文章编号:1671-6833(2024)03-0080-09

融合异常检测与区域分割的高效 *K*-means 聚类算法

尹宏伟[1,2], 杭雨晴[1,2], 胡文军[1,2]

(1. 湖州师范学院 信息工程学院,浙江 湖州 313000;2. 湖州师范学院 浙江省现代农业资源智慧管理与应用研究重点实验室,浙江 湖州 313000)

摘 要: 传统 *K*-means 及其众多改进算法缺乏显式处理异常样本的能力,导致其聚类性能容易受到异常样本的影响。针对此问题,提出一种融合异常检测与区域分割的高效 *K*-means 聚类算法。首先,通过构建统一聚类模型,形成异常检测与聚类之间的交互协同,以提高聚类性能。其次,利用近邻簇搜索技术对各类簇进行自适应的区域分割,以减少冗余计算,提高算法执行效率。最后,为验证所提方法的有效性,在多个合成数据集和真实数据集上分别进行测试。实验结果表明:所提算法聚类性能和执行效率优于其他算法;在添加 10%异常样本的 Wine 数据集上准确度可达 0.911。

关键词:聚类; *K*-means; 异常检测; 区域分割; 近邻簇搜索; 自适应

中图分类号: TP391 **文献标志码**:A **doi**:10.13705/j.issn.1671-6833.2024.03.010

作为机器学习与数据挖掘的主要技术之一,聚类分析可在无监督条件下,通过学习数据内在分布结构,将具有较高相似度的样本归为相同类簇,反之将相异的样本分为不同类簇。目前,主要的聚类分析方法包括以下五类:基于划分的 *K*-means 和谱聚类算法[1-2]、密度聚类算法[3]、网格聚类算法[4]、层次聚类算法[5]以及基于模型的聚类算法[6]等。其中,*K*-means 算法设定每个类簇由唯一的类簇中心进行标识,将聚类问题形式化为样本点与类簇中心的距离最小化问题,通过样本点分配与类簇中心更新两个阶段的交替迭代,从而实现对样本集合的划分。在样本点分配阶段,根据所有样本点到类簇中心的距离计算结果,判定样本点所属类簇;在类簇中心更新阶段,将该类中所有样本点的平均值设定为新的类簇中心。虽然 *K*-means 算法被广泛应用于多种任务场景[7-8],但其算法效果容易受到异常样本的干扰,并且计算成本与样本数量线性相关。

近年来,针对 *K*-means 算法容易受到异常样本干扰的问题,引入了多种异常检测机制对算法进行改进。例如,通过引入样本局部密度的度量机制在数据预处理时实现异常检测[9]、基于混合概率模型的异常检测方法[10]、基于采样策略的异常检测方法[11]以及通过类簇中心的迭代交换来计算类簇的紧凑程度,从而判断各样本点是否属于异常[12]。虽然引入异常检测机制提高了聚类性能,但是这些方法均采用分阶段实现的方式,没有形成融合异常检测和聚类的统一模型,无法获取良好的聚类结果。此外,异常检测机制极大地增加了算法的计算成本,降低了聚类过程的执行效率,限制了算法在大规模数据集上的应用。

针对以上问题,本文提出一种融合异常检测与区域分割的高效 *K*-means 聚类算法 (efficient *K*-means with region segment and outlier detection, E*K*-means)。与目前 *K*-means 及其各种改进型聚类算法相比,建立了融合异常检测的统一聚类模型。E*K*-means 能够在聚类过程中捕获异常样本,同时在消除异常样本后进行聚类,通过两个阶段的交互协同提升了聚类性能。另外,为解决由于异常检测导致的计算效率下降问题,采用 E*K*-means 对类簇进行自适应的区域分割,在保持算法聚类性能不受影响的前提下,

收稿日期:2023-10-20;**修订日期**:2023-11-24

基金项目:国家自然科学基金资助项目(62206094);湖州市公益性应用研究项目(2021GZ05);江苏省网络空间安全工程实验室开放课题(SDGC2237)

作者简介:尹宏伟(1990—),男,安徽安庆人,湖州师范学院副教授,博士,主要从事机器学习、模式识别及数据挖掘等方面的研究,E-mail: 02713@zjhu.edu.com。

通信作者:胡文军(1977—),男,安徽宣城人,湖州师范学院教授,博士,主要从事机器学习、模式识别及智能系统等方面的研究, E-mail: hoowenjun@foxmail.com。

引用本文:尹宏伟,杭雨晴,胡文军. 融合异常检测与区域分割的高效 *K*-means 聚类算法[J]. 郑州大学学报(工学版),2024,45(3):80-88. (YIN H W, HANG Y Q, HU W J. Efficient *K*-means with region segment and outlier detection [J]. Journal of Zhengzhou University (Engineering Science),2024,45(3):80-88.)

减少各区域中的冗余计算,以提升算法的效率。

1　相关工作

1.1　鲁棒 K-means 算法

由于传统 K-means 算法的类簇中心容易受到异常样本影响而无法准确获取,进而导致聚类算法性能的稳定性和准确性不足。近年来,针对此问题的研究工作主要分为两类:①利用矩阵范数建立正则化约束,削弱异常样本对聚类过程的影响;②引入异常检测机制,建立分阶段的数据聚类过程。

Huang 等[13]构建了基于范数的损失函数,减少异常样本的影响。Huang 等[14]提出了基于稀疏诱导范数的损失函数,增强了模型的鲁棒性。虽然以上方法能够有效提高聚类算法的性能,但无法显式获取异常样本,无法完全消除其影响。

与以上方法不同,Hautamäki 等[15]提出采用分阶段异常处理机制,在聚类算法的迭代过程中逐渐删除远离类簇中心的异常样本。Im 等[9]提出采用局部密度对样本进行异常检测。Li 等[10]引入混合概率模型在聚类过程中检测异常样本。Zhang 等[12]建立了基于局部搜索技术的 K-means 聚类算法,通过在目标函数中增加惩罚项以实现异常检测。上述算法分阶段执行聚类与异常检测,忽视了两者间的相关性,无法实现良好聚类,并且由于异常检测增加了大量计算,使得聚类算法的效率受到影响,限制了其在复杂大规模数据集上的应用效能。

1.2　快速 K-means 算法

标准 K-means 算法的时间复杂度为 $O(nkt)$,与样本集规模线性相关。近年来,为提高 K-means 算法在大规模数据集上的执行效率,主要的研究方向包括:①近似 K-means 算法,通过各类近似算法获取 K-means 的聚类结果,具有部分精度的损失;②精确 K-means 算法,通过减少迭代计算的次数,实现算法加速,同时保持聚类精度不变。

