一种基于网格和密度的数据流聚类算法*)

高永梅[1] 黄亚楼[1,2]

(南开大学信息技术科学学院 天津 300071)[1] (南开大学软件学院 天津 300071)[2]

摘 要 在"数据流分析"这一数据挖掘的应用领域中,常规的算法显得很不适用。主要是因为这些算法的挖掘过程不能适应数据流的动态环境,其挖掘模型、挖掘结果不能满足实际应用中用户的需求。针对这一问题,本文提出了一种基于网格和密度的聚类方法,来有效地完成对数据流的分析任务。该方法打破传统聚类方法的束缚,把整个挖掘过程分为离线和在线两步,最终通过基于网格和密度的聚类方法实现数据流聚类。

关键词 聚类,网格,基于密度的聚类,数据流

A Grid and Density-based Clustering Algorithm for Processing Data Stream

GAO Yong-Mei[1] HUANG Ya-Lou[1,2]

(College of Information Science & Technology, Nankai University, Tianjing 300071)[1]

(College of Software, Nankai University, Tianjing 300071)[2]

Abstract In the field of data stream analysis, conventional methods seem not quite useful. Because neither they can adapt to the dynamic environment of data stream, nor the mining models and results can meet users' needs. A grid and density based clustering method is proposed to effectively address the problem. With this method, the mining procedure is divided into online and offline two parts and grid and density based clustering method is used to get final clusters for data stream.

Keywords Cluster, Grid, Density-based clustering, Data stream

1 引言

随着信息技术的迅猛发展和广泛应用,数据的采集、传输和共享等技术都已达到相当高的水平,各行各业都在以惊人的速度产生和积累着大量的数据流。有效地分析这些数据,发现其中隐藏的知识显得尤为重要。如通过跟踪网络中数据的流动变化来研究网络流通模式以及可能的非法入侵;再如在传感器的应用中,跟踪反馈数据的变化,来对外界环境做出正确的估计,从而做出正确的决策。

然而多数常规的聚类算法认为待挖掘的数据集是固定的,集中研究如何最小化处理时间及内存占用。由此数据流的动态特性对传统聚类提出了新要求。

1)在数据流分析过程中,数据一经处理,除非特意保存,否则不能被再次取出处理。

数据流规模大,完全存储需要很高的时间、空间代价。访问历史数据不能像静态数据那样随意进行检索和遍历。

2)数据点不停地流入数据量逐渐增大,算法中接收数据点所需的空间、时间不应随之快速膨胀。

4)挖掘结果可以反映出数据流聚簇随时间的变化规律。

数据点在时间轴上以一定速率顺序地产生,以形成数据流,分析数据流连续变化的规律很有价值。用户不光要知道某一时刻的聚类结果,也希望得到在某一个时间段中产生的聚簇结果、聚簇如何随时间变化,从而得到数据流的生成规律。

针对如上的问题,本文提出一种基于网格和密度的聚类算法(Grid and Density based clustering algorithm for analyzing Data Stream ,GDDS)来有效地完成对数据流的分析任务。

2 基于网格和密度的聚类算法

2.1 基于网格的聚类算法

基于网格的聚类把多维数据空间划分成一定数目的单元,然后在这种数据结构上进行聚类。其特点是处理速度独立于对象的数目,仅依赖于量化空间中的网格数。

2.2 基于密度的聚类算法

基于密度的聚类将簇看作是数据空间中被低密度区域分割开的高密度对象区域,其特点是能够发现任意形状的聚簇。

2.3 基于网格和密度的聚类算法

基于网格和密度的聚类正是结合以上两种聚类思想,其优点是能够很好地处理高维数据和大数据集并能发现任意形状的聚簇,典型算法有 CLIQUE 等。

本文算法 GDDS 属于基于网格和密度的聚类。GDDS 把数据空间划分为网格单元,然后在网格的基础上对稠密的单元进行聚类。网格结构如图 1 所示。

0.0	0.1	0.2	0.3
1.0	1.1	1.2	1.3
2.0	2.1	2.2	2.3
3.0	3.1	3.2	3.3

图 1 网格空间标号图

首先把数据空间每一维分成 m 个相等的小区间,得到由

*)教育部重点科学技术研究项目(02038);南开大学亚洲研究中心项目(AS0405)资助。**高永梅** 博士研究生,主要从事数据挖掘方面的研究;**黄亚楼** 博士生导师,教授,南开大学软件学院院长,主要从事智能控制与智能信息。

