近似概念格及其增量构造算法研究 *

林春杰[1,2], 普杰信[1], 张瑞玲[2]
(1. 河南科技大学 电子信息工程学院, 河南 洛阳 471022; 2. 洛阳师范学院 信息技术学院, 河南 洛阳 471022)

摘 要: 针对传统概念格处理不完备信息的局限,给出了处理形式背景有缺值现象的概念格扩展模型——近似概念格,在此基础上提出改进的概念格增量构造算法。该算法引入哈希技术和最近父节点的增量计算方法,从加速定位生成元和更新边这两个关键过程改进 Godin 算法。采用随机数据集设计实验,实验表明,改进的算法可有效提高对形式背景有缺值现象概念格的建格效率,尤其是对数据规模和发生关系概率较大的数据集,算法的高效性更明显。

关键词: 近似概念格; 形式概念分析; 不完备形式背景; 增量构造算法
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2012)01-0025-03
doi:10.3969/j.issn.1001-3695.2012.01.006

Approximation concept lattice and incremental constructing algorithm

LIN Chun-jie[1,2], PU Jie-xin[1], ZHANG Rui-ling[2]
(1. *College of Electronic & Information Engineering, Henan University of Science & Technology, Luoyang Henan* 471022; 2. *Academy of Information Technology, Luoyang Normal College, Luoyang Henan* 471022, *China*)

Abstract: The classic concept lattice is limited in incomplete information. In order to solve this limitation, presented a new concept lattice model-approximation concept lattice, witch could be used to deal with missing-value in formal context. On that basis, it designed an improved incremental constructing algorithm based on hash technique and incremental computation of nearest father nodes. Extensive experiments on the random data set demonstrate the improvements of the construction efficiency, especially for the data sets with large scale and density.

Key words: approximation concept lattice; formal concept analysis; incomplete formal context; incremental constructing algorithm

0 引言

概念格(concept lattice)[1]作为形式概念分析的基础数据结构,本质上反映了内涵和外延的统一,是一种具有极大潜力和有效的形式化工具,被广泛用于信息检索[2]、软件工程[3]、知识发现[4]、本体研究[5]等领域。经典的概念格基于完备的形式背景,而现实生活中,信息的非完备(对象的属性值缺损)现象是广泛存在的。一旦出现缺值,传统的处理方法有删除法、补全法和扩展属性法。然而删除法和补全法容易造成知识的缺失或增加,扩展属性法虽然能够保留原有形式背景的信息,但同时增加了形式背景的规模。因此,研究基于不完备形式背景的概念格模型具有重要的意义。

由于概念格节点数量是指数级的,导致构造过程非常耗时,所以概念格构造是形式概念分析在应用过程中首先要解决的一个基础性问题。概念格构造算法分为批处理算法和增量算法两类。批处理算法根据确定的数据一次建格,如 Bordat 算法[6]及其改进算法[7]。增量算法是将当前要插入对象的内涵与格中所有概念的内涵求交,根据交的不同结果进行不同操作,典型的增量算法有 Godin[8]算法等。批处理算法在形式背景稠密的情况下性能较好,但缺乏灵活性;增量算法能够有效地在原概念格基础上进行动态更新和维护。影响增量算法效率的关键是生成元的定位和新节点边的更新。针对该问题一些学者提出了改进的增量算法。文献[9]提出采用树结构对概念格节点进行组织,能够约束新生节点的父节点和子节点的搜索范围。文献[10]采用链表结构组织格节点,利用索引表,实现了对概念格子节点的快速查找。文献[11]提出通过跟踪最大概念来提高概念格构造效率。这些研究都不同程度地提高了建格效率。

本研究针对以上两个问题,定义了缺值背景及其相应的近似概念格,并在此基础上给出了概念格增量构造算法。该算法在定位生成元和更新边这两个方面对 Godin 算法进行改进。实验结果表明,该算法有效地提高了建格效率。

