软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(5):1064−1078 [doi: 10.13328/j.cnki.jos.004590]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

云平台下基于粗糙集的并行增量知识更新算法*

张钧波[1,2], 李天瑞[1], 潘 毅[2], 罗 川[1], 滕 飞[1]

[1](西南交通大学 信息科学与技术学院,四川 成都 610031)

[2](Department of Computer Science, Georgia State University, Atlanta, USA)

通讯作者: 李天瑞, E-mail: trli@swjtu.edu.cn

摘 要: 日益复杂和动态变化的海量数据处理,是当前人们普遍关注的问题,其核心内容之一是研究如何利用已有的信息实现快速的知识更新.粒计算是近年来新兴的一个研究领域,是信息处理的一种新的概念和计算范式,主要用于描述和处理不确定的、模糊的、不完整的和海量的信息,以及提供一种基于粒与粒间关系的问题求解方法.作为粒计算理论中的一个重要组成部分,粗糙集是一种处理不确定性和不精确性问题的有效数学工具.根据云计算中的并行模型 MapReduce,给出了并行计算粗糙集中等价类、决策类和两者之间相关性的算法;然后,设计了用于处理大规模数据的并行粗糙近似集求解算法.为应对动态变化的海量数据,结合 MapReduce 模型和增量更新方法,根据不同的增量策略,设计了两种并行增量更新粗糙近似集的算法.实验结果表明,该算法可以有效地快速更新知识;而且数据量越大,效果越明显.

关键词: 云计算;MapReduce;粗糙集;增量学习

中图法分类号: TP311

中文引用格式: 张钧波,李天瑞,潘毅,罗川,滕飞.云平台下基于粗糙集的并行增量知识更新算法.软件学报,2015,26(5):1064−1078. http://www.jos.org.cn/1000-9825/4590.htm

英文引用格式: Zhang JB, Li TR, Pan Y, Luo C, Teng F. Parallel and incremental algorithm for knowledge update based on rough sets in cloud platform. Ruan Jian Xue Bao/Journal of Software, 2015,26(5):1064−1078 (in Chinese). http://www.jos.org.cn/1000-9825/4590.htm

Parallel and Incremental Algorithm for Knowledge Update Based on Rough Sets in Cloud Platform

ZHANG Jun-Bo[1,2], LI Tian-Rui[1], PAN Yi[2], LUO Chuan[1], TENG Fei[1]

[1](School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)

[2](Department of Computer Science, Georgia State University, Atlanta, USA)

Abstract: The increasing complexity and dynamic change of massive data processing currently receive widespread attention. One of its core content is to study how to use the existing information to achieve rapid updating of knowledge. Granular computing (GrC), a new computing paradigm of information processing, is an emerging research field which is mainly used to describe and deal with uncertain, vague, incomplete and massive data, and provides a solution based on the granularity and the relationship between the granularities. As an important part of GrC, rough set theory is an effective mathematical tool to deal with the uncertainty and imprecise problems. Based on the MapReduce model in cloud computing, this paper first presents a parallel algorithm for computing the equivalence classes, decision classes and the association between them in rough set theory. A parallel algorithm is then designed for computing rough set approximations from large-scale data. To adapt to the dynamic real-time system, the MapReduce model and incremental method are combined to build two parallel incremental algorithms for updating rough set approximations in different incremental strategies. An

* 基金项目: 国家自然科学基金(61175047, 61100117, 61202043); 国家自然科学基金联合基金(U1230117); 四川省科技支撑计划(2012RZ0009); 西南交通大学优秀博士学位论文培育项目; 中央高校基本科研业务费专项资金(SWJTU12CX098)

收稿时间: 2013-03-28; 修改时间: 2013-05-03, 2013-12-06; 定稿时间: 2014-02-17; jos 在线出版时间: 2014-08-19

CNKI 网络优先出版: 2014-08-19 14:37, http://www.cnki.net/kcms/doi/10.13328/j.cnki.jos.004590.html

extensive experimental evaluation on big data sets show that the proposed algorithms are very effective and have better performance with the increasing size of the data.

Key words: cloud computing; MapReduce; rough set; incremental learning

粒计算是当前信息处理中一种新的计算范式,可用于描述和处理不确定的、模糊的、不完整的和海量的信息以及提供一种基于粒和粒间关系的问题求解方法.它可对问题进行不同层次的抽象和处理来寻求不同粒度上的近似解,以达到简化问题和快速求解的目的.通过使用不同信息粒度,粒计算可以容忍不确定、不完全或有噪音的信息,从而获得具有鲁棒性的解决方案.粗糙集是一种处理不确定性和不精确性问题的有效数学工具,是粒计算理论中的一个重要组成部分.它无需提供问题所需处理的数据集合之外的任何先验知识,可以在保持分类能力不变的前提下,通过知识约简,导出问题的决策或分类规则,为决策提供服务,已广泛应用于数据挖掘、机器学习、模式识别及商务智能等领域.

针对静态的信息系统,研究人员提出了许多基于粗糙集理论的知识发现方法.Blaszczynski 等人设计了变精度粗糙集方法下的一个序列覆盖规则抽取算法[1].Chen 等人基于幂集树提出了一种特征提取的粗糙集方法[2].Hong 等人给出利用模糊粗糙集从不完备定量数据中挖掘知识的方法[3].Inuiguchi 给出了从两个决策表中基于粗糙集获取规则的方法[4].Kaneiwa 讨论了从多个数据集中知识发现的粗糙集方法[5].Leung 等人提出了从区间值信息系统中提取分类规则的方法[6].Miao 等人提出了基于粗糙集理论的杂合算法,用于信息检索中的文本分类[7].Wang 等人采用粗糙集方法来构造模糊决策树[8].Wu 等人提出了不完备模糊信息系统下粗糙集模型的知识获取方法[9].Hu 等人采用邻域粗糙集方法来构造邻域分类器[10].Yao 讨论了三枝决策应用于概率粗糙集模型的方法[11].张清华等人提出了在已有知识基(粒)空间下寻找目标集合(概念)的近似集的方法[12].

然而,这些针对静态的信息系统基于粗糙集的知识发现方法在面对大规模的复杂现实问题时,都面临高复杂性计算的困难.因此,如何提高基于粗糙集的知识发现算法的效率,以充分体现粗糙集解决不确定性问题的优势,成为粗糙集研究领域的主要任务之一.增量式知识更新方法由于其能够充分利用已有的知识,可以有效提高知识更新的效率,因此,很多学者致力于利用增量式知识更新方法来提高基于粗糙集的知识发现的效率.Shan 等人首次提出了基于粗糙集的增量式规则获取算法[13].Guo 等人给出了基于搜索树的规则增量挖掘方法,其优点是不需要创建区分矩阵[14].Zheng 等人提出了基于规则树的增量式高效知识获取算法,其特点是在原有的决策树规则集基础上进行规则的增量式更新,避免了重复学习,提高了效率[15].Fan 等人提出了增量式提取具有最大长度的决策规则方法[16].王利等人讨论了新增记录与已有条件属性等价类的关系以及对规则集的影响,提出了基于变精度粗糙集模型的增量式规则获取算法[17].Liu 等人在变精度粗糙集模型中定义了覆盖矩阵和支持矩阵,提出了对象增加时,通过动态更新覆盖矩阵和支持矩阵方法可以实现兴趣知识的快速更新[18].Zhang 等人给出了邻域粗糙集模型下多对象增加删除时近似集快速更新的方法[19].

