一种用于位置数据库结构调整的增量聚类算法*

马 帅[1,2+], 唐世渭[1,2], 杨冬青[1,2], 王腾蛟[1,2]

[1](北京大学 计算机科学技术系,北京 100871)

[2](北京大学 视觉与听觉信息处理国家重点实验室,北京 100871)

An Incremental Clustering Algorithm for the Topology Adjustment of Location Databases

MA Shuai[1,2+], TANG Shi-Wei[1,2], YANG Dong-Qing[1,2], WANG Teng-Jiao[1,2]

[1](Department of Computer Science and Technology, Peking University, Beijing 100871, China)

[2](National Laboratory on Machine Perception, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-10-62756374, E-mail: mashuai@db.pku.edu.cn, http://www.pku.edu.cn

Received 2003-11-24; Accepted 2004-03-17

Ma S, Tang SW, Yang DQ, Wang TJ. An incremental clustering algorithm for the topology adjustment of location databases. *Journal of Software*, 2004,15(9):1351~1360.

http://www.jos.org.cn/1000-9825/15/1351.htm

Abstract: How to effectively organize and store the profile of moving objects in a mobile environment, where then can effectively lower the paging and update cost, is an important problem in location management. Combining data mining into the mobile environment is a challenging research task, which has broad applications. Zone partition can effectively optimize the topology of location databases and efficiently reduce the cost of location paging and location update. But with the evolving time, the mobile users' moving patterns may change, so the original partitions may not match the current moving patterns. Thus one of the important problems, which need to be solved, is how to partition the zones dynamically. Clustering method can solve the static zone partition well, but face with the dynamic zone partition problem. If the clustering method is still used to solve this problem, it means that the zones are partitioned again from scratch, which doesn't utilize the original partitions and need great cost. In this paper an incremental clustering method is provided to solve the dynamic zone partition problem, which adjusts the original zone partitions with less cost and guarantees all the conditions needed for zone partition problem in the meanwhile.

Key words: incremental clustering; data mining; location database; location management; mobile communication

* Supported by the National High-Tech Research and Development Plan of China under Grant Nos.863-317-01-04-99, 2002AA4Z3440 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032705 (国家重点基础研究发展规划(973)); the Foundation of the Innovation Research Institute of PKU-IBM (北京大学-IBM 创新研究院项目资助)

作者简介: 马帅(1975－),男,山东潍坊人,博士,主要研究领域为数据挖掘,数据仓库,移动计算;唐世渭(1939－),男,教授,博士生导师,主要研究领域为数据库,信息系统;杨冬青(1945－),女,教授,博士生导师,主要研究领域为数据库,信息系统;王腾蛟(1973－),男,博士,讲师,主要研究领域为数据库,信息系统.

摘 要: 在移动通信网络环境中,如何合理地组织和存储移动对象的配置信息,从而有效地降低查询和更新代价是位置管理中的一个重要问题.将数据挖掘应用到移动计算环境中是一项具有挑战性的研究课题,具有广阔的应用前景.区域划分能够优化位置数据库的拓扑结构,有效地降低查询和更新代价.但是随着时间的迁移,用户的移动模式会发生改变,导致原有区域的划分与当前的移动模式不符,因此产生了动态区域划分这一亟待解决的重要问题.聚类可以很好地解决区域划分问题,而对于动态区域划分问题,如果仍然采用聚类来解决,就等于重新划分,没有充分利用原有划分的信息,所需代价很大.提出了一种增量的聚类算法来解决动态区域划分问题.该方法以较小的代价调整原有划分,使得新得到的划分仍然满足区域划分所需满足的条件.

关键词: 增量聚类;数据挖掘;位置数据库;位置管理;移动通信

中图法分类号: TP311 **文献标识码**: A

在移动通信网络环境中,一个最重要的也是最具挑战性的问题是位置管理(location management)[1].位置数据库(location databases)是存储移动用户配置信息(profile)的分布式数据库,其中移动用户的位置(location)是配置信息中的一项重要内容.位置更新和位置查询要更新和查询位置数据库中的移动用户的位置信息.位置数据库的拓扑结构及其优化技术是一项提高位置管理性能的重要方法.为了充分地利用有限而宝贵的网络资源来为移动用户提供更好的服务,位置管理需要紧密结合移动用户的行为特征来提高位置管理服务的质量.数据挖掘是发现知识的最有效的方式之一,通过挖掘用户的移动日志可以有效地发现用户的移动特征.此外,移动环境中的数据常常是海量的、动态的,通常同时包含空间和时间的特性,这为传统的数据挖掘提出了新的挑战,研究基于数据挖掘的位置管理具有重大的理论研究意义和商业应用价值.

聚类是数据挖掘领域中的一个重要分支,它既可以作为一个单独的工具发现数据库中数据分布的一些深入的信息,也可以作为数据挖掘中其他分析算法的一个预处理步骤.所谓聚类,就是将数据库中的数据进行分组,使得组内的数据尽可能相似而不同组的数据尽可能不同[2].迄今为止,研究人员已经提出了许多聚类算法,聚类可以分为基于划分[3]、基于层次[4~7]、基于密度[8,9]、基于网格[10,11]、基于模型的方法[2],文献[2,12,13]对此做了很好的分析和总结.此外,由于数据库中的数据通常是不断变化的,原来得到的模型/模式可能与新的数据不匹配,得到针对更新后数据库的新的模型/模式通常有两种解决方法:一是重新运行算法;另一种就是增量的算法.通常前者的代价太大,因此如何设计增量的数据挖掘算法是当今的一个重要的挑战,增量的聚类算法[13~15]从不同角度提出了解决数据库数据变化的问题.

