软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(11):2836−2850 [doi: 10.13328/j.cnki.jos.005343]

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

一种基于密度的分布式聚类方法*

王 岩[1], 彭 涛[1,2], 韩佳育[1], 刘 露[1]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

通讯作者: 刘露, E-mail: liulu12@mails.jlu.edu.cn

摘 要: 聚类是数据挖掘领域中的一种重要的数据分析方法.它根据数据间的相似度,将无标注数据划分为若干聚簇.CSDP 是一种基于密度的聚类算法,当数据量较大或数据维数较高时,聚类的效率相对较低.为了提高聚类算法的效率,提出了一种基于密度的分布式聚类方法 MRCSDP,利用 MapReduce 框架对实验数据进行聚类.该方法定义了独立计算单元和独立计算块的概念.首先,将数据拆分为若干数据块,构建独立计算单元和独立计算块,在集群中分配独立计算块的任务;然后进行分布式计算,得到数据块的局部密度,将局部密度合并得到全局密度,根据全局密度计算中心值,由全局密度和中心值得到每个数据块中候选聚簇中心;最后,从候选聚簇中心选举出最终的聚簇中心.MRCSDP 在充分降低时间复杂度的基础上得到较好的聚类效果.实验结果表明,分布式环境下的聚类方法 MRCSDP 相对于 CSDP 更能快速、有效地处理大规模数据,并使各节点负载均衡.

关键词: 聚类;分布式计算;MapReduce;独立计算单元;独立计算块

中图法分类号: TP181

中文引用格式: 王岩,彭涛,韩佳育,刘露.一种基于密度的分布式聚类方法.软件学报,2017,28(11):2836−2850. http://www.jos.org.cn/1000-9825/5343.htm

英文引用格式: Wang Y, Peng T, Han JY, Liu L. Density-Based distributed clustering method. Ruan Jian Xue Bao/ Journal of Software, 2017,28(11):2836−2850 (in Chinese). http://www.jos.org.cn/1000-9825/5343.htm

Density-Based Distributed Clustering Method

WANG Yan[1], PENG Tao[1,2], HAN Jia-Yu[1], LIU Lu[1]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbol Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China)

Abstract: Clustering is an important method for data analysis in the field of data mining. The function of clustering is to divide unlabeled data divided into several groups according to the data similarity. CSDP is a density-based clustering method. When data size is large or data dimensionality is high, the efficiency of clustering is relatively low. In order to improve the efficiency of clustering algorithm, this paper proposes a density-based distributed clustering method, called MRCSDP, which uses MapReduce to cluster text data. This method introduces the definition of independent calculation unit and independent calculation block. First, data are split into several data blocks which are used to construct independent calculation unit and independent calculation block. The task for each independent calculation block is assigned. Then the distributed calculation is conducted to obtain the local density of the data blocks. The local

* 基金项目: 国家自然科学基金(60903098); 吉林省发改委产业技术研究与开发专项(2015Y055); 吉林省科技厅重点科技攻关项目(20150204040GX); 吉林大学研究生创新基金(2016183)

Foundation item: National Natural Science Foundation of China (60903098); Industry Technology Research and Development Projects of Jilin Province Development and Reform Commission (2015Y055); Key Scientific Research Project of Jilin Province Department of Science (20150204040GX); Graduate Innovation Fund of Jilin University (2016183)

本文由复杂环境下的机器学习研究专刊特约编辑张道强教授推荐.

收稿时间: 2017-04-14; 修改时间: 2016-06-16; 采用时间: 2017-08-23

densities are combined to obtain the global density. The center value is calculated according to the global density. Based on the global density and the center value, the candidate cluster centers of each data block can be obtained. Finally, the global cluster centers are obtained by calculating the density of all candidate cluster centers. MRCSDP can achieve better clustering performance by reducing time complexity. Experimental results show that compared to CSDP, MRCSDP can process large scale data more effectively with load-balancing on each computing nodes.

Key words: clustering; distributed computing; MapReduce; independent calculation unit; independent calculation block

随着网络信息量的不断增长,人们对获取特定领域信息需求的越来越大.聚类作为一种数据分析的重要方法,旨在根据对象间的相近程度将无标注的数据划分为若干聚簇.与分类不同,聚类是一种无监督学习,不需要任何有标注的训练数据.研究者已经提出了很多聚类算法,比如基于距离的聚类(*K*-means[1])、基于密度的聚类(DENCLUE[2])、基于网格的聚类(CLIQUE[3])等.在现实生活中,聚类在很多领域也有着广泛的应用,例如自然语言处理[4]、多文档自动文摘[5]、搜索引擎[6]等.

CSDP(clustering by fast search and find of density peaks)是一种基于密度的聚类算法[7],CSDP 认为,聚类中心具有两个明显的特点:(1) 聚类中心的密度高于其邻居节点的密度;(2) 聚类中心与比其密度更高的节点具有很大的距离.CSDP 根据节点间的密度与节点间的距离得到聚类中心.但是,由于 CSDP 计算节点间密度与节点间距离的计算复杂度较高,所以并不适用于大规模数据的处理.而且,由于 CSDP 中节点间的计算相关性很高,所以传统的分布式处理框架并不适合 CSDP 算法.另外,传统的聚类方法(如 *K*-Means[1])往往需要提前指定聚簇数量,或者需要迭代计算聚簇中心来得到最终的聚簇分布.

为了解决以上问题,本文提出了基于密度的分布式聚类算法 MRCSDP,通过改进 CSDP 算法,使 MRCSDP 对大规模数据进行聚类分析具有更好的效果.在既不需要事先确定聚簇数量也不需要迭代计算聚簇中心的前提下,对文本数据进行聚类.MRCSDP 首先对数据进行分块处理,两个数据块构成一个独立计算单元,并将独立计算单元划分到不同的独立计算块中.一个节点处理一个独立计算块,并得到局部聚类中心.通过重新计算中心值,得到全局聚类中心.最后,根据聚类中心进行聚类.独立计算单元和独立计算块在一定程度上平衡了各个节点的计算任务,均衡负载[8,9],降低了时间复杂度.根据局部聚类中心重新选取全局聚类中心,提高了聚类中心的准确度.我们采用准确率和兰德指数两种度量标准验证了 MRCSDP 的聚类效果.实验结果表明,MRCSDP 算法可以有效地对文本进行聚类,并且在很大程度上降低了时间复杂度.

本文的主要贡献如下:

1) 提出了一种基于密度的分布式聚类方法——MRCSDP,利用 MapReduce 框架分配计算任务,对节点进行聚类;
2) 提出了独立计算单元和独立计算块的概念,在进行独立计算块中的局部密度计算的过程中,降低了通信开销;
3) 通过重新计算局部聚类中心得到全局聚类中心,在提高聚类中心准确率的同时,减少了运行时间;
4) 从多个角度将 MRCSDP 与 CSDP 算法进行对比,实验结果表明,MRCSDP 算法可以在获得较高准确率的同时降低时间复杂度.

1 相关工作

聚类方法在不同领域都起着重要的作用.探索和研究高效的分布式聚类方法在数据挖掘领域也具有重要的研究意义.本节将概述有代表性的基于密度聚类方法和 MapReduce 分布式模型在不同领域的应用.

