·综述 ·评论 ·争鸣 ·

数据挖掘中分类算法的可扩展性研究

乔向杰[1,2],陈功平[2]

(1. 北京科技大学 信息工程学院,北京 100083; 2. 信阳师范学院 计算机科学系,河南 信阳 464000)

摘 要:分类是数据挖掘中最重要的技术之一,而且应用领域非常广泛,但面对新出现的海量数据,目前已有的许多分类算法不具备良好的伸缩性,不能从巨大的数据集中快速而准确地发现有用的知识. 针对这一问题,本文对分类算法的可扩展性方法进行了深入的研究,并对各种方法进行了分析和对比,从而便于研究和开发者对已有的算法进行改进和扩展,以适应数据挖掘技术的不断发展.

关键词:分类; 大数据集; 可扩展性

中图分类号: TP312 **文献标识码**: A **文章编号**: 1003-0972(2006)02-0239-04

0 引言

数据挖掘(data mining)又称数据库中的知识发现(KDD),是当今众多学科领域特别是数据库领域最前沿、最活跃的研究课题之一. 在数据挖掘算法中,分类(classification)是具有广泛应用领域的最重要的问题之一,它是发现属于同一类的数据对象的共同特性的过程,其目的是通过分析训练数据集的特点构造一个准确的分类器,该分类器可用于对未知类别的样本进行类别的判断[1]. 分类问题虽已在其他领域进行了很多的研究,但面对新出现的海量数据,包含数以百万计样本的非常大的训练集是很常见的,如何快速而准确地发现隐藏于其中的主要分类规则,即算法的有效性和可伸缩性,仍是一个值得研究的问题. 本文针对这一问题,对目前已有算法的可扩展性方法进行了总结和分类,并对各种方法进行了分析和对比,从而为在实际应用中选择高效的挖掘方法以及算法本身的改进提供了参考和依据.

1 算法的可扩展性

Raju Addala[2]认为,算法具有良好的可扩展性是指算法具有处理大量的数据或加速数据挖掘过程的能力. 我们认为,能从大数据集中快速而准确地发现隐藏于其中的主要分类规则,即认为算法具有良好的可扩展性. 在这里,我们主要强调大数据集.

进行可扩展性研究的原因主要基于以下几点:一、增加分类的准确性和有效性. 这也是对分类算法进行扩展的主要原因. 二、算法对空间的要求. 大部分决策树算法如ID3和C4.5都限制训练样本驻留内存,由于训练样本在主存和高速缓存之间换进换出,算法就可能变得效率低下. 三、从算法的时间复杂度来看,关键因素也在于样本的数量以及每个样本的属性数和每个属性的值的数目. 尤其是基于关联规则的分类算法往往要多次扫描整个数据库,因此数据库大小就成为最主要的影响因素. 四、为了发现小事件情况(special cases). 由于噪音数据的存在,为了避免将小事件错认为假事件和过适应(over-fitting)问题,必须将数据集增大到足够多的样本,以便于准确有效地发现小事件.

2 可扩展性方法

Raju Addala将可扩展性方法分为面向算法的(Algorithm-oriented)方法,即对算法本身进行优化,以减少算法复杂度、优化搜索和并行处理等. 以及面向数据的(Data-oriented)方法,即对数据集进行分解,将算法应用到每一个数据子集,然后将结果组合到一起形成可理解的有用的知识的方法. 我们在这里采用Foster J. Provost的分类方法,即分为设计快速的算法(design a fast algorithm)、数据分割(partition the data)和关系性表达(use a relational representation).

2.1 设计快速的算法

设计快速的算法是一个能减少计算量和扫描数据库次数的最直接的扩展方法. 对于具有线性复杂度的大数据集来说,快速的算法在实际的数据挖掘中可能是远远不够的,但快速算法和其它的扩展方法结合起来使用将会是非常有效的. 这一方法的主要实现技术包括限制模型空间、采用启发式搜索、优化算法的编程以及并行处理等.

