第 37 卷 第 24 期
Vol.37 No.24

计 算 机 工 程
Computer Engineering

2011 年 12 月
December 2011

• 软件技术与数据库 •　　文章编号：1000—3428(2011)24—0025—03　　文献标识码：A　　中图分类号：TP311

支持实时增量更新的闭子树聚类算法

黄 伟，郭 鑫，周清平

(吉首大学信息管理与工程学院，湖南 张家界 427000)

摘 要：现有的树聚类算法在树数据库实时更新后无法及时更新已有的聚类结果。为此，建立一种支持实时增量更新的闭子树聚类模型，以解决闭子树的增量聚类问题并提高聚类效率。针对树的半结构化特性，将结点语义和结点-边的结构特性结合在一起，提出一种准确率更高的树相似性度量方法，在此基础上，利用 CTUM 算法、TC 算法和 UTC 算法，分别解决闭子树增量更新、聚类和增量聚类等问题。实验结果表明，该算法具有较高的运行效率和聚类准确率。

关键词：聚类算法；数据挖掘；闭子树；增量更新

Closed Subtree Clustering Algorithm Supporting Real-time Incremental Update

HUANG Wei, GUO Xin, ZHOU Qing-ping

(School of Information Management and Engineering, Jishou University, Zhangjiajie 427000, China)

【Abstract】 In real application environment, when tree database carries out live updating, the present tree-cluster algorithm can not update existing cluster result. In consideration of the semi-structured characteristics and the lower accuracy rate of similarity measurement of tree, this paper puts forward a similarity measuring method of combining knot semantics and structure feature of one side of knot. On that basis, it brings forth Closed Tree Update Mining (CTUM) algorithm, Tree Cluster(TC) algorithm and Update Tree Cluster(UTC) algorithm, which can separately solve problems of increment updating, clustering and increment clustering of close subtree. Experimental result proves that the novel algorithms are efficient and practicable to own higher executing efficiency and better cluster accuracy rate.

【Key words】 clustering algorithm; data mining; closed subtree; incremental update

DOI: 10.3969/j.issn.1000-3428.2011.24.009

1 概述

近年来，有关树挖掘的研究受到了广泛关注，如频繁子树挖掘、树聚类与分类，被普遍应用于分子生物学、金融、电信、电子商务等领域，树挖掘技术已成为数据挖掘领域倍受关注的研究热点。在频繁子树挖掘方面，文献[1]提出了有序树挖掘算法，文献[2-3]提出了基于投影分枝的频繁子树挖掘算法，文献[4]提出了无序树挖掘算法，这些算法具有较好的子树挖掘效率。在树聚类分析方面，文献[5]和文献[6]分别提出 XML 文档树的聚类方法，2 种算法在处理大数据量时，需要的运行时间较长，聚类效率较低。在增量更新方面，文献[7]提出 PFUP 算法，然而该算法只适用于关联规则挖掘。本文提出一种支持实时增量更新的闭子树聚类算法，以解决增量式树聚类问题，提高算法可用性。

2 基本概念

定义 1(有序标号树[1]) 有序标号树 $T=(V,E)$ 是一个具有标号和根结点的有向无环图，其中，V 表示树结点集合，$E=\{(x,y)\,|\,x,y\in V\}$ 表示树中的边集合，用 $L=\{l_0,l_1,\cdots,l_n\}$ 表示一个标号集，结点 V 与 L 存在一个对应关系 $l:\ V\to L$，l_i 称为结点标号。有序是指树中每层结点按照从左至右的顺序形成兄弟关系。

定义 2(树结点关系[5]) 令 $V_x=(f,l,\mu)$ 表示一个树结点，其中，f 是 x 的父结点的序号；l 是 x 的序号；μ 表示以 x 为根的子树的最右叶子结点的序号，那么对树中任意 2 个结点 X 和 Y，用 $V_X=(f_X,l_X,\mu_X)$ 和 $V_Y=(f_Y,l_Y,\mu_Y)$ 表示，存在以下 3 种关系[2]:

