一种基于约束的多维数据异常点挖掘方法*

李翠平[1+], 李盛恩[1], 王 珊[2], 杜小勇[2]

[1](中国科学院 计算技术研究所,北京 100080)

[2](中国人民大学 数据与知识工程研究所,北京 100872)

A Constraint-Based Multi-Dimensional Data Exception Mining Approach

LI Cui-Ping[1+], LI Sheng-En[1], WANG Shan[2], DU Xiao-Yong[2]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Institute of Data and Knowledge Engineering, Renmin University of China, Beijing 100872, China)

+ Corresponding author: Phn: 86-10-62515259, Fax: 86-10-62519453, E-mail: cuiping_li@263.net

http://www.ict.ac.cn

Received 2002-04-22; Accepted 2002-12-04

Li CP, Li SE, Wang S, Du XY. A constraint-based multi-dimensional data exception mining approach. *Journal of Software*, 2003,14(9):1571~1577.

http://www.jos.org.cn/1000-9825/14/1571.htm

Abstract: Data exceptions often reflect potential problems or dangers in the management of corporation. Analysts often need to identify these exceptions from large amount of data. A recent proposed approach automatically detects and marks the exceptions for the user and reduces the reliance on manual discovery. However, the efficiency and scalability of this method are not so satisfying. According to these disadvantages, the optimizations are investigated to improve it. A new method that pushes several constraints into the mining process is proposed in this paper. By enforcing several user-defined constraints, this method first restricts the multidimensional space to a small constrained-cube and then mines exceptions on it. Experimental results show that this method is efficient and scalable.

Key words: OLAP; data mining; constraint; exception; materialize

摘 要: 数据中的异常点常常反映了企业经营中潜伏的问题或暗藏的商机,数据分析人员经常需要从大量的数据中找出这些异常点.最近提出的一种从数据中自动发现异常点的方法,将人们从繁重的体力劳动中解放出来.然而,该方法在计算效率和伸缩性方面还存在很多不足.针对这些不足,对该方法进行了优化和改进,提出了一种基于约束的多维数据异常点挖掘方法.通过在数据挖掘过程中引入约束条件,首先将数据立方体限制到一个小的多维空间,然后再从中找出异常点.实验结果表明该方法非常有效.

关键词: 联机分析处理;数据挖掘;约束;异常点;实体化

* Supported by the National Natural Science Foundation of China under Grant No.60273017 (国家自然科学基金), the National High-Tech Research and Development Plan of China under Grant No.2002AA4Z3420 (国家高技术研究发展计划(863)), the National Grand Fundamental Research 973 Program of China under Grant No.2001CCA03003 (国家重点基础研究发展规划(973))

第一作者简介: 李翠平(1972－),女,山西高平人,博士,讲师,主要研究领域为数据库,数据仓库,数据挖掘.

中图法分类号: TP311 文献标识码: A

随着数据仓库和联机分析处理(OLAP)技术的发展,大量的数据被存放在数据仓库系统中.目前,大多数数据仓库仅用于多维联机分析处理,然而,随着数据挖掘技术的发展,人们希望能够在数据仓库系统中引入强有力的数据挖掘工具,从而使得更智能、更快速的数据分析成为可能[1].最近提出的一种从数据中自动发现异常点的方法——数据立方体探查方法[2](discovery-driven exploration of OLAP data cubes,简称 DDE),将人们从繁重的体力劳动中解放出来.该方法首先观察数据分析人员是如何分析数据的,然后自动检测数据中的异常点并将其标示出来.根据标示出来的异常点,数据分析人员就可以按照数据的层次结构逐层向下钻取,找出数据异常发生的原因.该方法的最大贡献是将数据分析人员从繁琐的手工操作中解放出来,同时降低了人为出错的概率,故而将 OLAP 工具带入了一个智能化交互式操作的新阶段,具有非常深远的意义.然而,在计算效率和伸缩性方面,它还存在着如下不足:

第 1,要计算一个 d 维数据立方体(cube),需要计算 2^d 个由不同的维组合构成的子立方体(cuboid),若考虑维层次会更多.在这样大的数据集上进行的挖掘工作需要花费很长时间.考虑到每一次进行的数据挖掘工作通常只会涉及到 cube 中的一部分数据,如果只在与任务相关的数据集上进行操作,将会提高挖掘的效率和准确度.为此,我们引入了第 1 个约束条件——数据约束条件,它用来从 cube 中取出与挖掘任务相关的数据.有了这个约束条件,负责中国地区销售业务的经理可以指定只在中国地区的销售数据中挖掘客户的购买模式.