1 近似概念格基本概念

定义 1[1] 形式背景是一个三元组 $K=(G,M,I)$。其中:G 为对象集;M 为属性集;I 为 G 与 M 之间的二元关系,即 $I\subseteq G\times M$。$(x,y)\in I$ 表示对象 x 具有属性 y。

定义 2[1] 设形式背景 $K=(G,M,I)$,对集合 $A\subseteq G$,记 $A'=\{y\in M\mid \forall x\in G,(x,y)\in I\}$,相应地,对集合 $B\subseteq M$,记 $B'=\{x\in G\mid \forall y\in M,(x,y)\in I\}$。如果 $A'=B$ 且 $B'=A$,则称二元组

收稿日期: 2011-06-11; **修回日期**: 2011-07-25 **基金项目**: 国家自然科学基金资助项目(61050004);河南省重大科技攻关项目(102102310058);河南省基础与前沿项目(082300410270)

作者简介: 林春杰(1981-),男(朝鲜族),硕士,主要研究方向为粗糙集、概念格等(lynclcj@126.com);普杰信(1959-),男,教授,博士,主要研究方向为图像处理、模式识别;张瑞玲(1964-),女,教授,硕士,主要研究方向为粗糙集、概念格、电子商务.

(A,B)为形式背景 K 的形式概念,A、B 分别称为形式概念(A,B)的外延和内涵。

对于任意的 $x\in G,y\in M$,要么$(x,y)\in I$(记为$(x,y)=1$),要么$(x,y)\notin I$(记为$(x,y)=0$)。但是,如果存在某个对象 x 与某个属性 y 的关系不确定,则这样的关系对(x,y)称为缺值对(记为$(x,y)=^*$),并且认为它可能属于 I。

定义 3　I^* 为 G 与 M 之间的不确定关系,则$(x,y)\in I^*$ 当且仅当$(x,y)=1\wedge(x,y)=^*$,$(x,y)\notin I^*$ 当且仅当$(x,y)=0$,相应地,形式背景 $K=(G,M,I^*)$被称为缺值形式背景。

定义 4　设缺值形式背景 $K=(G,M,I^*)$,集合 $A\subseteq G$,记 $A'=\{y\in M\mid \forall x\in G,(x,y)\in I^*\}$,相应地,对集合 $B\subseteq M$,记 $B'=\{x\in G\mid \forall y\in M,(x,y)\in I^*\}$。如果 $A'=B$ 且 $B'=A$,则称二元组 $C=(A,B)$为形式背景 K 的近似概念。近似概念 C 的近似度 $\alpha_C=|POS(C)|/|A|$,其中 $POS(C)=\{x\mid x\in X,\forall y\in Y,(x,y)=1\}$。

近似概念的近似度描述了概念的不确定程度,并且 $0\leqslant\alpha_C\leqslant 1$。

定义 5　近似概念 $C_1=(X_1,Y_1)$和 $C_2=(X_2,Y_2)$,如果 $X_1\supseteq X_2$($Y_1\subseteq Y_2$),那么称 C_1 为 C_2 的父概念,C_2 为 C_1 的子概念,并记为 $C_1\geqslant C_2$。

显然,关系≥是缺值形式背景的所有概念集合上的一个偏序关系,它所诱导出的格称为近似概念格。

近似概念格基于偏大近似的思想,使概念描述的外延能够表示出某种属性“可能”覆盖的对象,是补全法的一种扩展。如果形式背景是完备的,近似概念格将退化为经典概念格,可以证明它是一个完备格。

2　增量构造算法

2.1　算法的基本思想

增量算法的基本思想是:将当前插入对象的内涵与格中所有概念的内涵相交,然后根据相交的结果采取不同的操作。算法将新概念格中的节点分为三类[9]:不变节点、更新节点和新生节点。算法采用内涵升序遍历概念格,为了叙述方便,假定当前遍历的概念为 $C=(A,B)$,新增对象为 x,$\{x\}'$表示对象 x 具有的属性,对不同类型的概念定义如下:

定义 6　如果概念 $C=(A,B)$满足 $B\subset\{x\}'$,则称该概念为更新概念,概念 C 更新为 $C^*=(A\cup\{x\},B)$。

定义 7　如果概念 $C=(A,B)$满足 $B\cap\{x\}'\neq B$ 且 $B\cap\{x\}'\neq\varnothing$,对概念 C 的任意父概念 $C_F=(A_F,B_F)$若满足 $B_F\cap\{x\}'\neq B\cap\{x\}'$,则概念 C 称为生成元,同时生成新概念 $C^*=(A\cup\{x\},B\cap\{x\}')$。

定理 1　若当前概念 $C=(A,B)$对任意新生概念或更新概念 $C^*=(A^*,B^*)$,有 $B^*\neq B\cap\{x\}'$成立,则概念 C 为生成元。

证明　用反证法,若 C 不是生成元概念,则必然存在 C 的一个最大的父概念 $C_F=(A_F,B_F)$,且 $B_F\cap\{x\}'=B\cap\{x\}'$。若 $B_F\subseteq\{x\}'$,则 C_F 为更新节点,若 $B_F\neq B_F\cap\{x\}'$,则 C_F 为生成元概念,且生成新概念 $C_{new}=(A_F\cup\{x\},B_F\cap\{x\}')$,与已知条件矛盾,证毕。

显然,新生概念与生成元存在一一对应的关系,因此,生成新概念的关键是找到所有生成元。由定理 1 可知,新生概念仅与新近生成的概念和更新概念有关。因此,生成概念时搜索范围可以从所有大于 $C=(A,B)$的概念缩小到所有新生概念和更新概念。由于定位生成元的过程中需要多次遍历新生概念和更新概念,为了进一步提高定位生成元的效率,改进算法采用哈希表存储新生概念和更新概念。

哈希表是一种重要的数据结构,它将关键字空间映射到 m 个从 0 开始的连续整数集合中。查找元素时,根据关键字的值,通过哈希函数计算元素地址来直接访问元素。与其他查找方法相比,无须任何比较,做一次运算便可得到所查询的元素,其查找时间复杂度从 $O(2^{|B|})$下降到了 $O(1)$。本文采用的哈希函数定义如下:$h(B)=\sum_{x\in B}2^{k_x}\bmod m$,其中 k_x 为属性 x 的编号。

定理 2　若概念 $C^*=(A^*,B^*)$为当前的新生概念,则 C^* 的直接父概念为更新概念或新生概念。

证明　设该新概念 C^* 的生成元为 $C=(A,B)$,则 $A^*=A\cup\{x\}$,$B^*=B\cap\{x\}'$,设 $C_F=(A_F,B_F)$为 C 的一个直接父概念,则 $B_F\subseteq B^*$,即 $B_F\subseteq\{x\}'$,所以由定义 6 和 7 可知,概念 C_F 必为更新概念或新生概念,证毕。

更新边即找到新生概念的所有父概念和子概念,并删除新生概念与其子概念指向共同父节点的边。由定理 2 可知,新生概念的父概念必然在新生概念和更新概念中;新生概念的子概念为与其对应的生成元。同样,在寻找新生概念的父概念过程中,需要反复遍历新生概念和更新概念,确定它们之间的父子关系,所以改进算法利用增量式的最近父节点计算方法,在生成新概念和更新概念的同时计算拥有不同内涵数量的概念相应的父概念的集合。生成概念时,只需搜索与其内涵数量相对应的父概念,从而大大缩小了搜索父概念的范围。最后从生成元中删除与新生概念指向共同父概念的边,这便完成了边的更新。