虽然目前粒度变化下基于粗糙集的增量式学习算法很多,但在文献里很少利用大规模数据集来进行测试评价算法的性能和适用性,而且不同算法之间的比较与性能测试的相关结果较少,极大地制约了基于粗糙集和粒计算的知识发现理论与方法的发展与实际应用;其次,由于云计算技术可以极大地提高海量数据处理的效率[20],如何应用云计算技术提高基于粒计算和粗糙集理论的知识发现效率具有重要意义.但这方面的工作目前还很罕见,已有的工作有:Liu 等人给出了基于改进的分辨矩阵规则增量挖掘的并行算法[21];Qian 等人提出了云计算环境下基于 MapReduce 的属性约简方法[22];Zhang 等人给出了一种基于 MapReduce 的并行计算粗糙近似集的方法[23],并在此基础上提出了基于 MapReduce 的规则提取算法[24].粗糙集上、下近似集的计算,是基于粗糙集的属性约简和知识获取方法的基础.本文在已有工作的基础上,结合增量更新技术,提出了基于 MapReduce 的快速更新粗糙集近似集的并行增量算法,可以有效地加快动态知识更新过程.

1 相关理论

1.1 云计算编程模型MapReduce

MapReduce 是由 Google 公司提出的一种处理海量数据的并行编程模式[25],Apache 基金会在此基础上实现

了开源的 Hadoop 并行平台[26].用户将实际应用问题分解成若干可并行操作的子问题,设计相应的 Map 函数,Combine 函数(可选)与 Reduce 函数就能将自己的应用程序运行在云计算环境中.

- Map 函数:接受一个输入对,然后产生一个中间⟨*key*,*value*⟩对集.MapReduce 库把所有具有相同中间 *key* I 的中间 *value* 聚合在一起,经过归并/排序过程,然后把它们传递给 Reduce 函数.
- Combine 函数(可选):是本地的 Reduce 过程.从本地的 Map 接受一个中间 *key* 和相关的一个 *value* 集,形成一个较小的 *value* 集,并传给 Reduce.一般情况下,Combine 函数与 Reduce 函数是一致的.
- Reduce 函数:接受一个中间 *key* I 和相关的一个 *value* 集.它合并这些 *value*,形成一个较小的 *value* 集.

图 1 显示了 MapReduce 在开源云计算平台 Hadoop 上的流程图,其中,HDFS(Hadoop distributed file system)是指 Hadoop 分布式文件系统[26].

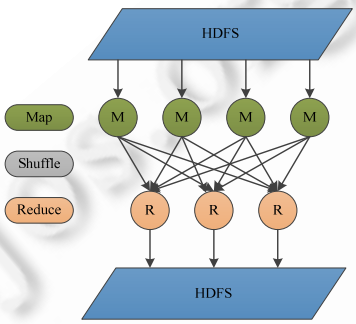


Fig.1 MapReduce model

图 1 MapReduce 模型

1.2 粗糙集模型

本节我们回顾下本文用到的一些粗糙集的基本定义[23,27].

设四元组 $S=(U,A,V,f)$是一个信息系统,其中,U 为非空有限的对象集合,称为论域;A 表示非空属性集合;$V=\bigcup_{a\in A}V_a$, 其中,V_a 称为属性 a 的值域;$f:U\times A\to V$ 是一个信息函数,满足对任意 $x\in U,a\in A$ 有 $f(x,a)\in V_a$.特别地,如果 $A=C\cup D$,其中,C 为非空有限条件属性集,D 为非空有限决策集,那么信息系统被称为决策表.

定义 1(对象 *x* 关于属性集 *B* 的信息集[23,27]). 设 $B=\{b_1,b_2,\ldots,b_l\}\subseteq A$ 为非空属性子集.对任意 $x\in U$,关于属性集 B 的信息集定义如下:

$$\bar{x}_B=\langle f(x,b_1),f(x,b_2),\ldots,f(x,b_l)\rangle \tag{1}$$

关于属性 B 的等价关系,也称为不可区分关系,表示为 $IND(B)$,定义如下:

$$IND(B)=\{(x,y)\mid(x,y)\in U\times U,\bar{x}_B=\bar{y}_B\} \tag{2}$$

等价关系 $IND(B)$把论域 U 切分成若干个等价类,记为

$$U/IND(B)=\{[x]_B|x\in U\} \tag{3}$$

其中,$[x]_B$ 表示等价类,$[x]_B=\{y\in U|(x,y)\in IND(B)\}$.为简便起见,我们用 U/B 代替 $U/IND(B)$.

定义 2(等价类 *E* 关于属性集 *B* 的信息集[23,27]). 设 $B\subseteq A$ 为非空属性子集.对任意 $E\in U/B$,关于属性集 B 的信息集定义如下:

$$\bar{E}_B=\bar{x}_B,x\in U \tag{4}$$

定义 3(上近似集与下近似集[23,27]). 设 $X\subseteq U,R$ 为一个等价关系,$U/R=\{E_1,E_2,\ldots,E_t\}$.$X$ 的上、下近似集定义如下:

$$\begin{cases}\underline{R}(X)=\{x\in U\mid[x]_R\subseteq X\}\\ \overline{R}(X)=\{x\in U\mid[x]_R\cap X=\varnothing\}\end{cases} \tag{5}$$

或者相当于:

$$\begin{cases}\underline{R}(X)=\bigcup\{E\in U/R\mid E\subseteq X\}\\ \overline{R}(X)=\bigcup\{E\in U/R\mid E\cap X=\varnothing\}\end{cases}\tag{6}$$

显然, $\underline{R}(X)\subseteq X\subseteq\overline{R}(X)$.$X$ 的边界域定义如下:

$$BND_B(X)=\overline{R}(X)-\underline{R}(X)\tag{7}$$

定义 4(决策 *D* 的上、下近似集[23,27]). 给出决策表 $S=(U,C\cup D,V,f)$.$U/D=\{D_1,D_2,\ldots,D_r\}$决策类的集合,是决策属性 D 把对象集 U 切分成了 r 个互斥的子集.设 $B\subseteq C$ 为非空属性子集,$IND(B)$为等价关系,决策 D 的上、下近似集定义如下:

$$\begin{cases}\underline{R}_B(D)=\{\underline{R}_B(D_1),\underline{R}_B(D_2),...,\underline{R}_B(D_r)\}\\ \overline{R}_B(D)=\{\overline{R}_B(D_1),\overline{R}_B(D_2),...,\overline{R}_B(D_r)\}\end{cases}\tag{8}$$

决策 D 的正域定义如下:

$$POS_B(D)=\bigcup_{D_i\in U/D}\underline{R}_B(D_i)\tag{9}$$

2 基于 MapReduce 的粗糙近似集计算方法

本节简要回顾我们提出的并行计算粗糙近似集的方法[23].

定义 5(决策表的划分[23]). 给出决策表 $S=(U,C\cup D,V,f)$.设 $S_i=(U_i,C\cup D,V,f),i\in\{1,2,\ldots,m\}$.它满足:

(1) $U=\bigcup_{i=1}^{m}U_i$;

(2) $U_j\cap U_k=\varnothing,\forall j,k\in\{1,2,\ldots,m\},j\neq k$.这意味着决策表 S 被切分为 m 个决策子表.

