

一种基于 Hadoop 的改进减法聚类算法

原 旭¹, 陈志奎¹, 赵 亮¹, 杨德礼²

(1 大连理工大学 软件学院, 辽宁 大连 116620; 2 大连理工大学 管理学部, 辽宁 大连 116024)

摘 要: 传统的减法聚类算法时间复杂度高, 算法不具有分布式特性, 不满足大数据处理的要求. 提出一种基于 Hadoop 的改进减法聚类算法, 利用 MapReduce 模型改进减法聚类执行过程, 实现求解邻域半径、初始化密度指标、更新密度指标和划分数据记录等过程的并行化. 实验结果表明, 同传统的串行算法相比, 提出的算法能够对大数据进行快速聚类, 同时表现出良好的稳定性与扩展性.

关键词: 减法聚类; Hadoop; MapReduce 模型; 大数据

中图分类号: TP391; TP301.6

文献标识码: A

文章编号: 1000-7180(2015)03-0151-05

An Improved Subtractive Clustering Algorithm Based on Hadoop

YUAN Xu¹, CHEN Zhi-kui¹, ZHAO Liang¹, YANG De-li²

(1 School of Software Technology, Dalian University of Technology, Dalian 116620, China;

2 School of Management, Dalian University of Technology, Dalian 116024, China)

Abstract: Traditional subtractive clustering algorithm's time complexity is pretty high, and it doesn't have the characteristic of distributed processing. Therefore, it is not suitable for the processing requirement in big data environment. This paper proposes an improved subtractive clustering algorithm which is based on Hadoop. It applies multiple MapReduce processes to implement the parallelization of subtractive clustering in solving neighborhood radius, initializing density index, updating density index and dividing the data records. Experiment demonstrates that comparing to the traditional serial algorithms, the proposed improved algorithm can indeed cluster the big data fast and has good stability and expansibility.

Key words: subtractive clustering; Hadoop; MapReduce; big data

1 引言

随着信息化社会的发展, 数据正以前所未有的速度增长. 面对越来越大的数据规模, 传统的聚类算法在时间复杂性和空间复杂性上都遇到了前所未有的挑战^[1-2]. 2004 年 Google 公司提出的一种分布式编程模型 MapReduce, 它能组织集群来处理大规模数据集, 成为云计算平台主流的并行数据处理模型. 随后 Apache 开源社区的 Hadoop 项目用 java 语言实现了该模型, 同时 Hadoop 项目还设计了开放源代码的云计算技术平台, 为大数据处理带来了方便条件^[3], 同时也为大数据环境下聚类算法的实现提

供了有力支撑.

为实现海量数据聚类分析, 现阶段一些学者已经实现多种基于分布式编程模型的并行聚类算法: 文献[4-5]中实现 k -means 算法的 MapReduce 并行化设计; 鲁伟明等人^[6]提出一种基于 MapReduce 的分布式近邻传播聚类算法; 文献[7]中给出一种基于 MapReduce 的分布式网络数据聚类算法; Qiwei Zhong 等人提出基于 MapReduce 的分布式谱聚类算法^[8]; 曹泽文等人提出基于云计算的海量文本聚类方法^[9]; 文献[10]中提出基于 MapReduce 的图聚类算法; 文献[11]中提出一种网格化聚类算法的 MapReduce 并行化研究.

收稿日期: 2014-09-01; 修回日期: 2014-10-28

基金项目: 国家自然重点基金项目(U1301253)

虽然基于 Hadoop 分布式平台的聚类算法研究很多,但是还没有针对于密度聚类的 MapReduce 并行化算法.因此,本文针对基于密度聚类的减法聚类^[12]算法进行并行化研究,提出一种基于 Hadoop 平台的改进减法聚类算法.本算法首先将减法聚类求解数据点间距离过程转换成矩阵相乘求解,并利用 MapReduce 模型实现矩阵求解的并行化^[13],生成数据点间距离矩阵.然后再利用距离矩阵中的数据项和 MapReduce 模型实现减法聚类求解邻域半径、初始化密度指标、更新密度指标和划分数据记录四个过程的并行化.实验的结果表明,相比传统的减法聚类算法,本文算法能够对大数据进行更加快速的聚类,同时表现出良好的稳定性与扩展性.

