ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.11, No.10, April 2015
E-mail: eduf@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

基于MapReduce的DBSCAN聚类算法的并行实现

林阿弟,陈晓锋

(厦门大学 计算机科学系,福建 厦门 361005)

摘要: DBSCAN是一种简单、有效的基于密度的聚类算法,用于寻找被低密度区域分离的高密度区域。DBSCAN是最经常被使用、在科学文献中被引用最多的聚类算法之一。在数据维度比较高的情况下,DBSCAN的时间复杂度为$O(n^2)$。然而,在现实世界中,数据集的大小已经增长到超大规模。对此,一个有效率的并行的DBSCAN算法被提出,并在MapReduce平台下实现它。首先,对已经预处理过的数据进行划分。接下来,局部的DBSCAN算法将对每一块划分好的数据空间实现聚类。最终,利用合并算法对上一阶段的聚类结果进行合并。实验结果验证了并行算法的有效性。

关键词: DBSCAN; MapReduce; 聚类算法; 并行算法; 数据挖掘

中图分类号: TP391 **文献标识码:** A **文章编号:** 1009-3044(2015)10-0161-04

DOI:10.14004/j.cnki.ckt.2015.0436

The Realization of MapReduce-based DBSCAN Density-base Clustering Method

LIN A-di, CHEN Xiao-feng

(Department of Computer Science, Xiamen University, Xiamen 361005, China)

Abstract: DBSCAN is an effective density-based clustering method which is designed to find high-density regions which are separated by low-density regions. DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature. In the case of the data of high dimension, the computation complexity of DBSCAN is $O(n^2)$. However, it is challenging due to the size of datasets has been growing rapidly to extra-large scale in the real world. In this paper, an efficient parallel density-based clustering algorithm is proposed and implemented by using MapReduce. Furthermore, we adopt a quick partitioning strategy for data which has been preprocessed is adopted. Then, Local DBSCAN process for each subspace divided by the partition profile is implemented to generate clusters. At last, the clusters which are generated in the previous phase are merged.

Key words: DBSCAN; mapreduce; clustering algorithms; parallel algorithms; data mining

DBSCAN[1]于1996年被提出以后便被广泛使用。DBSCAN基本时间复杂度是(n*找出样本点的Eps邻域中的点所需要的时间),其中n是样本点的大小。低维数据空间下,利用一些空间索引结构,如kd树[2]、R树[3]、R*树[4]等,时间复杂度可以降到$O(n\log n)$.高维数据空间下,DBSCAN的时间复杂度为$O(n^2)$。PDBSCAN[5]首次采用dR*-tree提出了一个有效的DBSCAN并行算法。然而,创建dR*-tree在海量数据情况下非常的困难,而在数据是高纬度时则毫无效率。MR-DBSCAN[6]提出了基于MapReduce[7]平台下的DBSCAN并行算法。MR-DBSCAN提出了巧妙的数据划分方法,很好的解决了在海量的低维度的数据集进行数据划分时可能产生的负载平衡问题。然而这两个算法均无法有效处理海量的高纬度的数据集。通过对这两个算法进行改进,结合它们的优点,提出了一个适用于海量的高纬度的数据集的DBSCAN并行算法。DBSCAN并行算法分为四阶段。首先,选择数据在选中二维上的划分点。在选中的维度上,依据该维的数据域,将数据分成m份,记下每份数据的点的数目,之后从这m份数据的边界点中选出a个作为划分点。然后,根据各个维的划分点,得到了数据划分。接着,调整得到的数据划分作为局部DBSCAN算法的输入,实施局部DBSCAN算法。最后,利用合并算法对上一阶段的聚类结果进行合并。

1 DBSCAN算法介绍

1.1 DBSCAN的簇

DBSCAN聚类算法需要用户自己确定两个参数Eps和MinPts。其中,Eps为用户定义的半径,MinPts为定义一个点为核心点时其邻域内要求的最少点数。点的邻域的定义将在下文阐述。在给出DBSCAN的簇的定义之前,我们需要知道如下的一些定义:

定义1:(点p的Eps邻域)用NEps(P)表示点P的Eps邻域,dist(p,q)表示点p、q之间的距离,则点P的Eps邻域定义为NEps(p) = {q∈ D | dist(p,q) ≤ Eps}。即点p的Eps邻域为所有与点p的距离不大于Eps的点的集合。

对于簇C,要求对于在簇C中的每一点p,则在簇C中存在一点q,p在点q的Eps邻域内,且q的Eps邻域内的点数>=

收稿日期:2015-02-16
作者简介:林阿弟(1990—),男,福建莆田人,硕士,主要研究方向为计算智能和数据挖掘。

MinPts.我们在给出簇的详细的定义之前,先给出下面一些概念和定义。

若点p的Eps邻域的点数>=MinPts,则点p为核心点。点q的Eps邻域的点数<MinPts,但它落在某个核心点的邻域内,就称点q为边界点。

定义2:(直接密度可达)点p从点q直接密度可达仅当:

1) p∈NEps(q)

2) |NEps(q)|≥MinPts(核心点条件)

定义3:(密度可达)如果存在一串样本点$p_1,p_2\cdots.p_n$,$p_1=q$,$p_n=p$,假如点p_{i+1}从p_i直接密度可达,那么点p从点q密度可达。

在上面我们提过,对于簇C中的任意一点,它必处在一个核心点的邻域中,可以证明,对于同个簇C中的两个边界点,必存在同一个核心点,使得它们同时从该核心点密度可达。为了能够表达这种关系,我们引入了密度相连的定义。

定义4:(密度相连)如果存在任意一点o,点p从点o密度可达,并且点q从点o密度可达,那么点q到点p密度相连。

现在我们可以对簇进行定义了。噪声集可以定义为不在所有簇中的点的集合。

定义5:(簇)D为样本点集,簇C是D的一个非空子集且满足如下条件:

1)∀点p,q:如果p∈C且q从p密度可达,则q∈C。

2)∀点p,q∈C:p与q密度相连。

定义6:(噪声集)$C_1,\cdots,C_k$为样本点集D下满足条件Eps_i和$MinPts_i$,i=1,…,k下的簇。噪声集是D中不属于任何C_i的点的集合,表示为noise={p∈D|∀i:p∉C_i}。

下面的两个引理对于保证下面的DBSCAN算法的正确性至关重要。从直观上看,给定Eps和MinPts,我们可以通过下面两个步骤来发现一个簇。找到样本点集中任意一个核心点做为种子。然后,找出所有与这个种子密度可达的点(包括种子)从而得到了由该种子扩展成的簇。

引理1:点p为样本点集D的一点,且|NEps(p)|≥MinPts,点集合O = {o|o∈D且o从p密度可达}为一个簇。

引理2:簇C为样本点集D中的一个簇,点p为簇C中满足|NEps(p)|≥MinPts的点,则簇C与点集合O = {o|o从p密度可达}等价。

1.2 DBSCAN算法

对于不同的簇C_i取不同较为理想的Eps_i和$MinPts_i$是非常困难的,但是存在全局的且较为理想的Eps和MinPts[1].所以,DBSCAN算法一般取全局的Eps和MinPts。为了发现一个簇,DBSCAN先从样本点集中找到任意一点p,找出所有从该点密度可达的点。如果点p是核心点,由引理2知,这些点构成了一个簇。如果,点p不是核心点,便没有点从点p密度可达,DBSCAN从样本点集中寻找下一点,重复上面的步骤。详细的DBSCAN算法DBSCAN(SetOfPoints, Eps, MinPts)和DBSCAN调用的最重要的函数ExpandCluster见[1]。

当两个簇靠得很近时,它们可能会有交点,由簇的定义知道这些交点必是边界点,那么它们同时属于这两个簇,这种情形我们称之为平局问题。从ExpandCluster知道,这时在DBSCAN算法实施时,它们将归于一个簇,先发现它们的那个簇!而一些在聚类过程中原先被划分为噪声的点,在之后的聚类中,如果发现它从某个点密度可达,则可以改变它的标记为某个簇的标号。