第 2,由于空间和时间的限制,实际数据仓库中的 cube 很难全部预先实体化,即使只考虑与任务相关的数据也是如此.当在此 cube 中挖掘数据异常点的时候,需要实时地计算所有的数据单元(cell),在这种情况下,查询响应时间将会增加.考虑到实际中用户往往希望某些维保持在较高层次,某些维钻取到较低层次,我们引入了第 2 个约束条件——层约束条件, 它用来限制给定数据集在某些维上能够钻取到的最底层.有了层约束条件,我们就可以利用数据仓库中某些已经实体化的 cuboid 来计算另外一些没有预先实体化的 cuboid,从而提高计算效率.尽管当需要继续钻取到层约束条件之下时,需要重新计算 cube,但这种“分而治之”的方法可以帮助系统实现真正意义上的“联机分析”性能.

第 3,异常点的定义标准因人而异.例如,公司的总裁可能认为超过期望值 40%的数据才算作异常,而对于部门经理来说,超过期望值 5%的数据可能就是非常值得注意的了.因此,根据人们对异常点理解上的差异,我们引入了第 3 个约束条件——异常约束条件,它通常以数据差或比的形式对异常点进行定义.

根据上述分析可以看出,要提高 cube 上多维数据异常点挖掘的效率,需要引入 3 个约束条件:(a) 数据约束条件,保证只在与挖掘任务相关的数据集上操作;(b) 层约束条件,限制挖掘任务能够钻取到的最底层;(c) 异常约束条件,指定用户对异常点的定义标准.

本文在文献[2]提出的 DDE 方法的基础上,提出了一种基于约束的多维数据异常点挖掘方法.本文第 1 节介绍相关背景知识及问题定义.第 2 节介绍基于约束的多维数据异常点挖掘方法.第 3 节对该方法的性能进行实验分析.第 4 节介绍了相关工作.第 5 节给出本文的结论.

1 背景知识及问题定义

数据仓库中的数据通常组织成多个维的形式,即 $D=\{d_1,d_2,\ldots,d_n\}$.每个维 d_i 组织成一个层次 H_i,每个层次由一些维层组成.所有维层次的笛卡尔集 $\Gamma=H_1\times H_2\times\ldots\times H_n$ 可以用一个格结构来描述[3].格结构是一个有向图,它的每个结点代表一个 cuboid,边代表各个 cuboid 之间的相互依赖关系.在实际中,为了降低查询处理的时间,通常将格结构中的一些 cuboid 预先实体化存储在数据仓库中.

通常,数据分析人员通过一些 OLAP 操作,如切片、切块、向下钻取等来分析数据,发现数据中存在的模式.尽管这些工具可以帮助数据分析人员探查 cube 中的数据,但该过程是手工实现的.他们通常根据自己的直觉或假设,逐步地探查 cube 中的数据,以期从中发现一些异常点.这种手工操作方式不但琐碎而且容易出错.

DDE 方法自动检测数据中的异常点并将其标示出来,实现了数据探查过程的自动化.简单地说,一个异常点

就是一个数据单元,它的实际值与期望值差别较大.数据单元的期望值是通过一个数学统计模型算出来的.整个异常点的挖掘过程分为 3 个阶段:第 1 步,计算 cube 中定义的数据聚集值,如求和或者求平均等;第 2 步,进行模型拟合,确定模型的系数,计算标准偏差;第 3 步,计算异常指示值,确定异常点.后两步可以和第 1 步交叠进行,因此,DDE 方法非常有效.

然而,如前所述,因为数据集太大,在整个 cube 上进行异常点挖掘工作的代价还是非常大.本文通过在数据挖掘的过程中加入一些约束条件对该方法进行改进和优化.

