ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 38,No. 2,Mar. 2023,pp. 426－438
DOI:10. 16337/j. 1004-9037. 2023. 02. 017

http://sjcj. nuaa. edu. cn
E-mail:sjcj@nuaa. edu. cn
Tel/Fax:＋86-025-84892742

基于RAPIDS的无参DBSCAN算法

卢建云[1,2]，邵俊明[1]，张　蔚[3]

(1. 电子科技大学计算机科学与工程学院,成都611731;2. 重庆电子工程职业学院人工智能与大数据学院,重庆401331;3. 中国电子科技集团公司第二十九研究所,成都610036)

摘　要：具有噪声的基于密度的空间聚类(Density-based spatial clustering of applications with noise，DBSCAN)能够发现不同密度和大小的类簇,对噪声也有很好的鲁棒性,被广泛地应用到数据挖掘的任务中。DBSCAN通常需要调整参数*MinPts*和*Eps*以达到更优的聚类效果,但往往在搜索最优参数的过程中会影响DBSCAN的性能。本文从两个方面优化DBSCAN,一方面,提出一种无参的方法优化DBSCAN全局参数选择。无参方法利用自然最近邻获得数据集的自然特征值,并将自然特征值作为参数*MinPts*值。然后,根据自然特征值计算自然特征集合,利用自然特征集合中的数据分布特性,分别采取统计最小值、平均值和最大值3种方式得到*Eps*值。另一方面,采用集成数据科学实时加速平台(Real-time acceleration platform for integrated data science,RAPIDS)的图形处理器(Graphics processing unit，GPU)计算加快DBSCAN算法的收敛速度。实验结果表明,本文提出的方法在优化DBSCAN参数选择的同时,取得了与密度峰值聚类(Density peaks clustering，DPC)相当的聚类结果。

关键词：集成数据科学实时加速平台;图形处理器;具有噪声的基于密度的空间聚类;自然最近邻;聚类

中图分类号：TP391　　**文献标志码**：A

Parameter-Free DBSCAN Algorithm Based on RAPIDS

LU Jianyun[1,2]，SHAO Junming[1]，ZHANG Wei[3]

(1. School of Computer Science and Engineering,University of Electronic Science and Technology of China,Chengdu 611731,China;2. School of Artificial Intelligence and Big Data,Chongqing College of Electronic Engineering，Chongqing 401331,China;3. The 29th Research Institute,China Electronics Technology Group Corporation,Chengdu 610036,China)

Abstract：Density-based spatial clustering of applications with noise (DBSCAN) can find clusters of different densities and sizes, is also robust to noise, and is widely used in data mining tasks. DBSCAN needs to adjust the parameters *MinPts* and *Eps* to achieve a better clustering effect, but it often affects the performance of DBSCAN in the process of searching for the optimal parameters. This article optimizes DBSCAN from two aspects. On one hand, a parameter-free method is proposed to optimize DBSCAN global parameter selection. The parameter-free method uses the natural nearest neighbor to obtain the natural feature value of the data set, and uses the natural feature value as *MinPts*. Then, the natural feature set is calculated according to the natural feature value, and three strategies (i.e. statistics of minimum, mean and maximum) are used to obtain the *Eps* values by using the data distribution characteristics of the natural feature set. On the other hand, it uses the graphics processing unit (GPU) of the real-time

基金项目:重庆市教委科学技术研究项目(KJQN202103109)。
收稿日期:2022-09-30;**修订日期**:2022-12-12

acceleration platform for integrated data science (RAPIDS) platform to accelerate the convergence of DBSCAN algorithm. The experimental results show that the proposed method can optimize DBSCAN parameter selection while obtaining the comparable clustering results of density peaks clustering (DPC) algorithm.

Key words: real-time acceleration platform for integrated data science (RAPID); graphics processing unit (GPU); density-based spatial clustering of applications with noise (DBSCAN); natural nearest neighbor; clustering

引　言

具有噪声的基于密度的空间聚类(Density-based spatial clustering of applications with noise, DBSCAN)能够识别不同大小和密度的类簇,并且对噪声也有很好的鲁棒性,一直是数据挖掘和机器学习研究的热点,被成功地应用到众多领域如车联网目标识别[1]、船舶轨迹聚类[2-3]、离群检测[4-5]、隐私保护[6]等。DBSCAN通过定义数据集的核心数据对象,利用核心数据对象之间的直接密度可达,密度可达和密度相连实现数据对象的聚类和噪声检测。DBSCAN利用参数*MinPts*和*Eps*定义核心数据对象。参数*MinPts*指在数据对象邻域半径为*Eps*内的最少邻居数目。*MinPts*和*Eps*之间具有一定的关联性,在实际应用中,可以设置好*MinPts*,然后调整*Eps*,也可以设置好*Eps*,再调整*MinPts*。为了获得更优的聚类效果,DBSCAN往往会搜索最优的参数值,但在这个过程中大量的计算会影响算法的性能。

为了提高DBSCAN的性能,本文从两个方面对其性能进行优化。第一,利用自然最近邻获得数据集的自然特征值作为*MinPts*。然后,根据自然特征值计算自然特征集,利用自然特征集中数据对象的分布特性计算*Eps*候选值。第二,利用集成数据科学实时加速平台(Real-time acceleration platform for integrated data science,RAPIDS)的图形处理器(Graphics processing unit,GPU)计算加快DBSCAN收敛。将得到的*MinPts*和*Eps*参数输入基于GPU的DBSCAN算法,输出聚类结果。实验结果表明,本文提出的DBSCAN优化方法是有效的,并且取得了与密度峰值聚类(Density peaks clustering, DPC)相当的聚类结果。

