一种改进的 BIRCH 分层聚类算法*

赵玉艳 郭景峰 郑丽珍 李 晶
(燕山大学信息科学与工程学院 秦皇岛 066004)

摘 要 由于传统的 BIRCH 算法是用直径来控制聚类的边界,因此如果簇不是球形,它就不能很好地工作,而且传统的 BIRCH 算法只适用于单表。针对 BIRCH 的这些缺点,本文提出了一种改进的 BIRCH——IBIRCH 算法,该算法首先通过 ID 传播把多个表联系起来,使得 BIRCH 算法可以适用于多表的情况,再通过计算共享最近邻密度,可以发现任意形状的簇。实验表明,该算法不仅具有较强的可伸缩性,还可以得到较高精确的聚类结果。

关键词 BIRCH 算法,层次聚类,ID 传播,SNN 密度

Improved BIRCH Hierarchical Clustering Algorithm

ZHAO Yu-Yan GUO Jing-Feng ZHENG Li-Zhen LI Jing
(College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004)

Abstract The traditional BIRCH clustering algorithm has many shortcomings, such as it is only fit for single table and only finds the global clusters. For these shortcomings, we introduce an improved algorithm——IBIRCH algorithm. First, this algorithm joins every table through the tuple ID propagation to be applied in relational databases. Then, find arbitrary clusters using the shared nearest neighbor density algorithm. The experiment shows the efficiency and scalability of this approach.

Keywords BIRCH algorithm, Hierarchical clustering, Tuple ID propagation, SNN density

1 前言

聚类分析根据相似性对数据对象进行分组,发现数据空间的分布特征,是数据挖掘的一类主要方法。

传统的聚类算法可分为五类,即划分方法、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等[1]。这些算法只是在单表中寻找相似对象,然而现在大量的数据存储在关系数据库中。如何利用关系数据库,并从关系数据库中挖掘知识,得到聚类结果,是现在聚类研究的一个重要方向。

BIRCH 算法[2]是一种非常有效的、传统的层次聚类算法,该算法能够用一遍扫描有效地进行聚类,并能够有效地处理离群点。但是,BIRCH 算法是用直径来控制聚类的边界,如果簇不是球形,BIRCH 算法就不能很好地工作。针对传统聚类算法本身的局限性,及 BIRCH 算法的这一缺点,本文提出了一种改进的 BIRCH——IBIRCH 算法,该算法首先通过 ID 传播把多个表联系起来,使得 BIRCH 算法可以适用于多表的情况,再通过计算共享最近邻密度,可以发现任意形状的簇。实验表明,该算法不仅具有较强的可伸缩性,还可以得到较高精确的聚类结果。

本文的第 2、3 节分别简要介绍了 BIRCH 算法、共享最近邻密度算法,第 4 节对所提出的算法进行了描述,第 5 节给出了实验研究结果,最后是对本文的总结。

2 BIRCH 算法

2.1 BIRCH 算法的主要思想

层次聚类是聚类分析中一种常用的方法。根据层次分解的方式不同,层次聚类方法可分为凝聚层次聚类和分裂层次聚类。凝聚层次聚类采用自底向上的策略进行聚类,它从单成员聚类开始,把它们逐渐合并成更大的聚类。在每一层中,相距最近的两个聚类被合并。反之,则是分裂层次聚类。BIRCH 算法综合了层次凝聚和迭代的重定位方法,首先用自底向上的层次算法,然后用迭代的重定位来改进结果。它的主要思想是:扫描数据库,建立一个初始存放于内存中的聚类特征树,然后对聚类特征树的叶结点进行聚类。

2.2 聚类特征(CF)与聚类特征树

BIRCH 算法的核心是聚类特征和聚类特征树的使用。定义 1 和定义 2 分别给出了它们的定义。

定义 1(CF) 一个聚类特征(CF)是一个三元组(N,LS,SS),其中 N 是簇中的点的数目,LS 是 N 个点的线性和,SS 是 N 个点的平方和。

定义 2(CF 树) 一棵 CF 树是一个带有分支因子(一个结点可以具有最大子结点数)B 的平衡树。每一个内部结点对于其每个子结点都包含一个 CF 三元组。每个叶结点也代表一个簇,并且对其中每个子簇都包含一个 CF 条目。在叶结点中的子簇要有一个不超过给定阈值 T 的直径。