一定粒度的多维网格构成的数据空间。在 n 维数据空间中用一个多维数组 $D[d1][d2][d2]\cdots[dn]$ 来在逻辑上表示该网格结构。各维按顺序 $d1,d2,d3,\cdots,dn$ 排列，每一维上的小区间按边界值大小顺序排列。数组中每个元素都可对应到数据空间的某一个格子。我们为每一个格子定义一个字符串标号来标识它。

如图 1 所示，以二维数据对象为例，$ID(D[i][j])=i.j$。同理，对更高维的数据对象也可以得到这样的映射，对于每一个数据点都可以把它定位到空间对应的网格中。用 x_i、$x_i[j]$、tsi 分别表示数据流中第 i 个到达数据点、该数据点在第 j 维上的取值以及到达时间。每个格子的存储结构定义如下：

grid:$\langle id,count,\mu,\partial,t,t^2\rangle$

其中 id：格子的标号；$count$：格子内数据点的个数；μ、∂ 为 d 维向量，它们第 j 维上的分量为：$\mu[j]$、$\partial[j]$。

$$\mu[j]=\sum_{x_i\in\text{此格子}}x_i[j] \tag{1}$$

$$\partial[j]=\sum_{x_i\in\text{此格子}}x_i^2[j] \tag{2}$$

$$t=\sum_{x_i\in\text{此格子}}t_i \tag{3}$$

$$t^2=\sum_{x_i\in\text{此格子}}t_i^2 \tag{4}$$

μ、∂，作为统计信息通过简单的计算可得到格子内数据点的均值、方差、到达时间的均值、方差（μ'，∂' 代表均值和方差）

$$\mu'=\frac{\sum_{x_i\in\text{此格子}}X_i[j]}{count}=\frac{\mu[j]}{count} \tag{5}$$

$$\partial'=\sqrt{\frac{\sum_{x_i\in\text{此格子}}(X_i[j]-\mu')^2}{count}}=\sqrt{\frac{\partial[j]-2\mu'\mu[j]}{count}+\mu'^2} \tag{6}$$

同理，时间的均值和方差也可类似地推导得到。随着新数据点的到达，新信息很容易添加到原来的信息中。x^{new}，μ^{new}，∂^{new} 分别代表新到达的数据点、更新后的统计信息。更新过程如下：

$$coun=count+1 \tag{7}$$

$$\mu^{new}[j]=\mu[j]+x^{new}[j] \tag{8}$$

$$\partial^{new}[j]=\partial[j]+x^{new2}[j] \tag{9}$$

3 基于网格和密度的数据流分析算法

3.1 挖掘过程

本文紧紧围绕动态数据流分析这一核心问题，以满足用户的需求为目的。在这一前提下，将挖掘过程分为在线和离线两个部分：在线过程快速接收输入数据流，其产生的结果作为挖掘的中间结果维护起来，并且随着新数据点的流入该中间结果实时地更新。依据一定的时间框架周期性地选取某些特定时刻的中间结果，保存到外存作为离线过程的输入。离线过程由用户调用，针对用户的查询给出最终挖掘结果。通过这样两个协调运作的处理过程实现动态、快速地处理流式数据，并且通过设计恰当的数据流统计信息可以很好地满足用户对数据流分析的需求。挖掘模型如图 2。