Peng 等[16]提出一种新型近邻关系,比较样本与最近邻之间的距离,缩小样本计算范围。Giffon 等[17]将稀疏因子乘积作为类簇中心矩阵的近似值,通过降低数据维度减少计算复杂度。

Hamerly[18]根据样本与类簇中心的距离设置其上下边界,并将三角不等式应用于边界,减少冗余距离计算。作为 Hamerly 算法的扩展,Drake[19]提出的 Ann (annular K-means) 算法将三角不等式应用于以样本为圆心的环形结构。Newling 等[20]提出 Exp-ns(exponion K-means) 算法将三角不等式应用于以类簇中心为球心的球形结构,以定位样本标签的变动范围。由于上述限制上下界方法操作复杂且加速能力有限,Xia 等[21]首次提出对类簇进行分割,减少不同区域的冗余计算。

在以上基于 K-means 的各类改进算法中,缺少同时考虑异常检测与计算效率的方法。因此,本文针对以上问题提出了融合异常检测与区域分割的高效 K-means 聚类算法。一方面,通过改进传统 K-means 的目标函数,将异常检测融合于聚类过程,实现统一模型的构建;另一方面,引入近邻簇搜索技术[21],实现各类簇自适应的区域分割,以减少样本聚类过程中的冗余计算。

2　E*K*-means

2.1　目标函数

给定数据集 $X=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n\}\in\mathbf{R}^{n\times d}$,其中 n 为样本数量,d 为特征维度。不失一般性,可设定该数据集中包含 z 个异常样本,则该数据集的异常样本集合为 $Z\in\mathbf{R}^{z\times d}$。算法将数据集划分为不同的类簇,最终聚类结果表示为 $\{\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_k,Z\}$。为了实现异常检测与聚类过程的交互协同,建立统一聚类模型,其目标函数为

$$\begin{cases}\min\limits_{\boldsymbol{C},\boldsymbol{W},\boldsymbol{O}}\sum\limits_{i=1}^{n}o_i\sum\limits_{j=1}^{k}w_{ij}\|\boldsymbol{x}_i-\boldsymbol{c}_j\|_2^2;\\ \text{s.t.}\ \sum\limits_{j=1}^{k}w_{ij}=1,w_{ij}\in\{0,1\};\\ \sum\limits_{i=1}^{n}o_i=n-z,o_i\in\{0,1\}。\end{cases}\tag{1}$$

式中:$\boldsymbol{x}_i$ 为第 i 个样本;$\boldsymbol{c}_j$ 为第 j 个类簇中心;w_{ij} 表示样本所在类簇;o_i 表示第 i 个样本是否属于异常样本。聚类指示矩阵 $\boldsymbol{W}\in\mathbf{R}^{n\times k}$ 由元素 w_{ij} 构成,当 $w_{ij}=1$ 时,第 i 样本被分配至第 j 类簇;当 $w_{ij}=0$ 时,第 i 样本不属于第 j 类簇。异常指示矩阵 $\boldsymbol{O}\in\mathbf{R}^{n\times 1}$ 由元素 o_i 构成,当 $o_i=0$ 时,第 i 样本为异常样本;反之为正常样本。

在标准 K-means 算法中,目标函数的迭代求解过程包括 2 个阶段,分为类簇中心更新阶段与样本分配阶段。根据本文提出的目标函数,本文算法的迭代过程包括 3 个阶段,分别为类簇中心更新阶段、样本分配阶段以及异常检测阶段。通过 3 个阶段的交替优化完成式(1)的求解,其中类簇中心矩阵 $\boldsymbol{C}$ 的求解与标准 K-means 算法一致。不同于标准 K-means 算法求解过程,在样本分配阶段中,引入了区域分割技术,提升聚类指示矩阵 $\boldsymbol{W}$ 的求解速度。在新增的异常检测阶段中,通过计算样本点的离群程

度求解异常指示矩阵 $\boldsymbol{O}$。

2.2 函数求解

如式(1)所示,目标函数中包含了类簇中心矩阵 $\boldsymbol{C}$、聚类指示矩阵 $\boldsymbol{W}$ 以及异常指示矩阵 $\boldsymbol{O}$,采用交替优化策略进行求解,求解过程如下所示。

步骤 1 固定异常指示矩阵 $\boldsymbol{O}$ 及聚类指示矩阵 $\boldsymbol{W}$,求解聚类中心矩阵 $\boldsymbol{C}$。当矩阵 $\boldsymbol{O}$ 与矩阵 $\boldsymbol{W}$ 固定时,式(1)转化为如下问题:

$$\min_{\boldsymbol{C}} \sum_{i=1}^{n}\sum_{j=1}^{k} o_i w_{ij} \| \boldsymbol{x}_i - \boldsymbol{c}_j \|_2^2 。 \tag{2}$$

对式(2)中 $\boldsymbol{c}_j$ 求导可得

$$2\sum_{i=1}^{n}\sum_{j=1}^{k} o_i w_{ij}(\boldsymbol{x}_i - \boldsymbol{c}_j) 。 \tag{3}$$

令式(3)等于 0,获取聚类中心,如式(4)所示:

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{n} o_i w_{ij} \boldsymbol{x}_i}{\sum_{i=1}^{n} o_i w_{ij}} 。 \tag{4}$$

步骤 2 固定异常指示矩阵 $\boldsymbol{O}$ 及聚类中心矩阵 $\boldsymbol{C}$,求解聚类指示矩阵 $\boldsymbol{W}$。当矩阵 $\boldsymbol{O}$ 及 $\boldsymbol{C}$ 固定时,令 $\boldsymbol{H} = \boldsymbol{O} \| \boldsymbol{X} - \boldsymbol{C} \|_2^2$,则目标函数可转变为

$$\begin{cases} \min\limits_{\boldsymbol{W}} \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k} w_{ij}\boldsymbol{H}; \\ \text{s.t.} \sum\limits_{j=1}^{k} w_{ij} = 1, w_{ij} \in \{0,1\}。 \end{cases} \tag{5}$$

显然,通过将样本点分配到最近类簇以更新 w_{ij},所以聚类指示矩阵 $\boldsymbol{W}$ 的求解为

$$w_{ij} = \begin{cases} 1, j = \underset{j \in \{1,2,\cdots,k\}}{\operatorname{argmin}} o_i \| \boldsymbol{x}_i - \boldsymbol{c}_j \|_2^2; \\ 0, \text{其他}。 \end{cases} \tag{6}$$