2 基于 Hadoop 的改进减法聚类算法

2.1 改进的减法聚类算法

从传统的减法聚类算法执行过程可以看出,算法的主要计算步骤为计算每一个数据点与其他所有数据点间的距离.本文提出的改进减法聚类算法将求解数据点间的距离转换成矩阵求解,即将原始数据集备份成两份,每一数据集的数据点表示矩阵行,数据点的每一维数据表示矩阵行中的一个数据项,进而形成矩阵 M 和 N ,这样按照公式

$$(MN)_{ij} = \sum_{r=1}^n (a_{ir} - a_{jr})^2 = (a_{i1} - a_{j1})^2 + (a_{i2} - a_{j2})^2 + \dots + (a_{in} - a_{jn})^2 \quad (1)$$

求解矩阵的过程即为计算数据点间的距离,得到距离矩阵 G .然后利用 MapReduce 计算模型实现矩阵求解的并行化.最后利用得到的距离矩阵和 MapReduce 模型对求解邻域半径、初始化密度指标、更新密度指标和划分数据点等过程进行并行化处理.

2.2 减法聚类算法的并行化设计

2.2.1 求解距离矩阵的 MapReduce 并行化设计

2.1 节中提出利用矩阵求解思想对计算数据点间距离进行优化,当矩阵 M 和 N 很大时,我们可以将矩阵进行分块,然后将矩阵 M 中的分块数据分别与矩阵 N 中的分块数据按照式(1)进行求解,通过 Hadoop 平台实现求解过程的并行处理.生成数据点距离矩阵的 MapReduce 并行算法.具体步骤如下:

输入:数据点矩阵 M 和 N

输出:数据点间距离矩阵 G

①Map 函数读取矩阵 M 和 N 中的每条数据记录,并对 M 中每条数据记录中的元素 $M[i, j]$ 产生

一系列的键值对 $\langle (i, k), (M, j, M[i, j]) \rangle$, 其中 $k = 1, 2, \dots$, 直到矩阵 N 的行数.同样,对于 N 中每条记录中的元素 $N[i, j]$, 产生一系列的键值对 $\langle (k, i), (N, j, N[i, j]) \rangle$, 其中 $k = 1, 2, \dots$, 直到矩阵 M 的行数.

②Combine 函数收集所有 Map 过程的输出,将具有相同 key 值的数据项合并到一起形成 $\langle \text{键值}, \{\text{记录集合}\} \rangle$, 然后将所有数据项转送给多个 Reduce 过程.

③在 Reduce 函数中,将具有相同键值的 $(M, j, M[i, j])$ 和 $(N, j, N[i, j])$ 分别按照 j 值排序并放在不同的两个列表里面.将两个列表中的第 j 个元素 $M[i, j]$ 个 $N[i, j]$ 相减求平方,然后将这些结果项相加得到 sum, Reduce 函数最后输出键值对 $\langle (i, k), \text{sum} \rangle$.

④ $\langle (i, k), \text{sum} \rangle$ 中的 sum 值表示第 i 个数据点与第 k 个数据点的距离,进而得到数据点间距离矩阵 G .

在数据点间距离矩阵求解的过程中,通过设置 MapReduce 计算模型初始化全局邻域半径变量,并在 Reduce 函数中累加所有 sum 结果,即可得到邻域半径的计算结果,邻域半径为

$$r_a = r_b = \frac{1}{2} \sqrt{\frac{1}{n(n-1)} \sum_{j=1}^n \sum_{i=1}^n ||x_i - x_j||^2} \quad (x_i, x_j \text{ 为数据集中记录}).$$