2.1.1 限制模型空间

这种方法的典型应用是线性回归分类方法、简单神经元方法和单层决策树(decision stumps)方法. 由于模型空间小,这些简单的学习算法都能很快地学习,比如K-最临近分类方法的时间复杂度为O(1),但一旦数据集增大,这种线性复杂度的优势就会立即消失.

收稿日期: 2005-01-16;**修订日期**: 2005-09-20

作者简介:乔向杰(1977-),女,河南信阳人,在读博士,主要研究方向:数据挖掘,人工心理.

2.1.2 启发式搜索

决策树分类的贪心、分而治之的方法都是启发式搜索的成功应用 . 当只考虑非连续性数据时, ID3(C4.5)的时间复杂度是随着样本数线性增长的,为 $O(ea^2)$,其中 e是样本数, a是属性数 . 但是由于连续性数据需要反复排序,因此最终的复杂度超过了 $O(e^2)$.

由于用贪心启发式算法构造的决策树缺乏可理解性,通常我们用易于理解的规则来表示知识,为了提高在产生规则时的准确性,最一般的方法是采用减少错误剪枝法 (reduced-error pruning). 但这种方法的可伸缩性不是很好,比如在 C4.5中,时间复杂度达到了 $O(e^3)$. Fürnkranz 和 Widmer[2]证明用增量减少误差修剪算法 (incremental reduced error pruning, IREP)能够提高速度,其计算复杂度估计为 $O(e\log^2 e)$. 但用 IREP进行学习的准确性比 C4.5要低,所以 Cohen又通过进一步细化不同规则的评估原则、不同停止准则和预处理优化来改进 IREP算法,提出了 RIPPER算法 . RIPPER算法在准确性上能同 C4.5相竞争,而且维持了 IREP的时间复杂度 $O(e\log^2 e)$.

Domingos提出了不分而治 (conquering without separating)方法,相对于分而治之 (divide and conquer)方法大大提高了分类的准确性 . 在分而治之方法中,首先要对数据集分区,分区的原则是一次基于一个具有最大信息熵的属性变量,然后循环进行属性分枝 . 由于分区后的子节点只覆盖了初始数据集的一个子集,因而很可能会丢掉原始训练集中有用的信息资源从而对分类结果的准确性有所影响 .这种方法往往对挖掘具有大量多值属性 (Many-Valued Attributes)的数据集分类结果不准确 . 比如,对如表 1中的训练集,通过运行 C4.5,我们得到了图 (1)中的树,这是一棵很好的树,因为每一个叶子都覆盖了是同一类别 . 但是 C4.5没有发现规则 r1: "IF <Humind = high> THEN No". 这是因为 r1已经被更一般的规则 r_2: "IF <Temp = hot> THEN No"所覆盖,这是选择了 Temp作为分区变量的原因 . 因此当我们分类新的样本 (cool, high)时, C4.5就用最右边的叶子节点并分类为 "Yes",但在训练集中, "IF <Humind = high> THEN No"的样本要多于 "IF <Temp = cool> THEN Yes",因此,应将其分类为 "No".

表 1 训练集

Tab.1 Training set

TEMP	HUMID	PLAY?
mild	dry	yes
mild	normal	yes
mild	dry	yes
hot	high	no
hot	high	no
hot	high	no
hot	high	no
hot	dry	no
hot	normal	no
cool	dry	yes

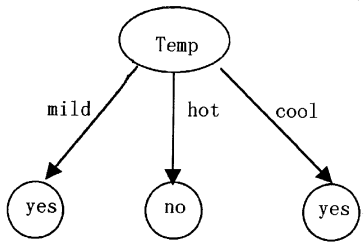

图 1 C4.5生成的树

Fig.1 Decision-tree using C4.5 algorithm

而 Domingos提出的不分而治的方法要求每个规则的归纳产生都必须来自整个数据集,也就是说,在任何时间都不进行分区,因而其产生的规则就更全面准确,被称为全局规则 (Global rules)[3],而分而治之算法产生的规则由于每次都基于一个子集,因而称为局部规则 (local rules).