其中,由于异常指示矩阵将检测为异常样本的 o_i 值设定为 0,使得此类样本 w_{ij} 的取值不会对式(4)的求解产生影响。通过该方式,在此次迭代过程中被检测为异常的样本不参与分配,也不会影响正常样本点的分配。另一方面,为提高样本的分配速度,算法利用近邻簇搜索,对样本进行区域分割,以减少冗余距离计算,详细步骤见 2.3 节。

步骤 3 固定聚类指示矩阵 $\boldsymbol{W}$ 及聚类中心矩阵 $\boldsymbol{C}$,求解异常指示矩阵 $\boldsymbol{O}$。当矩阵 $\boldsymbol{W}$ 和矩阵 $\boldsymbol{C}$ 固定时,令 $\boldsymbol{M} = \boldsymbol{W} \| \boldsymbol{X} - \boldsymbol{C} \|_2^2$,目标函数可转化为

$$\begin{cases} \min\limits_{\boldsymbol{o}} \sum\limits_{i=1}^{n} o_i \boldsymbol{M}; \\ \text{s.t.} \sum\limits_{i=1}^{n} o_i = n - z, o_i \in \{0,1\}。 \end{cases} \tag{7}$$

显然,通过去除最大距离值的 z 个异常样本,可以最小化式(7),故异常指示矩阵 $\boldsymbol{O}$ 为

$$\begin{cases} o_i = \begin{cases} 0, \boldsymbol{x}_i \in Z; \\ 1, \text{其他}。 \end{cases} \\ \text{s.t. } Z = \max\limits_{\zeta_i \in X} \sum\limits_{i=1}^{z} w_{ij} \| \zeta_i - \boldsymbol{c}_j \|_2^2。 \end{cases} \tag{8}$$

式中:$Z \in \mathbf{R}^{z \times d}$ 为异常集合,记录异常样本序号。在求解 o_i 的过程中,$\boldsymbol{M}$ 描述了该样本的离群程度。通过样本离群程度判断异常样本,将异常样本添加至异常集合,并将其 o_i 设为 0。为减少异常检测计算量,仅对部分异常指示矩阵 $\boldsymbol{O}$ 进行更新。即若正常样本到所属类簇中心的距离值超过异常样本到最近类簇中心的距离值,更新两者的异常指示。

2.3 基于近邻簇搜索的区域分割

在样本分配阶段,为提高求解聚类指示矩阵 $\boldsymbol{W}$ 的计算效率,本文算法引入近邻簇搜索技术[21],对类簇进行区域分割,以减少冗余计算。首先定义各类簇之间的单向近邻关系,如定义 1 所示。

定义 1 单向近邻簇。对于任意类簇 $\boldsymbol{B}_u$ 和 $\boldsymbol{B}_v$,其类簇中心分别为 $\boldsymbol{c}_u$、$\boldsymbol{c}_v$,$\max\limits_{\boldsymbol{x}_i \in \boldsymbol{B}_u}(o_i \| \boldsymbol{x}_i - \boldsymbol{c} \|)$ 为 $\boldsymbol{B}_u$ 半径长度 r_u,当满足以下不等式条件,$\boldsymbol{B}_v$ 即为 $\boldsymbol{B}_u$ 的单向近邻簇:

$$(\boldsymbol{c}_u - \boldsymbol{c}_v)/2 < r_u。 \tag{9}$$

根据定义 1,可获取 $\boldsymbol{B}_u$ 满足条件的 m 个单向近邻簇,同时根据类簇中心间距离,由近及远进行排序,构建 $\boldsymbol{B}_u$ 的单向近邻簇序列,其中令 l 表示该序列的序号,则 $\boldsymbol{B}_u$ 的区域分割方式 $\Delta^{\{l\}}$ 为

$$\begin{cases} \| \boldsymbol{x} - \boldsymbol{c}_u \| \leqslant dis_l, l = 1; \\ dis_l < \| \boldsymbol{x} - \boldsymbol{c}_u \| \leqslant dis_{l+1}, 1 < l < m; \\ dis_l < \| \boldsymbol{x} - \boldsymbol{c}_u \| \leqslant r_u, l = m。 \\ \text{s.t. } dis_l = \dfrac{\| \boldsymbol{c}_u - \boldsymbol{c}_l \|}{2}。 \end{cases} \tag{10}$$

式中:当 $l = 1$ 时,$\Delta^{\{l\}}$ 为静态域;当 $l > 1$ 时,$\Delta^{\{l\}}$ 为多个动态域。如图 1 所示,黄色菱形表示类簇中心,红色虚线表示两类簇中心中垂线。以球簇 $\boldsymbol{B}_1$ 为例,根据式(9),$\boldsymbol{B}_2$ 和 $\boldsymbol{B}_4$ 为其单向近邻簇,而 $\boldsymbol{B}_3$ 不为其单向近邻簇。根据式(10),球簇 $\boldsymbol{B}_1$ 被分割为 1 个静态域及 2 个动态域。

在当前迭代中,静态域中样本点的聚类指示不会改变。而动态域中样本点的聚类指示可能改变,且其变动范围如定理 2 所述。因此,算法在求解聚类指示矩阵时,省略了不影响聚类结果的距离计算。

定理 1 对于 $\boldsymbol{B}_u$ 中任意样本点 $\boldsymbol{x}$,若满足 $\boldsymbol{x} \in \Delta^{\{1\}}$,则有 $\forall \| \boldsymbol{x} - \boldsymbol{c}_u \| \leqslant \| \boldsymbol{x} - \boldsymbol{c}_v \|$。

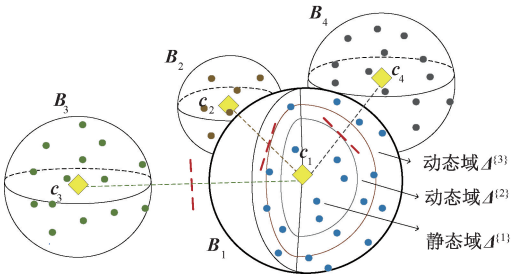


图 1　基于近邻簇搜索的区域分割

Figure 1　Region segment based on neighbor clusters search

证明:∵ $\forall \boldsymbol{x} \in \boldsymbol{B}_u$, $\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant r_u$, $\forall \boldsymbol{x} \in \Delta^{\{1\}}$, $r_u \leqslant \dfrac{\|\boldsymbol{c}_u-\boldsymbol{c}_v\|}{2}$

$$\therefore \|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant r_u \leqslant \frac{\|\boldsymbol{c}_u-\boldsymbol{c}_v\|}{2} = \frac{\|\boldsymbol{c}_u-\boldsymbol{x}+\boldsymbol{x}-\boldsymbol{c}_v\|}{2}$$

$$\text{且} \frac{\|\boldsymbol{c}_u-\boldsymbol{x}+\boldsymbol{x}-\boldsymbol{c}_v\|}{2} \leqslant \frac{(\|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{x}-\boldsymbol{c}_v\|)}{2} \Rightarrow$$

$$\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant \frac{(\|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{x}-\boldsymbol{c}_v\|)}{2} \Rightarrow$$

$$2\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant \|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{x}-\boldsymbol{c}_v\| \Rightarrow$$

$$\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant \|\boldsymbol{x}-\boldsymbol{c}_v\|$$