定理 1(两个等价类的合并[23]). 给出决策表 $S_1=(U_1,C\cup D,V,f)$与 $S_2=(U_2,C\cup D,V,f)$.设 $B\subseteq C\cup D$ 为非空属性子集,$E\in U_1/B$ 和 $F\in U_2/B$ 是两个来自不同决策表的等价类.下面的结果成立:

(1) 如果 $\bar{E}_B=\bar{F}_B$, 那么等价类 E 和 F 可以合并为一个等价类 $G,G=E\cup F$ 且 $\bar{G}_B=\bar{E}_B=\bar{F}_B$;

(2) 如果 $\bar{E}_B\neq\bar{F}_B$, 那么等价类 E 和 F 不能合并为一个等价类.

定理 2(多个等价类的合并[23]). 给出决策表 $S=(U,C\cup D,V,f)$和决策子表 $S_i=(U_i,C\cup D,V,f),i\in\{1,2,\ldots,m\}$. $\forall B\subseteq C$,令 $U/B=\{E_1,E_2,\ldots,E_t\}$, $U_i/B=\{E_{i1},E_{i2},...,E_{ip_i}\}$ 和 $E_{all}=\bigcup_{i=1}^{m}U_i/B=\{E_{11},E_{12},...,E_{1p_1},...,E_{m1},E_{m2},...,E_{mp_1}\}$. 那么, $\forall E_k\in U/B$,我们有 $E_k=\bigcup\{F\in E_{all}\mid\bar{F}_B=\overline{E_k}_B\}$.

定理 3(决策类的合并[23]). 给出决策表 $S=(U,C\cup D,V,f)$和决策子表 $S_i=(U_i,C\cup D,V,f),i\in\{1,2,\ldots,m\}$.$\forall B\subseteq C\cup D$, 令 $U/D=\{D_1,D_2,\ldots,D_r\}$, $U_i/D=\{D_{i1},D_{i2},...,D_{iq_i}\}$ 和 $D_{all}=\bigcup_{i=1}^{m}U_i/B=\{D_{11},D_{12},...,D_{1p_1},...,D_{m1},D_{m2},...,D_{mq_1}\}$. 那么, $\forall D_j\in U/D$,我们有 $D_j=\bigcup\{F\in D_{all}\mid\bar{F}_D=\overline{D_j}_D\}$.

定义 6(等价类与决策类的相关性[23]). 给出决策表 $S=(U,C\cup D,V,f),B\subseteq C.U/B=\{E_1,E_2,\ldots,E_t\},U/D=\{D_1,D_2,\ldots,D_r\}$.如果 $E_k\cap D_j=\varnothing,E_k\in U/B,D_j\in U/D$,我们称 E_k 和 D_j 是相关的,表示为 $Ass(E_k,D_j)$=True;否则,$Ass(E_k,D_j)$=False.

定理 4(合并相关性[23]). 给出决策表 $S=(U,C\cup D,V,f)$和决策子表 $S_i=(U_i,C\cup D,V,f),i\in\{1,2,\ldots,m\}$.$\forall B\subseteq C$,令 $U/B=\{E_1,E_2,\ldots,E_t\},U/D=\{D_1,D_2,\ldots,D_r\}$, $U_i/B=\{E_{i1},E_{i2},...,E_{ip_i}\}$ 和 $U_i/D=\{D_{i1},D_{i2},...,D_{iq_i}\}$. 设

$$\overline{E_{ih}}=\overline{E_k},\overline{D_{ig}}=\overline{D_j}(E_k\in U/B,D_j\in U/D,E_{ih}\in U_i/B,D_{ig}\in U_i/D,i\in\{1,2,...,m\}),$$

如果 $Ass(E_{ih},D_{ig})$=True,那么 $Ass(E_k,D_j)$=True.

定理 5(等价类的 *Boolean* 表示[23]). 给出决策表 $S=(U,C\cup D,V,f)$.$\forall B\subseteq C$,令 $U/B=\{E_1,E_2,\ldots,E_t\},U/D=\{D_1,D_2,\ldots,D_r\}$.下面的结果成立:

(1) 如果 $Ass(E_k,D_{j_1})=\text{True}$ 且 $Ass(E_k,D_{j_2})=\text{True},(j_1\neq j_2,j_1,j_2\in\{1,2,...,r\},k\in\{1,2,...,t\})$, 那么 $\forall D_j\in U/D$, $E_k\not\subset\underline{R}_B(D_j)$, 表示为 $Boolean(E_k)$=False;

(2) 如果$\exists!j\in\{1,2,\ldots,r\},Ass(E_k,D_j)$=True$(k\in\{1,2,\ldots,t\})$,那么 $E_k\subseteq\underline{R}_B(D_j)$, 表示为 $Boolean(E_k)$=True.

定理 6(根据相关性计算上近似集). 给出决策表 $S=(U,C\cup D,V,f)$.$\forall B\subseteq C$,令 $U/B=\{E_1,E_2,\ldots,E_t\},U/D=\{D_1,D_2,\ldots,D_r\}$.$\forall D_j\in U/D,D_j$ 的上近似集可以表示为

$$\overline{R}_B(D_j)=\bigcup\{E_k\in U/B\mid Ass(E_k,D_j)=\text{True}\} \tag{10}$$

证明:因为 $Ass(E_k,D_j)$=True$\Leftrightarrow E_k\cap D_j\neq\varnothing$,那么根据定义 3,有:

$$\overline{R}_B(D_j)=\bigcup\{E_k\in U/B\mid E_k\cap D_j\neq\varnothing\}=\bigcup\{E_k\in U/B\mid Ass(E_k,D_j)=\text{True}\}.$$

证毕. □

定理 7(根据相关性计算下近似集). 给出决策表 $S=(U,C\cup D,V,f)$.$\forall B\subseteq C$,令 $U/B=\{E_1,E_2,\ldots,E_t\}$,$U/D=\{D_1,D_2,\ldots,D_r\}$.$\forall D_j\in U/D$,$D_j$ 的下近似集可以表示为

$$\underline{R}_B(D_j)=\bigcup\{E_k\in U/B\mid Ass(E_k,D_j)=\text{True}\text{且}Boolean(E_k)=\text{True}\} \tag{11}$$

证明:证明过程同定理 6 的证明.证毕. □

2.1 MapReduce算法设计与实现

根据上文,我们给出了并行计算近似集的流程,如图 2 所示.

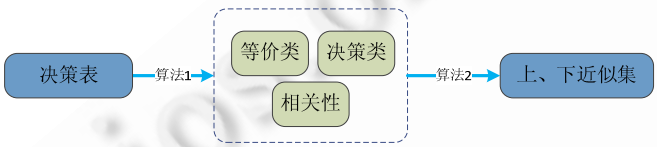


Fig.2 Flowchart of the parallel computing method

图 2 并行计算方法流程图

根据定理 2~定理 4,我们给出了基于 MapReduce 的等价类、决策类及其相关性并行计算的流程图(如图 3 所示),并设计了相关算法,见算法 1.

