多代表点特征树与空间聚类算法*)

黄添强[1,2] 秦小麟[2] 王金栋[2]
(福建师范大学数学与计算机科学学院计算机科学系 福州 350007)[1]
(南京航空航天大学计算机科学与技术系 南京 210016)[2]

摘 要 空间数据具有海量、复杂、连续、空间自相关、存在缺损与误差等的特点,要求空间聚类算法具有高效率,能处理各种复杂形状的簇,聚类结果与数据空间分布顺序无关,并且对离群点是健壮的等性能,已有的算法难以同时满足要求。本文提出了一个适合处理海量复杂空间数据的数据结构-多代表点特征树。基于多代表点特征树提出了适合挖掘海量复杂空间数据聚类算法CAMFT,该算法利用多代表点特征树对海量的数据进行压缩,结合随机采样的方法进一步增强算法处理海量数据的能力;同时,多代表点特征树能够保存复杂形状的聚类特征,适合处理复杂空间数据。实验表明了算法CAMFT能够快速处理带有离群点的复杂形状聚类的空间数据,结果与对象空间分布顺序无关,并且效率优于已有的同类聚类算法BIRCH与CURE。

关键词 空间聚类,空间数据,多代表点特征树

Multi-representation Feature Tree and Spatial Clustering Algorithm

HUANG Tian-Qiang[1,2] QIN Xiao-Lin[2] WANG Jin-Dong[2]
(Department of Computer Science and Engineering, Fuzhou University, Fuzhou 350002)[1]
(Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)[2]

Abstract Spatial data have the features of largeness, complexity, continuity, spatial autocorrelation, missing data and error in spatial database. These characters require that a good spatial clustering algorithm must be high efficient, and should be able to detect clusters of complicated shapes, and the clusters found should be independent of the order in which the points in the space are examined, and should be not be impacted by outliers. The existed algorithms can not work well. Clustering algorithm based on multi-representation feature tree named CAMFT is proposed. A new data structure is firstly proposed to condense data, which drew the strongpoint from BIRCH algorithm and CURE algorithm, and then the algorithm that included the idea of random sampling is proposed to enhance the ability to detect very large data. As well as, the multi-representation feature tree can keep clusters of complicated shapes, so it can be used to detect spatial clusters. Experimental results show the algorithm can identify clusters of complicated shapes efficiently in large spatial database that have many outliers, and outperform BIRCH algorithm and CURE algorithm in efficiency.

Keywords Spatial clustering, Spatial data, Multi-representation feature tree

聚类是数据挖掘技术中应用最广泛与最活跃的技术之一。它已经在生物学、地理学、遥感学、药学、人类学、市场营销与经济学等许多领域广泛应用,它被用于植物分类、遥感分类、疾病分类、图像处理、模式识别和文本检索等。在空间数据中聚类技术也得到广泛应用,由于空间数据固有的特性,空间聚类技术与一般的面向统计数据的聚类技术不同,有特殊的要求。

空间数据是具有空间或位置分量的数据,一般具有空间与非空间属性。空间属性的表示可以通过特定的位置属性来实现,如地址、经纬度、芯片上某个相对位置,或者是隐含在基于位置的数据库划分中。对空间数据的查询可以通过空间运算符的查询来进行,如:"包含"、"相交"、"东"、"南"、"附近"等。由于空间数据本身固有的特点,空间数据常常使用特定的数据结构来存储,如四叉树、K-D树、R树等。空间数据由于它本身的数据环境决定了空间数据具有数据结构复杂,数据量大、连续、存在缺损与误差等特点。因此,面向空间数据挖掘的空间聚类算法应具有以下特点[1]:

(1) 算法必须具有高效率,能在大型多维数据库中高效地工作;

(2) 能够检测不同形状的簇;

(3) 找到的聚类与数据分析的顺序无关;

(4) 聚类算法对离群点是不敏感的。

有许多面向统计数据的聚类算法可以直接用于空间数据聚类,如k-means算法[2]、EM算法[3]、K-medoids[4]算法等,但是,这些算法的效率不高。它们是假设基于内存处理数据的算法,当数据量较大时,使用这些算法是不可行的。T. Zhang等提出的BIRCH算法[5]具有O(n)的高效率,目前是处理海量空间数据较理想的方法之一。但是,这个算法有2个缺点:第一是CF树的每个节由于大小限制只能包含有限数目的结点,这不利于表达用户所处理数据的自然聚类;第二是BIRCH用基于簇中心的统计量判断距离,并且用直径或半径来控制聚类的边界,如果簇不是球形的,它就不能很好地工作。对于第一个缺点,BIRCH在阶段3用一个全局或半全局的聚类算法进行改进,但是,第二个缺点极大地降低了

*)国家自然科学基金(No. 49971063),国家高技术研究发展计划(863)(No. 2001AA6330101-04),航空科学基金项目(02F52033),江苏省自然科学基金(No. BK2001045)。黄添强 博士,研究方向为数据挖掘技术;秦小麟 博士生导师,当前主要研究方向为安全数据库、空间数据库、时空数据库、信息安全等;王金栋 博士,研究方向为计算机应用技术。

