第47卷第4期 中国科学技术大学学报 Vol.47,No.4
2017年4月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Apr. 2017

文章编号:0253-2778(2017)4-0350-08

基于Spark的组合分类器链多标签分类方法

王 进,王 鸿,夏翠萍,欧阳卫华,陈乔松,邓 欣

(重庆邮电大学计算智能重庆市重点实验室,重庆 400065)

摘要:随着数据挖掘技术在现实问题中的广泛应用,多标签学习现已成为数据挖掘技术中的一个研究热点.组合分类器链(ECC)算法是一种性能较好的多标签分类方法,其分类效果好、准确度高,但该算法的时空复杂度较高,不能适应大规模多标签数据分类任务.为此提出了一种基于Spark的组合分类器链多标签分类方法,将串行组合分类器链算法的各步骤进行了并行化实现.通过单机实验和集群并行化实验,证明该方法对大规模多标签数据集具有良好的适应能力和加速比,且分类效果不输于传统的串行多标签分类方法.

关键词:多标签学习;组合分类器链;Apache Spark;MapReduce

中图分类号:TP391 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2017.04.010

引用格式:王进,王鸿,夏翠萍,等. 基于Spark的组合分类器链多标签分类方法[J]. 中国科学技术大学学报,2017,47(4):350-357.

WANG Jin, WANG Hong, XIA Cuiping, et al. Ensembles of classifier chains for multi-label classification based on Spark[J]. Journal of University of Science and Technology of China, 2017,47(4):350-357.

Ensembles of classifier chains for multi-label classification based on Spark

WANG Jin, WANG Hong, XIA Cuiping, OUYANG Weihua, CHEN Qiaosong, DENG Xin

(*Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing* 400065, *China*)

Abstract: With the wide application of data mining technology, multi-label learning has become a hot topic in the data mining domain. Although ensembles of classifier chains (ECC) algorithm is a multi-label learning method which is effective and accurate, its complexity of time and space is so high that it cannot adapt to the large-scale multi-label classification tasks. A new algorithm named Spark ensembles of classifier chains(S-ECC) was proposed based on Spark platform on which a parallel implementation was conducted of each step of the sequential ECC algorithm. The test results in stand-alone and cluster environments show that S-ECC has a good adaptability to large-scale data with a high speedup, and that it is no less capable than the traditional sequential program.

Key words: multi-label learning, ECC; Apache Spark; MapReduce

0 引言

随着信息技术的发展,互联网数据规模激增,表现形式丰富多样.传统有监督学习认为,每个样本只具有一个标签,缺乏准确表述事物的复杂语义信息的能力.现实世界中的事物很复杂,通常同时拥有多

收稿日期:2016-08-28;**修回日期**:2016-12-08

基金项目:重庆市基础与前沿研究计划(cstc2014jcyjA40001,cstc2014jcyjA40022),重庆教委科学技术研究项目(KJ1400436)资助.

作者简介:王进(通讯作者),男,1979年生,博士/教授. 研究方向:数据挖掘、人工智能. E-mail: wangjin@cqupt.edu.cn

个语义，有效明确解释事物具有的多个语义的一个直接方法就是给一个事物标注多个标签，因此多标签学习(multi-label learning)应运而生.在多标签学习中，每个样本可能包含一个或多个标签，被多个标签标注的样本能够更好地表现事物语义信息的多样性[1].

多标签组合分类器链(ECC)算法[2]是多标签分类算法的一种，其核心思想是将多标签学习问题转换为一个或者多个单标签的学习过程.该算法考虑了标签之间的关联性并且加入了随机因素，在实际使用中分类效果好，但由于训练阶段需要对同一样本集进行不同标签顺序的多次训练，构建多个训练模型，因此时间复杂度和空间复杂度较高，随着数据量的增大，采用传统串行算法难以应对规模越来越大的数据集，出现运行时间过长，内存溢出等情况，不能满足工程需求.近几年来，大数据技术的发展为解决此类问题提供了理想的条件和思路.Apache Spark 是伯克利大学 AMP 实验室研发的类 Hadoop MapReduce 的通用并行框架[3]，Spark 拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要多次迭代的 MapReduce 算法.弹性分布式数据集(RDD)是 Spark 的核心概念，它是一个容错的、并行的数据结构，可以让用户显式地将数据存储到磁盘和内存中，并能控制数据的分区情况，通过使用 RDD 提供的数据并行接口，可以像操作本地数据那样操作分布式数据[3].由于使用了内存计算技术，RDD 非常适合于需要多次迭代的机器学习算法优化.