1　相关工作

DBSCAN提出以来一直是数据挖掘和机器学习领域的研究热点。研究学者从参数选择、性能提升和工程应用3个方面对DBSCAN进行了研究。

在DBSCAN参数选择方面,Uncu等[7]提出了一种基于网格的三级聚类方法。首先,利用网格将数据集进行划分,尽量使同密度的数据对象处于同一个网格中;其次,对密度相似的网格进行合并,然后找出最合适的*MinPts*和*Eps*;最后,将得到的参数输入DBSCAN对数据集进行聚类。Ram等[8]提出了一种增强的DBSCAN以解决同一类簇内部密度变化较大导致生成多个类簇的问题。Enhanced DBSCAN通过跟踪每个核心对象的*Eps*邻域的变化从而实现局部密度检测,如果核心对象的局部密度变化不大于设定的阈值并且满足均质化指标,则允许核心对象进行扩展,进而实现聚类。Varied DBSCAN[9]利用*k*-dist图选择多个*Eps*值,依次对每一个*Eps*值进行聚类,已被聚类的对象不参与下次聚类。有研究人员对Varied DBSCAN中*k*值的选择进行了改进,提出了一种根据数据集特征自动化确定*k*值的方法[10]。另外,文献[11]提出了一种自动确定*MinPts*和*Eps*的方法,首先,初始化一个随机的*Eps*值进行DBSCAN聚类,如果没有发现类簇,则对*Eps*值增大0.5。反之,如果数据集中10%的数据对象被聚类,则将得到的类簇从数据集中移除,再调整*MinPts*和*Eps*的值进行DBSCAN聚类,直到数据集中95%的数据对象被聚类,其余

的数据对象则认为是噪声。ADBSCAN的一个不足是需要指定数据集聚类的数目。王志和等[12]对DBSCAN聚类初始点和*Eps*进行优化，通过反向*k*最近邻和相似度矩阵找出密度最大的数据对象作为聚类初始点，然后对当前类簇的密度进行分析，采用自适应的方式计算当前类簇的*Eps*。

在DBSCAN性能提升方面，Wang等[13]提出了一种鸟群优化方法寻找全局最优的*Eps*值，以降低高维大数据对计算*Eps*值的影响。文献[14]提出了一种近似DBSCAN算法，KNN-BLOCK DBSCAN利用K最近邻(K nearest neighbors, KNN)块来确定核心数据对象，认为一个数据对象如果与他的邻居具有相同的密度分布，则他们属于同一种类型即核心数据对象、边缘数据对象或者噪声。通过快速的近似KNN算法检测核心区块，非核心区块和噪声区块，再将核心区块进行合并，最后把非核心区块中的数据对象聚类到合适的核心区块。Luo等[15]提出了基于Spark的DBSCAN，首先，采用随机采样的方法将数据集进行划分，其次，在本地并行地进行DBSCAN聚类，最后，基于聚类中心将本地的聚类结构进行合并。文献[16-17]将改进的DBSCAN与Spark平台并行聚类计算理论相结合，大幅减少了DBSCAN对内存的占有率，并且与基于Hadoop平台进行了对比分析。

在DBSCAN工程应用方面，文献[1]将DBSCAN应用到交通场景中的多目标跟踪，对毫米波雷达采集的数据进行聚类，解决多径噪点难以去除和纵向的交通目标点云难以区分的问题。DBSCAN也被应用到船舶轨迹聚类[2-3]，得到船舶运动典型轨迹，对船舶的行为和自动识别系统进行研究。文献[6]和文献[18]将DBSCAN应用到隐私保护中，以降低数据外包聚类运算过程中存在的隐私泄露风险。除此之外，DBSCAN还被应用到离群检测[4-5]、电力负荷预测[19]、城市时空数据挖掘[20]等。

2 RAPIDS平台

2.1 RAPIDS概述

RAPIDS是英伟达公司推出的一款针对数据科学和机器学习的GPU加速平台，为研究学者提供在GPU上执行端到端的数据分析和机器学习的管道。RAPIDS依赖CUDA原语进行底层计算优化，但为用户提供了友好的Python接口利用GPU并行性和高带宽内存速度。除此之外，RAPIDS还支持数据预处理任务，利用DataFrame接口与各种机器学习算法集成以实现端到端的管道加速，而不需要关心数据输入输出所需要的序列化工作。RAPIDS还支持多节点、多GPU部署，从而能够极大地加速处理和训练大规模数据。

RAPIDS平台有4个特点：(1)轻松集成，开发者能够以最少的代码修改量，将Python数据分析代码轻松集成到RAPIDS平台；(2)模型精度高，通过对模型进行快速地迭代和调整，或者不断地部署模型来提高机器学习模型的准确性；(3)降低训练时间，通过近乎交互式的方式使用户尽早调整模型，以降低模型的训练时间；(4)平台开源，RAPIDS基于Apache Arrow构建，由英伟达维护，具有可定制、可扩展、可互操作特性。

2.2 RAPIDS架构

RAPIDS平台架构如图1所示。RAPIDS平台架构分3层，最底层是基于Apache Arrow内存计算的数据结构层；中间层是包含用于加速数据处理和分析库cuDF、机器学习库cuML、图分析库cuGraph、深度学习库cuDNN和可视化库RTX；最上层是基于中间层的数据应用

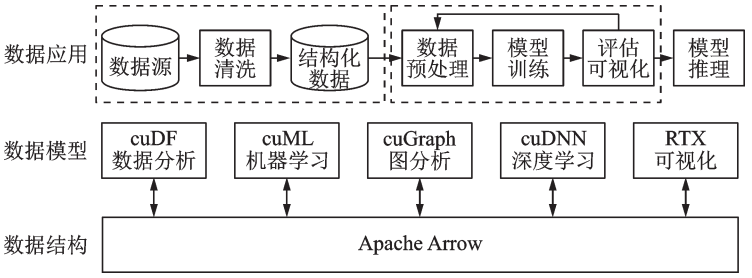


图1 RAPIDS平台架构图

Fig.1 RAPIDS architecture diagram

层,包括数据清洗、模型训练、模型推理。下面主要对中间层包含的工具库进行介绍。

(1)cuDF是一个基于Python的GPU DataFrame库,用于执行数据加载、连接、聚合和过滤等操作,为机器学习模型训练做准备。

(2)cuML是为研究人员提供在GPU上运行传统的结构化ML任务,而无需深入了解GPU编程的细节。通常情况下,cuML的Python接口与scikit-learn的接口相匹配。对于大规模数据,基于GPU的模型训练可比CPU快10~50倍。