(1)父子关系。若 $f_Y=l_X$，则称结点 X 是结点 Y 的父亲。

(2)子孙关系。若 $l_X<l_Y$ 且 $\mu_X\geqslant\mu_Y$，则称结点 Y 是结点 X 的子孙结点，也称 $V_X\supset V_Y$。

(3)小于关系。若 $\mu_X<l_Y$，则称 $V_X<V_Y$。

定义 3(有序树匹配与出现) 给定模式树 d 和数据树 D，如果满足下列条件，存在映射关系 $f:V_d\to V_D$，则称 f 为 d 到 D 的匹配函数(关系)，或称 d 在 D 中出现，d 是 D 的子树。结点数目为 k 的子树称为 k 子树。可假设空树 $\perp$ 与所有树匹配。

(1) f 是一一对应的。

(2) $L_d(v)=L_D(\phi(v))$。

(3) $(v_1,v_2)\in E_d \text{iff}(\phi(v_1),\phi(v_2))\in P_D$。

(4) $preorder(v_1)<preorder(v_2)$ 是指对树的结点先序遍历后结点 v 的顺序编号。

定义 4(有序树数据库) 令 $TDB=\{T_i\,|\,i=1,2,\cdots,n\}$ 表示一个有序树数据库，其中，每个元组用 $(TID,String)$，元组中 TID

基金项目：吉首大学校级科研基金资助项目(11JD051)；吉首大学教学改革研究基金资助项目(10JD043)

作者简介：黄 伟(1981－)，男，讲师、硕士，主研方向：聚类算法，数据挖掘；郭 鑫，助教、硕士；周清平，教授、博士

收稿日期：2011-05-19　**E-mail：**jianghai079@126.com

是树在数据库中的 *ID*， *String* 是有序树的串表示。

定义 5(闭子树) 给定有序树数据库 *TDB*，如果 *TDB* 中不存在与子树 *T* 支持度相同的真超树，则称 *T* 为 *TDB* 中的闭树模式，*TDB* 中所有的频繁闭树模式(频繁子树)集合记为：

$$CF(TDB)=\{T\,|\,T\in F(TDB)\exists Q\in F(TDB)\ T\subset Q, rel\_sup(T,TDB)=rel\_sup(Q,TDB)\}$$

基于上述概念与定义，实时增量更新的闭子树聚类问题可以描述为：给定一个有序树数据库 $TDB=\{T_i\,|\,i=1,2,\cdots,n\}$，当数据库随时间发生改变时，运行聚类算法并得到 *n* 个互不相交的聚类 $C=\{C_1,C_2,\cdots,C_n\}$，使每个聚类树的相似度最小化，类间相似度最大化。

3 实时增量更新的闭子树聚类算法

定义 6(有效树库) 假设在一定时间间隔 $T=(t_1,t_2)$ 内，对有序树数据库 *TDB* 进行了更新，包括新增子树集 *tdb* 或删除子树集 *tdb'*，*d* 和 *d'* 分别为 *tdb* 和 *tdb'* 的大小，令 TDB^* 表示更新后的树数据库，则有 $TDB^*=TDB\cup tdb-tdb'$，称 TDB^* 为有效树库或更新树库。

性质 1 令 *t* 表示 *TDB* 中的频繁闭子树，若在某时间段内新增子树集 *tdb* 或删除子树集 *tdb'*，则 *t* 在有效树库 TDB^* 中仍然频繁的充要条件为：$t_{TDB^*}\geqslant s\times(D+d-d')$，其中，$t_{TDB^*}$ 表示 *t* 在 TDB^* 的支持数；*s* 为最小支持度阈值；*D* 为 *TDB* 中树的数目；*d* 和 *d'* 分别为 *tdb* 和 *tdb'* 的大小。