CF 是一个包含簇的信息的三元组,它提供了关于簇的信息的概括。CF 树采用自上而下的搜索方式,CF 树中的每个节点都包含关于其子簇的聚类特征信息。随着点不断加入聚类问题中,最终构建出 CF 树。一个点总是插入与自身最近的簇(由一个叶结点表示)中。如果叶结点的直径超过 T,则对树进行分裂和平衡(与 B 树中的做法类似)。聚类特征树的大小可以通过调节参数来改变。如果要存储的树需要的内存超过了主内存,那就要减小阈值 T,重新建立一棵树。这个

*)国家自然科学基金(60673136)。**赵玉艳** 硕士研究生,主要研究方向:数据挖掘、关系聚类。

重建过程并不需要将整个记录扫描一次，而是建立在老的树的叶节点的基础之上的。图 1 给出了一个 CF 树的例子，其中分支因子 B = 7，阈值 T = 6。

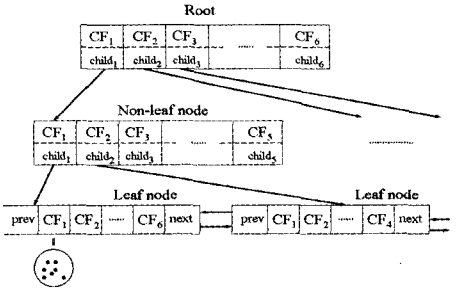

图 1　CF 树

BIRCH 算法通过一次扫描就可以进行较好的聚类，因此该算法适合于大数据量。但是，由于 BIRCH 算法是用直径来控制聚类的边界，使得该算法只适用于类的分布呈球形的情况。

3　共享最近邻(SNN)密度

共享最近邻(SNN)密度[3]这一算法是由 Levent Ertoz 等提出的，其核心思想是根据共享的最近邻来定义两个对象之间的相似性，通过引入 CURE 中代表点的思想定义核点，能处理包含有任意形状、大小子类时数据的聚类问题；同时又引入 DBSCAN 中密度的思想，使 SNN 能实现有噪声、孤立点和不同密度子类时数据的聚类。下面是有关 SNN 密度这一算法中几个比较重要的定义。

定义 3(SNN 相似度)　对象 p 和 q 之间的 SNN 相似度定义为 p 和 q 共享的 k-最近邻个数，即

Similarity(p,q) = size(nn[p] ∩ nn[q])

其中：nn[p]和 nn[q]分别是 p 和 q 的 k-最近邻集合，size(A)是集合 A 的大小。

定义 4(核心点)　一个点是核心点，如果在该点给定邻域(由 SNN 相似度和用户提供的参数 Eps 确定)内的点数超过某个阈值 MinPts，其中 MinPts 也是用户提供的参数。

定义 5(边界点)　边界点不是核心点(即它的邻域内没有足够的点使它成为核心点)，但是它落在一个核心点的邻域内。

定义 6(噪声点)　噪声点是既非核心点，也非边界点的任何点。

定义 7(直接密度可达)　给定一个对象集合 D，如果 p 是在 q 的 Eps 邻域内，而 q 是一核心对象，则对象 p 从 q 出发是直接密度可达的。在图 2 中，q 到 p_1，p_1 到 p 是直接密度可达的。

定义 8(密度可达)　如果存在一个对象链 $p_1, p_2, \cdots, p_n$，$p_1 = q$，$p_n = p$，对 $p_i \in D$，$(1 \leq i \leq n)$，p_{i+1}是从 p_i 关于 Eps 和 MinPts 直接密度可达的，则对象 p 是从 q 关于 Eps 和 MinPts 密度可达的。在图 2 中，q 到 p 是密度可达的。