BIRCH 算法的实用性。Cure 算法[6]的优点是可以聚类复杂多边形，算法要对所有数据划分后分别进行聚类，缺点是聚类的效率有待于提高。本章借鉴了这两种算法的优点，提出一种新的数据结构多代表点特征树，用多个代表点来表示簇，用多代表点特征树压缩数据集信息，然后利用多代表点特征树进行聚类。这种方法具有 BIRCH 的高效率，又克服了它的缺点，能够处理复杂形状的簇，同时对离群点也更加健壮。

1 相关工作

聚类算法研究非常活跃，针对不同的应用、数据类型与聚类目的，已有大量的聚类算法提出。聚类算法大体上可分为如下 9 类：

基于划分的方法。最著名与常用的划分算法是 k-mean[2]、k-medoids[4]与它们的变种。

基于层次的方法。它分为凝聚层次聚类与分裂层次聚类。凝聚层次聚类是自底向上的策略，首先将每一个对象作为一簇，然后合并它们，直到满足某个条件，著名的有 BIRCH[5]、CURE[6]、ROCK[7]等；分裂层次聚类正好相反，首先把所有的对象看作一簇，然后逐渐细分成越来越小的簇，直到达到某个终结条件为止，有 DIANA 与 MONA[4]等。

基于密度的方法。这类方法可以发现任意形状的簇。著名的有 DBSCAN[8]、OPTICS[9]、DENCLUE[10]等。国内也有人提出改进的基于密度的算法[11]。

基于网格的方法。这种方法采用一个多分辨率的网格数据结构将空间分成有限数目的单元，聚类操作在单元内进行。这种方法处理的时间独立于对象的数目，处理速度快。著名的有 STING[12]、WaveCluster[13]、CLIQUE[14]等。

基于模型的方法。这种方法为每个聚类假设一个数学模型，试图为数据查找合适的数学模型。主要有两类方法：统计的方法与神经网络方法。统计的方法包括 COBWEB[15]、LASSIT[16]与 AutoClass[17]等。神经网络方法包括 Rumelhart 和 Zipser 提出的竞争学习法[18]与 Kohonen 提出的 SOM[19, 20]等。

基于模糊集的方法。最著名的有 Höppner 等提出的 Fuzzy-means (FCM)[21]，Yager 和 Filev 提出的 mountain method (MM)[22]等，Yager 与 Filev 提出可以处理带有离群点的聚类算法 possibilistic-means clustering algorithm (PCM)，Davé 提出可以查找任意形状的模糊集聚类算法 fuzzy c-shells (FCS)[23]。

基于图理论的方法。有 Chameleon[24]，Delaunay triangulation graph (DTG)[25]、highly connected subgraphs (HCS)[26]、clustering identification via connectivity kernels (CLICK)[27]与 cluster affinity search technique (CAST)[28]等。

基于核函数的方法。国外有人提出 Kernel-K-means[29]，support vector clustering (SVC)[30]等算法，国内有人提出核聚类算法[31]。

以上算法中，可伸缩性的算法有 CLARA、CURE、CLARAN S、DBSCAN、DENCLUE、WaveCluster、BIRCH 等算法，CLARA 算法的时间复杂性是 $O(K(40+k))^2+K(n-K))$，CURE 算法的时间复杂性是 $O(n_{sample}{}^2\log n_{sample})$、CLARANS 的时间复杂性总体来说是 $O(n^2)$，DBSCAN 与 DENCLUE 是 $O(n\log n)$，BIRCH 与 WaveCluster 的时间复杂性是 $O(n)$。空间数据一般是海量的，基于主存的算法与计算复杂度高的算法都难以在空间数据中应用。

2 多代表点特征(MRF)树

BIRCH 算法提出的 CF-树[6]可以对聚类数据进行压缩，使得算法在大型数据库中取得高速度与可伸缩性，但它有上文提出的两大缺点。我们借鉴了它的优点，提出了带有多代表点的聚类特征树，这种数据结构具有捕捉复杂形状聚类的能力。

定义 1(代表点) 从一个簇中选择出的在形状上具有一定代表性的 n 个点，通过收缩因子 s，将这些点向质心收缩所得到的点称为簇的代表点。

定义 2(多代表点特征) 假设某个簇中包含 n 个 d 维的数据点，即数据对象$\{\vec{o_i}\}$，其中$\vec{o_i}=\{o_{i1},o_{i2},\cdots,o_{id}\}$，$i=1,2,\cdots,n$，定义为一个三元的数组如下：$MRF=(n,\vec{SL},Rep)$，其中 n 是簇中数据对象的数目，$\vec{SL}$是 n 个对象的线性和$(\sum_{i=1}^{n}\vec{o_i})$，$Rep$ 是簇的代表点集，用来代表这个簇。则这带有代表点的聚类特征三元组称为多代表点特征(Multi-Representation Feature，简称 MRF)。这个三元组记录了计算聚类的聚类的对象个数、代表点与质心，衡量簇之间距离通常采用的欧几里得距离和曼哈顿距离都可以通过代表点聚类特征求出，并且这些代表点概括了簇的几何或物理形状。

