高技术通讯 2012 年 第22 卷 第6 期:589 ~ 595

doi:10.3772/j.issn.1002-0470.2012.06.006

S-DBSCAN:一种基于 DBSCAN 发现高密度簇的算法①

孙 鹏②*** 韩承德** 曾 涛***

(*中国科学院研究生院 北京 100049)
(**中国科学院计算技术研究所 北京 100190)
(***天津师范大学计算机与信息工程学院 天津 300387)

摘 要 针对基于密度的带有噪声的空间聚类(DBSCAN)算法用于交互式数据挖掘时用户经常调整算法参数以发现感兴趣的知识以及数据集相对稳定的特点,提出了一种基于DBSCAN 发现高密度簇的算法——S-DBSCAN 算法,确定了需调整的算法参数——对象的邻域范围 ε(Eps)和满足核心对象条件的 ε 邻域内最小对象个数 MinPts,阐述了参数 ε 与 MinPts 的 3 种适合 S-DBSCAN 算法的变化情况,并给出了相应的证明,同时分析了算法的时间复杂度。在对真实和合成数据集的测试中,S-DBSCAN 算法相比 DBSCAN 算法具有较好的效率。

关键词 基于密度的带有噪声的空间聚类(DBSCAN), S-DBSCAN, 高密度簇, 聚类, 参数可变

0 引 言

在空间数据挖掘中,聚类是指把大量的 d 维数据对象(n 个)聚集成 k 个聚类($k < n$),使同一聚类内对象的相似性尽可能最大,而不同聚类内对象的相似性尽量达到最小。也就是说,形成聚类之后,同一个聚类内对象具有很高的相似性,而且与不属于该聚类的对象有迥然的差异(即不相似)[1,2]。在实际应用中,聚类挖掘得到了大量的应用,如对兴趣点(point of interest,POI)数据集进行聚类用于生成新的地理区域,并对这些区域进行商业区、商务区、工业区等分类[3]。Zhou 等人对个人使用移动设备采集的数据使用聚类分析生成有意义的地点,如家、办公室或常去的俱乐部等[4]。

基于密度的带有噪声的空间聚类(density-based spatial clustering of applications with noise,DBSCAN)算法是聚类挖掘中的经典算法之一,此算法通过检查数据集 D 中每个点的 ε 邻域来判断它是否是核心点,进而决定如何扩展簇。DBSCAN 算法能够识别各种复杂形状,有效排除噪声干扰,并且聚类结果不受输入顺序影响,其时间复杂度为 $O(n\log n)$[2,5]。在交互式聚类挖掘中,用户往往对同一数据集进行多次的聚类挖掘分析,以获取不同的有意义的或者感兴趣的知识。同时还要进行对算法的调整,主要是对算法参数的调整。对于 DBSCAN 算法,调整的参数主要有距离函数中的维度、对象的邻域范围 ε(记为 *Eps*)和满足核心对象(core object)条件的 ε 邻域内最小对象个数 *MinPts*。文献[6]讨论了距离函数中的维度变化的情况,本文讨论了(ε,*MinPts*)参数的调整。当发现高密度簇完全被相连的低密度簇所包含时,用户有时需要调整参数发现高密度簇。另外,数据挖掘中所处理的数据集往往是周期更新的[7],这种周期是一天甚至更长。进而,本文针对参数 ε 和 *MinPts* 的变化,提出了基于 DBSCAN 发现高密度簇的算法——S-DBSCAN 算法,并做了相应的分析。

1 相关工作

Ester 等人提出了用于解决增量聚类(incremental clustering)的基于密度的算法 IncrementalDBSCAN[7]。IncrementalDBSCAN 在数据仓库中发生插入和删除操作时无需根据 DBSCAN 算法全部重

① 863 计划(2009AA12Z220, 2009AA12Z226)资助项目。
② 男,1980 年生,博士生;研究方向:空间数据挖掘;联系人,E-mail:sunpeng@ict.ac.cn
(收稿日期:2011-02-25)