目前,对位置数据库结构的调整主要是考虑如何对小区(本文对小区(cell)和区域(zone)并不加以区分,表示一个唯一的注册地区,通过该注册地区可以唯一地确定移动用户的位置)进行划分,使得移动用户在同一划分中的小区之间移动频繁,而在不同划分的小区之间很少不或者不发生相互之间的移动.Badrinath 等人在文献[16]中第 1 次提出了根据用户的配置文件进行小区划分的思想,但是该文中的划分是针对单个用户的移动模式的,而且没有提供实现的算法.在考虑整个位置数据的组织结构时,从单个移动用户的角度是不合理的,应该考虑的是用户的整体行为特征.Das 和 Sen 在文献[17]中采用一个新的概念平均扇出率(average egress rate,简称 AER)来特征化用户的群体移动行为,然后根据 AER 来组织网络的层次结构的方法.小区的 AER 是指移动用户从该小区到其邻居小区的平均流出率;位置区域 LA(location area)的 AER 是指隶属于该 LA 的边界小区的 AER 的平均值.位置区域 LA 是由一群小区形成的簇,并且满足该簇中小区的 AER 的和小于给定阈值.算法首先将在小区级别上处理形成第 1 层的 LA,然后对 LA 循环处理,树状层次的每一层都是基于前一层的 LA 而形成,一直到最后没有合并后 AER 仍然小于给定阈值的 LA,最后生成一个节点作为树的根.

马帅等人在文献[18]中将小区划分的问题抽象为聚类问题,从数据挖掘的角度考虑群体移动用户的移动模式,提出了一种新的层次聚类算法,并且将此算法应用到位置数据库结构的动态调整.文献[17,18]中的方法具有一定的相似性,前者注重的是不同簇之间的流动性小,也就是松耦合,而后者从聚类的角度同时考虑不同聚类之间的松耦合与聚类内部之间的紧耦合.另一个区别在于,后者只是在小区的层次上对小区来聚类,从实现的角度,完全重新构造网络的拓扑结构是个相当复杂的工作,会严重影响系统当前的运行状况,影响系统的服务质量;而提供一种辅助位置数据库的动态调整手段来针对小区进行部分调整的策略是可行的.

随着时间的迁移,用户的移动模式会发生改变,原因有很多,有可能是用户兴趣的改变,也有可能是国家基础设施的建设,比如:公交车路线、道路、地铁路线的增加与调整,因此,小区的原有划分就有可能与当前的移动模式不符,不能正确地反映移动用户的群体行为特征.这就需要我们建立新的划分,产生新划分的办法有两种:一种是重新划分;另一种方法是动态调整当前划分.重新划分意味着抛弃以前的工作,从头再来,通常需要付出很大的代价.本文提出了一种增量的聚类算法来解决动态区域划分问题,以较小的代价调整当前的划分,使得新得到的划分仍然满足区域划分需要满足的条件.

本文第 1 节形式化描述区域划分和动态划分问题.第 2 节简述 Graph_Clustering 聚类算法.第 3 节详细介绍增量聚类算法 Graph_Clustering+.第 4 节是实验与性能分析.第 5 节是对全文的总结.

1 问题的提出

1.1 区域划分

将区域抽象为图的顶点,如果移动用户在两个区域之间移动频繁,则它们之间对应有一条边.从而将区域划分问题形式地描述如下,给定无向图 $G=(V,E)$和参数$\alpha\in[0,1]$,生成图 G 的子图集 $SubGs=\{G_1,G_2,\ldots\}$.用$G_i[V]$,$G_i[E]$分别表示子图 G_i 的点集和边集;$|\cdot|$表示集合的势,即集合中元素的个数.子图集 $SubGs=\{G_1,G_2,\ldots\}$需要满足下面的条件:

- 对于$\forall G_i\in SubGs$,满足 $G_i[V]\subseteq G[V]$和 $G_i[E]\subseteq G[E]$
- 对于$\forall G_i,G_j\in SubGs$,满足 $G_i[V]\cap G_j[V]=\varnothing$
- 对于 $SubGs=\{G_1,G_2,\ldots\}$,满足 $G_1[V]\cup G_2[V]\cup\ldots=G[V]$
- 对于$\forall G_i\in SubGs$,满足$|G_i[E]|\geq 1/2*\alpha*|G_i[V]|*(|G_i[V]|-1)$
- 对于$\forall G_i,G_j\in SubGs$,$G_i\cup G_j$后生成的图不满足上面的条件

第 1 个条件可以保证数据的有效性,生成结果集是图 $G=(V,E)$的子图集;第 2 个条件保证生成的子图互不相交;第 3 个条件保证聚类的完备性;第 4 个条件通过参数α来控制生成子图的紧密程度,当$\alpha=1$ 时,生成的子图是完全图;第 5 个条件保证生成的子图是极大子图,即结果集中任意两个子图合并后生成的图不满足前 4 个条件.