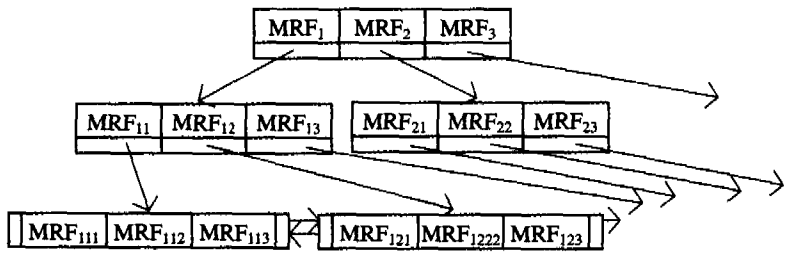

图 1 MRF-树示意图

定义 3(多代表点特征树，MRF-树) 一棵多代表点特征树(MRF-树)是带有分支因子与阈值参数的平衡树，它存储了层次聚类的簇的统计特征与几何(物理)特征，如图 1 所示。每一个内部结点(非叶子结点)的分支因子为 F，叶子结点的分支因子为 L(L 通常大于 F，是聚类个数 k 的函数)，每个结点阈值为 RT。分支因子 F 限定了每个内部结点最多含有的孩子的个数，即每个内部结点最多包含 F 个结点；阈值 RT 是指与距离最近的代表点的距离，它限定了存放在叶子结点中簇的大小。但与 BIRCH 的结点阈值 T 不同，BIRCH 中阈值 T 是指存放在叶子结点中簇的半径或直径，故每个簇的大小

是固定不变的，而 MRF 中的簇大小是可变的。当一个簇增大时，它的代表点将随着簇的边界扩大而移动，相应地，代表点或结点所能吸收的对象也增多，但代表点的个数不变。这样，结点所代表簇的大小没有严格限制的，并且可以有效地适应簇的各种形状，克服了引言中提到的 BIRCH 中 CF-树的缺点：

(1)这个定义使叶子结点所覆盖的对象数目可大可小，可以对应数据的自然聚类；而不像 BIRCH 中结点大小是固定的，用户一旦设定了阈值，结点的半径或直径就固定了，这不能对应数据各种大小不同的自然聚类。

(2)多代表点可以概括聚类的各种形状，有利于捕捉复杂形状的聚类。而不像 BIRCH 定义了固定半径(或直径)的球形的结点，这不利查找球形以外的各种聚类。

(3)分支因子与结点阈值影响了最终产生的树的大小。内部结点的形式为[MRF_i, $child_i$]，其中，$i=1,2,\cdots,F$；MRF_i代表第 i 个结点的子聚类，$child_i$ 是指向它的第 i 个子结点的指针。叶子结点的形式为[MRF_i, prev, next]，指针 prev, next 用来连接所有的结点，以提高查询速度。

2.1 MRF-树的插入操作

要建立一棵 MRF-树最基本的操作是插入操作，这种操作分两种：一是插入新数据点的操作，另一种是重建树时旧的条目重新插入新树的操作。插入新数据点的操作可以看作是含一个对象的条目的插入操作，下文中这两种操作都以插入条目论述。这里我们定义条目 A 的所有代表点(或对象 a)与另一条目 B 所有代表点距离的最小值，称为条目 A(或对象 a)与条目 B 的代表距离。插入算法的基本思想是：从根结点开始，由上向下查找代表距离最近的结点，达到叶子结点后，查找最近的条目 L_i，经测试阈值 RT，若 L_i 能吸收 E，则更新叶条目 L_i；若不可以，则插入一个新的条目到叶子结点。如果插入后这个结点分支因子不超过 L，则插入完成。否则，要进行分裂操作。

与结点或条目的最近距离是用代表点来计算，这种方式可以有效提高计算速度。代表距离形式如下：

$RDist(Node, MRF_i) = Min\{Dist(Node.Rep_m, MRF_i.Rep_n) | m=1,2,\cdots,c; n=1,2,\cdots,c\}$，其中 c 为代表点数目。

测试 RT 也是用 $RDist(Node, MRF_i)$ 进行计算。

插入条目 Node 到叶子结点后，要对路径上的每个内部结点的 MRF 信息更新。每插入一个点立即重新计算代表点，这种动态地计算代表点的方法，会使代表点更加具有轮廓的代表性，而且可以节省时间，提高效率。代表点计算算法如下：

算法 1 代表点的选择算法

```
Repre_Selection(C){
(1) M = the centroid of C; //C 为聚类代表点合并后的集合
(2)  for i=1 to c do{ // c 为用户定义代表点的数目
(3)   MaxRepDist=0;
(4)   foreach point p in C do{
(5)    if (i ==1)
(6)     MinRepDist=Dist(p,M) //第一个代表点是距离质心最远的点
(7)    Else
(8)     MinRepDist=Min{dist(p, q): q∈RepSet}; //第二个起的代表点是离已经选出的代表点最远的点。
(9)    If (MinRepDist)=MaxRepDist){
(10)    MaxRepDist=MinRepDist;
(11)    RepPoint=p
(12)   };
(13)  };
(14)  RepSet=RepSet∪{RepPoint}; //存放已选出的代表点。
(15) };
(16)Return RepSet;
};
```

若被插入结点能吸收插入条目道，则进行简单的结点信息更新，但若插入条目到被插入结点后，距离超过测试阈值时，要求进行分裂操作，就必须插入一个新的条目到被插入点的父结点来说明新出现的结点。若父结点有空间存放该条目，只需更新高一层父结点的结点信息 MRF 以反映更新；若父结点没有空间存放，这要对父结点进行分裂，这种更新可能一直达到传递到根结点。插入算法如下：

