计算机科学
COMPUTER SCIENCE

基于节点抽样的分布式二阶段聚类方法

张曼静，何玉林，李旭，黄哲学

引用本文

张曼静，何玉林，李旭，黄哲学. 基于节点抽样的分布式二阶段聚类方法[J]. 计算机科学，2025，52(2): 134-144.

ZHANG Manjing, HE Yulin, LI Xu, HUANG Zhexue. Distributed Two-stage Clustering Method Based on Node Sampling [J]. Computer Science, 2025, 52(2): 134-144.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240800040

基于节点抽样的分布式二阶段聚类方法

张曼静[1] 何玉林[1,2] 李 旭[1] 黄哲学[2]
1 人工智能与数字经济广东省实验室(深圳) 广东 深圳 518107
2 深圳大学计算机与软件学院 广东 深圳 518060
(zhangmanjing@gml.ac.cn)

摘 要 针对大数据聚类中存在的计算资源消耗大、聚类效率低的问题,提出了一种新的基于节点抽样的分布式二阶段聚类方法。该方法首先在各个本地节点对节点上的数据执行局部聚类操作,并基于局部聚类结果,从每个节点中抽取代表性的数据样本,然后将各节点选定的样本数据传输至中央节点。之后,在中央节点上,对合并的样本数据进行进一步的聚类分析,并将样本聚类的结果传回各个本地节点。最后,各本地节点结合自身的局部聚类结果和中央节点的样本聚类结果,完成最终的聚类标签统一。通过以上流程,所提方法实现了对集中式聚类算法的分布式改造,能够快速一致地完成对全局数据的聚类分析。理论分析和数值实验均表明,与传统的全量数据集中式聚类方法相比,二阶段聚类方法有效地结合了并行处理的高效性和集成分析的准确性,在保证聚类质量的前提下能够显著降低计算资源的消耗,是一种可行的大数据聚类分布式解决方案。

关键词:大数据聚类;分布式计算;节点抽样;并行计算;二阶段聚类
中图分类号 TP391.4

Distributed Two-stage Clustering Method Based on Node Sampling

ZHANG Manjing[1],HE Yulin[1,2],LI Xu[1] and HUANG Zhexue[2]
1 Guangdong Laboratory of Artificial Intelligence and Digital Economy(SZ),Shenzhen,Guangdong 518107,China
2 College of Computer Science & Software Engineering,Shenzhen University,Shenzhen,Guangdong 518060,China

Abstract To address the challenges of high computational resource consumption and low clustering efficiency in big data clustering scenarios,researchers have proposed an innovative distributed two-stage clustering method based on node sampling. This method first performs local clustering operations on the data at each local node,and then extracts representative data samples from each node based on the results of the local clustering. The selected sample data from each node is then transmitted to the central node. At the central node,further clustering analysis is conducted on the aggregated sample data,and the results of the sample clustering are transmitted back to the local nodes. Finally,each local node combines its own local clustering results with the sample clustering results to complete the final unification of clustering labels. Through this process,the two-stage clustering method has transformed the traditional centralized clustering algorithms into a more scalable,distributed model,and ensures the consistency of the clustering result to the global dataset. Theoretical analysis and experimental results both indicate that compared with the conventional full-data centralized clustering techniques,the two-stage clustering method offers a framework which effectively integrates the efficiency of parallel processing and the accuracy of integrated analysis. Without sacrificing the accuracy of clustering,it significantly improves clustering efficiency and reduces time costs,which provided a feasible and robust distributed solution tailored for the complexities inherent in big data clustering tasks.

Keywords Big data clustering,Distributed computing,Node sampling,Parallel computing,Two-stage clustering

到稿日期:2024-08-06 返修日期:2024-10-08
基金项目:深圳市科技重大专项项目(202302D074);广东省自然科学基金面上项目(2023A1515011667);深圳市基础研究面上项目(JCYJ20210324093609026);广东省基础与应用基础研究基金粤深联合基金重点项目(2023B1515120020)
This work was supported by the Science and Technology Major Project of Shenzhen(202302D074),Natural Science Foundation of Guangdong Province(2023A1515011667),Basic Research Foundation of Shenzhen(JCYJ20210324093609026) and Guangdong Basic and Applied Basic Research Foundation(2023B1515120020).
通信作者:何玉林(heyulin@gml.ac.cn)

1 引言

随着产业界数据量的急剧增长，数据以前所未有的速度积累，这使得人们对大数据的潜藏价值越来越关注。为了应对大规模的数据处理需求，现有的技术主要采用分布式架构的设计思路，通过并行计算的方式来提升大数据计算的效率。具体来说，分布式并行计算框架将大数据文件切分为多个小的数据块文件，这些小数据块文件被分布式地存储在各个集群节点上。在进行大数据分析时，分布式系统首先在每个节点上对其存储的小数据块进行计算处理。这些初步的计算结果随后被传输到主节点进行融合处理，以产生整体大数据的分析结果。此外，对于那些复杂的分析算法，如迭代式的机器学习算法，整个过程需要交替迭代进行，即上一次计算的输出会成为下一次计算的输入，从而逐步逼近最终解。大数据的增长速度远远超出了可用计算资源的增长速度，面对大数据的分析任务，特别是一些复杂的、迭代式的机器学习算法，工程技术人员时刻面临计算资源和数据扩展性方面的挑战。

聚类是机器学习中一种重要的无监督学习范式，其主要目的是根据数据点之间的相似性，将数据集中的数据分配到不同的簇中，旨在使同一簇内的数据相似性较大而不同簇间的数据相似性较小。聚类算法可以帮助揭示数据中的内在结构，从而为进一步的数据分析和决策提供支持。聚类算法的种类繁多，每种算法都有其独特的适用场景和优势。常见的聚类算法包括：基于划分的聚类（如 K-means 算法[1-2]）、基于层次的聚类（如 HAC 算法[3-4] 和 BIRCH 算法[5-6]）、基于密度的聚类（如 DBSCAN 算法[7-8] 和 OPTICS 算法[9]）、基于网格的聚类（如 CLIQUE 算法[10] 和 STING 算法[11]）、基于模型的聚类（如 EM 算法[12-13]）以及基于神经网络的聚类[14-15] 等。这些聚类算法各有优势和局限，合适算法的选择通常取决于数据的特性和分析的具体需求。

然而，数据量的指数级增加给聚类带来了新的挑战。大多数聚类算法都是迭代型算法，每一轮的计算都依赖于前一轮的结果，这对内存需求和数据扩展性造成了双重压力。因此，传统聚类算法在处理大规模数据集时往往效果不佳，主要是由于这些算法设计的初衷并未考虑大数据的应用场景，当数据的持续增长超出单机处理能力的极限时，算法将无法执行。这一局限凸显了在多台机器上并行处理聚类问题的急切需求，以及对分布式聚类框架进行深入探索和研究的必要性。随着大数据技术和分布式计算框架的快速发展，越来越多的研究和实践开始聚焦于开发和优化分布式聚类算法，使其能够在多台计算机的集群环境中运行，实现对大数据的高效处理和分析[16]。

在分布式聚类算法的具体实施中，每个集群节点首先独立完成对其数据子集的局部聚类任务，然后将这些初步的聚类结果传送到中心节点。中心节点负责对这些结果进行聚合，通过合并或进一步的分析处理来形成最终的聚类输出。这种集中式结果合并步骤可以显著减少网络中的实时数据交换的次数，降低通信成本，提升整体的计算效率。由于各计算节点相互独立地执行计算任务，因此这种架构能够极大地节省计算资源，加快处理速度，使得算法更适合处理大规模数据集。

然而，分布式聚类算法也面临着一些挑战，主要表现在以下几个方面。

1）计算开销大。对于高迭代的聚类算法，每次迭代都需要进行 I/O 操作，这增加了计算开销和时间延迟。尤其在基于 MapReduce 的分布式聚类方法中，模型要求每次迭代都要对数据进行读取和写入操作，这会增加计算开销。每次迭代的数据传输和存储都会引入额外的开销，导致整体计算效率降低。例如，在处理大规模数据集时，频繁的磁盘读写操作会导致系统 I/O 瓶颈，降低算法的执行效率。