(3)cuGraph是GPU加速图算法的集合,用于处理GPU DataFrames中的数据。cuGraph让研究人员注重图的分析,而不用关心技术或框架。cuGraph允许在cuDF中的数据清洗任务和cuML中的机器学习任务之间无缝传递数据。

(4)cuDNN是GPU加速的用于深度神经网络的原语库。cuDNN为标准例程提供了高度优化的实现,比如向前和向后卷积、池化、规范化和激活层。cuDNN使研究人员专注于训练神经网络,而不必关心底层GPU性能调整上。cuDNN加速深度学习框架包括Caffe2、Keras、MxNet、PyTorch和TensorFlow等。

(5)RTX可视化使研究人员能够创建AI增强型应用。智能图像处理、自动处理重复作业和计算密集型流程优化有助节约时间和资源,从而加快机器学习模型应用的速度。

3 无参DBSCAN算法

3.1 参数*MinPts*选择

首先,给出DBSCAN中数据对象局部密度的定义

$$\rho(x)=\left|N_{Eps}(x)\right| \tag{1}$$

式中:ρ为数据对象x的函数映射,ρ的取值范围是$[0, n-1]$,n表示数据对象x所在数据集中数据对象的数目;$\left|N_{Eps}(x)\right|$表示数据对象x在Eps邻域内的邻居数目。下面给出数据集中核心对象的定义

$$\rho(x)\geqslant MinPts \tag{2}$$

由式(2)可知,在已知数据对象的局部密度时,数据集中的核心对象由参数$MinPts$决定。如果数据对象x的局部密度大于等于$MinPts$,则x为核心对象。

对于DBSCAN算法,在不考虑绝对局部密度时(即不考虑参数Eps的值),数据集中大部分数据对象都应该是核心对象,只有少部分是边缘对象和噪声。这决定了设置参数$MinPts$的值不能过大,也不能偏小。因此,在不考虑参数Eps值时,先确定参数$MinPts$的值,使大部分对象都拥有至少$MinPts$个邻居。为了确定参数$MinPts$的值,文本采用自然最近邻概念,利用自然特征值替代$MinPts$的值。

定义1(自然最近邻) 给出包含n个对象的数据集$S=\{x_1, x_2, \cdots, x_{n-1}, x_n\}$,对于数据对象$x_i, x_i\in S$,若有数据对象$x_j, x_j\in S, x_j\neq x_i$的最近邻路径经历$x_i$,且当$S$中最离群的数据对象都有最近邻路径到达时,则称$x_j$为$x_i$的自然最近邻。

从自然最近邻的定义可知,求解自然最近邻时,最近邻路径是指一个对象是另外一个对象的k最近邻,但参数k的值不需要指定,通过k的值从1开始动态的增加,直到数据集中任何一个对象都至少是另外一个对象的近邻。因此,自然最近邻不需要指定最近邻个数或者邻域半径,它是一种无尺度的最近邻概念。在实际求解中,数据集中往往存在离群点,导致k的值很大,为了避免这种情况发生,核心思想是设置计算的终止条件,整个计算过程是对给定数据集的一个自适应过程,当迭代计算终止时,得到数据集中每个对象的自然最近邻。自然最近邻数目是一种量化的度量方法,能够反映数据集疏密分布情况。自然最近邻与密度结合被成功应用到聚类和离群检测中[21-22]。下面给出自然特征值的定义:

定义2(自然特征值) 给出包含n个对象的数据集$S=\{x_1, x_2, \cdots, x_{n-1}, x_n\}$,对于数据对象

$x_i, x_i \in S$，若有数据对象 $x_j, x_j \in S, x_j \neq x_i$ 的 sup_k 最近邻路径经历 x_i，且 sup_k 恰好满足 S 中最离群的数据对象都有最近邻路径到达时，则 sup_k 为自然特征值。

$$sup_k = \min\{k | \forall x \in S, \exists y \in S, x \neq y, \text{s.t.} x \in NN_k(y)\} \tag{3}$$

式(3)给出了自然特征值的形式化定义，sup_k 为满足 S 中最离群的数据对象都有 k 最近邻路径到达时的最小 k 值。根据式(3)中的定义，当数据集中包含离群点时，则为最离群的数据对象搜索最近邻路径时，sup_k 的值会不断增加。为了降低求解 sup_k 时间复杂度，当离群点的数量连续几次不再变化时，则停止搜索[22]。

因为 sup_k 表示数据集中所有数据对象的平均自然邻居个数，数据集中大部分数据对象都拥有 sup_k 个自然最近邻，在不考虑噪声和局部绝对密度时，数据集中大部分数据对象都是核心对象，所以将 DBSCAN 中的参数 $MinPts$ 的值设置为 sup_k。

$$MinPts = sup_k \tag{4}$$

3.2 参数 *Eps* 选择

图2给出了数据集自然最近邻数目分布(图2(b))以及自然邻居个数为 sup_k 的数据对象的分布(图2(a))。由图2(a)可以看出，自然邻居个数为 sup_k 的数据对象(标识为红色)几乎遍布了数据集的每个区域，也就是说覆盖了数据集中的不同密度区域，将满足这种特性的数据对象定义如下。

