文章编号: 1001-0920(2006)09-1011-04

基于扩展和网格的多密度聚类算法

邱保志, 沈钧毅
(西安交通大学 电子与信息工程学院, 西安 710049)

摘 要: 提出了网格密度可达的聚类概念和边界处理技术, 并在此基础上提出一种基于扩展的多密度网格聚类算法. 该算法使用网格技术提高聚类的速度, 使用边界处理技术提高聚类的精度, 每次聚类均从最高的密度单元开始逐步向周围扩展形成聚类. 实验结果表明, 该算法能有效地对多密度数据集和均匀密度数据集进行聚类, 具有聚类精度高等优点.

关键词: 聚类算法; 多密度数据集; 边界处理; 扩展聚类; 网格聚类
中图分类号: TP371 **文献标识码**: A

Grid-based and Extend-based Clustering Algorithm for Multi-density

QIU Bao-zhi, SHEN Jun-yi
(School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China. Correspondent: QIU Bao-zhi, E-mail: bzqiu@zzu.edu.cn)

Abstract: The clustering concept of grid density reachability and boundary point extraction technique are proposed. Based on that, a grid-based and extend-based clustering algorithm for multi-density data sets (GECM) is developed. In GECM, grid-based method is used to improve speed of clustering, and boundary point extraction technique is used to improve quality of clustering. A clustering starts with the highest density cell and then extends to its neighbors to form a cluster. Experiment results show that the proposed algorithm is efficient and effective for multi-density and uniformity density data sets with noises.

Key words: Clustering algorithm; Multi-density data set; Boundary point extraction; Extend-based clustering; Gird-based clustering

1 引 言

聚类是指将一组数据点划分为有意义的组, 它已经广泛应用于模式识别、信息检索、机器学习、数据挖掘等领域[1]. 当给定的数据集包含不同形状、不同大小、不同密度并含有噪声时, 如何发现聚类是一个亟待解决的问题.

目前已经提出多种聚类算法, 可分为: 划分方法、层次方法、基于密度的方法、基于网格的方法和基于模型的方法. 其中基于网格的方法在聚类过程中将网格中的点作为一个整体处理, 而不是考虑单元中每一个点, 基于这一特性, 该方法在所有的聚类方法中效率最高. 其优点是聚类的结果与输入数据的顺序无关, 算法的时间复杂度是数据点个数的线性函数, 速度快, 可扩展性好, 能识别不同形状的聚类. 典型的基于网格的算法有STING, Wavecluster, CLIQUE, 近几年又提出了GDILC[2], SCI[3]和移动网格聚类[4].

但基于网格的聚类算法存在下面的问题: 1) 用户输入的参数对聚类结果有很大的影响, 例如, 每一维上分段的个数和密度阈值的选择都会对聚类结果产生影响; 2) 基于网格的聚类算法不能用于对高维空间聚类; 3) 传统基于网格的算法只处理高密度单

收稿日期: 2005-09-08; **修回日期**: 2005-12-27.
基金项目: 国家自然科学基金项目(60173058).
作者简介: 邱保志(1964—), 男, 河南上蔡人, 副教授, 博士生, 从事数据挖掘的研究; 沈钧毅(1939—), 男, 江苏常熟人, 教授, 博士生导师, 从事数据挖掘、分布式系统等研究.

元, 低密度单元中的点作为孤立点被丢弃, 一旦聚类的边界落入低密度单元, 就会降低聚类精度, 可能造成小聚类的丢失; 4) 基于网格的算法不能用于对多密度数据集进行聚类. 对前两个问题已经有相应的研究成果, 而对后两个问题的研究较少. 传统的网格聚类算法(如STING, Wavecluster, CLIQUE 等)均没有涉及到聚类边界的处理和对多密度数据集的聚类. SCI 提出了根据网格的结构和聚类的内聚力识别聚类边界点的方法, 即将非空稀疏单元中的点归类于离它最近的高密度邻居单元中, 虽然SCI 的聚类精度高于 k-平均, CLIQUE, Cure 等传统算法的聚类精度[3], 但该方法产生的聚类结果中吸收了较多孤立点. 本文着重研究后两个问题.

现有的很多算法(如DBSCAN 等)仅对高密度区域进行聚类, 只注重发现数据集中不同形状、不同大小的聚类, 忽视了高密度区域周围低密度区域中的点形成的聚类, 所以这些算法不能发现与高密度聚类相邻的低密度聚类, 从而不能对多密度数据集进行聚类; 另外, 聚类过程中没有考虑到单个点形成的聚类和相似孤立点所形成的聚类.

