SCEA:一种适应高维海量数据的并行聚类集成算法

廖 彬[1],黄静莱[1],王 鑫[1],孙瑞娜[1,2,3],葛晓燕[1],国冰磊[4]
(1. 新疆财经大学统计与数据科学学院,新疆乌鲁木齐 830012; 2. 中国科学院信息工程研究所,北京 100093;
3. 中国科学院大学网络空间安全学院,北京 100093; 4. 新疆大学信息科学与工程学院,新疆乌鲁木齐 830008)

摘 要: 针对传统串行聚类集成算法在处理高维海量数据时效率低下的问题,提出基于Spark的并行聚类集成算法SCEA(Spark based Clustering Ensemble Algorithm).首先,通过主成分分析与成对约束结合的方法对算法输入数据进行预处理,达到数据降维并去除特征相关性的目的;其次,通过调用不同聚类算法获得基聚类成员后,采用三元组方法通过基聚类成员的簇标签构造出相似度矩阵,并调用层次聚类算法得到最终的聚类结果;最后,在调用MLlib中已有聚类算法的基础上,基于Scala对SCEA算法进行了实现.将SCEA与同类算法在多组数据集下进行对比测试,实验结果表明:总体上SCEA不仅较已有算法在准确率方面有所提高,并且通过分析运行时间、加速比以及可扩展性3个性能指标,证明了SCEA在算法性能上的优越性.

关键词: 并行聚类; 数据降维; 聚类集成; Spark聚类; 性能优化
中图分类号: TP301.6 **文献标识码:** A **文章编号:** 0372-2112 (2021)06-1077-11
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20191362

SCEA:A Parallel Clustering Ensemble Algorithm for High-Dimensional Massive Data

LIAO Bin[1],HUANG Jing-lai[1], WANG Xin[1],SUN Rui-na[1,2,3], GE Xiao-yan[1], GUO Bing-lei[4]
(1. *College of Statistics and Data Science,Xinjiang University of Finance and Economics,Urumqi,Xinjiang* 830012,*China*;
2. *Institute of Information Engineering,Chinese Academy of Sciences,Beijing* 100093,*China*;
3. *School of Networks Security,University of Chinese Academy of Sciences,Beijing* 100049,*China*;
4. *School of Information Science and Engineering,Xinjiang University,Urumqi,Xinjiang* 830008,*China*)

Abstract: In order to solve the problem of low efficiency in traditional serial clustering integration algorithm in processing high-dimensional massive data,we propose a parallel clustering integration algorithm named SCEA(Spark based Clustering Ensemble Algorithm) which is based on spark platform. The input data of the SCEA algorithm is preprocessed by the combination of principal component analysis and pairwise constraints,which can reduce the dimension of the data and remove the feature correlation. After obtaining the base clustering results using different clustering algorithms,similarity matrix is constructed by the cluster labels of the base cluster members based on the triple method,and the hierarchical clustering algorithm is used to get the final clustering results. On the basis of calling the existing clustering algorithm in the spark MLlib, the SCEA algorithm is implemented based on Scala language. The SCEA is compared with other similar algorithms in multiple data sets. The experimental results show that SCEA is not only improved in accuracy than existing algorithms,but also proves the superiority of SCEA in algorithm performance by analyzing three performance indexes:running time,speedup ratio and scalability.

Key words: parallel clustering;data dimensionality reduction;clustering integration;spark clustering;performance optimization

收稿日期:2019-12-09;修回日期:2020-09-25;责任编辑:梅志强
基金项目:国家自然科学基金(No.61562078);新疆天山青年人才计划项目(No.2018Q073);新疆维吾尔自治区自然科学基金(No.2016D01B014)

1 引言

据文献[1]统计,2017 年全球数据总量达到 21.6ZB,以每年 40% 左右的增长率计算,预计到 2020 年全球的数据总量将达到 40ZB,如何更好地管理并挖掘这些海量数据,是大数据研究的核心内容.聚类是机器学习中应用最为广泛的算法之一,是根据相似性原理对数据集进行分类的一种算法,常用方法有:K-means[2]、层次聚类[3]、谱聚类[4]等,根据算法基于原理的不同,可分为基于划分、层次、网格、密度、神经网络、图等 6 个主要类别.大量已有研究主要集中在对特定数据类型的聚类算法扩展、最优化聚类数量、不同类别(或聚集空间)数据的相关性、聚类集成等几个方面.而在大数据背景下,传统聚类算法除了在可扩展性上面临巨大挑战外,主要还存在以下两个问题:(1)不同的聚类算法擅长的数据集类型及数据结构(如:类别形状或尺寸)不同,比如:K-means 聚类算法更适用于球形的数据,而单链接层次聚类则擅长于检测连接模式;(2)同一聚类算法在相同数据集下,当初始参数不同时,聚类结果存在较大差异,并且往往不能计算出准确的分类数目.以上两个问题造成用户选择适当的聚类算法并找出合适的参数,面临着巨大的困难.为解决以上两个问题,使聚类结果更加稳定,研究人员提出了聚类集成算法;聚类集成是使用一系列基聚类算法进行学习,通过整合不同参数、不同数据子集上的基聚类结果,从而得到比单一算法更稳定、更优越的集成聚类结果.

虽然聚类集成算法较单一的基聚类算法在聚类效果上有所提升,但较于单一的基聚类算法,由于计算量的急剧增加,计算时间也成倍增加.特别是串行聚类集成算法在面对高维海量数据集时,始终受制于单台服务器计算能力、内存容量以及磁盘速度等因素的限制,已经难以满足大数据时代的速度要求.以谱聚类算法为例,即使 FastESC[5],EulerSC[6],U-SENC[7] 等算法通过近似估算等方法避免构建完整稀疏的相似性矩阵,以此提高单机环境下算法的计算效率,但是其计算时延还是受制于数据总量、数据维度、迭代次数等参数.随着 2003 年 Google 公开 MapReduce[8,9] 并行计算框架以来,已有工作大多针对单个聚类算法进行并行化,基于 MapReduce 计算模型的聚类算法成为提高聚类计算效率的主要方法.而基于 MapReduce 的并行聚类算法,大多在 Map 阶段由集群中的节点独立的运行完整的聚类算法,分别得到各自的聚类中心;在 Reduce 阶段对 Map 阶段得到的聚类中心点进行聚合,由此得出最终的聚类中心点.但是,利用 MapReduce 并行化聚类算法最大的问题,是由于 Map 阶段各任务之间的独立性,使得 Map 阶段对数据的切分打破了数据集内部数据之间的关联性,而这将对聚类的最终结果产生重大的影响.与 Hadoop 平台下以磁盘为核心的 MapReduce 计算模式不同,Spark 是以内存为核心的大数据计算框架,聚类数据集能够持久化到可以线性增长的分布式内存中,数据集内部数据之间的关联性在计算过程中得以保留;另外,通过有向无环图(DAG)对各基聚类算法流程进行调度,能够有效的提升算法执行效率.所以,利用 Spark 提升聚类集成算法的计算性能的同时,能够很好的解决 MapReduce 中出现的数据关联性丢失问题.在此背景下,为了解决已有聚类集成算法不能很好适应高维度高规模数据且计算效率低下的问题,提出基于 Spark 的并行聚类集成算法 SCEA(Spark-based Clustering Ensemble Algorithm),并做了如下几个方面的工作:

(1)首先,结合主成分分析和成对约束两种方法对数据进行预处理,其中主成分分析在不改变数据本身的特性前提下将高维数据转化为低维数据,而成对约束去除了特征之间的相关性,减少了盲目搜索无关数据信息的操作.