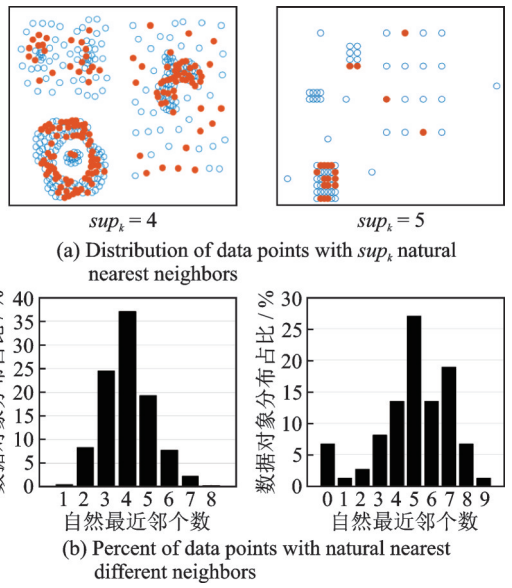


图2 自然最近邻分布

Fig.2 Distribution of natural nearest neighbors

定义3(自然特征集) 给出包含 n 个对象的数据集 $S = \{x_1, x_2, \cdots, x_{n-1}, x_n\}$，$sup_k$ 为数据集 S 的自然特征值，对于数据对象 $x_i, x_i \in S$，若 x_i 的自然最近邻个数为 sup_k，则 x_i 为自然特征对象，由自然特征对象组成的集合为自然特征集，记为 S_{sup_k}。

$$S_{sup_k} = \left\{ x_i | x_i \in S, \left| \text{RNN}(x_i) \right| = sup_k \right\} \tag{5}$$

为了降低变密度对DBSCAN的影响，一个直观的办法是获取数据集中不同密度区域的 Eps 值。由于 S_{sup_k} 中的数据对象的分布覆盖了数据集中的不同密度区域，因此，能够利用 S_{sup_k} 中的数据对象获取数据集中不同密度区域的 Eps 值。下面给出选择 Eps 值的3种策略。

策略1(自然特征集 *Eps* 均值) 给出包含 n 个对象的数据集 $S = \{x_1, x_2, \cdots, x_{n-1}, x_n\}$，$sup_k$ 为数据集 S 的自然特征值，S_{sup_k} 为 S 的自然特征集，数据集 S 的自然特征集 Eps 均值定义为

$$\text{mean}(S_{sup_k}) = \frac{1}{|S_{sup_k}| sup_k} \sum_{x_i \in S_{sup_k}} \sum_{x_j \in \text{RNN}(x_i)} d(x_i, x_j) \tag{6}$$

策略2(自然特征集 *Eps* 最大值) 给出包含 n 个对象的数据集 $S = \{x_1, x_2, \cdots, x_{n-1}, x_n\}$，$sup_k$ 为数据集 S 的自然特征值，S_{sup_k} 为 S 的自然特征集，数据集 S 的自然特征集 Eps 最大值定义为

$$\max(S_{sup_k}) = \max_{x_i \in S_{sup_k}} \{ \min_{x_j \in \text{RNN}(x_i)} (d(x_i, x_j)) \} \tag{7}$$

策略3(自然特征集 *Eps* 最小值) 给出包含 n 个对象的数据集 $S = \{x_1, x_2, \cdots, x_{n-1}, x_n\}$，$sup_k$ 为数据集 S 的自然特征值，S_{sup_k} 为 S 的自然特征集，数据集 S 的自然特征集 Eps 最小值定义为

$$\min(S_{sup_k}) = \min_{x_i \in S_{sup_k}} \{ \max_{x_j \in \text{RNN}(x_i)} (d(x_i, x_j)) \} \tag{8}$$

在式(6~8)中,x_i表示自然特征集中的数据对象,x_j表示x_i的自然最近邻,$d(x_i,x_j)$表示计算x_i和x_j之间的距离,文中采用欧氏距离。$\min(S_{sup_k})$表示数据集密集区域自然特征对象sup_k自然邻域的最大半径中的最小值,$\max(S_{sup_k})$表示数据集稀疏区域自然特征对象sup_k自然邻域的最小半径中的最大值,$\mathrm{mean}(S_{sup_k})$则表示整个自然特征集中数据对象sup_k自然邻域的平均半径。

对于图2中的自然特征集(红色标识的数据对象),图3给出了3种策略选择*Eps*值($\mathrm{mean}(S_{sup_k})$,$\min(S_{sup_k})$,$\max(S_{sup_k})$)的分布情况。从图3可以看出,左侧自然特征集的距离分布$\min(S_{sup_k})=0.474\,3$,$\mathrm{mean}(S_{sup_k})=0.918\,4$,$\max(S_{sup_k})=2.400\,5$)要小于右侧自然特征集的距离分布($\min(S_{sup_k})=1.414\,2$,$\mathrm{mean}(S_{sup_k})=2.516\,9$,$\max(S_{sup_k})=5$)。同时也说明右侧数据集的密度分布差异要大于左侧数据集的密度分布差异。当然,这样的密度分布差异也会受到噪声点的影响。在计算sup_k时,可以考虑将噪声点移除再计算sup_k。

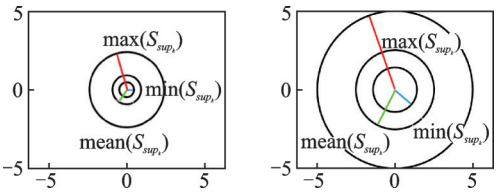


图3 3种选择*Eps*值策略

Fig.3 Three strategies for selecting the value of *Eps*

3.3 无参DBSCAN算法

基于RAPIDS的无参DBSCAN算法如算法1所示。在算法1中,共有5个步骤。第1步,计算数据集的自然特征值sup_k,该步骤主要是查找每个数据对象的自然邻居以及统计自然最近邻数目,时间复杂度为$O(n\lg n+sup_k n)$。第2步,根据sup_k值,查找满足自然最近邻数目为sup_k的数据对象,也就是计算自然特征集,时间复杂度为$O(n)$。第3步,利用自然特征集根据式(6~8)计算3种策略下的自然特征集*Eps*值,时间复杂度为$O(m\lg m)$,$m\ll n$。第4步,设置参数*MinPts*和*Eps*的值,然后运行RAPIDS平台DBSCAN算法进行聚类。第5步,返回聚类结果。算法1总的时间复杂度为$O(n\lg n)$。

算法1 Rap_freeDBSCAN()

输入:数据集S;

