第 45 卷第 4 期
2013 年 8 月

南 京 航 空 航 天 大 学 学 报
Journal of Nanjing University of Aeronautics & Astronautics

Vol. 45 No. 4
Aug. 2013

KNN 分类算法的 MapReduce 并行化实现

闫永刚[1] 马廷淮[1,2] 王 建[3]

(1. 南京信息工程大学计算机与软件学院,南京,210044; 2. 南京信息工程大学江苏省网络监控中心,南京,210044;
3. 南京大学电子科学与工程学院,南京,210093)

摘要:为了提高 k-nearest neighbor algorithm(KNN)算法处理大数据集的能力,本文利用 MapReduce 并行编程模型,同时结合 KNN 算法自身的特点,给出了 KNN 算法在 Hadoop 平台下的并行化实现。通过设计 Map、Combine 和 Reduce 3 个函数,实现了 KNN 算法的并行化。Map 函数完成每个测试样本与训练样本之间的相似度计算,Combine 函数作为一个本地的 Reduce 操作,用以减少中间计算量及通信开销,Reduce 函数则根据上述函数得到的中间结果计算出 k 近邻并作出分类判断。实验结果表明:较之以往的单机版方法,在 Hadoop 集群上实现的并行化 KNN 算法具有较好的加速比和良好的扩展性。

关键词:KNN 分类;并行计算;MapReduce 模型;Hadoop

中图分类号:TP391 **文献标志码**:A **文章编号**:1005-2615(2013)04-0550-06

Parallel Implementing KNN Classification Algorithm Using MapReduce Programming Mode

Yan Yonggang[1], *Ma Tinghuai*[1,2], *Wang Jian*[3]

(1. School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China;
2. Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, 210044, China; 3. School of Electronic Science and Engineering, Nanjing University, Nanjing, 210093, China)

Abstract: In order to improve the ability of KNN algorithm to process massive data, a new technique based on Hadoop platform is used. Considering the characteristics of the KNN algorithm itself, the parallelism of KNN based on the MapReduce programming model is implemented. Three functions are designed for the implementation of the parallelism, named Map, Combine and Reduce. The Similarity between each test instances and the training dataset are evaluated by Map function. For reducing the computational complexity and saving network bandwidth, the Combine function is used as a local Reduce operation. Reduce function is used to get the KNN classification based on the intermediate results. The experiment on the Hadoop platform shows the method has excellent linear speedup with an increasing number of computer nodes and good scalability.

Key words: KNN classification; parallel computing; MapReduce programming model; Hadoop

随着信息技术的进步以及信息化社会的发展,在科学研究、计算机仿真、互联网应用和电子商务[1]等领域,数据量呈现快速增长的趋势。比如,大型强子对撞机每年积累的新数据量为 15 PB 左右;沃尔玛公司每天通过 6 000 多个商店,向全球客户销售超过 2.67 亿件商品等。为了分析和利用这些庞大的数据资源,必须依赖有效的数据分析技术。数据挖掘技术是一种通过分析海量数据从中

基金项目:国家自然科学基金(61173143)资助项目;江苏省自然科学基金(BK2010380)资助项目;中国博士后科学基金(2012M511303)资助项目;江苏省高校优势学科建设工程资助项目。

收稿日期:2012-09-06;**修订日期**:2012-10-08

通信作者:马廷淮,男,教授,1974 年 3 月出生,E-mail:thma@nuist.edu.cn。

寻找潜在规律的方法，其中 KNN 算法是一种理论成熟且简单常用的分类算法，其时间复杂度较高，因此无法胜任大数据分析任务。为了解决传统数据挖掘算法面对海量数据无法处理的难题，当前，学术界主要从 3 个方面展开研究。

(1)编程模型。它主要解决 2 个基本问题[2]：一是编程模型的简单实用性，开发人员只需关注要解决的问题，不需要关心应用程序在大规模集群上运行的细节(如数据分布、调度任务和处理容错等)；二是编程模型要保证应用程序能在大规模集群上高效率、高可靠和可扩展地运行。然而传统的并行编程模型(如 P threads，Message Passing Interface(MPI)和 Open MP 等)抽象度不高，并且需要开发人员熟悉底层的配置细节，因此在扩展性方面遇到了巨大的障碍。