2.2　算法描述

以上介绍了增量构造算法的基本思想,在实现过程中,若概念格 L 为空格,则 L 中只包含($\{\}$,$\varnothing$)和($\varnothing$,$\{\}$)两个概念。更新概念格过程中,首先创建哈希表 H,然后按照内涵升序遍历概念格 L。在遍历过程中,将对象 x 依次与 L 中概念求交,根据交的结果以及在哈希表中的搜索结果,识别节点类型,执行相应的操作,如表 1 所示。同时,将得到的更新概念和新生概念插入哈希表和最近父节点链表中。

表 1　当前遍历概念为(A,B)、待插入对象为 x 时对概念格的操作

操作类型	更新概念	插入概念
判定条件	$B\cap\{x\}'=B$	Hash_Search(H, $B\cap\{x\}'$)$=\varnothing$
生成概念	$(A\cup\{x\},B)$	$(A\cup\{x\},B\cap\{x\}')$
父节点	不变	$\{C\mid C\in F[\,|B\cap\{x\}'|\,]\wedge C\subset B\cap\{x\}'\}$
子节点	不变	(A,B)

概念格节点 C 的数据结构设计含有如下数据:intention 用于存储节点的内涵集,extension 用于存储节点的外延集,parents 用于存储父节点集合,children 用于存储子节点集合,precision 用于表示节点的近似度。用 intention(x)表示对象 x 的内涵。算法的详细描述如下:

算法 1　增量计算最近父节点算法

输入:当前更新概念或新生概念 C,最近父节点集合 F。

输出:更新后的最近父节点集合 F^*。

```
CalNFConcepts (C,F)
1  for i←|C.intention| to MaxIntentCount
```

```
2  for each C′∈F[i]
3      if C. intention⊃C′. intention then
4          从 F[i]中删除 C′
5      else if C. intention⊂C′. intention
6          算法结束
7  F[i]←F[i]∪C
```

算法 2　改进的增量构造算法

输入:已有概念格 L 和新增对象 x。

输出:更新后的格 L^*。

```
ConstructLattice (L,x)
1  F←(U,∅), 初始化哈希表 H(h_i. count =0)
2  for each C∈L do
3    Intersection←C. intention∩Intention(x)
4    if Intersection = Intention(x) then
5        C. extension←C. extension∪{x}//更新节点
6        CalNFConcepts (C, F)
7        Hash_Insert(H, C. intention)
8        if C. intention = Intersection then
9            算法结束
10   else if Intersection≠∅ then
11   if Hash_Searcht (H, Intersection) = ∅ then
12     创建新建概念 C_new
13     C_new. extension←C. extension∪{x}
14     C_new. intention←Intersection
15     添加 C_new到 C 的边
16     for each C_F∈F[|C_new. intention|] do
17         if C_new. intention⊂C_F. intention then
18             添加 C_F. 到 C_new的边
19     删除 C 中与 C_new指向相同父概念的边
20     CalNFConcepts (C_new,F)
21     Hash_Insert(H, C_new. intention)
```

3　算法实验分析

为了进一步验证算法的有效性,对提出的算法与 Godin 算法和文献[11]的 MaxLink 算法在 Galicia(http://www. iro. umontred. ca/ ~galicia)生成的随机数据集上进行了实验。在实验中,选定对象数为 1 000,属性数为 15,属性和对象的关系概率分别为 0.2、0.33 和 0.5,数据缺失率为 10%,从而生成三个形式背景。实验将对象分为 10 组,每组 100 个对象,每次渐进式地插入 1 组对象,并记录运行时间。实验环境是 CPU Intel Core 2 1.80 GHz,内存为 1GB,算法采用 C++程序实现,开发环境为 Microsoft Visual Studio 2005。实验结果如图 1 ~3 所示。

