ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.6, June 2008, pp.1283−1300
DOI: 10.3724/SP.J.1001.2008.01283

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于多重分形的聚类层次优化算法*

闫光辉[1,2+], 李战怀[1], 党建武[2]

[1](西北工业大学 计算机学院,陕西 西安 710072)

[2](兰州交通大学 电子与信息工程学院,甘肃 兰州 730070)

Finding Natural Cluster Hierarchies Based on MultiFractal

YAN Guang-Hui[1,2+], LI Zhan-Huai[1], DANG Jian-Wu[2]

[1](School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

[2](School of Information and Electrical Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

+ Corresponding author: E-mail: yangh@mail.nwpu.edu.cn

Yan GH, Li ZH, Dang JW. Finding natural cluster hierarchies based on MultiFractal. *Journal of Software*, 2008,19(6):1283−1300. http://www.jos.org.cn/1000-9825/19/1283.htm

Abstract: A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. Moreover, there will exist more or less similarities among these large amounts of initial cluster results in real life data set. Accordingly, analyzer may have difficulty to implement further analysis if they know nothing about these similarities. Therefore, it is very valuable to analyze these similarities and construct the hierarchy structures of the initial clusters. The traditional cluster methods are unfit for this cluster post-processing problem for their favor of finding the convex cluster result, impractical hypothesis and multiple scans of the data set. Based on multifractal theory, this paper proposes the FCHO (fractal-based cluster hierarchy optimization) algorithm, which integrates the cluster similarity with cluster shape and cluster distribution to construct the cluster hierarchy tree from the disjoint initial clusters. The elementary time-space complexity of the FCHO algorithm is presented. Several comparative experiments using synthetic and real life data set show the performance and the effectivity of FCHO.

Key words: data mining; clustering; multifractal; post-processing; optimization

摘 要: 大量初始聚类结果之间存在强弱不同的相似性,会给用户理解与描述聚类结果带来不利影响,进而阻碍数据挖掘后续工作的顺利展开.传统聚类算法由于注重聚类形状及空间邻接性,或者考虑全局数据分布密度的均匀性,实际中均难以解决这一类问题.为此,提出了基于分形的聚类层次优化算法 FCHO(fractal-based cluster hierarchy optimization),FCHO 算法基于多重分形理论,利用聚类对应多重分形维数及聚类合并之后多重分形维数的变化程度来度量初始聚类之间的相似程度,最终生成反映数据自然聚集状态的聚类家族树.此外,初步分析了算法的时空复杂性,基于合成数据集和标准数据集的有关实验工作证实了算法的有效性.

* Supported by the National Natural Science Foundation of China under Grant No.60573096 (国家自然科学基金); the NSFC-JST Major International (Regional) Joint Research Project under Grant No.60720106001 (NSFC-JST 重大国际(地区)合作项目); the Foundation of Gansu Province Educational Department of China under Grant No.0604-09 (甘肃省教育厅基金)

Received 2007-03-01; Accepted 2007-10-09

关键词: 数据挖掘;聚类;多重分形;后续处理;优化
中图法分类号: TP181 文献标识码: A

聚类是将物理或抽象对象的集合分组成为由相似对象组成的多个集合的过程,其中属于同一个集合的对象之间的彼此相似,属于不同集合的对象之间彼此相异(按照某种度量机制).通过聚类操作,我们能够识别数据集的密集和稀疏区域,发现数据的全局分布模式,以及数据间有趣的相互关系.聚类作为数据预处理的重要手段和数据挖掘的基础性支持工具,在统计学和知识发现等领域得到了广泛而深入的研究,国内外研究者针对不同的应用、数据类型与聚类目的,进行了大量卓有成效的研究工作,取得了丰富的研究结果[1−3].

统计聚类方法[4−6]基于欧氏距离度量机制,在低维数据分析中获得了优良的性能和令人满意的聚类结果,但也存在无法发现非球形聚类或者大小差别很大的聚类的缺陷,因而可能降低无监督学习的精度和效率,并给用户进一步的数据分析工作带来了困难甚至误导用户的判断[7−15].基于层次、密度和网格的聚类技术在大数据集、增量及任意形状聚类方面与统计聚类技术相比具有较大的优越性,然而却无法应对聚类内部密度不均匀及聚类非邻接的情况(表现为在数据空间中离散分布的一些数据聚集,宏观上某些非邻接的数据聚集最终都属于同一个聚类,一些常见的分形数据集,如康托尘(Cantor set)).如图 1 所示,以均匀分布生成位于二维数据空间不同位置的 10 个矩形,以及利用正态分布生成的、分别位于二维空间的 4 个象限内的、由同一圆环分割而成的 4 个四分之一圆环(为对比实验方便,命名该数据集为 D2-20K),通过聚类操作我们容易得到如图 1 所示的 9 个聚类(其中红色圆圈对应稀疏数据点),分别对应中心交叉矩形框、4 个圆环聚类和位于四角的 4 个相交矩形框.然而,根据数据集的生成方式我们得知矩形数据集之间具有较强的相似性,同时,圆环数据集之间也具有较强的相似性.遗憾的是,当前的聚类技术忽视这种相似性,因而也就无法对初始聚类结果进一步优化,错失了发现更隐藏、更有趣模式的机会.基于模型的聚类方法[16,17]根据数据由潜在的概率分布生成的假设,试图优化给定的数据集和潜在概率模型之间的适应性,从而进行相应的聚类工作.基于模型的方法一方面需要对数据集施以较强的限定,而这种限定在实际应用中很难满足(如属性相互独立);另一方面,还存在训练时间过长及无法适应于大数据集的情况,同时也无法满足数据流等增量聚类的要求,因而难以在实际中应用.

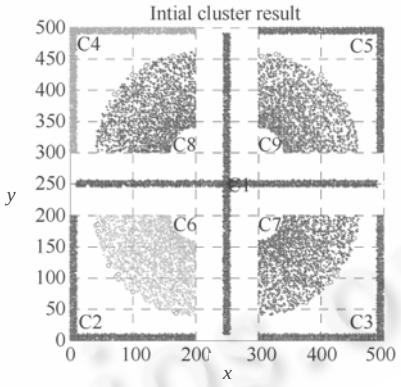

Fig.1 Initial cluster results

图 1 初始聚类结果

聚类定义的本质依赖于模式之间的相似性,属于同一个聚类的模式之间相似性最大,属于不同聚类的模式之间相似性最小,同样在聚类合并过程中也应该遵循这一原则,即进行合并的初始聚类之间也必然存在某种相似性,但这种相似性仅仅依靠聚类之间的距离无法准确度量.就我们所知,对于图 1 所示的数据集,当前的聚类算法得到的聚类结果都无法反映数据集的自然聚集状况.通过分析,容易发现该问题与传统聚类问题的特征及要求不完全一致,即对聚类合并的要求不仅限于空间邻接或距离上比较接近,而且更加突出聚类之间的整体相似性,这种相似性可以表现为空间形状的相似性、空间距离的接近性、聚类整体分布的相似性及以上因素的综合等,因而造成传统聚类技术由于仅考虑某一特性而无法成功地解决这一类问题.

我们将这种问题称为非邻接聚类问题,或将其推广为聚类后续处理问题.该问题主要关注于每个初始聚类作为一个整体而言的结构与行为特征,要求能够跨越空间和局部的联系,这与经典聚类算法主要关注空间联系和局部特性有着完全不同的侧重点.为了发现数据集中隐藏的,更有趣的及自然的模式,我们提出利用多重分形技术进行初始聚类结果的后续处理.

第 1 节介绍了相关的基本工作,并分析了当前聚类算法在面对聚类后续处理问题的不足之处;第 2 节详细论述了 FCHO 算法的基本思想、实现过程;第 3 节基于合成数据集和实际数据集对 FCHO 算法进行了验证性测试工作,给出了相关的运行结果及聚类合并家族树,对算法的时空复杂性进行了初步分析;最后对研究工作做了阶段性总结,同时展望了研究工作的方向.

1 相关工作

1.1 分形及分形维数

20 世纪 70 年代诞生的分形理论在空间数据查询效率分析[18,19]、多维索引技术[20]等研究领域得到了广泛的应用.下面给出一些相关的基本概念.

定义 1(分形). 一个数据集基于不同的观察尺度表现出部分和整体的相似性(可以表现为空间形状自相似性,也可以表现为统计分布上的自相似性),称该数据集为一个分形,许多实际的数据集呈现统计分形的特征.本文主要从数据集的统计分形角度来进行相关的聚类研究工作.

定义 2(嵌入维数). 嵌入维数是数据集位于的数据空间的维数,也就是数据集对应的属性数目.

定义 3(内在维数). 内在维数是数据集合的真正表现维数.

内在维数不随数据集嵌入维数的不同而改变,在一定程度上定量地描述了数据集内部结构的复杂性.例如:一条直线若位于二维平面内,其嵌入维数为 2,若位于三维空间之中,则嵌入维数为 3,但无论位于多少维空间中,其内在维数都大致为 1.此外,内在维数还可以在一定程度上反映属性之间的非线性相关性,如二维空间中的圆形对应的内在维数大致为 1.在实际应用中,鉴于内在维数的计算复杂性,经常采用分形维数来近似内在维数[19],图 2 给出了不同数据集位于不同的嵌入空间所对应的分形维数(第 1 行对应数据集分布情况,中间行对应关联分形维数,第 3 行对应相应的多重分形维数 q–D_q 曲线).

考虑到实际数据分布的多样性和复杂性,仅以某单一分形维数(如盒维数,信息维数及关联维数)为特征,难以区分单一分形集(single fractal)和多重分形集(multiple fractal),同时也存在不同分布的数据集却具有比较接近的单一分形维数的特殊情况.如图 2 所示,二维空间圆形与三维空间直线对应的关联分形维数比较接近,因此,单独利用关联分形维数也就无法区分这两个数据集.为了能够更精确地描述一个分形集,同时也为了能够更加有效地支持聚类后续处理工作,需要增加能刻画数据集更精细特征的不同的分形参数,由此引入了多重分形维数和多重分形理论,有关多重分形维数的详细内容见文献[21].