1)在线过程

在线部分是一个永不停止的数据处理引擎，实时接收新数据，通过联机处理模块把更新结合到原来挖掘的中间结果上，形成中间知识的最新模型。同时根据一定的时间框架（如文[2]中金字塔时间框架），在一定时刻把该时刻的中间知识回写到外存的中间知识库中。这样，通过在线模块周期性的回写就形成了历史时刻中间知识的集合，即形成中间知识库。

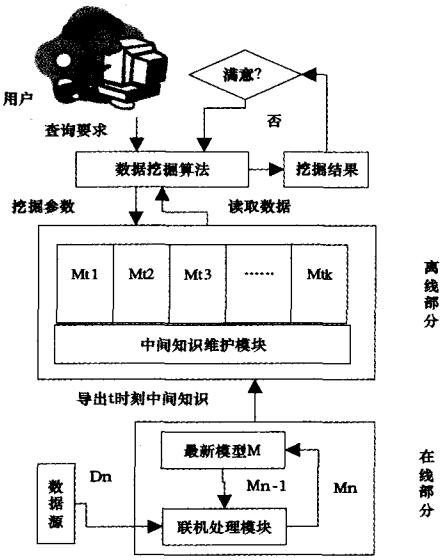

图 2 GDDS 数据流挖掘模型

随着数据点的流入，每个数据点将会按其本身的信息定位到对应的空间格子中，同时格子中的统计信息进行更新。在算法实现中，我们只存储有数据点的格子，并且建造哈希树，通过网格标号很容易访问到对应的格子。这样，对于高维数据会大大节省存储空间。有点的格子逐渐添加到格子列表 Table_Grid 中。Table_Grid 为算法中在线更新的中间知识，伪代码如下：

```
Online( p(x1,…,xd),c_time)
{
  id=get_id(p);
//检测标号为 id 的格子是否已添加到 Table_Grid
  if(G_exist(Table_Grid,id))
  {
    c_grid=G_get(Table_Grid,id);
    //检测当前格子中是否含有需剔出的孤立点
    if(! G_dense(c_grid))&&(! G_new(c_grid))
    //剔除当前格子中的孤立点
        {
            G_clear(Table_Grid,id);
        }
  }
else G_add(Table_Grid,id);
    G_update(id,p,c_time);
}
```

输入参数 $p(x_1,\cdots,x_d)$，c_time 分别代表新数据点及其到达的时刻；$get\_id(p)$ 用于得到数据点 p 所对应的格子标号；$G\_exist(Table\_Grid,id)$ 用来判断标号为 id 的格子是否存在于格子列表中。

$G\_dense(c\_grid)$ 用于判断格子是否稠密。首先求得这个格子所对应的密度值 $density$，当 $density>\tau$（τ 为用户定义阈值），该格子为稠密的。

$$density=\frac{\text{格子内数据点数}}{\text{数据空间数据点总数}} \tag{10}$$

$G\_new(c\_grid)$，格子内的数据点是否在时间上跨度很大，根据这两个条件剔除掉那些空间分布很稀疏且到达时间距现在很久远的孤立点；$G\_update(id,p,c\_time)$ 将新数据点的统计信息添加到对应的格子。

2)离线过程

离线部分是一个分析环境，用于得出用户所要的挖掘结果，其工作基础是在线部分积累的中间知识库。用户可以根据自己的需要设定参数和挖掘的时间窗口，最后得到对应时刻的挖掘结果，或给定时间段内到达数据点的聚簇情况，从而进一步研究数据流的变化过程。

A. 由某一时刻的中间知识得到结果簇。

从任意格子出发，采用深度优先遍历找到所有与当前格子连通的稠密的格子，形成一个簇。Unit：初始格子标号，num：当前要形成簇代号，D：数据空间的维数。代码如下：

```
Dfs(unit,num)
{
u. num=n;
for(j=1;j<D;j++)
  {
    //examine the left neighbor of unit in dimension j
    ul=id(j,unit,left);
    if (ul is dense) and (ul. clu_id is undefined)
    Dfs(ul,num);
//examine the right neighbor of unit in dimension j
    ur=id(j,unit, right);
if ( (ur is dense)&& (ur. clu_id is undefined))
Dfs(ur,num)
  }
}
```