(2)云计算平台。当前主要的云计算平台有 AbiCloud，Amazon 和 Hadoop 等，其中 Apache 社区的 Hadoop 项目[3]是用 Java 语言实现的模型，其开源的特点、强大的功能和良好的特性使得目前有大量企业和研究机构使用 Hadoop 进行研究或者利用 Hadoop 组建自己的云平台[4-5]。

(3)算法移植及模型扩展。要实现基于云计算的数据挖掘技术，关键就在于将传统的数据挖掘算法移植到云平台，目前很多国内外学者都对算法移植做了研究。文献[6]实现了 Hadoop 平台下关联规则挖掘算法 Apriori 的并行化。文献[7]对若干数据挖掘算法如协同过滤、朴素贝叶斯等算法展开了并行化研究，并在 Hadoop 平台上进行了集群实验，结果表明上述算法不仅可以实现 MapReduce 并行化，而且取得了良好的加速比效果。文献[8]实现了贝叶斯分类器的 MapReduce 并行化并将其应用到垃圾短信过滤系统中。文献[9]实现了 ID3 算法在 Hadoop 平台上的移植。此外，文献[10]中 Papadimitriou 等人提出了一个分布式协同聚类算法 DisCo，并将其在 Hadoop 平台上进行了 MapRedue 并行化实现，证明了 DisCo 算法能有效处理海量数据。文献[11]采用 MapReduce 的方式在多核集群上对多种机器学习算法进行了研究，结果表明随着核数目的增多，集群加速比呈现线性关系。此外，在 MapReduce 编程模型扩展和更新方面，Yahoo 公司通过在 MapReduce 步骤之后加入一个 Merge 步骤，形成了一个新的 MapReduce-Merg[12]框架，从而使得框架开发人员可以用自己提供的 Merge 函数做两个数据集合的合并操作。

另外，基于 Hadoop 平台的 Mahout 是一个 Apache开源机器学习子项目。它提供了一些数据挖掘领域的经典算法的并行化实现，包括分类、聚类、频繁子项挖掘算法等，是一个带命令行操作方式的并行数据挖掘工具箱。

本文在充分考察前人研究的基础上，结合 KNN 算法自身的特点，研究并实现 KNN 算法的并行化，同时将该算法移植到 Hadoop 平台上并进行实验。

1　MapReduce 简介

MapReduce[13]是 Google 公司提出一种并行编程模型，主要用于解决大规模数据集的并行计算问题。它首先对大数据集按块进行分割，然后将数据块分发到若干节点上，最后根据用户自定义的 Map/Reduce 任务执行相应的操作。在整个计算过程中，数据的输入输出均是采用〈key，value〉键值对的形式。详细的步骤如下：

(1)首先，用户程序对上传到 HDSF 文件目录中的数据进行分割(大小一般为 16～64 MB，其值可通过参数来设定)。

(2)使用 MapReduce 中的 Jobtracker 为集群分配 Map 或者 Reduce 任务。其中 Reduce 任务可由用户自己设定，而 Map 任务则是由文件块数决定的。

(3)执行 Map 任务，此步骤是由 worker 节点来执行的。Map 函数将当前节点上的文件解析成〈key，value〉键值对，之后根据 Map 函数的功能执行相应的操作，并将结果以键值对的形式缓存到当前节点的内存中。

(4)重新洗牌，这一步骤是与 sort 步骤同时进行的。主要是将相同 key 的记录送到同一个 Reduce 节点，而 sort 则是按照 key 值对 Reduce 数据进行分组。

(5) Reduce 操作，Reducer 将所读取到的〈key，list(value)〉键值对使用 Reduce 函数进行计算，得到最终结果并将其输出。

(6)待每个 Map 和 Reduce 任务都执行完毕后，Master 节点便唤醒用户程序并报告任务执行的相关信息。至此，一个完整的 MapReduce 任务便执行完成了。

从以上步骤可以看出，MapReduce 的核心是 Map 函数和 Reduce 函数，它们的功能是按一定的映射规则将输入的键值对〈key，value〉转换成另一个或一批键值对〈key，list(value)〉。Map 函数将大数据集解析成一批〈key，value〉键值对，计算处

