一种基于排序的XML文档数据交换算法

任 柯 杨 霞
(电子科技大学软件学院 成都 610000)

摘 要 在XML数据交换过程中,XQuery和XSLT将XML文档以树的形式在内存中转换,不仅速度慢,而且仅能处理小文件。为了快速并高效地处理大型的XML文档,定义了XML模式的表结构,应用排序方法实现了一个三阶段的数据交换算法。首先将源XML文档转换成符合源模式的表结构,然后按照目标模式对该表进行排序,最后将排序后的表转换成符合目标模式的XML文档。实验表明,该算法不但能高效地进行XML文档的转换,而且对大型的XML文档有着很好的扩展性。

关键词 数据交换 XML转换 模式映射 外排序
中图法分类号 TP310 **文献标识码** A

Sorting Based XML Data Exchange Algorithm

REN Ke YANG Xia
(School of Software, University of Electronic Science and Technology of China, Chengdu 610000, China)

Abstract In XML data exchange, XQuery and XSLT transform the XML document in memory in tree form, so they are not very efficient, and only can handle small documents. In order to transform large-scale XML documents efficiently, this paper defined the table of a schema, and proposed a sorting based three-phase XML data exchange algorithm. First, the algorithm transforms the XML document into a table, then, it sorts the table according to the target schema, and finally, it constructs a target XML document with the sorted table. The experiments show that the proposed algorithm can not only transform XML documents efficiently, but also be scalable to large-scale XML documents.

Keywords Data exchange, XML transformation, Schema mapping, External sorting

随着企业内部信息结构的调整,产生了大量的遗产数据。为了进一步有效利用遗产数据,我们需要将其转化为适合当前应用的数据格式。不同的企业间有不同的信息表示形式,例如电子商务,当企业间相互交换数据时,他们必须把接收到的数据转化为自己所能识别的格式。

数据转换问题就是把符合原数据模式的原实例数据转化为符合目标模式的数据实例。转换可以用源到目标的依赖即模式映射来表示。在一个模式映射下,可能有多个符合目标模式的数据实例。数据转换的目标是实例化一个目标数据,使之可以回答对目标模式所提出的查询请求。

数据交换是数据集成[1]研究领域的重要部分,在近几年得到了大量的研究。文献[2-4]对此作了详细的描述。大量的数据交换研究都是基于关系数据的;文献[5,6]研究了数据交换的理论基础;文献[7-9]又针对不同的问题进行了研究。

由于XML成为了数据交换的事实标准,XML数据交换也得到了广泛研究。文献[10-13]对XML数据交换进行了理论分析;文献[14-19]研究了如何进行目标数据的实例化;文献[20]研究了嵌套关系数据的交换问题。

1 相关工作

XQuery和XSLT之类基于树的转换器把XML文档看作树,采用DOM方法在内存中完成其转换。虽然它们都是图灵完整的语言,可以处理极其复杂的文档转换,但是实际应用中的文档转换往往是简单的。此外,XQuery和XSLT不能处理大的XML文档,也不能处理流式文档。因此需要研究流式或基于外存的转换方法。

· XML数据交换

文献[10,11]对XML数据交换的语义、一致性和查询应答问题进行了研究,重点探讨了解的存在性和如何用解来回答查询的问题。他们采用的解的计算方法是经典的chase算法,并没有考虑算法的性能。

· 流式XML转换

文献[14]提出了一种基于事件的转换方法,该方法仅需对输入扫描一次,因此可用在流式XML转换上。该算法基于属性文法,并产生一个有限状态转化器,只能处理类似重命名等的简单转换。文献[15]提出了一种转换语言,该语言工作在SAX的事件流上,可以集成到java的应用程序上。同样该语言也只能处理简单的XML文档转换。XStream[17]是一个图灵完成的编程语言,运用该语言程序员可以用函数式的树处理方式编写XML转换。处理过程最终以流的方式实现,但是输出随着输入的增加而增长。当输出文件在结尾增量增长时,该算法可以很好地工作;但是当输出文件随机在文