证明：设 t_{TDB} 和 $t_{|tdb-tdb'|}$ 分别表示 *t* 在 *TDB* 和增删集 $|tdb-tdb'|$ 的支持数，因为 *t* 为频繁闭子树，所以存在关系 $t_{TDB}\geqslant s\times D$、$t_{|tdb-tdb'|}\geqslant s\times(d-d')$，两者相加有：$t_{TDB}+t_{|tdb-tdb'|}\geqslant s\times D+s\times(d-d')$，又因为 $t_{TDB^*}=t_{TDB}+t_{|tdb-tdb'|}$，所以有：$t_{TDB^*}\geqslant s\times(D+d-d')$。证毕。

性质 2 若闭子树 t_k 不是频繁的，即 $t_k\notin L_k$，在数据库发生增量更新后，t_k 变成频繁 *k* 闭子树的必要条件是 $t_{tdb}\geqslant s\times(d-d')+t_{tdb'}$。

证明：因为 $t_k\notin L_k$，存在 $t_{TDB}<s\times D$，假设不等式 $t_{tdb}<s\times(d-d')+t_{tdb'}$ 成立，容易得到如下公式：

$$t_{TDB^*}=t_{TDB}+t_{tdb}-t_{tdb'}$$

$$t_{TDB^*}<s\times D+s\times(d-d')+t_{tdb'}-t_{tdb'}$$

$$t_{TDB^*}<s\times D+s\times(d-d')$$

$$t_{TDB^*}<s\times(D+d-d')$$

于是，t_k 在有效树库 TDB^* 中是非频繁的，与命题不符，上述性质成立。

与传统简单数据聚类算法相比，树具有较复杂的结构，在树聚类算法设计中，如何快速准确地计算树之间的相似性成为本文首先应考虑的问题。传统的相似性度量方法包括基于距离和基于语义特征 2 种，这 2 种方法显然无法应用到树的相似性度量中，树不仅包括组成树的结点和边，还包括结点和边所表达出的结构信息，因此，在计算树相似度时既要考虑结点相似度，也要考虑结构相似度，所以有：

$$sim(t_1,t_2)=\frac{|t(t_1)\cap t(t_2)|}{|t(t_1)\cup t(t_2)|}=\lambda_1 Sim_{sem}(t_1,t_2)+\lambda_2 Sim_{Stru}(t_1,t_2)$$

其中，$Sim_{sem}(t_1,t_2)$ 和 $Sim_{Stru}(t_1,t_2)$ 分别表示语义相似度(结点相似度)和结构相似度；$\lambda_1,\lambda_2\in(0,1)$ 分别为相应的权值。

树中 *k* 个结点包括结点名称、结点标号等信息，为了充分体现树结点的相似性，设 $t_1=(t_{11},t_{12},\cdots,t_{1k})$ 和 $t_2=(t_{21},t_{22},\cdots,t_{2k})$，其中，$t_{1i}$ 和 t_{2i} 分别表示 t_1 和 t_2 中第 $i(1\leqslant i\leqslant k)$ 个结点，则存在如下公式：

$$Sim_{sem}(t_1,t_2)=\sqrt{\sum_{i=1}^{k}(t(t_{1i})-t(t_{2i}))^2},\ \bar{t}=\frac{1}{k}\sum_{i=1}^{k}t_i$$

定义 t_1 和 t_2 的相似性系数为：

$$S(t_1,t_2)=\frac{1}{1+d(t_1,t_2)}$$

在求频繁闭子树的同时，记录所有树结点的深度遍历序号，树的每一条路径都可采用数字序列来表达，并且从频繁序列中找出相似路径集合，按下列公式求得树的结构相似度：

$$Sim_{Stru}(t_1,t_2)=\frac{1}{n+1}\left[\left(\sum_{i=1}^{n}\frac{1}{L(R_i)}\cdot V(R_i)\right)+b\right]$$