(2)其次,调用 Spark 下的基聚类算法对预处理后的数据集进行计算,再根据三元组的方法获得相似度矩阵,并调用层次聚类的方法得到最终的聚类结果.

(3)最后,将本文算法与同类算法在多组数据集下进行对比测试,通过实验测试分析证明了本文算法在准确率及加速比等性能指标上的优越性.

2 相关研究

已有大部分针对聚类集成算法方面的研究都是以进一步提高聚类划分结果质量为目标,很少针对聚类集成算法计算效率方面进行研究.其中,针对单个聚类算法计算效率方面,单机环境下诸如 FastESC,EulerSC,U-SENC 等算法主要通过近似估算等方法避免构建完整稀疏的相似性矩阵,从而达到提高计算效率的目的.而分布式计算平台下的工作主要集中在 MapReduce 与 Spark;在 MapReduce 计算模型下,文献[10]中提出适应于大规模数据集聚类的 MR-EGWO(MapReduce based Enhanced Grey Wolf Optimizer)算法,一种基于 MapReduce 的增强灰狼优化算法,通过引入新的变量对聚类过程进行优化,通过在 UCI 基准数据集上的测试结果证明了 MR-EGWO 算法处理大规模数据集的有效性.Kim 等人在文献[11]中提出了适用于 MapReduce 框架的基于密度的聚类算法 DBCURE-MR(Density-Based Clustering Algorithm for MapReduce),实现了簇查找的并行化操作,并在不同数据集上的测试结果证明了 DBCURE-MR 算法的有效性及可扩展性.Sardar T H 等人在文献[12]中设计了并行计算模型 MapReduce 下的 K-means 算法,在和已有的串行 K-means 算法对比实验后

证明了算法的有效性. 但是,在 MapReduce 平台下由于 Map 阶段各任务之间的独立性,使得 Map 阶段对数据的切分打破了聚类数据集内部数据之间的关联性,从而影响了最终聚类结果的准确性. 而在 Spark 平台下,由于聚类数据集可以存放在线性增长的分布式内存中,使得数据集内部数据之间的关联性得以保留,很好的解决了 MapReduce 平台下数据关联性缺失问题. 文献[13]中采用 Spark 的内存并行计算优势来加速 HBMC(Basin-Hopping Monte Carlo algorithm)算法的计算速度,实验结果表明基于 Spark 优化后的 HBMC 算法具有更快的收敛速度. Hosseini 等人在文献[14]中基于 Spark 实现了基于分布密度的聚类算法,该算法利用模糊加权相关系数作为相似性度量方法,利用 RDD 存储数据解决了 MapReduce 集群下 I/O 负载瓶颈问题,并通过与同类算法的比较证明了算法在精度及效率上的优越性. 王桂兰等人在文献[15]中提出一种基于 Spark 分布式内存计算的模糊 C 均值算法 Spark-FCM,Spark-FCM 对矩阵水平分割实现了数据的分布式存储以减少节点间的数据传输量,实验证明 Spark-FCM 算法具有良好的可扩展性且算法性能与数据量呈线性关系. 文献[16]提出一种在 Spark 下实现的基于禁忌搜索的聚类算法 Tabu,为了提高计算效率,该算法在计算点与点之间距离时使用了图计算框架 GraphX,性能相比 MapReduce 框架下的同类算法提高 10 倍. 文献[17]对基于 Spark 的 DBSCAN 算法的并行化进行了研究,通过合理利用 RDD 和设计 Sample 算子、map 函数、collectAsMap 算子、reduceByKey 算子,实现对寻找核心对象的密度可达数据点的过程的并行化.

在已有研究的基础上,本文提出一种能够适应高维海量数据的并行聚类集成算法 SCEA. 与已有工作不同之处在于:目前针对算法效率优化方面的研究只是针对单个独立的聚类算法,缺乏对聚类集成算法计算效率提升方面的研究;而本文提出的并行聚类集成算法 SCEA 是将并行优化后的单个聚类算法(如:K-means、Gaussian mixture[18]、Power iteration clustering[19]等)整合到聚类集成算法中,并融合主成分分析、成对约束及三元组等方法优化数据的处理过程,并基于 Scala 语言在 Spark 平台下对算法进行实现.

3　SCEA 并行聚类集成算法

3.1　数据预处理

本节通过主成分分析(Principal Component Analysis,PCA)与成对约束(Pairwise Constraints,PC)两种方法对算法输入数据进行预处理,从而达到对数据降维并删除数据特征之间相关性的作用. 首先,PCA 通过线性投影把样本数据从高维空间投影到低维空间,并使得投影后的样本数据能尽可能表示原始数据;其次,成对约束的方法作为先验知识,通过建立正关联约束(must-link)和负关联约束(cannot-link)两个约束条件来表示两个样本数据间的关系.

设算法输入数据为 $\boldsymbol{X}$,并设 $\boldsymbol{X}$ 数据中样本量及特征维度分别为 n 与 m,那么 $\boldsymbol{X}$ 可由式(1)表示:

$$\boldsymbol{X} = (x_{ij})_{n\times m} \tag{1}$$

设在 $\boldsymbol{X}$ 中第 j 个特征维度的均值为 u_j,可由式(2)进行计算:

$$u_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \tag{2}$$

设 $\overline{\boldsymbol{X}}$ 为 $\boldsymbol{X}$ 去均值化后得到的新样本,那么 $\overline{\boldsymbol{X}}$ 可由式(3)表示:

$$\overline{\boldsymbol{X}} = (x_{ij} - u_j)_{n\times m} = \left(x_{ij} - \frac{1}{n}\sum_{i=1}^{n} x_{ij}\right)_{n\times m} \tag{3}$$

可得到任意第 i 个样本的特征向量为:

$$\overline{\boldsymbol{X}_i} = (x_{i1}, x_{i2}, \cdots, x_{im})^{\mathrm{T}} \tag{4}$$

在式(4)样本特征向量的基础上,由式(5)计算数据集的协方差矩阵 $\boldsymbol{C}$(Covariance matrix):

$$\boldsymbol{C} = \begin{bmatrix} \mathrm{cov}(x_1, x_1) & \cdots & \mathrm{cov}(x_1, x_m) \\ \vdots & \ddots & \vdots \\ \mathrm{cov}(x_m, x_1) & \cdots & \mathrm{cov}(x_m, x_m) \end{bmatrix} \tag{5}$$

式(5)中的协方差矩阵为 $m * m$ 的方阵,具有 m 个特征向量,对角线上是特征 x_i 和 x_i 的方差,非对角线上是 x_i 和 x_j 的协方差. 可由式(6)计算其协方差矩阵的特征值 λ 和对应的特征向量 $\boldsymbol{u}$:

$$\boldsymbol{C}\boldsymbol{u} = \lambda\boldsymbol{u} \tag{6}$$

将特征值 λ 从大到小的顺序排序,根据主成分分析选取最大的前 k 个特征值和相对应的特征向量,并进行投影的过程,实现对高维数据的降维过程. 设降维后得到的数据为 $\boldsymbol{Y}$,那么 $\boldsymbol{Y}$ 可由式(7)计算:

$$\boldsymbol{Y} = \boldsymbol{u}^{\mathrm{T}}\overline{\boldsymbol{X}} \tag{7}$$

通过式(7)得到降维后的新数据 $\boldsymbol{Y}$ 后,再利用成对约束对监督信息得到最佳的投影方向,设其约束投影向量为 $\boldsymbol{Q} = \{q_1, q_2, \cdots, q_d\}$. 成对约束是最小化目标函数找到投影向量 $\boldsymbol{Q}$,设 $J(\boldsymbol{Q})$ 函数式(8)为成对约束目标函数,其中,设属于同类的为正关联约束集合,记为:$M \in \{(y_i, y_j)\}$,不属于同类的为负关联集合,记为:$C \in \{(y_i, y_j)\}$.