基于密度的聚类算法有很多,如传统的基于密度的聚类算法包括 DBSCAN[10]、OPTICS[11]、DENCLUE[12]等.Lee 等人[13]提出了一种改进的基于密度的聚类方法,用在线聚类、主题排序的技术和微博中的文本进行实时事件监控.该方法结合了动态的特征权值计算和近邻产生算法处理动态的文本流.为了获得事件实时的时间及空间信息,Lee[14]提出了一种基于密度的聚类方法.通过系统检测到的实时信息,提取并分析其时间及空间特征,利用聚类方法,达到风险控制的目的.Yang 等人[15]提出了一种可扩展的基于密度的聚类方法 SDC.该方法应用

可扩展的聚类方法对社交网络中的评论信息进行分析.对评论中的文本信息进行聚类,可以帮助识别社区用户所属领域并了解文本对应的主题.Yu 等人根据对聚类数据集的噪声属性的研究,提出了基于双亲传播聚类复合框架 AP2CE[16],应用于含有噪声数据的聚类.Yu 等人调查了聚类解决方案的选择组合问题,并提出了混合集群集合框架(HCE)[17]、混合聚类解决方案选择策略(HCSS)[17],用于聚类解决方案选择问题.Yu 等人提出了一种增量式半监督聚类框架(ISSCE)[18],利用随机子空间技术的优势约束传播方法,规范化剪切算法来执行高维数据聚类.Wang 等人提出了一种局部引力模型,设计了局部引力聚类模型和局部代理模型两种聚类算法[19],并通过几个测试用例验证了聚类算法的有效性.Wang 等人提出了一种可以通过统计测试自动检测聚类中心的新聚类算法[20].该算法可以通过测试数据自动识别聚簇中心,并且通过实验验证了相对于 CSDP 算法,它具有更高的鲁棒性.

MapReduce[21]是当前流行的大数据处理框架之一.MapReduce 是对一个任务进行分解与汇总的过程,主要包括 3 个阶段:Map 阶段、Shuffle 阶段和 Reduce 阶段.MapReduce 的框架如图 1 所示.

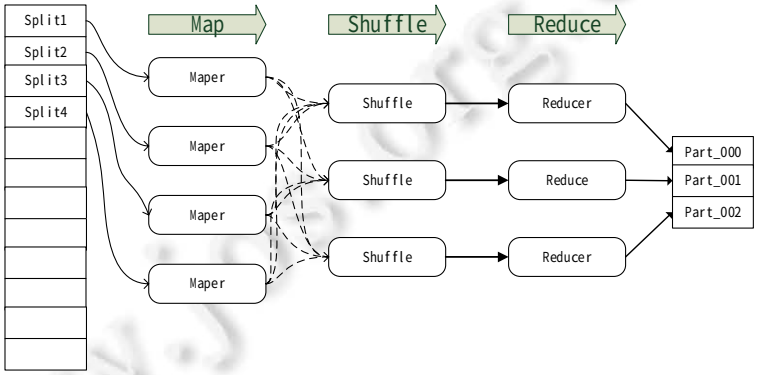


Fig.1 Framework of MapReduce

图 1 MapReduce 框架图

目前,研究者已提出很多基于 MapReduce 的大数据分析算法,如 Lü 等人提出了基于 MapReduce 的 *K*-means 的图像聚类算法[22].该方法基于 MapReduce 框架对传统的 *K*-means 算法进行分布式改进,使 *K*-means 能够运行在 Hadoop 集群上,并行处理海量数据,对海量数据进行聚类分析.实验结果表明,改进后的聚类算法与 *K*-means 算法相比具有更高的效率.Spark 是另一种流式大数据处理框架,Chen 等人基于 Spark 模型提出了一种新型的并行聚类算法 SHAS[23],并通过实验验证了算法可应用于密集型数据的聚类分析.Sinha 等人对原有的 *K*-means 算法使用 Spark 模型进行了分布式改进,提出了 SK-means[24]算法,通过实验可以验证,SK-means 比 *K*-means 更适合大数据处理.鲁伟明等人[25]提出了一种分布式 AP 聚类算法 DisAP,首先将数据点划分为相似的子集,然后并行使用 AP 聚类算法稀疏化各子集,将各子集稀疏化的数据再进行 AP 聚类,得到聚类中心;最后,根据聚类中心进行聚类.实验结果表明,DisAP 聚类算法比 AP 聚类算法更适合处理大规模数据.*K*-means 需要在聚类前给出聚类的数量,虽然 DisAP 不需要给出具体的聚类数量,但 AP 聚类过程是一个循环迭代的过程,对于大数据分析来说,循环迭代会影响数据分析的效率.本文提出了一种基于 MapReduce 的分布式聚类算法,不需要提前给出聚簇的具体数量,也没有循环迭代的过程.实验结果表明,该算法相对于 CSDP 算法对大规模高维数据有更高的效率.

2 问题定义

分布式计算主要考虑任务的分解以及各计算节点的负载是否均衡.本节将介绍 MRCSDP 模型在任务分解时所需要的定义及概念.

定义 **1**(数据块). 为了处理海量数据,数据集 S 被均等分解为多个少量数据集组成的集合$\{A,B,C,D\}$,即$S=A\cup B\cup C\cup D$,且$|A|=|B|=|C|=|D|$.分解后的少量数据集 A,B,C 和 D 被称为数据块.数据块具有两个性质.

1) 对于任意数据对象,该数据对象仅属于 1 个数据块.即 $A=\{a_1,\ldots,a_i,\ldots,a_m\}$,$B=\{b_1,\ldots,b_j,\ldots,b_m\}$,对于 $\forall a_i\in A,b_j\in B$,都有 $a_i\neq b_j$.

2) 数据集 S 为所有数据块的并集,即 $S=A\cup B\cup C\cup D$.

定义 **2**(独立计算单元). 在分布式模型的每个计算节点中,我们将两个数据块间组成的计算任务称为一个独立计算单元.对于每个独立计算单元中的两个数据块,我们称两个数据块为独立计算单元的左数据块 *lblock* 和右数据块 *rblock*.

图 2 描述了基于 MapReduce 的数据分解合并过程.图 2 中第 1 部分表示数据块,第 2 部分中 $A*A$ 和 $A*B$ 表示独立计算单元.

定义 **3**(独立计算块). 独立计算块表示为被划分到同一计算节点中的独立计算单元的集合.如图 2 所示,第 3 部分为由 $A*A$,$A*B$ 以及 $B*B$ 这 3 个独立计算单元构成的独立计算块.

定义 **4**(独立计算密度). 由独立计算块得到的聚类密度称为独立计算密度.在分布式计算模型中,每个独立计算单元会得到数据块的独立计算单元的局部密度值,将独立计算单元的局部密度值合并,构成独立计算块密度.图 2 中第 4 部分表示独立计算块密度,是由第 3 部分的独立计算块得到的.

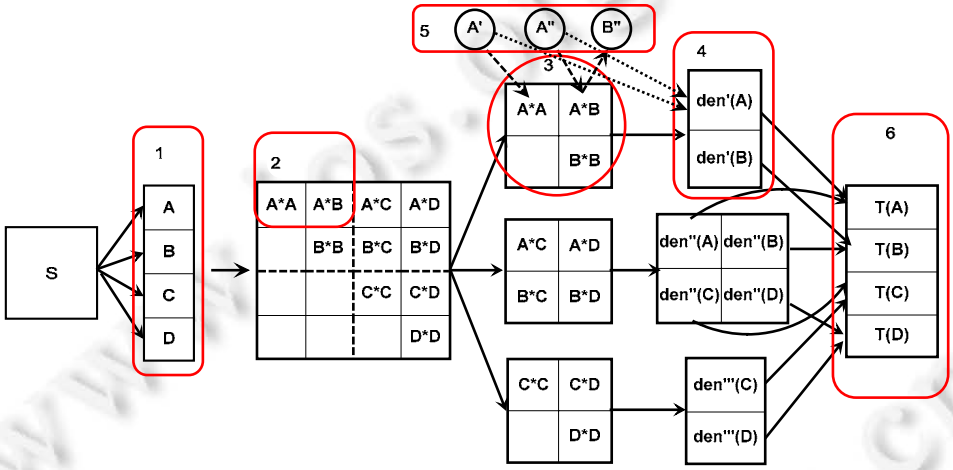


Fig.2 Process of data spliting and merging

图 2 数据分解及合并过程

3 CSDP 聚类方法

本节首先介绍一种基于峰值的密度聚类算法 CSDP,然后通过分析 CSDP 算法的优势与不足,提出了一种基于分布式的密度聚类算法 MRCSDP.MRCSDP 通过结合 CSDP 中的不用迭代的优势,以及 MapReduce 分布式计算框架的分解计算压力的优势,改进 CSDP 算法,使得 MRCSDP 算法适用于大数据处理.