1.2 动态区域划分

用户移动行为的改变在图 G 的反映如下:

- 导致移动用户在一些区域间$(u,v\in V,u\neq v)$的由频繁变为不频繁,对应在无向图 $G=(V,E)$中删除连接该区域之间的边,即,$G'=(V,E-\{(u,v)\})$
- 导致移动用户在一些区域间$(u,v\in V,u\neq v)$的由不频繁变为频繁,对应在无向图 $G=(V,E)$中增加该区域之间的边,即,$G'=(V,E\cup\{(u,v)\})$

图的结构改变后,子图集 $SubGs=\{G_1,G_2,\ldots\}$就有可能不满足划分的条件,解决的办法有两种:一种是重新划分,生成新的子图集 $SubGs'$;另一种方法是动态地调整来维护子图集,使得调整后的子图集 $SubGs'$满足划分的条件.动态区域划分问题就是,在图的边发生变化时,如何以较小的代价调整当前的划分,使得新得到的划分仍然满足区域划分需要满足的条件.

2 Graph_Clustering 聚类算法

增量聚类算法基于 Graph_Clustering 算法[18],该算法正确地解决了区域划分问题.在许多层次聚类算法中将数据模型化为图,ROCK[5]采用稀疏图来模型化数据;CHAMELEON[6]采用 K-NEAREST 邻居图表示稀疏图;AMOEBA[7]采用 DELAUNAY 图来模型化数据.结合数据特征,CURE[4]则隐含地采用了图的思想.Graph_Clustering 算法同样采用图来模型化数据,并且扩展这种模型化数据的方法使其应用到聚类算法中以用来表示聚类.

定理 1. 基于图的聚类算法正确地解决了在第 1.1 节提出的区域划分问题.

证明:由算法的描述[18]可知,初始聚类集中的聚类是图 $G=(V,E)$中的顶点,聚类的判断标准保证初始聚类集

中的任意聚类满足 $Goodness(C_i)\geq\alpha$,容易看出,初始聚类集除了条件 5 外,其他的条件都满足;此后每次循环的聚类合并保证 $Goodness(C_i,C_j)\geq\alpha$,这样就保证了除条件 5 之外的其他条件;算法的终止条件保证了条件 5,从而算法的最终聚类集满足在第 1.1 节中区域划分的所有条件,定理 1 得证. □

3 Graph_Clustering$^+$增量聚类算法

3.1 数据增量模型

增量维护通常考虑两种增量模型[13]:一是目前为止所有的数据,我们称之为 FDM(full data model)模型;二是数据的一部分,如,时间窗口(sliding window)内的数据,我们称之为 SDM(selective data model)模型.这两种数据的增量模型对应于不同的应用需求.对数据库的更新操作包括增加、删除和修改,由于修改看做是先删除后增加的操作,我们把数据库的更新变化看做是首先对其进行批处理删除操作,从数据库删除一些数据;然后对其进行批处理增加,向数据库中插入数据.这样对于 FDM 数据模型和 SDM 模型,我们都采用如图 1 所示的数据模型加以形象的表示.其中,D,D',D^-,Δ^-,Δ^+分别表示原始数据库、更新后的数据库、未发生变化的数据、删除的数据和增加的数据.因此,一种好的增量算法要能够适应如图 1 所示的增量数据模型.

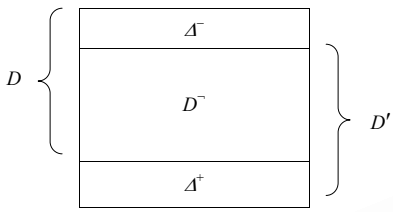

Fig.1 Incremental data model
图 1 增量数据模型

3.2 图的增量维护

图的顶点是对区域的抽象,顶点之间的边表示移动用户在对应的两个区域之间移动频繁.如果移动用户在该区域之间的移动次数超过一定阈值(如平均移动次数),那么我们说移动用户在该区域之间的移动是频繁的.由于我们采用图来模型化数据,需要对图进行增量的维护,假定区域没有发生变化,也就是图的顶点没有发生变化,因此可以说对图的增量维护就是对图的边进行动态的维护.

移动次数是从移动日志〈*old_zone*,*new_zone*〉中统计得到的,我们用 $F(u,v)$表示移动用户在区域 u,v 之间的移动次数,则有:

$$F_{D'}(u,v)=F_D(u,v)-F_{\Delta^-}(u,v)+F_{\Delta^+}(u,v).$$

通过增量的方式维护移动次数,就可以动态地维护图的结构,在此我们并不做过多的描述.

3.3 聚类的增量维护算法

由于图中边的增加和减少,导致当前的划分可能不满足第 1.1 节中区域划分的条件,本节介绍聚类增量维护算法,该算法动态地维护当前的划分,正确地解决了第 1.2 节提出的动态区域划分问题.

3.3.1 边的批处理

Graph_Clustering 聚类算法采用循环优化的方法,首先生成初始图,然后把图的每一个顶点当作一个聚类,采用自底向上(bottom-up)的层次聚类算法,每次聚类的结果集作为下一次聚类的输入,一直到聚类的结果不再改变为止.而过程 Clustering()的输入始终保持当前的聚类集满足区域划分的前 4 个条件.

如果每条边单独处理,使得每条边处理后的聚类集都满足区域划分的前 5 个条件,那么由于频繁地对聚类集进行调整,算法的效率将会很低.采用批处理的方法可以解决这个问题,其主要思想就是将条件放弱,使得每条边处理后的聚类集满足区域划分的前 4 个条件.

表 1 给出了边的变动对区域划分前 4 个条件的影响情况分析.从该表可知,我们可以将边分为 3 类:增加的边、删除不同聚类之间的边和删除同一聚类之间的边.我们可以先对增加的边和删除不同聚类之间的边进行批量处理,因为它们的变动并不影响聚类的前 4 个条件.然后我们处理删除同一聚类的边,由于这种边的删除可能会导致聚类不满足区域划分的前 4 个条件,这种情况下,每条边都要单独处理,使其满足聚类的前 4 个条件.