$$J(\boldsymbol{Q}) = \mathrm{trace}\left(\boldsymbol{Q}^{\mathrm{T}}\left(\boldsymbol{S}_C - \frac{n_M\sum_{(y_i,y_j)\in C}}{n_C\sum_{(y_i,y_j)\in M}}\boldsymbol{S}_M\right)\boldsymbol{Q}\right) \tag{8}$$

式(8)中,其中 $\boldsymbol{S}_M$ 是正关联约束散布矩阵,n_M 表示正关联约束的个数,其表达式如式(9)所示:

$$\boldsymbol{S}_M = \left[\frac{1}{2n_M^2}\sum_{(y_i,y_j)\in M}(y_i - y_j)(y_i - y_j)\right]^{\mathrm{T}} \tag{9}$$

另外,在式(8)中,$\boldsymbol{S}_C$ 是负关联约束散布矩阵,n_C 表示负关联约束的个数,其表达式如式(10)所示:

$$\boldsymbol{S}_C = \left[\frac{1}{2n_C^2}\sum_{(y_i,y_j)\in C}(y_i - y_j)(y_i - y_j)\right]^{\mathrm{T}} \tag{10}$$

而优化目标函数 $J(\boldsymbol{Q})$,可根据成对约束 $\boldsymbol{S}_M - \frac{n_M\sum(y_i,y_j)\in C}{n_C\sum(y_i,y_j)\in M}\boldsymbol{S}_M$ 来求解对应的特征值和特征向量,令 $\boldsymbol{Z}=\{\alpha_1,\alpha_2,\cdots,\alpha_d\}$ 为对 α 从大到小进行排列,当 $\boldsymbol{Q}$ 为 $\alpha>0$ 时,特征值所对应的特征向量,设 $\boldsymbol{Z}$ 为最终数据预处理后得到的数据集,最终 $\boldsymbol{Z}$ 可由式(11)计算得到:

$$\boldsymbol{Z}=\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Y} \tag{11}$$

3.2 基于三元组的聚类集成

聚类集成过程中,如何构造基聚类成员以及设计一致性函数,是两个最关键的核心问题. 其中,构造基聚类成员通常有 4 种方法:(1)对同一聚类算法选取不同的参数或初始值;(2)选择不同的数据集;(3)选择不同的特征子集投影到数据子空间;(4)选择不同的聚类算法. 本文在不同数据集下,选取不同的聚类算法进行对比测试,通过选取不同的聚类方法解决基聚类成员的问题. 第二个关键问题是设计一致性函数将聚类成员进行集成,即如何根据这些由基聚类成员得到的簇标签去构造数据点之间的相似度矩阵,本文中采用文献[20]中提出的三元组的方法获得相似度矩阵. 本文的聚类集成步骤如图 1 所示:首先,对预处理后的数据集 $\boldsymbol{X}$ 生成 r 个基聚类的结果;其次,对生成的聚类结果构造相似度矩阵 $\boldsymbol{S}$,最后利用层次聚类方法计算最终的聚类结果.

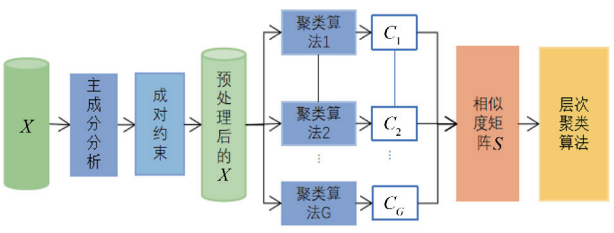


图1 数据集成流程图

设由式(11)预处理后的数据为 $\boldsymbol{Z}=\{z_1,z_2,\cdots,z_n\}$,根据对不同的基聚类算法进行计算,可得到 G 个基聚类结果组成的基聚类为 $\widetilde{\boldsymbol{C}}=\{c_1,c_2,\cdots,c_G\}$,其中每个 $c_i=\{c_i^1,c_i^2,\cdots,c_i^G\}$,而 $G=1,2,\cdots,h$. 把 $\boldsymbol{Z}$ 分成 h 个不相交的簇,由以上可知任意两个不同簇之间没有相同的类,且所有的簇的并集组合成数据 $\boldsymbol{Z}$.

本文通过选取三元组方法构造相似度矩阵,此方法依据存在较多相似性的数据点从而获得数据点之间的矩阵. 三元组的核心思想是:如果两个簇与第三个簇之间存在连接线,则认为这两个簇是存在相似性. 令 A_i 表示分区簇集合,x_i 表示数据点,正方形表示分区簇集合中的簇,c_i^j 是表示通过三元组具有相似性的簇,其中 c_i^j 与 x_i 之间的连线表示两簇连接的边,如图 2 所示.

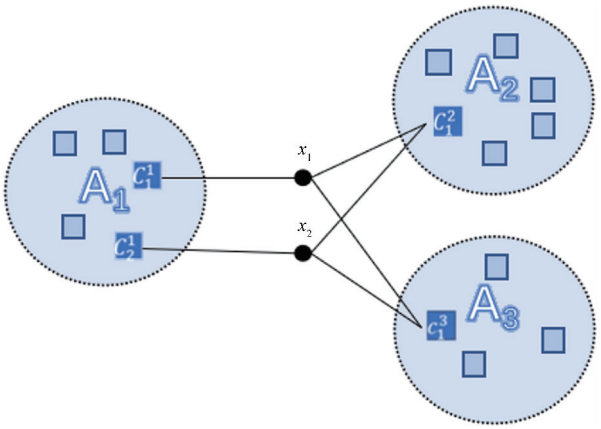


图2 三元组示例

如图 2 所示在分区簇集合 A_1 中,存在着簇 c_1^1 和簇 c_2^1,但这两个簇分别与 x_1 和 x_2 两个不同的数据点连接,从 A_1 簇来看,则 x_1 和 x_2 属于不同的簇. 理论上来说,这两个簇之间不相似,即簇 c_1^1 和簇 c_2^1 不相似;反之,如果存在簇 c_1^1 和簇 c_2^1 相似,则 x_1 和 x_2 之间也存在相似性. 根据上述三元组的思想,簇 c_1^1 和簇 c_2^1 具有两个连接三元组的连接线,且簇 c_1^2 和簇 c_1^3 分别是这两个连接三元组的中心,因此,簇 c_1^1 和簇 c_2^1 是相似的,对于分区簇 A_1 来说,数据点 x_1 和 x_2 也是相似的. 可见该方法不仅扩充了簇之间的相似性,还扩充了数据点之间的相似性信息,有利于具有复杂结构的数据聚类去寻找相似度矩阵.