算法 2 MRF-树的插入算法

```
Node * Insert_Entry_MRFtree(tree * T, Node * CurNode, Entry E, Float RT){
//查找最近的结点
(1) Entry NewE, Node * NewNode;
(2) if (CurNode is leaf node){//若当前结点是叶子结点
(3)   CLeafE=Closest_Entry(CurNode, E) //找到最近的条目
(4)   if (Absorb(CLeafE, E) is true //若吸收后阈值小于 RT 则吸收
(5)    return Null
(6)   else{//若吸收后阈值大于 RT 则插入新条目
(7)    NewNode=Node_Split_For_Leaf_Node(CurNode, E); //插入新条目后返回值有两种，若为 Null 则不需分裂节点，否则分裂结点。
(8)    Reture NewNode;
(9)    }
(10)  }
(11)else{// //若当前结点不是叶子结点
(12)  CNodeE=Closest_Child(CurNode, E);//查找最近的条目(的结点)
(13)   NewNode = Insert_Entry_MRFtree(T, CNodeE, E, RT);//递归寻找最近的接点
(14)  if (NewNode==Null){//不需分裂
(15)   Update_MRF(CurNode,CLeafE,E); //不许插入新结点，只需更新吸收结点的路径上的 CRF。
(16)   return Null;
(17)   }
(18)  else{//需要分裂
(19)   Update_MRF(CurNode,CLeafE,E); //更新 CRF
(20)   NewEnt=Create_New_Entry(NewNode);//产生新结点
(21)   NewNode=Node_Split_For_Nonleaf_Node(CurNode,NewNode); //分裂结点
(22)   if (NewNode==Null){//不需继续分裂
(23)    Merge_Not_Split(CurNode)
(24)    return Null;
(25)    }
(26)   else{//需要继续分裂
(27)    if (CurNode== * T{//若父结点为根节点
(28)     * T=Create_New_Root(CurNode, NewNode);
(29)     return Null;
(30)     }
(31)    else return NewNode;
(32)   }
(33)  }
};
```

插入算法要注意两点：

(1) 阈值 RT 是指对象与最近的代表点的距离不超过 RT。

(2) 当结点需要分裂时，叶子结点的父结点的代表点由叶子结点决定，上一层的代表点由下一层决定。

2.2 MRF-树的重建

当一棵 MRF-树增长到一定的高度，达到内存允许的极限时，程序要对 MRF-树进行重建。重建树时提高条目的阈值 RT 即可降低树的高度。设原有的树 OldT 的阈值为 RT_o，高度为 h，结点数为 S_o。给定一个新的阈值 $RT_i \geqslant RT_o$，利用 OldT 的叶子结点重建一棵新树 NewT。

重建树的方法与 BIRCH 中的 CF 树重建的方法类似，但要注意的是计算对象与结点之间的距离要用对象与代表点之间的代表距离，计算代表点用上文提到的算法 1 进行。MRF-树重建方法如下：

①在新树中生成对应的当前路径：把旧树的旧当前路径中的结点 copy 到新树的当前路径。

②将旧树的旧当前路径中的结点插入到新树：对旧当前路径中的每一个叶结点插入到新树中，根据新的阈值 RT_i 进行测试，判断结点的插入位置。测试时，用代表点计算插入结点（或条目）与新树上结点（或条目）的距离，按上文提出的插入方法插入，并重新计算代表点，更新结点的信息。

③释放新旧树中的空间：旧树中一条路径上所有叶子结点处理完毕，即可把这条路径所有的结点及叶子删除。新树中因为某些结点向前推移，这些结点可能是空的，也要删除。

④处理下一条路径：若旧树中存在未处理的路径，按上述方法继续处理。

算法 3 重建 MRF-树的算法

```
Void ReBuildCFRTree ( tree Ti, tree Ti+1, double RTi, double RTi+1){
(1) Ti+1=Null;
(2) CurPath=PathOfTree(Ti, (0, 0, ……, 0));
(3) while (CurPath ! = null){
(4)   CopyNodesOfCurrentPathToNewTree(Ti+1, CurPath);
(5)   for Each Ent of CurPath do{
(6)     Flag=FitInClosestPath(Ti+1, RTi+1, ClosestPath, Ent);
(7)     if(Flag== False) && (ClosestPath<CurPath)
(8)       FitInPath(Ti+1, RTi+1, ClosestPath, Ent);
(9)     else
(10)      FitInPath(Ti+1,RTi+1,CurPath, Ent);
(11)    FreeCurrentPath(Ti, CurPath);
(12)    CurPath=NextPathOfTree(Ti, CurPath);
(13) }
(14) FreeEmptyNodes(Ti+1);
(15)}
}
```

3 基于 MRF-树的算法 CAMFT

理论与实验都证明[32, 33]，采用适当的随机采样技术，采样数据可以保证极高精度的聚类几何形状信息。这为算法取得复杂形状聚类的代表点提供了可能并可以较大地提高算法的效率。我们提出了基于 MRF-树的聚类算法（Clustering Algorithm based on multi-representation Feature Tree，简称 CAMFT）。首先在保证精度的基础上对数据库进行取样，然后在采样数据上构建 MRF-树；利用 MRF-树提供的信息，就可以根据用户要求的聚类数进行凝聚聚类；最后，依据各聚类代表点，扫描数据库，把未分配的对象分配到各聚类，完成对所有对象的聚类。