本文基于 Spark 大数据处理技术，提出了一种并行化组合分类器链算法(Spark ensembles of classifier chains，S-ECC)，旨在使这一传统的多标签分类算法适应大规模多标签数据分类的要求.在 S-ECC 训练阶段，并行的训练多个随机分类器链，并在分类器链内部并行训练每个基分类器；在算法预测阶段，利用数据并行策略将测试数据集划分为多个分区，在每一分区上同时执行预测算法，得到每个测试集样本的预测结果.通过对算法每一阶段采用不同的并行策略，极大地提高了并行算法的性能.在单机和集群环境的实验证明，算法具有良好的大数据处理能力和稳定性，在集群资源充足的情况下，随着并行程序使用的计算核心数的增加，程序的加速比线性增长.

1　相关研究

1.1　Spark 并行框架

Apache Spark 是加州大学伯克利分校所开源的类 Hadoop 的 MapReduce 并行计算框架.Spark 保留了 Hadoop MapReduce 的优点，可以把计算过程中产生的中间结果缓存在内存中，从而减少磁盘读写开销，提高计算速度，在一些实验中较 Hadoop MapReduce 可以获得近百倍的速度提升[3]，因此 Spark 能更好地运用于需要多次迭代的数据挖掘和机器学习算法.

弹性分布式数据集(RDD)是 Spark 的核心数据结构.通过 RDD，Spark 可以基本一致地应用于不同的大数据处理场景，如 MapReduce、实时流数据、SQL、Machine Learning 和图计算等.

RDD 可以由包括本地文件系统、HDFS、HBase、Hive 等任何被 Hadoop 支持的存储源创建，也可以通过 Scala 内存数据集合创建.创建 RDD 后，用户可以设置 RDD 的存储级别，将 RDD 缓存在内存或磁盘中，下次重复使用时就不需重新计算，提高了程序性能.RDD 支持的操作可以分为转换操作和行动操作两种类型，其中转换操作从现有的 RDD 产生一个新的 RDD，行动操作在 RDD 上执行某种计算返回一个结果值.图 1 为 RDD 的操作类型示意图.

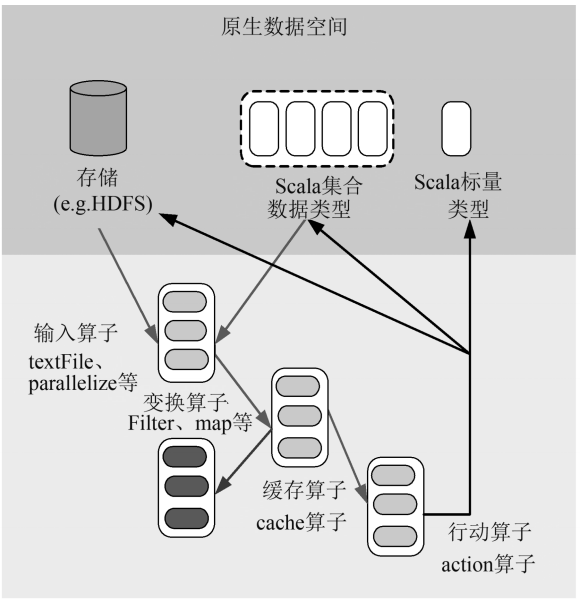

图 1　RDD 操作类型

Fig.1　Operation type of RDD

随着Spark技术的产生与发展，为了适应大数据应用的要求，很多机器学习和数据挖掘领域的算法被设计应用于Spark平台，获得了成倍的性能提升.Zhu[4]等将子空间聚类算法基于Spark并行化实现，应用于大数据集获得了较大的速度提升.Maillo[5]等提出了一种基于Spark的迭代精确K最近邻算法，该算法充分利用Spark内存计算的机制，性能比使用Hadoop MapReduce实现的相同算法提高了将近10倍.Kim[6]等结合深度学习技术、Spark大数据处理技术和GPU加速等技术，开发出一套运行于Spark上的深度学习框架.文献[7]提出了一种在内存不足的情况下自动缓存合适RDD的选择和替换算法，进一步提高了Spark程序的性能.