到稿日期:2013-07-09 返修日期:2013-09-31 本文受2013年度中央高校基本科研业务费专项项目青年教师基金(13NZYQN27)资助。
任 柯(1983—),男,硕士,主要研究方向为计算机应用、软件工程,E-mail:sichuanrenke1983@163.com;杨 霞 博士,副教授。

件中添加记录时，算法性能迅速下降。文献[19]实现了一个有效的SSXT算法。该算法可以处理由上至下的XSLT转换中的大部分。SSXT算法用堆栈维护输入数据，该堆栈的大小与输入文档树的深度成正比。XXST的缺点是仅能处理层次低的小的XML文档。

·多趟算法

文献[18]首次采用外存模型[23]来处理XML文档的转换。该算法限定了一类特定转换的资源利用，底层仍采用树转换器作为形式化基础。算法同样用堆栈实现。

·混合XML转换

文献[16]提出了一个"提取-转换-合并"的转换框架。该框架采用混合的方法。元组提取阶段把XML文档看作事件流；转换阶段把XML文档看作DOM树，并在内存中进行XML片段的转换；合并阶段首先把小的XML片段合并为大的XML片段，接着对已排好序的XML片段进行合并。该算法的主要工作在合并阶段完成。首先，数据在内存中递归分组；其次，进行外存的排序。当输入文档相对于内存较小时，该算法可以有效地工作，但是当输入文档很大时，性能迅速下降。此外，算法利用缓存来存储大的子树，这也会影响算法的性能。

2 XML文档和数据表

本节介绍XML、DTD和XML数据交换的相关知识。

2.1 XML文档和DTD

我们把XML文档看作是节点带标签的无秩树。用El表示元素类型的集合，用Str表示元素的可能取值范围。因为属性可以用节点的子元素表示，所以我们省去了节点的属性，以方便讨论。

给定有限集$E\subset El$，$V\subset Str$，一个XML树T是一个有限有序有向树，记为$\langle U,\downarrow,\rightarrow,lab,\rho,r\rangle$。

一个DTD(Document Type Definition) D是一个二元组$\langle P,r\rangle$，其中

·P是一个产生式的集合，每个产生式的左边是元素类型，右边是不包括根r的正则表达式，

$e::=\varepsilon \mid l,l\in E \mid e|e \mid ee \mid e^*$

·r是DTD所表示的树的根的元素类型，并且r不能出现在产生式的右边。

一个树T符合一个DTD($T\mid=D$)，需要满足两个条件。1)树的根的元素类型与DTD的根类型相同；2)树的任一节点的孩子结点的标签按照从左到右的顺序组成的字符串能够被该节点对应的产生式所接受。

为了讨论简便，我们用节点的子节点代替节点的属性，这并不影响本文算法的正确性。因为树的每个节点都对应一个元素类型，故在下文中我们不区分节点和元素。

定义1 DTD序列是DTD包含元素类型的全序，用(l_1，l_2，…，$l_{|E|}$)表示。DTD序可以用深度优先算法(DFS)通过下述规则递归得到：

·如果产生式形如$1\rightarrow\varepsilon$，$R(l)=(l)$；

·如果产生式形如$l\rightarrow l_1$，$R(l)=(l,l_1)$；

·如果产生式形如$l\rightarrow e^*$，$R(l)=(l,R(e))$；

·如果产生式形如$l\rightarrow e_1|e_2$，$R(l)=(l,R(e_1),R(e_2))$或者$R(l)=(l,R(e_2),R(e_1))$；

·如果产生式形如$l\rightarrow e_1e_2,\cdots,e_n,e_i(1\leqslant i\leqslant n)$是上述的正则表达式，那么$R(l)=(l,R_1,R_|,R_*)$。其中$R_1$是非重复节点的集合，$R_|$是选择节点的集合(|出现在$e_i$中)，$R_*$是重复节点的集合。

例1 如果D_S的产生式集合是：

dblp→inproceedings*

inproceedings→title,book,author*

那么D_S的序列是(inproceedings,title,book,author)。

例2 如果D_T的产生式集合是：

dbauthor→author*

author→name,col*

col→book,pub

pub→title*

那么D_T的序列是(author,name,col,book,pub,title)。

定义2(无数据节点) 如果e^*或者e^+出现在l的产生式的右边，那么e就是元素l的可重复子节点。那些有重复子节点的节点称为无数据节点。相反地，无重复子节点的节点称为数据节点。