Table 1 Effect of the edge changing to the first four conditions of zone partition

表 1 边的变动对区域划分前 4 个条件的影响

Edges	Within same cluster	Connecting different clusters
Add	√	√
Delete	×	√

定理 2. 经过批处理后的聚类集满足区域划分的前 4 个条件.

证明:批处理首先处理增加边,从表 1 中容易知道边增加后,变动后的聚类集满足区域划分的前 4 个条件;然后处理那些边的类型为删除且属于连接不同聚类之间的边,从表 1 可知,变动后的聚类集满足区域划分的前 4 个条件;最后处理那些边的类型为删除且属于同一聚类内部的边,由于保证了每条边处理后的聚类集满足前 4 个条件,因此容易知道变动后的聚类集仍然满足区域划分的聚类前 4 个条件.由于批处理的每一步保持了区域划分的前 4 个条件,易知批处理后的聚类集满足区域划分的前 4 个条件,定理 2 得证. □

批处理的优点在于,使得 Graph_Clustering$^+$增量聚类算法避免从头开始聚类,即,将图的每个顶点作为一个单独的聚类,而是以较小的代价调整当前的聚类集而得到满足前 4 个条件的聚类集.Graph_Clustering$^+$算法将调整后的聚类集作为过程 Clustering()的输入,循环迭代处理一直到满足区域划分所有的 5 个条件为止,从而提高了效率.

3.3.2 数据结构

由于在一个聚类中边的减少使其可能不满足区域划分的前 4 个条件,从而需要分裂当前聚类.我们采用的策略是将聚类中贡献小的顶点分裂出去形成包含一个顶点的单独聚类,循环处理直到当前聚类满足区域划分的前 4 个条件为止.所谓顶点的贡献是指该顶点的与聚类内部其他顶点的关联度,为了提高聚类分裂的效率,我们需要在聚类中动态维护其贡献.这样,新的数据结构如图 2 所示.其中 Id 存储聚类号;Vertex 存储聚类中的所有顶点的 id;verDegree 用来存储对应 Vertex 中各顶点的关联度,即顶点与其他顶点之间的边的数目;Neighbor 用来存储 Vertex 的邻居顶点;neiDegree 用来存储对应邻居顶点的关联度,即聚类与各对应邻居的边的数目;numOfVertex 存储顶点 Vertex 的数目;numOfEdge 存储顶点 Vertex 内部边的数目.

```
Class Node {
    Integer Id
    Array Vertex
    Array Neighbor
    Array verDegree
    Array neiDegree
    Integer numOfVertex
    Integer numOfEdge
}
```

Fig.2 Data structure

图 2 数据结构

3.3.3 聚类判断标准

采用图趋向于完全图的程度来描述一个聚类内部顶点的紧密程度,因为当一个聚类中 C_i 的顶点联系紧密时,则由 C_i 所形成的子图在图 Graph 中就越趋向于完全子图,这样我们采用聚类 C_i 中的边的数目与由 C_i 的顶点组成完全图的边的数目的比率作为判断一个聚类好坏的标准:

当$|C_i|$=1 时:$Goodness(C_i)\geq\alpha$;

当$|C_i|\geq$2 时: $Goodness(C_i)=C_i.numOfEdge/C^2_{C_i.numOfVertex}$.

用 $Co_link(C_i,C_j)$表示两个聚类中相邻顶点对的数目,即在图 Graph 中将两个聚类连接在一起的边的数目.这样,相应的两个聚类 C_i 和 C_j 合并后好坏的判断标准是:

$$Goodness(C_i,C_j)=(C_i.numOfEdge+C_j.numOfEdge+Co\_link(C_i,C_j))/C^2_{C_i.numOfVertex+C_j.numOfVertex}.$$

只有满足 $Goodness(C_i,C_j)\geq\alpha$的两个聚类才能合并.我们在算法中通过阈值$\alpha$来控制聚类中顶点的紧密程度.当图 Graph 中的顶点联系比较少时,相应的α设定得应小一些.反之,当图中的顶点联系比较紧密时,相应的α设定得应大一些.

由于数据结构发生了变化,对应的聚类合并算法也需要相应的变化,需要增加内部顶点关联度的处理,假定我们需要合并聚类 C_i 和 C_j 生成聚类 C.

3.3.4 聚类合并算法

$C.Vertex=C_i.Vertex\cup C_j.Vertex$

$C.Neighbor=C_i.Neighbor\cup C_j.Neighbor-C.Vertex$

$C.verDegree=C_i.verDegree\cup C_j.verDegree\cup((C_i.Neighbor\cup C_j.Neighbor)\cap C.Vertex).neiDegree$

$C.neiDegree=C_i.neiDegree \cup C_j.neiDegree-C.Vertex.neiDegree$

$C.numOfVertex=|C.Vertex|$

$C.numOfEdge=C_i.numOfEdge+C_j.numOfEdge+1/2*((C_i.Neighbor \cup C_j.Neighbor) \cap C.Vertex).neiDegree$

其中,合并后的聚类 C 的顶点是聚类 C_i 和 C_j 顶点的并集;聚类 C 的邻居是聚类 C_i 和 C_j 邻居的并集,然后去除那些合并后成为内部顶点的邻居顶点;聚类 C 顶点之间的关联度是聚类 C_i 和 C_j 顶点关联度的并集,然后将对应顶点的关联度加上那些成为内部顶点的邻居顶点的关联度;聚类 C 与其邻居的关联度是聚类 C_i 和 C_j 关联度的并集(如果有相同的邻居,则对应关联度的和作为合并后的关联度),然后去除那些成为内部顶点的关联度;聚类 C 内部顶点之间边的数目 *numOfEdge* 是聚类 C_i 和 C_j 的 *numOfEdge* 之和,然后加上那些合并后成为内部顶点的邻居顶点的关联度之和的 1/2(由图论易知边的数目是关联度的 1/2).