CSDP 是由 Rodriguez 等人于 2014 年发表于 Science 的一种基于密度的算法.该方法首先计算每个节点的密度,进而根据以下两个性质计算聚簇中心:(1) 聚簇中心的密度高于其邻居节点的密度;(2) 若存在某一节点的密度高于某一聚簇中心的密度,则该节点属于另一聚簇.CSDP 根据节点密度和节点间距离进行聚簇划分.节点 i 的密度 ρ_i 的计算方法如下:

$$\rho_i=\sum_{j=1}^{N}\chi(d_{ij},d_c),i\neq j \text{ and } i,j\in[1,N] \tag{1}$$

其中,

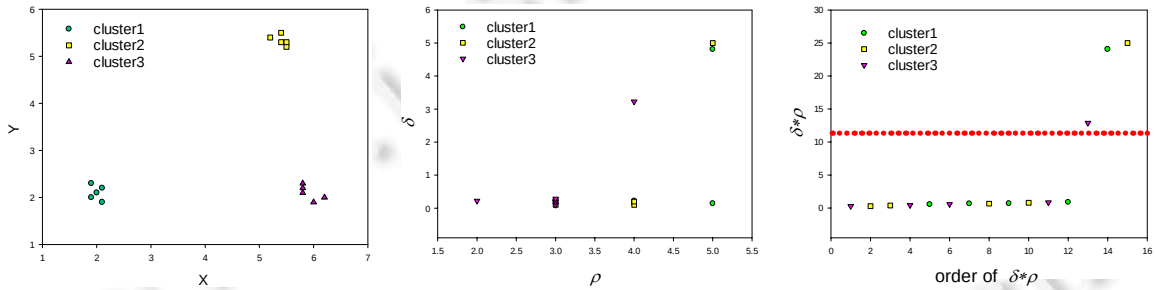
$$\chi(d_{ij},d_c)=\begin{cases}0, & d_{ij}-d_c\leq 0\\ 1, & d_{ij}-d_c>0\end{cases} \tag{2}$$

密度ρ_i表示与节点 i 的距离小于等于截断距离 d_c 的节点总数.d_{ij}是指节点 i 与节点 j 之间的距离,N 表示节点总数,d_c 表示截断距离,对于不同的数据集 d_c,需要选取不同的值,d_c 的选取过程将在实验中给出.在实际应用中,d_c需要根据对测试数据进行多次实验来确定,通过观察聚类准确性,选取使聚类效果最好的 d_c 参数.ρ_i 表示节点 i 与密度高于ρ_i 的节点 j 的最小距离.δ_i 描述了节点成为聚簇中心的可能性,本文将δ_i 称为中心值.δ_i 的计算方法如下:

$$\delta_i=\begin{cases}\min\limits_{\rho_i<\rho_j}(d_{ij}), & \rho_i<\max(\rho)\\ \max(d_{ij}), & \rho_i=\max(\rho)\end{cases}, j\in[1,N] \tag{3}$$

数据集中只有一个密度最大的节点,该节点密度为 $\max(\rho)$.当节点 i 不是密度最大的节点时,δ_i 表示节点 i 与密度比 i 点大的节点距离的最小值.当节点 i 为密度最大点时,δ_i表示距离节点 i 最大的节点与节点 i 之间的距离.

图 3(a)是数据点的聚簇分布图.通过将密度ρ_i和中心值δ_i分别作为点的 Y 和 X 轴的值,将其表示在直角坐标系中,可见,聚类中心与普通节点已分离,如图 3(b)所示.通过聚类中心的可能性权值计算公式,得到每个点为聚类中心的可能性,根据大小进行排序,进而通过 CSDP 算法找到了 3 个聚簇的聚类中心,如图 3(c)所示.我们用余弦相似度计算任意两个节点间的距离.通过截断距离 d_c 得到给定节点的邻居节点集合,根据邻居节点的数量来计算各个节点的ρ_i和δ_i,根据各节点的ρ_i和δ_i得到对应的聚簇中心可能性权值ρ_i*δ_i,根据图 3(c)所示,中间的虚线表示截断值,将聚簇中心可能性权值大于截断值的节点作为聚类中心,最后根据聚类中心进行聚类.



(a) 数据点分布图 (b) ρ vs. δ (c) δ*ρ vs. δ*ρ的排序值

Fig.3 Description of CSDP algorithm

图 3 CSDP 算法描述图

4 基于 MapReduce 的 CSDP 聚类方法 MRCSDP

由于 CSDP 的时间复杂度为 $O(n^2)$(本文第 5 节将给出分析过程),所以对于大量数据,单节点的计算量相对较大,我们将单节点计算分解为多节点的计算结构,很大程度上减少了聚类的运行时间.对于传统的分布式算法来说,主要是将整体任务分解为多个可以并行的独立任务,在分解过程中尽量减少独立任务间的耦合度.但是如果耦合度太低,可能会导致改进的分布式算法与原有算法出现定点漂移现象,即独立任务中的计算信息与全局信息差异比较大,导致计算结果出现问题.如果耦合度过高,那么会导致改进的分布式算法仅仅是传统算法的计算任务的简单分解,整体算法的复杂度并没有改变.本文提出的 MRCSDP 在求取ρ_i阶段时使用的是耦合度较高的分布式方式,在选取聚类中心时采用的耦合度低的分布式方式,这样使得 MRCSDP 不影响聚类效果的前提下具有更低的时间复杂度(本文第 5 节将给出分析过程).本节将介绍利用 MapReduce 将计算分块化的过程,并详细描述基于 MapReduce 的 CSDP 算法 MRCSDP 的聚类过程.

MRCSDP 主要包含 4 个 MapReduce 过程:(1) 用 MapReduce 将数据均等分块,进而保证分布式计算的负载均衡;(2) 用 MapReduce 进行独立计算块划分,得到独立计算单元间的密度与独立计算块间的密度,并保存在 HDFS 上;(3) 将独立计算块中的密度信息进行合并以得到全局密度信息,进而得到局部的聚簇中心;(4) 通过重

新计算局部聚簇中心得到全局聚簇中心,根据聚类中心进行聚类.

4.1 独立计算块密度计算

在并行计算时,运行最慢的节点是整个计算过程的瓶颈,所以在进行分布式计算时,应保证各节点的负载均衡.本文应用的数据集为文本数据,即多个文本文件存储在 HDFS 上.我们将数据集 S 进行均等划分,并对文本文件进行预处理.将文本中的特征表示为⟨*key*,*value*⟩键值对,其中,*value* 表示文本中的特征以及特征在一篇文档中出现的次数,*key* 表示对应的特征在文本文件中的行号.数据集划分的块数为 K,第 1 个 MapReduce 过程将文本对应特征分配新的键值 *key*′,*key*′随机分配,且 $0\leq key'\leq K$.在合并过程中,一个计算节点处理一个 *key*′下所有 *value* 值,并将文本保存在 HDFS 上.