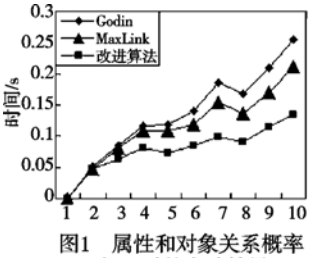

图1　属性和对象关系概率为0.2时的实验结果

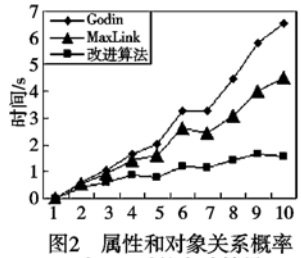

图2　属性和对象关系概率为0.33时的实验结果

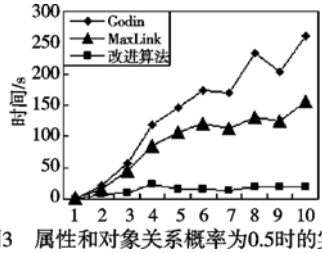

图3　属性和对象关系概率为0.5时的实验结果

图中水平轴表示对象组的顺序编号,垂直轴表示将该组对象插入到概念格中消耗的时间。如图 1 ~3 中实验结果所示,随着对象数量的增加,改进算法和其他两种算法的计算时间差值会增大,这与格节点的数量有很大关系。图 1 中三种算法的计算时间比较接近,这主要是由于属性和对象关系概率较低时概念格节点数量也较少。图 2、3 中 Godin 和 MaxLink 算法的计算时间随着对象数量的增加而显著增加,但新算法的耗时增加的幅度较小,所以实验结果表明改进算法不仅有效地加速了概念格构造过程,而且对规模和对象属性关系概率较高的数据集,改进算法的性能会更好。

4　结束语

针对经典概念格处理不完备信息的局限性,构建了近似概念格,使其能够描述不完备信息。并在此基础上给出了改进的概念格增量构造算法。分析影响概念格构造过程中的关键因素,考虑的重点在于缩小搜索空间、降低算法的时间复杂度,算法从加速定位生成元和更新边这两个关键过程改进 Godin 算法,得到高效的概念格增量构造算法。实验结果表明,算法具有较高的效率,适合处理具有缺值现象的大规模数据集下概念格构造问题。由于概念格的对偶性,该算法思想可以应用于基于属性的概念格增量构造算法。另外,基于近似概念格的知识获取及应用是下一步研究的工作。

参考文献:

[1] WILLE R. Restructuring lattice theory: an approach based on hierarchies of concepts[M]//Proc of the 7th ICFCA'09. Berlin: Spninger-Verlag, 2009: 314-339.

[2] YADAV B S. A conceptual model for user-centered quality information retrievalon the world wide Web[J]. Journal of Intelligent Information Systems, 2010, 35(1): 91-121.

[3] TONELLA P. Formal concept analysis in software engineering[C]// Proc of the 26th International Conference on Software Engineering. Washington DC: IEEE Computer Society, 2004: 743-744.

[4] POELMANS J, ELZINGA P, VIAENE S, *et al*. Formal concept analysis in knowledge discovery: a survey[C]//Proc of the 18th International Conference on Conceptual Structures. Berlin: Springer-Verlag, 2010: 139-153.

[5] FORMICA A . Ontology - based concept similarity in formal concept analysis[J]. Information Sciences, 2006, 176(18): 2624-2641.

[6] 陈庆燕. Bordat 概念格构造算法的改进[J]. 计算机工程与应用, 2010, 46(35): 33-35, 38.

[7] LINDIG C. Fast concept analysis[M] // STUMME G. Working with Conceptual Structures- Contributions to ICCS2000. Aachen: Shaker Verlag, 2000: 152-161.

[8] GODIN R. Incremental concept formation algorithm based on Galois (concept) lattice[J]. Computational Intelligence, 1995, 11(2): 246-267.

[9] 谢志鹏, 刘宗田. 概念格的快速渐进式构造算法[J]. 计算机学报, 2002, 25(5): 490-495.

[10] 蒋义勇, 张继福, 张素兰. 基于链表结构的概念格渐进式构造[J]. 计算机工程与应用, 2007, 43(11): 78-180.

[11] 余远, 钱旭, 钟锋, 等. 基于最大概念的概念格增量构造算法[J]. 计算机工程, 2009, 35(21): 62-64.