如图 2 所示，本算法由四部分组成：(1)随机取样；(2)建树；(3)叶结点聚类；(4)全局聚类。

3.1 随机取样

算法利用随机抽样技术，从数据库中抽取一部分数据进行聚类的预处理，不需要扫描整个数据库。一个好随机取样算法能在不到 2 秒的时间从几十万个数据点中取到几千个样本数据。显然，取样的时间相对数据聚类时间是微小的。算法采用文[32]提出的随机取样算法，只需遍历一次数据文件或数据库，且内存使用不随总数据量的大小而变化。根据切诺夫界[33]可以确定一个取样的下限。

图 2 CAMFT 算法的主要步骤

3.2 建树

首先要设置初始化的阈值 RT，当无法估计 RT 的值时，可以把它设为 0。一个合适的阈值对建树的效率是重要的。如果阈值初始化时设得太低，树需要反复重建，反之，树的高度太小，反映的信息不详细。一般地，对于没有经验的用户，在开始时把阈值设得比较低，然后动态地增加。增加阈值可以使用启发式方法，我们的方案是：把新的阈值设为旧树所有叶子结点中代表点的最近距离的平均值，这比 BIRCH 算法要简单，BIRCH 算法要计算所有结点的最近邻居对的局部近似值。

设定阈值后，开始扫描数据，将数据点按上述算法 2 插入到树中。若在扫描结束之前所需的内存超出限定，则提高阈值，按算法 3 重新建树，然后在扫描的断点继续建树。

在树的重建的过程中，密度较低的结点可以作为离群点处理。这样不但可以减少树的高度，同时可以减少离群点对聚类的影响。在磁盘上预留一定得空间，把密度较低的结点作为离群点写入磁盘。判断密度低的标准也是启发式的算法，我们的方法是把某个含有特别少对象的结点作为离群点。设某个结点对象数为 χ，已统计的所有结点对象数的均值 μ 与方差 σ，用不等式 $|(\chi-\mu)^2/\sigma| > 3$ 进行判断，满足不等式的被认为是离群点。

3.3 叶结点聚类

由于 MRF-树的分支因子的限制，对于个别特别大的簇可能被分裂在多个结点中。另外，由于异常顺序输入数据也可能造成一些簇分裂在不同的结点中，因此，在这个阶段，算法对所有叶子结点重新进行聚类。把个结点代表点当作结点的全体，根据代表点计算各结点的距离与用户输入的聚类数进行凝聚聚类。

3.4 全局聚类

前 3 个阶段，算法只是对随机取样到的数据进行了聚类。在最后阶段，算法扫描数据库对象与个代表点的距离对未标注的对象进行标注。算法至此结束。

4 算法的时空复杂性分析

首先，我们讨论时间复杂性中的计算复杂性。上面我们已经指出算法在第一阶段随机取样的代价是很小的（不到 2

秒的时间从几十万个数据点中取到几千个样本数据),故不作讨论。算法在第二阶段所耗费 CPU 的代价如下:设树的高度为 h(树的高度是由内存的大小与页面大小决定的),树的内结点分支因子为 F,为了插入一个点,需要访问的结点数为 $1+\log_F h$。在每一个结点为了查找最近的结点必须检查 F 个结点,访问结点的代价与对象纬度 d 有关,故插入所有的对象生成一棵树的代价是 $O(d*n*F(1+\log_F h))$。但是,通常树不是一次生成的,必须经过多次的重建。设最多有 l 个叶子结点需要重建,故重建树的代价是 $O(d*l*F(1+\log_F h))$。设第一次装入内存的对象数是 n_1,要建树的总结点数是 n。我们提高阈值的启发式方法是把 RT 设为所有最邻近的结点代表点之间的距离的平均值,每一次树的重建将把树的高度降为一半,故重建树的次数为 $\log_2(n/n_1)$。故重建树的总代价为 $O(\log_2(n/n_1)*d*l*F(1+\log_F h))$。所以第二阶段算法的总代价为 $O(d*n*F(1+\log_F h)+\log_2(n/n_1)*d*l*F(1+\log_F h))$。值得注意的是,第二阶段操作的对象都是第一阶段取样的对象,它们通常约为总对象的 3%。

对于空间数据的操作,时间复杂性更重要的是花费在 I/O 的时间。第 1 阶段对数据进行取样要扫描一次数据库,第 2 阶段在处理离群点时会访问内存,但这极少次数的访问可以忽略不计。第 3 阶段在内存进行,没有 I/O 时间。第 4 阶段算法再次访问一遍数据库。故本算法 I/O 的时间复杂性为 $O(n)$。