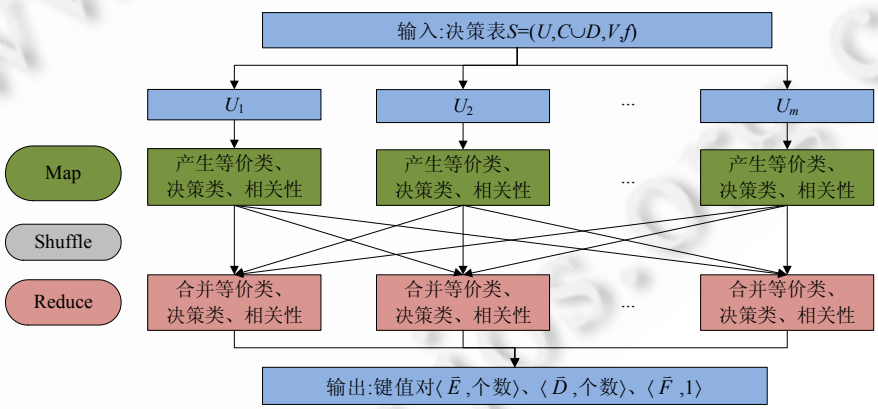


Fig.3 Flowchart of parallel computing equivalence classes, decision classes and associations based on MapReduce

图 3 基于 MapReduce 的等价类、决策类、相关性并行计算流程图

算法 1. 基于 MapReduce 的等价类、决策类、相关性并行计算算法(并行算法).

Map(*key*,*value*)

输入:*key* 为文档名;*value* 为 $S_i=(U_i,C\cup D,V,f)$.

开始:

for each $x\in U_i$ **do**

let *key*′=‘*E*’+ $\bar{x}_B$ //‘*E*’标识符号,用于等价类

output.collect(*key*′,1);

let *key*′=‘*D*’+$\bar{x}_D$ //‘*D*’标识符号,用于决策类

output.collect(*key*′,1);

let *key*′=‘*F*’+$\bar{x}_{B\cup D}$ //‘*F*’标识符号,用于描述等价类与决策类是否相关

output.collect(*key*′,1);

endfor

结束

Combine(*key*,*V*)/**Reduce**(*key*,*V*)

输入:*key* 为相对于属性集 *B* 或 *D* 或 *B*∪*D* 的信息集;*V* 为该信息集出现个数的列表;

开始:

if (*key.startsWith*(‘*F*’))

output.collect(*key*,1)

else:

let *sum*=0

for each *val* in *V* **do**

sum+=*val*

endfor

output.collect(*key*,*sum*)

endif

结束

我们通过以下两个方面优化了文献[23]中提出的算法:

- 在算法设计上,将计算等价类、决策类与相关性放到同一个函数中,用标识符‘*E*’,‘*D*’,‘*F*’分别区分;
- 添加 Combine 阶段:用于 Map 结束后本地合并等价类、决策类与相关性.

在计算完等价类、决策类与相关性之后,根据定理 5、定理 6,根据已有的 MapReduce 结果,我们给出快速计算上、下近似集的流程图(如图 4 所示).基于此,我们设计了快速计算上、下近似集的算法,见算法 2.

Fig.4 Flowchart of fast computing the upper and lower approximations

图 4 快速计算上、下近似集的流程图

算法 2. 计算粗糙集上、下近似集计算算法.

输入:MapReduce 程序输出的结果.

输出:各个决策 D_j 的上、下近似集的个数.

开始:

//计算决策 D 的上近似集的个数

for each $|\underline{R}(D_j)|$ **do**

let $|\overline{R}(D_j)|=0$;

for each $S=(U,C\cup D,V,f)$ **do**

if $Ass(E_k,D_j)$=True **then**

//$|\overline{E_k}|$ 为信息集 $\overline{E_k}$ 个数

$|\overline{R}(D_j)|=|\overline{R}(D_j)|+|\overline{E_k}|$;

endif

endfor

endfor

//计算决策 D 的下近似集的个数

for each E_k **do**

if $Ass(E_k,D_{j_1})==\text{True}$ **and** $Ass(E_k,D_{j_2})==\text{True}, j_1\neq j_2, j_1,j_2\in\{1,2,...,r\}$ **then**

$Boolean(E_k)$=False;

else $Boolean(E_k)$=True

endfor

for each D_j **do**

let $|\underline{R}(D_j)|=0$;

for each E_k **do**

if $Boolean(E_k)$=True **and** $Ass(E_k,D_j)$=True **then**

$|\underline{R}(D_j)|=|\underline{R}(D_j)|+|\overline{E_k}|$; //$|\overline{E_k}|$ 为信息集 $\overline{E_k}$ 个数

endif

endfor

endfor

输出每个决策的上下近似集的个数$|\overline{R}(D_j)|,|\underline{R}(D_j)|$.

结束

3 基于 MapReduce 的粗糙近似集的增量更新方法

这里,我们将讨论有新的数据进入到决策表的情况,假设新增决策表为 $S'=(U',C\cup D,V,f)$.下面我们将给出更新等价类、决策类及其相关性的相关理论.

推论 1. 给出决策表 $S=(U,C\cup D,V,f)$.$\forall B\subseteq C$,令 $U/B=\{E_1,E_2,\ldots,E_t\}$.新增决策表 $S'=(U',C\cup D,V,f)$,令 $U'/B=\{E'_1,E'_2,...,E'_{t'}\}$.对更新后的决策表 $S^*=(U^*,C\cup D,V,f)$,$U^*=U\cup U'$,设 $U^*/B=\{E_1^*,E_2^*,...,E_{t^*}^*\}$.那么,$\forall E_{j^*}^*\in U^*/B$,$\exists E_j\in U/B$ 或 $\exists E'_{j'}\in U'/B$,使得 $\overline{E_{j^*}^*}=\overline{E_j}$ 或 $\overline{E_{j^*}^*}=\overline{E'_{j'}}$.换言之,$E_{j^*}^*=E_j$ 或 $E_{j^*}^*=E'_{j'}$ 或 $E_{j^*}^*=E_j\cup E'_{j'}$.

证明:这里,我们用反证法来证明.$\forall E_{j^*}^*\in U^*/B$,假设 $\nexists E_j\in U/B$且$\nexists E'_{j'}\in U'/B$,使得 $\overline{E_{j^*}^*}=\overline{E_j}$ 或 $\overline{E_{j^*}^*}=\overline{E'_{j'}}$,那么 $E_j\not\subset E_{j^*}^*$ 或 $E'_{j'}\not\subset E_{j^*}^*$,也就是 $E_j\notin U^*$或 $E'_{j'}\notin U^*$.而这与 $E_j\in U\subseteq U^*$且 $E'_{j'}\in U'\subseteq U^*$ 相矛盾.因此,假设不成立.那么,$\forall E_{j^*}^*\in U^*/B,\exists E_j\in U/B$或$\exists E'_{j'}\in U'/B$,使得 $\overline{E_{j^*}^*}=\overline{E_j}$ 或 $\overline{E_{j^*}^*}=\overline{E'_{j'}}$.对于$\exists E_j\in U/B$ 或 $\exists E'_{j'}\in U'/B$,我们分 3 种情况来讨论:

(a) $\forall E^*_{j^*}\in U^*/B, \exists E_j\in U/B$ 且 $\nexists E'_{j'}\in U'/B$, 我们有 $E^*_{j^*}=E_j$;

(b) $\forall E^*_{j^*}\in U^*/B, \nexists E_j\in U/B$ 且 $\exists E'_{j'}\in U'/B$, 我们有 $E^*_{j^*}=E'_{j'}$;

(c) $\forall E^*_{j^*}\in U^*/B, \exists E_j\in U/B$ 且 $\exists E'_{j'}\in U'/B$, 我们有 $E^*_{j^*}=E_j\cup E'_{j'}$.