1.2 多标签学习和ECC算法

1.2.1 多标签学习定义

设$X=\mathrm{R}^d$是d维特征空间，$L=\{l_1, l_2, \cdots, l_q\}$为整个标签集合.训练集$T=\{D^{(n)}\mid 1\leqslant n\leqslant N\}$，每个样本$D^{(n)}=(x^{(n)},Y^{(n)})$由$d$维特征向量$x^{(n)}\in \mathrm{R}^d$和与$x^{(n)}$关联的标签子集$Y^{(n)}\in 2^L$组成.其中$Y^{(n)}$可表示为一个$q$维向量$y(n)=[y_1^{(n)},y_2^{(n)},\cdots,y_q^{(n)}]$，若$l_j\in Y^{(n)}$则$y_j^{(n)}=1$，否则$y_j^{(n)}=0$.多标签学习方法通过训练集得到映射函数$h:X\to 2^L$和$f:X\times L\to \mathrm{R}$，分别应对多标签分类和标签排序两个任务.对于测试样本$x^{(t)}$，通过函数h得到与其相关的标签集合$Y^{(t)*}$.($Y^{(t)*}$也可表示为q维二值标签向量$y^{(t)*}$)；通过函数f得到每个标签与样本的关联度$fy^{(t)*}=[fy_1^{(t)*},fy_2^{(t)*},\cdots,fy_q^{(t)*}]$，其中关联度也可看作对应标签与样本相关的置信度，是标签排序的依据，相关标签关联度通常大于不相关标签.

1.2.2 二元关系(BR)算法

BR方法[8]是一种较为常见和易于理解的多标签学习算法，它将每个标签的预测当作一个独立的单标签分类问题，这样就可以使用现有的单标签分类算法分别预测每个标签.由于BR方法没有考虑标签之间的关联性，因此分类效果往往不理想.

1.2.3 分类器链(CC)算法

分类器链算法的基本思想是将多标签学习问题转化为一条链式的二分类问题[2,9].CC算法实际上仍属于BR算法，但是与一般的BR算法不同，它将每个标签的单分类器组成一条分类器链，在预测时，链前面的分类器预测出的结果将会加入后面分类器的属性空间.图2给出了BR算法和CC算法在转化多标签问题时的不同.其中示例样本为(x,y)，$x=[0,1,0,1,0,0,1,1,0]$，$y=[1,0,0,1,0]$(为简便起见，设属性值和标签值为0/1二元表示).算法1描述了CC的训练过程.

BR转化

$h:x\longrightarrow$	y
$h1$:[0,1,0,1,0,0,1,1,0]	1
$h2$:[0,1,0,1,0,0,1,1,0]	0
$h3$:[0,1,0,1,0,0,1,1,0]	0
$h4$:[0,1,0,1,0,0,1,1,0]	1
$h5$:[0,1,0,1,0,0,1,1,0]	0

CC转化

$h:x'\longrightarrow$	y
$h1$:[0,1,0,1,0,0,1,1,0]	1
$h2$:[0,1,0,1,0,0,1,1,0,1]	0
$h3$:[0,1,0,1,0,0,1,1,0,1,0]	0
$h4$:[0,1,0,1,0,0,1,1,0,1,0,0]	1
$h5$:[0,1,0,1,0,0,1,1,0,1,0,0,1]	0

图2 BR和CC算法比较

Fig.2 The comparison between algorithms BR and CC

算法1.1 CC的训练过程

输入：训练集D={(x_1,y_1),…,(x_m,y_m)},q

输出：分类器链h={h_i,i∈1,…,L}

1.for(j=0; j<q ; j++)

//第j个二元分类器的训练和转换

2.D_j←{}

3.for(x,y)∈D

4.x′←[x_1,…,x_d,y_1,…,y_{j-1}]

5.D'_j←D'_j∪(x′,y_j)

//训练分类器h_j to 预测第j个标签y_j

6.h_j:D'_j→{0,1}

执行训练过程后，一个二元分类器链h={h_1,…,h_L}训练完成.算法2描述了CC的预测过程.

算法1.2 CC的预测过程

输入：分类器链h={h_1,…,h_L},测试样本x

输出：样本的标签向量$\hat{y}$

1.$\hat{y}$←[$\hat{y}_1$,…,$\hat{y}_L$]

2.for(j=1,j<L,j++)

3.x′←[x_1,…,x_d,$\hat{y}_1$,…,$\hat{y}_{j-1}$]

4.$\hat{y}_j$←h_j(x′)

5.return$\hat{y}$

1.2.4 组合分类器链(ECC)算法

为了克服CC算法的局限性，Read等[2]在CC算法的基础上提出了ECC算法，ECC算法基于CC算法，每次训练多个随机的分类器链，并得到多个CC预测结果，对多个结果进行投票，得到最终结果，这样在很大程度上降低了上述的风险.ECC算法描述如下：

步骤1 随机产生一条标签链，执行CC训练算法，得到CC分类模型C_j，$j\in\{1,\cdots,N\}$；

步骤 2　使用 C_j 对样本 x 进行预测，得到预测结果 L_j；

步骤 3　重复上述步骤 1 和步骤 2 多次，每个样本获得多个标签向量 $L=\{L_1,\cdots,L_N\}$；

步骤 4　用标签向量 L' 对每个标签进行投票，得到最终标签向量 $\hat{y}$.