B. 获得历史某一时间段的数据流聚簇结果，实现对数据流变化的分析。

该问题描述如下：

输入：时间段$(t_1,t_2)t_2>t_1$ 得出：(t_1,t_2)这一时间段所到来的数据点的聚簇情况。

解决方法：通过这两个时刻数据点的中间知识，得到这一时间段内数据点的聚簇情况。

T_1：t_1 时刻的中间知识网格。

T_2：t_2 时刻的中间知识网格。

T_{2-1}：(t_1,t_2)这一时间段所到来的数据点的中间知识网格。

逐个处理 T_1，T_2 中的格子，再添加到 T_{2-1} 中，具体过程如下：

· 对于 T_1，T_2 中共有的格子

T_2 格子对应的统计信息－T_1 中的；再添加到 T_{2-1}。

· T_1 有 T_2 中没有的

这样的格子是由噪声点形成的。在这个时间段内被剔除掉，因此没有必要添加。

· T_2 中有，T_1 中没有

很显然，这样格子内的点是在这一时间段内出现的，直接添加到 T_{2-1} 中。

最后在中间知识库 T_{2-1} 上进行如上 Dfs() 的聚类算法，即得结果。

3.2 关于孤立点的考虑

值得注意的一点是，在动态数据流环境中，孤立点的定义会与静态环境下不同，而相应的处理方式也不同。针对这一问题，本算法提出如下的解决办法：

中间知识要排除孤立点。孤立点是数据中非常强烈的噪声，这些数据不反映数据集的总体特性，反而会给数据集的统计信息中带来很大误差。数据流的孤立定义如下：

1)空间上的孤立性

孤立点出现在空间上的偏僻区域，体现空间上出现的偶然性。这与静态环境中的情况一样。

2)时间上的偶然性

因为在动态数据流的环境中，一个当前在空间上孤立的点，不能由此就断定它一定会是永远孤立的，永远不会形成集中区域。如果它在一个很长的超出考虑范围的时间段上一直保持孤立，那么认为这个点在那个时间上出现也是很偶然的。

满足上面两个条件的点认为就是孤立点，要剔除掉，因为按理论从时间上还是从空间上出现这样的点概率是很小的，它的位置信息和时间信息会给整个簇的统计信息带来偏差。

按照挖掘模型的过程，随着时间的不断推移，Table_Grid 不停地接收新数据点。同时，按照文[2]中提到的金字塔时间框架，周期性地把对应时刻的中间知识写到外存。这样，在外存就真正意义上地形成了中间知识库，用于离线过程的计算来满足用户的不同分析要求。

4 实验评价

4.1 算法性能分析

针对于引言中提出的数据流聚类应满足的要求对本文算法进行分析。

1)本算法在线接收数据流，能够及时把新数据添加到模型中，适合数据流高速流动的环境。

2)随着数据点不停地流入，数据量逐渐增大，算法中接收和处理数据流所需的时间和空间代价是和数据空间中非空网格个数处呈线性关系的，不会随着数据点的增加快速膨胀。

第一阶段在线接收数据流的过程中，时间复杂度为 $O(n)$，其中 n 为数据点的个数；离线阶段深度优先遍历密集网格，寻找连通的稠密网格集合形成聚簇。由于采用了哈希查找，因此时间复杂度为 $2km$，其中 k 为数据空间维数，m 为稠密网格数。

3)本算法只对原始数据点访问一次，而且基于数据流分析这一需求，建立了离线和在线的挖掘模型。依据一定的时间框架，把历史时刻的数据以一定浓缩形式保存到外存，设计简单合理的中间知识，能够对数据流随时间的变化进行分析。