2）通信成本高。各节点之间需要频繁地传输中间结果，尤其是在聚类中心更新阶段。这种频繁的网络通信会占用大量网络带宽，增加系统的负载和延迟。在基于 Spark 的分布式聚类方法中，节点间的数据传输和同步操作较多，特别是在高维数据集和大规模数据集的处理过程中，这种问题更加突出。通信成本的增加不仅会影响算法的执行效率，还会造成系统计算资源的浪费。

3）收敛速度慢。一些传统的聚类算法（如 K-means）在高维数据和大数据集上的收敛速度较慢，需要多次迭代才能达到收敛条件。在每次迭代过程中都遍历整个数据集，这进一步降低了算法整体的执行效率。高维数据的距离计算和聚类中心的更新需要大量的计算资源，导致每次迭代的时间成本较高。此外，数据集的规模越大，算法的收敛速度越慢，这严重影响了聚类分析的及时性和有效性。

4）难以形成通用的分布式聚类框架。分布式环境下，聚类算法的多样性和数据的多变性使得难以形成一个通用的、适用于所有场景的分布式聚类框架。依赖于中心点的聚类算法（如 K-means）通常假设数据分布具有某种特定的形状（如球形），但现实世界的数据分布往往更加复杂多样，这在跨节点数据分布不均匀的情况下尤为明显。此外，其他的如基于密度或基于层次的聚类算法，在分布式环境中实现时合并流程相对复杂，涉及到跨节点边界点的确定或复杂的结构合并，增加了实施难度。因此，很难设计出一个能够适应所有不同类型的数据和应用场景的通用分布式聚类框架。

为了应对分布式聚类中遇到的挑战，本文提出了一种基于节点抽样的分布式二阶段聚类方法（以下简称二阶段聚类）。该方法通过节点抽样和二阶段聚类的结果集成，有效地融合了并行处理的高效性和集成分析的准确性；同时无需对局部聚类算法的选取进行限制，可以对局部聚类算法进行分布式改良，使其适用于大数据聚类任务，提升了聚类算法的扩展性和适用性。这种策略保证了在处理大规模数据集时，聚类算法也能保持较低的资源消耗和较高的处理效率。

本文第 2 章将回顾聚类算法的相关研究；第 3 章将介绍二阶段聚类方法的主要概念与流程；第 4 章将给出基于节点抽样的二阶段聚类方法的具体实现，并对其时间复杂度进行分析；第 5 章将通过实验来证实新方法的有效性与准确性；第 6 章进行总结并展望未来的研究方向。

2 相关工作

聚类的目的是将数据集分割成多个簇，使得每个簇内部

数据点的相似度尽可能地高，而簇与簇之间数据点的相似度尽可能地低。聚类分析有助于从大量无标签数据中提取有意义的信息，以便对数据进行更深入的分析和理解，支持异常检测[17-18]、自然语言处理[19-20]、信息检索[21-22]及人脸识别[23-24]等众多实际应用场景。聚类过程主要包括数据清洗、特征工程、相似度计算、聚类、聚类结果评估等关键步骤。

尽管当前对聚类算法的研究取得了较大的进展，各种针对中小规模数据聚类任务的算法相继问世，但在大数据场景下，聚类算法仍面临挑战，包括如何高效处理及应对大规模的数据集、高维数据以及数据流中的动态变化。大数据聚类算法需要在保证聚类质量的同时，确保算法的执行效率和扩展性。分布式聚类算法提供了一种面向大规模数据集聚类问题的解决方案，非常适用于数据分布在多个计算节点上的情况。

不存在普适的聚类算法可以模型化各类型数据所呈现出来的多样分布结构[25]。因此，根据数据类型结构及应用场景的不同，存在多种聚类算法，包括基于划分的聚类、基于层次的聚类、基于密度的聚类、基于网格的聚类、基于模型的聚类和基于神经网络的聚类等等[26-27]。

基于划分的聚类算法要求在聚类开始前预先确定聚类的数量或聚类中心，通过不断的迭代计算来逐步优化目标函数的误差值，直到误差值趋于稳定或满足一定的收敛条件，从而确定最终的聚类结果。基于划分的聚类的优点包括收敛速度快和易于实现规模扩展，主要缺点是需要预先设定聚类数目。此外，初始聚类中心的选择、存在的噪声数据以及聚类数目的设置均会对最终聚类结果产生显著影响，这些因素的不确定性增加了聚类分析的复杂性。这一类聚类的代表算法包括K-means聚类算法[1-2]、FCM(Fuzzy C-means)算法[28]和K-modes算法[29]等。以K-means为代表的存在聚类中心点的算法是一类广泛应用于分布式系统的局部聚类算法，其分布式版本通常通过MapReduce框架实现，Map阶段处理数据分块的聚类，Reduce阶段则通过合并不同数据块的聚类中心点来合并聚类结果[30-32]。然而，存在聚类中心点的算法在分布式环境下面临一些限制。

1)对数据分布的限制。传统的以中心点为基础的聚类算法通常假设簇是球形的。如果数据簇本身不是接近球形，或簇大小和密度差异显著，则单纯使用中心点的聚类效果往往不能满足实际需要。

2)对聚类效果的限制。依赖中心点的局部聚类算法需要在每个节点生成代表性的中心点，然后在全局合并这些中心点。如果中心点的选择不具代表性或存在误差，则全局聚类的结果可能会大打折扣。

3)对异常点的限制。中心点方法通常对噪声和离群点较为敏感。例如，在K-means中，离群点可能极大地影响中心点的计算，从而影响整个聚类的质量。

对不依赖中心点的聚类算法难以形成一套通用的分布式计算框架，例如层次聚类算法、基于密度的聚类和基于模型的聚类等。层次聚类算法也称为树聚类算法，是一种通过形成数据的层次结构序列来解决聚类问题的方法。这种算法可以通过自底向上的聚合方式或自顶向下的分裂方式来实现。层次聚类算法的优势在于无需预设聚类数目，并且能够清晰地展示不同簇之间的层次关系。然而，层次聚类具有不可回溯的特点，限制了其灵活性；同时，计算复杂度较高，导致了高昂的计算成本，使其在处理大规模数据集或需要实时反馈的应用场景中性能受损。层次聚类算法也可以在分布式环境中实现。通常，每个节点执行局部的层次聚类，然后将结果合并成代表全局的层次结构。代表性的层次聚类算法，如BIRCH聚类算法[33-34]，通过一个被称为聚类特征树的数据结构来逐步构建聚类。在分布式环境中，每个节点可以构建自己的聚类特征树，然后通过合并这些树来形成最终的聚类。对这些算法的结果进行合并通常需要一些较为复杂的定制化计算，例如BIRCH聚类算法合并多个聚类特征树需要考虑如何保持聚类结构的一致性和完整性，这涉及到复杂的合并逻辑。

基于密度的聚类算法也是一类常用的不依赖中心点的聚类算法，其核心思想是利用数据点之间的局部密度，将高密度区域内的点聚集在一起，形成簇。这使得基于密度的聚类方法可以处理具有复杂形状和大小的簇，同时对噪声和异常值具有较好的鲁棒性，代表性的算法有DBSCAN[35]，OPTICS[36]和DENCLUE[37]等。对于基于密度的聚类算法，通常来说，不存在聚类中心点，一般需要计算局部聚类的类边界点，通过合并边界点或重叠区域来集成局部聚类结果。然而，对这些算法结果的合并通常需要一些较为复杂的定制化计算[38-39]，例如PDSDBSCAN[40]合并局部聚类的边界点时需要精确识别和处理边界上的数据点，这需要额外的交互和同步，以确保整个数据集的连贯性和一致性。

基于神经网络的聚类算法(也称为深度聚类算法)运用神经网络架构来提取数据的深层次特征表示，进而执行聚类分析。与传统的聚类算法相比，这类算法的优势在于其能够直接整合深度学习模型的鲁棒性，专注于挖掘对聚类任务至关重要的特征[15,41]。基于神经网络的聚类算法在处理复杂和高维数据方面表现出色，尤其是在图像、语音和文本数据的聚类任务中[42]。然而，如何有效地将深度学习模型拆分成多个部分并协调它们之间的通信和协作是一个具有挑战性的问题，仍有待解决。

为了更好地整合分布式环境中聚类算法的特性并应对面临的挑战，设计了一种基于节点抽样的分布式大数据二阶段聚类方法。首先，在各集群节点独立执行对大数据子集的聚类，以获得初步的局部聚类结果。然后，在每个节点上抽取出代表性的数据子集，并将这些子集汇总成一个能反映整体大数据集特性的样本集。接着，对该样本集进行进一步的聚类分析。最后，利用以上步骤的聚类结果，集成各节点的局部聚类信息，实现对整体大数据的综合聚类结果输出。