2 DBSCAN的并行实现

2.1 DBSCAN并行算法的数据划分

2.1.1 选择数据的划分点

选中数据集中数据维度中的2个维度,根据第i维上的数据域将数据分成m_i份,记下每份数据的点的数目,i=1,2;假设有N个计算节点,N=a[1]*a[2],则从这m_i份数据的边界点中选出a[i]个作为数据的划分点。选取规则如下:

```
Part(begin,end,a[i],avg)
        IF a[i]>1
            previousEnd = findPartEnd(begin,end-(a[i]-1),
avg);
            Part(previousEnd+1,end,a[i]-1,avg);
            RETURN previousEnd;
        END IF
            RETURN end;
END
找到各个划分点:
findPartEnd(begin,end,avg)
    sum = 0;
    FOR  i  FROM begin TO end-1
        sumNext = 0;
        将第i部分的点的数目累加到sum中
        sumNext = sum + 第i+1部分的点的数目
        IF  avg-sum <= sumNext-avg
            RETURN i;
        END IF
    END FOR
    RETURN end;
END
```

在上面的伪代码中,avg为样本点的数目除以a[i],begin、end分别表示m_i份中的第begin份和第end份。

2.1.2 根据已得到数据的划分点划分数据

根据2.1.1得到的各个维度算得的划分点,得到最终的数据划分。下面举一个例子,来说明总的数据划分的过程。假设原始的数据集如下,

表1 原始的数据集(例子)

点的标号	1	2	3	4	5	6	7	8	9	10	11	12
X维的数据	0.2	0.3	0.3	0.2	0.5	0.6	0.2	0.25	0.8	0.9	0.2	0.8
Y维的数据	0.3	0.3	0.2	0.2	0.3	0.2	0.6	0.3	0.7	0.7	0.7	0.9

在这里,我们的数据已经提前归一化到0.0-1.0,我们选取X、Y维上的数据,均分为N份,除了最后一份,每份的数据域范围为[i*(1/N),(i+1)*(1/N)),i=0,…,N-2,最后一份的数据域范围是[(N-1)*(1/N),1.0],这里N取10。如果,假设这时有6个计算节点,我们希望根据X、Y维,把数据划分成3*2份,那么这时根据2.1.1可算出在X、Y维上的数据划分点,最终的得到的6个数据

划分结果如下：

表2　数据划分结果(例子)

数据在Y维的划分 \ 数据在X维的划分	[0.0,0.3）	[0.3,0.7）	[0.7,1.0]
[0.0,0.4)	1,4,8,	2,3,5	
[0.4,1.0]	7,11	6	9,10,12

2.2 DBSCAN的并行化实现

2.2.1 Local DBSCAN

在2.1节中最后得到的数据划分并不是Local DBSCAN[6]的输入，事实上Local DBSCAN的输入是上面得到的数据划分和它的邻居数据划分的一部分拷贝的总和。在阐述Local DBSCAN的输入时，我们需要说明清楚我们DBSCAN并行算法的原理。针对2.1.1得到的数据划分，我们知道如果仅仅在上面划分下运行DBSCAN算法的程序，那么可能有如下三种情况发生：划分上运行DBSCAN得到的簇与全数据集运行DBSCAN得到的簇没有差别。全数据集上运行DBSCAN得到簇c1和c2，在划分上得到的簇也是簇c1和c2，仅有的差别最多是平局上的处理的差异，结果可以算是一致的。在全数据集运行DBSCAN算法时，簇c1和簇c2应该算是同一个簇，但在当前的数据划分下运行时，则变成两个簇。