3)本文算法是一个基于网格和密度的方法。初始网格粒度的选取对结果有影响，这是基于网格聚类的一个常见问题。实验中初始粒度的选取是基于内存可用空间的。所以说随着内存空间的增大，算法的准确性会更高。

4.2 实验结果分析

1)模拟数据

模拟数据流实验中提取出三个时刻 $t0,t1,t2$（$t0<t1<t2$）所对应的中间知识，如图 3 中 A、B、C。其中 $t0$ 时刻的数据总量为 1000 条，$t1$ 时刻为 1400 条，$t2$ 为 2200 条。

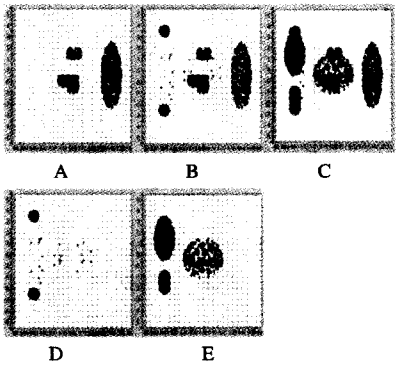

图 3 模拟数据挖掘结果

图 3 中 D、E 分别反映出[$t0,t1$]、[$t1,t2$]这两个时间段内数据点的到达情况。它们分别是由 $t1$ 时刻的中间知识库与 $t0$ 时刻的相减，$t2$ 时刻的中间知识库与 $t3$ 时刻的相减得到的。挖掘时间如表 1。

表 1 时间单位/ms

数据点数	T0 时刻 (0+1000)	T1 时刻 (1000+400)	T2 时刻 (1400+850)
在线接收	113(ms)	140(ms)	187(ms)
离线聚类	997(ms)	1086(ms)	1200(ms)

由此我们可以看出这种基于中间知识库的挖掘模型很容

易得到数据流在某一时间段内的聚簇情况。

2)实际数据

在实际数据及实验中,我们采用了 UCI 数据集的 KDD Cup 1999 Data,该数据集共 4898431 条,我们取其中的 1%这一部分。这一数据集共有 23 类。其中 smurf, normal 和 neptune 这三类数量明显很多,占总数据的 98%。我们取出这三类作为实验数据。在实验中,我们把数据集三等分,每隔 16200 条数据保存一次中间知识到硬盘。最终挖掘结果正确性比较如表 2。

表 2 实际数据挖掘结果

真实分布		挖掘结果	
类名称	记录数	类标号	记录数
normal	9727	1	11641
Smurf	28079	2	24015
Neptune	10720	3	12870

表 3 中显示出在数据流的三次添加过程中的离线和在线的处理时间,同时与处理静态数据集的经典算法 K-Means、Birch 进行时间效率比较。

在此我们可以看到,GDDS 算法在线吸收数据点过程所用时间基本是和数据点个数呈正比例关系;在离线聚类过程

表 3 时间单位/s

数据点数	0→1/3	(1/3→2/3)	(2/3→全部)
GDDS 在线	2.4(s)	2(s)	1.9(s)
GDDS 离线	72(s)	93(s)	107(s)
K-Means	84(s)	170(s)	251(s)
经典 Birch	121(s)	230(s)	334(s)

中所用时间没有随着数据点个数增加而膨胀,变化趋势不是剧烈,这是因为离线聚类的时间复杂度是和网格个数呈线性关系。K-Means、Birch 这两算法所用时间基本随着数据点个数线性增长。随着数据点个数的增大、数据的逐渐密集,GDDS 会表现出更好的时间特性。

GDDS 作为一种可继承、支持增量的聚类方法对处理数据流问题有很强的针对性。同时,GDDS 的聚类实质是一种基于网格和密度的聚类。这种巧妙的设计使得该挖掘模型对数据流的分析十分适用。

结束语 针对于数据流的挖掘,本文提出一种基于网格和密度的算法。该算法打破了传统的挖掘模式,以中间知识库为核心,分两步来挖掘,既能高效地处理动态数据,还能很好地满足用户的需求。