换言之, $\forall E^*_{j^*}\in U^*/B, \exists E_j\in U/B$ 或 $\exists E'_{j'}\in U'/B, E^*_{j^*}=E_j$ 或 $E^*_{j^*}=E'_{j'}$ 或 $E^*_{j^*}=E_j\cup E'_{j'}$. □

推论 2. 给出决策表 $S=(U,C\cup D,V,f)$, 令 $U/D=\{D_1,D_2,\ldots,D_r\}$. 新增决策表 $S'=(U',C\cup D,V,f)$, 令 $U'/D=\{D'_1,D'_2,\ldots,D'_{r'}\}$. 对更新后的决策表 $S^*=(U^*,C\cup D,V,f), U^*=U\cup U'$, 设 $U^*/D=\{D^*_1,D^*_2,\ldots,D^*_{r^*}\}$. 那么, $\forall D^*_{j^*}\in U^*/D, \exists D_j\in U/D$ 或 $\exists D'_{j'}\in U'/D$, 使得 $\overline{D^*_{j^*}}=\overline{D_j}$ 或 $\overline{D^*_{j^*}}=\overline{D'_{j'}}$. 换言之, $D^*_{j^*}=D_j$ 或 $D^*_{j^*}=D'_{j'}$ 或 $D^*_{j^*}=D_j\cup D'_{j'}$.

证明:证明过程同推理 1 的证明. □

推论 3. 给出决策表 $S=(U,C\cup D,V,f)$. $\forall B\subseteq C$, 令 $U/B=\{E_1,E_2,\ldots,E_t\}, U/D=\{D_1,D_2,\ldots,D_r\}$. 新增决策表 $S'=(U',C\cup D,V,f), U'/B=\{E'_1,E'_2,\ldots,E'_{t'}\}, U'/D=\{D'_1,D'_2,\ldots,D'_{r'}\}$. 对更新后的决策表 $S^*=(U^*,C\cup D,V,f), U^*=U\cup U'$, 设 $U^*/B=\{E^*_1,E^*_2,\ldots,E^*_{t^*}\}, U^*/D=\{D^*_1,D^*_2,\ldots,D^*_{r^*}\}$. 设 $\overline{E^*_{j^*}}=\overline{E_j}=\overline{E'_{j'}}$ 且 $\overline{D^*_{j^*}}=\overline{D_j}=\overline{D'_{j'}}$, 如果 $Ass(E_k,D_j)$=True 或 $Ass(E_{k'},D_{j'})$=True, 那么有 $Ass(E^*_k,D^*_j)=\text{True}$.

证明:证明过程同推理 1 的证明. □

根据推论 1~推论 3, 我们知道在数据增加时, 等价类、决策类及其相关性可以通过增量更新方法快速得到. 基于此, 我们给出了并行增量更新的方法, 如图 5 所示. 根据原有的中间结果和新增的决策表, 我们分别处理, 并合并成新的等价类、决策类及其相关性, 然后根据算法 2 直接求得上、下近似集. 具体的算法将在下一节中详细讨论.

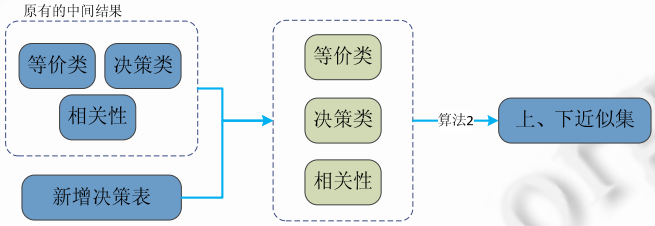


Fig.5 Flowchart of the parallel and incremental updating method

图 5 并行增量更新方法流程图

3.1 并行增量算法设计与实现

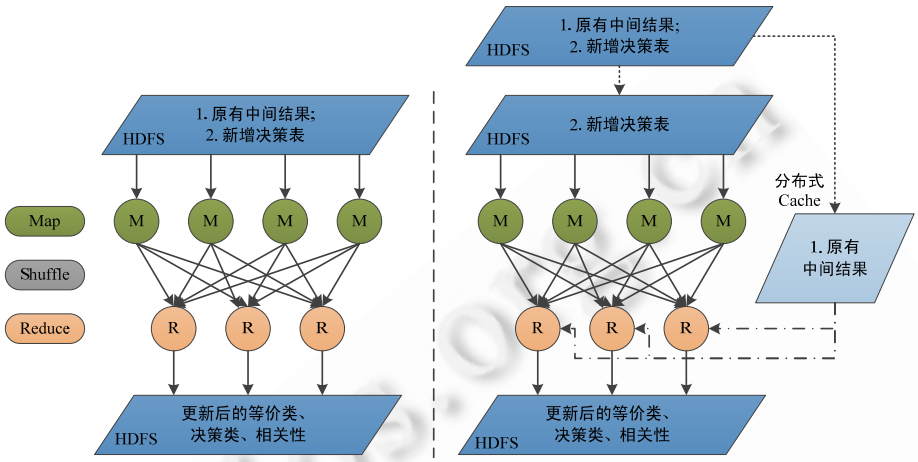
根据推论 1~推论 3 可知, 我们需要额外的存储空间来保存原始决策表产生的中间结果. 给出决策表 $S=(U,C\cup D,V,f), B\subseteq C, U/B=\{E_1,E_2,\ldots,E_t\}, U/D=\{D_1,D_2,\ldots,D_r\}$, 那么中间结果会产生 t 个等价类、r 个决策类及 $t\times r$ 个它们之间的相关性. 因为每个等价类包含 $|B|$ 个属性, 每个决策包含 $|D|$ 个属性, 相关性只有一个值, 所以需要的总的辅助存储空间为 $O(|B|\times t+|D|\times r+t\times r)$.

因为涉及到并行编程模型 MapReduce, 所以如何设计增量算法变得额外重要. 不同于串行增量算法, 如何将原有中间结果合并到现有过程, 在并行增量算法变得至关重要. 为此在本文中, 我们尝试了从 Map 阶段和 Reduce 阶段分别导入原有中间结果, 并为此设计了两种不同的基于 MapReduce 的并行增量算法.

3.1.1 并行增量算法 I

对于原有决策表 $S=(U,C\cup D,V,f)$ 和新增决策表 $S'=(U',C\cup D,V,f)$. 一般来说, MapReduce 算法产生结果的数据规模远远小于之前的数据规模, 这里, 我们的更新方法很好地利用了这个性质, 通过对原有决策表的 MapReduce 产生的中间结果保存, 将其与新增决策表 $S'=(U',C\cup D,V,f)$ 同时作为新的 MapReduce 算法的输入(即, 从 Map 阶段导入原有中间结果), 其流程如图 6(a)所示, 具体操作详见算法 3. 在 Map 阶段, 对新增决策子表及原来中间结果

子集进行分别处理;在 Combine 和 Reduce 阶段,进行等价类、决策类及相关性的合并.然后,我们再调用算法 2 即可快速更新粗糙集上、下近似集.



(a) 并行增量算法 I (b) 并行增量算法 II

Fig.6 Two different strategies

图 6 两种不同的策略

算法 3. 基于 MapReduce 的等价类、决策类、相关性并行增量更新算法(并行增量算法 I).

Map(*key*,*value*)

输入:*key* 为文档名.*value* 为:

(1) 新增决策子表 $S_i'=(U_i',C\cup D,V,f)$;

(2) 原来的中间结果子集:$OldRet_j$=⟨*Information Set*,*Count*⟩键值对.