设 M 代表格结构 I 中所有已经实体化的 cuboid(假定 base-cuboid 总是要被实体化).如前所述,在多维数据中挖掘异常点需要 3 个约束条件:数据约束条件 C_{data}、层约束条件 C_{lev} 和异常约束条件 C_{exc}.数据约束条件 C_{data} 可以通过基于条件的数据过滤、数据切片或切块等来设定.层约束条件 C_{lev} 通过指定可以钻取到的最底层来设定.这两个约束条件将 cube 限定到一个小的多维空间中,称作“受限的数据立方体”(constrained-cube,简称 c-cube),它由满足条件 $C_{data}\cap C_{lev}$=true 的那些数据单元组成.其中的每个数据单元称作一个“受限的数据单元”(constrained-cell,简称 c-cell).异常约束条件 C_{exc} 给数据分析人员提供了根据个人特定情况定义一个数据异常点的灵活标准.

给定一个实体化 cuboid 集合 M,一个数据约束条件 C_{data},一个层约束条件 C_{lev} 和一个异常约束条件 C_{exc},基于约束的多维数据异常点挖掘问题就是,用最小的代价从 cube 中找出满足条件 $C_{data}\cap C_{lev}\cap C_{exc}$=true 的所有数据单元.

例 1:一个数据仓库由 3 个维组成,分别是时间、地区和产品.时间维上的层次是日、月、年.地区维上的层次是市、国家、洲.该模式包含的惟一度量值是某个时间某个地区某个产品的销售额.

假定数据约束条件 $C_{data}\equiv$(国家=“中国”),它表示挖掘工作所关心的数据集中在中国,其他地区的数据和本次挖掘任务无关.再假定层约束条件是 $C_{lev}\equiv$(时间=2,地区=1,产品=1),其中的数字表示某维可以钻取到的最底层.有了这样两个约束条件的限制,原来的 cube 被限制到一个小的 c-cube,它由 3 个维组成:时间、地区和产品.时间维上具有两层:年和月,地区维上具有两层:市和国家(由于数据约束条件的限制,聚集到洲是无意义的).这时,cuboid 的个数由原来的 32 个下降到 18 个.将异常约束条件设为 $C_{exc}\equiv(r\geq 2.5)$,r 代表标准偏差,定义为 $r=|y-y'|/\sigma$.其中 y 是实际的数据单元值,y'是预期的数据单元值,σ是标准偏离[2].基于约束的多维数据异常点挖掘就是要从这个 c-cube 中找出 $r\geq 2.5$ 的所有的数据单元.

如果 cube 全部实体化,那么构造上述 c-cube 非常简单(只需返回满足条件的值即可),从而在其上的异常点挖掘工作也会很快.但前面讲过,实际中 cube 全部实体化是不可能的.因此,本文假定,只有部分 cuboid 被实体化.我们的工作就是用这些实体化的 cuboid 来有效地构造 c-cube,并从中快速地找出所有的异常点.

2 基于约束的多维数据异常点挖掘方法

基于约束的多维数据异常点挖掘过程分为如下两个逻辑步骤:(1) 指定数据挖掘任务所涉及的数据集.通过在原来的 cube 上应用数据约束条件和层约束条件,得到任务相关数据集,即 c-cube.该过程可以看做是在原来的 cube 上进行的查询操作.(2) 确定异常点.通过一个数学统计模型,确定 c-cube 中的异常点(由异常约束条件 C_{exc} 定义).

接下来,我们将首先介绍 c-cube 的有效构造算法,然后介绍怎样从中找出异常点.

2.1 c-cube的构造算法

在 cube 上应用数据约束条件和层约束条件之后,得到了一个 c-cube 的定义.现在的工作是利用已经实体化的 cuboid 来快速构造该 c-cube.我们使用 c-cube 中所有 cuboid 的集合 CID 来代表 c-cube,假定已经实体化的 cuboid 集合是 M.我们的任务是设计好的算法,为每一个 cuboid cid($cid\in CID$,称为目标)寻找一个实体化的 cuboid m($m\in M$,称为输入),使得 cid 能从 m 计算出,并且所用的总代价最小.

计算 CID 中的一个 cuboid 的代价包括两部分:扫描输入的代价和聚集目标的代价.