新计算,而这种重新计算往往代价巨大。当插入删除对象 p 发生时,他们定义 p 的影响对象集(affected objects) $\text{Affected}_D(p)$ 为 p 的 ε 邻域中的所有点和与这些点密度可达的点的并集。点对象 p 是从点对象 q 密度可达的,是指存在一条点对象链 p_1, …, p_n, 该链的 p_{i+1} 是从 p_i 直接密度可达的,其中 $p_1 = q$, $p_n = p$。相比于增量聚类中的影响对象集的影响范围小这个特点,本文的 S-DBSCAN 算法由于参数变化将会产生大量的影响对象集对象,而不仅仅是新增或删除 p 点所影响的周围的一些对象。IncrementalDBSCAN 算法会根据影响对象集首先计算插入种子对象集 $\text{UpdSeed}_{\text{Ins}}$ 或删除种子对象集 $\text{UpdSeed}_{\text{Del}}$。种子对象集定义为更新 p 后核心对象变为非核心对象或者非核心对象变为核心对象的 *Eps* 邻域中的新的全局数据集范围中的核心对象。当插入操作发生后,将会发生 4 种情况:噪音、新建、吸收、合并。当删除操作发生后,移除、缩小、分裂可能会发生。Böse 等人探索了并行和增量聚类结合的算法[8]。该算法利用 Map-Reduce 进行并行计算,且与 Map-Reduce 编程模型不同,批处理采用了 stream processing 方式。Goyal 等人探讨了把多个新的对象先组合成块的方式再进行增量聚类的方法[9]。该方法首先聚类新的数据集,而不是将新的数据集逐个地渐增地添加到原有的簇中。

Wu 提出了渐变聚类(gradual clustering)即纵向增量聚类(vertically incremental clustering)的概念[6]。渐变聚类主要用于解决距离函数中用户所选属性字段或者维数发生了变化这种情形,例如用户一开始只选择了一小部分属性进行距离函数的计算,在后续的聚类中又选择了更多的属性进行计算。该算法的核心思想是利用三角不等性减少距离计算数量。该方法首先在 n 维空间中存储各个对象与代表对象之间的距离索引,然后在 $n + m$ 维上利用这些预先存储的距离。

Li 等人提出了 IPC-DBSCAN 算法来处理参数变化的情形[10]。对于用户新输入的参数, IPC-DBSCAN 使用历史聚类的结果进行计算,而不是采用 DBSCAN 进行全新的计算。为简化问题,文献[10]仅仅分析了满足核心对象条件的 ε 邻域内最小对象个数 *MinPts* 的变化情况。而本文的算法同时支持对二者的调整。文献[10]将改变对象集 Changed_D 定义为 *MinPts* 变化后核心对象变化为非核心对象的点,或非核心对象变化为核心对象的点,并对 *MinPts* 变小时的下述 4 种情形进行了分析:(1)不变,(2)新建,(3)扩展,(4)合并。情形(2)和(3)均可能会把两个足够近的簇进行合并。同时分析了 *MinPts* 变大时的下述 4 种情形:(1)不变: Changed_D 为空集,没有任何变化发生;(2)移除:如果 Changed_D 包含一个仅由该对象为核心对象形成的簇时,则移走该对象所创建的簇;(3)收缩:如果一些核心对象变为非核心对象,则该对象 *Eps* 邻域中的核心对象如果不在其他的边界对象的 *Eps* 邻域中,则变为噪音对象;(4)分裂连接两个高密度区域的核心对象如果消失的话,则该簇需要进行分裂。为了计算改变对象集 Changed_D,每个对象需要存储额外的信息:(1) *Eps* 邻域中的点的数量,记为 Neighbors. count;(2)每个 *Eps* 邻域中的点到该核心对象的距离,而本文的 S-DBSCAN 算法却不需要存储这些额外信息。