定义 4(多重分形维数). 对于一个在区间 $[r_1,r_2]$ (无标度区)内呈现统计自相似特征的数据集,其多重分形维数 D_q 定义如下:

$$D_q=\begin{cases}\lim\limits_{r\to 0}\dfrac{\sum_i p_i\mathrm{Log}p_i^q}{\log r}, & q=1\\ \lim\limits_{r\to 0}\dfrac{1}{q-1}\dfrac{\mathrm{Log}\sum_i p_i^q}{\mathrm{Log}r}, & q\neq 1\end{cases},\qquad r\in[r_1,r_2] \tag{1}$$

其中,r 是覆盖数据空间所用的网格的边长,P_i 为网格的奇异性测度,即网格所包含的数据点数目与全部数据点数目之比.已经证明,$\lim\limits_{q\to 0}D_q=D_0$ 即是豪斯道夫维数(盒维数),$\lim\limits_{q\to 1}D_q=D_1$ 为信息维数,$\lim\limits_{q\to 2}D_q=D_2$ 为关联维数,这 3 种维数同时也是在实际中经常用到的几种分形维数.根据式(1),理论上可以给出无限多的 D_q 定义,从而得到广义分形维数谱.其中,$q>0$ 的高阶维数 D_q 反映了分形集中点的聚集程度,即由 q 个点构成的集团分布情况,相应地,$q<0$ 的高阶维数 D_q 则反映了分形集空隙分布情况.由此可见,引入广义分形维数可以更精确地反映数据集的

分布情况,从而为数据集描述提供了更加有力的工具.另外,广义分形维数的引入也为我们对数据集进行分类带来了更有力的工具.例如:当两个数据集的单一分形维数如 D_0 接近甚至相同时,单独依靠 D_0 就无法区分这两个数据集合,而如果它们对应的信息维数 D_1 或关联维数 D_2 取值不同,则我们就可以根据 D_1 或 D_2 对它们进行分类.基于这种思路,可选的 q 次分形维数越多,则其相应的区分不同数据集的能力也就越强.考虑到实际应用的需求及计算资源的限制,一般情况下,利用 D_{10} 及 D_{-10} 作为可选的分形维数.从图 2 第 3 行容易看出具有相近 D_2 值的圆形和直线数据集对应的 D_{-10} 和 D_{10} 差别明显,可以作为区分这两个数据集的特征使用.

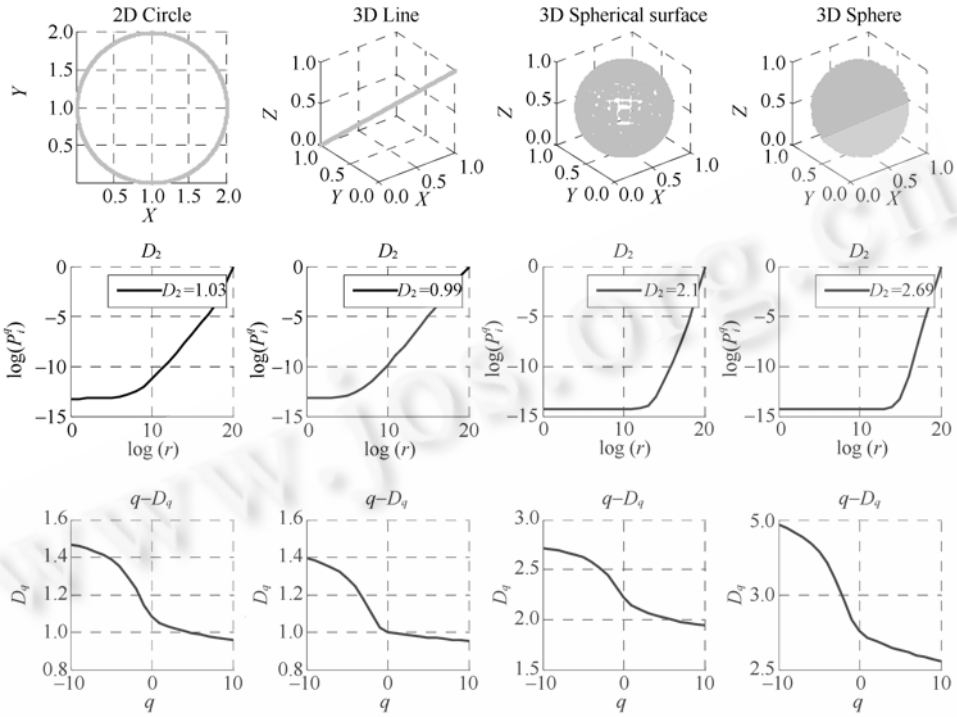

Fig.2 Data sets and the q–D_q plots

图 2 数据集及其对应 q–D_q 曲线

针对实际数据集中可能包含各种形式的分布情况、聚类密度不均匀以及多重分形维数的特点,我们考虑采用 D_{-10},D_2 和 D_{10} 作为度量聚类相似程度的主要依据,其中, D_{-10},D_2 和 D_{10} 在聚类合并过程中各有侧重点:在判断聚类能否合并的情况下,D_{-10},D_2 和 D_{10} 具有同样的权重(主要目的在于从聚类的离散度,密集度及形状等多方面来综合考虑),而聚类合并之后(假定聚类之间不存在相互包含的情况,则合并后聚类的离散度会大于合并之前聚类的离散度) D_{-10} 会发生比较大的变化,相应地,我们着重利用 D_2 和 D_{10} 来判断合并操作是否合法.由此,基于多重分形的聚类后续处理技术同时考虑聚类内部数据分布特点及聚类整体形态特征,可以发现潜在的、更能反映数据集本质特性的模式.

1.2 层次、网格、密度及分形聚类技术

层次聚类是一种有效的聚类技术,它通过将数据对象组成为一棵聚类的树,为用户提供多种可选的聚类结果,可以根据需要随时结束聚类操作过程,因而是目前广泛应用的一种聚类技术.层次聚类的具体实现过程分为两种形式:自底向上凝聚(agglomerative)与自顶向下分裂(divisive),凝聚与分裂操作都要基于某种特定的相似性度量机制.单连接方式与全连接方式是两种常见的度量聚类之间相似性的技术.单连接是计算分别位于两个聚类中的模式之间的最大相似度;全连接方式则相反,计算分别位于两个聚类中的模式之间的最小相似度,随后的

聚类凝聚与分裂操作也主要基于所得到的相似度来进行.BIRCH[7](balanced iterative reducing and clustering using hierarchies)算法引入聚类特征与聚类特征树进行概括聚类描述,首先扫描数据集并建立一个基于内存的聚类特征树,然后采用某个聚类算法对聚类特征树的叶子结点进行聚类操作.BIRCH 算法的主要问题在于其利用半径或直径来控制聚类的边界,因而无法适应于非球形聚类的情况.CURE[8](clustering using representatives)通过选择数据空间中固定数目的具有代表性的点来描述聚类,ROCK[9](robust clustering using linKs)算法实际上是 CURE 算法针对分类属性的扩充版本.Chameleon[10]算法首先构造数据集的 k-最近邻居图,然后基于图的划分技术将数据对象聚集为大量相对较小的子聚类,最后利用一个凝聚的层次聚类算法反复地合并子聚类来寻找真正的结果聚类.层次聚类算法一般需要用户输入参数,如聚类个数等来结束算法的执行.

网格聚类技术首先将对象空间划分为互不相交单元(网格)以构成网格结构,然后利用网格结构完成聚类操作,具有一遍扫描数据集及运行时间与数据点数无关的特点,在高维、海量数据聚类领域获得了较为广泛的应用,目前在数据流聚类领域也出现了利用网格技术进进行聚类的尝试.网格聚类技术的不足之处在于,网格的数目随着数据集合维度的增加而呈指数增长,针对这一缺陷,基于网格的聚类算法要么仅保留超过某一密度阈值的网格,要么采用 Apriori 原理进行网格的预先删除,而进行网格的预先删除需要各维之间的连接工作,无法适应动态数据流环境.STING[13]是首个基于网格的聚类技术,它利用一个互不相交的且层次嵌套的网格结构覆盖数据空间,通过一遍扫描数据集,利用底层网格所包含的统计信息逐层递推得到上层网格的统计信息,从而支持不同粒度的多次查询请求,同时,基于底层网格的统计信息也可以进行相关的聚类工作. Wave-Cluster[14]和 CLIQUE[15](clustering in QUEst)则是将基于网格与基于密度相结合的方法.

典型的基于密度方法包括:DBSCAN[11](density-based spatial clustering of applications with noise)、OPTICS[12](ordering points to identify the clustering structure)等.该方法将聚类定义为一组“密度连接”的点集,通过不断生长足够高密度区域从含有噪声的数据集中发现任意形状的聚类.DBSCAN 是第一种利用密度的聚类算法,它需要用户设置两个参数 Eps(密集区域半径)和 MinPts(最少密集近邻数目)用于控制正常聚类的密度.但由于实际高维数据经常分布不均匀,因而造成全局密度参数无法精确刻画其内在的聚类结构.OPTICS 算法针对 DBSCAN 算法的缺陷,提供一个逐步增加的密度区域顺序来代替聚类操作,并能从中发现用户所需要的聚类信息,但 OPTICS 算法本质上也是基于欧氏距离的,同样无法满足聚类后续处理的需求.

国内研究人员针对当前聚类算法的不足也做了一定的研究工作[22].提出将密度与距离度量机制相结合的聚类算法,提高了聚类的质量及发现任意形状互连聚类能力[23].提出结合参考点和密度的聚类算法,从而进一步地增强算法发现任意形状聚类能力.文献[24]提出一种基于主次属性划分的聚类方法及数据可视化方法.文献[25]提出二分网格划分机制结合动态网格密度来发现任意形状、非均匀密度分布的聚类.