理后生成中间结果 List(〈k2,v2〉);Reduce 函数将〈k2, List(v2)〉作为输入数据,对属于同一个 key 的 value 集进行计算处理。因此编程人员只需关注 Map 函数和 Reduce 函数就可以实现比较复杂的并行计算任务。MapReduce 的这种"分而治之"的思想可以有效地降低每一部分的运算复杂度,从而达到提高运算效率的目的。

2 KNN 算法的 MapReduce 并行化实现方法

2.1 KNN 分类算法的基本思路

KNN 算法最初是由 Cover 和 Hart 于 1967 年提出的[14],是一种典型的基于类比的学习算法。其算法思路比较简单:给定一个待分类的样本 x,首先找出与 x 最接近的或最相似的 K 个已知类标号的训练集样本,然后根据这 K 个训练样本的多数类的类标号来确定样本 x 的类标号。其算法串行实现的时间复杂度比较高,为 $O(nt)$,其中 n 为训练样本的总数,t 为特征属性的数目。可以看出,随着 n,t 增加,KNN 的计算量将会以 $n*t$ 乘积的形式迅速增长。因为每个待分类的样本都可以独立地进行 KNN 分类,前后之间没有计算顺序上的相关性,因此可以用并行化方法来解决 KNN 算法串行复杂度高的问题。

2.2 KNN 分类算法的 MapReduce 并行化方法

KNN 分类算法进行 MapReduce 并行化的基本思路:首先,Worker 节点从 HDFS 文件系统中将训练样本集和测试样本集下载到本地节点,之后对每个测试样本启动一次 Map 计算过程,完成测试样本到训练样本的距离计算,最后将中间计算结果送到 Reduce 节点上进行规约操作并生成最终结果。为了适合 MapReduce 计算模型,在 Map 操作之前要将待处理数据以行形式存储。具体操作就是对数据进行按行分片,要求片间数据无关,默认分片操作是由 MapReduce 自动完成的,不需要用户编写代码,当然,用户也可以自己给定。

2.2.1 Map 函数的设计

Map 函数的任务是读取训练测试集,形成键值对的映射。具体做法是首先调用内置函数(Split)按行读取样本并将其转换成特定格式的文件,以利于后期操作。之后遍历每个测试样本,计算其与每个训练样本之间的相似度,最后将结果存入 context 集合。输入数据〈key, value〉对的形式为〈行号,样本〉;输出数据〈key, value〉对的形式为〈行号,向量集合〈训练样本的类标识,相似度值〉〉,相似度值采用标准的欧氏距离度量。相应的伪代码如图 1 所示。

```
Input: 〈key, value〉
  Output: 〈key, vector〉Context context
    Begin
      //遍历所有训练样本 i,取出其类标识 t
     1: For i=0 to n (training dataset) do
     2:  t=FindCatalog (i);
      //遍历测试样本 k, 计算其与训练样本的欧氏距离,key 为
      测试样本的行号,结果存入 Context
     3:  For all k ∈ testfile do
     4:    Distance=EuclideanDistance (k, i);
     5:    Context. write (key, vector (t, Distance));
     6:  End For
     7:  End For
    End
```

图 1 Map 函数伪代码

2.2.2 Combine 函数的设计

当本地的 Map 函数执行完后,对于任意一个测试样本 k 都包含了 n(分配给本地计算的训练样本数目)个计算结果,如果此时不加处理地全部提交到 Reduce 节点去计算,一般而言 Reduce 节点数目是远少于 Map 节点的,这样势必会造成计算资源的浪费。为了最大限度地利用计算资源,本文设计了 Combine 函数,它相当于本地的 Reduce 操作,它可以从上一步骤 Map 函数输出的集合中取出键值对重新洗牌(即把数据重新组织,作为 Reduce 节点的输入。如本文中在本地节点取出 K 个最近邻操作),节约 Reduce 节点的计算及通信开销,因而使得整个并行化显得更为高效。具体步骤为:首先接受 Map 节点的键值对,存入 ArrayList 集合,然后对相同 Key 的 value 执行排序操作(从小到大)。根据用户设定的 K 值,先做一次最近邻计算,对每个测试样本输出前 K 条记录,最后传给 Reduce 节点。相应的伪代码如图 2 所示。