2.2.2 初始化密度指标的 MapReduce 并行化设计

通过密度指标求解公式 $D_i = \sum_{j=1}^n \exp\left[-\frac{||x_i - x_j||^2}{(0.5r_a)^2}\right]$ (x_i, x_j 为数据集中记录)可知,距离矩阵 G 的第 i 行的所有值恰好与数据对象 i 的密度指标 D_i 的各元素求解相对应.这个特性保证密度指标的计算适合 MapReduce 并行化设计.计算数据点密度指标的 MapReduce 并行算法具体步骤如下:

输入:数据点间距离矩阵 G

输出:各数据点的密度指标 D_i

①Map 函数读取距离矩阵数据项,各数据项以键值对 $\langle (i, k), \text{sum} \rangle$ 的形式存储,拆分数据项条目生成键值对 $\langle i, \text{sum} \rangle$ 的数据项.

②Combine 函数收集所有 Map 过程的输出数据项,将具有相同键值的 sum 归并到一起形成 $\langle \text{键值}, \{\text{记录集合}\} \rangle$, 然后将所有数据项转送给多个 Reduce 过程.

③Reduce 函数将得到的具有相同键值的数据

项按式(3)计算求和,得到各个数据点的密度指标 D_i , 输出 $\langle i, D_i \rangle$ 形式的键值对。

在数据点密度求解的过程中,通过设置 MapReduce 计算模型初始化全局最大密度指标变量,并在 Reduce 函数中比较密度指标值,可得到最大的密度指标,即可得到对应的初始聚类中心点。

2.2.3 更新密度指标的 MapReduce 并行化设计

通过密度指标更新公式 $D_i = D_i - D_{c_k} \exp\left[-\frac{\|x_i - x_{c_k}\|^2}{(0.5r_k)^2}\right]$ (x_i 为数据集中的记录, x_{c_k} 为第 k 次选出的聚类中心, D_{c_k} 为其对应的密度指标)可以看出,上一次求解的聚类中心点 x_{c_k} 在距离矩阵 G 中对应的行的所有值恰好与数据对象 i 的密度指标 D_i 的更新相对应。这个特点保证密度指标的更新过程适合 MapReduce 并行化处理。实现更新数据点密度指标的 MapReduce 并行算法的具体步骤如下:

输入:上一次求解的聚类中心点 x_{c_k} 在距离矩阵 G 中对应的行,各数据项上一次求解的密度指标

输出:更新后的各数据点密度指标 D_i

步骤:

①Map 函数读取上一次求解的聚类中心点在距离矩阵 G 中对应的行中的数据项,各数据项以键值对 $\langle (c_k, i), (\text{sum}, D_i) \rangle$ 的形式存储,拆分数据项并按照式(4)计算新的数据点密度指标 D_i ,生成新的键值对 $\langle i, D_i \rangle$ 。

②将更新后的密度指标键值对 $\langle i, D_i \rangle$ 写入文件。

由于此并行执行过程不需要 Reduce 过程参与,所以在设计 MapReduce 计算模型时可以省去 Reduce 函数的设计。此外在 MapReduce 计算模型初始化的过程中设置全局最大密度指标变量,并在 Map 函数中比较密度指标值,可得到更新后的密度指标最大值,即可得到本次求解过程对应的聚类中心点。比较本次求得的聚类中心点的密度指标与初始聚类中心点的密度指标,如果满足收敛条件结束密度指标更新过程,否则重新配置 MapReduce 编程模型,进入下一次密度指标的求解过程,得到新的中心点。

2.2.4 划分数据点的 MapReduce 并行化设计

当所有聚类中心点都得到以后,要将数据集中的其他数据点划分到各聚类中心的代表簇中。本文利用欧氏距离计算其他数据点与各聚类中心的距离,并将其划分到与之最近的聚类中心点所代表的

簇中。具体的 MapReduce 并行算法执行过程如下:

输入:各聚类中心点在距离矩阵 G 中对应的数据行

输出:各中心点代表的聚类结果簇

步骤:

①Map 函数按列读取各聚类中心点在距离矩阵 G 中对应数据行的数据项,各数据项以键值对 $\langle (c_k, i), \text{sum} \rangle$ 的形式存储,Map 函数每次读入各聚类中心点中第 i 个数据项,划分数据项比较其对应 sum 的大小,将最大 sum 对应的键值 (c_k, i) 改写成键值对 $\langle c_k, i \rangle$ 输出。

②Combine 函数收集所有 Map 过程的输出键值对 $\langle c_k, i \rangle$, 将具有相同键值的数据项合并到一起形成 $\langle \text{键值}, \{\text{记录集合}\} \rangle$, 然后将所有数据项转送给多个 Reduce 过程。

③Reduce 函数直接输出 $\langle \text{键值}, \{\text{记录集合}\} \rangle$ 到分布式文件系统,得到最终的聚类结果。

2.3 算法复杂度分析

设数据集中共有 n 个数据点,每个数据点共有 m 个属性,要将其划分成 k 类。

传统的减法聚类算法的主要操作步骤有:计算邻域半径,时间复杂度为 $O(mn^2)$;计算初始各数据点密度指标,时间复杂度为 $O(mn^2)$;更新各样本密度指标,时间复杂度为 $O(kmn)$, k' (等于 $k-1$) 是更新密度指标迭代的次数;划分各数据点到相应的聚类中心代表簇中,时间复杂度为 $O(kmn)$ 。因此,减法聚类算法总体时间复杂度为 $T = O(mn^2) + O(mn^2) + O(kmn) + O(kmn)$ 。

然而在 Hadoop 分布式处理平台下,存储在文件系统 HDFS 上的文件将会被划分成一些数据子集,所有数据子集被分配到不同的物理节点,在 Map 阶段,一个数据子集由一个 Map 任务来处理。假设 Hadoop 系统中共有 p 个物理节点参与计算,每个节点可以完成 q 个 Map 任务,则本文提出的基于 Hadoop 的改进减法聚类算法的主要时间复杂度包括以下几部分:计算数据点间距离矩阵和邻域半径,时间复杂度为 $O(2mn^2/pq)$;计算初始各数据点密度指标,时间复杂度为 $O(n^2/pq)$;更新各样本密度指标,时间复杂度为 $O(kn/pq)$;划分各数据点到相应的聚类中心代表簇中,时间复杂度为 $O(kn/pq)$ 。因此本文算法的总体时间复杂度为: $T = O(2mn^2/pq) + O(n^2/pq) + O(kn/pq) + O(kn/pq) + M(p)$, $M(p)$ 为 MapReduce 计算模型配置初始化需要的时间。理论上来看,本文提出算法的时间效率相比与传统的减法聚

类算法有很大提高.

3 实验与结果分析

为了检验本文提出算法的性能,对 UCI 数据集 Iris 数据点进行人工处理,生成包含不同数据量的新数据集样本,实验数据如表 1 所示. 每条数据记录点由 10 维数值型数据组成.

表 1 数据集描述

数据集	数据点数	数据集大小/GB
1	972 012	0.039 3
2	5 832 072	0.235 8
3	10 692 144	0.432 3
4	30 132 372	1.19
5	44 713 623	1.76
6	88 453 092	3.49

本文分别在单机环境和分布式环境下验证提出改进算法的性能. 分布式环境采用 1 个主节点,12 个从节点的 Hadoop 平台架构,每个节点的硬件配置如下: Intel(R) Core(TM)2 Duo CPU E7400@ 2.8 GHz,2 GB 内存.

3.1 单机环境下比较实验

本节将对 Hadoop 集群系统中一个数据处理节点和普通单机串行减法聚类算法在处理相同数据过模时所花费的时间进行比较分析,实验数据选用数据集 1 到 6(如表 1 所示),实验结果如图 1 所示.