合并聚类时有两个因素要加以考虑,一个因素是合并的两个聚类的共同的邻居 $Co\_neighbor(C_i,C_j)$;另一个因素是合并的两个聚类之间的联系 $Co\_link(C_i,C_j)$.第 1 个因素,使得我们的聚类算法充分考虑了空间聚类的特性,即:顶点的邻居对聚类的最终结果的影响,使得聚类的合并趋向更加有利;第 2 个因素保证合并聚类相互之间联系紧密.总之,需要在满足 $Goodness(C_i,C_j) \geq \alpha$的基础上,保证 $Co_neighbor(C_i,C_j)$最大的两个聚类合并.在图 3 中,假定当前顶点 0,1,2,3,顶点 4,5,6,7 和顶点 8,9 分别构成聚类 C_0,C_1,C_2,则对于聚类 C_2,$Co\_link(C_2,C_0)=2$,$Co\_link(C_2,C_1)=4$,所以聚类 C_2 和 C_1 合并($Co\_neighbor(C_2,C_0)=Co\_neighbor\ (C_2,C_1)=\varnothing$).表 2 给出了图 3 中各聚类以及聚类 C_1 和 C_2 合并后的数据结构.

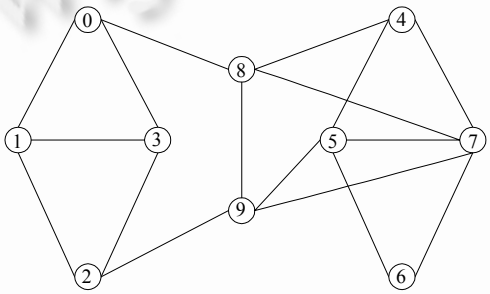

Fig.3 Example of clustering
图 3 聚类示例

Table 2 Data structure of clusters and merging of two clusters

表 2 聚类的数据结构与两聚类的合并

Id	Vertex	Neighbor	verDegree	neiDegree	numOfVertex	numOfEdge
C_0	{0,1,2,3}	{8,9}	{2,3,2,3}	{1,1}	4	5
C_1	{4,5,6,7}	{8,9}	{2,3,2,3}	{2,2}	4	5
C_2	{8,9}	{0,2,4,5,7}	{1,1}	{1,1,1,1,2}	2	1
$C_1 \cup C_2$	{4,5,6,7,8,9}	{0,2}	{3,4,2,5,3,3}	{1,1}	6	10

3.3.5 聚类过程 Clustering()

我们发现,采用文献[18]中 Clustering()过程将会导致最后聚类集中各聚类中顶点数目分布不均匀,那是因为每次合并考虑的是在已经合并的聚类集 ClusterResultSet 中查找一个能够合并的聚类,使得某些聚类过于庞大,细节参见文献[18].在这里我们合并的是 ClusterInitialSet 中的聚类,从随后的实验中可以发现取得了相当好的效果.基于 Clustering()过程的重要性,下面我们较为详细地介绍该过程.

过程 Clustering(ClusterInitialSet,α)首先将聚类的结果集 ClusterResultSet 赋为$\varnothing$,然后逐个考虑初始聚类集 ClusterInitialSet 中的第 1 个聚类 C_i(*ClusterInitialSet.size*>0),同时将该聚类从 ClusterInitialSet 删除.如果在 ClusterInitialSet 找到另外一个聚类 C_j($0 \leq j <$*ClusterInitialSet.size*)满足条件 $Goodness(C_i,C_j)$,并且两个聚类的共同邻居 $Co_neighbor(C_i,C_j)$最大,则合并聚类 C_i 和 C_j,并将合并后的聚类 C 加入聚类结果集 ClusterResultSet;否则将 C_i 作为一个新的聚类 C 加入聚类结果集 ClusterResultSet,循环处理直到 ClusterInitialSet 中没有聚类为止.

下面我们详细地讨论不同类型变动边的处理算法,假定当前边为(u,v),并且顶点 u 属于聚类 *Cluster*_1,顶点

v 属于聚类 *Cluste*r_2.

3.3.6 增加边

如果对当前的图增加一条边,那么这条边有两种不同的情况:

- 连接着两个不同的聚类,即 *Cluster*_1≠*Cluster*_2
- 在一个聚类的内部,即 *Cluster*_1=*Cluster*_2

第 1 种情况使得两个聚类之间的关联度增加,对于聚类 *Cluster*_1,如果其邻居已经包含顶点 v,则仅仅需要对应邻居的关联度增加 1;如果其邻居中未包含顶点 v,则需要增加 v 为其邻居,对应的邻居关联度为 1.采用同样的方法处理聚类 *Cluster*_2.第 2 种情况使得当前的聚类对划分的第 4 个条件的满足性增强.由于增加的边属于聚类的内部,因此仅仅需要对 u 和 v 的顶点关联度增加 1.