由于 ECC 算法采取多条随机产生的不同标签序列的 CC 组合，减轻了单个 CC 由内部二分类器排列顺序问题带来的不利影响，因此 ECC 在分类效果上表现良好，在多标签数据挖掘中获得了广泛的应用，但由于其需要训练多个 CC 模型，算法的时空复杂度较高，难以适应大规模数据集的多标签分类任务.

2　S-ECC 算法

本文结合 Spark 并行计算技术，设计并实现了基于 Spark 的并行 ECC 算法(S-ECC 算法)，图 3 是 S-ECC 算法示意图.由于 ECC 算法实质上是在 CC 算法的基础上做集成，ECC 算法依赖于 CC 算法，因此本文也设计和实现了基于 Spark 的并行 CC 算法(S-CC 算法).

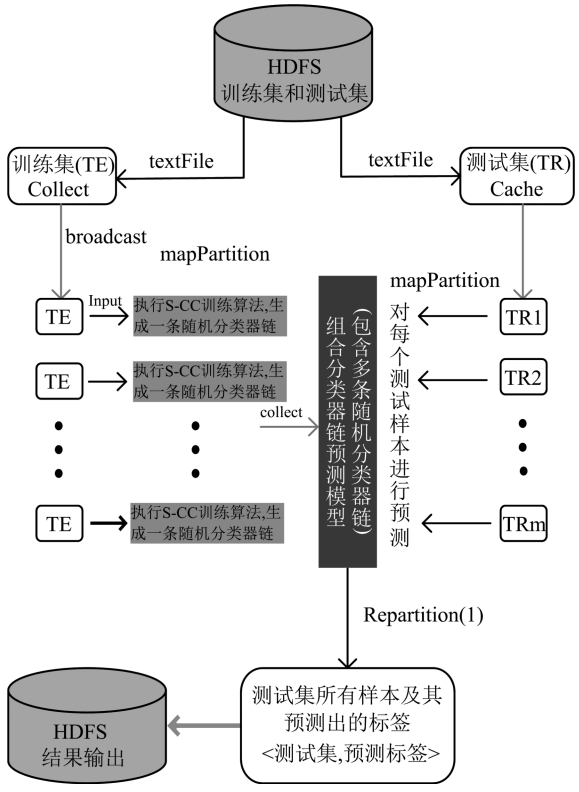

图 3　S-ECC 算法示意图

Fig.3　S-ECC algorithm schematic

2.1　S-CC 算法

在训练阶段 CC 需要训练多个二元分类器，这些分类器的训练过程是独立且互不影响的，因此可以对每个分类器并行训练.实际上分类器内部的训练过程也可以并行化，即采用并行化的基分类器训练算法，但是这种策略对于 CC 算法并不合适.由于多标签数据集的标签维数通常很大，并行地训练每一个基分类器的并行度已经很高，且并行的基分类算法为了减少时空复杂度和通信开销，通常采用一些近似优化方法，导致其算法精度较低.

训练结束后，CC 得到了一条由所有的基分类器组成的分类器链.在预测阶段分类器链依次预测每一个标签.由于链中前一个基分类器的输出作为后一个基分类器的输入，因此基分类器之间不能并行地预测.在预测一个样本标签集合时，由于每个样本的预测过程彼此独立，故可以将样本集合划分为多个分区，每个分区包含一个样本集合的子集，所有分区的数据子集并行地进行预测，最终汇总得到全部预测结果.

2.1.1　训练阶段

在 S-CC 算法的训练阶段，由于需要学习的训练集每个样本的所有真实标签都已知，因此分类器链上的基分类器可以并行训练，训练完成后将所有基分类器组成分类器链.算法步骤如下：

(Ⅰ)读入并广播训练集 $D=\{(x_1,y_1),..(x_m,y_m)\}$，使用 Spark 提供的广播方法将训练集 D 广播到集群的每一台机器上；

(Ⅱ)构造一个含有 L 个基分类器的 RDD，其中 L 为 D 的标签数，并为其中的每个基分类器编号 j ($1\leqslant j\leqslant L$)，设构造的 RDD 为 rdd1，类型为 RDD[(序号，基分类器)]；

(Ⅲ)对 rdd1 执行 Map 操作，Map 函数流程如下：

①初始化基分类器，设置分类器参数.