设连接簇 c_i 和簇 c_j 的边的权重为 P_{ij},P_{ij} 是由这两个簇共同包含的数据点个数得到,即如式(12):

$$P_{ij}=\frac{|X_i\cap X_j|}{|X_i\cup X_j|} \tag{12}$$

其中数据点 x_i 属于簇 C_i 的集合,相邻点 C_{ij}^k 是相邻簇 C_k 与两个簇 C_i 和 C_j 之间连接三元组的最小值:

$$C_{ij}^k=\min(P_{ik},P_{jk}) \tag{13}$$

设 $C_{\max}$ 是两个簇 C_i 和 C_j 之间最大值:

$$C_{\max}=\max(C_i,C_j) \tag{14}$$

C_{ij} 是两个簇 C_i 和 C_j 之间 $q(1<q<\infty)$ 个三元组的总和,C_{ij} 可由式(15)计算:

$$C_{ij}=\sum_{k=1}^{q}C_{ij}^k \tag{15}$$

设簇 C_i 和 C_j 之间相似性为 $\mathrm{sim}(i,j)$,其中,D 为衰减因子($D\in[0,1]$)表示两个不同事物之间相似的置信水平:

$$\mathrm{sim}(i,j)=\frac{C_{ij}}{C_{\max}}*D \tag{16}$$

设数据点 x_i 和 x_j 之间的相似性 $s_m(i,j)$ 为:

$$s_m(i,j)=\begin{cases}1, C(x_i)=C(x_j)\\ \mathrm{sim}(i,j), C(x_i)\neq C(x_j)\end{cases} \tag{17}$$

使正关联矩阵间距离尽可能小,负关联矩阵间的距离尽可能大,得到数据点 x_i 和 x_j 之间的相似矩阵 $\boldsymbol{S}(x_i,x_j)$,相似矩阵是考虑到两个数据点不属于同一簇时的相似性:

$$\boldsymbol{S}(x_i,x_j)=\begin{cases}1, C(x_i)=C(x_j)\\ \dfrac{1}{M}\sum\limits_{m=1}^{M}s_m(x_i,x_j), C(x_i)\neq C(x_j)\end{cases} \tag{18}$$

式(18)中 M 为个体聚类结果的个数,计算式(18)可获得数据点之间的相似度矩阵为 $\boldsymbol{S}$.

3.3　SCEA 算法步骤

在 3.1 节完成数据预处理以及 3.2 节基于三元组进行聚类集成的基础上,SCEA 算法核心步骤如算法 1 所示. 总体上,算法 1 的第 1 ~ 13 行是根据 3.1 节内容进行数据预处理的过程,第 14 行是数据预处理后的数据 $\boldsymbol{Z}$ 分别并行调用 Spark 中不同的聚类算法,将聚类后的结果存储到 $\widetilde{\boldsymbol{C}}$ 中,第 16 行将对 $\widetilde{\boldsymbol{C}}$ 分组处理后,赋值给 G,第 16 ~ 22 行进行基于三元组的聚类集成计算,23 行将 22 行得到的相似矩阵,通过调用 Spark 的层次聚类算法最终得到聚类结果 $\boldsymbol{R}$,并将结果返回.

算法 1　SCEA(Spark-based Clustering Ensemble Algorithm)

Input　Parameter:数据记录输入 $\boldsymbol{X}=(x_{ij})_{n\times m}$

Output　Parameter:$\boldsymbol{R}$

1　$u_j\leftarrow\dfrac{1}{n}\sum\limits_{i=1}^{n}x_{ij}$　//计算主成分的 j 个维度的特征均值

2　$\overline{\boldsymbol{X}}\leftarrow\left(x_{ij}-\dfrac{1}{n}\sum\limits_{i=1}^{n}x_{ij}\right)_{n\times m}$　//更新样本数据

3　$\boldsymbol{C}\leftarrow\begin{bmatrix}\mathrm{cov}(x_1,x_1) & \cdots & \mathrm{cov}(x_1,x_m)\\ \vdots & \ddots & \vdots\\ \mathrm{cov}(x_m,x_1) & \cdots & \mathrm{cov}(x_m,x_m)\end{bmatrix}$　//计算主成分分析的协方差矩阵

4　$\boldsymbol{Cu}\leftarrow\lambda\boldsymbol{u}$　//计算协方差的特征向量及特征值

5　$\boldsymbol{Y}\leftarrow\boldsymbol{u}^{\mathrm{T}}\overline{\boldsymbol{X}}$　//主成分降维后得到的数据集

6　if $\boldsymbol{C}=\varnothing\&\&\boldsymbol{Z}=\boldsymbol{Y}$ or $M\neq\varnothing\&\&\boldsymbol{C}\neq\varnothing$ //$\boldsymbol{C}$ 为空时,$\boldsymbol{Z}=\boldsymbol{Y}$,$M$ 和 $\boldsymbol{C}$ 不为空时

7　　$\boldsymbol{S}_M\leftarrow\left[\dfrac{1}{2n_M^2}\sum\limits_{(y_i,y_j)\in M}(y_i-y_j)(y_i-y_j)\right]^{\mathrm{T}}$ // 计算正关联散布矩阵

8　　$\boldsymbol{S}_C\leftarrow\left[\dfrac{1}{2n_C^2}\sum\limits_{(y_i,y_j)\in C}(y_i-y_j)(y_i-y_j)\right]^{\mathrm{T}}$ //计算负关联散布矩阵

9　　$\{\alpha_1,\alpha_2,\cdots,\alpha_d\}\leftarrow\boldsymbol{S}_M-\dfrac{n_M\sum(y_i,y_j)\in C}{n_C\sum(y_i,y_j)\in M}\boldsymbol{S}_M$//求解特征值和特征向量

10　if $\alpha>0$ //特征向量大于0,则可获得 $\boldsymbol{Q}$ 成对约束后的特征矩阵

11　　$\boldsymbol{Z}\leftarrow\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Y}$//通过成对约束后得到的数据集

12　　end if

13　end if

14　$\widetilde{\boldsymbol{C}}\leftarrow\{\mathrm{SparkAlgorithm1}(\boldsymbol{z}_1^{\mathrm{T}}),\mathrm{SparkAlgorithm2}(\boldsymbol{z}_2^{\mathrm{T}}),\cdots,\mathrm{SparkAlgorithmN}(\boldsymbol{z}_n^{\mathrm{T}})\}$　//对 $\boldsymbol{Z}$ 数据根据不同 Spark 聚类算法进行计算,聚类后归纳到 $\widetilde{\boldsymbol{C}}$ 中

15　$G\leftarrow\mathrm{GroupBy}(\widetilde{\boldsymbol{C}})$//对 $\widetilde{\boldsymbol{C}}$ 分组,得到 G

16　$P_{ij}\leftarrow\dfrac{|X_i\cap X_j|}{|X_i\cup X_j|}$//计算连接簇 i 和簇 j 的边的权重

17　$C_{ij}^k\leftarrow\min(P_{ik},P_{jk})$ //计算簇 i 和簇 j 之间的连接三元组的最小值

18　$C_{\max}\leftarrow\max(C_i,C_j)$ //计算簇 i 和簇 j 之间的最大值

19　$C_{ij}\leftarrow\sum\limits_{k=1}^{d}C_{ij}^k$ //计算簇 i 和簇 j 之间的连接三元组的总和

20　$\mathrm{sim}(i,j)\leftarrow\dfrac{C_{ij}}{C_{\max}}*D$ //计算簇 i 和簇 j 之间的相似性

21　$s_m(i,j)\leftarrow\begin{cases}1, C(x_i)=C(x_j)\\ \mathrm{sim}(i,j), C(x_i)\neq C(x_j)\end{cases}$//数据点 x_i 和 x_j 之间的相似性

22　$S(x_i,x_j)=\begin{cases}1, C(x_i)=C(x_j)\\ \dfrac{1}{M}\sum\limits_{m=1}^{M}s_m(x_i,x_j), C(x_i)\neq C(x_j)\end{cases}$　//得到相似矩阵