3.3.7 删除边

如果对当前的图删除一条边,那么这条边同样也有两种不同的情况:

- 连接着两个不同的聚类,即 *Cluster*_1≠*Cluster*_2
- 在一个聚类的内部,即 *Cluster*_1=*Cluster*_2

如果边是属于不同聚类之间的,对边的删除意味着降低了不同聚类之间的联系,容易知道聚类对删除该边后的图仍然满足区域划分的条件,当然也就满足区域划分的前 4 个条件.对聚类 *Cluster*_1,需要对应邻居 v 的关联度减 1;如果对应的邻居关联度变为 0,则需要将邻居 v 及其关联度删除.采用同样的方法处理聚类 *Cluster*_2.

如果边是属于聚类内部的,对边的删除意味着降低了一个划分内部的联系,使得聚类未必满足区域划分的条件 4.如果该聚类仍然满足条件,那么容易知道当前的聚类对删除该边后的图仍然满足区域划分的条件.这种情况下,仅仅需要在顶点 u 和 v 的顶点关联度减 1.

如果聚类不满足区域划分的条件 4,就需要对该聚类进行分裂,使得新生成的聚类集中的每个聚类都满足区域划分的条件 4.我们采用的方法是将贡献小的顶点依次从该聚类中删除,然后将该顶点作为一个单独的聚类,这个过程一直到该聚类满足条件 4.需要注意的是,由于聚类的分裂,使得一些本来是删除同一聚类内部的边变为删除不同聚类之间的边,因此需要动态地判断边的删除类型.

下面给出聚类的分裂算法,假定我们需要分裂聚类 C_i,并且顶点 u 是贡献最小的顶点,且 u 的贡献为 w.我们根据图来生成仅仅包含顶点 u 的聚类 C_u.下面的算法是关于如何从聚类 C_i 中删除 C_u,生成新的聚类 C.

3.3.8 聚类分裂算法

由于聚类分裂算法无法采用和第 3.3.3 节一样的方式加以形式的描述,我们就对分裂算法进行描述,可以看出聚类 C 的生成仅仅需要聚类 C_u 和 C_i,而不需要额外的知识(如访问图),从而提高了算法的效率.

分裂后聚类 C 的顶点是聚类 C_i 顶点减去顶点 u.聚类 C 的邻居的关联度是聚类 C_i 的邻居的关联度减去聚类 C_u 的邻居的关联度,然后去除那些关联度为 0 的、其邻居为 C_i 的邻居减去关联度为 0 的邻居.如果顶点 u 的贡献 w 大于 0,则增加 u 作为聚类 C 的邻居,并且其邻居的关联度为 w.计算聚类 C 顶点之间的关联度,首先将 C_i 的顶点关联度中除去顶点 u 的,然后将是 C_u 的邻居的顶点的关联度减 1.聚类 C 内部顶点之间边的数目 *numOfEdge* 是聚类 C_i 的 *numOfEdge* 减去 w.顶点的数目是 C_i 顶点的数目减去 1.需要注意的是,如果分裂后聚类 C 仍然不满足区域划分的条件 4,则需要继续对聚类 C 进行分裂.表 3 给出了采用聚类分裂算法将顶点 4 从表 2 中的聚类 C_1 分裂出去的结果,其中聚类 C_3 是根据图 3 计算出来的,而聚类 C_4 是根据聚类 C_1 和 C_3 计算出来的.

Table 3 Split of a cluster

表 3 聚类的分裂

Id	Vertex	Neighbor	verDegree	neiDegree	numOfVertex	numOfEdge
C_1	{4,5,6,7}	{8,9}	{2,3,2,3}	{2,2}	4	5
C_3	{4}	{5,7,8}	{0}	{1,1,1}	1	0
C_4	{5,6,7}	{8,9,4}	{2,2,2}	{1,2,2}	3	3

综上所述,Graph_Clustering$^+$增量聚类算法的主要思想是对图的增量维护和对聚类的动态调整,充分利用了已有的挖掘结果,从而有效地提高了算法的效率.对聚类的动态维护主要是通过分析边变动对聚类的影响,采用批处理的方法调整当前的聚类集,使其满足区域划分的前 4 个条件,然后通过过程 Clustering()循环迭代处理

调整后的聚类集,直到满足区域划分的所有条件.算法也对 Clustering()过程做了改进,使得聚类的结果更加合理.

4 实验与性能分析

我们采用 Stanford 大学无线网络小组开发的仿真程序产生的测试数据 SUMATRA(Stanford University Mobile Activity TRAces)[19].SUMATRA 包含 90 个区域的海湾地区的位置信息.我们通过合成,生成了包含 90 个区域的 637K 行数据和包含 18K 个区域的 20M 行数据.数据的格式为⟨*old_zone*,*new_zone*⟩,其中 *old_zone* 为移动前的区域,*new_zone* 为移动后的区域.

实验的硬件环境是兼容机:1.7GHz 的 Pentium 4 CPU、主存为 512M、硬盘为 20G;软件环境是:操作系统为 Microsoft Windows 2000 Professional,算法采用 Java 编写.

整个实验分为 3 部分:第 1 部分验证本文中对过程 Clustering()改进后对聚类效果的改善;第 2 部分进一步地测试控制参数α对算法效率的影响;第 3 部分是增量聚类算法 Graph_Clustering$^+$与聚类算法 Graph_Clustering 的性能对比情况.