Liu 等人提出了基于 *K*-Means 的参数变化的算法[11]。该算法分析了当 K 递增或递减即 $K+1$ 或 $K-1$ 时对聚类结果的影响,即分裂(split)和合并(merge),对于这两种情况,算法分为两步,第一步寻找新的中心点,第二步在新的中心点上再次进行 *K*-Means 算法,由于新的中心点相对于原有的中心点较少,例如 Split 时会产生两个新的中心点,便缩小了搜索范围。

刘青宝等人提出了基于相对密度的增量式聚类算法 RDBIClustering[12]。该算法解决了 DBSCAN 算法采用一个全局参数时,高密度聚类结果被包含在相连的低密度聚类结果中的问题,而有些应用场景却希望能把二者区分开。RDBIClustering 算法所采用的策略是在 ExpandCluster 过程中,计算条件 $(1 - rd_{\text{CoreSet}}(\text{object})) < \delta$,其中 $rd_{\text{CoreSet}}(\text{object})$ 是指对象关于 k 近邻的相对密度,用以对密度渐变所产生累加效应进行阈值检查,使得在数据密度连续渐变的情况下,也能区分不同密度等级的类。我们的算法与 RDBIClustering 不同的是通过用户调整 *Eps*、*MinPts* 这两个参数来发现高密度的簇,而且本文给出了三类参数的变化,用以指导用户如何调整参数。

2 S-DBSCAN 算法

本节首先讨论适用于 S-DBSCAN 算法的(*Eps*, *MinPts*)三种变化情形,然后给出 S-DBSCAN 算法的设计,并证明算法的正确性,最后讨论算法的时间复杂度。

2.1　*Eps*,*MinPts* 的 3 种变化

带有噪声的空间聚类(DBSCAN)中的参数对象的邻域范围 *Eps* 和满足核心对象条件的 ε 邻域内最小对象个数 *MinPts* 相比于历史请求的参数各有 3 种变化情况:变大,不变,变小。根据排列组合(*Eps*, *MinPts*)共有 9 种变化情况,其中(*Eps* 不变,*MinPts* 变大),(*Eps* 变小,*MinPts* 变大),(*Eps* 变小,*MinPts* 不变)这 3 种变化情况会使形成核心对象的条件变得更为严格,下文针对这 3 种变化情况进行分析。

(1) *Eps* 不变,*MinPts* 变大

Eps 不变对一个对象成为核心对象的条件不产生影响,而根据 DBSCAN 算法[5] 中的定义 2 中核心对象的条件 $|N_{Eps}(q)| \geqslant MinPts$,其中 $N_{Eps}(q)$ 表示一个点对象的 *Eps* 近邻,知 *MinPts* 变大会使一个对象成为核心对象的条件变的更为严格,故总体来讲,*Eps* 不变,*MinPts* 变大会使得核心对象在新的参数下可能成为核心对象、边界对象或噪音,而边界对象可能成为边界对象或噪音,而噪音仍然是噪音。这就使得原有的簇会产生收缩(Shrink,见图 1(a))、分裂(Split,见图 1(b))甚至解散(Dismiss,见图 1(c))。