23　SparkHierarchicalClustering()//$\boldsymbol{S}$ 矩阵通过层次聚类算法进行聚类,得到最终结果 $\boldsymbol{R}$

24　return $\boldsymbol{R}$

在进行数据预处理过程中,算法第 1 行将主成分的 j 个维度的特征均值赋值给 u_j,第 2 行根据式(3)将样本数据进行更新,第 3、4 行分别计算主成分分析的协方差矩阵以及对应的特征值与特征向量,第 5 行中 $\boldsymbol{Y}$ 为经过主成分分析处理后得到的结果. 第 6 行中,判断条件为如果 M 和 $\boldsymbol{C}$ 为空并且 $\boldsymbol{Z}=\boldsymbol{Y}$,或者 M 和 $\boldsymbol{C}$ 都不为空,第 7、8 行分别计算正关联散步矩阵和负关联散步矩阵,第 9 行在第 7、8 行的基础上求解特征值和特征向量,如果特征向量大于 0,则通过 11 行计算得到成对约束后的结果. 第 15 行将不同 Spark 聚类算法计算返回的 $\widetilde{\boldsymbol{C}}$ 进行分组,赋值给 G,第 16 ~ 20 行分别计算两个簇 C_i 和 C_j 之间的边权重、连接三元组的最小值及最大值、连接三元组的总和、以及两个簇之间的相似性,第 21 行分别计算任意两个数据点 x_i 和 x_j 之间的相似性,第 22 行在 21 行的基础上得到相似性矩阵,最后 23、24 行通过调用 Spark 的层次聚类算法,得到最终的聚类结果并返回.

在 SCEA 算法中,其中计算协方差矩阵的复杂度为 $O(m^2n)$,计算协方差矩阵特征值及特征向量的复杂度为 $O(m^3)$,所以整体上主成分分析阶段的复杂度为 $O(m^2n+m^3)$. 而在分别计算各基聚类算法阶段,其算法复杂度由具体的聚类算法决定,设单个聚类算法的个数为 $\alpha(\alpha\geqslant 2)$,第 $i(i\in[1,\alpha])$ 个聚类算法的复杂度为 ψ_i,如:当基聚类算法为 K-means 时,$\psi=O(knmt)$ 其中,k 为聚类数,t 为迭代次数;可以得到所有的基聚类算法的复杂度为:$O\left(\sum\limits_{i=1}^{\alpha}\psi_i\right)$. 由于簇的个数相对数据

样本大小 n 要小的多,并且三元组计算以及最终集成阶段的层次聚类相对基聚类计算的复杂度几乎可以忽略不计. 所以,SCEA 算法的复杂度约为:

$$O\left(m^2 n + m^3 + \sum_{i=1}^{\alpha} \psi_i\right) \tag{19}$$

由式(19)可以看出,SCEA 算法的时间复杂度主要受到样本量、样本特征数、基聚类个数、迭代次数等因素的影响,而主成分分析主要目的是减小特征维度数 m 值,当样本数 n 值不断增加而基聚类个数及迭代次数固定时,SCEA 算法随着样本数的增长而线性增长,理论上可以利用 Spark 并行计算优势缩短 SCEA 计算时间.

4 实验及结果分析

4.1 实验环境及数据集配置

本节实验环境将 Spark 部署在虚拟集群环境中,虚拟软件采用 VMware Workstation Pro 15,模拟集群环境宿主服务器为 PowerEdge R740,虚拟环境操作系统为 ubuntu 18.04.1,虚拟集群配置及其它详细信息如表 1 所示.

表 1 实验环境描述

项目	描述
物理服务器操作系统	Window Server 2016 64bit
物理服务器 CPU	Xeon Platinum 8180M 2.5GHzMagny-Cours 56 threads
物理服务器内存	192GB = (16G DDR4 RDIMM 2400MHz) * 12
物理服务器磁盘	64T = (WD/HGST 8T 7.2K 256M SATA) * 8
物理服务器 SSD	3 * 960GB SATA SSD
网卡信息	Broadcom BCM5722 1000Mbps
虚拟化软件	VMware Workstation Pro 15
虚拟机操作系统及内核	Linux 4.18.0-15-genericubuntu 18.04.1 gcc 7.3.0
虚拟机配置	2 processor + 2GB memory + 20GB disk
Hadoop 版本	2.7.0
Spark 版本	2.4.3-hadoop2.7
Java 版本	12.0.1
Scala 版本	2.13.0

为了对 SCEA 算法准确度,计算效率等指标进行实验测量,本文从 UCI 数据库和文本数据集中选取了 14 组无监督的数据集,为了排除 YARN 等组件可能对 SCEA 算法准确度测试时产生的影响,Spark 在虚拟集群中选择以 Standalone 模式运行.

实验选取无监督聚类算法所用到的数据集有 14 组:20 Newsgroup、Glass、Iris、Letters、Pendigits、Sonar、Statlog、Wine、Yeast、Bala. Scale、Half ring、CNAE-9、USPS 及 MNIST. 在这 14 组数据集中,文本数据集采用了机器学习中最常用的 20 Newsgroup 和 letters 来做文本聚类实验,其中 20 Newsgroup 中包括了 20 条新闻组,数据量接近 2 万条消息. 其余的数据集来自于 UCI 数据库,这 14 组无监督数据集详细信息如表 2 所示.

表 2 14 组无监督数据集

数据集	类别数	特征维数	数据量
20 Newsgroup	26214	20	18864
Bala. Scale	625	3	4
CNAE-9	1080	9	857
Glass	6	10	214
Half ring	400	2	2
Iris	3	4	150
Letters	26	16	20000
Pendigits	10	16	10992
Sonar	2	60	208
Statlog	7	36	6435
Wine	2	13	178
Yeast	10	8	1484
USPS	10	16 * 16	256 * 1100 * 10
MNIST	10	28 * 28	6w * 784

半监督聚类算法的实验方面,总共选取了 20 Newsgroup、CNAE-9、Letters、Optdigit、Arcene、Sonar 共计 6 组数据集. 其中,文本数据集有 20 Newsgroup 和 Letters 两组,UCI 数据集中包括了 CNAE-9、Optdigit、Arcene、Sonar 四组,数据集详细信息如表 3 所示.

表 3 6 组半监督数据集

数据集	类别数	特征维数	数据量
20 Newsgroup	26214	20	18864
Arcene	900	2	10000
CNAE-9	1080	9	857
Letters	26	16	20000
Optdigit	5620	10	62
Sonar	2	60	208

在进行算法性能测试时,SCEA 算法在 Spark on Yarn 模式下运行,实验数据集使用标准的时间序列 synthetic_control 数据集,由于该数据集初始状态下只有 600 * 60 的数据规模,文件大小为 281KB,原始数据集太小无法满足性能测试的数据量需求. 所以,实验中将 synthetic_control 中的数据进行复制扩展(为了避免复制样本与原样本的完全重复,复制样本的数据为源数据加上[-0.1,0.1]区间的随机值),将原始数据集分别扩大 10 倍、50 倍,100 倍,500 倍,1000 倍及 5000 倍,其

数据量和数据规模如表 4 所示.

表 4　性能测试中的 synthetic_control 数据集

数据集	放大系数	数据集规模	数据文件大小
Synth_control	1	600 * 60	281KB
Synth_control * 10	10	6000 * 60	2817KB
Synth_control * 50	50	30000 * 60	14081KB
Synth_control * 100	100	60000 * 60	28162KB
Synth_control * 500	500	300000 * 60	140808KB
Synth_control * 1000	1000	600000 * 60	281616KB
Synth_control * 5000	5000	3000000 * 60	1408077KB

4.2　算法准确度及 NMI 指标对比

本节分别选取 4.1 节中表 2 与表 3 中所示无监督数据集和半监督的数据集,并在单机环境下对算法进行对比测试,对这两部分的数据集运用不同的算法,进行 5 次独立的运算,最终的聚类结果选择其平均值,再对其准确度进行比较. 其中在无监督聚类中,运行本文的算法 SCEA 算法时,不使用成对约束条件(即:正关联散布矩阵和负关联散布矩阵). 将表 2 中的 20 Newsgroup、Glass、Iris、Letters、Pendigits、Sonar、Statlog、Wine、Yeast、Bala. Scale、Half ring、CNAE-9 数据集,分别对比 EACA[10]、CEKF[11]、WAMM[12]、HCSS[13]、NCAI[14]、WOCE[15]、U-SPNC[7]以及本文的 SCEA 算法最终的聚类准确率,其中,EACA 算法是聚类集成算法,其余算法都是改进后的聚类集成算法,通过运行上述算法与本文的 SCEA 最终得到的聚类结果的准确度比较如表 5 所示.