从本质上分析,层次聚类技术、网格及密度聚类技术所采用的相似性度量机制仍然是基于欧氏距离和邻接特征的,因而也就无法跳出只能发现空间上邻接的聚类的局限,无法解决非邻接聚类问题.

分形理论通过分析部分与整体的相似性来描述数据的分布特征,是研究数据之间的相似性以及数据集部分与整体相似性的有力工具.多重分形既关注数据集的整体分布与行为特征,同时又引入多重分形维数来描述数据集局部的结构与行为特征,可以提供更精细的相似性度量机制.因此,利用多重分形理论同时考虑部分与整体相似性的固有特点,引入多重分形技术进行相关的聚类工作尤其是聚类后续处理工作在理论与技术上是可行的.Barbará 首次提出基于分形的 FC(fractal clustering)聚类算法[26],分析了 FC 算法的时空复杂度.针对合成与实际数据集的实验结果显示 FC 算法对属性数目及数据集规模具有较好的可扩展性,在算法的执行时间、存储需求及聚类结果的优劣方面对 FC 算法、BIRCH 及 CURE 等聚类技术进行了实验对比分析,结果表明,FC 算法具有相对优良的聚类结果,此时,上述三者面对大数据集的执行时间与存储空间需求基本相当.FC 算法具有发现任意形状聚类及增量聚类的特点,但算法仍然利用聚类之间的最近距离作为聚类合并的唯一依据,因而同样无法解决非邻接聚类问题.此外,FC 算法也无法得到聚类合并的层次结构.基于多重分形技术,我们提出了聚类层次优化算法 FCHO(fractal-based cluster hierarchy optimization),FCHO 算法利用整型 Z-Ordering 技术[27]通过一遍扫描数据集来创建初始的底层网格结构,基于该底层网格结构利用密度和网格的聚类技术来生成初始聚类,

结合多重分形技术优化初始聚类结果,针对优化的初始聚类结果自底向上构建聚类家族树,从而在更高程度上支持用户的数据分析工作,进而发现数据中更为隐蔽的模式.

2 基于分形的聚类层次优化算法

2.1 数据结构及概念

设 $A=\{A_1,A_2,...,A_d\}$ 是有界域的集合, $S=A_1\times A_2\times...\times A_d$ 是一个 d 维数据空间,其中 $A_1,A_2,...,A_d$ 表示 S 的 d 个维或 d 个属性域. $X=\{x_1,x_2,...,x_N\}$ 表示 S 上的 N 个点的集合,其中, $x_i=\langle x_{i_1},x_{i_2},...,x_{i_d}\rangle$ 表示一个数据点, $x_{i_j}\in A_j$ 表示数据点 x_i 的第 j 维的值,也就是数据点 x_i 的第 j 维坐标.

为便于计算数据集的分形维数,类似于STING算法构建多层网格结构,以二维数据空间为例,所构建的多层网格结构如图 3 所示.数据空间格式化为边长为 1 的单位网格,则嵌套的下层网格的边长按照如下方式划分: $1/2^0,1/2^1,...,1/2^{level}$,其中 *level* 为网格结构的层数.顶层网格对应数据空间整体,包含 $4(2^2,d$ 维空间为 $2^d)$ 个第二层网格(边长为 1/2),其中,上层的每一个网格均包含其直接下层的 4 个网格(边长为上层网格边长减半).

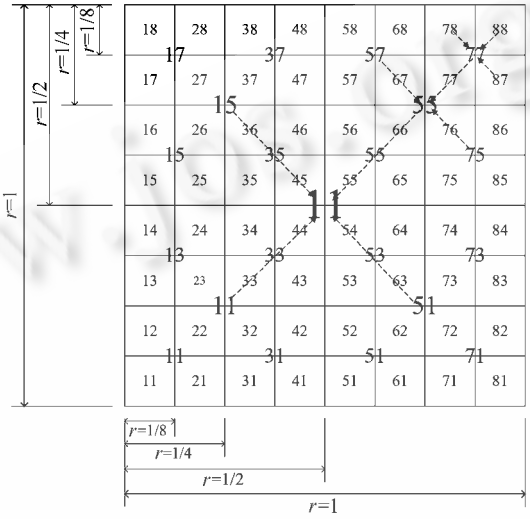

Fig.3 2D embedded grid structure and Z-ordering code

图 3 二维嵌套网格及 Z-ordering 编码

为了满足对网格进行存储以及分形维数的计算需要,[27]引入整型 Z-ordering 编码技术对最低层网格编址,该技术可以实现从底层网格所包含统计信息到最上层网格所包含统计信息的映射工作,图 3 以二维空间为例给出底层网格坐标逐层映射为其直接高层网格坐标的过程,其中,网格坐标的修改操作按照式(2)进行.

$$\begin{cases} C_d=C_s, & \left(\dfrac{C_s-1}{2^{i-1}}\right) \text{ MOD } 2=0 \\ C_d=C_s-2^{i-1}, & \left(\dfrac{C_s-1}{2^{i-1}}\right) \text{ MOD } 2=1 \end{cases} \tag{2}$$

其中, C_s 为修改之前的维坐标值; C_d 为修改之后的对应维坐标值; i 为当前合并的层次数. i=1 对应由最低层向次低层合并, i=2 对应由次低层向其直接上层合并,依此类推.

定义 5(网格选择度). 网格 $Grid_i$ 的选择度为

$$selectivity(Grid_i)=\frac{count(Grid_i)}{|X|},$$

其中, $count(Grid_i)$ 为网格 $Grid_i$ 所包含的数据点数, $|X|$ 为数据集总的数据点数.根据多层嵌套网格结构定义形式及网格坐标映射机制,只要保存底层网格结构及其相应的数据点数目信息,容易得到任意层、任意网格的选

择度.

定义 6(密集网格). 对于 $Grid_i$,若 $selectivity(Grid_i) > |X| \times \tau$,其中,$\tau$ 为网格密集度阈值,则 $Grid_i$ 为密集网格.

定义 7(初始聚类). 数据空间中邻接的密集网格的最大集合即为初始聚类.两个网格 $Grid_i$ 和 $Grid_j$ 邻接的条件之一是它们之间共享一个公共超平面,或者存在另一个网格 $Grid_k$,使得 $Grid_i$ 与 $Grid_k$ 邻接,同时 $Grid_k$ 与 $Grid_j$ 邻接.

规定网格对应的选择度后,给定网格密集度阈值 τ,容易得到密集网格信息,相应地我们可以通过判断邻接的密集网格来进行初始的聚类操作,从而得到初始的聚类结果,图 1 为合成数据集对应的初始聚类情况.分析基于密度和网格的聚类算法过程易知,选择不同的网格密度阈值并不会从根本上改变聚类的形状,虽然我们可以通过改变网格的边长来得到不同的聚类结果,但无论我们怎样改变网格的边长,也无法将所有矩形框生成为一个聚类,同时将 4 个四分之一圆环形成为一个聚类.

因此,我们考虑在已经得到了一定数目的、能基本反映数据集分布情况的初始聚类结果的基础上,如何对初始聚类结果进一步优化,以求能更精确地反映数据的自然聚集情况.基于前面的分析,我们设计了 FCHO 算法对所得到的聚类结果进行后续处理, 从而发现数据中更为隐蔽的模式.

为便于对算法进行描述,表 1 列出了算法所使用的主要符号.

Table 1 Symbols

表 1 算法使用符号

No.	Symbol	Signification
1	S	Data space
2	A	Bounded interval
3	X	Data set
4	*Current_cluster*	Cluster number
5	$Cluster_i$	$Cluster_i$
6	D_{i_q}	q_{th} fractal dimension of $Cluster_i$
7	$Cluster_{ij}$	Merge of $Cluster_i$ and $Cluster_j$
8	D_{ij_q}	q_{th} fractal dimension of $Cluster_{ij}$
9	G_i	Cluster set for merging
10	ε_q	Threshold value of q_{th} fractal dimension for grouping
11	δ_q	Threshold value of q_{th} fractal dimension for merging

2.2 FCHO算法描述

分形维数是反映数据分布复杂程度的一个重要特征参数,随着数据集结构形态的不同而变化,在一定程度上定量地描述了数据集内部结构的复杂性,并且同一数据集的分形维数不会因嵌入空间维数及嵌入位置不同而发生显著变化.表 2 列出了图 1 中的聚类所对应的分形维数 $D_{i_q}(q=-10,2,10)$,图 4 为合成数据集初始聚类结果对应的 q–D_q 曲线,容易看出,位于四角的交叉矩形框和中心交叉矩形框所对应的分形维数基本接近,而位于 4 个象限内的 4 个四分之一圆环所对应的分形维数基本接近.考虑到多重分形维数的性质,位于同一数据空间的聚类如果具有比较接近的多重分形维数,并且合并之后聚类所对应的分形维数与原聚类对应的分形维数相比变化较小,则根据分形维数的性质可知合并后的聚类内部结构特征与合并前的聚类内部结构特征相似,相应地,这两个聚类从相似性角度而言应该合并.