表 4 给出了对包含 90 个区域的 637K 行数据分别采用文献[18]中的和本文中的 Clustering()过程的实验结果.由于划分中区域个数期望分布得比较均匀,因为如果分布不均匀则会导致系统的负载不均衡,从而使得有的设备资源浪费,有的设备资源紧缺.正如我们在第 3.3.5 节中的分析,实验结果验证了采用新的方法会使聚类中元素个数的分布更加均匀,聚类的效果要明显地好于原来的方法.

Table 4 Comparison of clustering results' quality

表 4 聚类效果比较

Control parameter	Clustering procedure	Cluster number	Element number of clusters	Mean number
α=0.1	New	9	10,8,6,13,13,12,12,6,10	10
	Original	13	13,20,9,1,1,1,11,1,1,1,1,13,17	7
α=0.15	New	15	8,7,6,6,7,6,6,6,6,4,4,7,6,6,5	6
	Original	18	12,13,1,1,1,5,7,10,1,8,1,1,1,1,1,1,12,13	5

我们对数据量为 20M、区域个数为 18K 和数据量为 637K、区域个数为 90 的数据采用聚类算法来测试控制参数α对算法执行效率的影响.从图 4 中可以看出,参数α对算法执行效率的影响很小.同时我们也可以看出,在大量移动日志的情况下,聚类算法始终保持着良好的性能,具有良好的扩展性.

我们在$|D|=|D'|=16M,|\Delta^-|=|\Delta^+|=2M$ 的情况下,对增量聚类算法 Graph_Clustering$^+$和聚类算法 Graph_Clustering 测试随着控制参数的改变其效率的对比.从图 5 可以看出,算法的性能对比情况比较稳定,其原因是算法的效率受控制参数的影响较小,也可以看出 Graph_Clustering$^+$算法的效率要比 Graph_Clustering 的效率高得多.

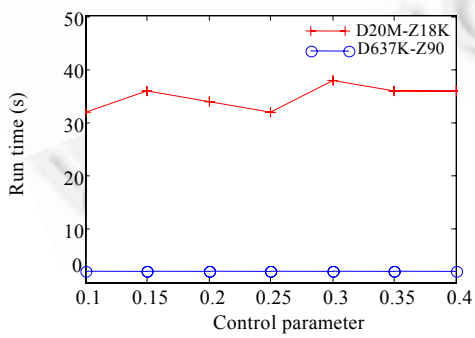

Fig.4 Effect of control parameter to the algorithm's efficiency

图 4 控制参数对算法性能的影响

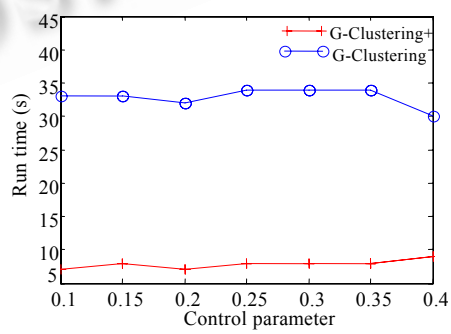

Fig.5 Comparison of algorithms' efficiency with varying control parameter

图 5 算法性能随控制参数改变的对比

我们在$|D|=|D'|=10M,|\Delta^-|=|\Delta^+|,\alpha=0.1$ 的情况下,通过变换$|\Delta^-|=|\Delta^+|$数量的大小(1M~10M),对增量聚类算法 Graph_Clustering$^+$和聚类算法 Graph_Clustering 进行了性能的对比实验.从图 6(a)可以看出,0.6 是分水岭,即当

$|\Delta^-|=|\Delta^+|$小于$|D|$的 60%的时候,增量维护算法 Graph_Clustering$^+$的代价要小于聚类算法 Graph_Clustering;而当$|\Delta^-|=|\Delta^+|$大于$|D|$的 60%的时候,增量维护算法 Graph_Clustering$^+$的代价要大于聚类算法 Graph_Clustering.当$|\Delta^-|=|\Delta^+|$是$|D|$的 50%的时候,图的动态维护和非动态维护的代价是基本一样的,因为我们需要扫描同样多的数据;当$|\Delta^-|=|\Delta^+|$小于$|D|$的 50%的时候,图的动态维护代价要小于非动态维护的代价;而当$|\Delta^-|=|\Delta^+|$大于$|D|$的 50%的时候,图的动态维护代价要大于非动态维护的代价.而当$|\Delta^-|=|\Delta^+|$是$|D|$的 60%的时候,虽然图的增量维护增加了代价,但是由于聚类动态调整的代价要小于重新聚类的代价,二者的影响抵消,从而使得 60%成为分水岭.

我们对图的增量维护进行了改进,当Δ^-和Δ^+数量之和大于 D'的数量时,我们通过直接扫描 D'来计算新的图;当Δ^-和Δ^+数量之和小于 D'的数量时,我们仍然采用第 3.2 节的方法动态地维护图.通过图 6(b)我们可以看出,改进后增量维护算法 Graph_Clustering$^+$的代价要小于聚类算法 Graph_Clustering,尤其是数据的变化不大的时候($|\Delta^-|=|\Delta^+|$的变化在 10%~30%),增量聚类算法的性能要高得多.即使新的数据 D'和原来的 D 数据完全不同($|\Delta^-|=|\Delta^+|=|D|=|D'|,|D^-|=0$).原因在于移动用户的群体移动特征虽然发生改变,但是变化往往是渐进的,或者只是其中部分用户的移动特征发生了改变,从而群体的移动特征变化得不是很明显.这正是增量聚类维护算法的优越性所在.

