基于 MapReduce 的改进 DBSCAN 算法

赖丽萍 聂瑞华 汪疆平 黄家鸿

(华南师范大学计算机学院 广州 510631)

摘 要 针对聚类算法 DBSCAN 中存在的 Eps 参数问题和处理海量数据集时的效率问题,提出 OPDBSCAN 算法。OPDBSCAN 算法通过交叠分区获取局部 Eps 以降低全局 Eps 参数对聚类质量的影响,并结合 MapReduce 并行框架提高算法效率。实验结果表明,OPDBSCAN 算法的效率和聚类质量都高于原 DBSCAN 算法。

关键词 OPDBSCAN,MapReduce,Eps,K-dist 图,交叠分区

中图法分类号 TP30 **文献标识码** A

Improved DBSCAN Algorithm Based on MapReduce

LAI Li-ping NIE Rui-hua WANG Jiang-ping HUANG Jia-hong

(School of Computer, South China Normal University, Guangzhou 510631, China)

Abstract Aimed at solving DBSCAN's problems of the Eps parameters and the efficiency of processing of massive data sets, the article put forward a new algorithm called OPDBSCAN. It uses overlapping partitions to get a local Eps for reducing the effect of global Eps, then uses MapReduce to cluster in parallel to improve the efficiency. At last, the experiment shows that the OPDBSCAN can cluster faster and better.

Keywords OPDBSCAN, MapReduce, Eps, K-dist, Overlap-partion

1 引言

大数据时代,数据爆炸式增长,需要一种快速有效的数据挖掘算法来提取大数据中隐含的、未知的、非平凡的及有潜在应用价值的信息,用以决策支持和趋势预测[1]。但是,传统的串行数据挖掘算法已经无法满足海量数据的处理,而云计算的出现为海量数据的存储和计算提供了一种廉价的解决方案。

Hadoop 平台是云计算的一种开源实现,主要由分布式文件系统 HDFS、并行计算框架 MapReduce 和分布式式数据库组成[2]。其中 MapReduce 具有高并发和吞吐能力,能够为海量数据处理提供保障。所以本文通过搭建 Hadoop 平台,利用 MapReduce[3] 技术提高 DBSCAN 算法聚类效率并优化参数。

2 DBSCAN 算法

2.1 DBSCAN 算法

DBSCAN 算法[4] 于 1996 年由 Ester Martin 等人提出,是一种基于密度的经典聚类算法,能够发现任意形状的簇并过滤数据集中的噪声,该算法被广泛应用在气象云图[5]、图形处理[6] 和城市规划[7] 等领域。

DBSCAN 算法通过用户给定的密度阈值,将空间对象中具有一定密度的区域划分为类。密度阈值由 Eps 和 $minPts$ 两个参数组成,其中 Eps 为半径参数,$minPts$ 为半径邻域内应该包含的对象值。

DBSCAN 算法在接收到用户输入的 Eps、$minPts$ 参数后,开始对数据集执行聚类,具体步骤如下。

(1)在数据集中任意取一个未处理的点 p,统计 p 的 Eps 邻域的 $N_{Eps}(p)$,如果 $N_{Eps}(p)$ 包含的对象个数大于等于 $minPts$,则建立新簇 C,将 $N_{Eps}(p)$ 内所有对象加入簇 C。如果 $N_{Eps}(p)$ 包含的对象个数小于 $minPts$,将 p 暂标记为噪声点(其中符号 $N_{Eps}(p)$ 代表以 p 为中心、Eps 为半径的范围内包括的对象个数)。

(2)对 C 中所有未处理的对象 q,计算其 Eps 邻域 $N_{Eps}(q)$,若 $N_{Eps}(q)$ 包含至少 $minPts$ 个对象,则将 $N_{Eps}(q)$ 中未标记为其他簇的对象加入 C。

(3)重复步骤(2),继续处理 C 中未处理对象,直到没有新的对象加入当前簇 C。

(4)重复步骤(1)—(3),直到所有对象都归入了某个簇或标记未噪声。

2.2 DBSCAN 算法的优缺点

DBSCAN 算法最大的优点是可以发现任意形状的簇,自动确定簇的数量,并不受数据集中的噪音数据影响。

算法主要存在以下缺点:

(1)算法需要逐个查找核心结点并通过迭代确定所有密度可达对象,该过程使得算法时间复杂度达到 $O(n^2)$。如果在聚类之前建立 R* 树索引和一个基于距离的 K-dist 图,算法复杂度依然达到 $O(n\log n)$[8],时间复杂度高成为算法处理

本文受教育部—中移动基金项目(MCM20130651),广州市科技和信息化局基金项目(2014Y2-00006)资助。

赖丽萍(1989—),女,硕士生,主要研究领域为大数据、云计算,E-mail:947293009@qq.com;**聂瑞华**(1963—),男,教授,主要研究领域为大数据、云计算、高性能计算、信息安全等;**汪疆平**(1970—),男,高级工程师,主要研究领域为大数据、云计算、信息安全等;**黄家鸿**(1989—),男,硕士生,主要研究领域为大数据、云计算、数据挖掘。

海量数据的瓶颈。

(2)*Eps* 和 *minPts* 参数需要事先根据经验人为确定，而这本身是难以确定的。所以导致算法对参数敏感，聚类结果也会因为它们的细微变化而产生较大的差异。

(3)使用全局统一的 *Eps* 不能很好地反映数据的分布情况，会影响聚类质量。

(4)DBSCAN 算法处理边界点的归属问题存在争议，目前主要采取谁先发现就归谁的策略，这在一定程度上影响了聚类的精度。

3 DBSCAN 算法改进与分区

3.1 算法改进思想

针对 DBSCAN 算法中存在的缺陷，国内外学者提出了很多的改进算法。例如 AGD-DBSCAN[9] 算法利用数据集特点找到自适应的 *Eps* 和 *minPts* 参数；FDBSCAN[10] 算法则主要是对 DBSCAN 的聚类速度做改进；OPTICS[11] 算法就是通过生成有序队列表优化 DBSCAN 算法的 *Eps* 参数。但是，这些算法的提出都是基于单机模式下优化参数的选择问题或者提高算法运行效率，并没有提出并行模式下针对海量数据集处理进行改进，也没有在保证聚类质量的前提下，对以上缺陷进行改进。由此本文提出 OPDBSCAN 算法，其利用交叠分区获取局部 *Eps* 来解决因参数 *Eps* 的选择困难而导致的聚类质量差的问题；同时通过 MapReduce 使算法并行化，从而达到提高算法处理海量数据的效率的目的。

OPDBSCAN 算法基于 MapReduce 技术实现，所以整个实验过程需要在 Hadoop 环境下进行，具体改进思路如下。

(1)将数据集对象做某一维投影分布特征统计分析，划分为若干个交集不为空的子区域，发送到多个处理器。

(2)Map 处理器接收到分区数据集，建立 K-dist 排序图，确定各个分区的 *Eps* 值，实现局部聚类。

(3)将交叠分区中的数据对象发送到 Reduce 处理器，合并不同分区中的相同数据对象，并获得数据对象的类簇号。

(4)根据 Reduce 阶段得到的结果，获得整个数据集的类簇结构。

3.2 获取分区 *Eps*

DBSCAN 算法通过不断检查核心点和邻域实现聚类，在聚类过程中，数据对象与其邻域外的对象没有直接的联系，所以有利于数据分区和实现数据并行化，在一定程度上数据分区也能解决数据量增大而导致的内存溢出的问题。通过分区摈弃全局统一的 *Eps* 参数，采用局部的 *Eps* 参数实现聚类从而提高聚类质量。

在算法 PDBSCAN[12] 的分区方法中，把整个数据集划分为若干个交集为空的子区域，这种划分方法存在将数据集本身的关系切断的可能，即可能使得一些边界上的类变成噪音，大类变成小类，最后导致分区合并时算法重新扫描边界点的邻域，增加了算法的复杂度。本文提出的交叠分区有利于减小分区对边界对象的影响，同时降低分区合并时算法的复杂度。为了使分区内的数据分布尽可能均匀，基于数据集的分布特性进行分区，这种方式能更方便算法后期的分区聚类合并。