本文提出的二阶段聚类方法不受特定聚类算法的限制，其允许每个节点根据实际数据特性选择最适合的局部聚类算法。该方法通过在节点上选取合适的数据样本代替局部聚类中心点，不仅减少了对数据分布形式的依赖，还有效避免了由于中心点选取不当或异常值导致的聚类误差；此外，对于新加入的数据，只需对其进行抽样并合并到原有样本中，无需对全部大数据集进行重复处理，大大提高了聚类问题的处理效率。

3 二阶段聚类方法的主要流程

3.1 局部聚类

首先给出在分布式大数据存储环境下数据块的定义：数据集 D 被分解为多个数据块组成的集合 $D_1,D_2,\cdots,D_k$，若满足条件：

1）$D_i\neq\varnothing,\forall i\in\{1,2,\cdots,k\}$；

2）$\bigcup_{i=1}^{k}D_i=D$。

则称 $D_1,D_2,\cdots,D_k$ 为 D 的一个完全数据块。若还有：

3）$D_i\cap D_j=\varnothing,\forall i,j\in\{1,2,\cdots,k\},i\neq j$。

则称为 D 的一个完全不相交数据块。

当前主流的大数据存储采用分布式文件管理系统 HDFS，它的设计原则是将大型文件分割成固定大小的数据块（默认是 128 MB 或 256 MB），这些数据块分散存储在整个集群的多个节点上。尽管数据被分割成多个块，但这些块的集合完整地代表了整个大数据文件。任何单一数据块的信息都是从原始完整数据中分割出来的，所有块的聚合可以恢复原始数据。因此，HDFS 文件本身就是大数据的一个完全数据块。由于数据被分割并分布式存储，因此，不同的数据块可以在集群的不同节点上并行处理。这不仅提高了数据处理的效率，也优化了资源的利用。

在分布式聚类中，局部聚类是一个重要的步骤，它允许在分布式环境中对数据进行有效管理和分析。局部聚类的核心思想是将大规模的数据集分布在不同的节点上，每个节点独立地对其托管的数据子集进行聚类分析。在分布式计算框架中，特别是使用 Hadoop 和其文件系统 HDFS 时，可以有效地部署分布式聚类算法，其中每个 HDFS 数据块可以被视为进行局部聚类的一个单元。在每个节点上，可以独立地对数据块 D_i 进行聚类分析，得到数据块的局部聚类结果 M_i。这一步骤不涉及跨节点的数据交换，可显著减少网络通信的负担，提升对大数据聚类问题的处理速度。

3.2 节点抽样

在分布式大数据环境中进行节点抽样是关键的数据处理步骤，它使得可以在不必处理整个大数据集的前提下，有效地进行数据处理与分析。节点抽样通常用于减小数据处理的负担，通过抽样样本的代表性来保证计算结果的一致性和稳定性。

本文将不同节点上的数据块视为不同层次的数据，因此节点抽样可以被视为一种分层抽样方法。具体来说，在分布式数据环境中，每个节点持有的数据集可以被视为一个单独的层。这些层可能基于地理位置、数据类型、用户群体或任何其他逻辑方式定义。分层抽样允许并行处理，加快了数据预处理和初步分析的速度。每层的独立抽样保证了样本的代表性以及与全局数据的一致性。这种方法适应于各种数据和节点的配置，可以灵活地调整抽样策略以适应不同的数据层。

我们给出两种经典的分层抽样策略[24]。设总数据量为 N，每层数据量为 $N_1,N_2,\cdots,N_L$；总样本量为 n，每层样本量为 $n_1,n_2,\cdots,n_L$。

1）比例分配。每一层的抽样比等于总的抽样比：对于第 i 层，$\frac{n_i}{N_i}=\frac{n}{N}$。

2）最优分配：对于第 i 层，$n_i=n\frac{N_i\sigma_i}{\sum_{j=1}^{L}N_j\sigma_j}$。其中，$\sigma_i$ 为第 i 层的样本标准差。

如果每一层的标准差都非常接近，那么最优分配和比例分配的精度也接近。事实上，最优分配的方差一定比按比例分配的方差小，但是最优分配涉及到各层的样本标准差，这并不容易得到。因此，如果它们之间相差不大，比例分配可能是一种更高效的选择。

对于总样本量 n，同样可以参考分层抽样策略下的总样本量计算公式：$n=\frac{\sum N_i^2\sigma_i^2}{\sigma^2+\sum N_i^2\sigma_i^2}$，其中 σ 代表总体大数据的标准差。如果计算出的 n 较大，则可以对其进行修正（一般认为 $n>5\%N$ 就需要修正），修正后的 $n'=\frac{n}{1+n/N}$。在实际应用中，可以根据具体场景和效果来选择样本数量。在节点抽样过程中，我们将不同的数据块视为不同层次的数据，并使用分层抽样的方法独立地获得抽样样本。

需要注意的是，节点抽样必须在局部聚类完成后进行，抽样策略需要根据局部聚类的结果进行调整。由于局部聚类结果中可能存在异常点，抽样策略需要确保样本中包含每个局部类的数据，并且包含所有异常点，因此，除了分层抽样的策略以外，还需要额外的抽样规则。在本文的实验中，通过将每个异常点视为一个独立的类，从每个类中先随机抽出一个数据进入样本，再对剩下的数据进行等比例的分层抽样，以保证样本对异常点和类的包含性。

3.3 样本聚类

在分布式大数据环境中，利用中心节点进行全局聚类分析并将结果回传给各个节点是一个复杂的过程，需要根据不同的局部聚类算法，通过局部聚类的中心点合并，或者边界点处理和重叠区域的计算，来完成全局聚类。本方法通过局部聚类获得局部聚类结果；通过节点抽样获得局部代表点；通过对融合的代表点聚类获得样本聚类结果，其结果也可以视为一种全局的聚类结果。由于此方法包含了对局部和样本的两次聚类，因此将其称为二阶段聚类方法。

1）在完成局部聚类和节点抽样之后，从各个节点抽取的局部样本需要传输到一个中心节点。在这里，所有的样本数据会汇集在一起，得到全局数据的样本 S_{total}。

2）使用汇总的样本数据 S_{total} 在中心节点上执行聚类算法，获得样本的二阶段聚类结果 M_{sample}。这一步骤的目的是识别整个分布式系统中数据的全局模式和结构。

3）将样本聚类结果 M_{sample}（各个样本点的聚类标签）回传给原始的各个节点。每个节点将使用这些样本聚类结果来标记其本地数据。

3.4 二阶段聚类结果映射

在中心节点完成样本的二阶段聚类后，将样本的聚类标签传回其原始的节点。本地节点需要进行进一步的分析来整合局部聚类结果 M_i 和样本聚类结果 M_{sample}，最终的目标是将

样本二阶聚类的结果映射到本地的所有数据上，获得本地数据的最终聚类结果 M_{final_i}。因此，需要构造一种有效的策略来整合局部和全局聚类结果，确保本地数据与全局模型的一致性。

本节提出一种依赖于样本聚类标签众数的映射方式，对于节点上的每个局部类别，选择其中被抽中样本的二阶段聚类标签的众数作为该类的新标签。在映射过程中，由于所有的局部聚类结果都会被映射到样本的聚类标签上，因此最终的聚类数目主要以样本聚类的结果为主，最终聚类的簇数必然小于或等于样本聚类的簇数。值得注意的是，众数映射的方法导致局部聚类生成的簇在最终集成时只能合并变大，而不能变小。因此，在局部聚类的过程中，可以设置比预期更高的簇数，使得局部簇的数量更多；在样本聚类时设置预期的簇数，这样在最终集成时局部簇会发生合并，使得最终的簇数能够被样本聚类的结果所限制。

这种映射方式，可以有效地整合分布式环境下的聚类结果，确保全局聚类结果在每个本地节点都能得到正确的体现，这对于确保数据驱动决策的准确性和可靠性至关重要。最终的聚类映射实际上是集成了局部聚类和样本聚类两次聚类的结果，因此，基于节点抽样的分布式二阶段聚类方法在一定程度上也是一种聚类集成方法。