将数据集 S 划分为若干数据块后,对于每个数据块中的数据节点 $f_i\in S$,构造独立计算单元 $f_i*f_j,i,j=1,2,\ldots,K$,且 $i\leq j$,f_i 为该独立计算单元左数据块,f_i 为该独立计算单元右数据块.首先,根据左右数据块编号以及独立计算块边长给独立计算单元分配所属编号.独立计算块编号的计算公式如下:

$$BlockNum=\sum_{m=1}^{i'}S_m+\frac{j-i}{L} \tag{4}$$

其中,i 和 j 为左右数据块的编号,$i'=\frac{i-1}{L}$,S_m 为数据块数量 K 与独立计算块边长 L 构成的递减序列:

$$S_m=\left\{\frac{K}{L},\frac{K}{L}-1,\frac{K}{L}-2,...,0\right\}.$$

将独立计算块编号存储在键值对⟨*key*″,*value*″⟩中,*key*″表示为独立计算块的编号,*value*″为独立计算块 *key*″的一个独立计算单元.在第 2 个 MapReduce 的 Reduce 阶段,根据各个独立计算单元得到对应的局部密度值.独立计算单元的任务是计算两个数据块中各数据节点的局部密度.如 $A*B$ 作为一个独立计算单元,其任务是计算数据块 A 和 B 中任意两点间的距离 d,根据这个距离判断是否增加数据节点的密度.节点密度值将根据节点所在的数据块分开存储,即根据节点所属的数据块 A 和 B 分别保存到对应的密度文件 A''和 B''中.我们将 A''和 B''中存储的节点密度称为独立计算单元密度,即局部密度值 *den*′.

得到局部密度值后,将独立计算单元得到的局部密度 *den*′合并为独立计算块密度 *den*.独立计算块是由多个独立计算单元组成,由各个独立计算单元得到的独立计算单元密度可能属于同一个数据块.一个计算节点处理一个独立计算块,这样可以对两个独立计算单元得到的属于同一个数据块的独立计算单元密度进行合并,在处理过程中减少了通信开销.将独立计算块中属于同一数据块的所有独立计算单元密度合并所得的密度作为该数据块的局部密度,用 den_{block_i} 表示,i 为数据块编号,$i=1,2,\ldots,K$.如图 2 所示,第 3 部分是由 $A*A$,$A*B$ 和 $B*B$ 组成的独立计算块,独立计算单元 $A*B$ 和 $B*B$ 会产生数据块 A 和 B 独立计算单元密度并存储到对应的文件 A',A''和 B''中(如图 2 第 5 部分所示),其中,A'和 A''存储数据块 A 的独立计算单元密度,数据块 A 独立计算块密度 den_A 是由 A'和 A''中的独立计算单元密度合并所得到的.独立计算块密度的算法描述如下.

算法 **1**. 独立计算块密度.

Mapper 1

输入:由若干数据块组成的集合 D,独立计算块边长 L.

输出:⟨*key*″,*value*″⟩键值对.

1. **for** i=1 to K
2. **for** j=i to K
3. 构造独立计算单元 f_i*f_j
4. 根据公式(4)计算独立计算单元编号 *BlockNum*
5. **end for**
6. **end for**
7. *key*″←*BlockNum*

8. $value''\leftarrow f_i*f_j$
9. 输出键值对$\langle key'',value''\rangle$

Reducer 1

输入:键值对$\langle key'',value''\rangle$,截断距离 d_c.

输出:独立计算块的密度.

1. **if** *lblock*=*rblock* **then**
2. **for each** node $i\in lblock$
3. **for each** node $j\in lblock$
4. 计算节点 i 与其他节点 j 的距离 $d_{ij}(i\neq j)$
5. **if** $d_{ij}\leq d_c$ **then**
6. $den_j\leftarrow den_j+1$
7. **end if**
8. **end for**
9. **end for**
10. **end if**
11. **if** *lblock*≠*rblock* **then**
12. **for each** node $i\in lblock$
13. **for each** node $j\in rblock$
14. 计算节点 i 与其他节点 j 的距离 d_{ij}
15. **if** $d_{ij}\leq d_c$ **then**
16. $den_i\leftarrow den_i+1$
17. $den_j\leftarrow den_j+1$
18. **end if**
19. **end for**
20. **end for**
21. **end if**
22. 将独立计算块密度保存在 HDFS 中

4.2 全局密度及局部聚类中心计算

第 3 个 MapReduce 过程描述了如何通过独立计算块密度得到全局密度,进而计算局部聚类中心.第 3 节已经描述了 CSDP 算法需要计算的两个参数:节点密度ρ_i以及节点中心值δ_i.MRCSDP 算法将第 4.1 节中得到的局部密度合并为全局密度ρ_i,并通过对各个数据块进行并行计算得到局部节点中心值δ_i,最后,通过全局密度ρ_i 和局部节点中心值δ_i 得到局部聚类中心.

首先判断局部密度文件属于哪个数据块,并形成相应的$\langle key''',value'''\rangle$对,$key'''$表示数据块的编号,$value'''$表示数据块对应的局部密度文件.将$\langle key''',value'''\rangle$对中的局部密度进行合并,即同一个 key'''值对应的局部密度文件合并为全局密度文件.计算一个数据块中每个节点与数据块内其他节点的中心值δ_i,通过全局密度ρ_i 和局部节点中心值δ_i 来计算聚类中心的可能性权值,将得到的局部聚类中心保存到 HDFS 上.聚类中心的可能性权值计算公式如下:

$$P_i=\rho_i*\delta_i \tag{5}$$

P_i 越大,其对应的节点 i 为聚类中心的可能性就越大.在计算局部聚类中心时,需要设置一个阈值 P_c,当 $P_i\geq P_c$ 时,对应的节点 i 被视为候选聚类中心,保存在 HDFS 上.不同的数据集需要设置不同的值,阈值 P_c 也可以通过进行多次实验并观察聚类准确性来获取,或者由相关方向的专家提供.计算局部聚类中心的算法描述如算法 2 所示.

算法 **2**. 计算局部聚类中心.

Mapper 2

输入:独立计算块的密度 *den*.

输出:⟨*key*‴,*value*‴⟩键值对.

1. 提取独立计算块密度 *den* 对应的块编号 *BlockNum*
2. *key*‴←*BlockNum*
3. *value*‴←*den*

Reducer 2

输入:⟨*key*‴,*value*‴⟩键值对.

输出:数据块候选局部聚类中心集合 *CS*.

1. 将同一 *key*‴下的独立计算块密度合并为全局密度ρ
2. 利用公式(3)计算数据块各个节点的中心值δ
3. 利用公式(5)计算节点的聚类中心的可能性权值 *P*
4. **for each** node $i \in CS$
5. **if** $P_i \geq P_c$ **then**
6. $CS=CS \cup i$
7. **end if**
8. **end for**

4.3 MRCSDP模型分布式聚类过程

在计算全局聚类中心之前,MRCSDP 模型将从不同数据块中得到的候选聚类中心进行合并,得到⟨*key*⁗⁗,*value*⁗⁗⟩键值对.为使所有局部聚类中心分配到同一个计算节点中,*key*⁗⁗为固定值,*value*⁗⁗为局部聚类中心.根据局部聚类中心,得到全局聚类中心.首先提取键值对中的局部聚类中心 *value*⁗⁗,将所有局部聚类中心放到集合 *CF* 中.对于任意节点 $i \in CF$,根据其全局密度为ρ_i,计算节点 *i* 与集合 *CF* 中其他节点的中心值 δ_i',若 δ_i' 小于节点 *i* 原有的中心值δ_i,则更新节点 *i* 的中心值为 δ_i',通过候选聚类中心的全局密度ρ_i与中心值δ_i,根据公式(5)重新计算节点 *i* 成为聚类中心的可能性权值.计算聚类中心的可能性权值的算法描述如算法 3 所示.