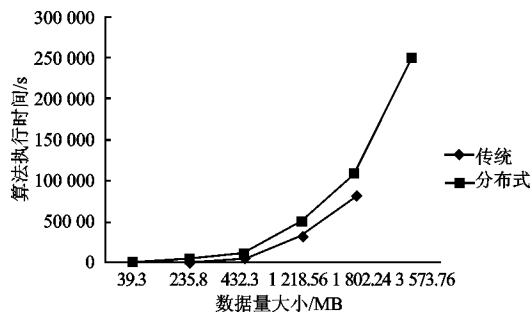


图 1 单机处理比较实验

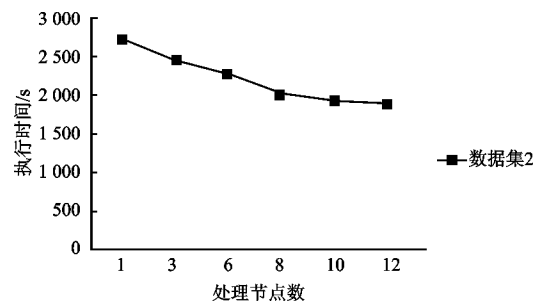
由实验结果可以看出,在数据量较小时,普通串行算法的处理时间要少于 Hadoop 平台下的算法. 这是因为在 Hadoop 平台下,算法的每次执行都需要初始化配置 MapReduce 模型,模型的启动和配置过程需要消耗一定的系统资源,而且分布式平台下,节点间需要通信,这也将消耗一定的系统时间. 因此处理小数量时,算法实际计算消耗占用的时间比例相对较小,造成分布式算法效率不及普通的串行减法聚类算法.

然而,当数据量增大到一定程度时,由于计算机

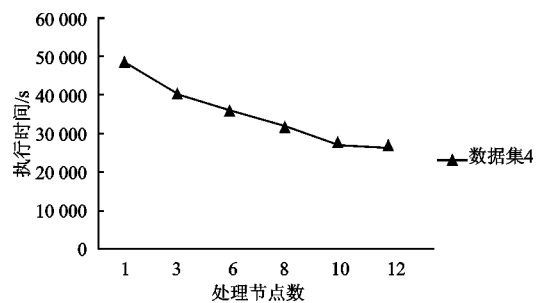
硬件资源的限制,普通串行算法会抛出内存不足的异常(图中没有相应的处理时间点). 相反情况,只有一个处理节点的 Hadoop 集群系统却能很好应对数据量的增加,顺利完成在大数据量下的计算任务. 由此得出,利用 MapReduce 分布式模型实现的减法聚类算法在 Hadoop 平台上表现出了良好的数据处理性能,充分体现了 Hadoop 处理大数据量的能力和优势.

3.2 分布式环境下实验分析

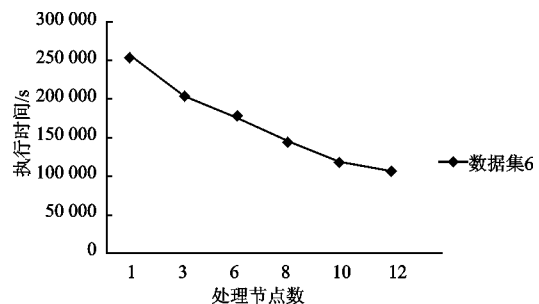
为了测试本文提出算法在 Hadoop 平台下的性能,本节进行如下加速比实验. 实验数据采用数据集 2、4、6(如表 1 所示),分别选取 1、3、6、8、10、12 个数据处理节点在三个数据集上执行本文提出的算法,考察在逐渐增加数据处理节点的情况下,分布式系统完成任务所需要的时间,实验结果如图 2 所示.



(a)数据集 2 上的加速比实验



(b)数据集 4 上的加速比实验



(c)数据集 6 上的加速比实验

图 2 算法加速比实验

从图中可以看出,每个任务运行的时间都随着

节点数量的增加而降低,增加节点可以显著提高系统对同等规模数据的处理能力,说明 MapReduce 模型在处理减法聚类算法时有较好的加速比。然而在数据集 2 的处理节点超过 8 个,数据集 4 和 6 处理节点超过 10 个时,增加节点之后算法的运行时间降低的并不明显,原因在于随着节点增加,算法的通信量急剧增加,通信时间急剧增加。所以对于给定数据量的数据,在分布式环境下不是处理节点越多,处理时间越快,要衡量数据量与节点的处理能力和节点间通信代价的关系,避免浪费分布式资源。