Table 2 D2-20K data set clusters and their fractal dimension
表 2 D2-20K 数据集聚类及其对应的分形维数

Cluster ID	C_{10}	C_2	C_{10}	Cluster description
C_1	1.461 57	1.274 07	1.159 0	Centre cross square, $\|C_1\|$=4000
C_2	1.401 9	1.298 6	1.215 3	Low left cross square, $\|C_2\|$=2000
C_3	1.411 7	1.297 8	1.210 2	Low right cross square, $\|C_3\|$=2000
C_4	1.446 2	1.297 0	1.198 0	Upper left cross square, $\|C_4\|$=2000
C_5	1.420 1	1.292 6	1.164 8	Upper right cross square, $\|C_5\|$=2000
C_6	2.677 4	1.836 5	1.652 2	Low left quarter cirque, $\|C_6\|$=2000
C_7	2.653 6	1.828 1	1.658 6	Low right quarter cirque, $\|C_7\|$=2000
C_8	2.727 0	1.827 6	1.621 7	Upper left quarter cirque, $\|C_8\|$=2000
C_9	2.697 5	1.822 2	1.643 6	Upper right quarter cirque, $\|C_9\|$=2000

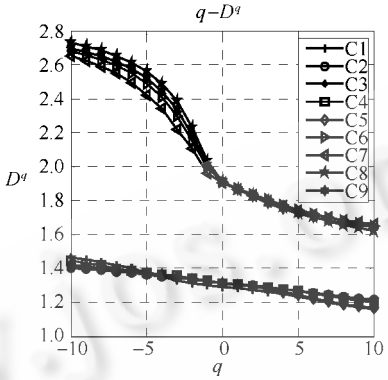

Fig.4 q–D_q plots of the initial cluster results

图 4 初始聚类结果对应 q–D_q 曲线

FCHO 算法主要分为初始聚类和优化聚类两个过程.其中初始聚类过程可以利用当前任何一种聚类算法,为了与优化聚类过程共用数据结构,我们设计了基于网格与密度的初始聚类过程.下面主要说明聚类优化的处理过程.

具体处理过程如下:

依据式(3)和式(4)确定聚类优化及合并的阈值参数 ε_q 和 δ_q :

$$\varepsilon_q = \frac{\left|\max(D_{i_q}) - \min(D_{i_q})\right|}{\lfloor Current\_cluster/2 \rfloor} \tag{3}$$

$$\delta_q = 2 \cdot \varepsilon_q \tag{4}$$

其中, $\min(D_{i_q})$ 及 $\max(D_{i_q})$ 分别对应当前所有聚类的 q 次分形维数的最小和最大值.

计算两两聚类分形维数的差值,按照式(5)生成可能合并聚类集合.对至少包含两个聚类的集合,选择集合中差值最小的两个聚类,进行可能的聚类合并操作,利用式(6)确认相关的合并工作,并从集合中删除合并的聚类,重复以上操作直至可能合并聚类集合为空.

$$|D_{i_q} - D_{j_q}| \le \varepsilon_q, q = -10, 2, 10 \tag{5}$$

$$|D_{ij_q} - D_{i_q}| \le \delta_q \ and \ |D_{ij_q} - D_{j_q}| \le \delta_q, q = 2, 10 \tag{6}$$

其中, D_{i_q}, D_{j_q} 和 D_{ij_q} 分别对应聚类 $Cluster_i$, $Cluster_j$ 和 $Cluster_{ij}$ 的 q 次分形维数, ε_q 和 δ_q 参照式(2)与式(3)定义.

初始化聚类步骤见算法 1,一遍聚类层次优化过程见算法 2,其中构造可能合并聚类集合的过程见算法 3,聚类合并操作见算法 4.

算法 1. Initialization step.

Input: normalized data set

Output: initial cluster results

Begin

construct the bottom grid structure

set cluster number as 1

for all the grids unlabelled

set true label

recursively check the neighbors

increase the cluster number by 1

End

算法 2. Optimization step.

Input: grid structure of the initial clusters

Output: grid structure of the merged result

Begin

optimize the initial clusters

calculate D_{i_q} and ascending sort

initialize ε_q and δ_q

generate the possibly merge cluster sets

call merge operation

End

算法 3. Generate the possibly merge cluster sets.

Input: ascending sequence of clusters G, ε_q and δ_q

Output: sets of possibly merge clusters

Begin

$\Delta D_{ij_q} = (D_{j_q} - D_{i_q}), i = 1,2,...,Current\_cluster - 1, j = i + 1, q = -10,2,10$

1st step

for the maximal $\Delta D_{ij_{-10}}$ satisfied with $\Delta D_{ij_{-10}} > \delta_{-10}$, group G into two sets G_{i_1} and G_{i_2}

repeat the group operation on G_{i_1} and G_{i_2}, get the cluster groups which satisfied with $D_{i_{-10}}$

2nd step

for the result groups of the first step, execute the same operation according to $D_{i_{10}}$, and get the cluster groups which satisfied with $D_{i_{-10}}$ and $D_{i_{10}}$

3rd step

for the result groups of the second step, execute the same operation according to D_{i_2}, and get the cluster groups which satisfied with $D_{i_{-10}}$, $D_{i_{10}}$ and D_{i_2}

End

算法 4. Merge operation.

Input: groups of clusters for merging G_i

Output: merging result of G_i

Begin

$G_i' = \varnothing$, flag=false;

while $|G_i| \geq 2$ and not flag

flag=true;

for all $\{(D_i, D_j) | D_i \in G_i, D_j \in G_i\}$

$Cluster_{ij} = Cluster_i \bigcup Cluster_j$

if $|D_{ij_q} - D_{i_q}| \leq \sigma_q$ *and* $|D_{ij_q} - D_{j_q}| \leq \sigma_q$

flag=false;

$$\Delta D_{ij} = \sum_{q=2,10} \frac{\left|D_{ij_q} - D_{i_q}\right| + \left|D_{ij_q} - D_{j_q}\right|}{2}$$

if ΔD_{ij} is minimum

$Cluster_i = Cluster_{ij}$, $D_i = D_{ij}$

$G_i = G_i - D_i$, $G_i = G_i - D_j$, $G_i' = G_i' + D_i$

if $G_i \neq \varnothing$ $G_i' = G_i' \bigcup G_i$

End

从算法 2 可以看出,聚类层次合并算法耗费时间取决于聚类对应的分形维数计算工作,对于合并操作最多需要计算 $\sum_{i=1}^{Max\_group} C_{c_i}^2$ 次分形维数,其中 $c_i = |G_i|$,而 $\sum_{i=1}^{Max\_group} c_i = Current\_cluster$, Max_group 为当前最大的可能合并聚类分组数目,对于实际应用而言,初始化得到的聚类数目比较少,并且分形维数计算过程共享网格聚类算法所采用的多层网格结构,聚类层次合并工作完全基于主存进行.此外,根据最后得到的 G_i' 中所包含的合并后聚类结果,递推执行聚类优化算法,最终可以得到初始聚类的合并家族树,有利于用户对初始聚类进行后续处理工作.

3 实验与分析

基于图 1 对应的 D2-20K 数据集与聚类操作经常使用的标准数据集 t8-8K 及 t7-10K 从算法有效性和算法执行效率两个方面进行了相关的实验工作,算法代码利用 Delphi 编写,操作系统为 Windows Server 2003,硬件环境为 PIII 1.7G,512MB RAM.

3.1 算法的有效性测试

D2-20K 数据集实验针对图 1 所示的初始聚类结果进行聚类的优化工作.其中,初始聚类对应的分形维数见表 2,阈值参数 $\varepsilon_{-10} \approx 0.33$, $\delta_{-10} \approx 0.66$; $\varepsilon_2 \approx 0.14$, $\delta_2 \approx 0.28$; $\varepsilon_{10} \approx 0.12$, $\delta_{10} \approx 0.24$.算法第 1 遍执行得到两个可能合并聚类集合分别为 G_1={C_1,C_2,C_3,C_4,C_5}和 G_6={C_6,C_7,C_8,C_9},聚类合并结果为:C_1 与 C_3 合并,C_2 与 C_4 合并,C_6 与 C_8 合并,C_7 与 C_9 合并.为对比起见,我们把任意两个聚类合并之后的分形维数以表 3 列出.

Table 3 Fractal dimensions and their differences of the clusters and the merged clusters of D2-20K data set

表 3 D2-20K 数据集初始聚类对应的分形维数及两两合并以后之差

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
	1.461 6	0.069 9	0.054 7	0.083 8	0.063 4	0.527 8	0.526 8	0.508 1	0.516 4
C_1	1.274 0	0.028 2	0.026 4	0.031 3	0.028 6	0.281 2	0.277 0	0.276 8	0.274 1
	1.159 0	0.041 7	0.028 3	0.052 4	0.034 8	0.246 6	0.249 8	0.231 4	0.242 3
	1.653 9	1.401 9	0.059 1	0.078 5	0.092 0	0.487 4	0.486 4	0.467 7	0.548 6
C_2	1.314 5	1.298 6	0.031 6	0.031 4	0.043 8	0.268 9	0.264 7	0.264 5	0.261 8
	1.228 8	1.215 3	0.027 4	0.047 1	0.048 1	0.218 5	0.221 7	0.203 2	0.286
	1.616 4	1.590 2	1.411 7	0.077 5	0.066 8	0.490 4	0.489 3	0.557 2	0.478 9
C_3	1.312 3	1.329 9	1.297 8	0.040 8	0.032 0	0.269 3	0.265 1	0.264 9	0.262 2
	1.212 9	1.240 2	1.210 2	0.036 6	0.034 8	0.221 0	0.224 2	0.292 3	0.216 7
	1.681 7	1.608 1	1.553 0	1.446 2	0.079 6	0.496 9	0.548 5	0.477 2	0.485 4
C_4	1.316 9	1.329 2	1.256 6	1.297 0	0.033 8	0.269 8	0.265 5	0.265 3	0.262 6
	1.230 9	1.253 7	1.167 4	1.198 0	0.045 8	0.227 1	0.283 0	0.211 9	0.222 8
	1.644 1	1.558 7	1.600 5	1.600 7	1.420 1	0.590 5	0.514 6	0.495 9	0.504 2
C_5	1.311 9	1.251 7	1.327 2	1.328 6	1.292 6	0.272 0	0.267 7	0.267 5	0.264 8
	1.196 7	1.141 9	1.222 3	1.227 2	1.164 8	0.318 5	0.246 9	0.228 4	0.239 4
	2.154 1	2.442	2.286 5	2.212 6	2.086 4	2.677 4	0.161 8	0.143 1	0.599 8
C_6	1.473 6	1.629 4	1.441 1	1.441 7	1.327 1	1.836 5	0.080 0	0.079 7	0.314 4
	1.382 7	1.398 3	1.222 1	1.219 1	1.090 0	1.652 2	0.081 8	0.063 4	0.285 4
	2.171 1	2.275 4	2.423 8	2.099 1	2.223 5	2.492 8	2.653 6	0.569 0	0.115 2
C_7	1.473 3	1.440 6	1.630 9	1.332 5	1.436 8	1.752 3	1.828 1	0.306 3	0.078 1
	1.381 7	1.224 4	1.399 1	1.145 3	1.181 7	1.573 6	1.658 6	0.262 7	0.037 1
	2.157 5	2.285 4	2.102 7	2.514 6	2.267 1	2.519 6	2.624 4	2.727 0	0.096 8
C_8	1.471 8	1.440 0	1.330 6	1.629 0	1.432 7	1.753 4	1.521 5	1.827 6	0.074 5
	1.380 1	1.235 8	1.123 6	1.400 5	1.181 12	1.605 5	1.377 5	1.621 7	0.022 3
	2.165 2	2.090 0	2.180 7	2.185 9	2.382 9	2.671 2	2.514 8	2.484 4	2.697 5
C_9	1.466 5	1.329 8	1.434 6	1.436 5	1.622 6	1.515 0	1.747 0	1.750 4	1.822 2
	1.370 9	1.142 6	1.218 5	1.226 5	1.359 8	1.362 5	1.614 0	1.610 4	1.643 6