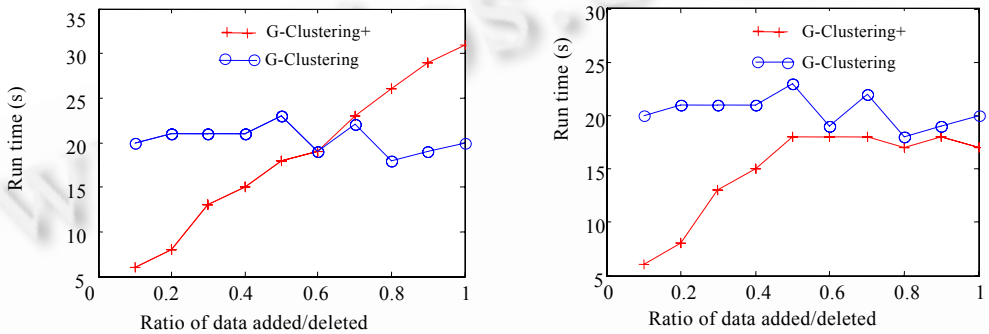

Fig.6 Comparison of algorithms' efficiency with varying added/deleted data

图 6 算法性能随数据改变的对比

5 结 论

将数据挖掘应用到移动计算环境中是一项具有挑战性的研究课题,具有广阔的应用前景.本文提出了一种增量的聚类算法来解决动态区域划分问题,以较小的代价调整当前的划分,使得新得到的划分仍然满足区域划分需要满足的条件.增量聚类算法通过对图的增量维护和划分的动态调整使得算法的效率大大提高,同时也改进了算法的聚类效果,使得得到的区域划分更为合理.实验证明了上述结论.

References:

[1] Wong VWS, Leung CM. Location management for next generation personal communication networks. IEEE Network, 2000,14(5): 18~24.

[2] Han JW, Kambr M. Data Mining Concepts and Techniques. Beijing: Higher Education Press, 2001. 335~393.

[3] Ng R, Han J. Efficient and effective clustering method for spatial data mining. In: Bocca JB, Jarke M, Zaniolo C, eds. Proc. of the 20th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publisheers, 1994. 144~155.

[4] Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases. In: Haas LM, Tiwary A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1998. 73~84.

[5] Guha S, Rastogi R, Shim K, ROCK: A robust clustering algorithm for categorical attributes. In: Proc. of the 15th Int'l Conf. on Data Engineering. IEEE Computer Society, 1999. 512~521.

[6] Karypis G, Han E-H, Kumar V. Chameleon: Hierarchical clustering using dynamic modeling. IEEE Computer, 1999,32(8):68~75.

[7] Estivill-Castro V, Lee I. AMOEBA: Hierarchical clustering based on spatial proximity using delaunay diagram. In: Forer P, Yeh AGO, He J, eds. Proc. of the 9th Int'l Symposium on Spatial Data Handling. Hong Kong: Study Group on Geographical Information Science of the International Geographical Union, 2000. 7a.26~7a.41.

[8] Ester M, Kriegel HP, Sander J, Xu X. A density based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis E, Han JW, Fayyad UM, eds. Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 226~231.

[9] Ma S, Wang TJ, Tang SW, Yang DQ, Gao J. A new fast clustering algorithm based on reference and density. In: Dong GZ, Tang CJ, Wang W, eds. Proc. of the WAIM Conf. Heidelberg: Springer-Verlag, 2003. 214~225.

[10] Wang W, Yang J, Muntz R. STING+: An approach to active spatial data mining. In: Proc. of the 15th Int'l Conf. on Data Engineering. IEEE Computer Society, 1999. 119~125.

[11] Aggrawal R, Gehrke J, Gunopulos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In: Jagadish HV, Mumick IS, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1996. 94~105.

[12] Fasulo D. An analysis of recent work on clustering algorithms. Technical Report, 01-03-02, Seattle: Department of Computer Science and Engineering, University of Washington, 1999. http://citeseer.ist.psu.edu/fasulo99analysi.html

[13] Ganti V, Gehrke J, Ramakrishnan R. DEMON: Mining and monitoring evolving data. In: Proc. of the 16th Int'l Conf. on Data Engineering. IEEE Computer Society, 2000. 439~448.

[14] Charikar M, Chekuri C, Feder T, Motwani R. Incremental clustering and dynamic information retrieval. In: Proc. of the 29th Annual ACM Symp. on the Theory of Computing. New York: ACM Press, 1997. 626~635.

[15] Ester M, Kriegel H-P, Sander J, Wimmer M, Xu XW. Incremental clustering for mining in a data warehousing environment. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 1998. 323~333.

[16] Badrinath BR, Imielinski T, Virmani A. Locating strategies for personal communications networks. In: Proc. of the Int'l Workshop on Networks for Personal Communications. IEEE Computer Society, 1992. II.1-II.9.

[17] Das SK, Sen SK. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. The Computer Journal, 1999,42(6):473~486.

[18] Ma S, Wang TJ, Tang SW, Yang DQ, Gao J. Dynamic reorganization of location databases based on clustering. Journal of Software, 2003,14(5):963~969 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/963.htm

[19] SUMATRA: Stanford University Mobile Activity TRAces. http://www-db.stanford.edu/sumatra/

附中文参考文献:

[18] 马帅,王腾蛟,唐世渭,杨冬青,高军.基于聚类的位置数据库动态重组.软件学报,2003,14(5):963~969. http://www.jos.org.cn/1000-9825/14/963.htm