每个输入 $m\in M$ 的扫描代价是

$$t_{\text{load}}(m)=\frac{size(m)}{pagesize}\times t_{\text{I/O}},\tag{1}$$

其中 $size(m)$表示 m 中数据单元的数目,$pagesize$ 是一个页面所能容纳的数据单元数目,$t_{\text{I/O}}$ 是读一个磁盘页面所需的代价.$t_{\text{I/O}}$ 可以通过统计的方法得出,比如,如果一个磁盘页面大小为 8K 字节,读一个文件大小为 8M 字节的文件用了 10s 的话,则 $t_{\text{I/O}}$ 的时间为 0.01s.

由输入 $m\in M$ 到目标 $cid\in CID$ 的聚集代价是

$$t_{\text{compute}}(cid,m)=size(m)\times t_{\text{map}}(m)+size(m)\times t_{\text{cpu}}(cid,m),\tag{2}$$

其中 $size(m)$表示 m 中数据单元的数目,$t_{\text{map}}(m)$是将 m 中的一条元组的维信息映射到目标 cid 中某条元组的维信息所需要的代价,$t_{\text{cpu}}(cid,m)$是将 m 中的一条元组聚集到目标 cid 中某条元组所需要的代价.$t_{\text{map}}(m)$和 $t_{\text{cpu}}(cid,m)$同样是采用统计的方法得出的.

由此,使用一个输入 $m\in M$ 计算一个目标 $cid\in CID$ 的代价是

$$cost(cid,m)=t_{\text{load}}(m)+t_{\text{compute}}(cid,m).\tag{3}$$

综上所述,给定一个 cube 的实体化 cuboid 集合 M,计算一个 c-cube 的所有 cuboid 集合 CID 所用的总代价是

$$T_{\text{total}}(CID,M)=\sum_{cid\in CID,m\in M}cost(cid,m).\tag{4}$$

现在的任务是尽量减少上述表达式的值,这是一个 NP 复杂问题[4].为了在可以接受的时间里得到一个有效的解决方案,我们提出了两个实用的近似优化的算法来同时计算 CID 中的多个 cuboid.

2.1.1 简单算法

最简单的从实体化的 cuboid 计算 c-cube 的方法是独立地为每个目标挑选一个最佳输入.但该方法存在的一个严重不足就是无法实现输入共享.例如:假定我们需要计算 $A^2B^3C^3$ 和 $A^3B^2C^2$(A^2 指维 A 的第 2 层,A^3 指维 A 的第 3 层,以此类推),实体化的 cuboid 集合是 $A^2B^2C^3$ 和 $A^2B^2C^2$.显然,根据前面的代价模型,我们选用 $A^2B^2C^3$ 来计算 $A^2B^3C^3$,使用 $A^2B^2C^2$ 来计算 $A^3B^2C^2$.但如果 $A^2B^3C^3$ 和 $A^3B^2C^2$ 都选用 $A^2B^2C^2$ 作为输入的话,尽管 $A^2B^3C^3$ 使用了一个不太好的输入,但由于它们可以共享扫描,总的计算代价有可能降低.这启发我们去寻找一个全局优化的算法,以便同时计算多个 cuboid.

2.1.2 AGO 算法

AGO(approximate global optimal)算法的基本思想是采用如下两种优化技术:(1) 共享排序,减少 CPU 代价和 I/O 代价;(2) 共享扫描,减少 I/O 代价.为了共享排序,将具有公共前缀的所有 cuboid 组织成一棵树.这样,通过流水线就可以将在同一棵树上的所有 cuboid 同时计算出来.为了共享扫描,将多个 cuboid 的计算问题转化为单源最短路径问题,然后采用 Dijkstra 算法来为多个 cuboid 同时选定最好的实体化 cuboid 集合.AGO 算法的具体步骤如下:

将 CID 中的所有 cuboid 组织成树(若干棵).首先,构造一个辅助图 $G_1=(V_1,E_1)$,V_1 是所有的结点集合,代表 CID 中的所有 cuboid,也就是说,V_1 等于 CID.如果结点 $i\in V_1$ 不用排序就可以计算出结点 $j\in V_1$,则存在一条有向边$\langle i,j\rangle\in E_1$.这一步的结果是将所有的目标组织成了若干棵树,扫描一次输入,就可以将每个树上的所有结点同时计算出来.我们用 X 代表 G_1 中入度为 0 的结点(树根),现在问题变成了为 X 中的每个 cuboid 寻找一个最佳输入.