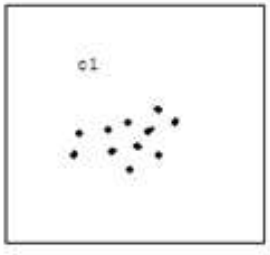


图1　簇仅在一个划分的内部

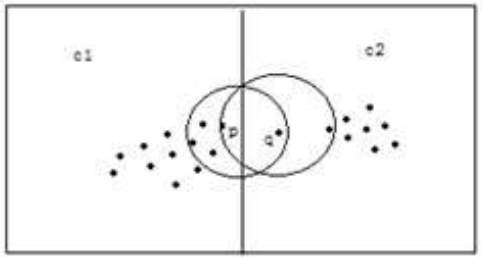


图2　相邻划分上得到的两个簇在全数据集下也是两个簇

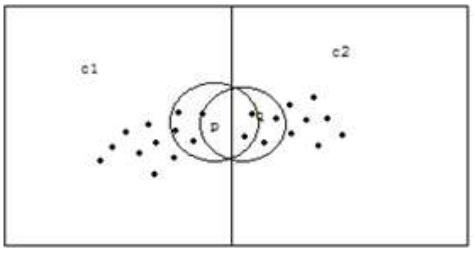
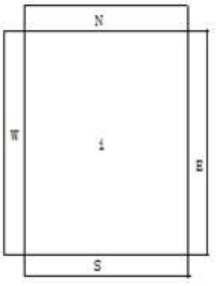


图3　相邻划分上得到的两个簇的在全数据集下为一个簇，需要合并

观察图1、图2、图3发现，图2和图3有个共同点，即在某个划分内，存在距离划分边界<Eps内存在某个核心点，它的邻域中存在点分布在该划分的相邻划分上。图2的情况下并不影响最终聚类的结果，图3的情况下却影响到最终聚类的结果，这是为什么呢？在图2中，左边的划分距离划分边界<Eps内存在核心点p，而在右边的划分上存在点q处在点p的邻域内，但是点q只是边界点，所以从阐述过的DBSCAN算法上看，簇c1和簇c2应该算是不同的簇，如果簇c1和簇c2是同一个簇，那么点p应该与簇c2中的所有点密度相连，这显然是错误的。

现在，我们重点注意图3。左边的划分距离划分边界<Eps内存在核心点p，而在右边的划分上存在点q处在点p的邻域内，并且点q是核心点。同理，右边的划分距离划分边界<Eps内存在核心点q，而在右边的划分上存在点p处在点q的邻域内，并且点p是核心点。如果对上面的结论进行分析的话就可以得出：即在某个划分内，存在距离划分边界<Eps内存在某个核心点p，它的邻域中不在本划分上的点集设为MC(p)，存在相邻划分上距离划分边界<Eps内某个核心点q，点q邻域中不在本划分上的点集设为MC(q)，且p∈ MC(q)，q∈ MC(p)。上面的等价表述是，p与MC(p)的并集与q与MC(q)的并集的交集非空。

上面结论的严格证明参照[5]. 根据上面的结论，MR-DBSCAN提出了Local DBSCAN下的数据输入集。它具有如下形式：



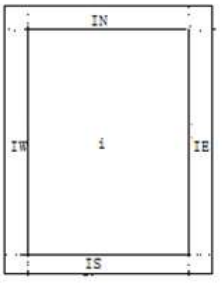


图4　LocalDBSCAN的输入数据集　图5　划分i需要拷贝出去的部分

即第i个输入数据集应当是2.1中得到的第i个数据划分与距离划分边界<=bEps(bEps>=Eps)的点集的并。由上面知道Local DBSCAN的划分内部也有距离划分边界<=bEps(bEps>=Eps)的点集需要拷贝给相应的相邻的划分。我们给定MC集的定义如下：

MC(C, S) = {o ∈ {q} ∪ (NEps(q)\S)|q ∈ C ∩ S ∧ |NEps(q)|≥ MinPts∧NEps(q)\S≠∅}。其中，C为簇，S为划分。

现在，假设有某个簇跨越了划分i和它的相邻的划分，那么根据上面的讨论，必定存在划分i上的MC集与它的相邻的划分的交集非空。关于这部分的严格证明参见[5]。而根据上面提到的Local DBSCAN的输入数据集，MC集必定产生于划分i的IN,IE,IS,IW和相邻划分的拷贝N,E,S,W，准确的说，MC集是IN, IE, IS,IW上的核心点和其邻域在N,E,S,W上的点的并集[6]，这就是Local DBSCAN的输出，这里我们与[6]中的Local DBSCAN的输出不同，我们只统一输出MC集，而不关心它是IN, IE, IS, IW上哪一部分的核心点产生。至于Local DBSCAN的其余部分与DBSCAN基本相同。