由定理 1 可知,若样本 $\boldsymbol{x} \in \Delta^{\{1\}}$,在本次迭代中,其最近类簇中心将维持不变,无须更新其聚类指示。

定理 2　对于 $\boldsymbol{B}_u$ 中任意样本点 $\boldsymbol{x}$,若满足 $\boldsymbol{x} \in \Delta^{\{l\}}$,且 $l>1$,则有

(1) $\forall \boldsymbol{x} \in \Delta^{\{l\}}$, $\|\boldsymbol{x}-\boldsymbol{c}_u\| < \|\boldsymbol{x}-\boldsymbol{c}_{l+1}\|$;

(2) $\exists \boldsymbol{x} \in \Delta^{\{l\}}$, $\|\boldsymbol{x}-\boldsymbol{c}_l\| < \|\boldsymbol{x}-\boldsymbol{c}_u\|$。

证:(1) $\forall \boldsymbol{x} \in \Delta^{\{l\}}$, $\|\boldsymbol{x}-\boldsymbol{c}_u\| < \dfrac{\|\boldsymbol{c}_u-\boldsymbol{c}_{l+1}\|}{2}$

$$\text{且} \frac{\|\boldsymbol{c}_u-\boldsymbol{c}_{l+1}\|}{2} \leqslant \frac{(\|\boldsymbol{c}_u-\boldsymbol{x}_u\|+\|\boldsymbol{x}-\boldsymbol{c}_{l+1}\|)}{2} \Rightarrow$$

$$\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant \frac{(\|\boldsymbol{c}_u-\boldsymbol{x}_u\|+\|\boldsymbol{x}-\boldsymbol{c}_{l+1}\|)}{2} \Rightarrow$$

$$2\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant \|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{x}-\boldsymbol{c}_{l+1}\| \Rightarrow$$

$$\|\boldsymbol{x}-\boldsymbol{c}_u\| \leqslant \|\boldsymbol{x}-\boldsymbol{c}_{l+1}\|$$

(2) $\forall \boldsymbol{x} \in \Delta^{\{l\}}$, $\|\boldsymbol{x}-\boldsymbol{c}_u\| > \dfrac{\|\boldsymbol{c}_u-\boldsymbol{c}_l\|}{2}$

当 $\|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{c}_l-\boldsymbol{x}\| = \|\boldsymbol{c}_u-\boldsymbol{c}_l\|$

$$\text{则} \frac{\|\boldsymbol{c}_u-\boldsymbol{c}_v\|}{2} = \frac{(\|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{x}-\boldsymbol{c}_v\|)}{2} < \|\boldsymbol{c}_u-\boldsymbol{x}\| \Rightarrow$$

$$(\|\boldsymbol{c}_u-\boldsymbol{x}\|+\|\boldsymbol{x}-\boldsymbol{c}_v\|) < 2\|\boldsymbol{c}_u-\boldsymbol{x}\| \Rightarrow$$

$$\|\boldsymbol{x}-\boldsymbol{c}_v\| < \|\boldsymbol{x}-\boldsymbol{c}_u\|$$

由定理 2 可知,若 $\boldsymbol{x} \in \Delta^{\{l\}}$,相较于排序在 l 之后的单向近邻簇,其与原类簇中心距离值更小。且存在部分样本最近类簇中心由原类簇中心变动为排序在 l 之前某一单向近邻簇。故若 $\boldsymbol{x} \in \Delta^{\{l\}}$,$\boldsymbol{x}$ 的变动范围为原分配类簇及前 i 个 $\boldsymbol{B}_u$ 的单向近邻簇。

2.4　算法分析

算法分为初始化阶段与迭代阶段。首先,利用 K-means++[22] 初始化类簇中心,并完成初次样本分配。其次,根据目标函数迭代更新类簇中心、聚类指示矩阵及异常指示矩阵。算法将异常检测融入聚类过程中,并对样本空间进行区域分割,同时考虑算法性能及效率。EK-means 算法在无异常样本时,所有的 o_i 为 1,即退化为标准 K-means。算法主要步骤如下。

步骤 1　输入数据集 X,类簇数 k,异常数 z;

步骤 2　执行 K-means++,初始化类簇中心矩阵 $\boldsymbol{C}$ 和聚类指示矩阵 $\boldsymbol{W}$;

步骤 3　根据式(8),初始化异常指示矩阵 $\boldsymbol{O}$;

步骤 4　根据式(4),更新类簇中心矩阵 $\boldsymbol{C}$;

步骤 5　根据式(10),更新各类簇区域分割 Δ;

步骤 6　根据式(6),更新聚类指示矩阵 $\boldsymbol{W}$;

步骤 7　根据式(8),更新异常指示矩阵 $\boldsymbol{O}$;

步骤 8　重复步骤 4~7,直到类簇中心不发生变化,输出聚类指示矩阵 $\boldsymbol{W}$、异常指示矩阵 $\boldsymbol{O}$。

计算 EK-means 算法时间复杂度,步骤如下。

步骤 1　更新异常指示矩阵 $\boldsymbol{O}$ 的时间复杂度为 $O(n)$;

步骤 2　搜索单向近邻簇的时间复杂度为 $O(k^2)$;

步骤 3　排序单向近邻簇的时间复杂度为 $O(km\log m)$,其中 $m(1 \leqslant m \leqslant k)$ 为单向近邻簇数;

步骤 4　更新聚类指示矩阵 $\boldsymbol{W}$ 的时间复杂度为 $O(m(n_a-n')+n)$,其中 n_a-n' 为动态域的样本数。实际上,去除异常样本点后,减少了类簇的半径,扩大了静态域的范围。相比 Ball-K-means 算法,动态域样本由 n_a 减少为 n_a-n'。

随着迭代中聚类结果变得更稳定,静态域扩大,时间消耗会越来越少。由于静态域无须距离计算,进一步提升了算法效率。EK-means 在最差条件下的时间复杂度与其他算法的对比如表 1 所示,其中 n 为样本数;k 为类簇数;t 为迭代次数;m 为邻居簇数;n_a 为动态域样本数;n' 为动态域减少样本数。Ann[19]、Exp-ns[20]、Ball-K-means[21] 及 EK-means 优化了算法的执行效率,而其余算法并未改进执行效率。