表 5　无监督聚类结果(准确率%)

DATA SETS	EACA	CEKF	WAMM	HCSS	NCAI	WOCE	U-SPNC	SCEA
20 Newsgroups	25.60	26.15	28.85	30.17	34.99	33.01	35.18	**35.25**
Bala. scale	55.13	56.29	56.34	57.34	56.08	57.68	**59.38**	58.17
CNAE-9	73.26	75.18	75.80	81.10	81.05	78.68	**84.56**	82.42
Half ring	77.49	82.63	85.70	90.79	89.57	87.08	91.68	**92.09**
Iris	73.19	79.37	79.60	84.86	75.59	91.43	91.08	**92.36**
Glass	50.80	43.33	44.14	52.39	50.61	51.20	56.08	**56.46**
Letters	30.56	31.25	31.48	32.27	28.57	29.67	**35.42**	34.15
Pendigits	52.38	56.23	54.30	63.25	60.73	61.55	63.88	**64.18**
Sonar	50.21	53.93	53.35	**62.10**	59.50	54.34	62.05	60.87
Statlog	52.01	56.03	55.33	56.62	56.98	56.07	59.45	**59.63**
Wine	70.47	76.24	80.78	79.72	83.23	70.85	83.58	**84.32**
Yeast	33.68	36.84	34.56	39.56	**42.55**	37.97	42.48	40.06

对于无监督聚类算法运行结果(如表 5 所示)中,粗体表示准确度最高的值. 从表格中可以看出:SCEA、U-SPNC、HCSS 等算法的准确度明显比 EACA 算法准确度要高,其范围在 4.23% ~ 19.17% 之间,这是由于 EACA 算法采用聚类集成的思想,而 CEKF 算法、WAMM 算法、HCSS 算法、NCAI 算法、WOCE 算法是以 EACA 算法作为基础方法,并在此基础上进行改进. 特别的,对于 20 Newsgroups、Half ring、Iris、Glass、Pendigits、Statlog、Wine 7 组数据集,SCEA 相比 EACA 等 7 种算法都取得了最好的准确率,提高的准确度范围在 0.78% ~ 4.07% 之间. 其中 U-SPNC 在 Bala scale、CNAE-9、Letters、Pendigits 四组数据集上取得了最好成绩,而 HCSS 与 NCAI 分别在 Sonar 与 Yeast 上取得了最好成绩. 但是,通过表 5 数据的对比,SCEA 虽然在 Bala scale、CNAE-9、Letters、Sonar、Yeast 5 组数据集上没有取得最优准确率,但是离最好成绩分别仅差 1.21%、2.14%、1.27%、1.23%、2.49%.

将表 3 中的 6 个半监督数据集,采用 5% 的样本作为成对约束信息,例如,5% 的样本数量有 1000 个,则随机选取正关联约束个数为 500 个,负关联约束个数为 500 个,再通过 BGCM[22]算法、SKMS[23]算法、NBF[24]算法和 WOCE[15]算法比较其准确度. 其中 BGCM 算法是基于图形的半监督聚类集成算法,BGCM 算法包无监督和半监督两种算法,SKMS 算法是一种基于核函数的半监督聚类算法,NBF 算法是在半监督聚类集成中寻找局部最优解的算法,WOCE 算法是基于群体智慧的聚

类集成方法,SCEA 算法与其他算法准确度的比较如图 3 所示.

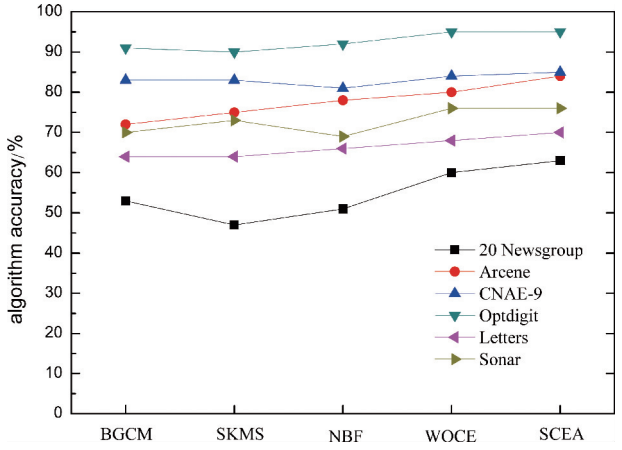


图3 半监督算法准确度对比

对于半监督聚类算法运行结果(如图 3 所示),添加成对约束后的半监督聚类集成算法 SCEA 算法的准确度也得到了一定的提高. 在文本数据集 20 Newsgroup 数据和 Letters 数据中,本文的 SCEA 算法的准确度要高于 BGCM 算法、SKMS 算法、NBF 算法和 WOCE 算法,与 WOCE 算法的准确度相差较少. 在 UCI 数据库中的 Optdigit 数据集和 Arcene 数据集,本文的算法相对其他算法而言准确度得到了提高或持平,与 WOCE 算法较为接近. CNAE-9 数据集不仅运用在无监督中也运用在半监督中,可看出本文的 SCEA 算法无论是在无监督算法还是在半监督算法都比其他算法准确度要高;对于 Sonar 数据集中,也运用在了无监督算法中,可看出本文算法的准确度在无监督算法中未得到提升,在半监督算法中的准确度与 WOCE 算法一致. 总体来说,SCEA 算法准确度相对 BGCM 算法、SKMS 算法、NBF 算法得到了提高,与 WOCE 算法较为接近,有些数据集略高于或持平于 WOCE 算法.

如图 4 所示为在 PenDigits、Letters、USPS、MNIST 四种数据集上,对将 EACA、CEKF、WAMM、HCSS、NCAI、WOCE 以及 U-SPNC7 种算法与 SCEA 进行 NMI% 指标对比的情况. 其中 U-SPNC 算法在 Letters、MNIST 两组数据上取得了最好成绩,NMI% 分别达到了 45.12 及 74.25;而 SCEA 算法在 PenDigits 及 USPS 上取得了最好成绩,NMI% 分别达到了 84.28 及 72.69. 虽然,SCEA 在 Letters、MNIST 两组数据上表现并没有 U-SPNC 算法优异,但是差距并不明显,由于 U-SPNC 采用的基聚类算法为谱聚类,而 SCEA 则是采用的 MLlib 中 6 种最常用的基聚类算法,基聚类的不同造成最终聚类结果具有一定的差异性.