Chamelenon[5]可以对不同密度的数据集进行聚类, 但聚类精度不高; 最新提出的多密度聚类算法SNN[6]用代表点和核心点处理噪声和建立聚类, 能够发现相对均匀的区域, 其性能高于 k-means, DBSCAN, CURE, Chamelenon 等算法. 但其算法的时间复杂度为 $O(n^2)$, 对一些多密度数据集聚类的结果精度也不高, 如图1(a)和(c)所示.

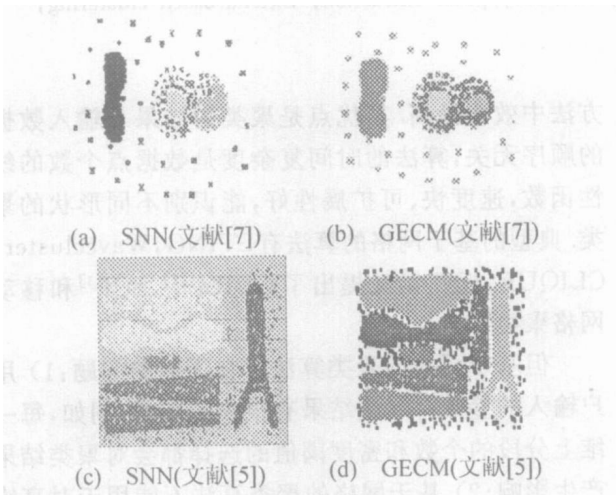

图1 SNN 与GECM 聚类结果比较

本文提出的GECM 算法使用网格聚类技术提高聚类的速度, 使用限制性 k-近邻来提高聚类的精度. 根据聚类边界均有一个密度明显的跳变的原则, 每次聚类都从未聚类的最高密度单元开始逐步向外围扩展得到一个聚类, 从而达到对多密度数据集的聚类. 其特点是只对数据集进行一遍读取, 能够发现任意形状、任意大小的聚类, 能够对多密度和均匀密度的数据集进行正确地聚类, 时间复杂度低, 聚类精度高, 能发现单点聚类和小聚类.

2 基于扩展的子空间聚类

针对现有的基于网格聚类算法都不能有效用于对多密度数据集聚类和聚类精度不高的事实, 首先提出了网格密度可达的概念和网格聚类的概念, 提出了使用限制性 k-近邻进行聚类边界处理技术, 最后给出了利用广度优先搜索技术的网格聚类算法.

2.1 定 义

给定一个 d-维数据集 D, 其属性 $(A_1, A_2, \cdots, A_d)$ 都是有界的, 不妨设第 i 维上的值在区间 $[l_i, h_i)$ 中, $i = 1, 2, \cdots, d$, 则 $S = [l_1, h_1) \times [l_2, h_2) \times \cdots \times [l_d, h_d)$ 即为 d-维数据空间. 将每一维分成 m 个长度相等且不相交的区间段, 每个区间都是左闭右开的等长区间, 这样将数据空间分割为 m^d 个网格单元. 网格单元在第 i 维上的长度为 $\delta_i = (h_i - l_i)/m$, 第 i 维上的第 j 个区间段可由

$$I_{ij} = [l_i + (j-1)\delta_i, l_i + j\delta_i], \quad j = 1, 2, \cdots, m$$

得出.

一个网格单元的邻居是指与该单元有相邻边界的单元或有相邻点的单元. 每个单元的邻居数为 3^{d-1}(处于数据空间边界的单元除外)[3].

定义 1[3] 一个网格单元的密度是指该网格单元中数据点的个数.

现有基于网格的聚类方法区分高密度单元和低密度单元, 聚类只对高密度单元聚类. 为了能识别不同密度、不同大小的聚类, 本文不区分高密度单元与低密度单元, 并用 $\mathrm{den}(C)$ 表示网格单元 C 的密度.

定义 2 网格单元 C_1 是 C 的邻居单元且都非空, 如果 $\mathrm{den}(C_1)/\mathrm{den}(C) > \mathrm{DDT}$, 称 C_1 直接密度可达 C, 其中DDT 是预先定义的密度递减阈值.

密度阈值DDT 由用户给出, 用它指定聚类中相邻的网格单元应该满足的条件, 在聚类过程中, 用它寻找聚类的主要轮廓、判断聚类的边界, 即网格密度在哪些网格单元出现跳变.