3　仿真实验

为测试算法的性能及效率,分别在 4 个合成数

表 1 不同算法的时间复杂度

Table 1 Time complexity of different algorithms

算法	时间复杂度
K-means	$O(knt)$
K-means++[22]	$O(knt)$
K-mediods[23]	$O(knt)$
KMOR[15]	$O(knt)$
NK-means[9]	$O(knt)$
Fast t-*K*-means++[10]	$O(knt)$
Ann[19]	$O((k\log k + n\log k + k^2 + kn)t)$
Exp-ns[20]	$O((k^2\log k + n\log\log k + k^2 + kn)t)$
Ball-*K*-means[21]	$O((k^2 + km\log m + mn_a + n)t)$
E*K*-means	$O((k^2 + km\log m + m(n_a\text{-}n') + n)t)$

据集和 6 个真实数据集上进行仿真实验,采用 9 个聚类算法进行对比。对比算法可分为传统的聚类算法、鲁棒的聚类算法以及快速的聚类算法。其中 *K*-means、*K*-means++和 *K*-mediods 是基于划分的传统聚类算法;KMOR、*NK*-means 和 Fast t-*K*-means++是具有异常检测的鲁棒聚类算法;Ann、Exp-ns 和 Ball-*K*-means 算法是减少了距离计算次数的快速聚类算法。性能评价采用主流聚类评价指标:准确度 *ACC* 及归一化互信息 *NMI*。以上评价指标的取值均为[0,1],当其取值越接近 1 时表示聚类性能越优异。

实验环境为 Intel i7-10875 H,30 GHz,16.0 GB RAM,MATLAB2020,CLION2020。

3.1 合成数据集仿真实验

构建 4 个人工合成数据集{D_1, D_2, D_3, D_4},每个数据集中包含不同类簇分布的正常样本以及不同比例的异常样本,D_1 和 D_4 为球状类簇分布,各数据集的类簇分布如表 2 所示。

表 2 合成数据集的信息

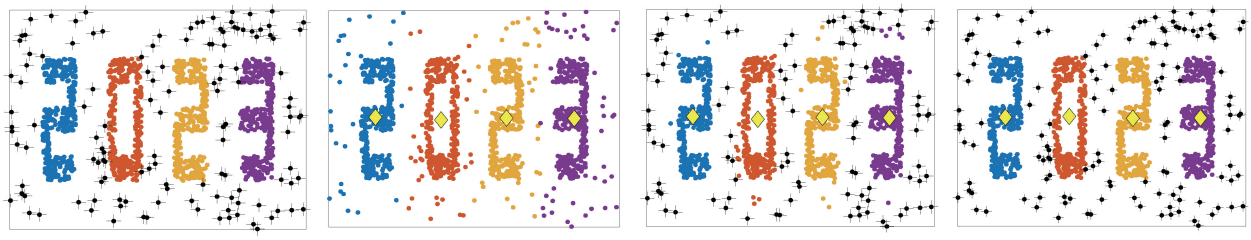
Table 2 Information of synthetic datasets

数据集	样本数	特征数	类簇数	异常数据占比/%
D_1	945	2	9	5
D_2	1 419	2	4	10
D_3	6 000	2	5	20
D_4	4 100	3	4	5

图 2 为各算法在合成数据集 D_2 上的聚类效果。其中,黄色菱形表示类簇中心,黑色加号表示异常样本点。*K*-means 算法将异常样本标记为正常样本。与 KMOR 算法相比,本文算法标记出了所有异常样本,证明了本文算法能够更加准确地识别异常样本。

合成数据集 D_3 中共包含 6 000 个样本,其异常样本数量占比为 20%。图 3 为各算法在合成数据集 D_3 上的聚类效果。如图 3 所示,在 KMOR 和 *K*-means 算法的聚类效果中,类簇中心偏移,导致大量异常样本被误标记为正常样本的情况,且后者的表现更为明显。而在 E*K*-means 算法的聚类结果中,类簇中心未发生偏移,只有极少量异常样本被误判。本文算法能有效降低异常样本对于聚类过程的影响。

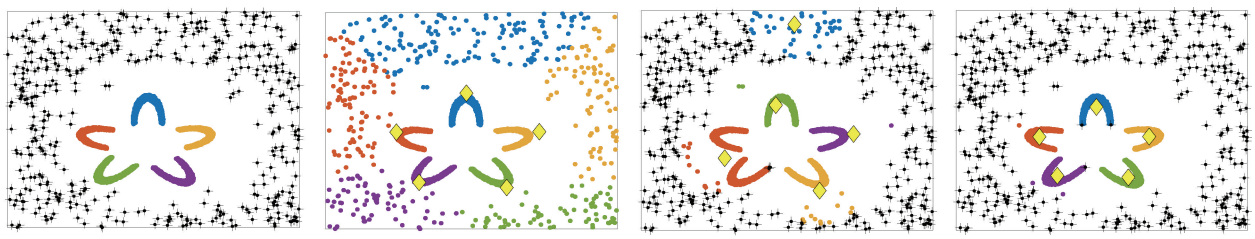
为测试 E*K*-means 算法的执行效率,进一步记录了在人工合成数据集上完成聚类的距离计算次数与算法执行时间,如图 4 所示,其中,*DC* 表示欧氏距离计算次数的倒数,*T* 表示时间的倒数。为方便观察对比效果,将 D_3 和 D_4 的欧氏距离计算次数除以 10



(a) 原始数据集 (b)*K*-means (c)KMOR (d)E*K*-means

图 2 在合成数据集 D_2 的聚类效果

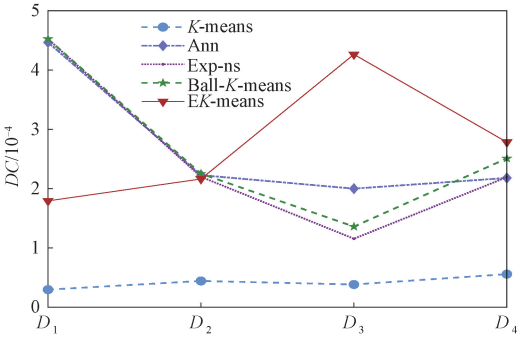
Figure 2 Clustering results on D_2



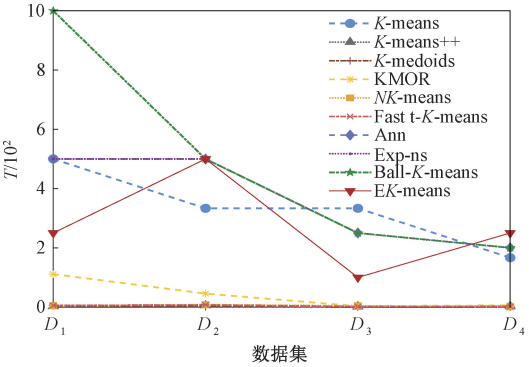
(a) 原始数据集 (b)*K*-means (c)KMOR (d)E*K*-means