输出:聚类结果$C=\{C_1,C_2,\cdots,C_m\}$。

开始

1. 计算数据集S的自然特征值sup_k;
2. 根据sup_k计算自然特征集S_{sup_k};
3. 根据S_{sup_k}和自然最近邻域,分别计算$\mathrm{mean}(S_{sup_k})$、$\min(S_{sup_k})$、$\max(S_{sup_k})$;
4. 设置*MinPts*和*Eps*,运行基于RAPIDS的DBSCAN算法;
5. 返回聚类结果$C=\{C_1,C_2,\cdots,C_m\}$;

结束

4 实验结果与分析

4.1 实验设置

实验采用本地和公有云两种实验环境,本地实验环境用来进行算法有效性实验,采用MATLAB2019a实现,公有云实验环境用来进行算法性能实验,采用Python3.6实现。两种实验环境的详细配置信息如表1所示。

用于有效性实验的数据集如表2所示。采用ARI[23]和NMI[24]作为聚类结果评价指标。用于性能实验的数据集如表3所示。实验数据集包含人工数据集和真实数据集[25]。

表1 实验环境设置

Table 1 Settings of experiments

实验环境	CPU	主频/GHz	内存/GB	GPU	实例规格
本地	i7 8核	3	16		
公有云	Intel Xeon(Skylake) Platinum 8163 4核	2.5	15	1*NVIDIA T4 16 GB 8.1TFLOPS	ecs.gn6i-c4g1.xlarge

4.2 算法有效性实验结果与分析

用于算法聚类有效性实验的数据集有14个(表2),其中9个数据集带有类别标签,通过聚类评价指标ARI和NMI进行实验结果分析,其他5个数据集通过聚类结果可视化进行实验结果分析,如表4所示。

在表4中,给出了数据集名称、参数*MinPts*和*Eps*的取值,NC表示聚类结果的数目(噪声点也计算为一类)。从表4可以看出,本文算法在$Eps=\text{mean}(S_{sup_k})$或者$Eps=\max(S_{sup_k})$时,能够识别出正确的聚类数目,除了数据集t4比实际类别数目多出一个类别。当$Eps=\min(S_{sup_k})$时,由于*Eps*值相对较小,*MinPts*值不变,DBSCAN会将数据集划分成更多的类簇(如x4和t4)或者将数据集中更多的数据对象划分为噪声点(如O_1、data_uc_cv_n、data_uc_n和Compound)。$Eps=\min(S_{sup_k})$时,能够识别数据集中密集区域的类簇,而$Eps=\max(S_{sup_k})$时,能够识别数据集中稀疏区域的类簇,也可以将二者进行结合识别不同密度的类簇。整体来看,当$Eps=\text{mean}(S_{sup_k})$时,数据集获得比较好的聚类结果。

表5给出了算法聚类有效性的ARI和NMI结果,由表4可知,8个数据集在$Eps=\text{mean}(S_{sup_k})$或者$Eps=\max(S_{sup_k})$时取得了最好的聚类结果。而数据集R15在$Eps=\min(S_{sup_k})$时取得了最好的聚类结果(ARI=0.86,NMI=0.91),原因是数据集R15中数据对象的分布非常密集,只需很小的*Eps*就能满足邻域内至少包含*MinPts*个邻居的要求。从ARI指标来看,数据集Jain取得了最高的0.91,有5个数据集的ARI大于等于0.81,数据集Iris和Seeds的ARI分别是0.67和0.36,数据集Digit的ARI为0.38。再从NMI指标来看,数据集R15取得了最高的0.91,有4个数据集的NMI至少取得了0.82,数据集Jain的NMI为0.79,数据集Iris、Seeds和Digit分别取得了0.71、0.44和0.63。从ARI和NMI的均值来看,有6个数据集的均值大于等于0.835,最高为0.885,数据集Iris、Seeds和Digit的均值分别为0.69、0.40和0.51。

表2 有效性实验数据集信息

Table 2 Information of datasets for effectiveness experiment

数据集	样本数	维度	类别数
Aggregation	788	2	7
Compound	399	2	6
Flame	240	2	2
Jain	373	2	2
Pathbased	300	2	3
R15	600	2	15
Iris	150	4	3
Seeds	199	7	3
O_1	74	2	4
data_uc_cv_n	127	2	3
data_uc_n	192	2	3
x4	610	2	3
t4	8 000	2	6
Digits	3 823	64	10

表3 性能实验数据集信息

Table 3 Information of datasets for efficiency experiment

数据集	样本数	维度
Avila	20 867	10
Worms	105 600	2
3D_spatial_network	434 874	3
Household_power_consumption	350 000	7

表4 算法聚类有效性实验结果

Table 4 Experimental results of clustering effectiveness validation

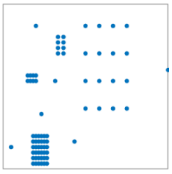
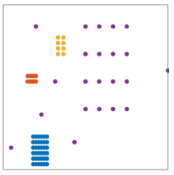
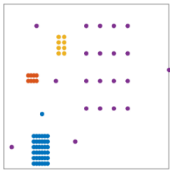
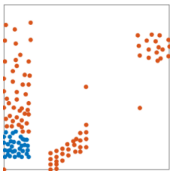
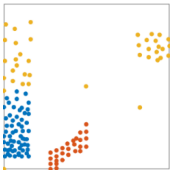
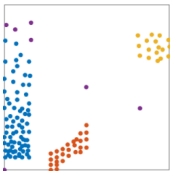
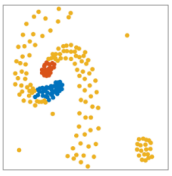
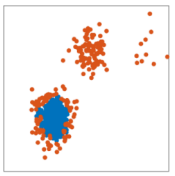
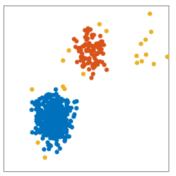
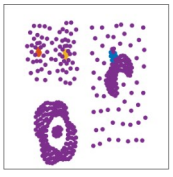
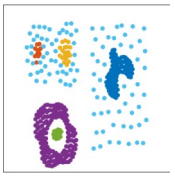
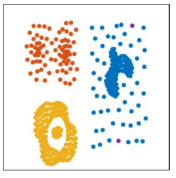
数据集	*MinPts*	*Eps*		
		min(S_{sup_k})=1.414 2	mean(S_{sup_k})=2.516 9	max(S_{sup_k})=5
O_1	sup_k=5	 NC=1	 NC=4	 NC =4
		min(S_{sup_k})=0.232 7	mean(S_{sup_k})=0.367 7	max(S_{sup_k})=0.576 4
data_uc_cv_n	sup_k=6	 NC =2	 NC =3	 NC =4
		min(S_{sup_k})=0.199 2	mean(S_{sup_k})=0.389 3	max(S_{sup_k})=0.615 8
data_uc_n	sup_k=5	 NC =3	 NC =4	 NC =3
		min(S_{sup_k})=0.135 1	mean(S_{sup_k})=0.237 6	max(S_{sup_k})=0.588 4
x4	sup_k=6	 NC =6	 NC =2	 NC =3
		min(S_{sup_k})=4.219 5	mean(S_{sup_k})=4.924 0	max(S_{sup_k})=13.832 3
t4	sup_k=11	 NC =18	 NC =8	 NC =2
		min(S_{sup_k})=0.474 3	min(S_{sup_k})=0.918 4	min(S_{sup_k})=2.400 5
Compound	sup_k=4	 NC = 4	 NC = 6	 NC = 4