②由 D 生成新的训练集：

$$D'=\{(x'_1,y'_1),\cdots,(x'_m,y'_m),x'_j=[x_1,\cdots,x_d,y_1,\cdots,y_{j-1}],y'_j=y_j\}.$$

式中，j 为基分类器编号，且 $1<=j<=L$.

③将 D' 放入基分类器中训练，构建分类模型.

④返回一个二元组，其中第一个元素为编号，第二个元素为经过训练的基分类器.

(Ⅳ)对经过 Map 操作后生成的新 RDD 执行 Reduce 操作，将其变成一个本地集合保存下来.

2.1.2 预测阶段

在S-CC算法训练阶段执行完成之后，分类器链已经构建成功，算法进入预测阶段.由于需要进行预测的测试集数据的预测过程彼此独立，因此可以在测试集上实现数据并行，算法步骤如下：

(Ⅰ)读入测试集 T，生成一个包含所有测试集样本的RDD，记为rdd1；

(Ⅱ)对rdd1执行Map操作，Map函数流程如下：

①提取测试样本特征向量tf；

②使用CC方法根据tf预测出样本的所有标签

③返回预测标签向量 $\boldsymbol{p}$.

(Ⅲ)对于经过Map操作后生成的新RDD执行Reduce操作，将预测出的所有测试样本标签聚集为一个本地集合 P；

(Ⅳ)返回 P.

2.2 S-ECC算法

ECC算法每次训练多个随机的分类器链，由于这些分类器链的训练过程相互独立，因此可以并行训练多条随机分类器链.在预测阶段，对于任一样本，ECC算法利用训练完成的多条分类器链分别对同一个样本预测一个结果，再对所有结果汇总进行投票得到最终的结果，其中多条分类器链可以并行地预测同一个样本.在预测一个样本集合时，可以使用S-CC算法的并行策略，将样本集合划分为多个分区，不同分区并行预测，汇总得到整个样本集的预测结果.

2.2.1 训练阶段

S-ECC算法的训练阶段由S-CC算法的训练过程构成，其算法步骤如下：

(Ⅰ)读入并广播训练集 $D=\{(x_1,y_1),\cdots,(x_m,y_m)\}$，使用Spark提供的广播方法将训练集 D 广播到集群的每一台机器上；

(Ⅱ)生成包含有 N 条随机乱序的标签序号链的RDD，记为LC，其中 N 为分类器链的条数；

(Ⅲ)对LC执行Map操作，Map函数流程如下：

①按照乱序标签序号链的顺序执行一次S-CC算法，对 D 进行学习，生成一个S-CC分类模型 M；

②返回 M.

(Ⅳ)对经过Map操作后生成的新RDD执行Reduce操作，将该RDD转化为包含有N个分类器链的本地集合Models，将Models保存，供预测阶段使用.

2.2.2 预测阶段

S-ECC对于每个测试样本的预测是独立的，因此可以数据并行.算法步骤如下：

(Ⅰ)读入测试集T，生成一个包含所有测试集样本的RDD，记为rdd1；

(Ⅱ)对rdd1执行Map操作，Map函数流程如下：

①提取测试样本特征向量tf；

②将训练的 N 个分类器链Models分别预测tf，得到 N 个预测结果 $P(P=[p_1,\cdots p_N])$；

③将 P 内部的所有结果进行投票，得到最终结果 P'；

④返回 P'.

(Ⅲ)对经过Map操作后生成的新RDD执行Reduce操作，得到整个测试集每个样本预测结果的集合.

2.2.3 算法复杂度分析

在S-ECC算法的训练阶段，由于采用了基于Spark的并行策略，将每一个单标签二元分类器的全部训练任务并行执行，因此在计算资源充足的情况下，理论上每个计算节点上算法的时间复杂度较ECC算法降低了 $N*L$ 倍.其中 N 为组合分类器链的数目，L 为样本的标签数.理论上每个计算节点的空间复杂度降低了约2倍，这是因为ECC算法在串行训练基分类器时需要存储一份完整的训练集，然后从中生成一份用于训练基分类器的单标签训练集.对于S-ECC算法，每个计算节点只需要存储用于训练基分类器的单标签训练集即可.

在S-ECC算法的预测阶段，由于将测试集分为了多个分区，每个分区并行进行预测，且单样本预测过程中每条分类器链也是并行预测的，因此在计算资源充足的情况下，理论上每个计算节点上的时间复杂度较ECC算法降低了 $N*P$ 倍.其中 N 为组合分类器链数，P 为测试集分区数.理论上每个计算节点的空间复杂度降低了 P 倍.