2.2.2 DBSCAN并行算法的合并阶段

DBSCAN并行算法的合并阶段是结合MR-DBSCAN的Stage3和Stage4.1，在单机下实现。首先，我们的输入采取冗余输入，输入为所有数据划分中非噪声的数据和该数据划分对应

的MC集，而不仅仅是所有数据划分中非噪声的数据集。在求MC交集时采用的是类似MR-DBSCAN中Stage3的算法，但这个过程在单机下运行，从而得出哪些簇需要合并。然后，根据产生的结果采取与Stage4.1基本相同的方法对Local DBSCAN产生的簇号进行全局下的编号，最后赋予非噪声的数据于全局下的簇标号，这样通过采取冗余输入我们可以省去MR-DBSCAN下的Stage4.2。

3 实验

我们分布式下使用的软件是MapReduce的开源实现Hadoop-0.20.2版本,关于Hadoop的配置参照[8].

在单机环境下，实验步骤如下：数据预处理→DBSCAN算法

在分布环境下，我们通过如下的实验步骤得到结果：数据预处理→选择划分点→划分→局部DBSCAN算法→合并。

我们在3.20GHZ Intel(R) Pentium(R) 4 cpu,2GB DDR2内存,80GB5400转的硬盘上伪分布环境下测试53052条二维数据，系统为ubuntu 10.04 lts,Eps=0.02,MinPts=8,我们在此实现的数据二划分，即将数据根据划分算法一分为二，实验结果如表3所示：

表3 实验结果

单机下(S)	1+2169=2170
伪分布下(S)	1+25+20+677+1=724

而在两台3.60GHZ Intel Core(TM) i7-3820,8GBDDR2内存，200GB7200转的硬盘的电脑组成的集群下，分布式中的局部DBSCAN费时75(S)，单机下为1+138=139(S)，由于所测的数据量不够大，在集群下没有表现应有的优势。但是仍然得到了加速。可以预见，随着数据规模的增大，分布式DBSCAN算法的优势将会更明显。

4 结论

从实验结果可以看出，单机下聚类所使用的时间与伪分布式环境下的时间的比为2170/724=2.997，实验效果显然是比较不错的，在以后的实验中，我们将实现数据的多划分，并测试大数据集在集群下的实验结果，从而找出海量的高纬度的数据集下有效的DBSCAN并行算法。

参考文献：

[1] Ester M, Kriegel H P, Sander J, et al. A density based algorithm for discovering clusters in large spatial databases[C]. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, 1996: 226 - 231.

[2] Bentley J L. Multidimensional binary search trees used for associative searching[J]. Communications of the ACM, 1975, 18 (9): 509-517.

[3] Guttman A. R-trees: a dynamic index structure for spatial searching[J]. ACM, 1984, 14(2): 47-57.

[4] Beckmann N, Kriegel H P, Schneider R, et al. The R*-tree: an efficient and robust access method for points and rectangles [M]. ACM, 1990,19(2): 322-331.

[5] Xu X, Jager J, Kriegel H P. A fast parallel clustering algorithm for large spatial databases[J]. Data Min. Knowl. Discov, 1999(3): 263 - 290.

[6]Yaobin He, Tan Haoyu, Luo Wuman, Huajian Mao, Di Ma, Shengzhong Feng, Jianping Fan.MR- DBSCAN: An Efficient Parallel Density-based Clustering Algorithm using MapReduce [J]. 2011 IEEE 17th International Conference on Parallel and Distributed Systems,2011: 473-480.

[7] Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters[J]. Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. Berkeley,CA, USA: USENIX Association, 2004: 10.

[8] 陆嘉恒.Hadoop实战[M].北京：机械工业出版社，2011.