4 二阶段聚类方法的具体实现

4.1 聚类方法

本文提出的基于节点抽样的二阶段聚类方法包含局部聚类、节点抽样、样本二阶段聚类和聚类结果映射 4 个步骤，流程示意图如图 1 所示。

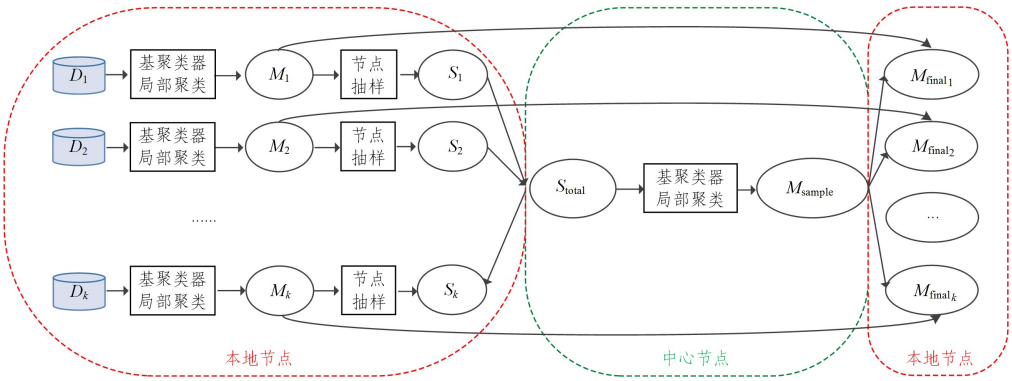


图 1 基于节点抽样的二阶段聚类方法的流程图

Fig. 1 Flowchart of two-stage clustering method based on node sampling

在局部聚类过程中，不需要对数据块的分布进行限制，也不要求各数据块中的数据量相同，各数据块之间可以存在明显差异。由于最终的聚类标签主要由样本二阶段聚类的结果产生，因此也不需要对局部聚类的结果进行限制，各数据块的聚类结果可以不同，不需要强制各数据块聚类成相同数量的类。故在数据块局部聚类后，本地节点上会存在由局部聚类算法完成划分的多个类。

节点抽样在局部聚类之后进行，需要对局部聚类的结果进行抽样。抽样策略应确保在每个局部类中至少抽取一个样本，使得每个局部的簇至少在样本中有一个代表点，这样可以保证局部的簇能够进行最终的结果映射。需要说明的是，对于某些局部聚类算法，聚类结果中可能存在异常点。为了使最终的聚类结果可以映射到数据集中的每个点，所有的局部聚类的异常点都需要包含在抽样的样本中。为此，给出两种方法来保证异常点的全面覆盖。

1)异常点的独立分类：每个异常点都是一个单独的类，类中仅有一个元素。在抽样过程中设置每个局部类中至少有一个样本被抽取，对剩余的数据按照给定的抽样策略抽取，从而保证最终样本中包含所有的异常点。

2)异常点的大类归并：可以将所有异常点视为一个大类，并直接将这个大类归入样本集中，以简化抽样过程并保持样本的完整性。此外，其余样本可以按照给定的抽样策略进行抽样。

二阶段聚类方法在中心节点上使用汇总的样本数据进行聚类分析，可以识别整个分布式系统中数据的全局模式和结构。本方法使用抽样样本代替传统的中心点或者计算复杂的局部簇代表点，通过二次样本聚类简化了传统方法中复杂的合并或划分局部聚类结果的过程，因此适用于不同的基聚类算法，为大数据聚类提供了一个通用的分布式处理框架。

将全局聚类结果回传给各个节点。每个节点将使用这些聚类结果标记其本地数据，从而获得一致的全局聚类结果。由于局部聚类的每个异常点被视为一个单独的类且包含在样本中，因此，局部的异常点也会被映射到最终样本的二阶段聚类结果中。这种方法能够确保即使是局部异常点，也能在全局聚类结果中得到体现和处理。我们采用的众数映射方法能够显著减弱小概率事件或异常点对最终聚类结果的影响。众数映射通过选择局部簇中出现频率最高的样本聚类结果来标记此簇，从而可以有效地减少由小概率事件或异常点引起的干扰和偏差。

如图 1 所示，二阶段聚类中，大部分的操作(局部聚类、抽样及两次聚类结果的集成)都是在本地节点上运行，只有很少的操作(样本聚类)在中央节点运行。之前的方法将所有的聚类结果都放在中央节点上进行处理，导致中央节点复杂度过高，成为分布式聚类的瓶颈。与此不同，二阶段聚类不需要将所有节点的聚类结果传输到中央节点，只需将节点的抽样样本发送到中央节点。中央节点仅负责对收到的抽样样本进行第二阶段的聚类分析。在中央节点完成第二阶段的聚类后，将得到的聚类结果回传给本地节点，在本地节点上进行最终

的聚类结果整合，这进一步减轻了中央节点的计算负担。因此，本地节点负责局部聚类、节点抽样及两次聚类结果的整合；中央节点只进行样本的二阶段聚类。这一阶段的计算任务相对简单，只涉及少量的样本，样本量甚至还可以根据节点数量和抽样率进行调整。

基于节点抽样的二阶段聚类算法的具体步骤如算法 1 所示。

算法 1 基于节点抽样的二阶段聚类方法

数据：一个大数据集$\mathcal{D}$

Step 1：局部聚类

1. 对于每个节点执行以下操作：
2. 局部聚类
3. 保存局部聚类结果

Step 2：节点抽样

4. 对于每个节点执行以下操作：
5. 从每个局部聚类结果中进行抽样：确保至少从每个类中抽取一个样本，且样本需要包含所有异常点
6. 将抽样数据传输到中心节点

Step 3：二阶段聚类（全局聚类）

7. 在中心节点执行：
8. 聚合所有节点的抽样数据
9. 对样本聚合数据进行聚类
10. 保存全局聚类结果
11. 将全局聚类结果回传给每个节点

Step 4：聚类结果映射

12. 对于每个节点执行以下操作：
13. 对节点内的每个局部类执行以下操作：
14. 收集该类中样本的全局聚类标签
15. 计算众数以确定该类的新全局标签
16. 使用新的全局标签更新局部类标签

结果：输出更新后的每个节点的全局聚类结果

4.2 复杂性分析

所提二阶段聚类方法通过分布式策略处理大数据聚类问题，其时间复杂度主要受局部聚类算法复杂度的影响。二阶段聚类的计算时间可以分为 4 个部分。

1）局部聚类。局部聚类在每个节点上独立运行，处理各自节点的数据。假设总数据量为 N，共有 L 个节点，每个节点处理$\frac{N}{L}$的数据量。如果直接聚类的时间复杂度为 $O(c(N))$，则选用同样的算法进行局部聚类的时间复杂度为 $O\left(c\left(\frac{N}{L}\right)\right)$。

2）节点抽样。在每个节点上进行节点抽样操作，时间复杂度为 $O\left(\frac{N}{L}\right)$。

3）聚合节点抽样结果并对样本进行二阶段聚类。在这一步操作中，处理的是聚合后的样本数据。假设样本总量为 n，采用同样的聚类算法，时间复杂度为 $O(c(n))$。这个步骤处理的数据量较少，所以时间复杂度比对全数据直接聚类低很多。

4）局部聚类结果映射。如果在每个节点上进行比例分配抽样，使得每个节点上的样本数量都接近$\frac{n}{L}$，则在每个节点上对局部类中被抽中的样本求其样本聚类标签的众数的复杂度低于 $O\left(\frac{n}{L}\log\frac{n}{L}\right)$。这也进一步证明了二阶段聚类方法在分布式环境下的高效性。

5 实验验证与分析

本章将基于仿真数据和公开数据集来对基于节点抽样的二阶段聚类方法的可行性和有效性进行验证。仿真数据主要是为了提供一种直观的可视结果而使用，可以更好地显示出聚类的运行过程；公开数据主要是评价二阶段聚类方法在复杂数据集上的实际聚类效果。其中，实验 1 到实验 3 是在仿真环境中完成的，而实验 4 则是在分布式环境中实现的。

仿真环境是在 Python 3.8 环境上搭建的，使用了 Intel i7-10700 CPU 和 32 GB 内存；分布式环境建立在配置了 YARN 调度器的 Spark 集群上，集群由 5 个计算节点组成，每个节点配备了 48 核的 Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70 GHz、256 GB 内存和 2 TB 外存。集群使用的软件包括 Spark 2.4.0，HDFS 3.0.0，YARN 3.0.0 以及 JDK 1.8.0_212，操作系统为 CentOS Linux release 7.9.2009(Core)。

