的对象作为核心对象, 形成核心对象集core_set.

step 3: 原型获取.

step 3.1: 根据式(10), 确定核心对象之间的近邻关系, 构建邻接矩阵Conn_M;

step 3.2: 采用遍历算法搜索邻接矩阵, 计算邻接矩阵中连通子图个数, 得到聚类中心个数I;

step 3.3: 选取密度因子降序排序后前I个对象作为聚类中心.

step 4: 分派. 将剩余数据对象划分至距离最近的高密度对象所在的簇中, 返回聚类标签Label.

2 实验结果与分析

实验环境: 内存为4.00 GB, 操作系统为Microsoft Windows7, 编译环境为MatlabR2014a.

数据集包括合成数据集、UCI数据集和人脸识别数据集[11]. 详细信息见表1, 编号1～编号6为合成数据集, 用来检验算法在不同数据分布形态下的聚类效果; 编号7～编号16是UCI数据集, 用来检测算法在真实数据下的聚类效果; 编号17为人脸识别数据集, 用来检测算法在高维数据下的检测效果. 算法使用标准化互信息(normalized mutual information, NMI)[21]、准确率(accuracy, ACC)[21]、纯度(Purify)[22]、兰德指数(rand index, RI)[23]、FM指数(fowlkes and mallows index, FMI)[24]、杰卡德相似系数(jaccard similarity coefficient, JC)[24]聚类评价指标从多角度衡量聚类质量.

表1 数据集的基本信息

NO.	datasets	data Sources	m	class	instance
1	Compound[25]	synthesis	2	6	399
2	R15[25]	synthesis	2	15	600
3	Spiral[25]	synthesis	2	3	312
4	Aggregation[25]	synthesis	2	7	788
5	Jain[25]	synthesis	2	2	373
6	4k2-far[25]	synthesis	2	4	400
7	Wine[26]	UCI	13	3	178
8	Sonar[27]	UCI	60	2	208
9	Soybean[27]	UCI	35	4	47
10	Zoo[27]	UCI	16	7	101
11	Parkinson[2]	UCI	22	2	195
12	Glass[26]	UCI	9	7	214
13	Iris[26]	UCI	4	3	150
14	Tic-Tac-Toe[26]	UCI	10	2	958
15	Mushroom[21]	UCI	22	2	8 124
16	German Credit[20]	UCI	20	2	1 000
17	Face dataset[11]	ORL	10 304	10	100

在对比实验中, 将K-means、FCM、CDP算法的参数: 原型个数设置为正确的聚类个数, 各运行10次, 取各聚类指标的均值作为最终的聚类效果, 其他算法则使用最优的聚类效果作为最终的结果.

2.1 人工合成数据集

实验选取的人工合成数据集包含了以下形态: 多密度、流型螺旋、多形状、微型簇、簇间半包含以及簇间嵌套, 并含有随机噪声或桥接噪声. Compound数据集共有6个密度分布不均匀的聚类, 且聚类之间

存在嵌套分布,用以测试算法能否准确识别多密度聚类和嵌套的聚类. R15 数据集包含 15 个微型簇,用来检测算法是否能完整识别数据集中所有聚类. Spiral、Jain 数据集为流型簇,簇之间为半包含关系,用来检测算法是否可以识别任意形状的聚类. Aggregation 数据集共有 7 个形状不同的聚类,簇之间存在桥接噪