开始:

(1) 对新增决策表的处理:

for each $x\in U_i'$ **do**

let *key*′=‘*E*’+$\bar{x}_B$

output.collect(*key*′,1);

let *key*′=‘*D*’+$\bar{x}_D$

output.collect(*key*′,1);

let *key*′=‘*F*’+$\bar{x}_{B\cup D}$

output.collect(*key*′,1);

endfor

(2) 原来的 MapReduce 结果子集处理:

for each ***val*** in $OldRet_j$ **do**

let *key*′=*val.getKey*(·);

let *value*′=*val.getValue*(·);

output.collect(*key*′,*value*′);

endfor

结束

Combine(*key*,*V*)/**Reduce**(*key*,*V*)

输入:*key* 为相对于属性集 *B* 或 *D* 或 *B*∪*D* 的信息集;*V* 为该信息集出现个数的列表.

开始:

if (*key.startsWith*('*F*'))

output.collect(*key*,1)

else:

let *sum*=0

for each *val* in *V* **do**

sum+=*val*

endfor

output.collect(*key*,*sum*)

endif

结束

3.1.2 并行增量算法 II

在第 2 种增量策略中,我们利用 Hadoop 中的分布式 Cache 直接将原有中间结果导入到 Reduce 阶段,其流程如图 6(b)所示,详细的算法描述见算法 4.在 Map 阶段,只对新增决策子表进行处理;在 Combine 阶段,进行等价类、决策类及相关性的合并;在 Reduce 阶段,先进行初始化,导入原有的中间结果,再进行等价类、决策类及相关性的合并.之后,我们同样调用算法 2 即可快速更新粗糙集上、下近似集.

算法 4. 基于 MapReduce 及分布式 Cache 的等价类、决策类、相关性并行增量更新算法(并行增量算法 II).

Map(*key*,*value*)

输入:*key* 为文档名;*value* 为新增决策字表 $S_i'=(U_i',C\cup D,V,f)$.

开始:

//对新增决策表的处理:

for each $x\in U_i'$ **do**

let *key*'='*E*'+$\vec{x}_B$

output.collect(*key*',1);

let *key*'='*D*'+$\vec{x}_D$

output.collect(*key*',1);

let *key*'='*F*'+$\vec{x}_{B\cup D}$

output.collect(*key*',1);

endfor

结束

Combine(*key*,*V*)

输入:*key* 为相对于属性集 *B* 或 *D* 或 *B*∪*D* 的信息集;*V* 为该信息集出现个数的列表.

开始:

if (*key.StartsWith*('*F*'))

output.collect(*key*,1)

else:

let *sum*=0

for each *val* in *V* **do**

sum+=*val*

endfor

output.collect(*key*,*sum*)

endif

结束

Reduce(*key*,*V*)

初始化输入:原来的中间结果:$OldRet_j$=⟨*Information Set*,*Count*⟩键值对,存于分布式 Cache.

输入:*key* 为相对于属性集 *B* 或 *D* 或 *B*∪*D* 的信息集;*V* 为该信息集出现个数的列表.

Let *old_ret*=∅

初始化:

```
  for each val in OldRet_j do
    let key'=val.getKey(·);
    let value'=val.getValue(·);
    old_ret.put(key',value');
  endfor
```

初始化结束

开始:

```
  if (key.startsWith('F'))
    output.collect(key,1)
  else:
    let sum=0
    for each val in V do
      sum+=val
    endfor
    old_val=old_ret.get(key);          //如果不存在 key,则返回值 0;反之,则返回对应的值
    sum+=old_val
    output.collect(key,sum)
  endif
```

结束

3.2 算法复杂度分析

给出决策表 $S=(U,C\cup D,V,f),B\subseteq C$,在文献[28]中,徐章艳等人利用基数排序的思想,给出了一种计算等价类 U/B 时间复杂度为 $O(|B||U|)$的算法.因此,假定工作核数为 p,并行计算等价类、决策类、相关性的时间复杂度(即算法 1 的时间复杂度)为

$$O(|B||U|/p+|D||U|/p+|B\cup D||U|/p)=O(|B\cup D||U|/p).$$

对应地,算法 1 将产生$|U/B|$个等价类、$|U/D|$个决策类和$|U/B\cup D|$个相关性作为算法 2 的输入,因此,算法 2 的时间复杂度为

$$O(|U/D|\log|U/B\cup D|+|U/B\cup D|\log|U/B\cup D|+|U/B|\log|U/B|)=O(|U/B\cup D|\log|U/B\cup D|).$$

因为并行算法由算法 1 和算法 2 构成,所以并行算法的时间复杂度为

$$O(|B\cup D||U|/p+|U/B\cup D|\log|U/B\cup D|).$$

给出原有决策表 $S=(U,C\cup D,V,f)$的中间结果和新增决策表 $S'=(U',C\cup D,V,f),B\subseteq C$.在并行增量算法中,算法 3 针对新增决策表部分的时间复杂度与算法 1 相似,为 $O(|B\cup D||U'|/p)$.因为中间结果有$|U/B|$个等价类、$|U/D|$个决策类和$|U/B\cup D|$个相关性,所以其时间复杂度为 $O(|B\cup D||U/B\cup D|/p)$.因此,算法 3 的时间复杂度为

$$O(|B\cup D||U'|/p+|B\cup D||U/B\cup D|/p)=O(|B\cup D|(|U'|+|U/B\cup D|)/p).$$

因为并行算法由增量算法与算法 2 构成,令 $U^*=U\cup U'$,所以并行增量算法的时间复杂度为

$$O(|B\cup D|(|U'|+|U/B\cup D|)/p+|U^*/B\cup D|\log|U^*/B\cup D|).$$

在并行增量算法中,即算法 3 和算法 4,它们只是导入原有中间结果的阶段有所不同,其时间复杂度相同.

假设原始对象集为 U,增加的对象集为 U',增加后的对象集为 $U^*=U\cup U'$,并行算法和并行增量算法的时间复杂度分别为:

- 并行算法:$O(|B\cup D||U^*/p+|U^*/B\cup D|\log|U^*/B\cup D|)$;
- 并行增量算法:$O(|B\cup D|(|U'|+|U/B\cup D|)/p+|U^*/B\cup D|\log|U^*/B\cup D|)$;

我们不难发现:当 U'趋于无穷时,$U^*=U\cup U'$将近似等于 U',两者时间复杂度趋于相等.这说明,随着增量数据的不断增多,增量算法的优势会越来越弱.

4 实验分析

在文献[23]中,我们已经验证了基于 MapReduce 的并行算法可以很有效地处理大数据,这里我们只测试了并行增量算法的性能.

我们利用开源云计算平台 Hadoop-1.0.1(http://hadoop.apache.org/)和 Java 1.7.0_05 在 5 台计算机(AMD Opteron Processor 2376 处理器,8 核,主频 2.3 GHz,16GB 内存)构建的云计算环境下进行实验,其中 1 台作为主节点,4 台作为计算节点,每个计算节点提供 4 个 Map 槽和 Reduce 槽.

为了验证增量算法的性能,我们从 UCI 中选取了大数据集 KDD99,该数据集含有近 500 万条记录和 1 个决策属性及 41 个条件属性,其中 6 个为符号型属性,35 个数值型属性.我们的算法目前只针对符号型数据,因此,我们首先对 35 个数值型属性进行离散化处理.此外,我们用 WEKA[29]数据生成器 RDG1,通过设置不同的样本数、属性个数、类别数据,产生了 3 个更大的数据集,大小分别为 1.8GB,3.2GB 和 6.4GB,详细的数据描述见表 1.