假设有 n 个元素的数据集 D，任意元素 X 有 d 维属性 $(X_1, X_2, X_3, \cdots, X_d)$。对数据集做第 $i(1\leqslant i\leqslant d)$ 维的分布投影。令 $\min(X)$、$\max(X)$ 为数据集的下限和上限，即[$\min(X)$，$\max(X)$]为数据集 D 的分布区间。利用直方图来表征数据的分布情况，图内矩形代表一定区域内的数据对象占总数的比例。画直方图需要先确定组数，组数不能过大或过小，一般组数 m 和数据对象总数 n 满足以下公式：

$$m\approx(n-1)^{\frac{2}{3}}$$

得到组数 m 后，求出每个组的跨度 k。

根据数据集 D 的分布区间[$\min(X)$，$\max(X)$]和组数 m，画出密度直方图，如图 1 所示。

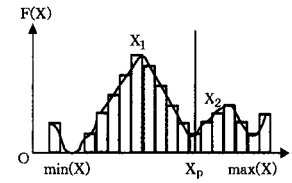


图 1 数据在 X 轴的密度分布

将图 1 中矩形的中点连接成一条曲线，画出直方图的拟合曲线图。引入阈值 λ，当相邻的两个波峰 X_{i-1}，X_i 满足公式 $|X_{i-1}-X_i|\geqslant\lambda$ 时，求出波谷 $(X_{p1}, X_{p2}, X_{p3}, \cdots, X_{pn})$，并将 $\{X_{pi-1}-2k, X_{pi}+2k\}$ 划分为一个分区。为了避免因以不同维度分区而导致的算法缺陷，选择包含交叉边界点最少的那一维投影作为分区依据，或者选择数据集投影最均匀的那一维作为分区。最后将数据集 D 划分为 n 个交叠的数据分区 $\{\min(X), X_{p1}+2k\}$，$\{X_{p1}-2k, X_{p2}+2k\}$，$\cdots$，$\{X_{ps-1}-2k, \max(X)\}$，将分区记作 $A_1, A_2, \cdots, A_s$，分别发送到 s 个处理器。

处理器接收发送过来的分区 A_i，建立基于距离的 K-dist 排序图，用以表示数据集中所有对象的第 k 最近邻距离的排序图。在传统的 DBSCAN 中，为了确定 *Eps*，会先指定 $minPts(k=minPts)$[9,13] 值和一个百分比值(通常为 85%)。用户则根据 K-dist 图曲线，将百分比对象对应的纵坐标定为 *Eps* 值[14]。如图 2 所示，画出 250 个对象在 $k=4$ 的曲线图，将横坐标 85%处即 A 点的 $Dist_{minPts}$ 值设为分区 *Eps*。

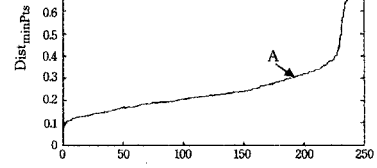


图 2 $minPts=4$ 排序曲线

4 基于 MapReduce 并行 DBSCAN

交叠分区为自动获取局部 *Eps*、减少全局 *Eps* 参数对算法聚类质量的影响和算法并行化提供了条件。将算法 OPDBSCAN 迁移到云计算平台，结合用户自定义的 Map 函数和 Reduce 函数实现并行化。整个并行化过程中，数据块以 $\langle key, value\rangle$ 键值对的形式进行运算。

4.1 Map 过程

Map 处理器得到分区 *Eps* 值后，处理器通过 Map 方法实现局部聚类。首先将分区数据集转换为 $\langle key, value\rangle$ 键值对，*key* 对应 *Pid* 的点对象标识、*value* 对应对象属性值和交叉区域标记 *Bid* 的字符串。调用 Initial DBSCAN() 函数对数据集

进行初始聚类，最后生成中间聚类结果〈*key′*，*value′*〉对应〈*Pid*，*FidCid*+*Core*〉，其中 *Pid* 为点标识，*Fid* 为分区编号，*Cid* 为类簇编号，*Core* 为是否为核心，取值 ture/fasle。