表 2　人工合成数据集上的聚类结果比较

数据集	算法	参数	ACC/%	NMI	Purify	JC	RI	FMI
Compound	K-means	$k=6$	62.6566	0.7149	0.8321	0.4549	0.8406	0.6337
	DBSCAN	$\mathrm{MinPts}=4, \mathrm{EPS}=2.2$	78.4461	0.8053	0.7845	0.5552	**0.9341**	0.7324
	FCM	$k=6$	65.6642	0.7103	0.8321	0.4627	0.8427	0.6404
	CDP	$k=6, d_c=0.4$	65.9148	0.7407	0.8321	0.4808	0.8442	0.6535
	DPC	$d_c=1.25$	63.1579	0.7589	**0.9321**	0.4529	0.8266	0.6251
	LPC		77.4436	0.7679	0.7744	0.4968	0.9304	0.6700
	CA-FM	$k=10, \alpha=0.0275$	**83.2080**	**0.8594**	0.8321	**0.6015**	0.8977	**0.7748**
Spiral	K-means	$k=3$	34.6154	0.00005	0.3494	0.1960	0.5540	0.3278
	DBSCAN	$\mathrm{MinPts}=10, \mathrm{EPS}=1$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	FCM	$k=3$	33.9744	0.0002	0.3429	0.1955	0.5541	0.3272
	CDP	$k=3, d_c=0.24$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	DPC	$d_c=1.7443$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	LPC		**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	CA-FM	$k=2, \alpha=0.05$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Aggregation	K-means	$k=7$	73.3503	0.8036	0.8883	0.5676	0.8958	0.7321
	DBSCAN	$\mathrm{MinPts}=4, \mathrm{EPS}=0.83$	82.7411	0.8894	0.8274	**1.0000**	**1.0000**	**1.0000**
	FCM	$k=7$	79.6954	0.8427	0.9315	0.6433	0.9187	0.7926
	CDP	$k=7, d_c=0.03$	87.6904	0.8756	0.8858	0.7625	0.9374	0.8673
	DPC	$d_c=1$	94.0355	0.9705	0.9403	0.9591	0.9911	0.9793
	LPC		98.7310	0.9700	0.9873	0.9599	0.9912	0.9796
	CA-FM	$k=5, \alpha=0.3807$	**99.6193**	**0.9896**	**0.9962**	0.9898	0.9850	0.9949
R15	K-means	$k=15$	79.5000	0.8989	0.7950	0.6075	0.9606	0.7704
	DBSCAN	$\mathrm{MinPts}=5, \mathrm{EPS}=0.32$	78.1667	0.9121	0.7850	0.5927	0.9627	0.7642
	FCM	$k=15$	99.6667	0.9942	0.9967	0.9866	0.9991	0.9932
	CDP	$k=15, d_c=0.24$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	DPC	$d_c=0.95$	99.5000	0.9922	0.9950	0.9801	0.9987	0.9900
	LPC		92.1667	0.9410	0.9217	0.7915	0.9851	0.8846
	CA-FM	$k=10, \alpha=0.3333$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Jain	K-means	$k=2$	78.5523	0.3690	0.7855	0.5348	0.6621	0.7005
	DBSCAN	$\mathrm{MinPts}=4, \mathrm{EPS}=2.8$	73.9946	0.0001	0.7399	0.5000	0.4530	0.7071
	FCM	$k=2$	77.4799	0.3555	0.7748	0.5218	0.6501	0.6894
	CDP	$k=2, d_c=0.01$	86.0590	0.5052	0.8606	0.6497	0.7594	0.7904
	DPC	$d_c=0.95$	86.0590	0.5052	0.8606	0.6497	0.7594	0.7904
	LPC		89.2761	0.5709	0.8928	0.7135	0.8080	0.8348
	CA-FM	$k=9, \alpha=0.0268$	**92.2252**	**0.6447**	**0.9223**	**0.7804**	**0.8120**	**0.8779**
4k2-far	K-means	$k=4$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	DBSCAN	$\mathrm{MinPts}=4, \mathrm{EPS}=0.5$	99.7500	0.9941	**1.0000**	0.9940	0.9982	0.9970
	FCM	$k=4$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	CDP	$k=3, d_c=0.01$	87.7500	0.8430	0.8775	0.7287	0.9085	0.8441
	DPC	$d_c=0.2168$	83.2500	0.9081	**1.0000**	0.7749	0.9312	0.8803
	LPC		87.2500	0.8745	0.8725	**1.0000**	**1.0000**	**1.0000**
	CA-FM	$k=10, \alpha=0.15$	**100**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

声,用来检测算法能否处理带有桥接干扰的聚类. 4k2-far数据集含有4个簇和随机噪声,用来测试算法能否在噪声干扰下准确聚类.

表2给出了各算法的聚类结果评价指标值,表中加粗的数据是聚类指标最好的情况. FCM、K-means算法均采用最小误差平方和作为迭代的目标函数,因此二者均不能有效地处理非球形簇,如Spiral、Jain数据集.但FCM算法引入了模糊理论[26],其聚类效果好于K-means的聚类结果. DBSCAN算法虽然可以处理非球形簇,但DBSCAN算法采用固定邻域来度量密度,算法无法有效地处理多密度数据集,所以在Jain数据集上聚类效果较差.

DPC算法的聚类效果整体优于K-means、FCM、DBSCAN聚类算法,但DPC算法的中心选取过程依赖于数据对象在决策图中的分布,不同的截断参数会产生不同形状的决策图,可能会造成聚类中心与其他对象之间的特征差距变小,难以选取正确的聚类中心,最终影响聚类结果,如算法在4k2-far数据集上聚类效果不佳.

LPC算法聚类效果好于DPC算法,虽然LPC不需要显式的输入参数,但仍需要根据实际的聚类个数和决策图来选择聚类中心.