其中，*n* 为子树(t_1 和 t_2)数量较多的第 1 层子树的数量；R_t 为子树(t_1 和 t_2)数量较多的第 *t* 层子树的根结点；$V(R_t)$ 为递归函数，计算方法为判断 R_t 是否为叶子结点，若是，则有 $V(R_t)=f(R_t)$，若 R_t 为普通结点，则通过下式计算：

$$V(R_t)=f(R_t)+\frac{1}{N(C_t)\sum_{e\in C}\frac{1}{L(e)}V(e)}$$

其中，$f(R_t)$ 为布尔函数，如果 R_t 在相似路径中，则结果为 1，否则为 0；C_t 为 R_t 的子结点集合；$N(C_t)$ 为子结点集合 C_t 的基数；$L(x)$ 是层数函数；*b* 为布尔值，若 2 个子树的根结点相似，则 *r* 值为 1，否则为 0。

闭子树增量更新算法 CTUM 思路如下：在原始树数据库基础上，运行闭子树挖掘算法得到所有频繁闭子树，增量更新包括 2 种情形：删除不再频繁的闭子树和增添新的频繁闭子树。对于第 1 种情况，任意频繁闭子树 $t\in L_k$，如果存在子树 $t'\in L_{k-1}-L_{k-1}^*$，则子树 *t'* 在有效树库中不再频繁，并且算法将记录所有的频繁闭子树并得到 $L_{k-1}-L_{k-1}^*$。对于第 2 种情况，分别遍历增减树库 *tdb* 和 *tdb'* 并计算候选子树的支持度，如果 $t_{tdb}<s\times(d-d')+t_{tdb'}$ 成立，则根据性质，该候选子树为非频繁的，停止搜索。除此之外，遍历未改变树库 $TDB-tdb'$，计算候选子树支持数，判断 $t_{TDB^*}\geqslant s\times(D+d-d')$ 是否成立，若是，则该子树为频繁的，否则应予以删除。

本文设计一个动态聚类算法，对更新后的频繁闭子树进行聚类，将所有相似的树结构聚合在一起。从闭子树集中随机选取一棵子树，计算该子树与其他子树的相似度，将得到的相似度结果从大到小进行排列，通过判断拐点并将对应的相似度作为阈值，所有大于该阈值的子树视为相似树结构，聚成一类，从闭子树集删除这些子树，按相似方法得到其他树类集，用 $C=\{C_1,C_2,\cdots,C_n\}$ 表示 *n* 个互不相交的树类。算法 TC 步骤如下：

算法 1 TC

输入 频繁闭子树集 *L*，密度 *d*

输出 所有树类 *C*

```
TC(L,d):
While(L!=null) do
t=Rand();
For all temp∈L do
```

Sim(t,temp);
End for all;
threshold=Sort();
S={all|Sim(all,t)>threshold};
If |S|⩾threshold then
C←S; L=L−S;
End If
End TC

利用算法 TC 可快速有效地进行树聚类。本文结合 CTUM 和 TC 算法，提出如下增量更新的闭子树聚类算法 UTC：

算法 2 UTC

输入 频繁闭子树集 L，新增子集 L_n，删除子集 L_d

输出 频繁闭子树集 L_n

UTC (L, L_n, L_d):
For all temp∈L do
S[]=Sim(t,temp);
End for all;
L_n←Max{ Sort(s[])};
L_n←L_n−L_d;
End UTC

4　实验结果及分析

为验证新算法的运行效率及可行性，利用 C++实现算法并进行大量实验。本文通过 8 个参数调整树数据的分布规律，生成数据的默认参数为：S100 P0.6 L10 I5 C4 N100000 H6 F4，限于篇幅，仅选取部分实验数据罗列。

第 1 部分实验测试 CTUM 算法的执行效率，设置不同大小的 *tdb* 和 *tdb'*，(*tdb* , *tdb'*)如下：U1(1 000,200) U2(2 000, 300) U3(3 000,400) U4(4 000,500) U5(5 000,600)，实验结果如图 1 所示。