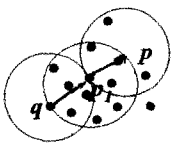

图 2　MinPts = 5，Eps = 1 cm

SNN 密度度量一个点被类似的点(关于最近邻)包围的程度，因此使用它可以发现任意形状的簇。

4　改进的 BIRCH 分层聚类算法：IBIRCH

传统的 BIRCH 分层聚类算法之所以只能发现球状的簇，是由于聚类特征树是用直径来控制聚类的边界。因此，要解决这一问题，必须对聚类特征树进行一些改进。

本文利用 SNN 密度中 Eps 和阈值 MinPts 这两个参数，将足够高密度的区域划分为簇，避免了用直径来控制聚类的边界，可以发现任意形状的聚类。

4.1　改进的聚类特征(ICF)与聚类特征树

定义 9(ICF)　一个改进的聚类特征是一二元组(N，CoSe)，其中 N 是簇中的点的数目，CoSe 是连接簇内点的邻近度图中边的加权和，边的权值是指边所关联的两个对象之间的相似性。

定义 10(ICF 树)　一棵改进的 CF 树是一个带有分支因子(一个结点可以具有最大子结点数)B 的平衡树。每一个内部结点对于其每个子结点都包含一个 ICF 二元组。每个叶结点也代表一个簇，并且对其中每个子簇都包含一个 ICF 条目。叶结点的子簇由 SNN 密度算法得到。

ICF 是一个包含簇的信息的二元组，其中 CoSe 用来计算簇内的平均凝聚性和簇间的分离性。假定 N_i 和 N_j 分别指簇 i 和簇 j 中的对象数，则 cohesion(C_i)，separation(C_i,C_j)分别是指簇内的平均凝聚性和簇间的平均分离性。

$$cohesion(C_i) = \frac{\sum_{x \in C_i, y \in C_i} similarity(x,y)}{N_i(N_i - 1)}$$

$$separation(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} similarity(x,y)}{N_i N_j} (i \neq j)$$

显然，对于凝聚性，值越高越好；而对于分离性，值越低越好。当需要分裂簇时，选择簇内平均凝聚性最小的簇进行分裂。当需要合并时，则合并分离性最小的两个簇。ICF 树随着记录一个一个的加入而自动生成：一个记录被放入那个离它最近的且密度可达的叶结点(类)中去。如果该记录到所有已存在的叶结点中没有密度可达的记录，并且叶结点不满，则创建一个新项。否则，必须分裂叶结点，并且在每次分裂后，跟随一个合并步。合并的目的是提高空间利用率，并避免不对称的数据输入顺序带来的问题。ICF 的大小可以通过调节阈值 MinPts 来改变，如果要存储的树需要的内存超过了主内存，那就要增大阈值 MinPts，重新建立一棵树。下面给出了 ICF 树中一个记录的添加过程。

```
输入：数据库 D，初始的 ICF 树，需加入的记录 Ent，MinPts，Eps；
输出：加入记录 Ent 后的 ICF 树；
进程：
从根节点开始，自上而下选择最近的孩子节点，直到叶子结点 NL；
If Ent 到 NL 中的某个对象是密度可达的，Then
  把 Ent 插入到该对象所在的簇中；
  更新 ICF 的值；
  Else
    If NL 中有空间可以插入 Ent，Then
      把 Ent 作为一个新的结点(簇)插入 ICF 树中；
      更新 ICF 的值；
    Else
        Do
          找出 NL 中分离性最大的一对记录 N1，N2(假设 N1 的凝
            聚性大于 N2)；
            Ent 取代 N2 的原位置，作为一个新的结点(簇)插入
            ICF 树中；
            更新 ICF 的值；
            将 NL 的父结点设为 NL，N2 设为 Ent；
          While (NL 中没有空间)
          将 Ent 插入到 NL 中；
          找出 NL 中分离性最小的一对记录 N'1，N'2；
          If N'1，N'2 不等于 N1，N2；Then
```