表5 算法聚类有效性实验ARI和NMI结果

Table 5 ARI and NMI results of clustering effectiveness validation

数据集	*MinPts*	*Eps*	聚类数目	评价指标		
				ARI	NMI	均值
Aggregation	sup_k=5	min(S_{sup_k})=0.474 3	NC=3	0.10	0.37	0.235
		mean(S_{sup_k})=0.918 4	NC=6	0.58	0.68	0.63
		max(S_{sup_k})=2.400 5	NC=7	**0.81**	**0.87**	**0.84**
Compound	sup_k=4	min(S_{sup_k})=0.570 1	NC=4	0.03	0.11	0.07
		mean(S_{sup_k})=0.701 7	NC=6	**0.87**	**0.87**	**0.87**
		max(S_{sup_k})=0.813 9	NC=4	0.74	0.80	0.77
Flame	sup_k=4	min(S_{sup_k})=0.570 1	NC=3	0.07	0.21	0.14
		mean(S_{sup_k})=0.701 7	NC=6	0.23	0.41	0.32
		max(S_{sup_k})=0.813 9	NC=3	**0.88**	**0.82**	**0.85**
Jain	sup_k=5	min(S_{sup_k})=0.583 1	NC=6	—	0.10	—
		mean(S_{sup_k})=0.820 4	NC=9	0.12	0.33	0.225
		max(S_{sup_k})=1.612 5	NC=5	**0.91**	**0.79**	**0.85**
Pathbased	sup_k=7	min(S_{sup_k})=0.992 5	NC=3	0.45	0.56	0.505
		mean(S_{sup_k})=1.377 4	NC=3	**0.84**	**0.83**	**0.835**
		max(S_{sup_k})=1.520 7	NC=3	0.65	0.69	0.67
R15	sup_k=10	min(S_{sup_k})=0.279 4	NC=16	**0.86**	**0.91**	**0.885**
		mean(S_{sup_k})=0.243 5	NC=14	0.57	0.83	0.70
		max(S_{sup_k})=0.222 6	NC=13	0.34	0.74	0.54
Iris	sup_k=5	min(S_{sup_k})=0.2	NC=3	0.06	0.22	0.14
		mean(S_{sup_k})=0.351 1	NC=3	**0.67**	**0.71**	**0.69**
		max(S_{sup_k})=0.346 4	NC=4	0.49	0.59	0.54
Seeds	sup_k=5	min(S_{sup_k})=0.487 4	NC=5	0.18	0.38	0.28
		mean(S_{sup_k})=0.656 8	NC=4	0.30	0.42	0.36
		max(S_{sup_k})=0.756 6	NC=3	**0.36**	**0.44**	**0.40**
Digit	sup_k=19	min(S_{sup_k})=21.91	NC=4	0.11	0.40	0.26
		mean(S_{sup_k})=21.57	NC=4	0.11	0.40	0.26
		max(S_{sup_k})=20.81	NC=6	**0.38**	**0.63**	**0.51**

表6给出了本文算法与K-means,DPC,DBSCAN和SC算法ARI和NMI的对比实验结果。K-means、DPC、DBSCAN和SC都在R15数据集上取得了最好的聚类结果,因为R15数据集服从规则高斯分布,数据相对集中,没有噪声点的影响。DPC和DBSCAN在Aggregation数据集上取得了最高的ARI和NMI,SC在Jain和Digit上取得了最好的聚类效果。本文算法和DBSCAN在数据集Compound

表6 聚类算法对比实验ARI和NMI结果

Table 6 ARI and NMI results of compared clustering algorithms

数据集	K-means		DPC		DBSCAN		SC		本文算法	
	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI
Aggregation	0.77	0.88	0.99	0.99	0.99	0.99	0.96	0.96	0.81	0.87
Compound	0.78	0.80	0.64	0.84	0.97	0.94	0.62	0.83	0.87	0.87
Flame	0.51	0.48	1.0	1.0	0.94	0.86	0.95	0.89	0.88	0.82
Jain	0.33	0.37	0.71	0.51	0.94	0.86	1.0	1.0	0.91	0.79
Pathbased	0.46	0.55	0.47	0.54	0.65	0.70	0.53	0.60	0.84	0.83
R15	0.99	0.99	0.99	0.99	0.97	0.98	0.99	0.99	0.86	0.91
Iris	0.73	0.75	0.76	0.81	0.73	0.70	0.88	0.87	0.67	0.71
Seeds	0.74	0.72	0.75	0.71	0.53	0.49	0.73	0.71	0.36	0.44
Digits	0.74	0.78	0.46	0.72	0.77	0.68	0.93	0.93	0.38	0.63
均值	0.67	0.70	0.75	0.79	0.83	0.80	0.84	0.86	0.73	0.77