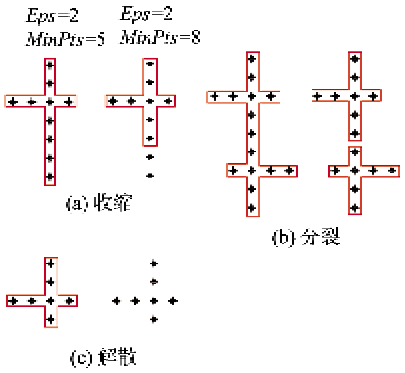

图 1　*Eps* 不变,*MinPts* 变大时的簇变化

(2) *Eps* 变小,*MinPts* 变大

DBSCAN 算法[5] 中的 regionQuery(currentP, *Eps*)函数一般是 *Eps* 的单调递增函数,故当 *Eps* 变小时,数据集 D 中任意一点 *p* 的 *Eps* 邻域内的对象个数具有非递增性质。例如距离度量方法如果采用欧氏距离或曼哈顿距离时,*Eps* 邻域内对象的个数随着 *Eps* 减小是非递增的,故 *Eps* 变小会缩小邻域查询范围,进而使得一个对象成为核心对象的条件变得更为严格。*MinPts* 变大也同样会使一个对象成为核心对象的条件变的更为严格,故类似于情形(*Eps* 不变,*MinPts* 变大),该类参数变化同样会产生三种簇的变化结果:收缩(见图 2(a))、分裂(见图 2(b))、解散(见图 2(c))。

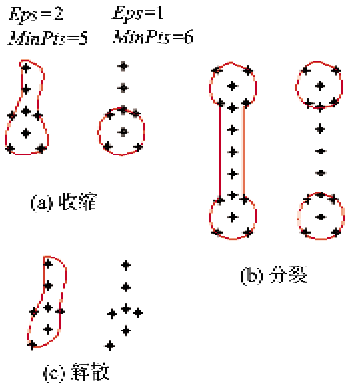

图 2　*Eps* 变小,*MinPts* 变大时的簇变化

3) *Eps* 变小,*MinPts* 不变

根据以上分析,*MinPts* 不变的情况下,*Eps* 变小会使一个对象成为核心对象的条件变的更为严格,故同理原有的簇同样可能会产生三种簇的变化情形:收缩(见图 3(a))、分裂(见图 3(b))、解散(见图 3(c))。

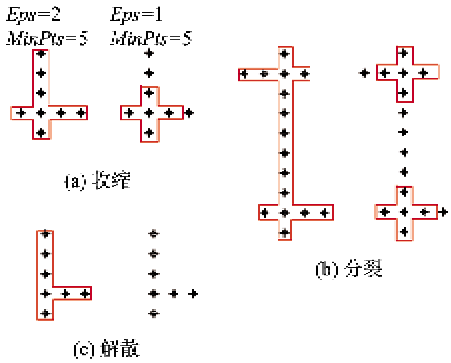

图 3　Eps 变小,MinPts 不变时的簇变化

2.2　S-DBSCAN 算法

Eps,*MinPts* 上述的三种变化情况均使得数据集中的对象 *p* 成为核心对象的条件变得更为严格,而其他 6 种变化则使得对象 *p* 成为核心对象的条件变得更为宽松,或者宽松和严格并存,而本文分析的三种变化情况则会使得原有的簇产生收缩、分裂或者解散,这就使得新产生的簇密度相同或高,故本文根据该特点提出了获得高密度簇的算法 S-DBSCAN 算法。下面先给出记号约定:DBSCAN 算法中原有的参数记为 *Eps'*,*MinPts'*,新的参数记为 *Eps*,*MinPts*,全体数据集数量记为 *n*,原有的噪音数量记为 *noise'*。

表 1 描述了该算法。S-DBSCAN 算法的核心思想是仅在原有的簇中重新根据 DBSCAN 算法进行计算，而无需考虑原有噪音对象，该算法首先把非噪音对象的新的 clusterId 设置为 UNDEFINED，而对噪音对象仍然设为 NOISE，然后该算法遍历所有的簇中对象，仅处理新的 clusterId 为 UNDEFINED 的对象，并选取其中的任意一点进行扩展，算法中 ExpandCluster 函数和 DBSCAN 算法相同，只是在该算法中 regionQuery(currentP, *Eps*) 函数的搜索范围缩小为原簇中的对象，即只需要在原簇中进行 *Eps* 近邻查询。