构造另一个辅助图 $G_2=(V_2,E_2)$.开始,V_2 等于 X.对每一个 $m\in M$,如果至少存在一个 $x\in X$ 能够由它计算出来,则将 m 加入到 G_2.对每一个 $x\in X$,如果能从 m 计算出来,则往 G_2 中增加一条从 m 到 x 的边$\langle m,x\rangle$,并且将该边的权值记为 $W_{\text{edge}}(m,x)=t_{\text{compute}}(x,m)$.用 Y 代表在 G_2 中出现的那些 $m,m\in M$.现在的任务是从 G_2 中找到 Y 的一个子集 $Y'\subseteq Y$,使得每个目标 $x\in X$ 都选择了一个输入 $y\in Y'$同时,表达式 $\sum_{y\in Y'}t_{\text{load}}(y)+\sum_{x\in X,y\in Y',x\prec y}t_{\text{compute}}(x,y)$ 的值最小.

给 G_2 增加一个虚拟根结点.从根结点 $root$ 到每一个结点 $y\in Y$,增加一条边$\langle root,y\rangle$,并且将该边的权值记为 $W_{\text{edge}}(root,y)=t_{\text{load}}(y)$.现在,$G_2$ 变成一个带权的有向图,并且由根出发可以到达该图中的任一结点.这时我们就可以利用 Dijkstra 算法来计算从单源 $root$ 出发到每个叶子结点的最短路径.所有的出现在最短路径上的结点 $y\in Y$ 就构成我们所要求的最佳实体化 cuboid 集合 Y'.

将图 G_1 和 G_2 集成起来,就得到了受限数据立方体最优构造计划.

例 2:现在我们来看一下如何利用 AGO 算法来构造例 1 中所定义的 c-cube.为了简单起见,我们分别用 A,B,C 来代表时间、地点、产品这 3 个维.目标集合 *CID* 是{$A^2B^1C^1$,$A^3B^1C^1$,$A^2B^2C^1$,$A^3B^2C^1$,A^2B^1,A^3B^1,A^2B^2,A^3B^2,A^2C^1,A^3C^1,B^1C^1,B^2C^1,A^2,A^3,B^1,B^2,C^1,all}.假定输入集合 *M* 为{$A^1B^1C^1$,$A^1B^2C^1$,A^1C^1}.算法的第 1 步结束后,我们得到了第 1 个辅助图 G_1(如图 1 所示).

根据 G_1,可以得到所有的树根集合 *X*={$A^2B^1C^1$,$A^2B^2C^1$,A^2C^1,B^1C^1,C^1}.第 2 步和第 3 步结束后,我们得到了第 2 个辅助图 G_2(如图 2 所示,其中的实线表示由根到叶子结点的最短计算路径).

根据 G_2,可以得到 *X* 的最佳输入集合 *Y*′={$A^1B^1C^1$,A^1C^1}.

最后,将这两个图加以集成,就得到了构造例 1 中定义的 c-cube 的全局优化计划.

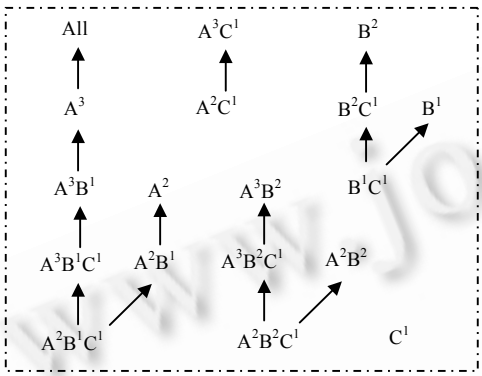

Fig.1 The first auxiliary graph
图 1 第 1 个辅助图

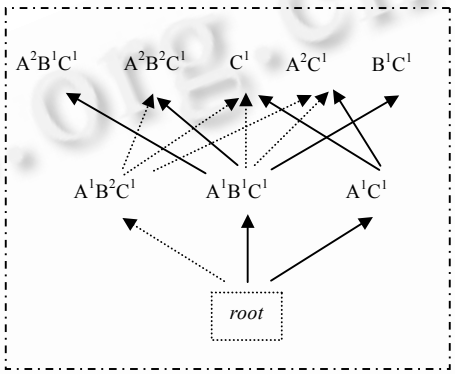

Fig.2 The second auxiliary graph
图 2 第 2 个辅助图

2.2 从c-cube中发现异常点

本文采用文献[2]中介绍的异常点定义和计算技术.不同之处是,我们将文献[2]中采用的统计模型加以扩展,使其可以处理维层次.同时,我们还给数据分析人员提供了可以通过设置不同的异常约束条件来定义异常点的灵活性.