3 实验与结果分析

3.1 评价指标

实验中，从以下几个方面来评价算法的性能和可扩展性：①汉明损失(Hamming loss).Hamming loss用于衡量样本中的误分标签平均数量，Hamming loss越小，误分标签数量越少，多标签学

习方法的性能越好，指标最优取值为 0.②微观 F_1 值（Micro F_1-measure）.与 Hamming loss 相比，Micro F_1-measure 更关注相关标签的分类结果.Micro F_1-measure 值越大，多标签学习方法的性能越好，指标最优取值为 1.③排序损失（Ranking loss）.Ranking loss 用于衡量样本中相关标签的关联度小于不相关标签的关联度的标签对的比例.Ranking loss 越小，多标签学习方法的性能越好，指标最优取值为 0.④运行时间（Runtime）.在试验中，测试了 S-ECC 算法运行时间.⑤加速比（Speedup）.为了验证并行算法和串行算法的效率提升，测试了算法加速比：

$$\text{Speedup}=\frac{\text{base\_line}}{\text{parallel\_time}} \tag{1}$$

式中，base_line 为串行程序运行时间，parallel_time 为并行程序运行总时间.

3.2　数据集

实验使用了五个大规模多标签数据集，均为 Mulan 多标签学习库[10]提供，数据集分别为：EUR-Lex（directory codes）[11]，EUR-Lex（subject matters）[11]，MediaMill[12]，NUS-WIDE-cVLADplus[13]，NUS-WIDE-bow[14].数据集概要信息如表 1 所示.

表 1　实验数据集概要信息

Tab.1　Summary of experimental data sets

数据集名	样本数	特征数	标签数
EUR-Lex (directory codes)	19 348	5 000	412
EUR-Lex (subject matters)	19 348	5 000	201
MediaMill	43 907	120	101
NUS-WIDE-cVLADplus	269 648	128	81
NUS-WIDE-bow	269 648	500	81

3.3　实验环境

本实验使用的 Spark 平台搭建在真实的物理集群上，表 2 为集群硬件概要信息，表 3 为集群软件概要信息.

表 2　集群硬件概要信息

Tab.2　Cluster hardware overview

Intel(R)至强 E5 CPU 数量(个)	CPU 计算核心数量(个)	内存容量 (GB)	硬盘容量 (TB)
32	192	1024	64

表 3　集群软件概要信息

Tab.3　Cluster software overview

操作系统	Cent OS6.5
Java 版本	1.7.0_71
Scala 版本	2.10.4
Hadoop 版本	2.5.2
Spark 版本	1.4.0

3.4　实验结果与分析

本文对串行 ECC 程序和并行 S-ECC 程序分别进行了实验，其中串行程序使用 Mulan[10]多标签算法库中的程序，基分类器选择了经典的 C4.5 决策树.Mulan 程序使用 Java 语言实现，与 Spark 程序均运行于 JVM 上，提高了实验的公平和客观性.串行 ECC 算法的实验结果如表 4 所示.

表 4　串行 ECC 算法实验结果

Tab.4　Experimental results of sequential ECC algorithm

数据集	汉明损失	微观 F_1 值	排序损失	运行时间/s
EUR-Lex (directory codes)	DNF	DNF	DNF	DNF
EUR-Lex (subject matters)	DNF	DNF	DNF	DNF
MediaMill	0.031	0.556	0.054	13 856
NUS-WIDE-cVLADplus	0.022	0.230	0.084	246 415
NUS-WIDE-bow	DNF	DNF	DNF	DNF

表 4 中，DNF 代表程序在一周之内没有运行完毕，从中可以看到，串行 ECC 算法在大数据集上运行很慢，无法在规定的时间内得到结果.此外，实验分别选取了一阶策略、二阶策略、高阶策略[15]中经典的多标签算法（BR，CLR[16]，RAkEL[17]），以 J48 为基分类器验证 ECC 的有效性.由于高阶多标签算法时间复杂度高的特点，实验验证了上述算法在两个较小数据集上的性能.表 5 至表 8 分别为各算法在 MediaMill、NUS-WIDE-bow 上的实验结果，加粗处为最好结果.实验结果表明，ECC 算法分类效果好，时间复杂度高，因此并行化 ECC 具有很大的现实意义.