表 3 中主对角线及左下角各单元分别对应聚类各自的分形维数及两两合并之后的分形维数(第 1 行对应

D_{-10},第 2 行对应 D_2,第 3 行对应 D_{10}),而右上角单元分别对应聚类合并之后与合并之前的分形维数差值(其中第 2 行元素为相应的 D_2 差值的均值,第 3 行元素为相应的 D_{10} 差值的均值,而第 1 行元素为第 2 行元素与第 3 行元素之和).图 5 给出了相关聚类合并前后的 q–Dq 曲线的对比情况,第 3.3 节的图 12A 为相应的实验结果.

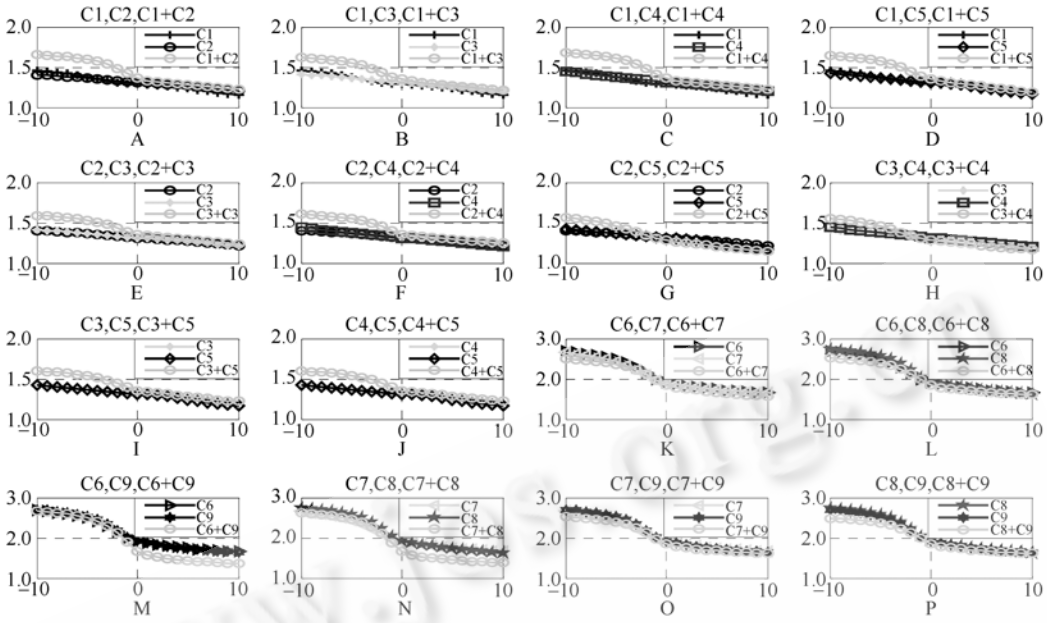

Fig.5 Post-Processing results of the clusters of D2-20K data set

图 5 D2-20K 数据集聚类后续处理结果

标准数据集验证工作我们选择 t8-8K 和 t7-10K 进行.其中,t7-10K 数据集包含 10 000 个数据点,分别对应 9 个形状各异、密度变化较小的聚类及一些噪声数据点;而 t8-8K 数据集包含 8 000 格数据点,分别对应 8 个不同形状、不同大小、不同密度的非球形聚类及大量的噪声数据点.我们的主要目的是对初始聚类结果做进一步的处理,并期望发现未知的、有趣的模式.

图 6A 为 t7-10K 数据集聚类初始结果,图 6B 为对应的 q–Dq 曲线.聚类相应的分形维数及合并之后的变化列于表 4.聚类优化算法对应阈值参数 $\varepsilon_{-10}\approx 0.18$, $\varepsilon_2\approx 0.08$, $\varepsilon_{10}\approx 0.08$, $\delta_{-10}\approx 0.36$, $\delta_2\approx 0.16$, $\delta_{10}\approx 0.16$.算法第 1 遍执行得到可能合并的聚类集合分别为: G_1={C_1,C_4,C_7,C_9},G_2={C_2,C_3,C_6},G_5={C_5,C_8};对应的合并结果为 C_1 与 C_4 合并,C_7 与 C_9 合并,C_5 与 C_8 合并,C_2 与 C_6 合并,其中每个集合中所包含聚类两两合并之后分形维数变化情况如图 6C～图 6K 所示.最终的实验结果见第 3.3 节的图 13A 所示.

图 7A 为 t8-8K 数据集的初始聚类结果,图 7B 为对应的 q–D_q 曲线.聚类相应的分形维数及合并之后的变化列于表 5.聚类优化算法参数: $\varepsilon_{-10}\approx 0.16$, $\varepsilon_2\approx 0.07$, $\varepsilon_{10}\approx 0.07$, $\delta_{-10}\approx 0.32$, $\delta_2\approx 0.14$, $\delta_{10}\approx 0.14$.算法第 1 遍执行的可能分组为: $G_1=\{C_1,C_4,C_5,C_6\}$, $G_2=\{C_2,C_8\}$, $G_3=\{C_3\}$, $G_7=\{C_7\}$,得到的聚类合并结果为 C_1 与 C_5 合并, C_4 与 C_6 合并.值得注意的是,从图 7I 明显可以看出 C_2 与 C_8 对应的多重分形维数曲线非常接近,如果仅仅考虑这一点,我们很容易得到 C_2 与 C_8 也能合并的结论,但合并之后聚类对应的分形维数与合并之前的分形维数变化程度非常剧烈,因此,我们可以断定在第 1 遍算法执行时不应该合并 C_2 与 C_8.最终得到的实验结果如第 3.3 节的图 14A 所示.

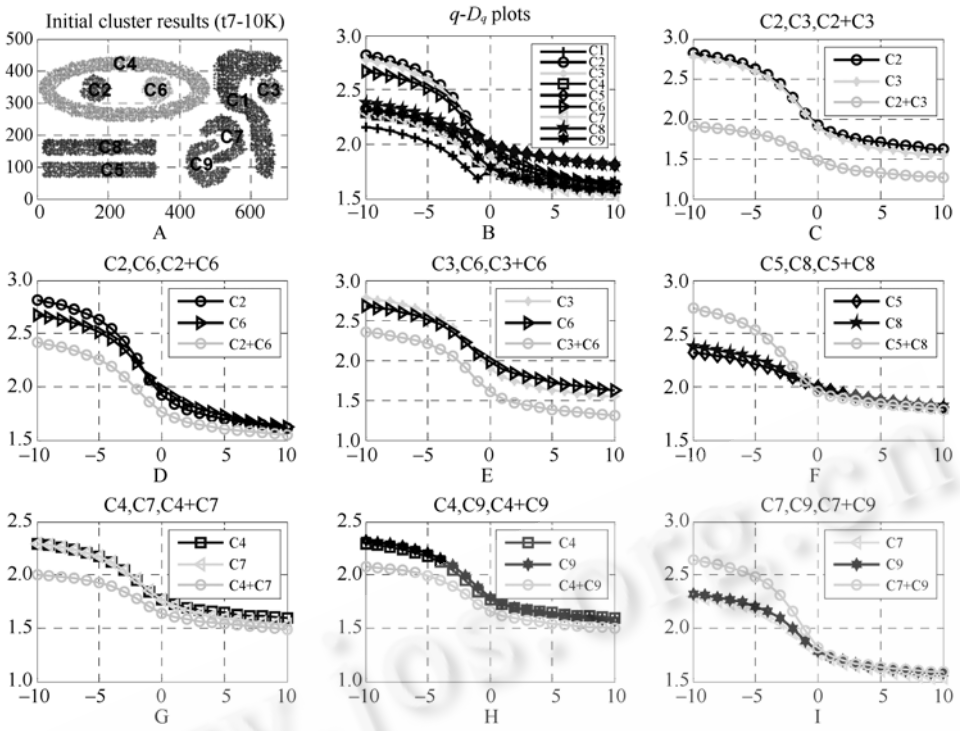

Fig.6 Post-Processing results of the clusters of t7-10K data set

图 6 t7-10K 数据集聚类后续处理结果

Table 4 Fractal dimensions and their differences of the clusters and the merged clusters of t7-10K data set