4 结束语

针对传统减法聚类算法不能满足大数据时代数据处理的要求,本文提出一种基于 Hadoop 平台的改进减法聚类算法。算法首先利用矩阵求解思想改进减法聚类中计算数据点间距离过程,并对求解数据点间距离进行 MapReduce 并行化。然后利用得到的数据点间距离矩阵计算邻域半径、初始化数据点密度指标、更新数据点密度指标和划分数据点到相应簇,并对上述四个求解过程进行 MapReduce 并行化。最后在单机和分布式环境下验证本文提出算法的性能。实验结果表明,在大数据量下,将本文算法部署在 Hadoop 集群上能够有效提高计算效率。在大数据时代,采用高效的 MapReduce 计算模型实现传统数据挖掘算法的并行化具有一定的现实意义。通过本文研究发现,减法聚类算法可以描述数据属性的重要性,今后将对减法聚类结果与属性相关程度进行进一步的研究,并利用 MapReduce 模型进行分布式部署。

参考文献:

- [1] Jiawei Han, Micheline Kamber, Jian Pei. 数据挖掘: 概念与技术[M]. 范明, 孟小峰, 译. 北京: 机械工业出版社, 2012.
- [2] Wu X, Zhu X, Wu G Q, et al. Data mining with big data[J]. Knowledge and Data Engineering, IEEE Transactions on, 2014, 26(1): 97-107.

- [3] 陆嘉恒. Hadoop 实战[M]. 北京: 机械工业出版社, 2011.
- [4] 江小平, 李成华, 向文, 等. K-means 聚类算法的 MapReduce 并行化实现[J]. 华中科技大学学报: 自然科学版, 2011, 39(1): 120-124.
- [5] 赵卫中, 马慧芳, 傅燕翔, 等. 基于云计算平台 Hadoop 的并行 k-means 聚类算法设计研究[J]. 计算机科学, 2011, 38(10): 166-168.
- [6] 鲁伟明, 杜晨阳, 魏宝刚, 等. 基于 MapReduce 的分布式近邻传播聚类算法[J]. 计算机研究与发展, 2012, 49(8): 164-174.
- [7] 陈东明, 刘健, 王冬琦, 等. 基于 MapReduce 的分布式网络数据聚类算法[J]. 计算机工程, 2013, 39(7): 76-82.
- [8] Zhong Q, Lin Y, Zou J, et al. Parallel spectral clustering based on MapReduce[J]. ZTE Communications, 2013, 2(11): 30-37.
- [9] 曹泽文, 周姚. 基于 MapReduce 的 JP 算法设计与实现[J]. 计算机工程, 2012, 38(24): 14-16.
- [10] 何国雄. 基于 MapReduce 的图聚类算法的研究与实现[D]. 长沙: 湖南大学, 2012.
- [11] 张磊, 张公让, 张金广. 一种网格化聚类算法的 MapReduce 并行化研究[J]. 计算机技术与发展, 2013, 23(2): 60-64.
- [12] 蒋雯倩, 李欣然, 钱军. 改进 FCM 算法及其在电力负荷坏数据处理的应用[J]. 电力系统及其自动化学报, 2011, 23(5): 1-5.
- [13] 向林泓, 陈芋文, 张昱琳. 基于 Hadoop 平台的高阶矩阵相乘 MapReduce 算法研究[J]. 计算机科学, 2013, 40(06A): 96-98.

作者简介:

原旭 男, (1970-), 硕士, 副教授. 研究方向物联网与云计算.

陈志奎 男, (1968-), 博士, 教授, 博士生导师. 研究方向为物联网与大数据计算. E-mail: zkchen@dlut.edu.cn.

赵亮 男, (1988-), 博士研究生. 研究方向为物联网与大数据计算.

杨德礼 男, (1949-), 教授, 博士生导师. 研究方向为管理决策分析、管理信息与决策支持系统、电子商务以及经济发展.