5.1 复合型数据仿真实验

本节首先展示了 DBSCAN 算法及其对应的二阶段聚类方法在复合型仿真数据（月亮型数据、球形数据和环形数据）上的实验，其目的是清晰地展示算法的执行过程。

5.1.1 实验描述与流程展示

该实验中的仿真数据分为四大簇，数据分布如图 2(a)所示。簇 1 和簇 2 是两类月亮型数据，各有 5 000 个样本点；簇 3 和簇 4 分别是球形和环形数据，各有 10 000 个样本点。考虑到数据集的形状，在本实验中选用 DBSCAN 作为基聚类算法对这些数据进行聚类。

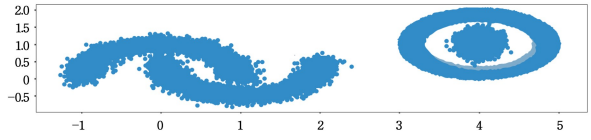
首先对总体数据进行 DBSCAN 聚类，设置最大半径 $Epsilon_{总体}=0.2$，最小点数 $minPts_{总体}=250$，聚类效果如图 2(b)所示。可以看到，DBSCAN 很好地区分出了 4 个类，并识别出少量异常点。

接下来，将数据集划分为 5 个完全不相交的数据块。二阶段聚类并不要求数据块与全量数据的分布一致。由图2(c)—图 2(g)可以看出 5 个数据块的数据存在明显的分布差异，其中数据块 1 包含 9 500 个数据点，数据块 2 包含 6 500 个数据点，数据块 3 包含 9 000 个数据点，数据块 4 包含 2 000 个数据点，数据块 5 包含 3 000 个数据点。

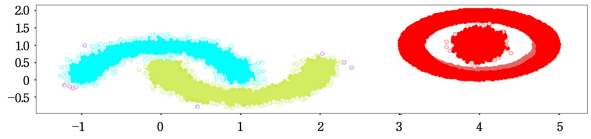
对每个数据块使用 DBSCAN 进行局部聚类，统一设置最大半径 $Epsilon_{局部}=0.2$，最小点数 $minPts_{局部}=150$，5 个数据块的聚类效果如图 2(c)—图 2(g)所示。可以看到，每个数据块都分别进行了聚类，在这些局部聚类结果中有可能会存在若干异常点。

接下来从每个数据块中抽取样本，并对总体样本进行聚类。在节点抽样时，先选取了节点上的异常点，然后从每个类中随机抽取一个样本。接着，对所有节点上的数据按照 10% 的比例进行随机抽样。采用 DBSCAN 算法进行样本聚类，设置最大半径 $Epsilon_{样本}=0.2$，最小点数 $minPts_{样本}=40$。聚类效果如图 2(h)所示。

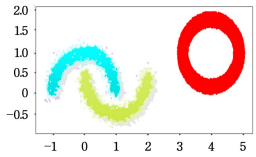
最后，对样本聚类结果和局部聚类结果进行映射，得到最终的聚类结果，如图 2(i)所示。可以看到，最终的聚类结果很好地区分了 4 个簇，并且异常点较少。异常点的减少主要是因为最终的聚类映射集成了两次聚类的结果：局部聚类中的异常点在样本聚类中有可能不是异常点，而被归为某一类，因此在映射过程中也会被归类为正常点；样本聚类中的异常点或某些聚类标签可能会在求众数的过程中被掩盖。因此，二阶段聚类一定程度上也是一种聚类集成方法。



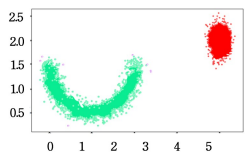
(a)数据集



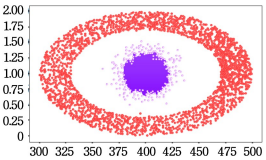
(b)全部数据 DBSCAN 聚类结果



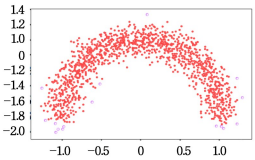
(c)块 1 DBSCAN



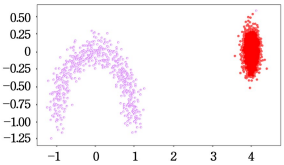
(d)块 2 DBSCAN



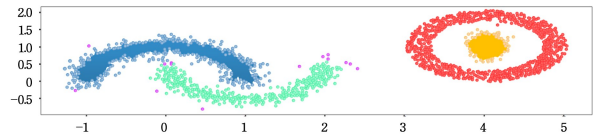
(e)块 3 DBSCAN



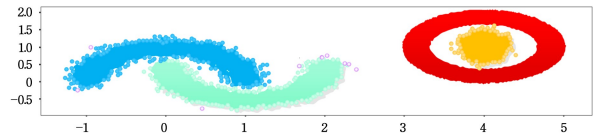
(f)块 4 DBSCAN



(g)块 5 DBSCAN



(h)样本 DBSCAN 聚类结果



(i)融合 DBSCAN 聚类结果

图 2 复合型仿真数据二阶段 DBSCAN 聚类流程与结果展示

Fig. 2 Presentation of two-stage DBSCAN clustering results on composite simulation data

5.1.2 参数选择讨论

本节讨论在二阶段聚类框架下如何进行聚类算法参数的选择。在图 2 所示的实验中，总体聚类 DBSCAN 算法的最大半径 $Epsilon_{总体}=0.2$，最小点 $minPts_{总体}=250$。局部聚类中，dataset 被划分为 5 个完全不相交的数据块。固定最大半径 $Epsilon_{局部}$，讨论最小点 $minPts_{局部}$ 的取值。

在二阶段聚类框架中，所有局部数据块聚类的簇(包括异常点)都会被映射到样本二阶段聚类的簇中，最终聚类的结果更多地取决于样本二阶段聚类的结果。因此，在局部聚类的过程中，可以适当地增加类的个数，使得局部聚类的簇更紧致，$minPts_{局部}$ 取值应较大。考虑到数据块分布可能不一致，各数据块的密度可能也大于 $\frac{总体密度}{数据块个数}$，因此局部聚类 $minPts_{局部}$ 应远大于 $\frac{总体\ minPts}{数据块个数}$，这里取 $minPts_{局部}=150$。

在样本聚类过程中，由于是较随机地抽取样本，因此，样本的密度应较靠近总体密度与抽样比例的乘积。最终的聚类标签主要取决于样本二阶段聚类的结果，在固定 $Epsilon_{样本}=0.2$ 的情况下，相较于局部聚类，$minPts_{样本}$ 会更接近 $minPts_{总体}$ 与抽样比例的乘积。考虑到抽样的随机性，$minPts_{样本}$ 需要取值偏大，以降低将不同类别样本聚在一起的概率。尽管这样可能会增加样本聚类识别的异常点个数，但如前所述，最终的聚类结果是局部聚类和样本二阶段聚类的集成，异常点数量会大幅减少。因此，选取较大的 $minPts_{样本}$ 是一个合理的策略。在本实验中，选取 $minPts_{样本}=40$。

本文提出的二阶段聚类方法本身可以视为一种聚类集成方法。另外，它也适合采用多种聚类算法进行集成：在样本聚类环节，可以选择不同的聚类算法进行多次聚类并集成，从而获得集成的样本聚类结果。由于最终的聚类标签主要取决于样本聚类的结果，因此只需对少量样本进行多次聚类集成，集成的效果提升会反映在全部数据上。

5.1.3 实验效果分析

针对仿真数据集，采用 DBSCAN 算法作为基算法进行总体聚类(总体 DBSCAN)和基于样本的二阶段聚类(二阶段 DBSCAN)。每个数据点将包含两个聚类标签：总体聚类标签和二阶段聚类标签。由于数据是仿真生成的，我们预先知道每个数据的真实标签。当某个数据的聚类标签与真实标签一致时，将其视为聚类结果正确，否则记为错误。分别计算数据集中总体聚类和二阶段聚类的聚类结果准确率，重复进行 1 000 次实验得到如表 1 所列的实验结果。可以发现，基于二阶段 DBSCAN 的聚类准确率远高于总体 DBSCAN 的结果，这表明所提方法在本实验中并不会降低聚类算法的精度，相反，还提升了聚类的准确率。