定义 3 如果 C 未被聚类的邻居单元都是到 C 直接密度可达的, 则称 C 是核心单元; 否则, 称 C 为边界单元.

定义 4 如果存在一个序列 $C_1, C_2, \cdots, C_p$ 满足 $C_1 = C'$, $C_p = C$, 且 C_i 到 C_{i+1} 是直接密度可达的, 则称 C' 到 C 是密度可达.

定义 5 一个聚类 C 是数据集中的一个非空子集, 满足: 1) $C = C_1 \cup C_2 \cup \cdots \cup C_p$, C_i 是网格单元;

2) 如果 $\mathrm{den}(C_r) = \max\{\mathrm{den}(C_i) \mid i = 1, 2, \cdots, p\}$, 那么任意 C_i 到 C_r 是密度可达的

即在一个聚类中有一个最高密度单元, 其他的网格单元到该最高密度单元都是密度可达的, 且聚类 C 是这些单元中点的并集

2.2　聚类边界点的提取技术

边界单元和核心单元形成聚类的骨架, 而边界点充实该骨架. 有时聚类的边界点可能落入聚类结果网格单元以外的网格单元中, 这就需要将聚类的边界点从这些单元中提取出来, 以提高聚类的精度.

假设未被聚类的网格单元 $\{\mathrm{CL}_i \mid i = 1, 2, \cdots, q, q \leq 3^{d-1}\}$ 是边界单元 CH 的邻居, 且 CL_i 到 CH 不是直接密度可达的. 那么, CH 所在的聚类中的边界点有可能落入 CL_i 中. 对每个 CL_i, 选取点对 (x, y), 满足 $d(x, y) = \min\{d(x', y') \mid x' \in \mathrm{CL}_i, y' \in \mathrm{CH}\}$ 和 $\{x \in \mathrm{CL}_i, y \in \mathrm{CH}\}$, 将 KNN 方法限制在 $\{x\} \cup \mathrm{CH}$ 集合上, 选取 x 的最近邻 y 和 y 的最近邻 z, 如果不等式(1) 成立, 则将 x 合并到 CH 中; 然后用同样的方法考察 $\mathrm{CL}_i - \{x\}$ 与 $\mathrm{CH} \cup \{x\}$. 一旦某个 x 不能归并于 CH 中, 则 CL_i 中的其他点也不可能满足不等式(1), 对该 CL_i 网格单元的边界处理结束, 然后处理 CL_{i+1} 网格单元(即 $i = i + 1$), 直到 $i > q$ 为止.

$$\frac{\sum_{y \in \mathrm{KNN}_x} d(x, y)}{\sum_{z \in \mathrm{KNN}_y} d(y, z)} \Big/ \frac{\sum_{z \in \mathrm{KNN}_y} d(y, z)}{\sum_{zz \in \mathrm{KNN}_z} d(z, zz)} < \epsilon \qquad (1)$$

其中: KNN_x 表示 x 在 CH 上的 k- 近邻的集合, $d(\cdot)$ 表示距离度量(本文使用欧几里德距离). $\sum_{y \in \mathrm{KNN}_x} d(x, y)$ 的大小反映了 x 点带方向的密度信息, 该值越大, 表示 x 点周围的密度越低; 反之密度越高.

将 x 点的密度与 CH 中点的密度进行比较, 只有二者的密度相近时, 才能说明 x 与 CH 中的点属于同一类; ϵ 是控制聚类边界延伸的重要参数, 为大于 1 的常数, 其取值越大, 聚类边界点越稀疏; 反之, 聚类的边界点的密度和聚类内部点的密度越一致, 所以 ϵ 的作用是控制聚类的延伸程度.

由于采用的是限制性 k- 近邻, k 的取值不能太大或太小. k 值太大, 会使式(1) 左面的值接近 1, 从而不能有效地提取聚类的边界点; k 值太小, 式(1) 左侧的值会很大, 可能丢失聚类的边界点. 所以 k 的取值应大于 1 且小于 $\mathrm{den}(\mathrm{CH})$, 为了避免取值的麻烦, 根据试验, $k = \mathrm{int}(\sqrt{\mathrm{den}(\mathrm{CH})}) + 1$ 是一个理想的取值, 所以算法中 k 不再作为聚类的参数.