和Pathbased上取得了最好的聚类结果。从整体实验的ARI和NMI均值来看,SC取得了最好的聚类结果,但SC需要预先指定聚类数目,DBSCAN聚类效果仅次于SC,但DBSCAN需要不断调整参数*Eps*和*MinPts*才能找出较优的结果。本文算法优于K-means,取得了与DPC相当的聚类结果,ARI和NMI比DPC均低0.02。本文算法的优势是能够在对精度要求不是非常高的自动化的场景中进行应用,同时能够达到较好的聚类效果。

4.3 算法性能实验结果与分析

在4个数据集上对本文算法进行了性能实验,最小的样本数为20 867,最大的样本数为434 874,数据集的维度最小是2维,最大是10维。算法的性能实验是在公有云平台进行,见表1,分为无RAPIDS和使用RAPIDS加速两种环境,算法的性能实验运行时间(Wall time)结果如表7所示。

表7 算法性能实验运行时间结果

Table 7 Runtime results of efficiency validation

数据集	运行时间Wall time/s		加速比
	无RAPIDS加速	有RAPIDS加速	
Avila	3.76	0.51	7.37
Worms	11.5	2.74	4.20
3D_spatial_network	37.2	53.7	0.69
Household_power_consumption	104.3	32.8	3.18

在表7中,数据集Avila获得了最高的加速比(无RAPIDS运行时间/有RAPIDS运行时间)7.37,数据集Worms和Household_power_consumption分别取得了4.20和3.18的加速比。而对于数据集3D_spatial_network取得了小于1的加速比,也就是说采用RAPIDS比不采用RAPIDS的运行时间要多。出现这种情况的原因是,系统在CPU和GPU之间传递数据花费较多的时间。通过运行时间分析,数据集3D_spatial_network采用RAPIDS的系统时间(System time)几乎是不采用RAPIDS的5倍,而算法本身所需的时间几乎没有变化。

RAPIDS平台加速本质是GPU计算,而GPU计算需要考虑GPU内存和带宽两个指标,这两个指

标会影响CPU和GPU之间的数据传输速度。当数据集的维度比较高时,RAPIDS能够获得很好的加速比(如Avila和Household_power_consumption),当数据集的维度比较低时,样本量不是非常大时,RAPIDS也会取得不错的加速比(如Worms),而随着数据集样本量的增加,RAPIDS的加速性能出现下降,甚至加速比会小于1(如3D_spatial_network)。整体来看,当数据集维度比较高时,能够在GPU实现多核并行计算,会获得较好的加速比。

除了算法的性能实验外,对表3中数据集进行计算DBSCAN参数*MinPts*和*Eps*的性能实验,实验采用本地实验环境,表8给出了计算参数时间的实验结果。从表8可知,对于数据集Avila,计算参数*Eps*需要14 s,对于数据集3D_spatial_network,计算参数*Eps*需要3 353 s。3D_spatial_network数据量是Avila的约21倍,而计算参数*Eps*所需时间约240倍。对于4个数据集,计算参数*MinPts*所需时间均小于1 s。但在求k最近邻时,需要花费更多的时间,最大的数据集3D_spatial_network需要1 486 s,最小数据集Avila需要15 s。整体来看,随着数据量的增加,计算参数*Eps*需要更多的时间。

表8 计算参数实验运行时间结果

Table 8 Runtime results of computing adaptive parameters

数据集	计算k最近邻时间/s	计算*MinPts*时间/s	计算*Eps*时间/s
Avila	15	0.06	14
Worms	52	0.15	203
3D_spatial_network	1 486	0.77	3 353
Household_power_consumption	1 342	0.76	2 092

5 结束语

本文提出了一种确定DBSCAN中*MinPts*和*Eps*参数值的方法,并且采用RAPIDS平台对DBSCAN算法进行加速。本文方法利用自然最近邻获得数据集的自然特征值作为*MinPts*。然后,根据自然特征值计算自然特征集,基于自然特征集计算*Eps*候选值,最后基于RAPIDS平台运行无参DBSCAN算法。综合实验结果表明,本文提出的确定DBSCAN算法参数值的方法是有效的,能够为进一步优化DBSCAN算法找到更优的全局的*Eps*参数值提供优化方向,并且采用RAPIDS能够提升DBSCAN性能。下一步工作将继续研究自然特征集的特性,获取数据集更多的知识。在算法性能方面将优化计算参数的性能,并探索RAPIDS的Dask框架进行多GPU高性能计算。

参考文献:

[1] 陆海凌,李洋,林赟,等.基于帧间DBSCAN聚类的毫米波雷达交通多目标跟踪方法[J].信号处理,2021,37(11): 2115-2124.
LU Hailing, LI Yang, LIN Yun, et al. Traffic multi-object tracking method of millimeter wave radar based on inter-frame DBSCAN clustering[J]. Journal of Signal Processing, 2021, 37(11): 2115-2124.

[2] 郭乃琨,陈明剑,陈锐.一种顾及时间特征的船舶轨迹DBSCAN聚类算法[J].测绘工程,2021,30(3): 51-58.
GUO Naikun, CHEN Mingjian, CHEN Rui. A DBSCAN clustering algorithm of ship trajectory considering time characteristics[J]. Engineering of Surveying and Mapping, 2021, 30(3): 51-58.

[3] 江玉玲,熊振南,唐基宏.基于轨迹段DBSCAN的船舶轨迹聚类算法[J].中国航海,2019,42(3): 1-5.
JIANG Yuling, XIONG Zhennan, TANG Jihong. Ship trajectory clustering algorithm based on DBSCAN[J]. Navigation of China, 2019, 42(3): 1-5.

[4] 黄剑柔,王茜,蔡星娟,等.一种多目标自适应DBSCAN离群点检测算法[J].小型微型计算机系统,2022,43(4): 702-706.
HUANG Jianrou, WANG Qian, CAI Xingjuan, et al. Multi-objective adaptive DBSCAN outlier detection algorithm[J]. Journal of Chinese Computer Systems, 2022, 43(4): 702-706.