2.1.3 优化算法编程

可通过采用高效的数据结构,如位向量 (bit vectors)、哈希表 (hash tables)及二叉查找树 (binary search trees)来进行算法的优化和扩展 .

决策树分类中,对于连续属性来说,在每个内部结点寻找其最优分裂标准的时候,都需要对训练集按照该属性的取值进行排序,而排序是个很浪费时间的操作 . 为此 SLIQ[4]算法采用了预排序的技术,以便能够消除在决策树的每个结点对数据集进行排序的需要; SLIQ采用广度优先策略构造决策树,即在决策树的每一层只需对每个属性列表扫描一次,就可以为当前决策树中每个叶子结点找到最优分裂标准,这也使得它能对驻留磁盘的数据集进行分类,从而克服了将整个数据集装入主存的局限性; SLIQ定义了新的数据结构以利于树的构造,它使用若干驻留磁盘的属性表和单个驻留主存的类表 . 但由于类表的大小随训练集中元组数目成比例增长,当类表不能放在主存时, SLIQ的性能下降 . 当所处理的驻留磁盘的数据太大,不能一次装入内存时, SLIQ的可伸缩性受限于它所使用的常驻内存的数据结构,而 SPRINT算法消除了所有的内存限制,因为它不使用整块的驻留在内存的数据结构 .

2.1.4 并行处理

任何一种独立的数据挖掘方法无论多么有效,它的时间复杂性至少是 $O(N)$,其中 N是数据库的规模,而并行性提供了一个算法的时间复杂性为 $O(N/P)$的机会,这里 P是可以利用的处理器的总量 .

大多数据挖掘任务都是可分解的,因此就可以利用并行处理来提高算法的运行速度 . 并行处理通常包括对搜索空间的并行处理 (search-space parallelization),使用共享内存和多处理器来并行搜索不同的空间,和并行匹配 (parallel matching),搜索空间中的每个节点 (决策树中的分枝)都要在样本中进行匹配,并对该节点进行生成和测试,因而对节点的评估代价很高,我们可以对其进行并行处理以实现快速学习和分类 .

目前有两种方法被用在决策树的并行化中[5]:第一种方法是使用并行的数据库管理系统作为后端,在数据库服务器中采用了这种方法.在这个系统中,他们将数据库划分成为多个二元关系表,同时通过并行的数据库查询过程,在一定程度上实现了数据挖掘的并行性.第二种方法是在元学习(meta-learning)中采用分而治之的策略.先将数据集根据一些属性分区成几个子集,然后在各个子集中再同时应用决策树算法,最后将对各个子集操作后的结果进行综合,得到最终结果.而实际上我们也可以在选择当前节点的最优属性对数据集进行分割的这一环节中将对各个属性的信息增量比的计算同步进行,实现并行操作.

并行处理方法的缺点是它需要大量的并行处理器或处理机,硬件实现较困难.同时,我们在对某种并行方法或综合方法是否有效不能脱离数据(数据的特征及数据的总量)来孤立地讨论,更不能单纯追求使用大量的处理器.因为无论是将数据载入各个处理器上还是将各结果综合或输出都是一项复杂而繁重的任务,它们都需要消耗大量的时间,而且输入/输出的耗时虽然最初随着处理器数量的增加而减少,但是最终决策树建立的时间消耗总量并不是理想中的一条单调下降的曲线,而是一条先下降后上升的不光滑曲线,并且随着处理器数量的增多处理中间变量所需要的空间也不断增大.

2.2 数据分割

通过分割数据以避免算法在整个数据集上运行,是解决在大数据集上进行数据挖掘的一个简单有效的和健壮的方法.系统在样本选择程序基础上选择一个或多个子集 $S_1, \cdots, S_n$,学习算法 $L_1, \cdots, L_n$ 分别在相应的子集中运行,并产生分类法 $C_1, \cdots, C_n$,然后这些分类法通过组合程序或者从 $C_1, \cdots, C_n$ 中选取或者将它们组合起来最终产生一个分类法 C.