算法 **3**. 计算全局聚类中心.

Mapper 3

输入:数据块候选局部聚类中心集合 *CS*,固定的 *KEY* 值.

输出:⟨*key*⁗⁗,*value*⁗⁗⟩.

1. 提取局部聚类中心候选点 $i \in CS$
2. *key*⁗⁗←*KEY*
3. *value*⁗⁗←*i*

Reducer 3

输入:⟨*key*⁗⁗,*value*⁗⁗⟩.

输出:全局聚类中心.

1. 构造所有候选节点集合 *CF*,*CF*←*CF*∪*value*⁗⁗
2. **for each** node $i \in CF$
3. 利用公式(3)计算节点 在候选点集合 *C* 中对应的中心权值 δ_i'.
4. **if** $\delta_i' < \delta_i$ **then**
5. $\delta_i \leftarrow \delta_i'$
6. **end if**
7. 利用公式(5)计算节点 的聚类中心的可能性权值 *P*

8. **if** $P_i \geq P_c$ **then**
9. 构建最终聚类中心集合 F,$F \leftarrow F \cup i$
10. **end if**
11. **end for**

MRCSDP 算法的框架如图 4 所示.该框架主要包括 4 个部分:第 1 部分描述了对数据集中各节点根据其标号进行随机分块的过程;第 2 部分根据数据块的标号构建独立计算单元,根据独立计算单元的左右数据块的标号划分独立计算块,并行计算数据块的局部密度;第 3 部分描述了合并数据块局部密度得到全局密度的过程,并根据全局密度计算数据块的局部聚类中心;第 4 部分通过合并局部聚类中心得到全局聚类中心,根据全局聚类中心进行聚类,得到最终聚类结果.

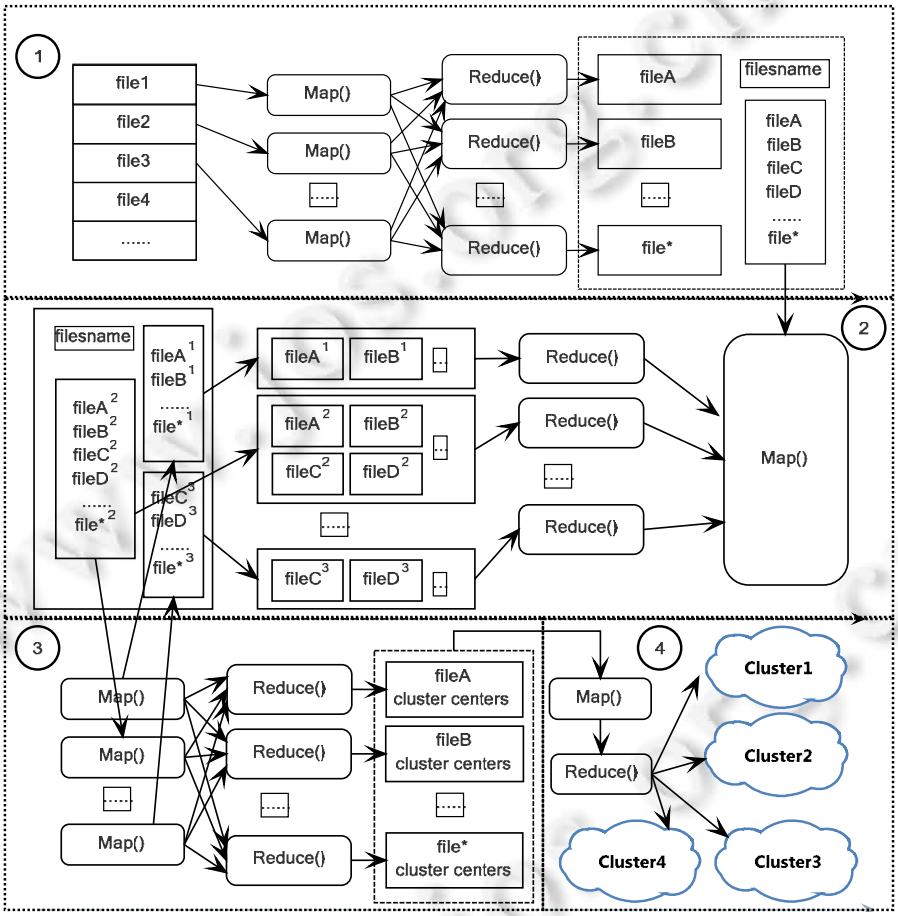


Fig.4 Framework of MRCSDP algorithm

图 4 MRCSDP 算法框架

5 实验与参数分析

本节使用 MapReduce 框架构建了一个基于密度的分布式聚类模型,并在 Reuters-21578 数据集(http://www.daviddlewis.com/resources/testcollections/reuters21578)上验证了我们提出的方法.Reuters-21578 数据集包含从路透社新闻专线收集的 21 578 篇文档,本文从中选取 9 个类别,共 4 700 篇文档用于实验中.第 5.1 节介绍了度量标准来评估我们提出的聚类方法.第 5.2 节给出了实验结果以及参数分析.

5.1 度量标准

本文使用准确率 Cluster Accuracy(CA)[26]和兰德指数 Rand Index(RI)[26]对聚类结果进行评价.CA 是一种比较经典的聚类度量标准,它表示被正确聚类的数据占总数据的百分比.CA 的计算方法如下:

$$CA(\Omega,C)=\frac{1}{N}\sum_{k}\max_{j}|W_k\cap C_j| \tag{6}$$

其中,$\Omega=\{W_1,W_2,W_3,\ldots,W_k\}$表示将数据进行聚类后得到的聚簇集合,其中,$W_k$ 表示第 k 个聚簇集合.$C=\{C_1,C_2,C_3,\ldots,C_j\}$表示文本数据真实的类别集合,其中,$C_j$ 表示第 j 个类别的文本数据.N 表示数据集中数据节点的数量.CA 越大,表示聚类的效果越好.

兰德指数 RI 作为另一种聚类度量标准,描述了聚类算法对无标注数据的区分程度,其计算公式如下:

$$RI=\frac{A+B}{C_n^2} \tag{7}$$

其中,A 表示在Ω与 C 中都为同一类别的元素对数,B 表示在Ω与 C 中都为不同类别的元素对数.比如,真实的聚簇划分如图 5(a)所示,若数据进行聚类后得到的聚簇集合如图 5(b)所示,则表示所有数据都被正确划分到其原本所在聚簇.此时,Ω与 C 中都为同一类别的元素对数为 $A=C_3^2+C_3^2=6$,Ω与 C 中都为不同类别的元素对数为 $B=C_3^1\times C_3^1=9$,此时,兰德指数 $RI=\frac{6+9}{C_6^2}=1$.若数据进行聚类后得到的聚簇集合如图 5(c)所示,此时,Ω与 C 中都为同一类别的元素对数为 $A=C_2^2+C_2^2=2$,Ω与 C 中都为不同类别的元素对数为 $B=C_2^1\times C_2^1+C_1^1\times C_1^1=5$,此时,兰德指数 $RI=\frac{2+5}{C_6^2}=0.467$.兰德指数的取值范围为[0,1],且兰德指数越高,表示聚类效果越好,聚类方法对数据的区分度就越高.