合并 N_1/,N_2/及其对应的子女结点;
更新 ICF 的值;
End if
End if
End if

4.2 IBIRCH 算法

IBIRCH 算法主要分为五个阶段。

第一阶段:由于传统的数据聚类算法都是在单一表上进行,如果对多个表进行聚类,就需要通过连接或者聚合将多个表中的数据结合到一个表中来。但是在结合的过程中会产生许多问题,如语义或信息的丢失、数据冗余、结合之后产生的表太大等,使得直接进行聚类十分困难。为了解决这一问题,本文采用了由 Yin Xiaoxin 和 Han Jiawei 等人提出的一种 ID 传播方法[4]。ID 传播方法通过 ID 的传播将各个表虚拟地联系起来,无需物化连接结果。该方法可以通过较少的冗余计算将不同关系表中的元组联系起来。

第二阶段:扫描所有数据,建立初始化的 ICF 树。把稠密数据分成簇,稀疏数据作为孤立点对待。将稠密数据装入内存。

第三阶段:在阶段二的基础上,构造一棵较小的 ICF 树。MinPts 减少,然后重新插入叶结点项(簇)。由于 MinPts 的减少,某些簇将合并。

第四阶段:进行全局聚类。通过计算簇内的平均凝聚性和簇间的平均分离性进行聚类。分裂凝聚性最小的簇,合并分离性最小的簇。

第五阶段:从第四阶段发现的各个簇中随机地选择一个点作为核心点,重新分布数据点,从而发现新的簇集合。由于页面大小的限制和参数 MinPts 的缘故,应当在一个簇中的点可能被分裂,而应当在不同簇中的点有时可能被合并。此外,如果数据集包含重复点,则这些点根据出现次数的不同,有时可能被聚到不同的类。通过多次重复本阶段,过程将收敛到一个局部最优点。

4.3 参数 Eps 和阈值 MinPts 的确定

本文中对于参数 Eps 和阈值 MinPts 的确定,类似于 DBSCAN[5]中参数的确定方法。基本方法是观察点到它的 k 个最近邻的距离(称为 k-距离)的特性。对于属于某个簇中的点,如果 k 不大于簇的大小的话,则 k-距离将很小。然而,对于不在簇中的点(如噪声点),k-距离将相对很大。因此,如果我们对于某个 k,计算每个对象与它的第 k 个最近对象之间的距离 k-dist,以递增次序将它们排序,然后绘制"排序 k-dist 图"(如图 6,横坐标代表排序后的对象,纵坐标为对应的 k-dist 值(图中为 4 dist)),则我们会看到 k-距离的急剧变化对应于合适的 Eps 值。如果我们选取该距离为 Eps 参数,而取它的值为阈值 MinPts,则 k-距离小于 Eps 的点将被标记为核心点,而其它的点将被标记为噪声或边界点。

图 3 样本数据

图 3 显示了一个样本数据集,而该数据的 k-距离图在图给出。用这种方法确定的 Eps 值依赖于 k,但并不随 k 改剧烈变化。如果 k 的值太小,则少量邻近点的噪声点将可能不正确地标记为簇。如果 k 的值太大,则小簇(尺寸小于 k 的簇)可能会被标记为噪声。实验表明,对于二维的数据集,一般 k 的取值为 4。

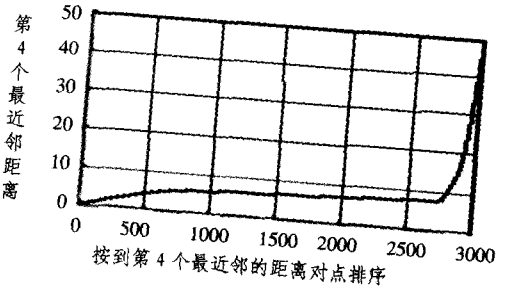

图 4 样本数据的 k-距离图

5 实验结果

在实验中,我们使用文[6]中使用的 DS1 数据集,如图 5 所示,该数据集包含 8000 个数据点,数据分布呈现 5 个不同大小、密度的圆形和椭圆形聚类以及噪声。