表 4 t7-10K 数据集聚类对应分形维数及两两合并以后之差

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
	2.157 7	0.392 3	0.087 1	0.115 4	0.542 1	0.281 8	0.089 3	0.547 7	0.090 4
C_1	1.719 7	0.223 8	0.027 6	0.069 4	0.289 5	0.180 6	0.024 7	0.288 9	0.047 4
	1.662 5	0.168 5	0.059 5	0.046 1	0.252 6	0.101 2	0.064 7	0.258 8	0.043 0
	1.939 4	2.817 1	0.696 0	0.083 7	0.742 1	0.216 3	0.645 5	0.599 0	0.751 1
C_2	1.530 1	1.788 1	0.381 4	0.047 1	0.403 9	0.145 6	0.363 0	0.323 9	0.408 1
	1.470 2	1.614 8	0.314 6	0.036 7	0.338 2	0.070 7	0.282 5	0.275 1	0.343 0
	1.846 6	1.903 8	2.777 8	0.111 4	0.759 7	0.596 4	0.277 0	0.677 1	0.505 7
C_3	1.755 7	1.380 8	1.736 4	0.085 2	0.426 1	0.322 4	0.174 1	0.368 1	0.281 7
	1.661 4	1.264 6	1.543 5	0.026 2	0.333 6	0.274 0	0.102 9	0.308 9	0.224 0
	2.011 1	2.328 1	2.448 0	2.289 9	0.338 1	0.097 0	0.174 5	0.241 3	0.191 5
C_4	1.637 5	1.734 6	1.629 9	1.722 6	0.193 0	0.074 4	0.098 4	0.133 3	0.102 8
	1.583 2	1.642 0	1.544 8	1.595 9	0.145 1	0.022 6	0.076 0	0.108 0	0.088 6
	2.008 1	1.839 9	1.657 6	2.268 0	2.321 7	0.789 4	0.430 2	0.066 0	0.351 2
C_5	1.538 2	1.458 0	1.409 9	1.621 8	1.935 6	0.447 5	0.257 0	0.043 5	0.186 4
	1.482 1	1.372 6	1.341 6	1.556 2	1.806 8	0.341 8	0.173 2	0.022 5	0.164 8
	2.162 6	2.411 0	2.352 8	2.361 6	2.132 4	2.672 0	0.459 3	0.658 7	0.491 1
C_6	1.600 6	1.669 8	1.467 1	1.733 3	1.441 7	1.842 7	0.263 6	0.369 8	0.277 0
	1.544 2	1.550 8	1.311 9	1.634 6	1.375 7	1.628 2	0.195 7	0.288 9	0.214 1
	2.401 6	1.769 3	1.704 3	2.000 9	2.112 9	1.793 9	2.294 3	0.310 4	0.047 3
C_7	1.719 8	1.366 3	1.529 3	1.583 8	1.546 1	1.493 0	1.670 6	0.171 0	0.014 8
	1.623 1	1.291 5	1.435 4	1.488 6	1.496 8	1.385 0	1.533 2	0.139 4	0.032 5
	2.017 7	1.958 7	1.694 9	2.426 7	2.735 4	2.111 2	2.147 8	2.376 1	0.312 9
C_8	1.532 4	1.531 6	1.461 5	1.675 1	1.885 8	1.513 0	1.625 7	1.922 8	0.180 1
	1.478 5	1.438 3	1.368 8	1.603 4	1.786 9	1.431 1	1.544 2	1.812 0	0.132 8
	2.360 2	2.051 1	1.981 3	2.081 0	1.924 9	2.115 9	2.644 0	2.196 6	2.320 0
C_9	1.659 8	1.333 3	1.433 8	1.591 5	1.628 7	1.491 7	1.697 4	1.628 7	1.694 7
	1.583 0	1.252 6	1.336 1	1.497 6	1.526 9	1.388 3	1.587 4	1.561 5	1.576 6

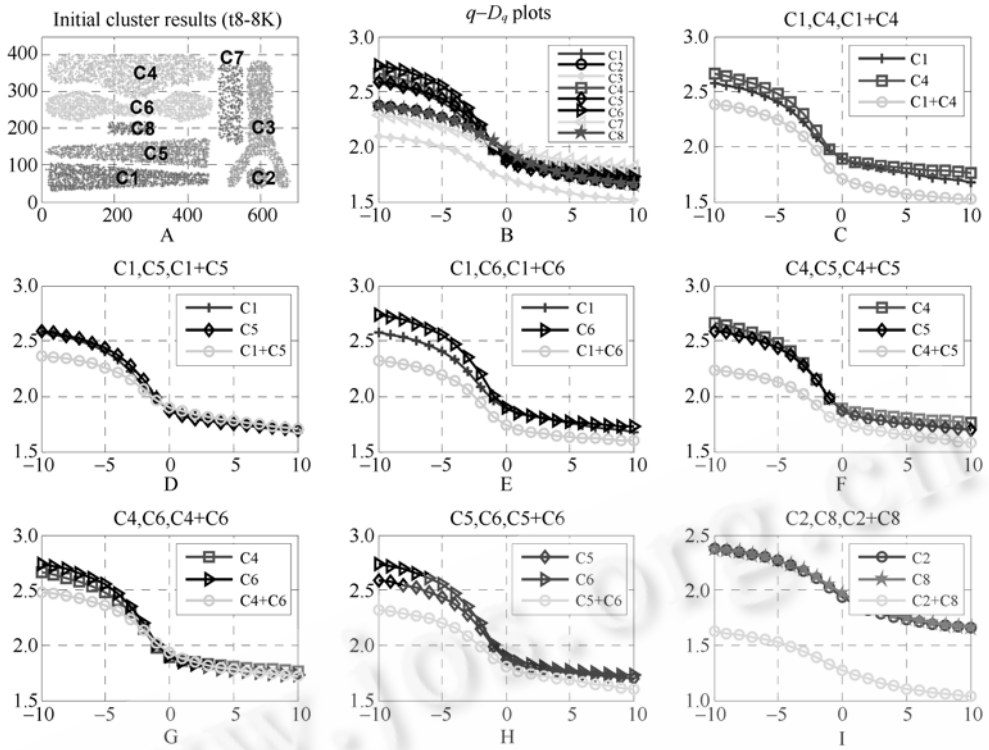

Fig.7 Post-Processing results of the clusters of t8-8K data set

图 7 t8-8K 数据集聚类后续处理结果

Table 5 Fractal dimensions and their differences of the clusters and the merged clusters of t8-8K data set

表 5 t8-8K 数据集聚类对应分形维数及两两合并以后之差

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
	2.579 4	0.058 3	0.301 7	0.398 3	0.049 0	0.250 2	0.734 8	0.300 8
C_1	1.822 2	0.045 7	0.178 5	0.200 3	0.028 0	0.143 7	0.389 8	0.186 2
	1.680 2	0.012 6	0.123 2	0.198 1	0.021 0	0.106 6	0.345 0	0.114 5
	2.501 0	2.373 5	0.157 7	0.635 4	0.033 2	0.579 9	0.680 2	1.264 9
C_2	1.776 9	1.823 1	0.086 9	0.331 4	0.012 7	0.332 3	0.313 0	0.643 5
	1.663 7	1.655 1	0.070 8	0.304 0	0.020 4	0.247 6	0.367 1	0.621 4
	2.078 6	2.126 0	2.098 7	0.306 7	0.342 1	0.288 9	0.296 0	0.455 8
C_3	1.557 2	1.692 3	1.649 2	0.155 1	0.194 7	0.160 5	0.144 4	0.274 9
	1.473 7	1.558 7	1.513 6	0.151 7	0.147 4	0.128 4	0.151 6	0.180 8
	2.388 1	2.064 5	2.207 5	2.662 3	0.264 3	0.033 9	0.384 0	0.224 3
C_4	1.632 7	1.502 0	1.591 4	1.843 8	0.117 3	0.011 8	0.181 4	0.124 5
	1.524 0	1.405 5	1.487 0	1.763 9	0.147 0	0.022 1	0.202 5	0.099 9
	2.364 4	2.3735	2.411 0	2.232 3	2.590 0	0.182 5	0.636 0	0.109 3
C_5	1.837 9	1.823 1	1.528 7	1.703 4	1.797 6	0.069 8	0.327 4	0.091 1
	1.709 1	1.655 1	1.457 3	1.582 9	1.696 0	0.112 7	0.308 5	0.018 2
	2.318 6	2.031 6	1.839 3	2.479 3	2.322 3	2.741 1	0.586 6	0.167 3
C_6	1.679 7	1.491 4	1.576 3	1.845 9	1.741 2	1.824 5	0.297 5	0.085 5
	1.596 6	1.443 0	1.491 4	1.722 9	1.598 3	1.726 1	0.289 0	0.081 8
	2.063 5	2.243 0	2.175 6	2.196 8	2.131 6	2.284 0	2.285 6	1.079 1
C_7	1.490 3	1.567 5	1.721 5	1.709 4	1.540 4	1.583 7	1.938 0	0.521 6
	1.403 5	1.368 8	1.592 7	1.587 8	1.447 8	1.482 4	1.816 8	0.557 5
	2.021 4	1.622 1	1.765 1	2.067 0	2.526 5	2.260 9	2.328 4	2.370 0
C_8	1.650 1	1.193 3	1.474 9	1.722 7	1.732 9	1.752 0	1.372 6	1.850 5
	1.555 4	1.036 0	1.405 8	1.611 9	1.676 2	1.611 0	1.180 6	1.659 6

3.2 算法效率分析

3.2.1 时间复杂度分析

FCHO 算法主要目的在于对能基本反映数据集分布情况的初始聚类结果进行后续处理,进而发现数据集自然聚集情况.因此,我们着重分析聚类后续处理过程的时间复杂度,其中,计算聚类结果的分形维数是影响聚类后续处理过程效率的关键因素.