2.2.3 Reduce 函数的设计

Reduce 函数是进行规约操作的,它的设计比较简单。其主要的任务是取出 K 个近邻,计算多数类的类别,并将其赋予测试样本。后台系统需要执行 shuffle 操作,以便将不同 Map 节点上相同 Key(即同一个测试样本)的数据送到同一个 Reduce 节点上执行排序、分类以及输出操作。其中输入数据〈key, value〉对的形式为〈测试样本的

ID，vector(训练样本类标识，相似度值)〉，输出数据〈key，value〉对的形式为〈测试样本的 ID，训练样本的类标识〉。相应的伪代码如图 3 所示。

```
Input：〈key，vector〉
  Output：〈key，vector〉Context context
    Begin
      //遍历每个测试样本，加入到 ArrayList 中执行排序
      value 为 Map 函数输出的 vector (t，Distance)
    1：  For all key and value do
    2：    ArrayList.add (vector (t，value))；
    3：    Sort (ArrayList)；
           //取 k 个最近邻，结果存入 Context
    4：    Context. write (key，ArrayList. get (k))；}
    5：  End For
    End
```

图 2　Combine 函数伪代码

```
Input：〈key，vector〉
Output：〈key，ID〉Context context
Begin
  //将键值对添加到 ArrayList 中执行排序操作，t 为训练样本的类标识
  1：  For all key and value do
  2：    ArrayList (vector (t，value))；
  3：    Sort (ArrayList)；
  4：    New ArrayList result；
         //将前 k 个样本添加到 result 中
  5：      result. add (key，ArrayList.get (k))；
         //应用传统 KNN 计算多数类的类别，结果存入 Context 中
  6：      Context. write (key，Tradition KNN (result))；
  7：  End For
End
```

图 3　Reduce 函数伪代码

2.3　算法复杂度分析

2.3.1　时间复杂度分析

在 MapReduce 编程模型中，HDFS 将大数据分割成若干 blocks，然后存储在不同的节点上。每个节点根据 Map 函数指定的操作在本地完成相应的功能。在本文中 KNN 计算量最大的部分(计算欧氏距离)被分散到了多个节点并行处理。如果集群规模为 p，那么其时间复杂度为：$O(ns)/p$。其中 n 为训练样本的总数，s 为特征属性的数目。

2.3.2　空间复杂度分析

因 KNN 算法是惰性算法，即它一次性将所有样本读入内存，故传统 KNN 的空间复杂度与样本数量呈线性关系，为 $O(n)$，n 为样本数目。但当 p 个节点参与 KNN 分类时，无论样本被划分成多少 blocks，总样本都是被分摊到 p 个节点中，因此算法的空间复杂度为 $O(n)/p$。

3　实验和结果分析

3.1　实验环境

实验集群规模为 7 台电脑，主要配置如下：CPU 型号为 Intel Core i3-380；内存为 2 GB；硬盘容量为 320 GB，Hadoop 版本为 0.20.2。集群的配置是参照 Hadoop 官方网站推荐的方法来搭建的。

3.2　实验及结果分析

实验数据。本文采用了人工合成数据和标准测试两种数据。人工合成数据是由 Matlab 中的随机函数 $R=\text{gamrnd}(A,B,v)$ 产生，本文合成的是具有 11 维属性的二分类数据，共计 1 251 110 个训练样本，大小约 100 MB。标准测试数据集 Cover-Type(依据地质变量来预测森林植被覆盖类型)取自于 UCI Machine learning repository。该数据集是具有 54 维的七分类数据，共计 581 012 个样本，其中训练样本为 387 342 个，测试样本为 183 671 个，数据集大小约 71.6 MB。

实验设置。本文采用的是基于 Java 的开源数据挖掘工具 Weka[15]，由于它集成了数据挖掘的大部分算法，包括数据预处理、分类和聚类等，且具有良好的交互界面，因此被学术界广泛使用。为了解决 Weka 系统在数据集较大时出现的内存不足等异常情况，实验前需要修改 RunWeka.ini 文件(将 maxheap 改为 400 MB 或者更大)。同时配置 TaskTracke 节点子任务的 Java virtual machine (JVM)内存 mapred.child.java.opts 为 1 000 MB。