表 5　汉明损失对比

Tab.5　Comparison of Hamming loss in different algorithms

	汉明损失			
数据集	ECC	BR	RAkEL	CLR
MediaMill	**0.031**	0.038	0.034	0.031
NUS-WIDE-cVLADplus	**0.022**	0.031	0.030	0.026

表 6 微观 F_1 值对比

Tab.6 Micro F_1 value

微观 F_1 值				
数据集	ECC	BR	RAkEL	CLR
MediaMill	**0.556**	0.505	0.551	0.550
NUS-WIDE-cVLADplus	0.230	0.147	**0.243**	0.227

表 7 排序损失对比

Tab.7 Comparison of Ranking loss in different algorihms

排序损失				
数据集	ECC	BR	RAkEL	CLR
MediaMill	0.054	0.201	0.134	**0.043**
NUS-WIDE-cVLADplus	**0.084**	0.298	0.197	0.129

表 8 运行时间对比

Tab.8 Comparison of Run time loss in different algorithms

运行时间				
数据集	ECC	BR	RAkEL	CLR
MediaMill	13 856	**1 231**	5 304	4 896
NUS-WIDE-cVLADplus	246 415	**20 932**	58 963	74 645

从表 9 中可以看到 S-ECC 和经典的串行程序在多标签分类的精度上基本相同.图 4 所示为 S-ECC 算法在使用不同线程数(# map)情况下的运行时间.

表 9 S-ECC 算法多标签分类效果

Tab.9 The effects of multi label classification with S-ECC algorithm

数据集	汉明损失	微观 F_1 值	排序损失
EUR-Lex (directory codes)	0.002	0.701	0.042
EUR-Lex (subject matters)	0.005	0.732	0.057
MediaMill	0.030	0.551	0.054
NUS-WIDE-cVLADplus	0.024	0.229	0.081
NUS-WIDE-bow	0.025	0.201	0.117

图 4 显示 S-ECC 程序的运行时间随着使用的线程数的增加而下降,但是下降的速度逐渐变慢,这是由于 Map 阶段数据集的广播和 Reduce 阶段数据聚合时产生的通信开销造成的.图 4 所示为 S-ECC 相对于串行程序的加速比,由于在规定时间内串行程序只在 MediaMill 和 NUS-WIDE-cVLADplus 两个数据集上运行完成,得到结果,因此只画出了这两个数据集上算法的加速比.

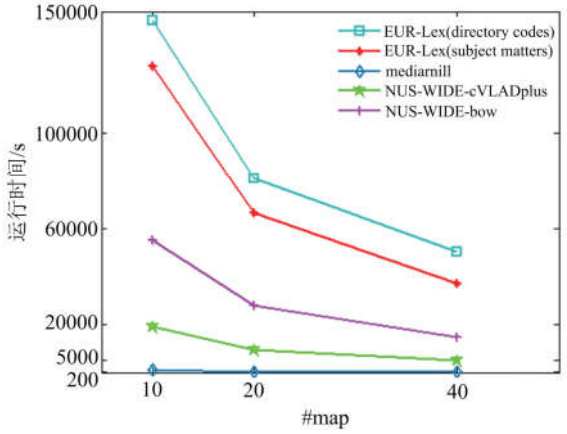

图 4 S-ECC 运行时间

Fig.4 Runtime of S-ECC algorithm

图 5 中,S-ECC 算法在实验数据集上的加速比随节点数近似线性增长,在 10 个、20 个和 40 个节点上的加速比分别大于 10、20 和 40,这主要是由两种原因造成的:一是在计算总时间时加入了读写文件的时间,分布式文件系统和 Spark 对读写文件的优化让并行程序读写文件的效率更高;二是单机环境下可能出现内存不足的情况,单机程序只能使用虚拟内存,而在集群环境下不仅内存充足,而且还有更多的高速缓存可以使用.加速比显示了 S-ECC 算法对大规模多标签分类问题的良好适应能力和可扩展性.

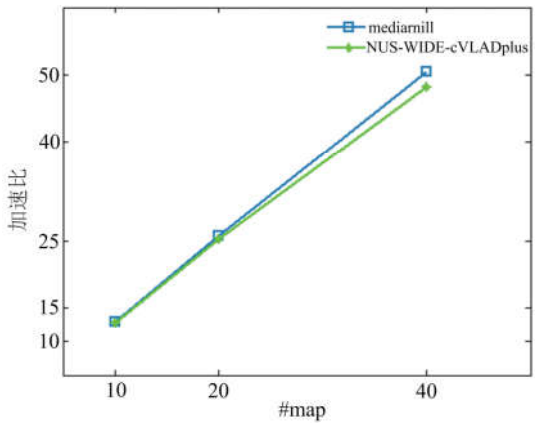

图 5 S-ECC 加速比

Fig.5 Speedup of S-ECC algorithm

4 结论

本文基于 Spark 并行框架设计和实现了 ECC 算法的并行版本 S-ECC 算法,目的是使 ECC 这一分类性能较好的多标签分类方法适应于大规模多标签数据分类,满足工程和实际应用的需求.结合

MapReduce 并行编程思想和 Spark 本身提供的内存迭代计算技术，提高了 S-ECC 适应大数据集的能力.在单机和集群上的实验表明，S-ECC 算法具有很好的加速比和扩展性，相对于串行程序，S-ECC 的分类精确度和其他评价指标并没有受到影响.