根据式(1)可知,计算分形维数需要得到不同层次网格所包含数据点的统计信息,因而需要对不同层次的网格进行扫描工作,在固定网格对应的维坐标顺序的情况下,为快速统计具有相同坐标值网格所包含的数据点数目,我们对网格队列基于坐标值进行快速排序,快速排序的时间复杂度为 $O(N\cdot\log N)$,其中,N 为参与排序的数据点数目.对 FCHO 算法而言,相应地,N 为每层网格的数目(最大为数据点数目),分形维数计算时间复杂度为 $O(Level\cdot N\cdot\log N)$,其中,$Level$ 为多层嵌套网格的层次数目,通常情况下 $Level << N$.因此,在仅存储底层网格结构情况下,基于 Z-ordering 技术计算分形维数的时间复杂度为 $O(N\cdot\log N)$.

算法 2 中,FCHO 算法执行一遍的时间主要在于需要计算分形维数 $\sum_{i=1}^{\mathrm{Max\_}group} C_{c_i}^2$ 次,即对每一个可能合并聚类集合计算其成员两两合并后所对应的分形维数,其中 $c_i=|G_i|$,而 $\sum_{i=1}^{\mathrm{Max\_}group} c_i = Current\_cluster$,$Current\_cluster$ 为当前聚类的数目.

对初始聚类的最佳可能合并分组与最坏可能合并分组的执行情况分别分析如下:

最坏可能合并分组对应为:一个聚类独立构成一个集合,其余聚类形成一个可能合并的聚类集合.此时,对于仅包含一个聚类的分组,不需要计算分形维数,而对于包含其余聚类的分组,其分形维数的计算次数为

$$\frac{(Current\_cluster-1)\cdot(Current\_cluster-2)}{2}.$$

此外,为了得到可能合并聚类分组,需要计算每个聚类对应的分形维数,次数为 $Current\_cluster$,因而,总的分形维数计算次数为

$$Current\_cluster+\frac{(Current\_cluster-1)\cdot(Current\_cluster-2)}{2},$$

即 $O(Current\_cluster^2)$.

最佳可能合并分组对应为:得到 $\left\lceil\frac{Current\_cluster}{2}\right\rceil$ 个可能合并的聚类集合,其中每个可能合并聚类集合包含初始聚类数目最多为 2.此时,所对应的分形维数计算次数为 $\left\lceil\frac{Current\_cluster}{2}\right\rceil$,加上计算每个初始聚类结果的分形维数次数 $Current\_cluster$,总的分形维数计算次数为 $\frac{3}{2}\cdot Current\_cluster$,即 $O(Current\_cluster)$.

最坏情况下,FCHO 算法一遍执行对应的分形维数计算次数为 $\frac{Current\_cluster^2}{2}+Current\_cluster$,相应地,创建聚类合并家族树需要计算 $\left(\frac{Current\_cluster^2}{2}+Current\_cluster\right)\cdot\log_2^{Current\_cluster}$ 次分形维数.

考虑计算分形维数对应的时间复杂度 $O(N\cdot\log N)$,则 FCHO 算法总的时间复杂度为: $O(Current\_cluster^2\cdot N\cdot\log N)$,由于 $Current\_cluster << N$,因而总体时间复杂度为 $O(N\cdot\log N)$.

图 8 为数据集规模与处理时间关系,其中,规模比例按照如下规则设置:比例系数为 1 对应原始的聚类结果,比例系数小于 1 表示对初始聚类结果按照对应比例进行采样,比例系数大于 1 表示对初始聚类结果按照对应比例减去 1 进行采样,然后增加采样得到的数据点拷贝,之所以这样做的目的在于维持数据集的分形维数基本不变,从而保证对所有比例规模的初始聚类结果其后续处理过程中分形维数的计算次数恒定.此外,将 3 个数据集

分别嵌入到不同维度的超平面中,所得到实验结果如图 9 所示.

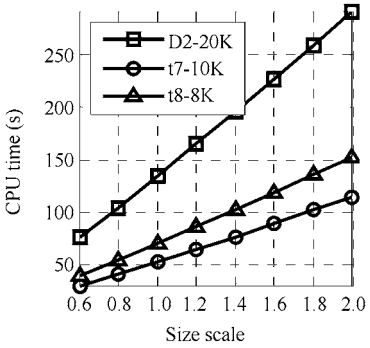

Fig.8 Data set scale vs. CPU time

图 8 规模与运行时间

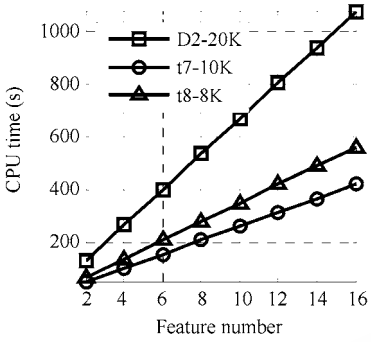

Fig.9 Data set feature number vs. CPU time

图 9 属性数目与运行时间

分析算法 2 的聚类合并处理过程,分形维数的计算次数是影响 FCHO 算法执行时间的关键因素.为了尽量减少分形维数的计算次数,FCHO 算法首先对初始聚类基于分形维数的接近程度进行分组,从而得到可能合并的聚类集合,这一过程可以显著减少分形维数的计算次数.如果不对初始聚类结果按照分形维数接近程度进行分组,则 3 个数据集对应的分形维数计算次数分别为(D2-20K:66;t7-10K:65;t8-8K:56),而分组以后对应的分形维数计算次数分别为(D2-20K:29;t7-10K:26;t8-8K:27).由此可见,虽然 t8-8K 的数据点数目小于 t7-10K 的数据点数目,但整个合并过程中 t8-8K 对应的分形维数计算次数为 27,大于 t7-10K 对应的分形维数计算次数 26.因此,在图 8 和图 9 中 t8-8K 数据集的运行时间超过 t7-10K 数据集的相应运行时间,这充分体现了分形维数计算次数对算法执行时间的影响程度.

3.2.2 空间复杂度分析

如第 2.1 节所述,初始聚类过程采用基于网格和密度的聚类技术,则聚类后续处理过程可以与初始聚类过程共享同一个嵌套多层网格结构,所增加的存储空间仅为每个初始聚类结果对应的网格结构的拷贝.如果无法与初始聚类结果共享存储空间,则聚类后续处理过程需要构建相应的底层网格结构,由于只需要存储非空的网格结构信息,所对应的存储空间的最大需求正比于数据点数目.假定数据集不包含异常数据点和稀疏数据点(异常点和稀疏点在初始聚类过程中已经排除),初始聚类结果共包含数据点数目为 N,则初始网格结构对应的存储空间正比于 N.在聚类后续过程中通过计算合并聚类所对应的分形维数来判断合并操作是否合法,需要保存每个聚类的拷贝,所需存储空间同样正比于 N.因此,算法总的存储空间需求与数据点数目呈线性关系,即 $2\cdot N$,相应地,算法的空间复杂度为 $O(N)$,合并算法空间需求与底层网格数目呈线性关系.数据集规模及属性数目与存储空间的关系如图 10 及图 11 所示.

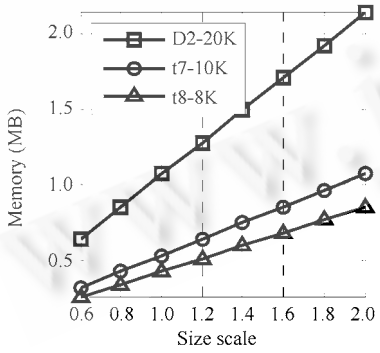

Fig.10 Data set scale vs. memory

图 10 规模与存储空间

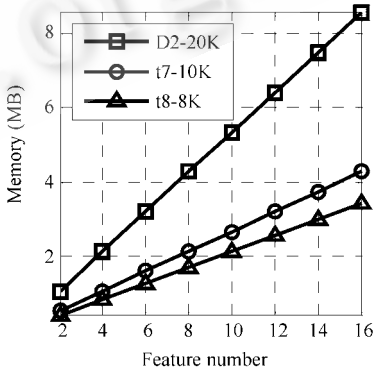

Fig.11 Data set feature number vs. memory

图 11 属性数目与存储空间

3.3 与传统层次聚类方法的对比分析

根据第 1.2 节,采用传统意义上的网格与密度聚类技术无法有效解决聚类的后续处理问题,基于划分的聚类技术如 K-Means 也无法成功应对这一问题.当前的层次聚类技术如 BIRCH 无法应用于非球形聚类情况;CURE 算法选择两个聚类代表点对之间的最小距离作为聚类合并依据,忽略了两个不同聚类中对象的聚集互连性等信息;ROCK 技术则仅仅强调对象间的互连性,完全不考虑对象之间近似度的信息;Chameleon 算法虽然综合考虑聚类之间的互连性与聚类内部特征,从而确定最相似的聚类进行合并操作,但需要用户确定如最近邻居数目、结果聚类数目及稀疏图划分平衡限制因子等参数.从本质上分析,层次聚类操作中通常使用各种距离如欧式距离作为相似性度量的主要依据,虽然进行初始聚类操作比较有效,但无法应用于相互之间差异较大的聚类的后续处理问题.

图 12～图 14 分别给出了 FCHO 算法与基于距离度量机制的层次聚类算法的对比实验结果,其中,子图 A 对应 FCHO 算法的运行结果,子图 B 对应单连接实验结果,子图 C 对应全连接实验结果.图中横线连接参与合并的聚类,竖线的位置表示聚类合并的先后顺序,位置相同的竖线表示聚类合并操作在同一遍聚类优化算法中进行.