表 1 S-DBSCAN 算法的描述

```
S-DBSCAN
Input: Eps, MinPts, Eps', MinPts', T.clusterId'
Output: T.clusterId
for all objects t in T do
    t.clusterId = (t.clusterId' == NOISE)? NOISE: UNDEFINED
end for
ClusterId := nextId(NOISE);
for all clusters c in T.clusterId' do
   for all objects t in c do
    if t.clusterId == UNDEFINED then
     if ExpandCluster(c,t,ClusterId,Eps, MinPts) then
       ClusterId := nextId(ClusterId)
     end if
     end if
   end for
end for
```

2.3 正确性证明

定理 1：当参数 $Eps \leqslant Eps'$ 且 $MinPts \geqslant MinPts'$ 时，原有的噪音对象仍然为噪音。

简证：当 $Eps \leqslant Eps'$ 时，$|N_{Eps}(q)| \leqslant |N_{Eps'}(q)|$，而噪音对象 q 满足 $|N_{Eps'}(q)| < MinPts' \leqslant MinPts$，故 $|N_{Eps}(q)| < MinPts$。问题得证。

根据**定理 1**，由于原有的噪音仍然为噪音，故在新的更为严格的参数下而无需计算噪音，也即噪音对象不会产生新的簇，故只需要在原有的簇中的对象进行计算。故 S-DBSCAN 算法和 DBSCAN 算法执行结果等价。算法正确性得证。

2.4 时间复杂度

在表 1 中 ExpandCluster 调用的 Eps 邻域查询可以采用 $R*$ 树（一种动态平衡树）加快速度，由于此时参与计算的全体数据集是 $(n - noise')$，而建立索引的时间可以建在非算法执行期间，故 *Eps* 邻域查询的时间复杂度为 $\log(n - noise')$，而表 1 中最外层调用 Eps 邻域查询的次数为 $(n - noise')$，故总体复杂度为 $(n - noise')\log(n - noise')$，最坏情况是当 $noise'$ 数量很小或接近为 0 时，算法时间复杂度退化为与 DBSCAN 算法等价。

而如果 $R*$ 树索引仍然使用原有建立在全体数据集上的话，则 *Eps* 邻域的查询的时间复杂度为 $\log(n)$，而最外层调用次数仍然为为 $(n - noise')$，故总的复杂度为 $(n - noise')\log(n)$，最坏情况是当 $noise'$ 数量很小或接近为 0 时，算法时间复杂度退化为与 DBSCAN 相同，即 $n\log(n)$。

通过上面的分析，S-DBSCA 算法对于原有的历史聚类结果中含有大量的噪音对象的情况，将会产生很好的执行效率，因为大量的噪音数据被过滤掉不参与计算。

3 性能验证

3.1 实验环境

本研究分别对真实数据和合成数据进行了试验。整个算法采用外存算法，外存索引采用 $R*$ 树，S-DBSCAN 算法仍然使用原有建立在全体数据集上的 $R*$ 树索引，故其时间复杂度为 $(n - noise')\log(n)$。

实验运行环境为一台处理器 AMD Athlon(tm) 64 X2 Dual Core Processor 3800 +，内存为 2G (DDR2)，硬盘转速为 7400r. p. m 的台式机，操作系统版本为 Ubuntu 9.10，kernel 为 2.6，文件系统是 ext4，gcc 版本为 4.4.1，编译优化选项为默认级别-O2。

3.2 真实数据性能验证

真实数据使用在文献[5]中使用的标准测试集 SEQUOIA 2000，它是一个真实空间数据集，含有 62556 个 California 的 POI 点。由于 S-DBSCAN 算法将不对原有的噪音对象进行计算，故 S-DBSCAN 算法对于噪音比例较高的原有的计算结果比较有效，也即 S-DBSCAN 执行时间与 $1 - \frac{noise'}{n}$ 相关。本文分别由低到高设计了三组不同 $1 - \frac{noise'}{all}$ 的实验，并分别涵盖了（*Eps* 变小，*MinPts* 变大），（*Eps* 变小，*MinPts* 不变），（*Eps* 不变，*MinPts* 变大）三种参数变化情况。