假定一个 c-cube 有 *n* 维,每个维 d_i 包含一个层次,每个层次具有 l_i 个维层.对 c-cube 的一个 cuboid 的第 *r* 维第 *j* 层第 *i* 个位置的一个值 $y_{i_1i_2\ldots i_n}$,我们将它的期望值 $\hat{y}_{i_1i_2\ldots i_n}$ 定义为如下一个函数,即

$$\hat{y}_{i_1i_2\ldots i_n}=f(\gamma^{G}_{i_{rj}|d_{rj}\in G}\,|\,G\subset\{d_{rj}\,|\,1\le r\le n,1\le j\le l_r\}), \tag{5}$$

其中 γ 代表该模型的系数.函数 *f* 的形式和系数的计算方式请参见文献[2].

例如,一个两维的数据立方体,维 A 具有两层:A^1,A^2,维 B 也具有两层:B^1,B^2.为了计算维 A 的第 1 层的第 *i* 个成员和维 B 的第 1 层的第 *j* 个成员所构成的数据单元的期望值,公式如下:

$$\hat{y}_{ij}=f(\gamma+\gamma_i^{A^1}+\gamma_j^{B^1}+\gamma_i^{A^2}+\gamma_j^{B^2}+\gamma_{ij}^{A^2B^1}+\gamma_{ij}^{A^1B^2}+\gamma_{ij}^{A^2B^2}). \tag{6}$$

实际值 $y_{i_1i_2\ldots i_n}$ 和期望值 $\hat{y}_{i_1i_2\ldots i_n}$ 的标准偏差用 $s_{i_1i_2\ldots i_n}$ 表示:

$$s_{i_1i_2\ldots i_n}=\frac{|\,y_{i_1i_2\ldots i_n}-\hat{y}_{i_1i_2\ldots i_n}\,|}{\sigma_{i_1i_2\ldots i_n}}. \tag{7}$$

当一个数据单元的标准偏差大于异常约束条件 C_{exc} 时,该数据单元就被称为一个异常点.这样,采用类似的方法,一个 c-cube 中的所有异常点就可以很快地确定出来.

3 性能分析

为了分析所提出的基于约束的多维数据异常点算法的性能,我们通过模拟实验的方法进行了详细的实验.实验所使用的机器是 Pentium933,内存 128 兆,操作系统是 Window NT.采用的编程语言是 VC 6.0.

实验采用的数据集是由 APB-1[5]的数据生成器(APB.EXE)所生成的.APB-1 是 OLAP 委员会提出的一个专门用于测试 OLAP 应用程序性能的测试平台.我们采用数据生成器所生成的数据集的一个子集作为测试所用的数据集,它具有 4 个维属性和一个度量属性.4 个维属性分别是:产品(P),时间(T),客户(C),销售渠道(H).维 P 具有层次 $P^1 \rightarrow P^2 \rightarrow P^3 \rightarrow P^4 \rightarrow P^5 \rightarrow P^6$;维 T 具有层次 $T^1 \rightarrow T^2 \rightarrow T^3$;维 C 具有层次 $C^1 \rightarrow C^2$;维 H 具有层次 H^1.惟一的度量值是销售额.我们假定输入集合 M 是 $\{P^1T^1C^1H^1, P^2T^2C^2H^1, P^3T^2H^1, T^2C^2H^1, T^1C^2\}$.

我们采用构造受限数据立方体所用的总的时间作为度量手段,在算法开始的时候记录一个开始时间,在算法结束的时候记录一个结束时间,二者之差即为构造一个受限数据立方体所用的总时间,它包括生成构造计划的时间和执行构造计划的时间,而执行构造计划的时间又包含读输入 m 的 I/O 时间和聚集到目标 cid 的 CPU 时间.