CDP算法依据距高密度点最小距离与距低密度点最小距离的差,将决策图中聚类中心与其他点进一步分离,便于选出聚类中心,但算法需要输入聚类个数,并且由于采用了截断距离d_c进行密度计算,导致其处理多密度数据的能力不强，如在Compound、Jain数据集上聚类效果不理想.

CA-FM算法在6个人工合成数据集中的4个均达到最佳聚类效果,在Compound数据集与Aggregation数据集的聚类效果大部分达到最佳,其余的聚类指标取值与最佳的聚类效果相差很小,说明CA-FM算法在处理多种数据分布的聚类时是有效的.

2.2 UCI数据集

实验选用了来自医疗、生物工程、地质勘探、化学等领域的UCI数据集,用来检验在真实、高维数据下的聚类效果. 其中Iris、Soybean、Zoo、Mushroom数据集来自生物工程领域,包含了不同动植物的各种特征; Parkinson数据集来自医疗领域,记录了病患的多种生理指标;Glass数据集来自化学领域,记录了玻璃的不同化学成分;Sonar数据集来自地质勘探领域,包含了不同物体的声呐强度;Tic-Tac-Toe数据集来自游戏博弈领域;German Credit来自金融信贷领域,记录了用户的信用情况. 表3给出了各算法在UCI数据集上的聚类结果评价指标值.

表 3 UCI数据集上的聚类结果比较

数据集	算法	参数	ACC/%	NMI	Purify	JC	RI	FMI
Parkinson	K-means	$k=2$	72.3077	0.0000	**0.7538**	0.5792	0.5975	0.7444
	DBSCAN	$\mathrm{MinPts}=5, \mathrm{EPS}=10$	50.2564	0.0904	0.7446	0.3320	0.4779	0.5093
	FCM	$k=2$	71.7949	**0.1037**	**0.7538**	0.4810	0.5929	0.6516
	CDP	$k=2, d_c=0.34$	65.6410	0.0140	**0.7538**	0.4552	0.5466	0.6260
	DPC	$d_c=9.8975$	73.3333	0.0533	**0.7538**	0.5321	0.6069	0.6948
	LPC		74.8718	0.0049	**0.7538**	0.6208	0.6218	0.7861
	CA-FM	$k=15, \alpha=0.1026$	**75.3846**	0.0001	**0.7538**	**0.7000**	**0.9404**	**0.7971**
Sonar	K-means	$k=2$	54.3264	0.0068	0.5433	0.3374	0.5013	0.5046
	DBSCAN	$\mathrm{MinPts}=1, \mathrm{EPS}=0.5$	35.0962	**0.2075**	0.4875	0.2922	0.5033	0.4546
	FCM	$k=2$	55.2885	0.0088	0.5529	0.3358	0.5032	0.5028
	CDP	$k=2, d_c=0.44$	50.0000	0.0517	0.5337	**0.4812**	0.4976	0.4818
	DPC	$d_c=0.8186$	50.9615	0.0011	0.5337	0.3450	0.4978	0.5133
	LPC		52.4038	0.0000	0.5337	0.3469	0.4987	0.5154
	CA-FM	$k=8, \alpha=0.2403$	**58.1731**	0.0228	**0.5817**	0.3473	**0.5110**	**0.5156**
Soybean	K-means	$k=4$	48.9362	0.5293	0.5745	0.3472	0.7391	0.5167
	DBSCAN	$\mathrm{MinPts}=8, \mathrm{EPS}=4$	**78.7234**	**0.8377**	0.7872	**1.0000**	**1.0000**	**1.0000**
	FCM	$k=4$	72.3404	0.7158	0.7872	0.4888	0.8316	0.6568
	CDP	$k=4, d_c=0.34$	63.8293	0.6376	**0.8085**	0.3739	0.8002	0.5500
	DPC	$d_c=0.667$	65.9574	0.6656	0.6596	0.4286	0.7817	0.6020
	LPC		70.2128	0.7786	0.8936	0.5213	0.8649	0.6947
	CA-FM	$k=5, \alpha=0.4255$	**78.7234**	**0.8377**	0.7872	**1.0000**	**1.0000**	**1.0000**

表3 (续)

