决策树的并行训练策略*)

刘欣阳　王国仁　乔百友　韩东红
(东北大学信息学院　沈阳110004)

摘　要　随着生物科学技术的发展，其数据量的增长也非常迅速，很难在一定合理的时间内对数据进行建模和分析，因此，对并行数据挖掘算法的研究已变成解决此问题的重要途径。决策树途径已被广泛用作一种重要的分类工具，本文研究了几种决策树的并行训练策略并对它们的性能进行了比较。

关键词　决策树，并行训练，数据挖掘

Parallel Strategies for Training Decision Tree

LIU Xin-Yang　WANG Guo-Ren　QIAO Bai-You　HAN Dong-Hong
(College of Information, Northeast University, Shenyang 110004)

Abstract　With the development of biology technology, the amounts of data increase very fast. It is difficult to analyze and model the data within reasonable amount of time, so studying parallel data mining algorithms has been becoming an important approach to solving the problem. Decision tree has been widely applied as an important classification tool. This paper surveys several parallel training decision tree strategies and compares their performance.

Keywords　Decision tree, Parallel training, Data mining

1　引言

蛋白质和DNA的功能预测是在功能基因组研究中最困难的问题，早期的工作发现，蛋白质和DNA的功能都与序列中的一些保守位点相关，现在很多基因数据库都提供这些位点的检索，但是多位点之间的相互作用对于基因功能的预测往往比单个位点精确和有效，因此利用数据挖掘技术对基因序列功能进行更精确的预测就变得很重要[1]。决策树，人工神经元网络和统计模型等许多机器学习算法都可以应用在计算生物学领域。决策树相对于其它学习模型建立模型的速度比较快而且容易理解[2]，而且决策树可以被转化SQL语句，这样就可以高效地访问数据库[3]，最后，决策树的分类精度要优于其他一些类分模型[4]。

决策树学习算法最早是1966年由Hunt等人提出的CLS算法[5]。1979年Quinlan提出了以信息熵的下降速度作为选取测试属性的标准的ID3算法[6]。现在的决策树分类系统基本都是基于以上的两种算法。比较著名的分类算法有C4.5[2]，CDP[7]，SLIQ[8]和SPRINT[9]。

训练集的大小直接影响了决策树的训练时间，早期的决策树分类算法只能处理一些比较小的训练集，然而生物信息学领域的数据量是非常大的，一般认为单条人类DNA序列在3G以上。现在普遍认为对大规模的数据进行学习可以提高分类的准确率而且可以发现一些比较不容易察觉的特殊情况。但是在catlett的论文中提到若是有一百万笔资料，用一台一般的工作站全力去跑一般的学习算法，那么通常至少需要数个月，才能真的学到一些东西或得到一些有意义的结果。这样的效率在实际应用上是无法接受的，因此需要一些其他的方法提升学习算法的效率。

Holte(1993)提出了使用结构简单而快速的学习算法，Holte本人则认为他的演算法比较适合处理资料库。由于构造简单化的限制，这个算法确实无法处理过于复杂的问题。另一种方法是简化资料法，把一些重复或者对学习不是很有帮助的资料剔除，留下精华，在catlet(1991a)的论文中，他测试了许多种简化学习资料的方法，可惜最后得到的结论是，这个方法不仅会降低正确度，同时也无法学到一些比较例外的特殊情况，如此就失去了当初将资料量变大的目的了，因此，简化资料的方法效能不佳。也可以降低描述语言的复杂度，这一招是降低描述各种范例资料所用的语言的复杂度。经过适当的分类，所有的范例资料复杂的程度可以有效地降低，这个方法也是由Catlett(1992,1991b)所提出，虽然在实际运作上有成效，但是最终学习算法的负荷仍然跟资料量大小成正比，最后在面对过于庞大的资料时，这个方法就变得无用武之地了。Qulian提出的增量式学习算法可以缓解内存不足的问题，但是它所得到的决策树的分类精度不高而且学习时间也不会减少，所以很难在合理的时间内处理大规模的数据。随着分散式系统和网络的发达，并行式计算已经被应用在许多领域，并行化决策树的学习过程既可以保证学习资料的规模也可以提高学习速度。

本文将一些并行方法进行了归纳和总结。第2部分介绍并行构建决策树的方法，第3部分介绍它们之间的比较，最后进行总结。

2　并行决策树的算法

目前决策树的并行策略很多，根据数据分片的不同方法可以分成动态数据分片(即根据任务进行分片)、静态数据分片和这两种方法的混合。

2.1　动态数据分片

在决策树分裂开始之前全部训练集都存储在主处理机

*)本文受国家自然科学基金项目资助(60273079)。

上。主处理机首先分裂根结点，假设将训练集划分为 m 个训练子集，主处理机将这 m 个训练子集分配给包括主处理机在内的 m 个处理机上，这 m 个处理机并行地构建决策树的子树。主处理机重复上面的过程，为没有任务的处理机分配训练集一直到所有的处理机都得到任务。这种方法是在文[10]中提出来的 STD 方法。