定义3(关键字，内容和位置) 节点的可达子孙节点是那些不经过可重复节点就可以到达的节点。我们引用文献[21]中所述的关键字定义方法，定义节点的关键字为该节点的父节点的关键字与该节点的可达子节点的合取。节点的可达子孙节点集合又叫做节点的内容。从节点的关键字中去掉节点的内容就得到了节点的位置。如果两个节点的位置相同，那么在树中有相同的父节点。

2.2 XML数据交换

我们引用文献[10,11]中的树模式来描述源模式和目标模式之间的依赖关系。

定义4(Source-to-Target dependencies，STDs) STDs是形如下述的表达式：

$\pi(\bar{x},\bar{y}),\alpha=,\neq(\bar{x},\bar{y})\rightarrow\pi'(\bar{x},\bar{y}),\alpha'=,\neq(\bar{x},\bar{y})$

其中π和π'是树模式，并且每个变量只出现一次。

定义5(XML数据交换) XML数据交换是一个三元组$\langle D_S,D_T,\Sigma_{ST}\rangle$，其中$D_S$和$D_T$分别是源模式和目标模式，$\Sigma_{ST}$是由多个STD构成的集合，每个STD表明$D_S$到$D_T$的依赖关系。

定义6(解) 已知XML树T符合D_S和XML数据交换$\langle D_S,D_T,\Sigma_{ST}\rangle$，如果$T'$符合$D_T$，并且$\langle T,T'\rangle$满足$\Sigma_{ST}$中的所有STD，那么$T'$称作$T$在$\langle D_S,D_T,\Sigma_{ST}\rangle$下的解。

本文仅仅考虑XML数据交换中的一部分，即对XML文档的重构。这类转换可以用一个STD来表示。

2.3 XML文档和数据表

本文把XML文档看作是数据在DTD下的一种组织方法。数据是元素的值，相应的DTD便是该数据的语义。因此XML数据交换就是重构数据，使之符合目标DTD所表示的语义。

定义7(数据表) 已知XML文档和相对应的DTD，数据表是一个表，表中的列是DTD中的元素类型，列的顺序与DTD的序列相同。数据表的内容就是XML文档中的元素的取值。

命题1 已知DTD，当输入为XML树T时，可以在T的

线性时间内构建一个数据表；当输入为数据表时，可以在数据表的线性时间内构建 XML 树 T。

证明：把数据表的建立看作一个填表的过程。无数据节点所对应的列用该节点在父节点所处的位置来填写；数据节点用该节点在文档中的值填写。数据表实际上等价于文档树的扩张，任何两个无数据节点如果有相同的祖先，那么对应的列有相同的值。从文档树到数据表的转换是一个一一映射，并且转换可以在一次扫描内完成。

定义 8（简化数据表） 简化数据表是将数据表中的无数据列移除后得到的表。

从 XML 文档的定义可以看到，拥有可重复子节点的元素没有值，是无数据节点，该元素在数据表中的值是该节点在父节点中的位置。图 1 是一个 XML 文档，表 1 对应的简化数据表是将“dblp”和“inproceedings”所对应的列移除后得到的表。

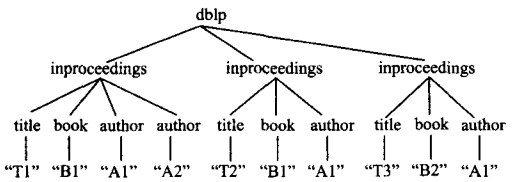

图 1 源文件 XML 树

表 1 源文件树的数据表

dblp	inproceedings	title	book	author
1	1	T1	B1	A1
1	1	T1	B1	A2
1	2	T2	B1	A1
1	3	T3	B2	A1

在 DTD 已知的情况下，一个数据表对应一个简化数据表，但是一个简化数据表可以有多个数据表。

命题 2 已知 DTD，当输入为 XML 树 T 时，可以在 T 的线性时间内构建一个简化数据表；当输入为简化数据表时，可以在该简化数据表的线性时间内构建 XML 树 T。