数据集	算法	参数	ACC/%	NMI	Purify	JC	RI	FMI
Zoo	K-means	$k=4$	74.4681	0.7590	0.7872	0.5644	0.8279	0.7347
	DBSCAN	$\mathrm{MinPts}=4, \mathrm{EPS}=2$	72.2772	0.6018	0.7228	**0.7748**	0.9438	**0.8733**
	FCM	$k=4$	65.3465	0.7154	0.8218	0.5072	0.8642	0.6779
	CDP	$k=4, d_c=0.34$	62.3762	0.7080	0.8218	0.4058	0.9815	0.5865
	DPC	$d_c=0.12$	77.2277	**0.8086**	**0.9109**	0.5780	0.8822	0.7338
	LPC		75.2475	0.7256	0.8317	0.5820	0.8596	0.7414
	CA-FM	$k=10, \alpha=0.3960$	**78.2178**	0.7370	0.7822	**0.7748**	**0.9938**	**0.8733**
Wine	K-means	$k=4$	58.4270	0.3804	0.7047	0.3449	0.7032	0.5160
	DBSCAN	$\mathrm{MinPts}=2, \mathrm{EPS}=1.3$	38.2022	0.0268	0.3989	0.4864	0.7324	0.6888
	FCM	$k=4$	65.7303	0.4073	0.6573	**0.6957**	**0.9034**	**0.8206**
	CDP	$k=4, d_c=0.14$	56.1768	0.3138	0.6067	0.3300	0.6212	0.4988
	DPC	$d_c=17.1472$	52.8089	0.3939	0.6461	0.3983	0.6104	0.5889
	LPC		66.8539	0.4017	0.6742	0.4739	0.7009	0.6554
	CA-FM	$k=9, \alpha=0.3370$	**70.7865**	**0.4193**	**0.7079**	0.4117	0.7190	0.5832
Tic-Tac-Toe	K-means	$k=2$	50.9395	0.0000	0.6534	0.3529	0.4997	0.5222
	DBSCAN	$\mathrm{MinPts}=4, \mathrm{EPS}=9$	64.5094	0.0000	0.6534	0.5285	0.5416	**0.7170**
	FCM	$k=2$	51.8789	0.0010	0.6534	0.3534	0.5002	0.5228
	CDP	$k=2, d_c=1.1$	64.5094	0.0052	0.6534	0.5285	0.5416	0.7071
	DPC	$d_c=10$	59.1858	0.0041	0.6534	0.4948	0.5164	0.6813
	LPC		57.9332	0.0552	0.0186	0.4675	0.5121	0.6486
	CA-FM	$k=16, \alpha=0.1002$	**70.0418**	**0.0694**	**0.7004**	**0.5808**	**0.5799**	0.6519
Iris	K-means	$k=3$	89.3333	0.7582	0.8933	0.6959	0.8797	0.8208
	DBSCAN	$\mathrm{MinPts}=10, \mathrm{EPS}=0.5$	68.6667	0.6044	0.6867	0.5375	0.7719	0.7054
	FCM	$k=3$	89.3333	0.7496	0.8933	0.6943	0.8797	0.8197
	CDP	$k=3, d_c=0.24$	72.0000	0.6357	0.7200	0.5493	0.7820	0.7145
	DPC	$d_c=0.3162$	90.6667	0.8057	0.9067	0.7248	0.8923	0.8407
	LPC		69.3333	0.7098	0.6933	0.5843	0.7777	0.7567
	CA-FM	$k=10, \alpha=0.3333$	**96.0000**	**0.870**5	**0.9600**	**0.8578**	**0.949**5	**0.9234**
Glass	K-means	$k=7$	50.0000	0.3621	0.5000	0.3393	0.5750	0.5521
	DBSCAN	$\mathrm{MinPts}=5, \mathrm{EPS}=0.7$	45.3271	0.3068	0.4953	0.3625	0.6793	0.5594
	FCM	$k=7$	49.5327	0.3369	**0.6308**	0.2340	**0.7266**	0.3854
	CDP	$k=7, d_c=0.29$	48.1308	0.3731	0.5794	0.2556	0.6844	0.4072
	DPC	$d_c=0.3789$	48.1308	0.3708	0.4953	0.3331	0.5437	0.5538
	LPC		48.1308	0.2870	0.4907	0.2607	0.6219	0.4217
	CA-FM	$k=6, \alpha=0.1401$	**53.2710**	**0.3839**	0.5327	**0.3847**	0.7001	**0.5821**
Mushroom	K-means	$k=2$	64.5987	0.0712	0.6460	0.4035	0.5426	0.5764
	DBSCAN	$\mathrm{MinPts}=20, \mathrm{EPS}=30$	64.5987	0.0712	0.6460	0.4035	0.5426	0.5764
	FCM	$k=2$	64.5987	0.0712	0.6460	0.4035	0.5426	0.5764
	CDP	$k=2, d_c=0.79$	64.5987	0.0712	0.6460	0.4035	0.5426	0.5764
	DPC	$d_c=1.93$	60.4628	0.0831	0.6460	0.3716	0.5328	0.5813
	LPC		64.3279	0.0683	0.6423	0.4020	0.5410	0.5748
	CA-FM	$k=100, \alpha=0.3845$	**68.7469**	**0.1104**	**0.6875**	**0.4148**	**0.5702**	**0.5868**
German Credit	K-means	$k=2$	67.1000	0.0120	**0.7000**	0.4906	0.5580	0.6618
	DBSCAN	$\mathrm{MinPts}=8, \mathrm{EPS}=15.9$	57.5000	0.0047	**0.7000**	0.4203	0.5145	0.5921
	FCM	$k=2$	67.0000	0.0129	**0.7000**	0.4865	0.5574	0.6576
	CDP	$k=2, d_c=1.59$	60.6000	0.0076	**0.7000**	0.4501	0.5220	0.5768
	DPC	$d_c=7.7000$	67.3000	0.0074	**0.7000**	0.5312	**0.5680**	0.7051
	LPC		68.4000	0.0015	**0.7000**	**0.5580**	0.5673	0.7068
	CA-FM	$k=15, \alpha=0.2$	**78.0000**	**0.2356**	**0.7000**	0.5000	0.5102	**0.7071**