图 3 在合成数据集 D_3 的聚类效果

Figure 3 Clustering results on D_3



(a)距离计算次数



(b)运行时间

图 4　各算法在合成数据集上的执行效率

Figure 4　Efficiency of algorithms on synthetic datasets

后进行作图。在图 4(a)中,Ann、Exp-ns、Ball-*K*-means 及 E*K*-means 算法皆优化了 *K*-means 算法的效率,减少了距离计算次数,其中 E*K*-means 的表现较为优异。由图 4(b)可以看出,E*K*-means 算法效率更高。因此,本文通过区域分割,有效提高了聚类算法的执行效率。异常样本的去除使各类簇分布更加紧密,静态域范围被扩大,单次迭代的距离计算次数减少。

3.2　真实数据集仿真实验

本文采用 6 个真实数据集验证所提方法的有效性,包括 Wine、Libras、Segmentation、Accelerometer、PenDigits、Letter,如表 3 所示。对 Wine 数据集添加值在[0,1]上的随机异常点,对其余 5 个真实数据集添加数值在[0,2]上的随机异常点。随机异常点的占比约为 10%,且其分布偏离正常样本。

表 3　真实数据集的信息

Table 3　Information of real datasets

数据集	样本数	特征数	类簇数	异常数
Wine	196	13	3	18
Libras	396	90	15	36
Segmentation	2 541	19	7	231
PenDigits	12 092	16	10	1 100
Letter	22 000	16	26	2 000
Accelerometer	168 300	3	3	15 300

表 4、5 为 E*K*-means 算法及多个对比算法的聚类结果,实验选取 10 次结果的平均值,表 4 为聚类结果的 *ACC* 值,表 5 为聚类结果的 *NMI* 值。

由表 4、5 可知,*K*-means++、Ann、Exp-ns 及 Ball-*K*-means 算法优化了初始类簇中心选取,但并不具备异常点检测能力。KMOR、*NK*-means 及 Fast t-*K*-means++算法在性能和异常检测方面表现较好,但各有限制。KMOR 算法无法检测偏离程度较小的异常样本,*NK*-means 无法处理较密集的异常样本,而 Fast t-*K*-means++无法处理不符合高斯分布的异常样本。本文算法在各个数据集上的聚类指标大部分优于对比算法,其余聚类指标接近于最佳聚类评分。因此,结合实验结果可以证明本文算法通过将异常检测融入聚类过程,有效提高聚类准确率。

图 5 为各算法在 6 个真实数据集上的执行效率,分别展示了算法欧氏距离计算次数及算法执行时间,其中 *DC* 为欧氏距离计算次数的倒数,*T* 为时间的倒数。为方便观察对比效果,将 Libras、Segmentation、PenDigits、Letter、Accelerometer 数据集的

表 4　真实数据集聚类 *ACC* 比较

Table 4　Real datasets clustering *ACC* comparison

算法	*ACC*					
	Wine	Libras	Segmentation	PenDigits	Letter	Accelerometer
K-means	0.848±0.171	0.171±0.039	0.538±0.070	0.552±0.559	0.227±0.008	0.379±0.001
K-means++	0.667±0.140	0.312±0.034	0.589±0.055	0.641±0.058	0.245±0.014	0.380±0.001
K-mediods	0.730±0.153	0.388±0.025	0.568±0.089	0.587±0.044	0.231±0.022	0.380±0.001
KMOR	0.832±0.148	0.407±0.021	0.570±0.043	0.669±0.034	0.246±0.010	0.411±0.002
NK-means	0.827±0.146	0.400±0.024	0.602±0.055	0.645±0.062	**0.259±0.009**	
Fast t-*K*-means++	0.752±0.159	0.329±0.064	0.581±0.048	0.649±0.054	0.242±0.010	0.383±0.000
Ann	0.873±0.679	0.403±0.030	0.583±0.015	0.661±0.038	0.252±0.008	0.378±0.001
Exp-ns	0.806±0.141	0.415±0.017	0.585±0.049	0.652±0.025	0.250±0.008	0.379±0.001
Ball-*K*-means	0.812±0.146	0.419±0.024	0.581±0.043	0.666±0.052	0.250±0.013	0.379±0.001
E*K*-means	**0.911±0.087**	**0.423±0.026**	**0.616±0.041**	**0.671±0.019**	0.256±0.009	**0.418±0.002**

表 5 真实数据集聚类 *NMI* 比较

Table 5 Real datasets clustering *NMI* comparison

算法	*NMI*					
	Wine	Libras	Segmentation	PenDigits	Letter	Accelerometer
K-means	0.624±0.166	0.164±0.049	0.520±0.084	0.534±0.040	0.297±0.012	0.036±0.001
K-means++	0.467±0.160	0.378±0.040	0.579±0.018	0.612±0.032	0.316±0.009	0.038±0.001
K-mediods	0.553±0.187	0.485±0.039	0.552±0.064	0.570±0.033	0.294±0.026	0.038±0.001
KMOR	0.683±0.138	0.549±0.017	0.625±0.023	**0.665±0.011**	0.333±0.008	**0.083±0.001**
NK-means	0.674±0.120	0.530±0.026	**0.637±0.016**	0.647±0.035	0.349±0.005	
Fast t-*K*-means++	0.550±0.169	0.397±0.085	0.577±0.018	0.620±0.032	0.315±0.016	0.030±0.000
Ann	0.679±0.107	0.524±0.027	0.586±0.015	0.627±0.018	0.336±0.008	0.037±0.001
Exp-ns	0.600±0.145	0.538±0.019	0.589±0.016	0.628±0.014	0.338±0.003	0.037±0.001
Ball-*K*-means	0.613±0.157	0.537±0.016	0.588±0.011	0.635±0.023	0.339±0.006	0.037±0.001
E*K*-means	**0.764±0.099**	**0.556±0.018**	0.601±0.009	0.659±0.016	**0.354±0.005**	0.071±0.001