2.3　多密度聚类算法

算法思想基于下面的事实: 聚类中点的密度高于聚类外部的密度和边界点的密度, 聚类边界的密度到聚类外部的密度有明显的跳变. 每次聚类都从最高密度的网格单元开始逐步向外扩展, 遇到边界单元时进行边界处理. 当一次聚类过程结束后, 从剩余的未聚类单元中找一个最高密度的单元, 继续聚类, 直到所有的网格单元都被聚类为止.

聚类算法为 GECM, 输入分别为 m, DDT, ϵ, 步骤如下:

Step1　划分: 将数据空间的每一维划分为 m 个不相交、等长的单元, 从而将整个数据空间划分为 m^d 个网格单元.

Step2　密度计算: 1) 将数据集 D 中的每个点映射到相应的网格单元中; 2) 计算每个网格单元的密度.

Step3　聚类: 如果存在未被聚类的网格单元, 执行下面 4 个步骤: 1) 从未被聚类的网格单元中选取一个密度最大的网格单元 C; 2) 从单元 C 开始按广度优先搜索, 求所有到 C 密度可达的单元 C_i, $i = 1, 2, \cdots, p$; 3) 如果 C_i 是边界网格单元, 进行边界处理; 4) C_i, $i = 1, 2, \cdots, p$ 的并集是一个聚类, 并输出该聚类.

需要注意的是上述算法中划分是指将 d 维数据集 D 的每一维分割为 m 个等长不相交的区间, 这样数据空间被划分为 m^d 个网格单元; 密度计算是指对数据集进行一遍扫描, 将数据点映射到相应的网格单元中, 同时计算每个网格单元的密度, 其中密度为 0 的网格单元标记为已聚类; 聚类是指将数据点划分为核心网格单元中的点、边界单元中的点和落入边界单元之外聚类的边界点. 前两种点通过测试网格密度可达而得到, 第 3 种要通过边界处理将落入边界网格单元之外的点提取出来. 根据定义 5 得到如下定理:

定理 1　C 是数据空间中最高密度单元, 所有到 C 密度可达的单元的并集是一个聚类.

定理 2　C 是数据空间中未聚类网格单元中密度最高的单元, 所有到 C 密度可达的未聚类单元的并集是一个聚类.

根据定理 1 和定理 2, 每次先从数据空间中寻找一个密度最大、未被聚类的网格单元 C, 以 C 为本次聚类的起点开始聚类. 方法是按照广度优先搜索原则, 寻找到网格单元 C 直接密度可达且未被聚类的网格单元 $\{C_i \mid i = 1, 2, \cdots, p\}$, 然后再对每个到 C 直接密度可达的网格单元继续进行广度优先搜索, 直到所有到 C 密度可达的网格单元都被搜索到为止. 在这个过程中, 如果遇到了边界网格单元, 就按照 2.2 节中的方法进行边界处理. 这样, 所有到 C 密度

可达的网格单元中点的并集就是一个聚类, 显然, 聚类的结果中包含了聚类的边界点; 重复 Step3 直到所有的非空单元都处理完毕为止

3 实验分析

实验使用的环境: CPU 为 Pentium Ⅳ2. 4G, 内存为 256 M, 操作系统为 Windows 2000 professional, 算法的编写使用 Borland C + + builder6. 0.

3.1 算法的时间复杂度

算法 GECM 第 1 步的时间复杂度为 $O(m^d)$; 第 2 步的时间复杂度为 $O(N + m^d)$; 第 3 步的时间复杂度由 4 部分组成: 1) 所有选取最高密度网格单元的时间复杂度最坏为 $O(m^{2d})$; 2) 计算所有密度可达的网格单元的时间复杂度最多为 $O(3^d m^d)$; 3) 边界处理的时间复杂度与边界网格单元数、落入聚类之外的数据点个数、涉及的网格单元数有关, 最坏情况下, 所有边界处理的时间复杂度不超过 $O(3^d m^d)$; 4) 所有形成聚类的时间复杂度最多为 $O(m^d)$.

所以整个 GECM 聚类算法的时间复杂度为 $O(N + m^{2d})$, 其中: N 为数据集中点的个数, d 为数据空间的维度, m 为每一维上划分区间的个数, CH 为边界单元. 由复杂度可以看出, 算法的时间复杂度是数据集大小 N 的线性函数, 该算法适用于对大数据集聚类. GECM 算法的时间复杂度低于 SNN 时间复杂度 $O(N^2)$, 稍高于 SCI 的时间复杂度 $O(N + m^d)$.