真实数据集的维度普遍较高,其数据对象的分布并不严格符合统计规律,且高维空间中的数据集形状难以确定,并非传统的球形簇,因此适用于球形簇的FCM、K-means算法的聚类效果不佳,如K-means在Soybean上效果较差,FCM在Zoo上效果较差. 由于高维空间上数据分布稀疏,DBSCAN算法的半径参数难以确定,算法在Iris、Wine、German Credit数据集上效果较差.

由于采用了在决策图中人工选择聚类原型的方式,DPC、LPC、CDP算法在大部分数据集上均可有效地聚类. LPC算法采用拉普拉斯中心性作为密度度量,其度量原理依据统计分布规律,对于高维数据集,密度度量会产生偏差,进而影响聚类结果,如算法在Iris、Glass数据集上效果不佳. DPC、CDP算法由于截断距离d_c的设置不合理,导致部分对象在决策图中的位置重合或者过于接近,干扰了聚类中心的选取,如CDP算法在Zoo、German Credit数据集上效果一般,DPC算法在Wine、Mushroom数据集上效果一般. CA-FM算法在各个数据集上的聚类指标大部分达到最佳,其余指标均与最佳指标相差不大. 说明CA-FM算法在处理真实数据的聚类时是有效的.

2.3 高维数据集

本文使用人脸识别数据集检测算法在高维数据下的聚类效果. ORL人脸数据库(olivetti research laboratory)来自剑桥Olivetti实验室. 共有40个不同年龄、不同性别和不同种族的对象. 每个人有10幅图像,图像尺寸是92×112. 从人脸数据集中随机抽取10个人物的图像,构成测试数据集Face dataset,共计100张图片,其中人脸部分表情和细节均有变化,例如笑与不笑、眼睛睁着或闭着、戴或不戴眼镜等. 数据集预处理方式为:将每张图像的像素矩阵转化成一维矩阵,即每张图像的维度为10 304,形成高维数据集.

算法的聚类结果如图2所示,每一行代表识别的不同聚类,每个聚类中亮度最高的图像表示算法识别的聚类原型. 详细聚类评价指标值如表4所示,可以看出, CA-FM算法对高维数据聚类是有效的.

图2 算法在人脸识别数据集上的聚类结果

表4 人脸识别数据集上的聚类结果比较

数据集	算法	参数	ACC/%	NMI	Purify	JC	RI	FMI
Face dataset	K-means	$k=10$	65.0000	0.7983	0.6500	0.5049	0.9491	0.6822
	DBSCAN	$\mathrm{MinPts}=1, \mathrm{EPS}=6$	10.0000	0.0000	0.1000	0.5000	0.9909	0.7071
	FCM	$k=10$	41.0000	0.5122	0.4100	0.2060	0.7766	0.3866
	CDP	$k=10, d_c=0.34$	88.0000	0.9582	0.9000	0.7887	0.9766	0.8851
	DPC	$d_c=3.0288\mathrm{e}^3$	81.0000	0.9188	0.8100	0.8000	0.9800	0.8926
	LPC		36.0000	0.6178	0.4300	0.2445	0.7859	0.4498
	CA-FM	$k=5, \alpha=0$	**100.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**

2.4 算法分析

2.4.1 时间复杂度分析

CA-FM算法的计算开销主要有:计算偏差因子、计算局部密度、计算α距离、建立邻接矩阵、计算子图个数和对象划分.