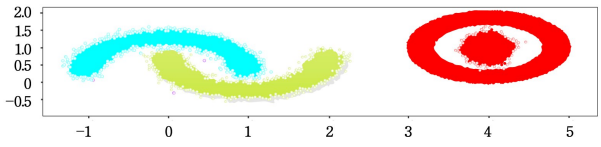
表 1 复合型仿真数据二阶段 DBSCAN 的聚类效果

Table 1 Results of two-stage DBSCAN clustering on composite simulation data

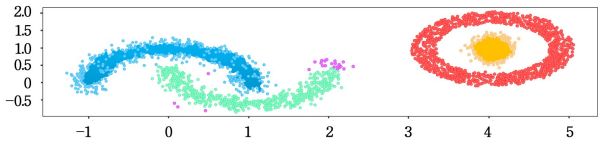
方法	平均准确率	准确率标准差	准确率 95%置信区间
总体 DBSCAN $Epsilon_{总体}=0.2, minPts_{总体}=250$.	89.52%	0.1544	(0.8856, 0.9047)
二阶段 DBSCAN $Epsilon_{局部}=0.2, minPts_{局部}=150$; $Epsilon_{样本}=0.2, minPts_{样本}=40$.	**98.08%**	**0.0520**	**(0.9775, 0.9839)**

图 3 展示了 1 000 次实验中的某次具有代表性的聚类结果。可以观察到，由于数据生成的随机性，簇 3 和簇 4 存在

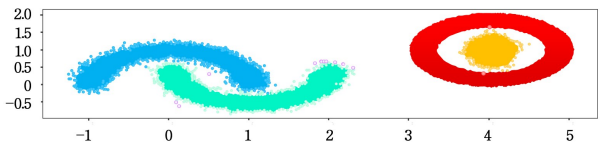
部分重叠的点。因此，在总体数据聚类中，DBSCAN 将簇 3 和簇 4 识别为一个类别，导致整个类别被错误识别，从而降低了总体聚类的准确性。然而，在局部聚类和样本聚类中，由于仅使用了部分数据，降低了重叠部分的密度，因此二阶段 DBSCAN 能够分别识别出簇 3 和簇 4。这也正是此实验中二阶段聚类方法的准确率远高于总体聚类的原因所在。



(a)全部数据 DBSCAN 聚类结果



(a)样本 DBSCAN 聚类结果



(c)融合 DBSCAN 聚类结果

图 3 复合型仿真数据 DBSCAN 二阶段聚类结果特例展示

Fig. 3 Special case of two-stage DBSCAN clustering on composite simulation data

这个结果也说明了二阶段聚类是对多个数据块的聚类结果的集成，相当于将不同局部视角的信息汇总起来。在两次聚类结果映射过程中，通过对样本聚类结果和局部聚类结果的综合考量，可以更准确地确定每个样本的最终类别，从而提升对数据集整体的聚类效果。

同时，局部聚类和样本聚类的分开执行可以方便对不同的数据分布分别进行参数调整。相较于总体聚类中所有数据都使用相同的参数的情况，这种分块聚类和样本聚类结合的方法能更好地适应数据的分布特点。因此，对局部聚类和样本聚类采用不同的参数调整也能够进一步提高聚类的准确率。

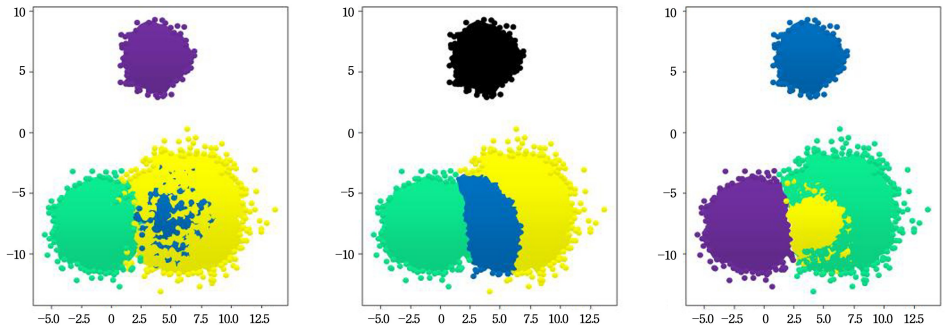
5.2 Gaussian Blobs Data Set 仿真实验

Gaussian Blobs Data Set 常用于测试和评估聚类算法的性能，其数据点被组织成多个高斯分布的“团块”，每个“团块”可以被视为一个高斯分布的簇，其特征包括均值(中心位置)、协方差(形状和方向)和可能的混合权重(如果数据集由多个高斯分布混合而成)。本节采用高斯混合模型(Gaussian Mixture Model，GMM) 对仿真的 Gaussian Blobs Data Set 进行分析，并进行了 GMM 聚类及二阶段 GMM 聚类的效果对比。

5.2.1 实验描述

在本实验中，通过仿真创建了一个包含 4 个高斯分布的 Gaussian Blobs Data Set，这些分布的参数(均值、标准差和样本数量)均由随机数生成，确保了数据的多样性和真实性。

实验的第一步是对整个数据集应用 GMM 聚类，以评估其聚类效果；随后，将数据集随机划分为 5 个独立的子集，并对每个子集独立应用 GMM 进行局部聚类。在局部聚类完成后，根据聚类结果对每个子集进行抽样，除异常点外，抽样比例为 20%；然后利用这些抽样样本再次进行 GMM 聚类；最后，将局部聚类和抽样聚类的结果结合起来，形成对整个数据集的全面聚类分析结果。GMM 聚类算法同样需要预先指定聚类簇的个数 n，在实验中设置 $n_{总体}=4$，$n_{局部}=5$，$n_{样本}=4$。图 4 展示了某次 Gaussian Blobs Data Set 聚类结果。



(a)实际类别 (b)GMM 聚类结果 (c)二阶段 GMM 聚类结果

图 4 Gaussian Blobs Data Set 二阶段 GMM 聚类结果展示

Fig. 4 Presentation of two-stage GMM clustering results on Gaussian Blobs Data Set

我们设置了 1 000 次重复实验来评估二阶段 GMM 在仿真数据集上的聚类效果。由于每次实验的数据集都是通过随机生成的参数构建的，这可能导致每次实验中的数据集存在显著差异。我们对每次实验的总体 GMM 聚类和二阶段 GMM 聚类的结果进行了效果评估。

5.2.2 实验效果分析

考虑到数据集中的高斯分布可能存在大量的重叠区域，选择了调整互信息(Adjusted Mutual Information，AMI)作为聚类效果的评估指标。AMI 能够更准确地衡量聚类结果与真实标签之间的一致性，即使在簇之间存在重叠的情况下也能给出合理的评估。AMI 得分越高，表示聚类结果与真实标签之间的相似度越高。

表 2 中的实验结果显示，总体 GMM 聚类的 AMI 评分均值略高于二阶段 GMM 聚类，但两者之间的差异仅为0.09%。为了进一步验证这种差异是否具有统计学意义，对两组 AMI 评分进行了假设检验。检验的 P 值为 0.98，表明这种差异

非常不显著。因此可以得出结论:二阶段 GMM 聚类的 AMI 评分与总体 GMM 聚类没有显著差异。

表 2 Gaussian Blobs Data Set 二阶段 GMM 聚类效果

Table 2 Results of two-stage GMM clustering on Gaussian Blobs Data Set

方法	平均 AMI/%	AMI 标准差	AMI 95%置信区间
总体 GMM $n_{总体}=4$	74.88	0.1423	(0.7399,0.7576)
二阶段 GMM $n_{局部}=5, n_{样本}=4$	**74.79**	**0.1437**	**(0.7390,0.7568)**

这一发现表明,尽管二阶段 GMM 聚类在处理 Gaussian Blobs Data Set 数据集时采用了不同于总体 GMM 聚类的策略,但其最终的聚类效果与总体 GMM 聚类相当。这一结果对于理解不同聚类算法在特定数据集上的表现具有重要意义,尤其是在数据分布复杂且存在重叠的情况下。

5.3 MNIST 数据集聚类

MNIST 数据集[44]是一个被广泛采用的手写数字图像集,常用于图像识别系统的性能验证。该数据集由 70000 张 28×28 像素的灰度图像组成,每张图像代表一个从 0 到 9 的数字,涵盖了 10 个不同的类别。本实验中,采用了 K-means 聚类算法对 MNIST 数据集进行了分析,并对比了传统 K-means 聚类与二阶段 K-means 聚类方法的效果。