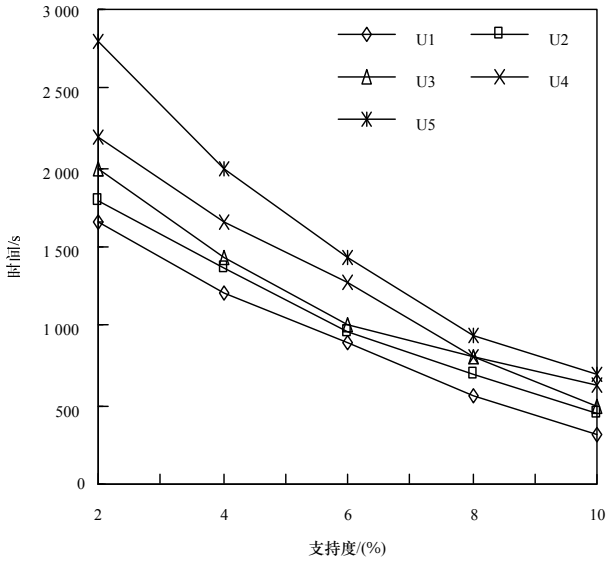

图 1　运行时间随支持度的变化情况

第 2 部分实验在增量环境下测试闭子树聚类算法的运行效率，增量参数设置同第 1 部分实验，在不同的相似度权值下，实验结果如图 2 所示。需要说明的是，随着语义权值的减少和结构权值的增加，所需时间随之减少，因为结构相似度的增加直接导致闭子树数量和聚类时间的减少，同时，在不同的增量数据库下，聚类效率良好。

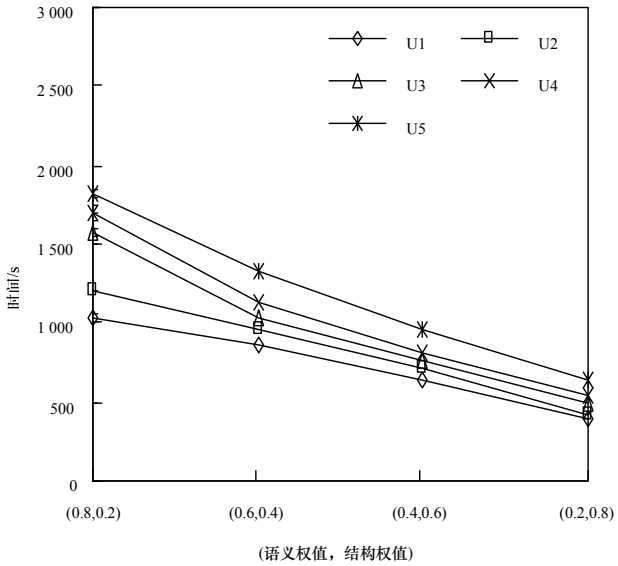

图 2　运行时间随相似度权值的变化情况

5　结束语

本文针对传统树聚类算法无法增量更新和数据量较大时算法效率较低等问题，提出一种支持实时增量更新的闭子树聚类算法，对更新后的频繁闭子树进行聚类，将所有相似的树结构聚合在一起。实验表明该方法是有效可行的。

参考文献

[1] Zaki M J. Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(8): 1021-1035.

[2] 朱永泰, 王　晨, 洪铭胜, 等. ESPM——频繁子树挖掘算法[J]. 计算机研究与发展, 2004, 41(10): 1720-1726.

[3] 赵传申, 孙志挥, 张　净. 基于投影分支的快速频繁子树挖掘算法[J]. 计算机研究与发展, 2006, 43(3): 456-462.

[4] Li Yun, Guo Xin, Yuan Yunhao. An Efficient Algorithm to Mine Unordered Trees[C]//Proceedings of the 8th IEEE/ACIS International Conference on Computer and Information Science. Shanghai, China: [s. n.], 2009: 331-336.

[5] Charu C, Aggarwal N, Zaki M, et al. XProj: A Framework for Projected Structural Clustering of XML Documents[C]// Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2007: 46-55.

[6] 吴扬扬, 雷　庆, 陈锻生, 等. 一种从 XML 数据中发现关系信息的方法[J]. 软件学报, 2008, 19(6): 1422-1427.

[7] 黄德才, 张良燕. 一种改进的关联规则增量式更新算法[J]. 计算机工程, 2008, 34(10): 38-39.

编辑　张正兴