算法采用KD树建立数据集的k近邻关系[2],则计算偏差因子的时间复杂度为$O(kn\log_2 n)$,计算数据对象距高密度点最小距离的时间复杂度为$O(n\log_2 n+n)$,计算局部密度的时间复杂度为$O(n\log_2 n+n)$,建立邻接矩阵的时间复杂度为$O(kn)$,计算子图个数即对邻接矩阵构成的有向图进行搜索,时间复杂度为$O(n+e)$,其中e为具有近邻

关系的边数,最大为nk. 在得到聚类中心时,算法只一次遍历便可完成对象划分,时间复杂度为$O(n)$. 算法的时间复杂度之和为$O(kn\log_2 n + 2n\log_2 n + 3n + 2kn)$,则算法的时间复杂度为$O(kn\log_2 n)$. 表5给出了CA-FM算法与对比算法的时间复杂度,其中Item表示算法迭代次数,t表示CDP算法中被迪杰斯特拉算法处理的对象个数,K表示数据集的真实聚类个数. 可以看出,CA-FM算法的时间复杂度优于DBSCAN、DPC、LPC和CDP算法.

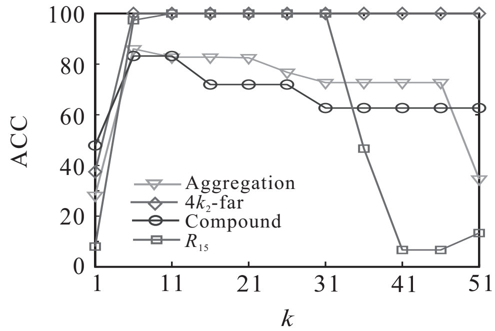
表5 算法的时间复杂度分析

算法	时间复杂度
K-means	$O(\text{Item} \times nK)$[13]
AP	$O(n^2 \log_2 n)$[13]
FCM	$O(\text{Item} \times nK)$[9]
DBSCAN	$O(n^2)$[10]
DP	$O(n^2)$[11]
LPC	$O(n^2)$[12]
CDP	$O(n^2 \log_2 n + tn^2 + n \log_2 K)$[13]
CA-FM	$O(kn\log_2 n)$

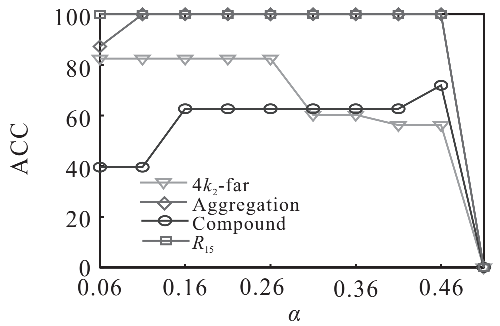
2.4.2 参数敏感性分析

CA-FM算法有两个参数:近邻对象数k、过滤因子α. 本文选取4个二维数据集与4个高维数据集对两个参数进行敏感性分析. 图3(a)和图3(c)给出了参数k与聚类精度之间的关系,可以看出,随着k增大,对象的采样空间不断扩展,局部密度可有效表征数据的真实分布情况,其聚类精度越来越高. 当k继续增大时,会导致数据对象的近邻关系延伸至其他簇中,造成多个簇合并从而导致聚类精度下降. 一般情况下,当k的取值为5～31时,算法能达到理想的聚类效果.

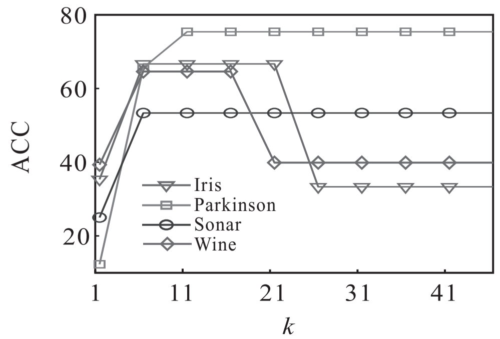
图3(b)和图3(d)给出了参数与聚类精度之间的关系. 参数α为过滤因子,其作用是过滤掉干扰聚类的对象. 对于二维数据集,当α取值偏小时,无法有效地去除全部的非核心对象,所以无法得到准确的聚类骨架,这时聚类精度偏低;对于高维数据集,其数据集的结构并不遵循传统的统计分布,α取值较小时不会对聚类精度产生影响,但随着参数α增大,部分核心对象被过滤掉,从而得不到完整的聚类骨架,造成聚类精度下降. 特殊情况下,当$\alpha = 1$时,表示全部数据对象均被视为非核心对象,聚类精度最差. 一般情况,α的取值为0.1～0.41时,算法能达到理想的聚类效果.