Table 1 Datasets

表 1 数据集

	数据集	样本数	条件属性	类别	容量大小(GB)
1	KDD99	4 898 421	41	23	0.48
2	Weka-1.8G	32 000 000	10	35	1.80
3	Weka-3.2G	40 000 000	15	45	3.20
4	Weka-6.4G	80 000 000	15	58	6.40

我们把每个数据集平均分成 10 份,其中 5 份作为原有数据,另外 5 份作为增量数据,我们第 1 次增加 1 份,第 2 次增加 2 份,…令 R_a 为增量数据与原有数据的比值,称为增量变化率.这样,R_a 分别为 20%,40%,60%,80%和 100%.然后测试这 5 种增量变化情况下,并行算法、并行增量算法 I、并行增量算法 II 的性能,如图 7 所示.

从图 7(a)中我们可以看出,在数据集 KDD99 上,随着数据的增多,运行时间基本维持不变.这是因为 KDD99 这个数据集相对于我们的云计算平台而言规模太小.从图 7(b)~图 7(d)可以看出,随着数据规模的不断增大,两种增量算法的性能越来越好.

为了更直观地表示增量变化率与运行时间之间的关系,我们采用以下增量加速比[30]来说明:

$$IncS=\frac{T_s}{T_i},$$

其中,T_s 表示并行算法的运行时间,T_i 表示并行增量算法的运行时间.

表 2 和表 3 分别显示了并行增量算法 I 和并行增量算法 II 的增量加速比.与并行算法(非增量)相比,增量并行算法可以更快地执行完成.由结果可以看出:增量算法在数据集 KDD99 效果并不好,加速并不明显.其原因是 Hadoop 运行平台本身的延时较大,使得在小规模数据上表现不佳.总体而言,当增量变化率为 20%时,并行增量算法 I 和并行增量算法 II 可以分别获得平均 1.658 0 倍和 1.728 9 倍的运行速度.同时,我们注意到,在前 3 个数据集上,增量加速比与增量变化率之间的关系并不明显,其中一个可能的原因是:当数据量不大时,系统测试时的波动性比较大.但是在最大的数据集 Weka-6.4G,我们发现,随着增量变化率的增加,增量加速比递减,这与第 3.2 节中的算法复杂度分析相匹配.同时说明,随着新数据的逐渐增多,并行增量算法慢慢地失去优势;反之,新增数据越少,性能加速则越明显.从这一层面来看,并行增量算法对不断更新的实时模型有更好的支持,可以有效地应用于实时性要求更强的在线应用.

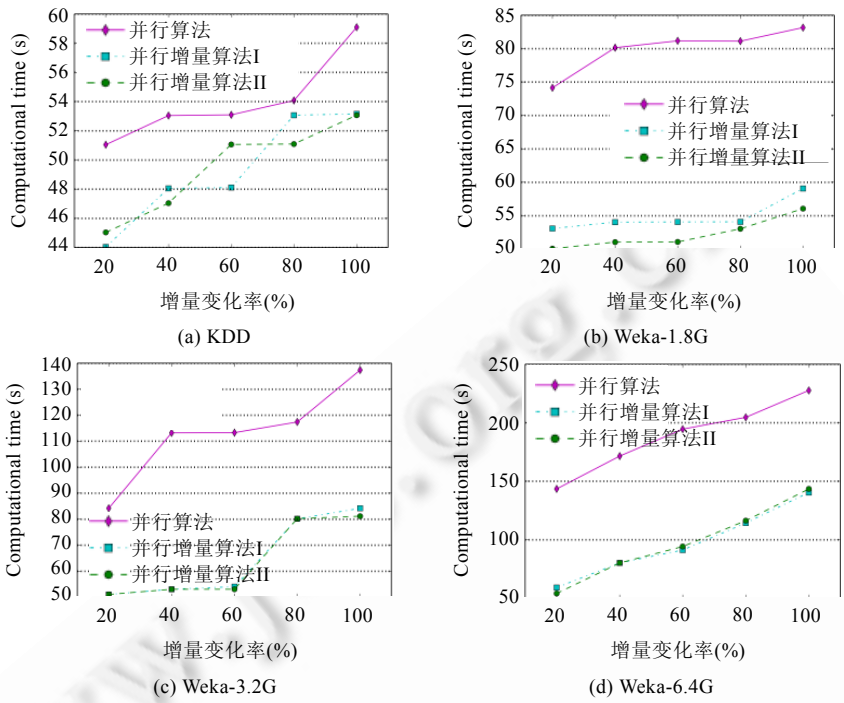


Fig.7 Computational time of three algorithms on different datasets

图 7 3 种算法在不同数据集下的计算时间

Table 2 IncS of parallel incremental algorithm I

表 2 并行增量算法 I 的增量加速比

数据集	增量变化率(R_a) (%)				
	20	40	60	80	100
KDD99	1.159 0	1.103 8	1.103 6	1.019 0	1.111 7
Weka-1.8G	1.649 3	2.133 8	2.096 0	1.464 6	1.631 9
Weka-3.2G	1.396 3	1.482 7	1.500 9	1.499 5	1.407 6
Weka-6.4G	**2.427 4**	**2.139 0**	**2.132 9**	**1.790 6**	**1.622 0**
Average	1.658 0	1.714 8	1.708 3	1.443 4	1.443 3

Table 3 IncS of parallel incremental algorithm II

表 3 并行增量算法 II 的增量加速比

数据集	增量变化率(R_a) (%)				
	20	40	60	80	100
KDD99	1.133 4	1.127 5	1.039 7	1.058 3	1.113 6
Weka-1.8G	1.648 9	2.134 2	2.135 2	1.463 7	1.692 6
Weka-3.2G	1.480 6	1.569 3	1.588 8	1.529 0	1.483 1
Weka-6.4G	**2.652 9**	**2.140 1**	**2.065 3**	**1.760 1**	**1.587 5**
Average	1.728 9	1.742 8	1.707 2	1.452 8	1.469 2

5 结 论

粗糙集算法已经广泛应用于数据挖掘与知识发现等领域.传统的粗糙集算法无法处理日益增长的海量数据.通过云计算中的 MapReduce 技术,用户可以很容易地使用云计算平台处理海量数据.不断变化的数据加大了云计算环境下知识更新的成本.本文结合 MapReduce 模型与增量更新算法,设计与实现了两种基于 MapReduce 的并行增量更新算法,并在较大的 4 个数据集上进行测试.结果表明:数据越多,增量算法取得的效果越好.同时,增量模型可以很好地利用原有结果,有效地减少运行时间,加快基于粒计算和粗糙集的海量数据挖掘与知识发

现等的过程,继而降低云计算环境下知识更新的成本,对不断更新的实时模型有更好的支持,可以有效地应用于实时性要求更强的在线应用.我们将在未来的研究中,继续探讨基于粗糙集和 MapReduce 的并行增量算法在海量知识发现与知识提取中的应用与实现.