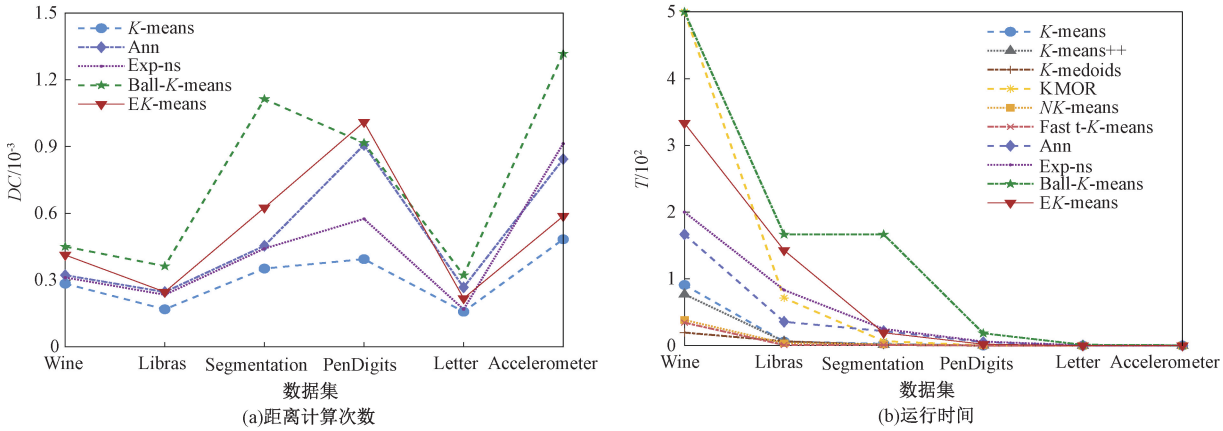


图 5 E*K*-means 算法在真实数据集上的执行效率

Figure 5 Efficiency of E*K*-means on real datasets

欧氏距离计算次数分别除以 10、10^2、10^3、10^4、10^4。在图 5(a)中,E*K*-means 算法在大部分数据集上达到了仅次于最少距离计算次数的 Ball-*K*-means 算法,且在 PenDigits 数据集上距离计算次数为最小值。在图 5(b)中,E*K*-means 算法对于效率的优化效果较好。因此,本文算法有效提高了聚类的执行效率。

图 6 为本文算法在 6 个真实数据集上的收敛情况。可以看出,本文算法在一定迭代次数内的目标函数值不变,证明其在真实数据集上可以快速收敛。

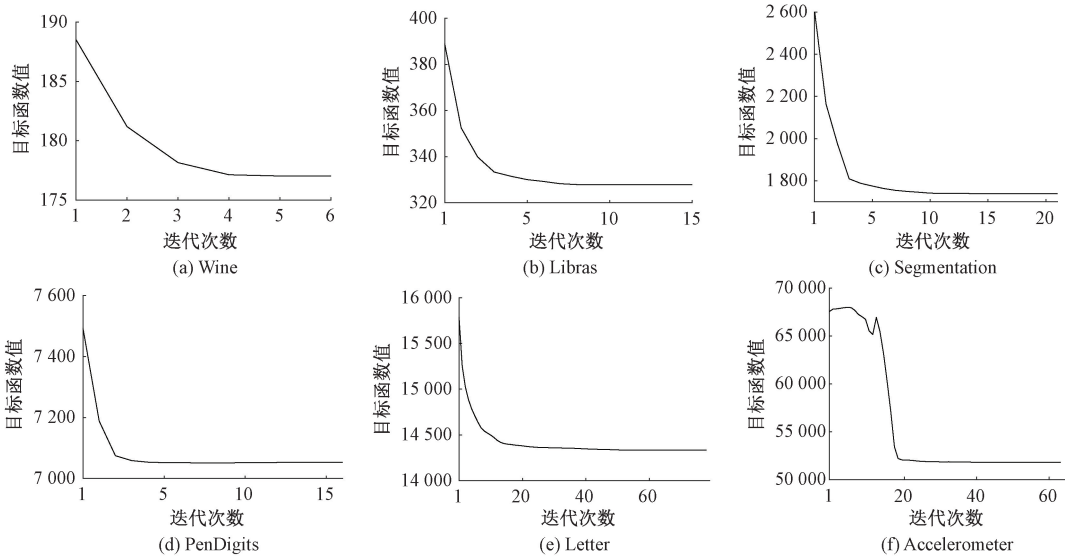


图 6 E*K*-means 算法在真实数据集上的收敛曲线

Figure 6 Convergence curves of E*K*-means on real datasets

表 6 记录了 10 个算法在异常数据占比分别为 0、5%、10% 条件下的 Wine、Libras、Segmentation、PenDigits 数据集的 *ACC* 对比。随着异常数据占比的增加,*K*-means、*K*-means++、Ann、Exp-ns 以及 Ball-*K*-means 算法的 *ACC* 值明显下降,*K*-mediods、KMOR、*NK*-means、Fast t-*K*-means++的 *ACC* 值的下降幅度较小,但 *K*-mediods 及 KMOR 算法的准确率较低。与以上算法相比,本文算法的 *ACC* 值下降幅度最小且聚类效果较优。本文算法通过在聚类中融入异常点检测,增强了算法的鲁棒性及准确率。

表 6　不同异常数据占比条件下的真实数据集聚类结果 *ACC* 对比

Table 6　Clustering *ACC* comparison of datasets with different proportions of outliers

算法	*ACC*											
	Wine			Libras			Segmentation			PenDigits		
	0	5%	10%	0	5%	10%	0	5%	10%	0	5%	10%
K-means	0.944	0.777	0.698	0.405	0.170	0.172	0.584	0.538	0.524	0.660	0.644	0.539
K-means++	0.950	0.883	0.789	0.434	0.380	0.320	0.607	0.598	0.589	0.674	0.665	0.579
K-mediods	0.958	0.806	0.798	0.400	0.410	0.380	0.589	0.602	0.568	0.597	0.596	0.587
KMOR	0.910	0.773	0.832	0.436	0.370	0.410	0.547	0.655	0.570	0.660	0.605	0.669
NK-means	0.954	0.918	0.856	0.422	0.415	0.400	0.585	0.610	0.602	0.705	0.665	0.645
Fast t-*K*-means++	0.949	0.915	0.874	**0.438**	**0.440**	0.340	0.580	0.584	0.581	**0.706**	**0.690**	0.588
Ann	0.947	0.901	0.810	0.420	0.390	0.405	0.565	0.595	0.583	0.680	0.647	0.652
Exp-ns	0.948	0.918	0.806	0.423	0.395	0.420	0.570	0.612	0.585	0.685	0.652	0.605
Ball-*K*-means	0.947	0.908	0.843	0.423	0.402	0.425	0.585	0.614	0.581	0.681	0.652	0.607
E*K*-means	**0.969**	**0.923**	**0.891**	0.421	0.422	**0.426**	**0.662**	**0.641**	**0.641**	0.688	0.663	**0.672**