OPDBSCAN 算法的具体 Map 过程如下：

(1)读取分区数据集、*Eps* 值和 *minPts* 值。

(2)对分区数据集执行 DBSCAN 算法。

(3)获得分区聚类结果，对于每个分区对象 *P*，如果 *P* 是交叠区域对象，则输出到 Reduce 处理器，格式为〈*Pid*，*FidCid*+*Core*〉；否则输出到 HDFS 本地文件中，格式为〈*Pid*，*Fid*+*Cid*〉。

(4)Map 阶段结束。

4.2 Reduce 过程

Reduce 处理器接收 Map 产生的中间数据〈*Pid*，*FidCid*+*Core*〉，Reduce 过程主要分为两个部分即 Reduce1 和 Reduce2，首先接收 Map 处理器发送过来的交叠区域的相同对象进行合并；然后根据交叠区域的合并结果，对整个数据集的类簇进行合并。

(1)相同对象合并阶段

该阶段的对象同时存在两个分区中，所以每个对象都有两个角色，具体有 6 情况：核心点与核心点、核心点与边界点、核心点与噪音点、边界点与边界点、边界点与噪音点、噪音点与噪音点。如图 3 所示，图中白色区域为交叠区间。

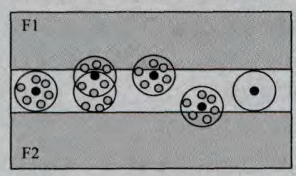


图 3　数据点组合分类

由图 3 可看出，要对两个分区中的类别进行合并，就需要依靠核心点。Reduce1 阶段的主要工作就是合并核心点，从而为下阶段的类别的合并打下基础。具体的 Reduce1 的执行伪代码如下所示。

```
输入：交叠分区数据集〈Pid，FidCid+Core〉
输出：相同点标识的数据对象归类列表
value_list=null //用于存储相同点标识合并的区号编号和类别编号列表
Merge=false;
  for each point p in values do
  if p. core do //如果 p 点是核心点
      Merge=ture
  end if
  if !value_list(p. Cid) and p. Cid !=noise do
      p. Fid and p. Cid combine into value_list
  end if
  end for
      if merge=true do
        Output(key，value_list+ture)
      end if
      else
        Output(key，value_list+false)
      end else
```

(2)合并数据集的类簇阶段

经过相同对象的合并之后，就需要根据该结果对数据集的类别进行合并。从图 3 可知，当点 *p*. *Core* 为 ture 时，*p* 则至少在一个类里为核心。如果 *p*. *Cid*!=nosie，说明 *p* 存在两个类别中，则可以将两个类别合并。因此在 Reduce2 阶段，当 *p*. *Core*=ture，且 *sizeof*(*p*. *Cid*)为 2 时，将两个类合并，该过程具体的伪代码如下所示。

```
输入：Reduce1 的输出
输出：T //用于保存待合并的类别编号
Data=Hdfs. open(Reduce1. Output)
    for each point p in Data do
      if p. core do
          if sizeof(p. Cid=2) do
            combine p. Cid into T//将核心点的类别编号放入 T 链表
          end if
      end if
    end for
for each C1，C2 in T
  if C1∩C2≠Ø
    then combine C1 and C2
    //合并 C1、C2 类的对象
  end if
end for
Return T//返回合并后的类别列表
```

合并完整个数据集后，需要根据交叠区域中合并的结果对整个数据集中合并的类别重新进行编号，从而得到统一的类别编号。

5　实验结果分析

基于 MapReduce 改进 DBSCAN 算法通过搭建 Hadoop 平台实现。实验由于条件原因选择 6 台 Lenovo 计算机组成一个集群环境。处理器配置为 Athlon 64×2；内存为 2GB，硬盘 320GB；网卡速度为 1000MB/s。软件环境选择 Linux 系统 Ubuntu12. 04，Hadoop0. 20，Eclipse4. 4。选择一台计算机作为主结点 master，其余设置为从结点 salve，搭建局域网络。

为了验证改进 DBSCAN 算法基于云计算平台的正确性，采用 KDD CUP 99 数据集用于测试。表 1 对比了单机模式下 DBSCAN 算法和基于 Hadoop 环境改进后的 DBSCAN 算法聚类准确性。通过表 1 可知，基于 MapReduce 改进的 DBSCAN 算法的准确率高于单机模式下 DBSCAN 算法的准确率。