需要指出的是,在图 13A 中,聚类 C_3 与 C_{7+9} 的合并操作看来比较奇怪,对照图 6A,按照人们的习惯性思维,应该将 C_3 与 C_{2+6} 合并,通过分析合并过程我们得知,聚类合并算法执行时确实将 C_2, C_3 及 C_6 划分为一个分组,但在合并过程中判断 C_2 和 C_6 合并是满足条件的,这里的合并操作非常鲜明地体现出在形状一致的情况下优先考虑聚类之间距离的特点,而一旦 C_2 和 C_6 合并完成,在 C_3 与 C_{7+9} 的合并过程中,算法又显示出在聚类间距离基本一致的情况下优先考虑形状的特性(此时, C_{7+9} 比 C_{2+6} 更加类似于 C_3),从而体现算法综合考虑距离与形状的特点.此外,对于图 14A 而言,当聚类合并中间结果为 $C_{1+5+4+6}$, C_{2+3}, C_7 与 C_8 时,利用原有的聚类分组算法我们得到 4 个可能合并分组,其中每个分组各包含一个聚类,按照判断聚类合并的标准此时已经不存在可能的合并聚类了,为了生成聚类合并的家族树,我们在对聚类分组时减少所使用的分形维数(仅利用 $D_{i_{-10}}$ 和 $D_{i_{10}}$,也可以仅利用其中之一),从而得到如图 14A 所示的结果.

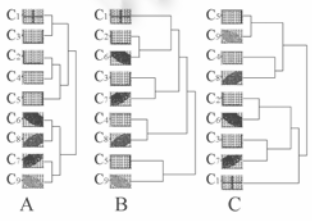

Fig.12 Result of D2-20K

图 12 D2-20K 数据集运行结果

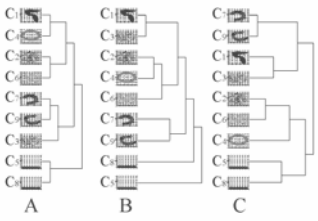

Fig.13 Result of t7-10K

图 13 t7-10K 数据集运行结果

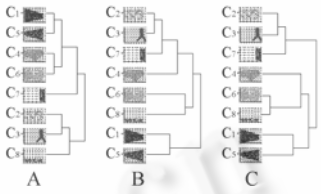

Fig.14 Result of t8-8K

图 14 t8-8K 数据集运行结果

对于图 12 而言,容易看出单连接与全连接的聚类合并结果无法精确反映数据集的自然聚集情况,而 FCHO 算法的执行结果则可以揭示数据集中隐藏的模式.分析图 13 及图 14 我们可以发现,FCHO 算法的执行结果与全连接算法的运行结果更为接近,而与单连接算法的运行结果相差较大.图 14 中, C_1 与 C_5 及 C_2 与 C_3 的合并操作在所有运行结果中都发生,并且合并层次也基本类似.图 13 中, C_7 与 C_9 及 C_5 与 C_8 的合并操作也存在类似情况,尤其是对于 FCHO 算法和全连接算法而言, C_7 与 C_9 及 C_5 与 C_8 的合并操作发生在相同的层次或非常接近的层次.而大多数情况下基于全连接的层次聚类算法比单连接层次聚类技术更倾向于发现有效的聚类层次结构,这也间接证明了 FCHO 算法的有效性.

4 结论及展望

FCHO 算法利用多重分形技术,综合考虑聚类形状、聚类间距离及聚类整体数据分布等特征进行聚类的层次合并操作,并最终生成反映数据自然聚集状态的聚类家族树,从而支持用户对聚类结果做进一步的分析工作.

算法与基于网格和密度的聚类算法共享存储结构,因而不需要增加大量额外的存储空间,同时具有参数自适应及易于实现的特点.在合成数据集及实际数据集上的实验表明 FCHO 算法不失为一种简单、高效的聚类后续处理方法.

References:

[1] Han JW, Kamber M,Wrote; Fan M, Meng XF, *et al*., Trans. Data Mining: Concepts and Techniques. Beijing: China Machine Press, 2001. 223−253 (in Chinese).

[2] Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Computing Surveys, 1999,31(3):264−323.

[3] Kotsiants SB, Pintelas PE. Recent advances in clustering: A brief survey. WSEAS Trans. on Information Science and Applications, 2004,1(1):73−81.

[4] Macqueen J. Some methods for classification and analysis of multivariate observations. In: LeCam L, Neyman J, eds. Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability. 1967. 281−297.

[5] Huang ZX. Extensions to the k-Means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 1998,2(3):283−304.

[6] Ng RT, Han JW. Efficient and effective clustering methods for spatial data mining. In: Bocca J, Jarke M, Zaniolo C. eds. Proc. of the 20th VLDB Conf. 1994. 144−155.

[7] Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. In: Jagadish HV, Mumick IS, eds. Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data. Quebec: ACM Press, 1996. 103−114.

[8] Guha S, Rastogi R, Shim K. CURE: an efficient clustering algorithm for large databases. In: Haas LM, Tiwary A, eds. Proc. of the 1998 ACM SIGMOD Int'l Conf. on Management of Data. Seattle: ACM Press, 1998. 73−84.

[9] Guha S, Rastogi R, Shim K. ROCK: A robust clustering algorithm for categorical attributes. Information Systems, 2000,25(5): 345−366.

[10] Karypis G, Han EH, Kumar V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer, 1999,32(8):68−75.

[11] Ester M, Kriegel H-P, Sander J, Xu XW. A density-based algorithm for discovering clusters in large spatial databases with noises. In: Simoudis E, Han JW, Fayyad UM, eds. Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining (KDD'96). AAAI Press, 1996. 226−231.

[12] Ankerst M, Breunig MM, Kriegel H-P, Sander J. OPTICS: ordering points to identify the clustering structure. In: Delis A, Faloutsos C, Ghandeharizadeh S, eds. Proc. of the 1999 ACM SIGMOD Int'l Conf. on Management of Data. Philadelphia: ACM Press, 1999. 49−60.

[13] Wang W, Yang J, Muntz R. STING: a statistical information grid approach to spatial data mining. In: Jarke M, Carey MJ, Dittrich KR, *et al*., eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases. Athens: Morgan Kaufmann Publishers, 1997. 186−195.

[14] Sheikholeslami G, Chatterjee S, Zhang AD. WaveCluster: a multi-resolution clustering approach for very large spatial databases. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases. New York: Morgan Kaufmann Publishers, 1998. 428−438.

[15] Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In: Haas LM, Tiwary A, eds. Proc. of the 1998 ACM SIGMOD Int'l Conf. on Management of Data. Seattle: ACM Press, 1998. 94−105.

[16] Lauritzen SL. The EM algorithm for graphical association models with missing data. Computational Statistics and Data Analysis, 1995,19(2):191−201.

[17] Cheeseman P, Stutz J. Baysian classification (autoclass): Theory and results. In: Fayyad UM, Piatetsky-Shapiro G, Smith P, Uthurusamy R, eds. Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, 1996. 153−180.

[18] Faloutsos C, Seeger B, Traina A, Traina Jr C. Spatial join selectivity using power laws. In: Proc. of the 2000 ACM SIGMOD international Conf. on Management of Data. Dallas: ACM Press, 2000. 177−188.

[19] Belussi A, Faloutsos C. Estimating the selectivity of spatial queries using the 'Correlation' fractal dimension. In: Proc. of the 21st Intl. Conf. on Very Large Data Bases (VLDB). 1995. 299−310.

[20] Faloutsos C, Kamel I. Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension. In: Proc. of the 13th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Minneapolis, Minnesota: ACM Press, 1994. 4−13.

[21] Wu MJ. Multifractal entropy and multifractal dimension spectrum. Acta Electronic Sinica, 1993,21(10):7−13 (in Chinese with English abstract).

[22] Qian WN, Gong XQ, Zhou AY. Clustering in very large databases based on distance and density. Journal of Computer Science and Technology, 2003,18(1):67−76.

[23] Ma S, Wang TJ, TG SW, Yang DQ, Gao J. A fast clustering algorithm based on reference and density. Journal of Software, 2003,14(6):1089−1095 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1089.htm

[24] Ren YG, Yu G. Clustering for multi-dimensional data and its visualization. Chinese Journal of Computers, 2005,28(11):1861−1865 (in Chinese with English abstract).

[25] Yue SH, Wang ZY. Bisecting grid-based clustering approach and its validity. Journal of Computer Research and Development, 2005,42(9):1505−1510 (in Chinese with English abstract).

[26] Barbará D, Chen P. Using the fractal dimension to cluster datasets. In: Proc. of the 6th ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining (KDD-2000). ACM Press, 2000. 260−264.

[27] Yan GH, Li ZH, Yuan L. The practical method of fractal dimensionality reduction based on Z-Ordering technique. In: Li X, Zaiane OR, Li ZH, eds. Proc. of the 2nd Int'l Conf. on Advanced Data Mining and Applications. Berlin: Springer-Verlag, 2006. 542−549.

附中文参考文献:

[1] Han JW, Kamber M,著.范明,孟小峰,等,译.数据挖掘:概念与技术.北京:机械工业出版社,2001.223−253.

[21] 吴敏金.多重分形熵与多重分形谱.电子学报,1993,21(10):7−13.

[23] 马帅,王腾蛟,唐世渭,杨东青,高军.一种基于参考点和密度的快速聚类算法.软件学报,2003,14(6):1089−1095. http://www.jos.org.cn/1000-9825/14/1089.htm

[24] 任永功,于戈.一种多维数据的聚类算法及其可视化研究.计算机学报,2005,28(11):1861−1865.

[25] 岳士弘,王正友.二分网格聚类方法及有效性.计算机研究与发展,2005,42(9):1505−1510.

闫光辉(1970－),男,河南商丘人,博士生,副教授,主要研究领域为数据挖掘,分形技术及应用.

党建武(1963－),男,博士,教授,博士生导师,主要研究领域为神经网络理论与应用,智能信息处理.

李战怀(1961－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术,网络存储.