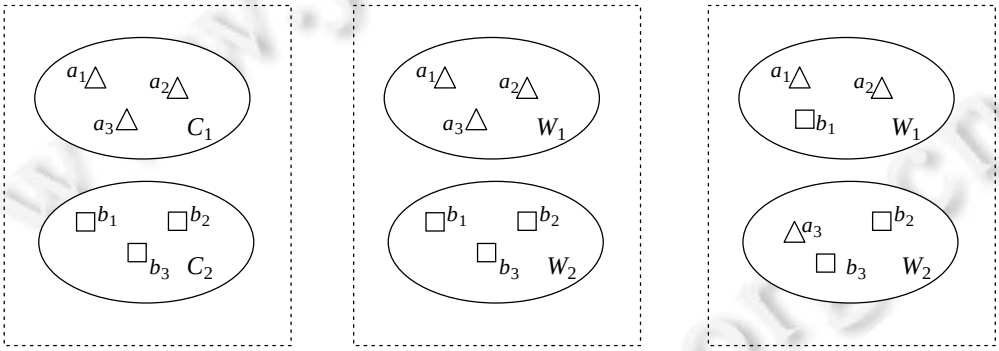


Fig.5 An example for calculating the Rand Index

图 5 一个用于计算兰德指数的示例

5.2 实验结果

本节通过 4 个实验来验证 MRCSDP 模型的可行性和高效性.第 1 个实验通过变化截断距离 d_c 来判断其取何值时准确率和兰德指数达到峰值.截断距离 d_c 直接影响每个节点的密度值,也会间接影响中心值δ.从图 6 可以看出:当截断距离 d_c 较小时,候选聚簇中心较多,不是聚簇中心的节点将被误认为是聚簇中心进而影响最终聚类效果;当截断距离 d_c 较大时,候选聚簇中心较少,使得本应作为聚簇中心的节点没有被正确识别,缺少聚簇中心,进而影响最终聚类效果;当截断距离 d_c 为 0.6 时,准确率和兰德指数达到峰值.因此,接下来的实验中,截断距离 d_c 被设置为 0.6.由实验 1 的结果可以看出,CSDP 的有效性略优于 MRCSDP.其主要原因有两个.

- 第一,在 MRCSDP 中,每个节点的类别标签是根据其最近的局部邻居节点的类别进行标注的.对于那些离聚簇中心较远的节点,其局部最近邻居节点有可能不是全局最近的邻居节点,这样就可能被标注类

别标签的准确性.

- 第二,d_c的选取是 CSDP 的关键,然而 MRCSDP 中局部δ_i对 d_c更加敏感,会间接影响到局部聚簇中心的选取,进而影响最终的聚类结果.δ_i对 d_c敏感度可能是由于局部信息无法完全表示全局信息导致的.

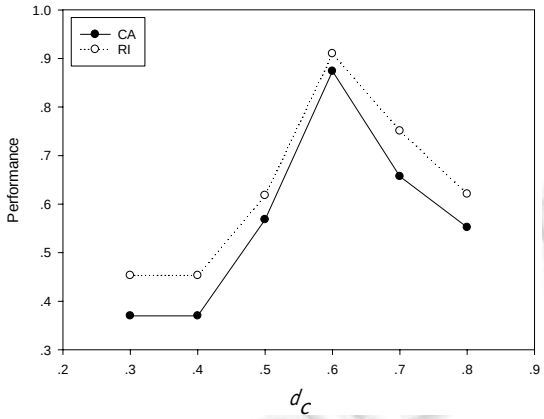


Fig.6 Clustering performance of MRCSDP on Reuters dataset with different d_c

图 6 不同 d_c下对 Reuters 数据集的 MRCSDP 聚类结果

第 2 个实验从不同角度将 MRCSDP 算法和 CSDP 算法进行对比.提取 Reuters-21578 数据集中的 4 700 篇文档数据,MRCSDP 算法将数据集划分为 4 个数据块,此时,MRCSDP 算法和 CSDP 算法的准确率和兰德指数相近,二者都将数据划分为 9 个聚簇,但 MRCSDP 算法的运行时间约为 CSDP 算法运行时间的 1/3.表 1 将 MRCSDP 算法与 CSDP 算法进行对比,通过进行 10 次实验,得到 MRCSDP 算法与 CSDP 算法的 CA 和 RI 的平均值,并给出了方差.

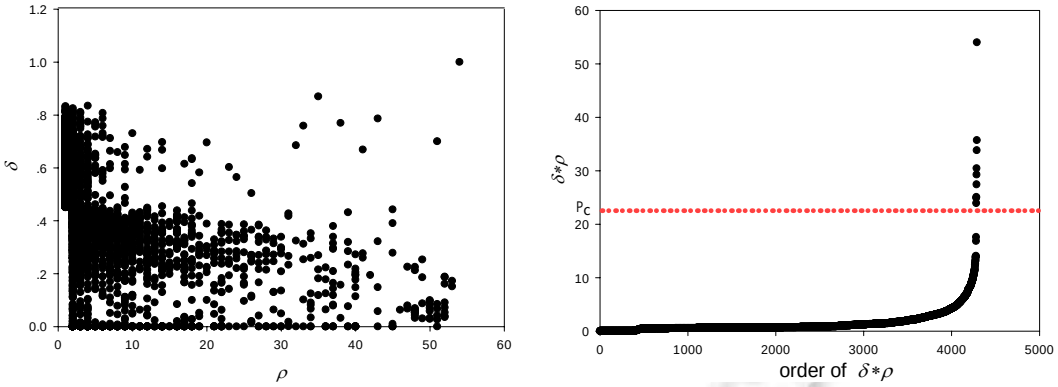
Table 1 Comparison between CSDP and MRCSDP on Reuters DataSet (d_c=0.6)

表 **1** Reuters 数据集上 CSDP 和 MRCSDP 结果对比(d_c=0.6)

	Number of clustering	Clustering time (s)	Number of splitting data	Number of data nodes	CA	RI
CSDP	9	680±18	–	4 700	0.874	0.91
MRCSDP	9	273±9	4	4 700	0.863±0.09	0.89±0.08

第 3 个实验提取 Reuters-21578 数据集中的 4 700 个文档数据,密度中心阈值 P_c为 23,将 MRCSDP 算法的运行效果与 CSDP 算法的运行效果进行对比.MRCSDP 将数据平均分成 4 个数据块,进行密度值、中心值计算.图 7 描述了 CSDP 算法的运行效果图.图 7(a)描述了节点的聚簇密度以及节点对应的中心值.右上角中心值较大的节点即为聚簇中心.图 7(b)为节点中心值的排序图以及其对应的聚簇中心可能性权值.图 8 描述了 MRCSDP 算法其中一个数据块对应的效果图.最终,密度中心候选节点如图 9 所示.可以看出,MRCSDP 可以在分布式的环境下实现 CSDP 算法.二者最终的聚簇中心相符.