第一组数据的(Eps',$MinPts'$)为(2000,18),1 $-\frac{noise'}{n}=1-\frac{55689}{62556}=0.109774$, S-DBSCAN 的执行时间相比于 DBSCAN 平均为 0.1459,略高于 1 $-\frac{noise'}{n}$。实验结果如图 4 所示。

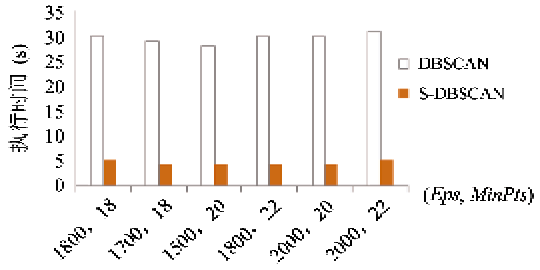

图 4　SEQUOIA 数据效率实验 1

第二组数据的(Eps',$MinPts'$)为(2000,7),1 $-\frac{noise'}{n}=1-\frac{34591}{62556}=0.447039$, S-DBSCAN 的执行时间相比于 DBSCAN 平均为 0.50558,略高于 1 $-\frac{noise'}{n}$。实验结果如图 5 所示。

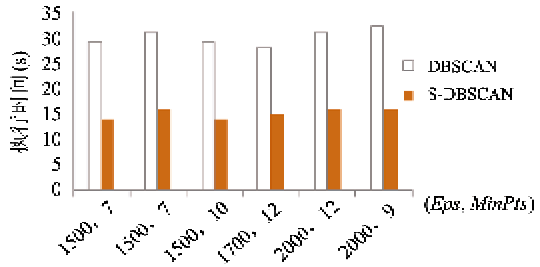

图 5　SEQUOIA 数据效率实验 2

第三组数据的(Eps',$MinPts'$)为(4000,8),1 $-\frac{noise'}{n}=1-\frac{7563}{62556}=0.879$, S-DBSCAN 的执行效率平均为 0.871,大约与 1 $-\frac{noise'}{n}$相同。实验结果如图 6 所示。

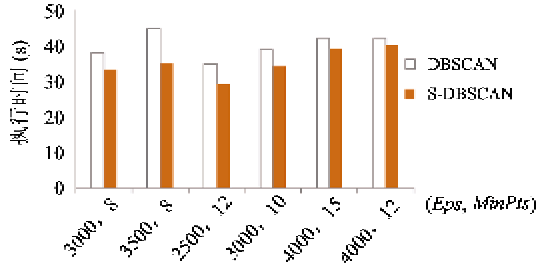

图 6　SEQUOIA 数据效率实验 3

实验分析:由于 S-DBSCAN 中采用的 R * 树索引不变,而 S-DBSCAN 的时间复杂度为(n-$noise'$)log(n), DBSCAN 的复杂度 $n\log(n)$,故理论上 S-DBSCAN 相对于 DBSCAN 的执行时间比正比于 1 $-\frac{noise'}{n}$,加上初始化簇对象为 UNDEFINED 的代价,而这种初始化往往在内存中执行,故相对于外存高昂的 I/O 代价,总体上会使得执行效率会略高于 1 $-\frac{noise'}{n}$。实验一和实验二的结果也验证了这一点。实验三的平均执行效率和 1 $-\frac{noise'}{n}$大体相同,即使采用了与上次执行时相同的 R * 树索引,但是搜索的结果不包含噪音数据的话,加上数据集在外存分布上的特点,这会减少一些页面的访问,故会冲销一部分或全部初始化簇所带来的性能损失。通过分析还发现在即使比率 1 $-\frac{noise'}{n}$达到近 0.9 的情况下,S-DBSCAN 算法比 DBSCAN 仍然具有更好的性能。

3.3　合成数据性能验证

本文使用文献[13]中的数据集生成器 SynDECA。共生成 100000 个二维对象,每一维的取值范围为 0.0 ~ 1000.0 之间的实数。与真实数据的实验类似,根据 1 $-\frac{noise'}{all}$由低到高设计了三组不同实验,并分别涵盖了三种参数的变化情况。