3.2 聚类结果比较

图 1(a) 和(b) 中的多密度数据集[7] 含有 2 352 个点. 图 1(a) 是 SNN 算法的聚类结果, 参数设定如下: 近邻个数为 65, 共享个数为 51, 聚类的结果有 4 个, 其他标记为噪声. 图 2(b) 是 GECM 算法的聚类结果, 参数设定如下: $m = 50$, $\epsilon = 2.5$, DDT = 0.6, 聚类时间为 0.125 s, 聚类结果有 57 个聚类, 其中 4 个大聚类(点个数分别是 1 046, 629, 263, 189), 53 个小聚类(包括 3 个单点聚类, 这里小聚类中的点数小于 9, 在图 1(b) 中小聚类的点使用同一种颜色标记). 从图 1(a) 和(b) 可以看出, GECM 聚类的结果合理, 能正确地识别出“8”字, 而 SNN 不能准确地识别

图 2 SCI 与 GECM 聚类结果比较

图 1(c) 和(d) 中的数据集[5] 含有 22 083 个点. 图 1(c) 是 SNN 的聚类结果, 它发现了 7 个聚类, 且将噪声当作聚类的一部分; 图 1(d) 为 GECM 的聚类结果, 参数设定如下: $m = 100$, $\epsilon = 4$, DDT = 0.85, 运行时间为 1.282 s, 总聚类个数为 198 个, 其中有 8 个大聚类(其数据点个数分别为: 4 703, 3 577, 3 316, 3 310, 3 057, 1 532, 550, 411), 190 个小聚类(包括 6 个单点聚类, 这里小聚类的点数小于 56 个点, 图 1(d) 中, 小聚类用同一种颜色标记). 从图 1(c) 和(d) 可以看出, SNN 聚类结果不合理, 而 GECM 算法的聚类结果合理、精度高.

图 2 是文献[3] 中提供的均匀密度数据集(9 993 点), 图 2(a) 是原始数据集, 图 2(b) 是 GECM 的聚类结果, 使用的聚类参数为: $m = 100$, $\epsilon = 4$, DDT = 0.9, 运行时间为 0.469 s, 图 2(c) 是 SCI 的聚类结果. 从图 2 可以看出, SCI 的聚类结果吸收了附近的噪声点, 而 GECM 的聚类结果没有吸收噪声, 这说明它的聚类精度高于 SCI.

对 30 个综合数据集进行测试(包括 CLIQUE, CURE, CHAMELEON, SCI, AUTOCLUST 中所使用的多密度和均匀密度数据集等), 结果表明, GECM 算法的聚类结果正确, 聚类精度高, 能发现小聚类和单点聚类, 运行速度快. 聚类精度高于传统的聚类算法和 SCI, SNN 等. 传统的聚类方法区分聚类点和孤立点/噪声, 孤立点和噪声不参与聚类而被舍弃. 事实上, 孤立点/噪声之间也存在一定的联系, 这些联系对数据分析也是有意义的, 所以本文中不定义孤立点/噪声, 它们也参与聚类过程, 这样可能形成单点聚类和小聚类. 虽然本算法中没有定义孤立点的概念, 但可以根据实际情况规定较小聚类中的点是孤立点.

4 结 语

本文提出的基于网格和扩展的多密度聚类概念是 DBSCAN 中相应概念的扩展, 提出的边界处理技术能准确地提取出聚类的边界点, 提高了基于网格技术的聚类精度. 研制的多密度聚类算法只对数据集进行一遍扫描, 算法的时间复杂度是输入数据大小的线性函数, 聚类的结果与输入数据的顺序无关; 能够对带噪声的多密度数据集聚类和均匀密度的数据集正确聚类, 发现任意形状、任意大小的聚类, 聚类结果精度高于传统的聚类算法和最新提出的 SNN, SCI 算法. 该算法的缺点是聚类结果对参数较为敏感.