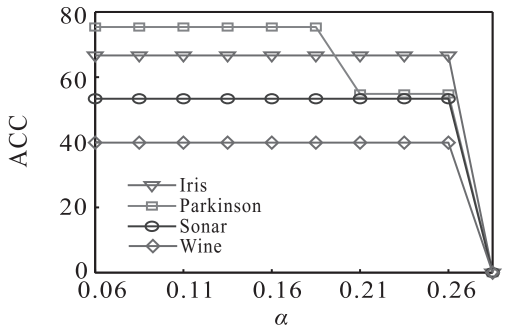
(a) 近邻参数k与精度的关系(二维数据集)



(b) 近过滤因子α与精度的关系(二维数据集)



(c) 近邻参数k与精度的关系(高维数据集)



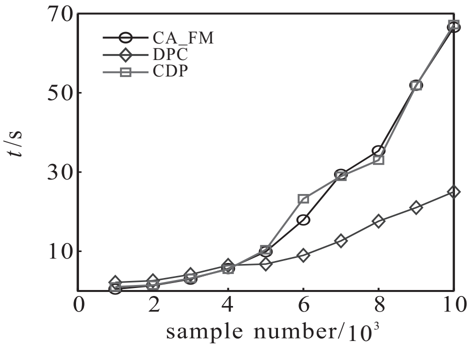
(d) 近过滤因子α与精度的关系(高维数据集)

图3 聚类精度与参数k、α的关系

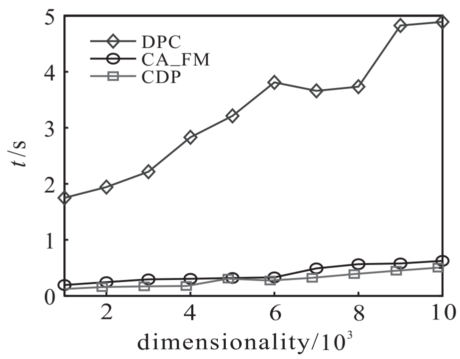
2.4.3 伸缩性分析

本文将合成数据集Flame[6]扩展为大样本量数据集,将UCI数据集Soybean扩充为高维度数据集,用来检验算法的运行效率,并与DPC、CDP算法进行比较. 由于LPC算法需要计算每一个维度的特征值,其时间消耗特别大,数据集维度1 000时,运行时间已经超过所显示范围,因此实验中没有给出与LPC算法的曲线. 图4(a)、图4(b)分别给出了样本量、维度与运

行时间的关系. 由图4可看出,CA-FM算法的运行时间介于DPC和CDP算法之间. 图4(a)中,为了在固定范围内得到直观的趋势对比,将DPC算法所需的复杂参数DATA提前进行预处理,因此,DPC算法的运行时间最少. 由图4(b)可知,CA-FM算法的时间消耗与CDP算法大致相当,而DPC算法由于需要人工选取决策图中的聚类原型,其运行时间与选取过程的耗时有关,因此曲线成不规则的上升趋势. 本算法的运行时间与样本量呈类正比函数关系,且对维度变化的敏感性不大,运行时间变化不大.



(a) 运行时间与样本量的关系



(b) 运行时间与维度的关系

图4 算法的运行时间与样本量、维度的关系

3 结 论

本文在定义偏差因子的基础上建立了非核心对象过滤模型,消除了噪声和边界点对原型提取的干扰;以提出的密度因子和邻接矩阵,解决了原型选取不合理的问题. CA-FM算法可以对含有噪声的多密度和高维数据集进行有效的聚类,并具有较高的精度,同时为研究聚类原型提供了一种框架.

参考文献(References)

[1] Kacprzyk J, Pedrycz W. Springer handbook of computational intelligence[M]. Berlin: Springer Publishing Company, 2015: 578-600.

[2] Li X L, Han Q, Qiu B Z. A clustering algorithm using skewness-based boundary detection[J]. Neurocomputing, 2018, 275: 618-626.

[3] Mondal S A. An improved approximation algorithm for hierarchical clustering[J]. Pattern Recognition Letters, 2018, 104: 23-28.

[4] Chen H Z, Wang W W, Feng X C, et al. Discriminative and coherent subspace clustering[J]. Neurocomputing, 2018, 284: 177-186.

[5] 李海林, 王成, 邓晓懿. 基于分量属性近邻传播的多元时间序列数据聚类方法[J]. 控制与决策, 2018, 33(4): 649-656.
(Li H L, Wang C, Deng X Y. Multivariate time series clustering based on affinity propagation of component attributes[J]. Control and Decision, 2018, 33(4): 649-656.)

[6] 李向丽, 曹晓锋, 邱保志. 基于矩阵模型的高维聚类边界模式发现[J]. 自动化学报, 2017, 43(11): 1962-1972.
(Li X L, Cao X F, Qiu B Z. Clustering boundary pattern discovery for high dimensional space base on matrix model[J]. Acta Automatica Sinica, 2017, 43(11): 1962-1972.)