第一组数据的(Eps',$MinPts'$)为(5.0,120),1 $-\frac{noise'}{n}=1-\frac{89739}{100000}=0.10261$, S-DBSCAN 的执行时间相比于 DBSCAN 平均为 0.1649,略高于 1 $-\frac{noise'}{n}$。实验结果如图 7 所示。

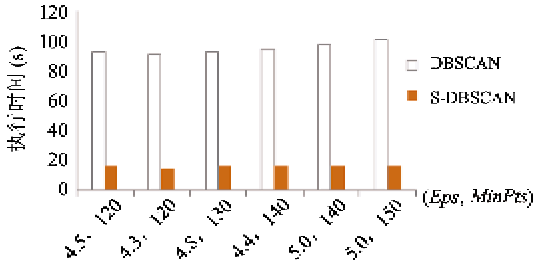

图 7　SynDECA 合成数据效率实验 1

第二组数据的(Eps',$MinPts'$)为(5.0,100),1 $-\frac{noise'}{n}=1-\frac{61082}{100000}=0.389$, S-DBSCAN 的执行时间相比于 DBSCAN 平均为 0.4789,略高于 1 $-$

$\frac{noise'}{n}$。实验结果如图 8 所示。

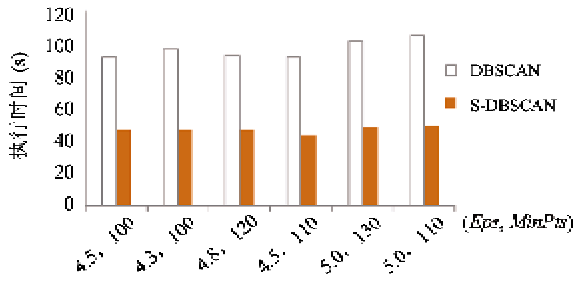

图 8 SynDECA 合成数据效率实验 2

第三组数据的(Eps', $MinPts'$)为(5.0,80),$1-\frac{noise'}{n}=1-\frac{32530}{100000}=0.6747$, S-DBSCAN 的执行效率平均为 0.709,略高于 $1-\frac{noise'}{n}$。实验结果如图 9 所示。

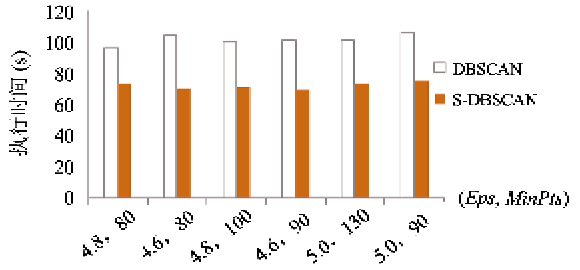

图 9 SynDECA 合成数据效率实验 3

实验分析:与真实数据集 SEQUOIA 2000 的实验结果类似,$1-\frac{noise'}{n}$越低,S-DBSCAN 算法相比 DBSCAN 执行效率越好,我们通过 QGIS 工具把参数变化前后生成的簇进行对比发现,无论对于真实数据集还是对于合成数据集算法的执行结果均是在原有的簇上产生了高密度的簇,这为指导用户如何调整参数获取高密度的簇提供了参考。

4 结 论

在交互式数据挖掘中,用户往往希望通过改变参数获取更加有意义的结果,同时对挖掘的数据进行周期更新而非事务性实时更新。针对这两个特点,本文分析了 *Eps* 和 *MinPts* 参数的 3 种变化情况,提出了一种基于 DBSCAN 发现高密度簇的算法,即 S-DBSCAN 算法。相比于以前的方法,本文实验结果表明 $1-\frac{noise'}{n}$达到一定的阈值前具有良好的效率,而该阈值往往即使很高,如在真实数据 0.9 和合成数据 0.7 的情况下都具有良好的性能提升。在今后的工作中,本文需要继续研究外存算法中索引对原有簇的对象的性能影响,探索并行化以及多核上加速该算法的方法。