这种方法会产生严重的负载不均，假设上面所提到的 m 个子训练集大小相等，但是只能由主处理机进行数据分配，假设主处理机第二次分裂产生 n 个子训练集，那么这次包括主处理机在内的所有处理机的训练集仅仅为前面处理机的 $1/n$。

因此文[10]还提到了一种 DTD 方法来缓解这个问题。DTD 方法的初始状态和 STD 方法一样，不同的是所有的处理机都参与训练集的分配。主处理机保存一个空闲处理机队列，开始时空闲处理机队列为全部处理机。无论哪一个处理机产生分裂都可以向主处理机申请空闲处理机。某一个处理机完成任务就向主处理机报告，主处理机将这个处理机放入空闲处理机队列。

虽然 DTD 方法可以比 STD 方法效果好，但是它的负载不均还是很严重的。一方面是它需要大量的数据移动代价，另一方面是目前都采用互信息量作为训练集划分的方法，平均概率分布的互信息量是最小的。因此每次分裂所得到的训练子集的大小是很不均匀的。

2.2 静态数据分布

2.2.1 *横向划分* 在初始状态所有的处理机保存的训练集的大小相等。所有的处理机共同按照某一种扩展策略(深度优先或广度优先)完成任务。比较典型的算法是 ScalParC[11]。ScalParC 算法是对 SLIQ 和 SPRINT 的改进。在 SLIQ 算法中首先提出了两个新数据结构：属性列表和类列表。因为在传统的算法中在每个结点都要对连续属性进行排序，但是在决定分裂标准时各个属性之间是独立的。因此 SLIQ 算法将各个属性从训练集中抽取出来形成属性列表，这样就可以首先对连续属性排序以避免在每个结点都进行排序。属性列表的每一项由属性值和类列表索引组成，存储在磁盘上。类列表由类值和结点指针组成，由于类列表的访问是随机和频繁的，所以类列表一定要放在主存中。并行 SLIQ 采用两种方法，一种是将类列表复制到每个处理机上即 SLIQ/R，另一种采用分布式类列表。SPRINT 中提出将类列表去掉，这样 SPRINT 可以处理更大的数据集。它只是将属性列表的表项增加了一项类值。在分裂的时候形成记录到结点的哈希表。并行 SPRINT 从局部训练集形成各个属性列表。在对连续属性排序时使用了并行排序算法并进行连续属性重分片。但是每次分裂需要整个哈希表，因此每个处理机通讯量是 $O(N)$。ScalParC 算法采用并行哈希表技术可以使每个处理机的通讯量为 $O(n_a N/P)$，n_a 是属性的个数。

静态数据划分的并行方法不适合采用深度优先的扩展方法，因为每个处理机包含的待扩展结点的训练实例不相等。采用广度优先的策略，所有的处理机将当前层的所有结点的信息统计完以后进行同步通信，这样可以减少通信的启动代价以及各个处理机任务的负载不均，但是通信的总量没有减少，随着树的层数的增加通讯代价变得越来越大，严重影响了执行的效率，而且由于统计图表是驻留在内存中，同时扩展多个结点需要比较大的内存或者需要额外的 I/O 开销。

2.2.2 *纵向划分* 纵向划分也就是将完整的属性列表分配到各个处理机上。由于各个属性和类属性之间的互信息是独立计算的，所以各个处理机之间仅仅比较互信息量的大小而不用进行全局的统计工作，可以减少大量的通信代价。属性列表分裂可以采用一个哈希表，这个哈希表由分裂属性形成，然后广播到各个处理机。但是它的弱点同样是负载不均。在较高层次时所有的属性都需要计算，但是到了较低层次时一些属性已经不可能成为训练集的划分标准也就不需要进行计算，这样会有很多处理机处于闲置状态。而且个个属性的性质不同，例如连续型属性和非连续型属性的处理过程以及计算代价都不同。因此各个处理机完成任务的时间是有很大差别的。我们可以将各个属性的互信息量的计算的复杂程度进行评估，将比较难处理的属性列分片到几个处理机上以达到负载均衡。当出现多余的属性列时可以重新分配属性列表。