首先,我们通过设置不同的数据约束条件和层约束条件,使得 c-cell 的个数大约从 1K 变化到 10K,将异常约束条件设置为 2.5.图 3(a)显示了两种算法(NAÏVE 和 AGO)随着 c-cell 的个数不同而发生变化的情况.从图中可以看出,当 c-cell 个数较少的时候,两个算法具有相似的性能,随着 c-cell 个数的增加,AGO 算法逐渐优于 NAÏVE 算法.这是因为 AGO 算法采用了扫描共享和排序共享技术,c-cell 个数越多,共享空间和机会就越大,所以总的计算代价会降低.

其次,我们测试了两种算法随着整体数据单元个数的不同而发生变化的情况.我们将所有维的层约束条件均设为 1,不设置任何数据约束条件,异常约束条件仍为 2.5.整体数据单元的个数从 11K 变化到 20K.图 3(b)显示了两种算法的性能结果.从图中可以看出,尽管两种算法的伸缩性都呈线性增长,但 AGO 算法总是要优于 NAÏVE 算法,这是因为简单算法只单独地考虑每一个目标,虽然局部得到了一些优化但未必全局最优.而 AGO 算法将所有的目标同时考虑在内,进行了全局优化,所以效率更高一些.

总的来说,上述实验证明,本文所提出的基于约束的多维数据异常点挖掘算法是非常有效的,AGO 算法由于同时考虑了所有的任务,对它们进行了全局的优化,从而获得了比 NAÏVE 算法更好的性能.

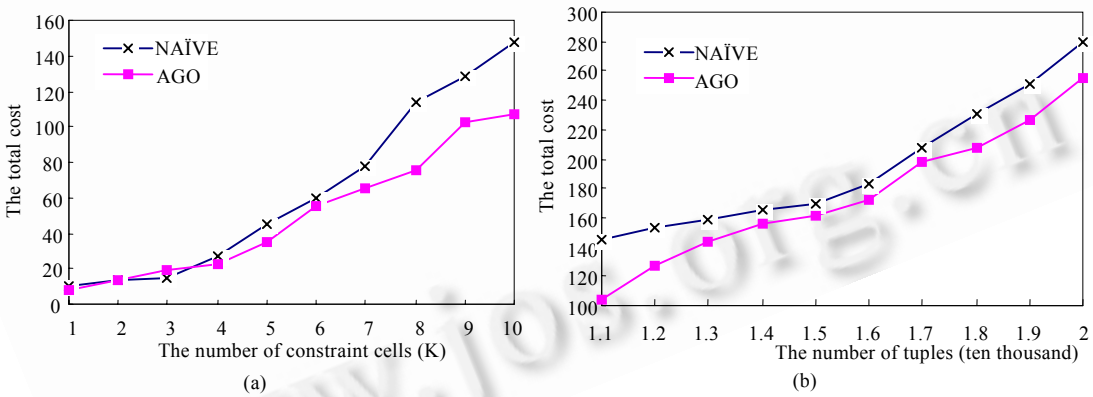

Fig.3 Performance comparison of two algorithms
图 3 两种算法性能比较

4 相关研究工作

与本文最相关的工作是文献[2]中提出的发现驱动的数据分析方法,它根据与一个数据单元相邻的其他数据单元的值来计算该数据单元的期望值.如果该数据单元的期望值与它的实际值相去甚远,则认为该数据单元是一个异常点.文献[2]的主要贡献在于,将这种繁琐的寻找异常点的工作加以自动化,从而减轻了数据分析人员的负担,将 OLAP 工具带入了一个智能化交互式操作的新阶段.但由于数据量太大,该方法在计算效率和伸缩性方面还存在着很多不足.本文针对这些不足,对该方法进行了优化和改进.

在数据挖掘领域,经常采用用户提供的各种约束条件来指导数据挖掘的过程,这些约束条件能够极大地缩减模式搜索的空间,从而提高了数据挖掘的效率.如文献[6]通过项目集限制进行关联规则的挖掘,文献[7]通过预期增益限制将密集数据集所产生的数以万计的规则缩减到用户真正感兴趣的那些规则,文献[8]通过只挖掘

那些兴趣度大于用户指定的某个条件的规则来提高挖掘效率.本文的贡献在于,将这种有效的采用用户约束条件来缩减搜索空间的方法应用到了从数据立方体中挖掘多维数据异常点的过程中,从而解决了文献[2]的计算效率低的不足.