该实验运行时的硬件配置为:操作系统,Windows XP;CPU,Intel Pentium 2.0 GHz;内存,512M;硬盘,60G。设阈值 MinPts=4。图 6 和图 7 分别显示了用 BIRCH 和 IBIRCH 算法对此数据集的聚类结果。图 10 则显示了随着元组数的增加,这两种算法的运行时间的变化情况。

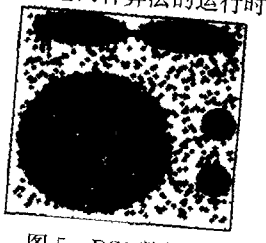

图 5 DS1 数据集

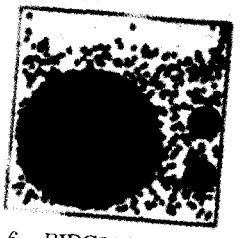

图 6 BIRCH 的聚类结果

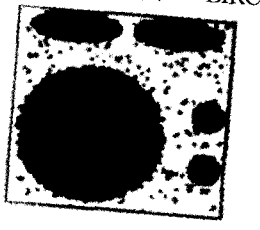

图 7 IBIRCH 的聚类结果

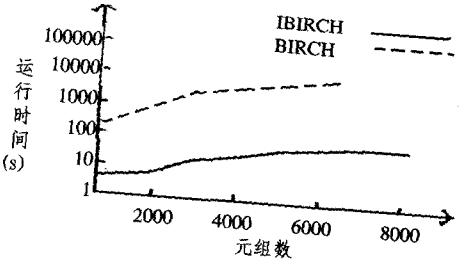

图 8 时间变化图

从图 6、7 可以看出,IBIRCH 算法能够揭示数据分布的自然簇特性,发现不同形状、大小和密度的聚类,有效处理噪声数据,得到较高精确的聚类结果。从图 8 可以看出,同样的条件下,IBIRCH 的运行时间远远小于 BIRCH。这主要是因为 IBIRCH 通过 ID 的传播把各个表联系起来,而 BIRCH 必

(下转第 206 页)

1	0	1	1	1
1	0	0	1	2
1	1	0	1	1
1	0	1	1	1

(a) t_4时的哈希表

8	2	6	7	4
7	1	0	10	9
2	5	3	9	6
8	3	6	0	8

(b) t_4时的哈希表

8	1	2	5	1
6	1	0	6	7
0	4	1	7	4
8	3	5	0	5

(c) 前10个项被移除

9	2	8	6	2
7	1	5	11	7
2	4	4	9	7
8	3	5	4	10

(d) t_{40}时的哈希表

图1 处理数据流时的哈希表状态图

4 性能分析

传统HCS分布式数据流处理模型是把单个数据流的挖掘结果,传输到上一层,反复挖掘上一层的结果,直到挖掘出最后的全局频繁模式。本文提出的方法也需要把从每个分布在各个结点的数据流的抽样数据传输到一个执行挖掘工作的结点上,所以两种模型都需要结点间进行通信。将从通信负载和算法运行时间两方面对HCS模型和本算法FFIDDS模型进行比较。

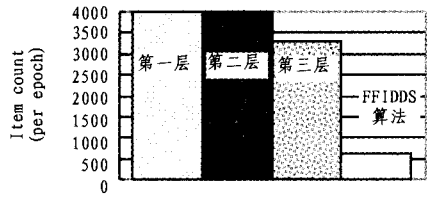

图2 传统HCS算法模型和FFIDDS模型结点通信的比较

分布式数据流挖掘可应用在网络安全中分布式拒绝攻击的早期检测和大规模分布式网络系统的"热点"的检测上。所以使用现实数据网络通信日志Internet2[8],并查找近期的热点来验证算法。使用一个由216个网络结点组成的分布式数据流。设置阈值s为0.01,传统分布式数据流挖掘模型中使每6个数据流为一组,根结点便在第4层,挖掘频繁项的算法使用lossy counting算法[6]。

在HCS数据流挖掘模型下采用216个单独数据流经过四层进行挖掘,需要通信三次才能把最终的挖掘结果输出,而FFIDDS模型只需要进行一次数据的通信。由图2可知,HCS模型每层通信数据量都比FFIDDS模型单次通信的信息量要大得多。另外,在系统的运行时间上,FFIDDS算法也要比基于传统HCS模型的算法少得多。