2.3 混合并行

首先将训练集平均分片。以横向划分的数据并行的方法各个处理机协作扩展结点。在决策树的高层通讯代价不会很大，当通讯量累计达到一定数量时将当前层的结点分成两部分，同时将处理机对应分成两组，每组对理机负责一部分待扩展结点，每组结应的训练实例数近似相等。这样可以减小负载失衡。为了确保每组的处理机有对应结点的全部训练实例，两个处理机组之间要进行数据交换。然后在组内进行数据平衡。每个组重复上面的步骤，一直到每个组只有一个处理机时通讯代价为零。最后将所有处理机上的子树合并为一棵决策树。

负载平衡方法。当某一组处理机完成任务时将这一组处理机加入处理机个数相等的处理机组，并将有任务组一半的数据分给加入组。

处理机分组机制。当累积通讯量等于组之间的数据交换代价加上组内数据平衡代价时进行处理机分组。即

$$\sum(CommunicationCost) >= MovingCost + LoadBalancing$$

这种方法比动态数据分配移动的数据少而且各个处理机上的训练实例数不会相差很大，当结点的数量超过处理机的个数时通讯代价变为零，所以通讯代价也相对较小。

前面提到的各种并行方法都是以整个训练集作为并行算法的输入，因此只要分裂标准相同它们得到的决策树就是相同的。在文[13]中提到了一种并行算法不是对整个训练集进行处理，我们称之为决策树合并。

2.4 决策树合并

首先将训练集平均分片，然后各个处理机以局部的训练集并行构造决策树，然后将各个处理机上的决策树合并为一棵决策树。这种方法在并行训练阶段没有通信代价，各个处理机建树的时间也大致相同，并行训练的过程只是在每个处理机串行地构造决策树。但是在后期的合并过程是很复杂的，而且合并之后的决策树与用全部训练集得到的决策树不相同，因此分类精度可能会有所下降。将决策树分成规则集，从根节点到叶子结点的一条路径就是一条合取规则，然后将所有的规则集合并为一个规则集。规则的合并是采用 Williams 的多决策树归纳合并的方法。

3 性能比较

表1给出了5种并行决策树的分析与比较。

动态数据分布的突出缺点是在处理机分配阶段只有一部分处理机参与计算，而且当任务量大小不均匀时负载平衡比

(下转第136页)

$1-f_{t(V\cup W)-}=1-Rf_-(V\cup W)=\inf\{\max(f_v(x),f_w(x))|x\in[x]_R\}$。

利用上面的证明方式可得：

$\max(1-f_{V-},1-f_{W-})\leqslant 1-f_{t(V\cup W)-}$，综上所述，得到 $V_-\cup W_-\subseteq(V\cup W)_-$，同理可证$(V\cap W)_-=V_-\cap W_-$，所以(3)式成立。

(4)：由 $V\subseteq W$ 得 $t_v\leqslant t_w$，$1-f_v\leqslant 1-f_w$。

$\inf\{t_v(x)|x\in[x]_R\}\leqslant\inf\{t_v(x)|x\in[x]_R\}$，

$\sup\{f_v(x)|x\in[x]_R\}\geqslant\sup\{f_w(x)|x\in[x]_R\}$。

$1-\sup\{f_v(x)|x\in[x]_R\}\leqslant 1-\sup\{f_w(x)|x\in[x]_R\}$。

因此，$t_{V-}\leqslant t_{W-}$，$1-f_{V-}\leqslant 1-f_{W-}$，即 $V_-\subseteq W_-$ 成立，

同样，由 $t_v\leqslant t_w$，$1-f_v\leqslant 1-f_w$，得：

$\sup\{t_v(x)|x\in[x]_R\}\leqslant\sup\{t_v(x)|x\in[x]_R\}$，

$\inf\{f_v(x)|x\in[x]_R\}\geqslant\inf\{f_w(x)|x\in[x]_R\}$。

$1-\inf\{f_v(x)|x\in[x]_R\}\leqslant 1-\inf\{f_w(x)|x\in[x]_R\}$。

因此，$t_{V^-}\leqslant t_{W^-}$，$1-f_{V^-}\leqslant 1-f_{W^-}$，即 $V^-\subseteq W^-$ 成立。从而(4)式成立。证毕。

结论 本文以粗糙集基数表示的形式建立了产生粗糙集的 Vague 集形式，提出了粗糙 Vague 集的概念，给出了相关定理，对于处理一定形式的数据或知识信息具有工具性的使用价值，对于粗糙集与 Vague 集结合的理论与应用研究具有一定意义。