实验结果。从图 4 中可以看出，当节点数目很少(1～2 个)时，基于 Hadoop 的并行 KNN 算法的处理效率远低于单机版处理效率，没有体现集群的优势，主要原因如下：对于 Hadoop 集群而言，一是当集群规模较小时，Job 的启动以及交互都需要消耗一定的时间，尤其是当实际计算任务时间在总用时中所占的比例较小时，这种集群优势就更难得到体现。二是网络传输速度的影响，主要体现在集群环境中数据的读取依赖于网络的传输速度，而对于

单机版的KNN而言,数据只需从本地一次性读入内存中运行,所以处理速度非常快。然而当节点数超过3个时,集群优势开始显现,节点数目为7时,用时节省非常明显。由此可知:随着节点数的增多,Hadoop集群的执行效率更加优于单机上的执行效率。

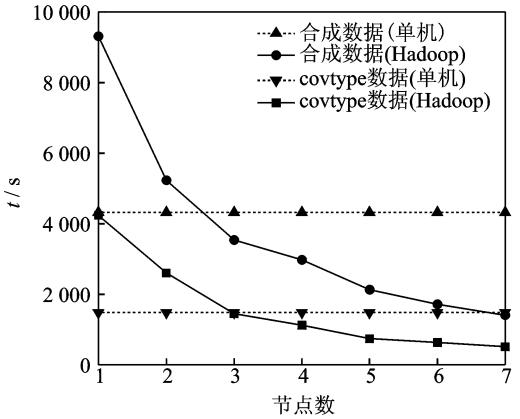

图4 单机与Hadoop集群执行时间的对比图

加速比参数是衡量一个系统在扩展性方面优劣的主要指标。从图5中可以看出,加速比是随着节点数目的增加而提升的,节点的增加可以显著地提高系统对同规模数据的处理能力。另外,从Hadoop集群相对于单机上的运行效率的相对加速比测试结果看,数据量越大,相对加速比也就越大,运行效率也就越好。这说明MapReduce在处理KNN分类算法上具有较好的加速比,可以预见,当节点数目更多时,这种性能优势将体现得更加明显。

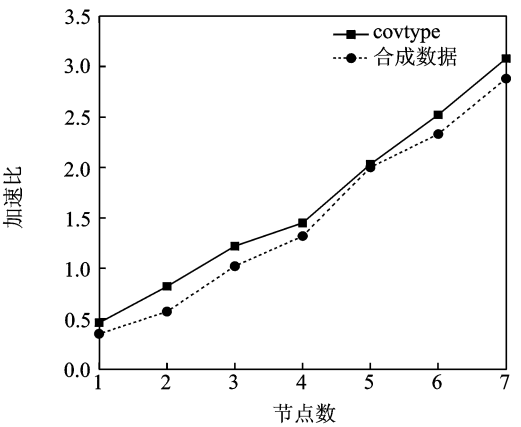

图5 加速比实验图

3.3 KNN的并行模型比较

当前,KNN的并行模型主要有多线程模型、消息传递模型、MapReduce 3种模型。现将MapReduce模型与其他两种模型作概括性的比较:

(1)多线程模型:如OpenMp并行模型,它是通过内存共享实现信息交互的,其优点是编程相对简单,将串行程序转换为并行程序时无须对代码作大的改动。不足之处是只能在共享存储结构的机器上运行,不能用于集群,不适应于大数据量的计算任务。如文献[16]采用OpenMp模型实现了KNN的并行化,实验结果所得出的加速比平均为1.3左右,虽然执行时间优于单机,但加速比不明显。

(2)消息传递模型:如MPI,它提供了一种与平台无关,可以被广泛使用的编写消息传递程序的标准,其优点是可以在集群上使用,因此可伸缩性很强。但它是采用进程间通信的方式来协调并行计算的,由此产生了并行效率较低、内存开销大、编程复杂等问题。

本文所采用的MapReduce编程模型由于隐藏了并行计算的细节(通信机制、负载均衡等),可以有效避免前两种模型所面临的问题,具有很好的应用价值。

4 结束语

本文主要探讨了KNN分类算法的并行化实现方案问题,并使用MapReduce编程模型将该算法在Hadoop平台上进行了设计和实现。相关实验结果表明在Hadoop平台上处理大数据集时,在数据规模相同的情况下,算法的执行效率与集群的规模有着直接关系。节点越多,加速比越明显,效率越高。

参考文献:

[1] Zhou A Y. Data intensive computing-challenges of data management techniques[J]. Communications of CCF, 2009,5(7):50-53.