4.3 算法计算效率实验

本节实验中将采用算法运行时间、加速比以及可

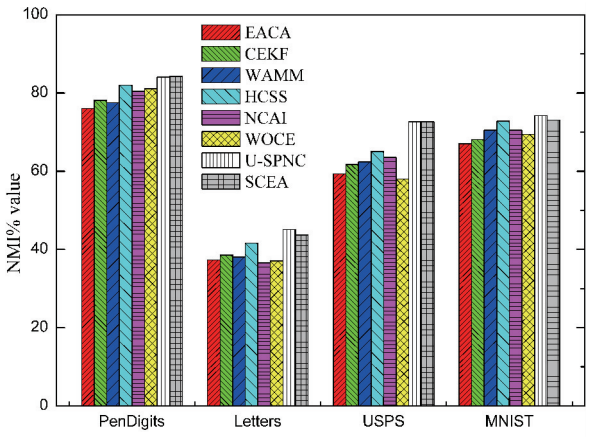


图4 算法NMI%指标对比

扩展性三种指标来对算法性能进行评价,其中算法运行时间是最能直观的反应算法执行效率的指标,执行时间越短,算法效率越高. 性能对比实验中,SCEA 从 Spark 中已有的 K-means、Gaussian mixture、Power iteration clustering、Latent Dirichlet allocation、Bisecting K-means 以及 Streaming K-means 六种算法中随机选择 3 种算法作为基聚类算法,将算法运行 5 次后取其均值作为实验结果.

如图 5 所示为 SCEA 算法在不同数据规模及 Spark 节点数下的运行时间对比,对比 synthetic_control、synthetic_control * 10、synthetic_control * 50 以及 synthetic_control * 100 四组数据可以发现,当 SCEA 计算数据量规模较小时,节点数的增加并不一定能带来运行时间的减少,反而有可能增加运行时间,这是由于算法用于迭代计算上的时间相对较小,并不能发挥 Spark 内存迭代计算的优势;反而当节点数增加时,由于节点间的通信成本增加,从而导致节点数增多,运行时间反而增加的情况发生. 对比 synthetic_control * 500、synthetic_control * 1000 以及 synthetic_control * 5000 三组数据集,可以发现随着 spark 节点数的增加,算法运行时间趋于线性式的减小,说明在计算数据量较大时,SCEA 算法在 spark 中的运行时间才能随节点数的增加而减小.

加速比指标是用来衡量并行环境下系统的性能指标,在保持算法与测试数据不变的条件下,加速比计算公式为:

$$\text{speedup}(n) = T_1/T_n \tag{19}$$

其中,T_1 为单节点条件下运行算法的结束时间,T_n 则表示在 n 个计算节点上并行计算该算法时的结束时间.

如图 6 所示为 SCEA 算法在不同数据规模及 Spark 节点数下的加速比对比,可以观察到 synthetic_control、synthetic_control * 10、synthetic_control * 50 以及 synthetic_control * 100 四组数据集由于数据量较

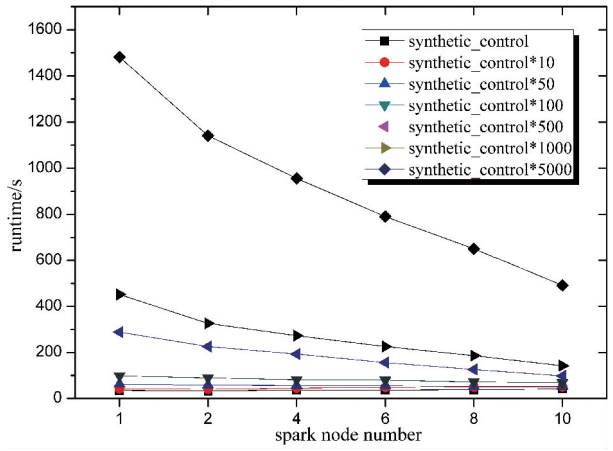


图5　SCEA在不同数据及Spark节点规模下的运行时间对比

小,加速比并没有呈线性增长的趋势. 与 synthetic_control、synthetic_control * 10、synthetic_control * 50 以及 synthetic_control * 100 四组数据集不同,观察 synthetic_control * 500、synthetic_control * 1000 以及 synthetic_control * 5000 三组数据集的加速比,可以发现当数据量不断增大时,可以发现加速比呈近似线性增长的趋势. 由于随着集群节点数的增加,会引入更多额外的节点间的通信开销,并且随着节点数的增加,节点之间的等待时延将会增大,所以线性的加速比是无法实现的. 从图 6 中可以看出,总体上随着 SCEA 处理数据量的增加,加速比能够随着节点数的增加呈近似线性增长的趋势.

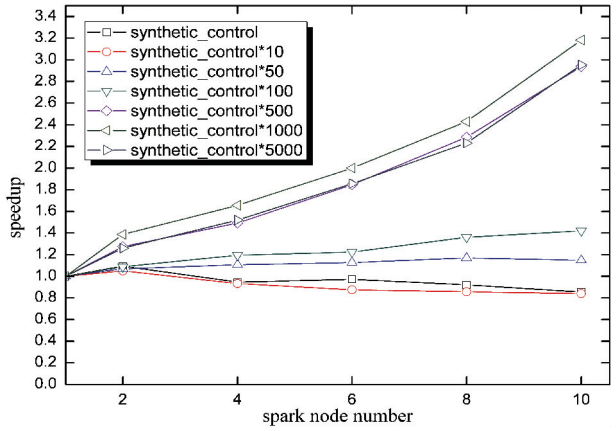


图6　SCEA算法在不同数据规模及Spark节点数下的加速比

如图 7 所示为 SCEA 算法在不同数据规模及 Spark 节点数下的可扩展性对比,可扩展性指标是用来评价算法的性能随着节点数的增加而按比例增长的能力,其计算公式如下:

$$\text{scaleup}(n) = \text{speedup}(n)/n \tag{22}$$

观察图 7 可知,随着节点数量的不断增加,SCEA 算法的可扩展性逐渐减小,但是 synthetic_control * 500、synthetic_control * 1000 以及 synthetic_control * 5000 三组数据集的可扩展性减小的速率远远小于数据量较小的 synthetic_control、synthetic_control * 10、synthetic_control * 50 以及 synthetic_control * 100 四组数据集,说明随着 SCEA 算法处理数据量的增长,算法可扩展性将增加. 其中,synthetic_control * 500、synthetic_control * 1000 以及 synthetic_control * 5000 三组数据集的可扩性非常接近,并且随着节点数的增加,可扩性有上升的趋势,表明 SCEA 算法适合在大规模的 Spark 集群下运行.

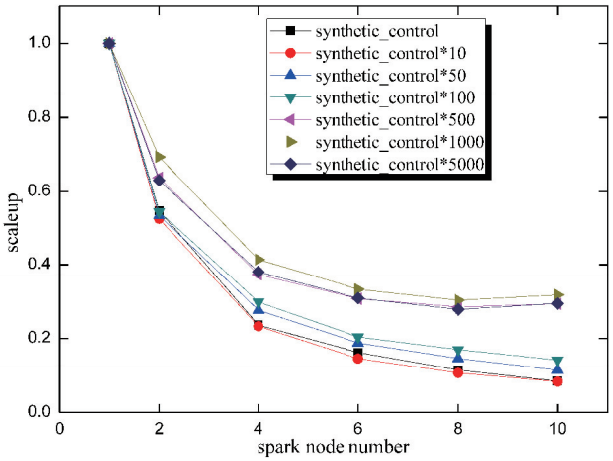


图7　SCEA在不同数据规模及Spark节点数下的可扩展性对比

5　结论及下一步工作

虽然聚类集成算法较单一的基聚类算法在聚类效果上有所提升,但是串行聚类集成算法在面对高维海量数据集时,始终受制于单台服务器计算能力、内存容量以及磁盘速度等因素的限制,已经难以满足大数据时代的速度要求. 为了提高已有聚类集成算法在处理高维度高规模数据时计算效率低下的问题,本文提出一种能够适应高维海量数据的并行聚类集成算法 SCEA. SCEA 算法首先通过主成分分析与成对约束结合的方法对算法输入数据进行预处理,达到数据降维并去除特征相关性的作用;其次,SCEA 通过调用不同聚类算法获得基聚类成员后,采用三元组方法通过基聚类成员的簇标签构造出相似度矩阵,并调用层次聚类算法得到最终的聚类结果;最后,SCEA 与同类算法在多组数据集下进行对比测试结果表明:SCEA 不仅较 EACA,CEKF 等已有算法在准确率等方面有所提高,并且在运行时间、加速比等性能指标上具有一定的优越性. 但是,由于目前 SCEA 仅支持 MLlib 下 K-means、Gaussian mixture、Power iteration clustering、Latent Dirichlet allocation、Bisecting K-means 以及 Streaming K-means 这 6 种基聚类算法,这大大限制了 SCEA

算法对基聚类算法的可选范围. 所以,下一步工作需要研究并适配更多跨平台的基聚类算法(如:Spectral Clustering、DBSCAN 等),以扩展 SCEA 支持的聚类算法选择范围.