首先对 MNIST 数据集进行了主成分分析(Principal Component Analysis,PCA),将其维度降至 50 维,随后应用 K-means 算法进行聚类,设定 10 个聚类类别。然后,将整个数据集随机划分为 4 个完全不相交的数据块。在实验中,每个数据块都独立地应用 PCA 降维至 50 维,并使用 K-means 算法进行局部聚类,其中每个子集的聚类类别数设定为 15,这一数值高于总体聚类的数量。

在局部聚类后,首先从每个类别中随机选取一个样本,之后对所有局部类中的数据进行抽样比例为 10%的随机抽样,接着合并所有抽样数据并进行 PCA 降维处理,然后执行 K-means 聚类。为确保样本聚类的类别数与总体聚类一致,设置 $K_{样本}=K_{总体}=10$。

对于 MNIST 数据集,进行一次总体聚类分析,因此只有一个总体聚类结果,不再对其进行标准差与置信区间的计算(总体 K-means 聚类的 AMI 标准差与置信区间标记为 NULL)。对于二阶段聚类,由于每次的子集划分是随机的,因此每次二阶段聚类的结果都有所不同,故重复 100 次二阶段聚类实验,并使用 AMI 指标的平均值来评估聚类效果。对比结果如表 3 所列。从中可以看出,二阶段 K-means 聚类的 AMI 评分显著高于总体 K-means 聚类。这表明二阶段聚类方法对于复杂的高维数据集(如 MNIST)依然有效。

表 3 MNIST 数据集二阶段 K-means 聚类效果

Table 3 Results of two-stage K-means clustering on MNIST dataset

方法	平均 AMI/%	AMI 标准差	AMI 95%置信区间
总体 K-means $K_{总体}=10$	43.03	NULL	NULL
二阶段 K-means $K_{局部}=15; K_{样本}=10$	**47.08**	**0.0133**	**(0.4682,0.4734)**

5.4 KDD Cup 1999 Data 数据集聚类

KDD Cup 1999 Data[45]是一个广泛用于入侵检测研究的基准数据集,由 DARPA 实验室在 1998 年的"DARPA Intrusion Detection Evaluation Program"中收集的网络流量日志构建,是当今最著名的网络入侵检测数据集之一。KDD Cup 1999 Data 包含超过 400 万条网络连接记录,每条记录描述了网络环境中的一次网络连接。这些连接被标注为正常(normal)或多种不同类型的攻击(attack)。每条记录由 41 个特征组成,包括基本特征、内容特征和基于时间的流量特征。

每次随机选取 KDD Cup 1999 Data 的 30%作为总的数据集,采用 K-means 算法,总体的聚类数为 $K_{总体}$。数据集的初始划分通过 Hadoop 分布式文件系统 HDFS 进行,使用弹性分布式数据集 RDD 将总数据分为 5 个完全不相交的数据块。同样地,对每个数据块分别进行 K-means 聚类,设置不同的局部聚类数 $K_{局部}$来评估不同参数的影响。接下来,从每个数据块中抽取样本,并对总体样本进行聚类。在节点抽样时,首先选取每个节点上的异常点,然后从每个类中随机抽取一个样本;接着对所有节点上的数据按照 10%的比例进行随机抽样。同样采用 K-means 算法对样本进行聚类,由于样本聚类的标签决定了最终集成的聚类标签,为了便于效果对比,样本聚类的簇数需要与总体聚类一致,因此设置 $K_{样本}=K_{总体}$。

由于真实的类别数目和我们设置的聚类数可能不一致,因此仍用 AMI 来衡量聚类效果。表 4 列出了不同参数设置下,二阶段 K-means 聚类方法的聚类结果。在每组参数下重复 100 次实验,取 AMI 的平均值作为最终实验结果。可以看到,KDD Cup 1999 Data 的二阶段 K-means 聚类效果优于总体 K-means 聚类。这表明即使在面对具有复杂多类型特征的数据集时,二阶段聚类算法也能提供准确的聚类结果。仿真数据和公开数据的实验结果表明,二阶段聚类方法能够适应不同类型的数据分布和结构。

表 4 KDD Cup 1999 Data 二阶段 K-means 聚类效果

Table 4 Results of two-stage K-means on KDD Cup 1999 Data

方法	平均 AMI/%	AM 标准差	AMI 95%置信区间
总体 K-means $K_{总体}=2$	60.89	0.0004	(0.6088,0.6089)
二阶段 K-means $K_{样本}=2, K_{局部}=6$	**62.96**	**0.0053**	**(0.6285,0.6306)**
总体 K-means $K_{总体}=4$	88.68	0.0292	(0.8811,0.8926)
二阶段 K-means $K_{样本}=4, K_{局部}=6$	**89.16**	**0.0082**	**(0.8900,0.8932)**
总体 K-means $K_{总体}=6$	88.39	0.0079	(0.8824,0.8855)
二阶段 K-means $K_{样本}=6, K_{局部}=6$	**88.78**	**0.0019**	**(0.8875,0.8882)**

6 结论与未来工作

6.1 结论

本文提出了一种基于节点抽样的二阶聚类方法,用于解决分布式存储的大数据聚类问题。该方法首先对数据进行局部聚类,然后从每个局部聚类结果中抽取样本,对样本进行

二阶聚类,并通过映射局部聚类结果和样本二阶聚类结果,得到最终的聚类结果。本方法具有以下优势。

1)提供了通用的分布式聚类框架。该方法为不同的聚类算法提供了一个统一的分布式计算模型。这意味着无论是基于中心的、基于密度的还是基于图的聚类算法,都可以通过这个框架高效地在分布式环境中运行,无需对算法本身进行大规模的修改或定制化计算。

2)提高了大数据聚类的计算效率。本文方法在不损失算法精度的前提下,能显著减少网络中的实时数据交换,降低通信成本,提升聚类算法的计算效率。此外,该方法极大地节省了计算资源,加快了处理速度,使得聚类算法更适合处理大规模数据集。

3)提升了大数据聚类的聚类质量。相较于传统的总体聚类方法,基于样本的二阶聚类方法可被视为一种集成聚类方法,从而提升了整体聚类效果。

4)具有更好的大数据聚类适用性。通过对不同数据块的局部聚类和样本聚类,可以方便地根据不同数据分布选择聚类算法并对聚类算法进行参数调整。这种针对性的调整能够更好地适应数据分布的特点,进一步提高聚类的准确率。

因此,基于样本的二阶聚类方法不仅能够有效地提升聚类效果,而且具有较强的适应性和稳健性,是一种高效的大数据聚类方法。

6.2 未来工作

考虑到基于节点抽样的二阶段聚类方法具备的处理动态数据的潜能以及采用集成策略所具有的优势,未来将围绕以下两个方面对二阶段聚类方法进行功能拓展与性能优化。

1)增量式更新。当新数据到达时,可以将其分发到相应的节点,每个节点只处理自己的局部数据,因此新增数据只会影响到相应节点的局部聚类结果。接着,根据一定的抽样策略从新增数据中抽取样本,并将其整合到原有的数据样本中,对整合后的样本再次进行二阶段样本聚类,并在每个节点上更新最终的聚类标签。因此,二阶段聚类方法在处理动态增加的数据时能够保持较好的性能,不会因为整体数据的变化而引起大规模的重新计算。这种并行处理方式使得二阶段聚类方法能够高效地处理大规模数据,并且能够在处理动态增加数据时保持高性能。

2)聚类集成。对不同的数据块,可以选取不同的局部聚类算法;也可以选择不同的聚类算法进行多次样本聚类,从而获得集成的样本聚类结果。因此,基于样本的二阶聚类方法为后续的集成学习提供了便利,使得算法更具灵活性和实用性。

参 考 文 献

[1] AHMED M,SERAJ R,ISLAM S M S. The k-means algorithm: A comprehensive survey and performance evaluation [J]. Electronics,2020,9(8):1295.

[2] IKOTUN A M,EZUGWU A E,ABUALIGAH L,et al. A comprehensive review of K-means clustering algorithms:Variants analysis and advances in big data era[J]. Information Sciences,2023,622:178-210.

[3] NIELSEN F,NIELSEN F. Hierarchical clustering [M]// Introduction to HPC with MPI for Data Science. Springer International Publishing,2016:195-211.