证明：对于每个树 T，我们可以按照命题 1，在 T 的线性时间内构建一个数据表。将该数据表的无数据列移除，便可以得到简化数据表。上述两步可以合并成一步同时进行，这样在对 T 扫描一遍的情况下便可以将 T 转换为简化数据表。

对于任一简化数据表，我们可以得到多个数据表，但是一旦 DTD 确定，通过如下 Expansion 算法便可以唯一确定一个数据表。同样该算法可以和命题 2 合并为一次扫描。

```
1. Expansion (SDT : File, D : DTD)
2. Begin
3. //deep first search to D
4. columnOrder := DFS(D)
5. //process the first line
6. Copy the first line of SDT to DT and fill "1" to dataless columns
7. for each line of SDT
8.     Compare it with previous line and find nColumn where the two
       lines have different contents
9.     Fill "1" to dataless columns behind nColumn;
10.    if nColumn isn't repeatable node
11.        Add "1" to its parent column;
12. return DT;
13. End
```

3 XML 数据交换算法

虽然 XML 适合用来进行数据交换，但是目前还没有关于数据交换的一致的框架。文献[10]采用了经典的 chase 过程来计算数据交换的解。文献[16]介绍了一个三步的交换框架，该框架包括元组提取、生成 XML 碎片和合并 XML 碎片，该框架在 clio[22] 中得到了应用。本文从排序的观点出发，提出了一个 XML 文档重构的三阶段算法。这 3 个阶段分别是数据表创建、目标排序和创建结构树。该算法可以描述为

$$T \rightarrow (DT \rightarrow) SDT \xrightarrow{D_T} SDT' \xrightarrow{D_T} (DT' \rightarrow) T'$$

从命题 1 和命题 2，我们很容易得到如下的命题。

命题 3 上述基于排序的三阶段数据交换算法能正确地进行 XML 文档的重构。

例 3 已知源 DTD 和目标 DTD（分别在例 1 和例 2 中），STD 可以描述为

dblp(inproceedings(title = x, book = y, author = z)) → dbauthor(author(name=z,col(book=y,pub(title=x))))

3.1 创建数据表

从命题 2 可以观察到，每个符合的 D_S 树 T 都可以被转化为简化数据表 SDT，该转化可以在 T 的线性时间内完成。因为无数据节点仅仅包含语义信息，对无数据节点的移除并不影响转换的结果。因此通过将表 1 中的列“dblp” 和 “inproceedings”移除，我们得到了简化数据表。

3.2 目标排序

由于简化数据表包含了树 T 中的所有数据，因此通过该简化数据表和目标 DTD 可以得到目标树。在这一步中，我们将简化数据表的列按照 D_T 的序列重新组织，然后对表进行排序。

在例 2 中，目标 DTD 中的列“name”与源 DTD 中的“author”相同，“book”和“title”在源和目标 DTD 中表示相同的元素类型。对简化数据表进行排序可以有两种方法。1）递归对简化数据表的列进行分组。先对“name”分组，对每个“name”再对“book”进行分组。2）把数据表中的每一行中各列的值合并，然后对合并后的数据进行排序，这样可以减少对外存的 I/O 访问。表 2 是对表 1 进行排序后得到的结果。

表 2 目标排序后的结果

name	book	title
A1	B1	T1
A1	B1	T2
A1	B2	T3
A2	B1	T1

3.3 创建结果树

在完成了对简化数据表的排序后，我们得到了排序后的简化数据表 SDT′。根据命题 2，该排序后的简化数据表和目标 DTD D_T 可以在 SDT′的线性时间内创建结果树。表 3 是根据表 2 和 D_T 得到的，图 2 是该数据交换的结果树即数据交换的解。

表 3 目标排序后的结果

dbauthor	author	name	col	book	pub	title
1	1	A1	1	B1	1	T1
1	1	A1	1	B1	1	T2
1	1	A1	2	B2	1	T3
1	2	A2	1	B1	1	T1