我们如果将数据集看成一个表,抽样就可通过选择行即实例抽样(Instance sampling)和选择列即特征抽样(Feature sampling)来分类抽样过程.对于多个子集的处理,还要选择是采用串行处理(Processing samples sequentially)还是并行处理(Processing samples concurrently).

实例抽样采用的策略通常有随机抽样和策略抽样.策略抽样主要用于类分布不均匀的训练集中,通过频繁地选择较少的类样本来平衡分配.Catlett发现,在决策树分类中,由于必须要对连续属性进行排序,所以主要的计算任务就在于寻找连续属性的分裂值.而通过在样本子集中寻找连续属性的分裂值决策树算法的时间复杂度由随着样本数呈二次方增长降为线性增长,而且不影响准确度.

进行特征抽样主要基于两个原因[3]:一是随着属性集的增长,算法出现过适应的几率也会增大,尤其是在小训练集中,如果我们能够选择一个好的属性子集通常就能增加算法的准确性.另一个原因是属性的数量也是算法执行时间复杂度的主要因素,而且这种复杂度通常不是随属性数线性增长的.特征选择的方法主要有采用现有相关知识,一方面可以根据领域专家的指导,筛选掉对于决策用处不大的属性;另一方面对每一个能够描述问题的属性变量而言,它们通常都有一个辅助的数据库来提供相关信息,比如一个邮政编码域就可能同一个非常大的有关人口统计的数据库相联系.在实践中,要选择一个域,该域就必须具有极大的相关性.另一个特征选择的方法就选择尽可能多的属性,然后再从中筛选出一个子集.即先计算单个属性对于目标规则的相关性,然后从这些属性中进行筛选和组合而得到一个实用的、具有高度相关性的属性子集.

数据分割法虽然可以用于大数据集的分类,但是其分类的准确性却不如一次使用所有的数据的方法高,而且不利于发现不频繁的小事件.

2.3 关系性表达

大多数据挖掘系统的设计都是假设数据集能够用一个简单的、驻留磁盘的表来表达,但在现实应用中,多表数据库是非常常见的,它不仅能充分表达单表中各属性之间的关系、属性值之间的关系及元组之间的关系,而且还包括各个表之间的关系.同时,多表操作往往需要大量的时间和空间需求,数据库系统通常能很好地进行维护和管理数据及数据之间的关系,而且有非常好的存储管理和数据恢复机制.

2.3.1 将 KDD同 DBMS整合

数据挖掘算法的计算代价往往都比较高,比如在决策树算法中,必须频繁地扫描数据库,并且还要依赖于数据的特性和值的分配等等,这些任务都需要大量的磁盘 I/O操作,而利用数据库技术就能很好地解决这一复杂计算及 I/O问题.在这种方法中,时间主要消耗在向 DBMS发送一系列的 SQL查询,而基本的搜索过程是由一些编程语言来实现的.

如何将分类算法与数据库技术结合起来,已成为现今该领域研究的关键问题之一.Agrawal和 Shim提出三种将挖掘算法与数据库技术结合使用的架构:几乎不使用数据库(decoupled)、稀疏结合(loosely-coupled)和紧密结合(tightly-coupled).

当前大多数据挖掘系统都采用的是第一种架构,即不利用 DBMS.他们提供自己的存储管理,这种方法的优势是能够根据特殊的挖掘任务自由调整其存储管理,这样系统的性能就能得到优化.缺点是没有充分利用数据库的成熟技术,数据库管理系统除了具备良好的存储管理能力,而且它的其它许多功能对数据挖掘过程也是非常有用的,比如,大多数 DBMS系统提供的恢复机制和日志机制,以及其并发控制能力还允许不同用户利用相同数据备份并同时进行数据挖掘查询等等.

有些数据挖掘系统也使用了 DBMS,但仅仅是用它来存储和恢复数据,这种系统采用在主编程语言中嵌入 SQL查询,用 SQL选择语句来从数据库中恢复感兴趣的记录集,这样主要的计算都耗费在将结果记录集从数据库的地址空间拷贝到应用程序的地址空间中,因此运行效率较

低.