(下转第 1019 页)

从图 3 可以发现机器人在这种情况下无法完成对目标物体的跟踪, 与前述实验比较分析可知, 出现这一情况的原因是忽略了大偏差对系统响应的影响 因此, 加入对大偏差估计的控制方法 响应曲线如图 4 所示, 误差曲线如图 5 所示

由图 4 和图 5 可以看出, 经过 8 次迭代系统即达到稳定, 一旦机器人末端执行器能跟踪目标物体的运动轨迹后, 就不再出现发散点 而在图 1 中, 系统达到收敛后还出现了 1 个发散点, 这说明小偏差情况下虽然对系统的影响可以忽略, 但并不是没有影响, 只是对系统的影响较小. 采用针对大偏差的控制方法, 可消除它对系统的影响

4　结　　语

本文提出了基于动态的方差最小化原理控制机器人, 采用动态的拟牛顿法估计图像雅克比矩阵可以将科研人员从标定摄像机和机器人模型的窠臼中解放出来 针对系统在个别时刻出现的不稳定现象, 采用RLS 算法可以改善输出响应 本文提出的对大偏差估计的控制方法可以较大地改善系统的输出响应, 增强了无标定视觉伺服控制理论的实用性 仿真实验证明本文提出的算法是有效的, 并取得了较好的效果

参考文献(References)

[1] Hutchinson S, Hager G, Corke P. A Tutorial Introduction to Visual Servo Control[J]. *IEEE Trans on Robotics and Automation*, 1996, 12(5): 651-670.

[2] Chaumette F, Malis E. 21/2D Visual Servoing: A Possible Solution to Improve Image-based and Position Based Visual Servoings[A]. *IEEE Int Conf on Robotics and Automation*[C]. San Francisco, 2000: 630-635.

[3] Malis E. Visual Servoing Invarant to Changes in Camera-intrinsic Parameters [J]. *IEEE Trans on Robotics and Automation*, 2004, 20(1): 72-81.

[4] Hosada K, Asada M. Versatile Visual Servoing without Knowledge of Ture Jacobin [A]. *Proc IEEE/RSJ Int Conf Intell Robot System* [C]. New York, 1994: 186-191.

[5] Kim G W, Lee B H, Kim M S. Uncalibrated Visual Servoing Technique Using Large Residual [A]. *Proc IEEE Int Conf Robotics and Automation* [C]. Taipei, 2003: 3315-3320.

[6] Piepmeier J A. Experimental Results for Uncalibrated Eye-in-hand Visual Servoing [A]. *Proc IEEE Int Conf Robotics and Automation* [C]. Annapolis, 2003: 335-339.

[7] Xiang L J, Si B Y, Xue D Y. Model Independent Uncalibration Visual Servo Control [J]. *Robot*, 2003, 25(5): 424-427.

[8] Xie X M, Ding F. *Adaptive Control System* [M]. Beijing: Press of Tsinghua University, 2002: 62-78.

[9] Corke P. A Robotics Toolbox for Matlab [J]. *IEEE Robotics and Aumatation*, 1996, 3(1): 24-32.

(上接第 1014 页)

参考文献(References)

[1] Levent Ertöz, Michael Steinbach, Vipin Kumar. A New Shared Nearest Neighbor Clustering Algorithm and Its Applications[A]. *2nd SIAM Int'l Conf on Data Mining* (*SDM* 2002) [C]. SIAM Press, 2002: 105-115.

[2] Zhao Y C, Song Jjunde. GDILC: A Grid-based Density-isoline Clustering Algorithm [A]. *Proc of* 2001 *Int Conf on Info-tech and Info-net*[C]. Beijing: IEEE Press, 2001: 140-145.

[3] Hsu C M, Chen M S. Subspace Clustering of High Dimensional Spatial Data with Noises[A]. *Advances in Knowledge Discovery and Data Mining*: *8th Pacific-asia Conf PAKDD* 2004 [C]. Heidlberg: Springer, 2004: 31-40.

[4] Eden W M Ma, Tommy W S Chow. A New Shifting Grid Clustering Algorithm [J]. *Pattern Recognition*, 2004, 37(3): 503-514.

[5] Karypis G, Han E H, Kumar V. Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling[J]. *IEEE Computer*, 1999, 32(8): 68-75.

[6] Levent Ertöz, Michael Steinbach, Vipin Kumar. Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data[A]. *Proc of the 3rd SIAM Int Conf on Data Mining* (*SDM* 2003) [C]. San Francisco: SIAM Press, 2003: 1-12.

[7] Vladimir E C, Ickjai Lee. Autoclust: Automatic Clustering via Boundary Extraction for Mining Massive Point-data Sets [A]. *Proc of the 5th Int Conf on Geocomputation*[C]. Greenwich, 2000.

[8] Zhao Y C, Song M, Xie F, et al. Clustering Datasets Containing Clusters of Various Densities [J]. *J of Beijing University of Posts and Telecommunications*, 2003, 26(2): 42-47.