在空间复杂性方面,本算法对内存并没有特别的要求,算法可以适应不同的内存。当内存空间不足时可以空间来换取时间。

5 实验

实验分为 3 个部分,第 1 部分验证算法的有效性;第 2 部分验证算法的效率;第 3 部分对算法的参数变化进行讨论。

5.1 算法的有效性

在算法有效性试验中,我们进行了大量的试验,限于篇幅,这里我们报告两个实验结果。

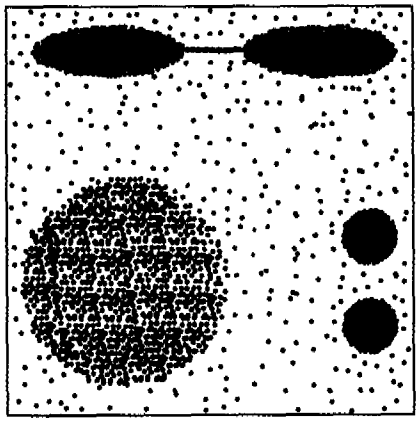

图 2 有效性实验的数据一

第一,为了能和已有的算法进行对比,我们选择了 BIRCH 算法与 CURE 算法都用到的含有 3 个球形聚类与 2 个椭圆聚类的实验数据,如图 3 所示。数据中包含一个大球形聚类、两个小球形聚类、用离群点连接的两个椭圆形聚类以及随机分布的离群点。BIRCH 算法能查找到两个椭圆聚类,但它把大球聚类分成两半,而且把两个小球并成一个聚类,聚类结果如图 4 所示。因为 BIRCH 的 CF-树的每个节点大小限制只能包含有限数目的结点,故它难于对应数据的每一个自然聚类,因此,它会把大球形聚类劈开,并且把小球形聚类合并。而本算法用代表点来表示聚类,使得每个节点所包含的对象数没有严格的限制,故能查找包含对象数目差异较大的聚类。如图 5 所示,本算法能够准确查找到一个大球、两个小球以及两个椭圆的聚类。CURE 算法同样能找到这些聚类,结果与本算法一致。

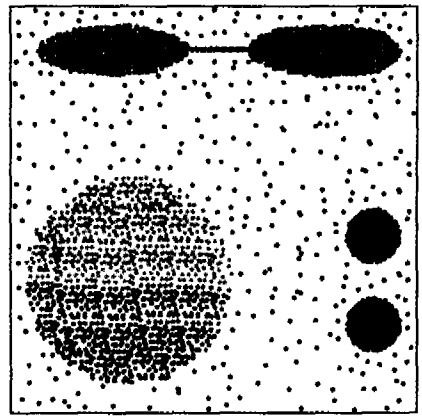

图 3 BIRCH 算法在数据一中的聚类结果

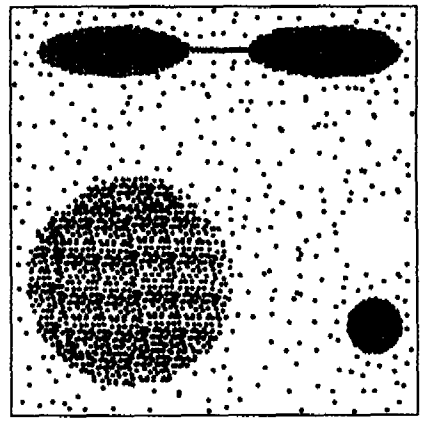

图 4 算法 CAMFT 在数据一中聚类结果

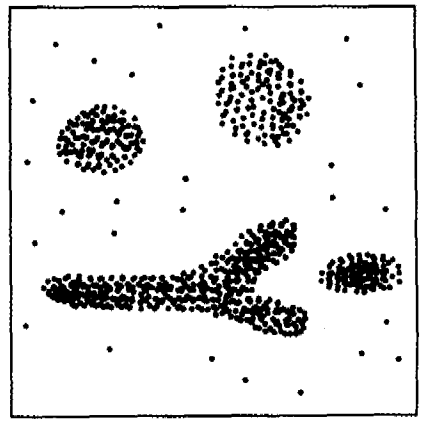

图 5 有效性实验的数据二

第二,我们用著名算法 DBSCAN 实验用到的数据,这个数据库包括 2 个类圆形,1 个"一"字,1 个"Y"字形的簇,以及 26 个离群点,如图 6 所示。BIRCH 算法能查找到 2 个类圆形

聚类与“一”字形聚类，但它把“Y”聚类分成两半，聚类结果如图 7 所示。因为 BIRCH 的 CF-树的每个节点大小限制只能包含有限数目的结点，故它难于对应数据的每一个自然聚类，因此，它会把复杂的“Y”聚类劈开。而本算法用代表点来表示聚类，使得每个节点所包含的对象数没有严格的限制，并且能捕捉复杂形状，故能全部聚类，如图 8 所示。