[5] 王跃飞,于炯,苏国平,等.ODIC-DBSCAN:一种新的簇内孤立点分析算法[J].自动化学报,2019,45(11): 2107-2127.
WANG Yuefei, YU Jiong, SU Guoping, et al. ODIC-DBSCAN: A new analytical algorithm for inliers[J]. Acta Automatica Sinia,2019,45(11): 2107-2127.

[6] 贾春福,李瑞琪,王雅飞.基于同态加密的DBSCAN聚类隐私保护方案[J].通信学报,2021,42(2): 1-11.
JIA Chunfu, LI Ruiqi, WANG Yafei. Privacy protection scheme of DBSCAN clustering based on homomorphic encryption[J]. Journal on Communication,2021,42(2): 1-11.

[7] UNCU O, GRUVER W A, KOTAK D B, et al. GRIDBSCAN: GRId density-based spatial clustering of applications with noise[C]//Proceedings of International Conference on Systems.[S.l.]: IEEE, 2007.

[8] RAM A, SHARMA A, JALAL A S, et al. An enhanced density based spatial clustering of applications with noise[C]//Proceedings of Advance Computing Conference. [S.l.]: IEEE, 2009.

[9] WANG S, LIU Y, SHEN B. VDBSCAN: Varied density based spatial clustering of applications with noise[C]//Proceedings of International Conference on Service Systems and Service Management.[S.l.]: IEEE, 2007.

[10] CHOWDHURY A, MOLLAH M E, RAHMAN M A. An efficient method for subjectively choosing parameter "k" automatically in VDBSCAN (varied density based spatial clustering of applications with noise) algorithm[C]//Proceedings of International Conference on Computer and Automation Engineering.[S.l.]: IEEE, 2010: 38-41.

[11] WANG W T, WU Y L, TANG C Y, et al. Adaptive density-based spatial clustering of applications with noise (DBSCAN) according to data[C]//Proceedings of 2015 International Conference on Machine Learning and Cybernetics (ICMLC). [S.l.]: IEEE, 2015.

[12] 王治和,曹旭琰,杜辉.一种优化初始点与自适应半径的密度聚类算法[J].计算机工程,2022,48(1): 51-59.
WANG Zhihe, CAO Xuyan, DU Hui. A density clustering algorithm with optimized initial points and adaptive radius[J]. Computer Engineering,2022,48(1): 51-59.

[13] WANG L, WANG H, ZHOU W, et al. A novel adaptive density-based spatial clustering of application with noise based on bird swarm optimization algorithm[J]. Computer Communications, 2021, 174: 205-214.

[14] CHEN Y, ZHOU L, PEI S, et al. KNN-BLOCK DBSCAN: Fast clustering for large-scale data[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 99: 1-15.

[15] LUO G, LUO X, GOOCH T F, et al. A Parallel DBSCAN Algorithm based on spark[C]//Proceedings of IEEE International Conferences on Big Data & Cloud Computing.[S.l.]: IEEE, 2016.

[16] SENG C L, MACDONALD B A, PARSONS M, et al. Accelerated superpixel image segmentation with a parallelized DBSCAN algorithm[J]. Journal of Real-Time Image Processing,2021(11): 1-16.

[17] 邓定胜.一种改进的DBSCAN算法在Spark平台上的应用[J].计算机科学,2020,47(S2): 425-429,443.
DENG Dingsheng. Application of improved DBSCAN algorithm on spark platform[J]. Computer Science, 2020, 47(S2): 425-429,443.

[18] 吴伟民,黄焕坤.基于差分隐私保护的DP-DBScan聚类算法研究[J].计算机工程与科学,2015,37(4): 830-834.
WU Weimin, HUANG Huankun. A DP-DBScan clustering algorithm based on differential privacy preserving[J]. Computer Engineering and Science,2015,37(4): 830-834.

[19] 张金金,张倩,马愿,等.基于改进的随机森林和密度聚类的短期负荷频域预测方法[J].控制理论与应用,2020,37(10): 2257-2265.
ZHANG Jinjin, ZHANG Qian, MA Yuan, et al. Short-term load frequency domain prediction method based on improved random forest and density-based spatial clustering of applications with noise[J]. Control Theory and Applications, 2020, 37(10): 2257-2265.

[20] 蔡莉,李英姿,江芳,等.面向城市热点区域的不平衡数据聚类挖掘研究[J].计算机科学,2019,46(8): 16-22.
CAI Li, LI Yingzi, JIANG Fang, et al. Study on clustering mining of imbalanced data fusion towards urban hotspots[J]. Computer Science,2019,46(8): 16-22.

[21] 汤鑫瑶,张正军,储杰,等.基于自然最近邻的密度峰值聚类算法[J].计算机科学,2021,48(3): 151-157.
TANG Xinyao, ZHANG Zhengjun, CHU Jie, et al. Density peaks clustering algorithm based on natural nearest neighbor[J]. Computer Science,2021,48(3): 151-157.

[22] 李士果,卢建云,邓剑勋.基于自然最近邻的离群检测方法研究[J].智能计算机与应用,2019,9(4): 40-44,50.
LI Shiguo, LU Jianyun, DENG Jianxun. Outlier detection methods based on natural nearest neighbors[J]. Intelligent Computer and Applications,2019,9(4): 40-44,50.

[23] MCCOMB C. Computes the rand index to describe the agreement between two partitions[EB/OL]. https://github.com/cmc-comb/rand_index,2021.

[24] CHEN M. Normalized mutual information[EB/OL]. https://www.mathworks.com/matlabcentral/fileexchange/29047-normalized-mutual-information,2023.

[25] DUA D,GRAFF C.UCI machine learning repository[EB/OL].http://archive.ics.uci.edu/ml,2019.

作者简介:

卢建云(1982-),男,博士,研究方向:数据挖掘、机器学习、图神经网络,E-mail: lujianyun@uestc.edu.cn。

邵俊明(1982-),**通信作者**,男,教授,研究方向:数据挖掘、深度学习,E-mail: junmshao@ uestc.edu.cn。

张蔚(1977-),女,高级工程师,研究方向:信号与信息处理、目标检测,E-mail: 2365454225@qq.com。

(编辑:夏道家)