[7] 张秦, 方志耕, 蔡佳佳, 等. 基于多元异构不确定性案例学习的广义区间灰数熵权聚类模型[J]. 控制与决策, 2018, 33(8): 1481-1488.
(Zhang Q, Fang Z G, Cai J J, et al. Generalized interval grey entropy-weight clustering model based on multiple heterogeneous uncertainty cases study[J]. Control and Decision, 2018, 33(8): 1481-1488.)

[8] Macqueen J. Some methods for classification and analysis of multivariate observations[C]. Proc of 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: California Presseedings, 1967: 281-297.

[9] Bezdek J C, Robert E, Full W. FCM: The fuzzy c-means clustering algorithm[J]. Computers & Geosciences, 1984, 10(2/3): 191-203.

[10] Ester M, Kriegel H P, Xu X W, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[J]. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96). Portland: Association for the Advan- cement of Artificial Intelligence, 1996: 226-231.

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

[12] Yang X H, Zhu Q P, Huang Y J, et al. Parameter-free laplacian centrality peaks clustering[J]. Pattern Recognition Letters, 2017, 100: 167-173.

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

[14] Qi X, Fuller E, Wu Q, et al. Laplacian centrality: A new centrality measure for weighted networks[J]. Information Sciences, 2012, 194: 240-253.

[15] Kumar K M, Reddy A R M. A fast dbscan clustering algorithm by accelerating neighbor searching using groups method[J]. Pattern Recognition, 2016, 58: 39-48.

[16] Li X L, Han Q, Qiu B Z. A clustering algorithm with affine space-based boundary detection[J]. Applied Intelligence, 2018, 48(2): 432-444.

[17] Qiu B Z, Cao X F. Clustering boundary detection for high dimensional space based on space inversion and Hopkins statistics[J]. Knowledge-Based Systems, 2016, 98: 216-225.

[18] Chen H P, Shen X J, Lv Y D, et al. A novel automatic fuzzy clustering algorithm based on soft partition and membership information[J]. Neurocomputing, 2017, 236: 104-112.

[19] Zhang Y, Madziuk J, Chai H Q, et al. Curvature-based method for determining the number of clusters[J]. Information Sciences, 2017, 415: 414-428.

[20] 陈晋音, 何辉豪. 基于密度的聚类中心自动确定的混合属性数据聚类算法研究[J]. 自动化学报, 2015, 41(10): 1798-1813.
(Chen J Y, He H H. Research on density-based clustering algorithm for mixed data with determine cluster centers auto matically[J].Acta Automatica Sinica, 2015, 41(10): 1798-1813.)

[21] Ding S F, Du M J, Sun T F, et al. An entropy-based density peaks clustering algorithm for mixed type data employing fuzzy neighborhood[J]. Knowledge-Based Systems, 2017, 133: 294-313.

[22] Aliguliyev R M. Performance evaluation of density-based clustering methods[J]. Information Sciences, 2009, 179(20): 3583-3602.

[23] 乔颖, 王士同, 杭文龙. 大规模数据集引力同步聚类[J]. 控制与决策, 2017, 32(6): 1075-1083.
(Qiao Y, Wang S T, Hang W L. Clustering by gravitational synchronization on large scale dataset[J]. Control and Decision, 2017, 32(6): 1075-1083.)

[24] 徐明亮, 王士同, 杭文龙. 一种基于同类约束的半监督近邻反射传播聚类方法[J]. 自动化学报, 2016, 42(2): 255-269.
(Xu M L, Wang S T, Hang W L. A semi-supervised affinity propagation clustering method with homogeneity constraint[J]. Acta Automatica Sinica, 2016, 42(2): 255-269.)

[25] Chen M, Li L J, Wang B, et al. Effectively clustering by finding density backbone based-on knn[J]. Pattern Recognition, 2016, 60: 486-498.

[26] Campo D N, Stegmayer G, Milone D H. A new index for clustering validation with overlapped clusters[J]. Expert Systems with Applications, 2016, 64: 549-556.

[27] Du M, Ding S, Xue Y. A novel density peaks clustering algorithm for mixed data[J]. Pattern Recognition Letters, 2017, 97: 46-53.

作者简介

邱保志 (1962−), 男, 教授, 博士, 从事机器学习与数据挖掘等研究, E-mail: iebzqiu@zzu.edu.cn;

张瑞霖 (1995−), 男, 硕士生, 从事机器学习与数据挖掘的研究, E-mail: zzurlz@163.com;

李向丽 (1965−), 女, 教授, 博士, 从事计算机网络等研究, E-mail: iexlli@zzu.edu.cn.

(责任编辑：孙艺红)