表 1　单机与 Hadoop 环境聚类准确率的比较

类别	数据量	单机	Hadoop
Normal	23454	93.37%	94.24%
DOS	63482	94.83%	96.09%
Probe	3266	83.47%	88.53%
U2R	135	73.23%	89.32%
R2L	4353	86.72%	87.31%
平均准确率		86.32%	91.10%

为了验证算法处理海量数据的时效性，随机生成大小为 1GB、2GB、4GB、8GB 和 16GB 的数据集用于测试。图 4 给出了单机模式下 DBSCAN 算法和 Hadoop 环境下 OPDBSCAN 算法执行效率的对比结果。由图 4 可知，在 Hadoop 环境下基于 MapReduce 改进的 DBSCAN 算法的加速比高于单机的 DBSCAN 算法的加速比。

自动生成大小为 2GB、4GB、6GB、8GB、10GB、12GB 的数据集来测试扩展率。从图 5 可以看出 OPDBSCAN 较原算法 DBSCAN 有稍好的扩展率，主要是因为在 OPDBSCAN 算法的 Reduce 阶段减少了结点的判断过程。

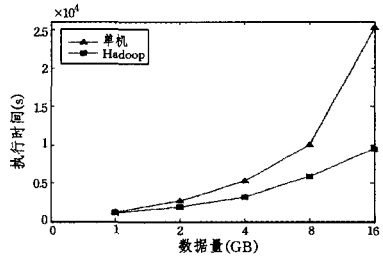


图 4　单机和 Hadoop 环境下执行效率的对比

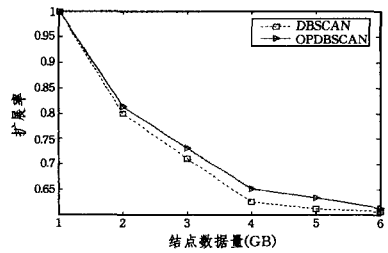


图 5　DBSCAN 算法与 OPDBSCAN 算法扩展率的对比

结束语　本文在研究 DBSCAN 算法的基础上，提出了改进的 OPDBSCAN 算法。OPDBSCAN 算法基于 Hadoop 平台实现，利用交叠分区并行获取局部自适应的 *Eps* 参数解决了 *Eps* 参数敏感问题，再通过 MapReduce 并行实现 DBSCAN，提高算法效率。最后实验证明，改进后的算法能比原算法更快、更好地实现聚类。

参 考 文 献

[1] 李爱国，厍向阳. 数据挖掘原理、算法及应用[M]. 西安：西安电子科技大学出版社，2012：29-35

[2] Ekan Y J，Pallickara S. MapReduce for data intensive scientific analysis[C]// eScience 2008：Proceedings of the Fourth IEEE，International Conference on eScience. Piscataway：IEEE Press，2008：277-284

[3] Dean J，Ghemawat S. MapReduce：Simplified data processing on large clusters [J]. Communications of the ACM，2008，51(1)：107-113

[4] Ester M，et al. A density based algorithm fordiscovering clusters in large spatial databases with noise[C]// Proc of 2nd Inter Conf Knowledge Discovering in Databases and Data Mining (KDD-96). Portland：AAAI Press，1996

[5] 侯荣涛，朱斌. 基于 DBSCA 聚类算法的闪电临近预报模型[J]. 计算机应用，2012，32(3)：847-851

[6] 李莉平，沈俊媛. 基于数据挖掘的 DBSCAN 算法及其应用[J]. 科技创业月刊，2009(8)：134-135

[7] 黄毅磊. DBSCAN 算法及在城市网格化管理中的应用[D]. 上海：上海交通大学，2010

[8] 夏鲁宁，荆继武. SA-DBSCAN：一种自适应基于密度聚类算法[J]. 中国科学院大学学报，2009(4)：530-538

[9] 陈刚，刘秉权，吴岩. 一种基于高斯分布的自适应 DBSCAN 算法[J]. 微电子学与计算机，2013，30(3)：27-30

[10] 周水庚，周傲英，金文，等. FDBSCAN：一种快速 DBSCAN 算法[J]. 软件学报，2000，11(6)：735-744

[11] Ankerst M，Breunig M，Kriegel H-P，et al . Optics：Ordering points to Identify the Clustering Structure[C]// Alex D，Christos F，Shahram G，eds. Proc ACM SIGMOD Int Conf on Management of Data. Philadephia：ACM Press，1999：49-60