[2] 王鹏,孟丹,詹剑锋,等.数据密集型计算编程模型研究进展[J].计算机研究与发展,2010,47(11):1993-2002.
Wang Peng, Meng Dan, Zhan Jianfen, et al. Review of programming models for data-intensive computing [J]. Journal of Computer Research and Development,2010,47(11):1993-2002.

[3] Apache Hadoop. The apache software foundation [EB/OL]. http://hadoop.apache.org, 2012-04-01.

[4] 陈康,郑纬民.云计算:系统实例与研究现状[J].软件学报,2009,20(5):1337-1348.
Chen Kang, Zheng Weimin. Cloud computing: system instances and current research[J]. Journal of

Software, 2009,20(5):1337-1348.

[5] 陈全，邓倩妮. 云计算及其关键技术[J]. 高性能计算发展与应用，2009，29(9)：2562-2566.
Chen Quan, Deng Qianni. Cloud computing and its key techniques[J]. Journal of Computer Applications, 2009,29(9):2562-2566.

[6] 李玲娟，张敏. 云计算环境下关联规则挖掘算法的研究[J]. 计算机技术与发展，2011，21(2)：43-46.
Li Lingjuan, Zhang Min. Research on algorithm of mining association rule under cloud computing environment[J]. Computer Technology and Development, 2011,21(2):43-46.

[7] 李军华. 云计算及若干数据挖掘算法的 MapReduce 化研究[D]. 西安：电子科技大学，2010.
Li Junhua. Research on cloud computing and some data mining mapreduce[D]. Xi′an: University of Electronic Science and Technology of China, 2010.

[8] 朱杰. 云计算在基于贝叶斯分类的垃圾短信过滤中的研究与应用[D]. 西安：电子科技大学，2010.
Zhu Jie. The junk message filter based on Bayes classification theory and cloud computing [D]. Xi′an: University of Electronic Science and Technology of China, 2010.

[9] 纪俊. 一种基于云计算的数据挖掘平台架构设计与实现[D]. 青岛：青岛大学，2009.
Ji Jun. Design and implement the prototype of data mining platform based on cloud computing[D]. Qing Dao: Qing Dao University, 2009.

[10] Papadimitriou Spiros, Sun Jimeng. DisCo: distributed co-clustering with map-reduce[C]//2008 Eighth IEEE International Conference on Data Mining. Pisa: IEEE,2008:512-521.

[11] Chu C T, Kim S K, Lin Y A. Mapreduce for machine learning on multicore[C]//Proceedings of Neural Information Processing Systems Conference19. Vancouver: MIT Press, 2006:281-288.

[12] Yang H C, Dasdan A, Hsiao R L. Map-reduce-merge: Simplified relational data processing on large clusters[C]//Proc of the 2007 ACM SIGMOD Int. Conf on Management of Data. New York: ACM, 2007:1029-1040.

[13] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1):107-113.

[14] Cover T M, Hart P E. Nearest neighbor pattern classification [J]. IEEE Transactions on Information Theory, 1967,13(1):21-27.

[15] University of Waikato. Waikato Environment for Knowledge Analysis [EB/OL]. http://www.waikato.ac.nz/,2012-02-12.

[16] 王宗跃，马洪超. 多核 CPU 的海量点云并行 KNN 算法[J]. 测绘科学技术学报，2010，27(1)：46-49.
Wang Zongyue, Ma Hongchao. K-nearest neighbors algorithm for large scale point clouds data with muticore CPU[J]. Journal of Geomantic Science Technology, 2010, 27(1):46-49.