还有一些文献对多维数据立方体中的模式发现问题进行了研究,如文献[9]提出的 cube-grade 问题对引起主要目标值发生改变的相关因素进行了研究,文献[10]提出的 DIFF 操作符用来挖掘高层某个数据单元的值极低于或者极高于其他数据单元的原因,文献[11]提出的 INFORM 操作符根据用户的浏览路径,归纳出用户的兴趣,从而快速返回用户感兴趣的数据,文献[12]提出的 RELAX 操作符从较低层出发,返回数据向上综合后发生异常的原因.本文可以看做是这些研究工作的延伸或者补充.

5 结 语

本文针对在大型多维数据集中挖掘异常点的问题进行了探索.现有方法中存在的关键问题是需要动态地构建 cube.为了提高数据挖掘的效率,我们在数据挖掘的过程中引入了 3 个约束条件:数据约束条件、层约束条件和异常约束条件.这些约束条件将大型多维空间限制到一个小的空间.由于数据分析人员通常都是自上而下地探查 cube,这种分而治之的方法能够帮助系统达到真正联机分析的性能.为了快速地构建 c-cube,我们提出了两种有效的算法:简单算法和 AGO 算法.实验证明,这种基于约束的多维数据异常点挖掘方法非常有效.以后的研究工作包括:(1) 寻找合适的方法帮助用户设置约束条件;(2) 设计更加有效的数据挖掘算法,等等.

References:

[1] Han J, Chee S, Chiang J. Issues for on-line analytical mining of data warehouses. In: Haas L, Tiwary A, eds. Proceedings of the SIGMOD'98 Workshop on Research Issues on Data Mining and Knowledge Discovery. Seattle: ACM Press, 1998. 2:1~2:5.

[2] Sarawagi S, Agrawal R, Megiddo N. Discovery-Driven exploration of OLAP data cubes. In: Schek H, Saltor F, Ramos I, Alonso G, eds. Proceedings of the 6th International Conference on Extending Database Technology. Valencia: Springer-Verlag, 1998. 168~182.

[3] Harinarayan V, Rajaraman A, Ullman J. Implementing data cubes efficiently. In: Jagadish H, Mumick I, eds. Proceedings of the ACM-SIGMOD International Conference on Management of Data. Montreal: ACM Press, 1996. 205~216.

[4] Liang W, Orlowska ME, Yu JX. Optimizing multiple dimensional queries simultaneously in multidimensional databases. VLDB Journal, 2000,8(3-4):319~338.

[5] http://www.olapcouncil.org/research/APB1R2_spec.pdf. 1998.

[6] Srikant R, Vu Q, Agrawal R. Mining association rules with item constraints. In: Heckerman D, Mannila H, Pregibon D, eds. Proceedings of the 1997 International Conference on Data Mining and Knowledge Discovery. AAAI Press, 1997. 67~73.

[7] Bayardo R, Agrawal R, Gunopulos D. Constraint-Based rule mining on large, dense data sets. In: Papazoglou M, ed. Proceedings of the 1999 International Conference on Data Engineering. Sydney: IEEE Computer Society, 1999. 188~197.

[8] Klemettinen M, Mannila P, Ronkainen P. Finding interesting rules from large sets of discovered association rules. In: Nicholas C, Mayfield J, eds. Proceedings of the 3rd International Conference on Information and Knowledge Management. ACM Press, 1994. 401~407.

[9] Imielinski T, Khachiyan L, Abdulghani A. Cubegrades: Generalizing association rules. Data Mining and Knowledge Discovery, 2002,6(3):219~257.

[10] Sarawagi S. Explaining differences in multidimensional aggregates. In: Brodie M, ed. Proceedings of the 25th International Conference on Very Large Databases. Edinburgh: Morgan Kaufmann Publishers, 1999. 42~53.

[11] Sarawagi S. User-Adaptive exploration of multidimensional data. In: Whang K, ed. Proceedings of the 26th International Conference on Very Large Databases. Cairo: Morgan Kaufmann Publishers, 2000. 307~316.

[12] Sathe G, Sarawagi S. Intelligent rollups in multidimensional OLAP data. In: Snodgrass R, ed. Proceedings of the 27th International Conference on Very Large Databases. Roma: Morgan Kaufmann Publishers, 2001. 531~540.