同数据库紧密结合的架构充分利用了数据库技术的优势,将数据挖掘的部分计算任务交给数据库系统来完成,避免了应用程序对训练集的多次顺序扫描和大量的数据传送,从而提高了算法执行的速度.许多数据挖掘系统都已采用这种方法来实现,比如 DBMiner.MinWang等人也提出了用 UDF(user-defined function,用户自定义函数)实现决策树分类的算法 MIND,但由于 UDF并不是数据库系统本身提供的函数,其功能仍然由用户编程实现,没有用到数据库的查询和处理功能,因此性能的提高受到限制.WesleyW. Chu等人也在标准 DBMS上实现了 KDS算法[7],他主要是通过一系列复杂查询的执行和 UDF调用来完成的.刘红岩等人也利用数据库技术实现了可扩展的分类算法 GAC-RDB[1],巧妙地利用关系数据库管理系统提供的聚集计算功能,将原来需编程实现的复杂的绝大部分操作用 SQL语句实现,完成各种类别分布的统计工作,从而极大地提高了算法的运行效率.

2.3.2 分布式数据挖掘

本质上,分布式数据挖掘包含 3步[8]: (1)划分待挖掘数据成 p个子集, p为可用的处理器个数,并把每个数据子集发送到各个处理器; (2)每个处理器运行数据挖掘算法于其局部数据子集,处理器可以运行不同的数据挖掘算法; (3)组合各个数据挖掘算法发现的局部知识成全局、一致的知识.类似于抽样方法,分布式处理也是以预测精度换取速度.

3 总结

以上所给出的各种方法之间并不互相排斥,因此在实际应用中,我们可根据情况同时选择多种方法来扩展算法,提高算法的可伸缩性.而如何将各种扩展技术进行整合,以达到最优的扩展性能,也是我们今后所研究的一个主要方向.

参考文献:

[1] LIU Hong-yan, LU Hong-jun, CHEN Jian. *A Scalable Classification Algorithm Exploring Database Technology*[J]. Journal of Software, 2002, 13(6): 1075-1081.

[2] FUMKRANZ J, WIDMER G. *Incremental reduced error pruning*[C]// *Machine Learning: Proceedings of the Eleventh Annual Conference*. New Brunswick, New Jersey, 1994. Morgan Kaufmann.

[3] LIU B, HSU W, MA Y. *Integrating classification and association rule mining*[C]// *Agrawal, R, ed. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. New York: AAAI Press, 1998, 80-86.

[4] Han Jiawei, MICHELINE Kamber. *Data Mining: Concepts and Techniques*[M]. Beijing: China Machine Press, 2001.

[6] ZHANG Xiao, YUN Shuang, LU Sanglu, etc. *Invesigation of Parallel Data Mining*[J]. Computer Engineering, 2003, 29(17): 58-59.

[7] CATLETT J. *Megainduction: a Test Fligh*[C]// *Proceedings of Eight International Workshop on Machine Learning*. Morgan kaufmann, 1991: 596-599.

[8] ZHANG Xueming, SHI Fazhong. *A Data Mining Method based on VisiBroker and Multi-thread*[J]. Computer Engineering and Applications, 2002(4): 198-200.

[9] 潘立武,王保保,李绪成.零售业销售数据关联规则挖掘算法关键思想研究[J].信阳师范学院学报(自然科学版), 2003, 16(1): 90-93.

Studies on Scalability of Classification Algorithm in Data mining

QIAO Xiang-jie[1,2], CHEN Gong-ping[2]

(1. School of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China;
2. Department of Computer Science, Xinyang Normal University, Xinyang 464000, China)

Abstract: Classification is one of the most important techniques in data mining, it has been largely applied in many fields. However, as the appearance of large amount of data, the existing classification algorithms do not scale well, they cannot mine useful knowledge from very large datasets quickly and accurately. This paper discusses scalability, gives an analysis and comparison of these algorithms. So that researchers and practitioners can scale up their existing algorithms to catch up with the development of data mining technology.

Key words: classification; large dataset; scalability

责任编校:郭红建