后续工作将进一步优化 S-ECC 算法性能，展开对其他分类效果较好的多标签算法的并行化研究，使现有的多标签算法可以应用于海量数据挖掘的应用场景中去.

参考文献(References)

[1] ZHANG M L, ZHOU Z H. A review on multi-label learning algorithms [J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(8): 1819-1837.

[2] READ J, PFAHRINGER B, HOLMES G, et al. Classifier chains for multi-label classification [J]. Machine Learning, 2011, 85(3): 333-359.

[3] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing [C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. San Hose, USA: USENIX Association, 2012: 141-146.

[4] ZHU B, MARA A, MOZO A. CLUS: Parallel subspace clustering algorithm on spark [A]// New Trends in Databases and Information Systems [M]. Springer, 2015, 539:175-185.

[5] MAILLO J, RAMÍREZ S, TRIGUERO I, et al. kNN-IS: An iterative spark-based design of the *k*-nearest neighbors classifier for big data[J]. Knowledge-Based Systems, 2016, 117: 3-15; doi: 10.1016/j.knosys.2016.06.012.

[6] KIM H, PARK J, JANG J, et al. DeepSpark: Spark-based deep learning supporting asynchronous updates and Caffe compatibility [J/OL]. https://arxiv.org/abs/1602.08191v1, 2016.03.08, 2016: arXiv: 1602.08191v1.

[7] DUAN M X, LI K L, TANG Z, et al. Selection and replacement algorithms for memory performance improvement in Spark[J]. Concurrency & Computation Practice & Experience, 2015, 28(8): 2473-2486.

[8] TSOUMAKAS G, KATAKIS I, VLAHAVAS I. Mining multi-label data [A]// Data Mining and Knowledge Discovery Handbook[M]. Springer, 2009: 667-685.

[9] READ J, PFAHRINGER B, HOLMES G. Multi-label classification using ensembles of pruned sets [C]// Proceedings of the 8th International Conference on Data Mining. Pisa, Italy: IEEE Computer Society, 2008: 995-1000.

[10] TSOUMAKAS G, SPYROMITROS-XIOUFIS E, VILCEK J, et al. Mulan: A Java library for multi-label learning[J]. Journal of Machine Learning Research, 2011, 12(2): 2411-2414.

[11] MENCÍA E L, FÜRNKRANZ J. Efficient pairwise multilabel classification for large-scale problems in the legal domain[C]// European Conference on Machine Learning & Knowledge Discovery in Databases. Antwerp, Belgium: Springer, 2008: 50-65.

[12] SNOEK C G M, WORRING M, VAN GEMERT J C, et al. The challenge problem for automated detection of 101 semantic concepts in multimedia[C]// Proceedings of the 14th ACM International Conference on Multimedia. Santa Barbara, USA: ACM Press. 2006: 421-430.

[13] CHUA T S, TANG J, HONG R, et al. NUS-WIDE: A real-world web image database from National University of Singapore[C]// Proceedings of the ACM International Conference on Image and Video Retrieval. Santorini, Greece: ACM Press, 2009: doi: 10.1145/1646396.1646452.

[14] SPYROMITROS-XIOUFIS E, PAPADOPOULOS S, KOMPATSIARIS I Y, et al. A comprehensive study over VLAD and product quantization in large-scale image retrieval[J]. IEEE Transactions on Multimedia, 2014, 16(6): 1713-1728.

[15] ZHANG M L, WU L. LIFT: Multi-label learning with label-specific features [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 37(1): 107-20.

[16] FÜRNKRANZ J, HÜLLERMEIER E, MENCÍA E L, et al. Multilabel classification via calibrated label ranking[J]. Machine Learning, 2008, 73(2): 133-153.

[17] TSOUMAKAS G, KATAKIS I, VLAHAVAS I. Random *k*-labelsets for multilabel classification [J]. IEEE Transactions on Knowledge & Data Engineering, 2011, 23(7): 1079-1089.