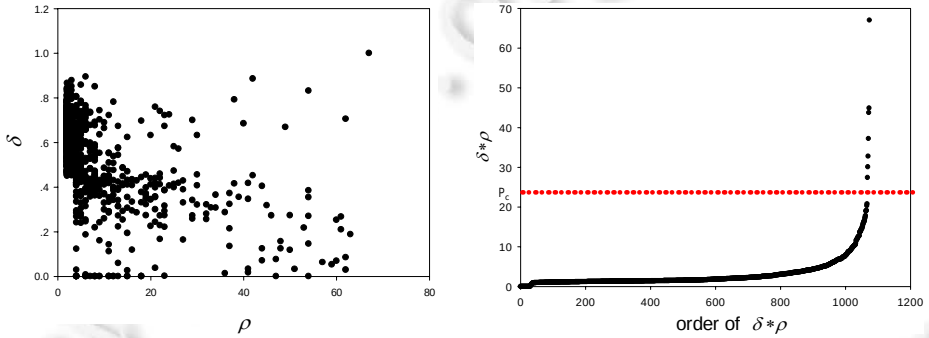
第 4 个实验比较 MRCSDP 算法、CSDP 算法、DisAP 算法[25]和 PKMeans 算法[1]的效率.MRCSDP 模型使用 MapReduce 对 CSDP 进行改进,降低了单个节点的计算密集性,将单节点的计算任务均衡地分配到集群的各个节点上,使聚类算法可以处理大规模高维数据.图 10(a)描述了 4 种算法在不同节点数下对应的运行时间.从运行时间可以看出,MRCSDP 算法的运行时间约为 CSDP 算法运行时间的一半.若数据量增加或主机内存有限,CSDP 算法的效率会更低.由于 DisAP 算法和 PKMeans 算法在聚类过程中需要进行多次迭代,而 MRCSDP 不需要进行迭代,所以 MRCSDP 的执行效率优于这两种分布式算法.从图中可以看出,PKMeans 的执行效率是优于 DisAP 的.由于在本实验中能够提前给出聚簇的数量,在这种情况下,PKMeans 的收敛速度是快于 DisAP 的,但是在某些无法给出聚簇数量的数据集中,PKMeans 是无法处理的.另外,DisAP 对参数敏感,参数的设置很容易影响迭代次数.在设置适当的参数后,由于 DisAP 算法的自身复杂性高,MRCSDP 的执行效率依旧优于 DisAP.



(a) 聚簇密度ρ vs. 聚类中心可能性权值δ (b) δ*ρ vs. δ*ρ的排序值

Fig.7 Performance results of CSDP algorithm

图 7 CSDP 算法执行结果



(a) 聚簇密度ρ vs. 聚类中心的可能性权值δ (b) δ*ρ vs. δ*ρ的排序值

Fig.8 Performance results in block of MRCSDP algorithm

图 8 MRCSDP 模型数据块执行结果

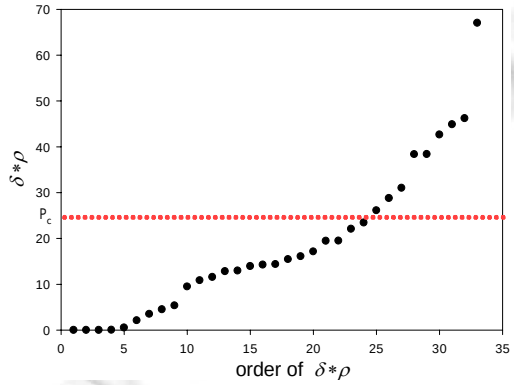
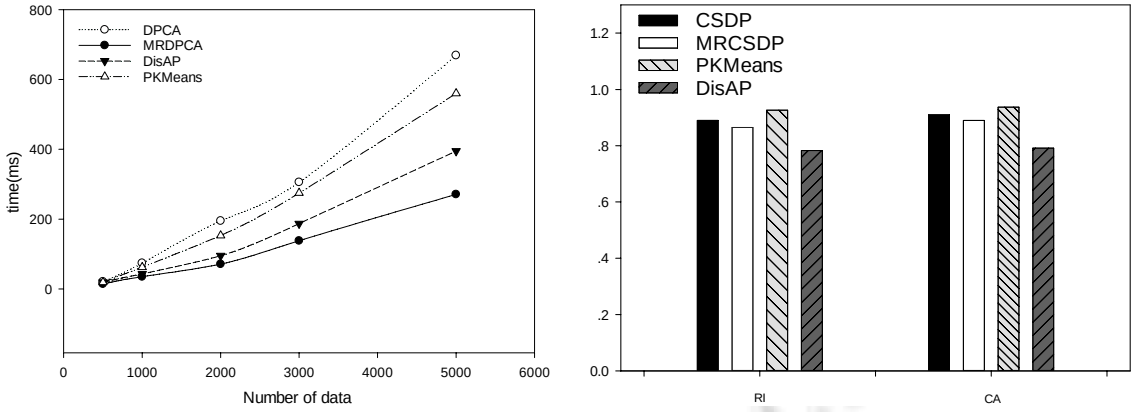


Fig.9 δ*ρ vs. the order of δ*ρ

图 9 δ*ρ vs. δ*ρ的排序值



(a) 4 种聚类算法对于不同数据量的执行时间 (b) 4 种聚类算法 RI 值和 CA 值的比较

Fig.10 Comparision of four clustering algorithms

图 10 4 种聚类算法运行效果比较图

第 5 个实验比较 MRCSDP 算法、CSDP 算法、DisAP 算法和 PKMeans 算法的有效性.图 10(b)描述了 4 种算法在 Reuters-21578 数据集上对应的 RI 和 CA.从实验中可以看出,PKMeans 的有效性高于其他 3 种无监督的聚类算法.由于监督型算法 *K*-Means 的优势,在算法初始化前就拥有准确的类簇数量信息,而且 PKMeans 是一种耦合度比较高的分布式实现方式,分布式处理过程仅仅是将计算任务进行简单的分解,并没有舍弃全局信息,在有效性表现上与 *K*-Means 基本相同,并且高于其他 3 种无监督聚类算法.对于 MRCSDP 算法和 CSDP 算法有效性分析在实验 1 中给出.MRCSDP 的有效性优于 DisAP,主要是因为在 DisAP 聚类过程中使用的是低耦合度的实现,舍弃了一些全局信息.在实际应用中,可以将 MRCSDP 和 PKMeans 进行合并,通过 MRCSDP 确定聚簇的个数,然后通过 PKMeans 进行聚类,这样,在保证有效性的前提下,又能够进行高效的聚类分析.

假设聚类数据节点的数量为 N,集群中的计算节点数量为 M,聚类算法主要包含两个阶段:1) 计算密度 ρ_i 阶段;2) 计算中心值 δ_i 阶段.CSDP 算法计算 ρ_i 阶段的时间复杂度是 $T_1=O(N^2)$,计算 δ_i 阶段的时间复杂度为 $T_1=O(N^2)$,所以得到的总的时间复杂度 T 为 $T=T_1+T_2=O(N^2)+O(N^2)$.在计算密度 ρ_i 阶段,由于是多机器并行,此阶段是简单地将 ρ_i 计算任务分解为多个子任务,由 M 个计算节点并行处理.由于节点间的通信以及数据传输导致的额外代价,此阶段的时间复杂度大于 $O(N^2/M)$.假设当分布式集群节点数量足够多时,节点间的通信以及数据传输导致的额外开销是线性的,集群的计算能力是单节点的 μM 倍,其中,$\mu<1$.我们令 $Q=\mu M$,此阶段的时间复杂度用 $O(N^2/Q)$表示.在计算聚类中心阶段,多块并行所用时间是原有计算时间的 $1/K^2$(K 为数据分解的块数).所以,最后的时间复杂度为

$$T=T_1+T_2=O(N^2/Q)+O((N/K)^2) \tag{8}$$

6 结 论

本文提出了一种基于密度的分布式聚类算法 MRCSDP,将 CSDP 算法采用 MapReduce 框架进行分布式计算,在保证准确率的同时提高了效率,并使各个节点负载均衡.MRCSDP 将数据进行均等划分,提出了独立计算单元和独立计算块的概念,将计算密集型任务并行处理.在得到独立计算单元对应的局部聚簇中心后,对局部聚簇中心的节点重新计算密度和中心值,得到全局聚簇中心,进而通过聚簇中心进行聚类.截断距离作为聚类算法的参数对局部聚簇中心有着重要影响,实验结果表明,MRCSDP 算法可以在分布式集群上取得与 CSDP 算法相近的聚类结果,并且在很大程度上降低了时间复杂度,可以用于处理大规模高维数据.