通过分析可以看到,GDDS 保证了挖掘效率,提高了系统的处理能力。通过周期性地把新数据结合到原有的知识库中,随时可以提供最新数据的知识展示,从而更有利于决策。

参 考 文 献

1 Barbara D. Requirements for Clustering Data Streams. SIGKDD Explorations, 2002, 3(2):23~27

2 Aggarwal C C, Han Jiawei, Wang Jiangyong, et al. A Framework for Clustering Evolving Data Streams. In: Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

3 Zhang T, Ramakrishnan R, Livny M. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: ACM SIGMOD Conference, 1996

4 Agrawal R, Gehrke J, Gunopulos D, et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: ACM SIGMOD Conference, 1998

5 Guha S, Mishra N, Motwani R. Clustering data streams. In: Proc. 41st IEEE Symp. on Foundations of Computer Science, November 2000. 359~366

(上接第 128 页)

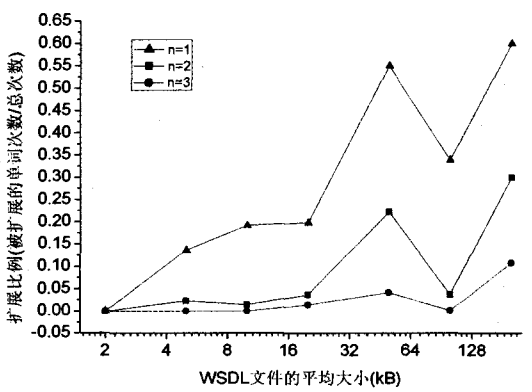

图 4 不同门限值和文件大小对扩展效果的影响

后两种情况表明:1) WordNet 作为通用的词典,所提供的同义词中有些在程序开发的命名中很少用到,因此其对于服务查找的用处不如想象的那么大;2)以 WSDL 文件中同一 operation 的相关元素和类型中出现的名称作为上下文来确定一个单词的义项时,所能提供的上下文信息还是不够丰富和精确。好在这两种情况只是降低了查准率而无损于查全率,在有人工检查查找结果的情况下,不至过于影响 SSBS 方法的应用价值。

结论和展望 针对如何在 Internet 范围内快速查找所需功能的 Web Service 问题,本文借鉴通用的信息检索方法,提出一种使用虚拟文档、分设多个字段、借助 WordNet 识别同义词的方法 SSBS。比起通用的 Web 信息检索工具,本方法提高了查准率和查全率,可以加快将来基于 Web Service 的系统的开发过程。比起 UDDI 这样的集中式注册方式,本方法的部署则更为灵活;比起 Semantic Web Service 技术,本方法虽然不能保证 100%的准确、不能用于完全自动化的开发过程,但执行时效率更高,与现有技术结合更容易。

下一步的工作有:改进确定最佳义项的匹配算法,引入常用缩略词表、N-gram 等基于统计的方法、识别词形变换,并计算在将来加入爬虫功能后将系统发布到网上接受公开测试。

参 考 文 献

1 Zhao Wen-feng, Chen Jun-liang. Toward Automatic Discovery and Invocation of Information-Providing Web Services [C]. In: 1st Asian Semantic Web Conference (ASWC 2006), Beijing, China, 2006. 474~480

2 Eric N, Greg L. Understanding SOA with Web Services [M]. Pearson Education, 2004

3 Massimo P, Takahiro K, Terry P R, et al. Semantic matching of Web Service capabilities [C]. First International Semantic Web Conference(ISWC 2002), Sardinia, Italy, 2002. 333~347

4 刘升平, 林作铨, 梅 婧, 等. 一种 XML 的模型论语义[J]. 软件学报, 2006, 17(5): 1089~1097

5 David G, Ophir F. Information Retrieval - Algorithms and Heuristics(2nd Edition)[M], Springer, 2004. 11~18

6 Miller G A. WordNet: A Lexical Database for English [J]. Communications of the ACM, 1995, 38(11):39~41