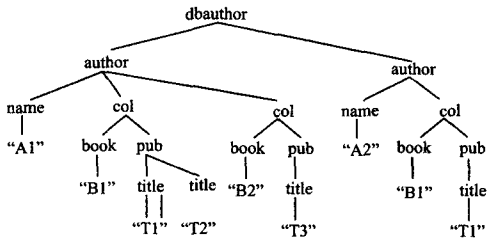

图 2　目标文件 XML 树

4　实验分析

为了测试算法的效率和可扩展性，我们对算法进行了实验分析。该算法采用 java 实现。实验在便携式电脑上完成，该电脑配置如下：双核 1.8GHz CPU，2GB 内存，Windows 操作系统和 java 虚拟机 1.7.0。每项实验进行了 10 次，结果取每次实验的平均值。

4.1　总体性能

我们采用例 2 中的 STD 测试了算法的可扩展性。首先，用整个 DBLP 数据集作为输入，只对元素“inproceedinds”进行转换。测试的数据集从 3MB 到 991MB。虽然输入数据集的文件将近 1GB，但是中间的数据表仅仅只有 300MB，因此排序可以在内存完成。结果如图 3 所示。

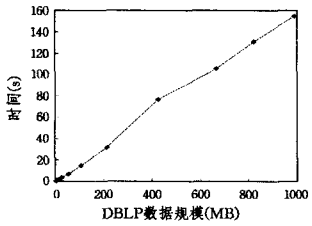

图 3　扩展性测试(DBLP)

图 3 表明，转换的运行时间几乎与 DBLP 文件的大小成正比。通过对中间数据表的观察，我们发现“inproceedings”元素的数量并不是与 DBLP 的大小成正比。

为了进一步分析该算法对输入数据的可扩展性，将“inproceedings”元素从 DBLP 文档中提取出来，从新进行了上述实验。针对排序过程，实现了基于内存的和基于外存的排序过程。图 4 为随着输入的变化，排序时间的响应变化。

在图 4 中，“internal”线是排序在内存中的时间变化曲线，它说明该算法在内存工作时是高效的。但是内存排序只能针对小文件进行，当应对 GB 级的文件时效率十分低。为了测试算法对大文件的支持程度，我们把 java 虚拟机的内存设置为 50MB，通过经典的 merge-sort 外存算法完成排序。在排序过程中数据表被分割为小于 50MB 的小文件。结果表明该算法在文件较大时(相对于内存)也有很高的效率。

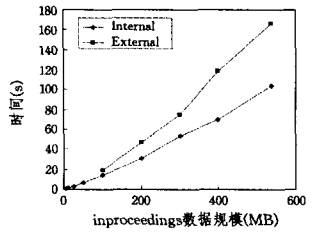

图 4　扩展性测试(inproceedings)

4.2　与 XQuery 和 XSLT 的比较

XQuery 和 XSLT 作为 W3C 的推荐转换标准被广泛应用。我们将所提出的算法与 ddxq/XQuery、saxon/XQuery 和 saxon/XSLT 进行了比较，实验仍然采用例 2 中的 STD。

首先对比了本文提出的 Sorting 排序方法与 ddxq、saxon 和 XSLT 3 种方法的内存使用。从图 5 可以看出，在数据规模增大的情况下，Sorting 算法占用的内存非常少，而其他 3 种算法的内存使用呈指数增长。此外，对算法的转换时间进行了比较，结果见图 6。虽然 XQuery 和 XSLT 可以进行 XML 文档的转换工作，但是它们仅能处理小文件，当输入文件达到 8MB 时，上述转换消耗 1GB 的内存，并且所用时间超过了 15 分钟，这严重限制了 XQuery 和 XSLT 的应用性能。相比之下，本文提出的算法不到 1 秒就能处理 8MB 的输入，因此不但占用的内存少，而且所需的时间短。

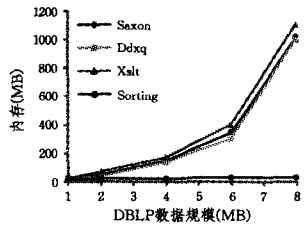

图 5　内存使用对比图

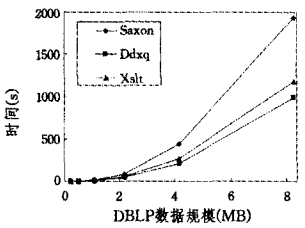

图 6　性能比较图

结束语　本文从排序的角度出发，提出了基于排序的 XML 文档转换算法，并证明了该算法的正确性。算法把 XML 数据交换看作是 XML 文档的重构，即 XML 文档内容的重新组织。实验表明，该算法不但能高效地进行 XML 文档的重构，而且对大型 XML 文档有很好的可扩展性。