参考文献

[1] Gantz J,Chute C,Manfrediz A,et al. The Diverse and Exploding Digital Universe:An Updated Forecast of Worldwide Information Growth Through 2011 [EB/OL]. http://wwww. ifap. ru/library/ book268. pdf,2019-8-25.

[2] Cristina T,Domingo G,José L,Balcázar,et al. Global optimality in k-means clustering[J]. Information Sciences,2018,439(3):9 -94.

[3] Peyman N,Saeid A,Mahmoud N,et al. Hierarchical clustering-task scheduling policy in cluster-based wireless sensor networks[J]. IEEE Transactions on Industrial Informatics,2018,14(5):1876 -1886.

[4] 刘荣胜,彭敏放,肖祥慧. 基于谱聚类集成的变压器在线故障诊断[J]. 电子学报,2017,45(10):2491 -2497.
Liu S R,Peng M F,Xiao X H. The transformer on-line fault diagnosis based on spectral clustering ensemble[J]. Acta Electronica Sinica,2017,45(10):2491 -2497. (in Chinese).

[5] He L,Ray N,Guan Y,et al. Fastlarge-scale spectral clustering via explicit feature mapping[J]. IEEE Transactions on Cybernetics,2018,99(2):1 -14.

[6] Wu J S,Zheng W S,et al. Euler clustering on large-scale dataset[J]. IEEE Transactions on Big Data,2018,4(4),502 -515.

[7] Huang D,Wang C D,Wu J,et al. Ultra-scalable spectral clustering and ensemble clustering[J]. IEEE Transactions on Knowledge & Data Engineering,2020,48(6):1 -15.

[8] Dean J,Ghemawat S. MapReduce:Simplifed data processing on large clusters[A]. Proceedings of the Conference on Operating System Design and Implementation (OSDI)[C]. New York:ACM,2004. 137 -150.

[9] Chen Q,Yao J,Li B,et al. PISCES:Optimizing multi-job application execution in MapReduce[J]. IEEE Transactions on Cloud Computing,2019,7(1):273 -286.

[10] Tripathi A K,Sharma K,Bala M. A novel clustering method using enhanced grey wolf optimizer and MapReduce[J]. Big Data Research,14(9),2018. 93 -100.

[11] Kim Y,Shim K,kim M S,Lee J S. DBCURE-MR:An efficient density-based clustering algorithm for large data using MapReduce[J]. Information Systems,42(6),2014. 15 -35.

[12] Sardar T H,Ansari Z. An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm[J]. Future Computing and Informatics Journal,2018,3(2):200 -209.

[13] Yang Y H,Zhang J F,Wang J B,et al. Spark-based improved Basin-Hopping Monte Carlo algorithm for structural optimization of alloy clusters[J]. Physics Letters A,383(5),2019. 464 -470.

[14] Hosseini B,Kiani K. A big data driven distributed density based hesitant fuzzy clustering using Apache spark with application to gene expression microarray[J]. Engineering Applications of Artificial Intelligence,2019,79(7):100 -113.

[15] 王桂兰,周国亮,萨初日拉,等. Spark 环境下的并行模糊 C 均值聚类算法[J]. 计算机应用,2016,36(2):342 -347.
Wang G L,Zhou G L,Sa C R L,et al. Parallel fuzzy C-means clustering algorithm in Spark[J]. Journal of Computer Applications,2016,36(2):342 -347. (in Chinese).

[16] Lu Y H,Cao B Y,Rego C,Glover F. A tabu search based clustering algorithm and its parallel implementation on Spark[J]. Applied Soft Computing,2018,63(2):97 -109.

[17] 朱子龙. 基于 Spark 的聚类算法实现与应用[D]. 南京:南京邮电大学,2018.
Zhu Z L. Implementation and Application of Clustering Algorithm Based on Spark[D]. Nanjing:Nanjing University of Posts and Telecommunications,2018. (in Chinese).

[18] Aprausheva N N,Sorokin S V. Recognition of unimodality and bimodality of a two-component Gaussian mixture with different variances[J]. Pattern Recognition and Image Analysis,2019,29(2):252 -257.

[19] Zhao Jun,Xu X Y. Distributed power iteration clustering based on GraphX[J]. Journal of Computer Applications,2016,36(10):2710 -2714.

[20] Iamon N,Boongoen T,Garrett S. Refining pairwise similarity matrix for cluster ensemble problem with cluster relations[A]. Discovery Science,11th International Conference,DS 2008[C]. Budapest,Hungary:Springer,2008. 222 -233.

[21] Jin C,Liu R Q,Chen Z Z,et al. A scalable hierarchical clustering algorithm using spark[A]. Proceedings of the 2015 IEEE First International Conference on Big Data Computing Service and Applications[C]. Redwood City,CA,USA:IEEE,2015. 418 -426.

[22] Gao J,Liang F,Fan W,et al. A graph-based consensus maximization approach for combining multiple supervised and unsupervised models[J]. IEEE Transactions on Knowledge and Data Engineering,2013,25(1):15 -28.

[23] Anand S,Mittal S,Tuzel O,et al. Semi-supervised kernel

mean shift clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014, 36(6): 1201 – 1215.

[24] Azimi J, Fern X Z. Active learning of constraints for semi-supervised clustering [J]. IEEE Transactions on Knowledge. Data Engineering, 2014, 26(1): 43 – 54.

作者简介

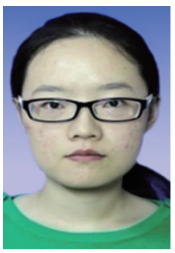
廖　彬　男,1986 年 6 月出生,四川内江人. 2014 年新疆大学计算机应用技术博士毕业. 研究方向为:机器学习,大数据计算技术等.

黄静莱(通信作者)　女,1994 年 5 月出生,四川成都人,新疆财经大学硕士,研究方向为:机器学习,大数据挖掘.
E-mail:1561259618@ qq. com

王　鑫　女,1995 年 11 月出生,重庆奉节人,新疆财经大学统计学硕士,研究方向为:机器学习,大数据计算等.

孙瑞娜　女,1982 年 11 月出生,安徽阜阳人,中国科学院信息工程研究所博士研究生. 研究方向为:数据挖掘、网络安全等.

葛晓燕　女,1980 年 3 月出生,新疆乌鲁木齐人. 2014 年华中科技大学管理科学与工程博士毕业. 研究方向:数据挖掘及大数据计算等.



国冰磊　女,1991 年 6 月出生,湖北襄阳人. 新疆大学信息科学与工程学院博士研究生. 研究方向为:大数据及绿色计算.