References:

[1] Blaszczynski J, Slowinski R, Szelag M. Sequential covering rule induction algorithm for variable consistency rough set approaches. Information Sciences, 2011,181(5):987−1002. [doi: 10.1016/j.ins.2010.10.030]

[2] Chen YM, Miao DQ, Wang RZ, Wu KS. A rough set approach to feature selection based on power set tree. Knowledge-Based Systems, 2011,24(2):275−281. [doi: 10.1016/j.knosys.2010.09.004]

[3] Hong TP, Tseng LH, Chien BC. Mining from incomplete quantitative data by fuzzy rough sets. Expert Systems with Applications, 2010,37(3):2644−2653. [doi: 10.1016/j.eswa.2009.08.002]

[4] Inuiguchi M, Miyajima T. Rough set based rule induction from two decision tables. European Journal of Operational Research, 2007,181(3):1540−1553. [doi: 10.1016/j.ejor.2005.11.054]

[5] Kaneiwa K. A rough set approach to multiple dataset analysis. Applied Soft Computing, 2011,11(2):2538−2547. [doi: 10.1016/j.asoc.2010.08.021]

[6] Leung Y, Fischer M, Wu WZ, Mi JS. A rough set approach for the discovery of classification rules in interval-valued information systems. Int'l Journal of Approximate Reasoning, 2008,47(2):233−246. [doi: 10.1016/j.ijar.2007.05.001]

[7] Miao DQ, Duan QG, Zhang HY, Na J. Rough set based hybrid algorithm for text classification. Expert Systems with Applications, 2009,36(5):9168−9174. [doi: 10.1016/j.eswa.2008.12.026]

[8] Wang XZ, Zhai JH, Lu SX. Induction of multiple fuzzy decision trees based on rough set technique. Information Sciences, 2008, 178(16):3188−3202. [doi: 10.1016/j.ins.2008.03.021]

[9] Wu WZ, Zhang WX, Li HZ. Knowledge acquisition in incomplete fuzzy information systems via the rough set approach. Expert Systems, 2003,20(5):280−286. [doi: 10.1111/1468-0394.00252]

[10] Hu QH, Yu DR, Xie ZX. Neighborhood classifiers. Expert Systems with Applications, 2008,34(2):866−876. [doi: 10.1016/j.eswa.2006.10.043]

[11] Yao YY. Three-Way decisions with probabilistic rough sets. Information Sciences, 2010,18(3):341−353. [doi: 10.1016/j.ins.2009.09.021]

[12] Zhang QH, Wang GY, Xiao Y. Approximation sets of rough sets. Ruan Jian Xue Bao/Journal of Software, 2012,23(7):1745−1759 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4226.htm [doi: 10.3724/SP.J.1001.2012.04226]

[13] Shan N, Ziarko W. Data-Based acquisition and incremental modification of classification rules. Computational Intelligence, 1995, 11(2):357−370. [doi: 10.1111/j.1467-8640.1995.tb00038.x]

[14] Guo S, Wang ZY, Wu ZC, Yan HP. A novel dynamic incremental rules extraction algorithm based on rough set theory. In: Proc. of the 4th Int'l Conf. on Machine Learning and Cybernetics. Piscataway: IEEE Press, 2005. 1902−1907. [doi: 10.1109/ICMLC.2005.1527256]

[15] Zheng Z, Wang GY. RRIA: A rough set and rule tree based incremental knowledge acquisition algorithm. Fundamenta Informaticae, 2004,59(2-3):299−313.

[16] Fan YN, Tseng TL, Chern CC, Huang CC. Rule induction based on an incremental rough set. Expert Systems with Applications, 2009,36(9):11439−11450. [doi: 10.1016/j.eswa.2009.03.056]

[17] Wang L, Wang GY, Wu Y. An incremental rule acquisition algorithm based on variable precision rough set model. Journal of Chongqing University of Posts and Telecommunications (Natural Science), 2005,17(6):709−713 (in Chinese with English abstract).

[18] Liu D, Li TR, Ruan D, Zhang JB. Incremental learning optimization on knowledge discovery in dynamic business intelligent systems. Journal of Global Optimization, 2011,51(2):325−344. [doi: 10.1007/s10898-010-9607-8]

[19] Zhang JB, Li TR, Ruan D, Liu D. Neighborhood rough sets for dynamic data mining. Int'l Journal of Intelligent Systems, 2012, 27(4):317−342. [doi: 10.1002/int.21523]

[20] Chen K, Zheng WM. Cloud computing: System instances and current research. Ruan Jian Xue Bao/Journal of Software, 2009,20(5): 1337−1348 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[21] Liu Y, Xu CF, Pan YH. A parallel approximate rule extracting algorithm based on the improved discernibility matrix. Lecture Notes in Artificial Intelligence, 2004,3066:498−503.

[22] Qian J, Miao DQ, Zhang ZH. Knowledge reduction algorithms in cloud computing. Chinese Journal of Computers, 2011,34(12): 2332−2343 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.02332]

[23] Zhang JB, Li TR, Ruan D, Gao ZZ, Zhao CB. A parallel method for computing rough set approximations. Information Sciences, 2012,194(1):209−223. [doi: 10.1016/j.ins.2011.12.036]

[24] Zhang JB, Li TR, Pan Y. Parallel rough set based knowledge acquisition using MapReduce from big data. In: Proc. of the 1st Int'l Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications (BigMine 2012). New York: ACM Press, 2012. 20−27. [doi: 10.1145/2351316.2351320]

[25] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM, 2008,51(1):107−113. [doi: 10.1145/1327452.1327492]

[26] White T. Hadoop: The Definitive Guide. 2nd ed., Sebastopol: O'Reilly Media/Yahoo Press, 2010.

[27] Pawlak Z. Rough Sets: Theoretical Aspects of Reasoning about Data. Dordrecht: Kluwer Academic Publishers, 1991.

[28] Xu ZY, Liu ZP, Yang BR, Song W. A quick attribute reduction algorithm with complexity of max($O(|C||U|)$,$O(|C|^2|U/C|)$). Chinese Journal of Computers, 2006,29(3):391−398 (in Chinese with English abstract). [doi: 10.3321/j.issn:0254-4164.2006.03.006]

[29] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: An update. SIGKDD Explorations, 2009,11(1):10−18. [doi: 10.1145/1656274.1656278]

[30] Zhang JB, Li TR, Chen HM. Composite rough sets for dynamic data mining. Information Sciences, 2014,257:81−100. [doi: 10.1016/j.ins.2013.08.016]

附中文参考文献:

[12] 张清华,王国胤,肖雨.粗糙集的近似集.软件学报,2012,23(7):1745−1759. http://www.jos.org.cn/1000-9825/4226.htm [doi: 10.3724/SP.J.1001.2012.04226]

[16] 王利,王国胤,吴渝.基于可变精度粗集模型的增量式规则获取算法.重庆邮电学院学报,2005,17(6):709−713. [doi: 10.1016/j.eswa.2009.03.056]

[20] 陈康,郑纬民.云计算:系统实例与研究现状.软件学报,2009,20(5):1337−1348. http://www.jos.org.cn/1000-9825/3493.htm [doi: 10.3724/SP.J.1001.2009.03493]

[22] 钱进,苗夺谦,张泽华.云计算环境下知识约简算法.计算机学报,2011,34(12):2332−2343. [doi: 10.3724/SP.J.1016.2011.02332]

[28] 徐章艳,刘作鹏,杨炳儒,宋威.一个复杂度为 max($O(|C||U|)$,$O(|C|^2|U/C|)$)的快速属性约简算法.计算机学报,2006,29(3):391−398. [doi: 10.3321/j.issn:0254-4164.2006.03.006]

张钧波(1986－),男,浙江余姚人,博士生,CCF 学生会员,主要研究领域为云计算,数据挖掘,粒计算,粗糙集.

罗川(1987－),男,博士生,CCF 学生会员,主要研究领域为数据挖掘,粗糙集,粒计算.

李天瑞(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据挖掘,粗糙集,粒计算,云计算.

滕飞(1984－),女,博士,讲师,CCF 会员,主要研究领域为数据挖掘,云计算.

潘毅(1960－),男,博士,教授,博士生导师,主要研究领域为并行计算与云计算,互联网络技术,生物信息学.