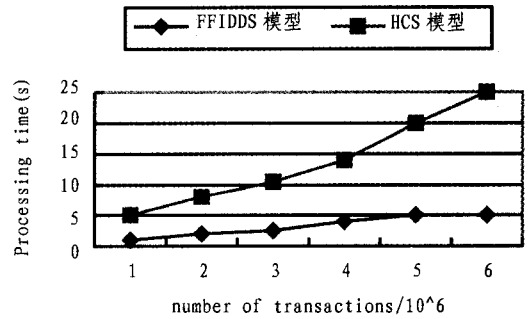

图3 算法运行时间的比较

结论 本文提出了一种在分布式数据流中查找近期频繁项的算法。它不同于先挖掘再聚合再挖掘的HCS模型,是先把分布式的数据流通过抽样聚合成一个数据流再进行频繁项挖掘。采用的抽样技术是基于密度的偏倚抽样,抽样操作由为每个单独数据流设置的监视结点完成。这种抽样算法根据数据密度改变抽样率,所以抽样的数据可以代表整个数据流的信息。频繁项的挖掘过程由基于滑动窗口模式和基于哈希的计数方式相结合的方法完成。实验结果表明,不论从通信负载和运行时间上,FFIDDS算法都要优于传统HCS模型。

参考文献

1 Arasu A, Manku G S. Approximate quantiles and frequency counts over sliding windows. In:PODS, 2004
2 Kim H A, Karp B. Autograph: Toward automated distributed worm signature detection. In:Proceedings of the 13th Usenix Security Symposium, 2004
3 Vitter J S. Random sampling with a reservoir[J]. ACM Transactions on Marhematical Software, 1985, 11(3): 37~57
4 Kollios G, Gunopoulos D, Koudas N. Efficient biased sampling for approximate clustering and outlier detection in large data sets [J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(5): 1170~1187
5 Manjhi A, Shkapenyuk V. Rinding(Recently) Frequent Items in Distributed Data Streams. In:Proceedings of the 21st ICDE, 2005
6 Manku G S, Motwani R. Approximate frequency counts over data streams. In:Proc. of VLDB, 2002
7 Karp R, Papadimitriou C, Shenker S. A simple algorithm for finding frequent elements in sets and bags. Transactions on Databas e Systems, 2003
8 Internet2. Internet2 Abilene Network. Hrrp://Abilene. internet2. edu

(上接第182页)

须对表进行物理的链接,这不仅浪费了空间,而且使得运行时间变长。因此,该算法具有较强的可伸缩性。

结束语 聚类是数据挖掘中的一个比较活跃的研究领域,目前在数据挖掘中已经开发出许多聚类算法。本文针对传统的BIRCH算法的局限性,提出了一种改进的BIRCH——IBIRCH算法,该算法首先通过ID传播把多个表联系起来,使得BIRCH算法可以适用于多表的情况,再通过计算共享最近邻密度,可以发现任意形状的簇。实验表明,该算法不仅具有较强的可伸缩性,还可以得到较高精确的聚类结果。

参考文献

1 Han Jiawei, Kamber M. Data mining: concepts and technique. China Machine Press, 2006
2 Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases[C]. In: Proc. ACM SIGMOD Conf. Management of Data, Montreal ,1996. 103~114
3 Ertoz L, Steinbach M, Kumar V. Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data[C]. In: Proc. of the 2003 SIAM International Conference on Data Mining, San Francisco, 2003. 150~155
4 Yin X, Han J, Yang J, et al. CrossMine: Efficient Classification Across Multiple Database Relations[C]. In:ICDE, Boston, 2004. 399~410
5 Ester M, Kriegel H, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]. In: Proc. 2nd Int Conf Knowledge Discovery and Data Mining (KDD'96), Portland , 1996. 226~231
6 Karypis G, Han E H, Kumar V. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling [J]. IEEE Computer, 1999, 32(8): 68~75