本文使用余弦相似度来度量两篇文档之间的相似度.为了得到更好的聚类效果,选取更合适的相似度度量标准来计算文档间的相似度,是我们下一步要研究的内容.

References:

[1] Zhao W, Ma H, He Q. Parallel *K*-means clustering based on MapReduce. In: Proc. of the Int'l Conf. on Cloud Computing. Springer-Verlag, 2009. 674−679. [doi: 10.1007/978-3-642-10665-1_71]

[2] He J, Pan W. A DENCLUE based approach to neuro-fuzzy system modeling. In: Proc. of the Int'l Conf. on Advanced Computer Control. IEEE, 2010. 42−46. [doi: 10.1109/ICACC.2010.5487269]

[3] Chen N, Chen A, Zhou LX. An incremental grid density-based clustering algorithm. Journal of Software, 2002,13(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/create_pdf.aspx?file_no=20020101&journal_id=jos

[4] Chowdhury GG. Natural language processing. Annual Review of Information Science and Technology, 2003,37(1):51−89.

[5] Cao Z, Wei F, Dong L, Li SJ, Zhou M. Ranking with recursive neural networks and its application to multi-document summarization. In: Proc. of the AAAI. 2015. 2153−2159.

[6] Silverstein C, Marais H, Henzinger M, Moricz M. Analysis of a very large Web search engine query log. In: Proc. of the Sigir Forum. ACM Press, 1999. 6−12. [doi: 10.1145/331403.331405]

[7] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014,344(6191):1492−1496. [doi: 10.1126/science.1242072]

[8] Owoputi O, O'Connor B, Dyer C, Gimpel K, Schneider N, Smith NA. Improved part-of-speech tagging for online conversational text with word clusters. In: Proc. of the Association for Computational Linguistics. 2013.

[9] Amini A, Wah TY, Saboohi H. On density-based data streams clustering algorithms: A survey. Journal of Computer Science and Technology, 2014,29(1):116−141. [doi: 10.1007/s11390-014-1416-y]

[10] Sander J, Ester M, Kriegel HP, Xu XW. Density-Based clustering in spatial databases: The algorithm gdbscan and its applications. Data Mining and Knowledge Discovery, 1998,2(2):169−194. [doi: 10.1023/A:1009745219419]

[11] Ankerst M, Breunig MM, Kriegel HP, Sander J. OPTICS: Ordering points to identify the clustering structure. Proc. of the ACM SIGMOD Record, 1999,28(2):49−60. [doi: 10.1145/304182.304187]

[12] Hinneburg A, Keim DA. An efficient approach to clustering in large multimedia databases with noise. Proc. of the KDD, 1998,98: 58−65.

[13] Lee CH, Chien TF. Leveraging microblogging big data with a modified density-based clustering approach for event awareness and topic ranking. Journal of Information Science, 2013,39(4):523−543. [doi: 10.1177/0165551513478738]

[14] Lee CH. Mining spatio-temporal information on microblogging streams using a density-based online clustering method. Expert Systems with Applications, 2012,39(10):9623−9641. [doi: 10.1016/j.eswa.2012.02.136]

[15] Yang CC, Ng TD. Analyzing and visualizing Web opinion development and social interactions with density-based clustering. IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans, 2011,41(6):1144−1155. [doi: 10.1109/TSMCA.2011.2113334]

[16] Yu Z, Han G, Li L, Liu J, Zhang J. Adaptive noise immune cluster ensemble using affinity propagation. In: Proc. of the IEEE Int'l Conf. on Data Engineering. IEEE Computer Society, 2016. 1454−1455. [doi: 10.1109/TKDE.2015.2453162]

[17] Yu Z, Li L, Gao Y, You J, Liu J, Wong HS, Han GQ. Hybrid clustering solution selection strategy. Pattern Recognition, 2014,47(10):3362−3375. [doi: 10.1016/ j.patcog.2014.04.005]

[18] Yu Z, Luo P, You J, Wong HS, Leung H, Wu S, Zhang J, Han FQ. Incremental semi-supervised clustering ensemble for high dimensional data clustering. IEEE Trans. on Knowledge and Data Engineering, 2016,28(3):701−714. [doi: 10.1109/TKDE.2015.2499200]

[19] Wang Z, Yu Z, Chen C, You J, Gu T, Wong HS, Zhang J. Clustering by local gravitation. IEEE Trans. on Cybernetics, 2017,99: 1−14. [doi: 10.1109/TCYB. 2017.2695218]

[20] Wang G, Song Q. Automatic clustering via outward statistical testing on density metrics. IEEE Trans. on Knowledge and Data Engineering, 2016,28(8):1971−1985. [doi: 10.1109/TKDE.2016.2535209]

[21] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. In: Proc. of the Conf. on Symp. on Opearting Systems Design and Implementation. Berkeley: USENIX Association, 2004. 137−150.

[22] Lü Z, Hu Y, Zhong H, Wu JP, Li B, Zhou H. Parallel *K*-means clustering of remote sensing images based on MapReduce. In: Proc. of the Web Information Systems and Mining. Berlin, Heidelberg: Springer-Verlag, 2010. 162−170. [doi: 10.1007/978-3-642-16515-3_21]

[23] Jin C, Liu R, Chen Z, Hendrix W, Agrawal A, Choudhary A. A scalable hierarchical clustering algorithm using spark. In: Proc. of the IEEE 1st Int'l Conf. on Big Data Computing Service and Applications. IEEE Computer Society, 2015. 418−426. [doi: 10.1109/BigDataService.2015.67]

[24] Sinha A, Jana PK. A novel *K*-means based clustering algorithm for big data. In: Proc. of the Int'l Conf. on Advances in Computing, Communications and Informatics. IEEE, 2016. 1875−1879. [doi: 10.1109/ICACCI.2016.7732323]

[25] Lu WM, Du CY, Wei BG, Shen CH, Ye ZC. Distributed near neighbor propagation clustering algorithm based on MapReduce. Computer Research and Development, 2012,49(8):1762−1772 (in Chinese with English abstract).

[26] Fahad A, Alshatri N, Tari Z, Alamri A, Khilil I, Zomaya AY, Foufou S, Bouras A. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE Trans. on Emerging Topics in Computing, 2014,2(3):267−279. [doi: 10.1109/TETC.2014.2330519]

附中文参考文献:

[3] 陈宁,陈安,周龙骧.基于密度的增量式网格聚类算法.软件学报,2002,13(1):1−7. http://www.jos.org.cn/ch/reader/create_pdf.aspx?file_no=20020101&journal_id=jos

[25] 鲁伟明,杜晨阳,魏宝刚,沈春辉,叶振超.基于 MapReduce 的分布式近邻传播聚类算法.计算机研究与发展,2012,49(8):1762−1772.

王岩(1992－),男,吉林白城人,硕士生,主要研究领域为数据挖掘,Web 挖掘,机器学习.

韩佳育(1989－),女,博士生,CCF 学生会员,主要研究领域为数据挖掘,社交网络,推荐系统.

彭涛(1977－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为数据挖掘,Web 挖掘,机器学习.

刘露(1989－),女,博士,主要研究领域为数据挖掘,Web 挖掘,机器学习.