4　结论

本文提出了一种融合异常检测与区域分割的高效 *K*-means 聚类算法(E*K*-means)。针对传统 *K*-means 算法容易受到异常样本影响,导致其聚类准确度较低的问题,构建统一聚类模型,通过异常检测与聚类过程的交互协同,获取较优聚类效果。同时,利用近邻簇搜索技术,对各类簇进行区域分割,减少了样本分配阶段的冗余计算,提高了算法执行效率。最后,通过合成数据集和真实数据集上的仿真实验证明本文算法的有效性。

E*K*-means 算法需要预指定异常样本的数量,但在实际应用中,异常样本的数量难以确定。此外,针对算法的不稳定性,思考是否结合多次运行的聚类结果,获得更稳定的聚类结果,因此,如何在异常样本数量未知的先验条件下,快速稳定地检测与处理异常样本将成为后续研究的目标和方向。

参考文献:

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

[2] BAI L, LIANG J Y, ZHAO Y X. Self-constrained spectral clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(4): 5126-5138.

[3] ZHANG Q H, DAI Y Y, WANG G Y. Density peaks clustering based on balance density and connectivity[J]. Pattern Recognition, 2023, 134: 109052.

[4] HIRECHE C, DRIAS H, MOULAI H. Grid based clustering for satisfiability solving[J]. Applied Soft Computing, 2020, 88: 106069.

[5] MONATH N, KOBREN A, KRISHNAMURTHY A, et al. Scalable hierarchical clustering with tree grafting[C]// Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM, 2019: 1438-1448.

[6] WAN Y C, LIU X B, WU Y, et al. ICGT: a novel incremental clustering approach based on GMM tree[J]. Data & Knowledge Engineering, 2018, 117: 71-86.

[7] 鲁斌, 范晓明. 基于改进自适应 *k* 均值聚类的三维点云骨架提取的研究[J]. 自动化学报, 2022, 48(8): 1994-2006.
LU B, FAN X M. Research on 3D point cloud skeleton extraction based on improved adaptive *k*-means clustering [J]. Acta Automatica Sinica, 2022, 48(8): 1994-2006.

[8] LI Z, TANG C, ZHENG X, et al. Unified *K*-means coupled self-representation and neighborhood kernel learning for clustering single-cell RNA-sequencing data[J]. Neurocomputing, 2022, 501: 715-726.

[9] IM S, QAEM M M, MOSELEY B, et al. Fast noise removal for *K*-means clustering[C]// the 23rd International Conference on Artificial Intelligence and Statistics(AISTATS)2020. Palermo: AISTATS, 2020: 456-466.

[10] LI Y M, ZHANG Y, TANG Q T, et al. T-*k*-means: a

robust and stable k-means variant[C]//International Conference on Acoustics, Speech and Signal Processing (ICASSP). Piscataway: IEEE, 2021: 3120-3124.

[11] GRUNAU C, ROZHON V. Adapting K-means algorithms for outliers[C]//the 39th International Conference on Machine Learning. Baltimore: ICML, 2022: 7845-7886.

[12] ZHANG Z, FENG Q L, HUANG J Y, et al. A local search algorithm for k-means with outliers[J]. Neurocomputing, 2021, 450: 230-241.

[13] HUANG S D, REN Y Z, XU Z L. Robust multi-view data clustering with multi-view capped-norm K-means[J]. Neurocomputing, 2018, 311: 197-208.

[14] HUANG S D, KANG Z, XU Z L, et al. Robust deep k-means: an effective and simple method for data clustering[J]. Pattern Recognition, 2021, 117: 107996.

[15] HAUTAMÄKI V, CHEREDNICHENKO S, KÄRKKÄINEN I, et al. Improving k-means by outlier removal[C]//Proceedings of the 14th Scandinavian conference on Image Analysis. New York: ACM, 2005: 978-987.

[16] PENG D W, CHEN Z Z, FU J C, et al. Fast k-means clustering based on the neighbor information[C]//ISEEIE 2021: 2021 International Symposium on Electrical, Electronics and Information Engineering. New York: ACM, 2021: 551-555.

[17] GIFFON L, EMIYA V, KADRI H, et al. Quick-means: accelerating inference for K-means by learning fast transforms[J]. Machine Learning, 2021, 110(5): 881-905.

[18] HAMERLY G. Making k-means even faster[J]. Proceedings of the 10th SIAM International Conference on Data Mining, 2010: 130-140.

[19] DRAKE J. Faster K-means clustering[EB/OL]. (2013-09-24)[2023-06-13]. http://hdl.handle.net/2104/8826.

[20] NEWLING J, FLEURET F. Fast K-means with accurate bounds[C]//Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48. New York:ACM, 2016: 936-944.

[21] XIA S Y, PENG D W, MENG D Y, et al. Ball k-means: fast adaptive clustering with No bounds[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44(1): 87-99.

[22] ARTHUR D, VASSILVITSKII S. K-means++: the advantages of careful seeding[C]//Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. New York: ACM, 2007: 1027-1035.

[23] KAUFMAN L, ROUSSEEUW P. Clustering by means of medoids[EB/OL]. (1987-01-01)[2023-06-13]. https://www.researchgate.net/publication/243777819_Clustering_by_Means_of_Medoids.

Efficient *K*-means with Region Segment and Outlier Detection

YIN Hongwei[1,2], HANG Yuqing[1,2], HU Wenjun[1,2]

(1. College of Information Engineering, Huzhou University, Huzhou 313000, China; 2. Zhejiang Key Laboratory of Smart Management & Application of Modern Agricultural Resources, Huzhou University, Huzhou 313000, China)

Abstract: In the traditional K-means and many improved algorithms, the inability to explicitly handle outliers, resulted in their poor clustering performance. To solve this problem, in this paper, an efficient K-means with region segment and outlier detection was proposed. Firstly, to obtain better clustering results, an unified clustering model to form an interactive collaboration between outlier detection and clustering was constructed. Secondly, to improve algorithm efficiency, clusters were adaptively segmented through near neighbor clusters search to reduce redundant calculations. Finally, on synthetic datasets and real datasets were tested to verify the effectiveness of the proposed method. The experimental results showed that EK-means algorithm outperformed other algorithms in terms of clustering performance and execution efficiency. The *ACC* could reach 0.911 in the Wine dataset.

Keywords: clustering; K-means; outlier detection; region segment; near neighbor clusters search; adaption