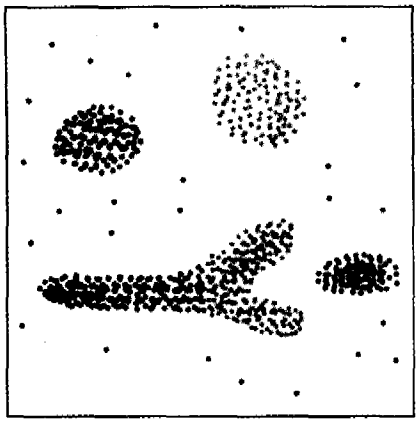

图 7　BIRCH 算法在数据二中的聚类结果

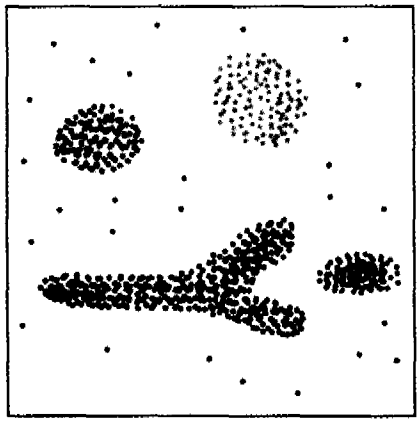

图 8　算法 CAMFT 在数据二中聚类结果

5.2　算法的效率

我们按 BIRCH 算法所提到的模拟数据产生器[5]，产生了 12 * 12 个聚类中心呈网格分布的、均值在聚类中心的 10000、20000、30000、40000、50000 个正态分布数据，如图 10 所示。算法 CAMFT 与算法 CURE 的采样率为 3%，算法 CAMFT 的代表点数取 3，算法 CURE 的划分区数为 2，实验结果如图 12 所示，实验结果表明算法 CAMFT 的聚类速度明显优于算法 BIRCH 与算法 CURE。

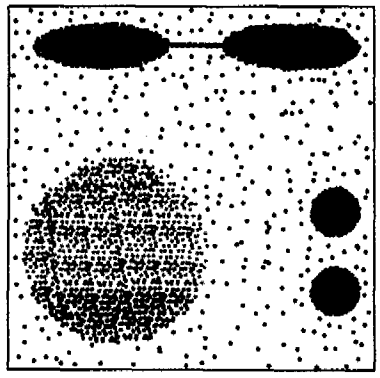

图 9　当收缩因子 s=0.1 时大球聚类分裂

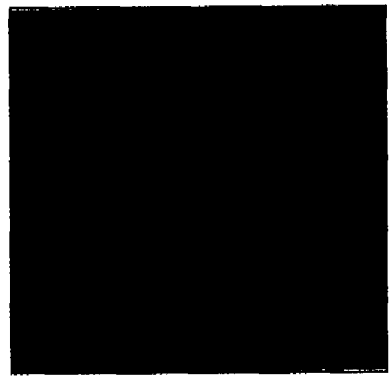

图 10　12 * 12 网格高斯随机数

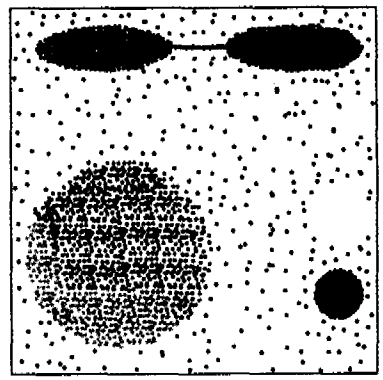

图 11　当收缩因子 s=0.9 时大球聚类分裂图

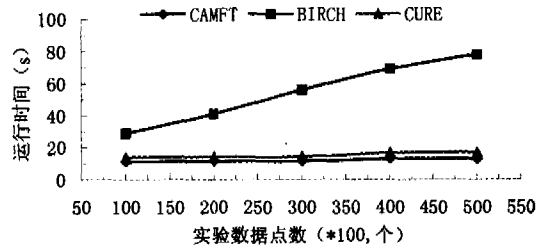

图 12　算法 CAMFT、BIRCH 与 CURE 的效率比较

5.3　参数讨论

我们用实验 1、2 数据对算法有影响的参数进行了研究：

(1)随机采样比例 r。实验表明，用上述第一种数据随机采样比例 s 达到 3%以上时，本算法将取得好的聚类；对于第二种数据，随机采样比例 s 达到 2%以上时，本算法就会取得好的聚类。

(2)代表点的数目 n。对于数据一，当代表点数目 $n>12$ 时，聚类效果较好。当 $n<12$ 时，大的球形将被劈开，不完整。对于数据二，当 $n>9$ 时即会取得较好聚类。因为只有足够的代表点，才能充分捕捉到聚类的形状。

(3)收缩因子 s。我们对 s 的取值从 0.1～1 之间用数据一进行实验，当 s 的值在 0.3～0.8 聚类结果较好。当 s 取小于 0.3 或大于 0.8 时，大球聚类可能被劈开，如图 9 与图 11 所示，当 s 取 0.1 与 0.9 时，大球聚类分裂。

(4)另外有两个参数，一是分支因子，一是阈值。为了提高效率，一个结点应放入一个页面。当数据的维度与内存的页面大小确定后，分支因子也自然确定。阈值的初值可以取 0，当然，根据不同的数据，可以赋予一定的初值显然会提高效率。在本实验中，在第一种数据中，RT 可设为 25～40；在第二种数据库中，RT 可设为 15～30。

小结 空间数据具有海量、连续、复杂、存在缺损与误差等的特点，要求空间聚类算法具有高效率，聚类结果独立于对象在空间的顺序，能处理各种不同形状的簇，并且对离群点是健壮的等性能，已有的算法难以同时满足要求。本文提出了一种基于多代表点的特征树，它基于 BIRCH 算法的思想，融入了 CURE 算法的优点，可以对海量的聚类数据进行压缩，并且能够捕捉复杂形状的簇。利用该数据结构，采用随机采样的方法，提出了一个适合处理空间数据的聚类算法 CAMFT，该算法能够满足上述空间聚类算法的要求，有效地快速地处理海量空间数据。实验结果表明了算法 CAMFT 可以识别不同形状的空间聚类，并且效率优于已有的适合处理空间数据的聚类算法 BIRCH 与 CURE。最后，本文对影响算法的参数进行了评价，得出了随机采样比例、代表点的数目、收缩因子、树结点的分支因子与阈值等参数与算法聚类结果的关系。