参考文献

1 Gau W L, Buehrer D J. Vague sets. IEEE Trans. Systems Man Cybernet, 1993, 23(2): 610～614

2 Pawlk Z, et al. Rough Sets. Communications of the ACM, 1995, 38(11): 89～95

3 Pawlk Z. Rough set theory and its application to data analysis. Cybernetics and Systems, 1998, 29(9): 661～668

4 Ziarko W. Introduction to the special issue on rough sets and knowledge discovery. International Journal of Computational Intelligence, 1995, 11(2): 223～226

5 曾黄麟. 粗集理论及其应用. 重庆：重庆大学出版社，1996

6 张文修，等. 粗糙集理论与方法. 北京：科学出版社，2001

(上接第131页)

较差。它的优点是当处理机分配完之后没有通信代价(除了动态负载平衡)。横向划分的缺点是通信代价随着树的增长变得很严重。它的优点是所有的处理机从始至终都参与任务计算，而且负载平衡比较好。纵向划分的缺点是随着树的层数的增加一些处理机将处于闲置状态。负载平衡需要额外的数据移动。而且每个处理机的负载不很均匀。这种方法比横向划分方法需要比较少的通信量，比动态数据划分方法的负载平衡度高。混合并行方法采用上述两种方法的优点。在各个处理机独立工作之前采用静态数据划分的并行方法，将各个子任务由单独的处理机负责，这样可以达到减少通信代价。这种方法同动态处理机组负责一个子任务，在处理机组内部采用静态数据分片的并行方法，一个处理机组保存了对应子任务的全部数据。各个处理机组之间需要数据交换，处理机组内部需要传送消息，当所有的处理机组只包含一个处理机时通信代价为零。各个处理机组的处理机的数量是和任务的大小成比例的，因此混合并行方法的负载平衡程度比较好。

表1 5种并行决策树的比较

方法	数据移动	传送统计图表	数据倾斜	负载平衡	精度
动态数据分片方法	多	没有	严重	差	高
横向数据分片方法	没有	多	中	好	高
纵向数据分片方法	没有	没有	中	中	高
混合并行方法	中	中	少	好	高
决策树合并	没有	没有	很少	很好	略低

总结和讨论 随着生物信息领域的飞速发展，出现越来越多的生物数据库，这些数据库的最大特点就是数据量庞大。传统的内存算法不适合庞大的生物数据，这样的执行时间有时是无法接受的，所以并行挖掘算法就变得很重要。前面对现阶段的并行建立决策树的算法进行了综述，并且对各个算法进行了性能评价。混合并行的方法在这些方法中是比较好的，它结合了各种方法的优点，但是文[8]没有考虑纵向划分的混合方法，纵向划分的方法在通信代价方面优于横向划分方法，而且生物数据会形成很多属性，属性值一般比较少，更适合纵向划分，因此这是值得研究的一个方面。

参考文献

1 Wang D, Wang X, Honavar V, Dobbs D. Data-Driven Generation of Decision Trees for Motif-Based Assignment of protein Sequences to Functional families. In: Proc. of the Atlantic Symposium on Computational Biology, Genome Information Systems&Technology. 2001

2 Quinlan J R. C4. 5: programs for Machine Learning. Morgan Kaufmann, San Matoo, CA, 1993

3 Agawal R, et al. An interval classifier for database mining applications. In: proc. of the VLDB Conf. Vancouver, British Columbia, Canada, Aug. 1992. 560～573

4 Michie D, spiegelhalter D J, Taylor C C. Machine Learning, Neural and Statistical Classification. Ellis Horwood, 1994

5 Hunt E B, Marin J, Stone P T. Experiments in Induction. Academic press, 1996

6 Quinlan J R. Discovering rules from large collections of examples: A cass study. In: Michie D, ed. Expert Systems in the Micro Electronic Age, Edinburgh University press, 1979

7 Agrawal R, Imielinski T, Swami A. Database mining: A performance perspective. IEEE Transactions on Knowledge and Data Eng., 1993, 5(6): 914～925

8 Mehta M, Agrawal R, Rissanen J. SLIQ: A fast scalable classifier for data mining. In: Proc. of the Fifth Int'l Conf. on Extending Database Technology, Avignon, france, 1996

9 Shafer J, Agrawal R, Mehta M. SPRINT: A scalable classifier for data mining. In: Proc. of the 22nd VLDB Conf. 1996

10 Chattratichat J, et al. Large scale data mining: Challenges and responses. In: Proc. of the Third Int7 Conf. on Knowledge discovery and Data Mining, 1997

11 Joshi M V, Karypis G, Kumar V. ScalparC: a new scalable and efficient parallel classification algorithm for mining large datasets. In: Proc. of the Intl. parallel processing Symposium, 1998