[12] 周水庚，周傲英. 基于数据分区的 DBSCAN 算法[J]. 计算机研究与发展，2000，37(10)：1153-1159

[13] 孙凌燕. 基于密度的聚类算法研究[D]. 太原：中北大学，2009

[14] 熊忠阳，吴林敏，张玉芳. 针对非均匀数据集的 DBSCAN 过滤式改进算法[J]. 计算机应用研究，2009，26(10)：3721-3723

(上接第 395 页)

[40] Yoshiki Y，Hung K T，Wayne L. FPGA-Based Smith-Waterman Algorithm：Analysis and Novel Design [M]// Reconfigurable Computing：Architectures，Tools and Applications. 2011：181-192

[41] Nuno S，Nuno R，Paulo F. Hardware accelerator architecture for simultaneous short-read DNA sequences alignment with enhanced traceback phase [J]. Microprocessors and Microsystems，2012，36(2)：96-109

[42] Azzedine B，Jan MC，Alba de M，et al. A Hardware Accelerator for the Fast Retrieval of DIALIGN Biological Sequence Alignments in Linear Space [J]. IEEE Transactions on Computers，2010，59(6)：808-821

[43] Steven D，Patrice Q. Hardware Acceleration of HMMER on FPGAs [J]. J Sign Process Syst，2010，58(1)：53-67

[44] Tim O，Bertil S，Darran N，et al. Using recon? gurable hardware to accelerate multiple sequence alignment with ClustalW [J]. Bioinformatics，2005，21(6)：3431-3432

[45] Scott L，Quinn O S. Accelerated large-scale multiple sequence alignment [J]. BMC Bioinformatics，2011，12(29)：466

[46] Vipin S，Michael K，Evan S，et al. Exploring the viability of the Cell Broadband Engine for bioinformatics applications[C]// Proc of IEEE IPDPS'07. California：IEEE，2007

[47] Vipin S，Michael K，Evan S，et al. Exploring the viability of the Cell Broadband Engine for bioinformatics applications [J]. Parallel Computing，2008，34(11)：616-626

[48] Adam S，Christian L，Philipp K，et al. SWPS3-fast multi-threaded vectorized Smith-Waterman for IBM Cell/B. E. and x86/SSE2 [J]. BMC Research Notes，2008，1(1)：107

[49] Adrianto W，Bertil S，Huiliang Zh，et al. High performance protein sequence database scanning on the Cell Broadband Engine [J]. Scientific Programming，2009，17(12)：97-111

[50] Abhinav S，Srinivas A. Parallel genomic alignments on the Cell Broadband Engine [J]. IEEE Transactions on Parallel and Distributed Systems，2009，20(10)：1600-1610

[51] Hans V，Sean R，Koen D B. Accelerating multiple sequence alignment with the Cell BE processor [J]. The Computer Journal，2010，53(6)：814-826

[52] Adrianto W，Chee K K，Bertil S. Multi-threaded vectorized distance matrix computation on the CELL/BE and x86/SSE2 architecture [J]. Bioinformatics，2010，26(10)：1368-1369

[53] Aarti S，Chen C，Liu W G，et al. A Hybrid Computational Grid Architecture for Comparative Genomics [J]. IEEE Transactions on Information Technology in Biomedicine，2008，12(2)：218-225

[54] Meng X D，Vipin C. A High-Performance Heterogeneous Computing Platform for Biological Sequence Analysis [J]. IEEE Transactions on Parallel and Distributed Systems，2010，21(9)：1267-1280

[55] 卢风顺，宋君强，银福康，等. CPU/GPU 协同并行计算研究综述[J]. 计算机科学，2011，38(3)：5-9，46