参 考 文 献

1 Dunham MH. Data mining introductory and advanced topics. Upper Saddle River, N. J. : Prentice Hall/Pearson Education, 2003. 221～243

2 MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley Symposium in Mathematics. Univ. of California, Berkeley, USA: Statistics and Probability, 1967

3 Lauritzen SL. The EM algorithm for graphical association model with missing data. Computational Statistics and Data Analysis, 1995, 19(2):191～201

4 Kaufman L, Rousseeuw PJ. Finding Groups in Data: An Introduction to cluster Analysis. New York: John Wiley & Sons, 1990

5 Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. In:Proceedings of the International Conference Management of Data (ACM-SIGMOD). Montreal, Canada, 1996. 103～114

6 Guha S, Rastogi R, Shim k. CURE: An Efficient Clustering Algorithm for Large Databases. In:Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98). Seattle , Washington, 1998. 73～84

7 Guha S, Rastogi R, Shim k. Rock: A robust clustering algorithm for categorical attributes. In: Proc. 1999 Int. Conf. Data Engineering (ICDE'99). Sydney, Australia, 1999. 512～521

8 Ester M, Kriegel H-P, Sander J, Xu X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In:Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, 1996. 226～231

9 Ankerst M, Breunig M, Kriegel HP, Sander J. OPTICS: Ordering points to identify the clustering structure. In: Proc. 1999 ACM-SIGMOD Int. Conf. Management of Data (DIGMOD'99). Philadelphia, PA, 1999. 49～60

10 Hinneburg A, Keim DA. An efficient approach to clustering in large multimedia databases with noise. In:Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98). New York, 1998

11 马帅，王藤蛟，唐世渭，杨冬青，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

12 Wang W, Yang J, Muntz R. STING: A statistical information grid approach to spatial data mining. In: Proc. 1997 Int. Conf. Very Large Data Bases (VLDB'97). Athens, Greece, 1997. 186～195

13 Sheikholeslami G, Chatterjee S, Zhang A. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In:the 24th International Conference on Very Large Data Bases. New York City, 1998. 428～439

14 Agrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dmensional data for data mining applications. In: Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98). Settle WA, 1998. 94～105

15 Fisher D. Improving inference through conceptual clustering. in Proc. 1987 AAAI Conf. Seattle, WA, 1987. 461～465

16 Gennari J, Langley P, Fisher D. Models of incremental concept formation Artificial Intelligence, 1989, 40(11-61)

17 Cheeseman P, Stutz J. Bayesian classification (AutoClass): Theory and results, in Advances in Knowledge Discovery and Data Mining. In: U. M. Fayyad, Piatetsky-Shapiro G., Smyth P. Uthurusamy R., eds. AAAI/MIT Press: Cambridge, MA, 1996. 153～180

18 Rumelhart DE, Zipser D. Feature discovery by competitive learning. Cognitive Science, 1985, 9:75～112

19 Kohonen T. Self-Organizing Maps, 3rd ed. New York: Springer-Verlag, 2001

20 Kohonen T. The self-organizing map. Proc. IEEE, 1990, 78(9):1464～1480

21 Höppner F, Klawonn F, Kruse R. Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition. New York: Wiley, 1999

22 Yager R, Filev D. Approximate clustering via the mountain method. in IEEE Transactions on Systems, Man & Cybernetics, 1994. 1279～1284

23 Davé R. Adaptive fuzzy c-shells clustering and detection of ellipses. IEEE Trans. Neural Netw., 1992, 3(5):643～662

24 karypis G, Han E-H, Kumar V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. COMPUTER, 1999, 32:68～75

25 Cherng J-S, Lo M-J. A hypergraph based clustering algorithm for spatial data sets. In: Proc. IEEE Int. Conf. Data Mining (ICDM'01). San Jose, California, USA, 2001. 83～90

26 Hartuv E, Shamir R. A clustering algorithm based on graph connectivity. Information Processing Letters, 2000, 76(4-6):175～181

27 Sharan R, Shamir R. CLICK: A clustering algorithm with applications to gene expression analysis. In:Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB). San Diego, California, 2000. 307～316

28 Ben-Dor A, Shamir R, Yakhini Z. Clustering gene expression patterns. Journal of Computational Biology, 1999, 6(3-4):281～297

29 Schölkopf B, Smola A, Müller K. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computat., 1998, 10(5):1299～1319

30 Biowulf AB-H, Horn D, Siegelmann HT, Vapnik V. Support vector clustering. Journal of Machine Learning Research, 2001, 2:125～137

31 张莉，周伟达，焦李成. 核聚类算法. 计算机学报，2005，25(6):587～590

32 Vitter J. Random sampling with a reservoir. ACM Transactions on Mathematial Software, 1985, 11(1):37～57

33 Motwani R, Raghavan P. Randomized algorithms. London Cambridge University Press, 1995