参考文献

[1] 杨小兵. 聚类分析中若干关键技术的研究:[博士学位论文]. 杭州:浙江大学计算机学院,2005. 8-9

[2] Han J W, Kamber M. Data Mining: Concepts and Techniques. 2nd ed. San Francisco: Morgan Kaufmann, 2006. 383-384

[3] Yasmina S M, Adriano M. Automatic classification of location contexts with decision trees. In: Proceedings of the Conference on Mobile and Ubiquitous Systems, Guimaraes, Portugal, 2006. 74-88

[4] Zhou C Q, Frankowski D, Ludford P, et al. Discovering personally meaningful places: An interactive clustering approach. *ACM Trans Inf Syst*, 2007, 25(3): 12

[5] Ester M, Kriegel H P. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, USA, 1996. 226-231

[6] Wu F, Gardarin G. Gradual clustering algorithms. In: Proceedings of the 7th International Conference on Database Systems for Advanced Applications, Los Alamitos, USA, 2001. 48-55

[7] Ester M, Kriegel H P, Sander J, et al. Incremental clustering for mining in a data warehousing environment. In: Proceedings of the 24th International Conference on Very Large Data Bases, New York, USA, 1998. 323-333

[8] Böse J H, Andrzejak A, Högqvist M. Beyond online aggregation: parallel and incremental data mining with online Map-Reduce. In: Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud, Raleigh, USA, 2010. 1-6

[9] Goyal N, Goyal P, Venkatramaiah K, et al. An efficient density based incremental clustering algorithm in data warehousing environment. In: Proceedings of 2009 International Conference on Computer Engineering and Applications, Manila, Phillipines, 2009. 556-560

[10] Li F, Liu S, Dou Z T, et al. An inheritable clustering algorithm suited for parameter changing. In: Proceedings of the 2004 International Conference on Machine Learning and Cybernetics, New York, USA, 2004. 1198-1203

[11] Liu S, Feng X J, Feng X. An inheritable algorithm for repeated clustering. In: Proceedings of the International

Conference on Computer Science and Software Engineering, Wuhan, China, 2008. 340-343.

[12] 刘青宝, 侯东风, 邓苏等. 基于相对密度的增量式聚类算法. 国防科技大学学报, 2006

[13] Vennam J R, Vadapalli S. SynDECA: a tool to generate synthetic datasets for evaluation of Clustering algorithms, In: Proceedings of the 11th International Conference on Management of Data, Goa, India, 2005. 27-36

S-DBSCAN: an algorithm for finding high density clusters based on DBSCAN

Sun Peng[*,**], Han Chengde[**], Zeng Tao[***]

([*] Graduate University of Chinese Academy of Sciences, Beijing 100049)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] College of Computer and Information Engineering, Tianjin Normal University, Tianjin 300387)

Abstract

Considering that when the algorithm based on density-based spatial clustering of applications with noise (DBSCAN) is applied to interactive data mining, certain algorithm parameters are usually adjusted to find new knowledge, and the data set used in data mining is relatively stable, this paper presents an algorithm for finding high density clusters based on DBSCAN, called the S-DBSCAN algorithm, and determines the parameters needing to be adjusted, the ε, neighborhood of an object, and the MinPts, minimal number of objects of ε-neighborhood to form a core object. Then three different combinations of the variations of ε-neighborhood and MinPts fit for the S-DBSCAN algorithm are introduced, and the rightness is demonstrated and the time complexity is analyzed. The experiments on real and synthetic data were performed to verify the efficiency and the results show that the S-DBSCAN algorithm has a better efficiency than DBSCAN.

Key words: density-based spatial clustering of applications with noise (DBSCAN), S-DBSCAN, high density clusters, clustering, parameter changing