[4] JACKSI K,IBRAHIM R K,ZEEBAREE S R M,et al. Clustering documents based on semantic similarity using HAC and K-means algorithms [C]// Proceedings of the 2020 International Conference on Advanced Science and Engineering(ICOASE). IEEE,2020:205-210.

[5] RAMADHANI F,ZARLIS M,SUWILO S. Improvement of the BIRCH algorithm for big data clustering [C]// IOP Conference Series: Materials Science and Engineering. IOP Publishing,2020,725(1):012090.

[6] ZHANG T,RAMAKRISHNAN R,LIVNY M. BIRCH:an efficient data clustering method for very large databases [J]. ACM Sigmod Record,1996,25(2):103-114.

[7] ESTER M,KRIEGEL H P,SANDER J,et al. A density-based algorithm for discovering clusters in large spatial databases with noise [C]// Proceedings of the Second International Conference on Knowledge Discovery and Data Mining(KDD'96). AAAI Press,1996:226-231.

[8] DENG D. DBSCAN clustering algorithm based on density [C]// Proceedings of the 2020 7th International Forum on Electrical Engineering and Automation(IFEEA). IEEE,2020:949-953.

[9] ANKERST M,BREUNIG M M,KRIEGEL H P,et al. OPTICS:Ordering points to identify the clustering structure [J]. ACM Sigmod Record,1999,28(2):49-60.

[10] AGRAWAL R,GEHRKE J,GUNOPULOS D,et al. Automatic subspace clustering of high dimensional data for data mining applications [C]// Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. ACM,1998:94-105.

[11] WANG W,YANG J,MUNTZ R. STING:A statistical information grid approach to spatial data mining [C]// Proceedings of the 23rd International Conference on Very Large Data Bases(VLDB'97). Morgan Kaufmann Publishers Inc.,1997:186-195.

[12] MOON T K. The expectation-maximization algorithm [J]. IEEE Signal Processing Magazine,1996,13(6):47-60.

[13] DO C B,BATZOGLOU S. What is the expectation maximization algorithm? [J]. Nature Biotechnology,2008,26(8):897-899.

[14] KAUFFMANN J,ESDERS M,RUFF L,et al. From clustering to cluster explanations via neural networks [J]. IEEE Transactions on Neural Networks and Learning Systems,2022,35(2):1926-1940.

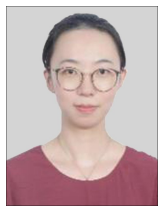
[15] REN Y,PU J,YANG Z,et al. Deep clustering:A comprehensive survey [J]. arXiv:2210.04142,2022.

[16] MAHDI M A,HOSNY K M,ELHENAWY I. Scalable clustering algorithms for big data:A review [J]. IEEE Access,2021,9:80015-80027.

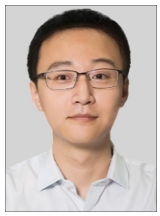
[17] OTHMAN S M,BA-ALWI F M,ALSOHYBE N T,et al. Intrusion detection model using machine learning algorithm on Big Data environment [J]. Journal of Big Data,2018,5(1):1-12.

[18] LI J,IZAKIAN H,PEDRYCZ W,et al. Clustering-based anomaly detection in multivariate time series data [J]. Applied Soft

Computing, 2021, 100: 106919.
[19] LI G, LIU F, SHARMA A, et al. Research on the natural language recognition method based oncluster analysis using neural network [J]. Mathematical Problems in Engineering, 2021, 2021(1): 9982305.
[20] CHOWDHURY G G. Natural language processing [J]. Annual Review of Information Science and Technology, 2003, 37(1): 51-89.
[21] SILVERSTEIN C, MARAIS H, HENZINGER M, et al. Analysis of a very large web search engine query log [C]// Proceedings of the ACM SIGIR Forum. ACM, 1999: 6-12.
[22] ZHAO S, ZHU L, WANG X, et al. Centerclip: Token clustering for efficient text-video retrieval [C]// Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022: 970-981.
[23] MENG Q, ZHAO S, HUANG Z, et al. Magface: A universal representation for face recognition and quality assessment [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 14225-14234.
[24] YANG L, CHEN D, ZHAN X, et al. Learning to cluster faces via confidence and connectivity estimation [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 13369-13378.
[25] SAMBASIVAM S, THEODOSOPOULOS N. Advanced data clustering methods of mining Web documents [J]. Issues in Informing Science & Information Technology, 2006, 3: 563-579.
[26] SUN J G, LIU J, ZHAO L Y. Research on Clustering Algorithms [J]. Journal of Software, 2008, 19(1): 48-61.
[27] OYEWOLE G J, THOPIL G A. Data clustering: application and trends [J]. Artificial Intelligence Review, 2023, 56(7): 6439-6475.
[28] HASHEMI S E, GHOLIAN-JOUYBARI F, HAJIAGHAEI-KESHTELI M. A fuzzy C-means algorithm for optimizing data clustering [J]. Expert Systems with Applications, 2023, 227: 120377.
[29] HUANG Z. Extensions to the k-means algorithm for clustering large data sets with categorical values [J]. Data Mining and Knowledge Discovery, 1998, 2(3): 283-304.
[30] BAHMANI B, MOSELEY B, VATTANI A, et al. Scalable K-Means+ [J]. Proceedings of the VLDB Endowment, 2012, 5(7): 622-633.
[31] MAO Y M, GAN D J, MWAKAPESA D S, et al. A MapReduce-based K-means clustering algorithm [J]. The Journal of Supercomputing, 2022, 78(4): 5181-5202.
[32] SARDAR T H, ANSARI Z. Distributed big data clustering using MapReduce-based fuzzy C-medoids [J]. Journal of The Institution of Engineers(India): Series B, 2022, 103(1): 73-82.
[33] ZHANG T, RAMAKRISHNAN R, LIVNY M. BIRCH: an efficient data clustering method for very large databases [J]. ACM Sigmod Record, 1996, 25(2): 103-114.
[34] RAMADHANI F, ZARLIS M, SUWILO S. Improvement of the BIRCH algorithm for big data clustering [C]// IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2020.
[35] SCHUBERT E, SANDER J, ESTER M, et al. DBSCAN revisited, revisited: why and how you should(still) use DBSCAN [J]. ACM Transactions on Database Systems(TODS), 2017, 42(3): 1-21.
[36] ANKERST M, BREUNIG M M, KRIEGEL H P, et al. OPTICS: Ordering points to identify the clustering structure [J]. ACM Sigmod Record, 1999, 28(2): 49-60.
[37] HINNEBURG A, KEIM D A. A general approach to clustering in large databases with noise [J]. Knowledge and Information Systems, 2003, 5: 387-415.
[38] MU C, HOU Y, ZHAO J, et al. Stream-DBSCAN: A streaming distributed clustering model for water quality monitoring [J]. Applied Sciences, 2023, 13(9): 5408.
[39] WU G, CAO L, TIAN H, et al. HY-DBSCAN: A hybrid parallel DBSCAN clustering algorithm scalable on distributed-memory computers [J]. Journal of Parallel and Distributed Computing, 2022, 168: 57-69.
[40] XIE M, AREF W G. PD-DBSCAN: A density-based clustering algorithm for exploratory data analysis and data mining [J]. Knowledge and Information Systems, 2013, 37(3): 685-722.
[41] KAUFFMANN J, ESDERS M, RUFF L, et al. From clustering to cluster explanations via neural networks [J]. IEEE Transactions on Neural Networks and Learning Systems, 2022, 35(2): 1926-1940.
[42] LIU Y, TU W, ZHOU S, et al. Deep graph clustering via dual correlation reduction [C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2022: 7603-7611.
[43] COCHRAN W G. Sampling Techniques [M]. John Wiley & Sons, 1977.
[44] LECUN Y, CORTES C, BURGES C. The MNIST database of handwritten digits [J]. Neural Computation, 1998, 10(5): 1191-1232.
[45] HETTICH S, KEGELMEYER W P. The UCI KDD Archive. KDD Cup 1999 Data[EB/OL]. https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data.



ZHANG Manjing, born in 1992, Ph.D, research associate. Her main research interests include statistical analysis theory and method for big data, design and application of big data machine learning algorithms.



HE Yulin, born in 1982. Ph.D, research fellow, is a member of CCF (No. 97303M). His main research interests include big data approximate computing technologies, multi-sample statistics theories and methods, and data mining and machine algorithms and their applications.

(责任编辑:柯颖)