参 考 文 献

[1] Lenzerini M. Data integration: a theoretical perspective[C]// PODS. New York, USA, 2002: 233-246

[2] Kolaitis P G. Schema mappings, data exchange, and metadata management[C]//PODS. New York, USA, 2005: 61-75

[3] Bernstein P A, Melnik S. Model management 2.0: manipulating richer mappings[C]//SIGMOD. New York, USA, 2007: 1-12

[4] Barceló P. Logical foundations of relational data exchange[J]. SIGMOD Rec., 2009, 38: 49-58

[5] 顾九春，刘璐. 基于 XML 的公路车辆智能监测信息交换研究[J]. 计算机应用研究，2012，29(8)：2985-2987

[6] Fagin R, Kolaitis P G, Popa L. Data exchange: getting to the core[J]. ACM Trans. Database Syst., 2005, 30: 174-210

[7] Gottlob G, Nash A. Data exchange: computing cores in polynomial time[C]//PODS. New York, USA, 2006: 40-49

[8] Libkin L, Sirangelo C. Data exchange and schema mappings in open and closed worlds[J]. Journal of Computer and System Sciences(In Press, Corrected Proof), 2010

[9] Fagin R, Kimelfeld B, Kolaitis P G. Probabilistic data exchange [C]//ICDT. New York, USA, 2010: 76-88

[10] Arenas M, Libkin L. XML data exchange: Consistency and query answering[J]. J. ACM, 2008, 55: 1-72

[11] Amano S, Libkin L, Murlak F. XML schema mappings[C]// PODS. New York, USA, 2009: 33-42

[12] David C, Libkin L, Murlak F. Certain answers for XML queries [C]//PODS. New York, USA, 2010: 191-202

[13] Amano S, David C, Libkin L, et al. On the tradeoff between mapping and querying power in XML data exchange[C]// ICDT. New York, USA, 2010: 155-164

(下转第 234 页)

2011,38(11):231-233

[9] Paul S. Andrews. An Investigation into Mutation Operators for Particle Swarm Optimization[C]// IEEE Congress on Evolutionary Computation. 2006:1044-1050

[10] 吴祁宗. 运筹学与最优化方法[M]. 北京:机械工业出版社,2003:114-117

[11] 赵小平. 差商最速下降法及其收敛性[J]. 华东化工学院学报,1992,18(6):807-809

[12] Yao Xin, Liu Yong, Lin Guang-ming. Evolutionary Programming Made Faster[J]. IEEE Transactions on Evolutionary Computation,1999,3(2):82-102

[13] Shi Yu-hui, Eberhart R C. A Modified Particle Swarm Optimizer [C]// Proceeding of IEEE International Conference on Evolutionary Computation. 1998,69-73

[14] Poli R, Kennedy J, Blackwell T. Particle swarm optimization[J]. Swarm Intelligence Journal,2007,1:33-57

(上接第 226 页)

[14] Nakano K, Nishimura S. Deriving Event-Based Document Transformers from Tree-Based Specifications [J]. Electronic Notes in Theoretical Computer Science,2001,44:181-205

[15] Becker O. Transforming XML on the Fly[C]// XML Europe. 2003

[16] Jiang H, Ho H, Popa L, et al. Mapping-driven XML transformation[C]// WWW, Banff. Alberta, Canada,2007:1063-1072

[17] Frisch A, Nakano K. Streaming XML transformations using term rewriting[C]// PLAN-X. Nice, France,2007:2-13

[18] Dvoráková J, Zavoral F. A Low-Memory SSXT Algorithm for XSLT Transformations[J]. Journal of Information Assurance and Security,2008,3:230-239

[19] Tanaka Y, Ito K, Fujima J. Meme media for clipping and combining Web resources[C]// International Conference on World Wide Web. Hong Kong,2001:201-210

[20] Fagin R, Haas L M, Hernández M, et al. Clio: Schema Mapping Creation and Data Exchange[C]// Lecture Notes in Computer Science. vol. 5600. Heidelberg Berlin: Springer,2009:198-236

[21] 宋玲,吕强,邓薇,等. 基于语义和结构的 XML 文档相似度的计算方法[J]. 中文信息学报,2012,26(5):59-64

[22] 陈谊,侯堃,新吉乐,等. 基于 XML 和关系数据库的可视化工作流系统[J]. 系统仿真学报,2012,24(1):110-116

(上接第 229 页)

测试。实验结果表明,本文提出的误差敏感的分类器算法在系统没有噪音的情况下分类预测的准确性要优于相关的算法;此外,误差敏感的分类器算法对噪音不敏感,当流数据中含有噪音时仍然具有很好的预测准确性。因此,该算法可以应用于大数据环境下数据流的在线分类预测。

参 考 文 献

[1] Domingos P, Hulten G. Mining high-speed data streams [C]// Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery And Data Mining. ACM,2000:71-80

[2] Yang H, Fong S. Moderated VFDT in stream mining using adaptive tie threshold and incremental pruning[M]// Data Warehousing and Knowledge Discovery. Springer,2011:471-483

[3] Hulten G, Spencer L, Domingos P. Mining time-changing data streams[C]// Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery And Data Mining. 2001:97-106

[4] Li W, Han J, Pei J. CMAR: Accurate and efficient classification based on multiple class-association rules[C]// IEEE International Conference on Data Mining. ACM,2001:369-376

[5] Han J. CPAR: Classification based on predictive association rules [OL]. http://sci2s.ugr.es/keel/pdf/algorithm/congreso/2003-Yin-CPAR.pdf,2003

[6] Thabtah F, Cowling P, Peng Y. MCAR: multi-class classification based on association rule[C]// The 3rd ACS/IEEE International Conference on Computer Systems and Applications. IEEE,2005

[7] 詹英,吴春明,王宝军. 一种与缓冲区紧耦合的环形循环滑动窗口的数据流抽取算法[J]. 电子学报,2011,39(4):2262-2267

[8] 崔贯勋,李梁,王柯柯,等. 关联规则挖掘中 Apriori 算法的研究与改进[J]. 计算机应用,2010,30(11):2952-2955

[9] 詹英,吴春明,王宝军. 基于 RCSW 的数据流速度异常检测算法研究[J]. 电子学报,2012,40(4):674-680

[10] 吴枫,仲妍,吴泉源. 基于增量核主成分分析的数据流在线分类框架[J]. 自动化学报,2010,36(4):534-542

[11] Tang L, Tian L F, Steward B L. Classification of broadleaf and grass weeds using gabor wavelets and an artificial neural network[J]. Transactions of the Asae,2003,46(4):1247

[12] Pfahringer B, Holmes G, Kirkby R. New options for hoeffding trees[M]// AI 2007: Advances in Artificial Intelligence. Springer,2007:90-99

[13] Gama J A O, Rocha R, Medas P. Accurate decision trees for mining high-speed data streams[C]// Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,2003:523-528

[14] Hashemi S, Yang Y. Flexible decision tree for data stream classification in the presence of concept change, noise and missing values[J]. Data Mining and Knowledge Discovery,2009,19:95-131

[15] Bifet A, Holmes G, Kirkby R, et al. Moa: Massive online analysis [J]. The Journal of Machine Learning Research,2010,99:1601-1604

[16] Oza N C. Online bagging and boosting[C]// 2005 IEEE International Conference on Systems, Man And Cybernetics. IEEE,2005:2340-2345

[17] Bifet A, Holmes G, Pfahringer B, et al. New ensemble methods for evolving data streams[C]// Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining. 2009:139-148

[18] Bifet A, Gavalda R. Learning from time-changing data with adaptive windowing[OL]. http://www.lsi.upc.edu/~abifet/TimevaryingE.pdf

[19] 王柯柯,崔贯勋,倪伟,等. 基于单元的快速的大数据集离群数据挖掘算法[J]. 重庆邮电大学学报:自然科学版,2010,22(5):673-677