# 流聚类问题六月份总结

**李省委**

数据流的聚类问题是数据挖掘领域聚类问题的一个特例，它具有一般聚类问题的特征，也有自身的特殊要求。我们经过实验和研读论文，认为数据流的聚类算法应该满足以下要求，并且提出我们的改进算法。

## 一、数据流聚类算法的四个特征

### （一）数据流的分布特征随着时间不断变化。

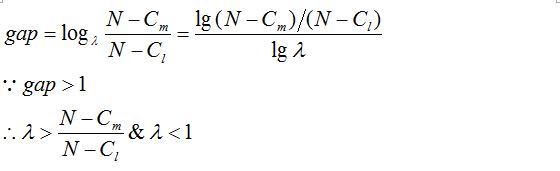
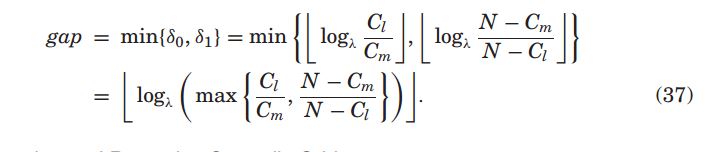
数据流应被视为随着时间不断变化的数据组成的无限过程。聚类结果，是它的分布特征之一。人们往往关注分布特征的变化及最近一段时间的数据分布状况。这种特点的表现有如下几个方面：

1、新簇不断出现，旧簇不断退化消亡。算法必须表现这种特征。

2、簇和离群点往往会相互转换角色。算法必须实现离散点可以发展为簇，簇会退化为离散点而被删除。

3、程序应该提供交互的方式，满足用户查询过去不同时间段聚类结果的要求。

衰减因子的方法，可以表现出数据的变化，原论文中，衰减因子和gap的计算公式如下：如下：



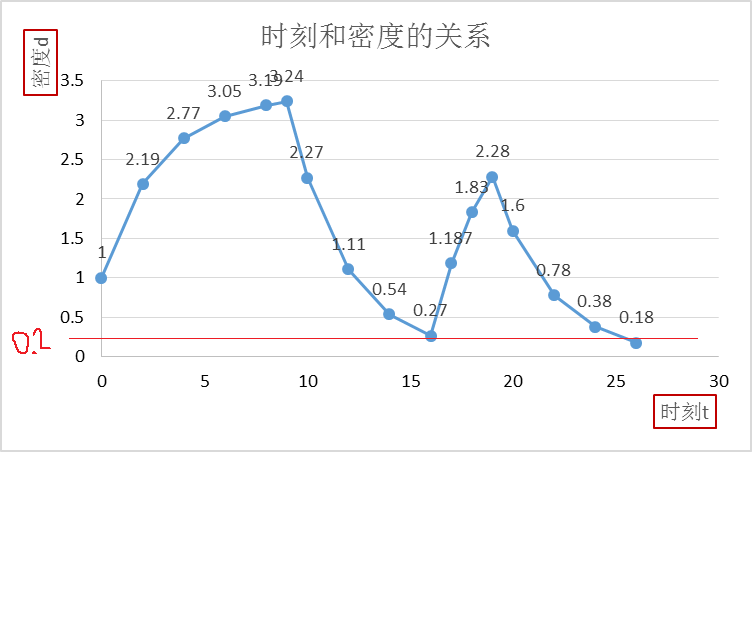
当网格数目N很大时，分式趋近于1，所以衰减因子也趋近于1。此时，数据基本上没有衰减，所以我们不能按照此公式确定衰减因子和时间间隔。

### （二）过期的历史数据会对聚类产生影响。

基于数据流的海量和高速特点，数据不能被重复访问，而为了利用足够的信息形成聚类，必须保留大量信息。这就会产生历史数据，过期的历史数据会影响现在的聚类结果，必须给予削弱。一般算法以数据的时标为依据，微簇的时标低于某一阈值时，该簇被删除。但是这种策略删除整个微簇而非微簇中旧的记录。这种策略的不足，如下所示：

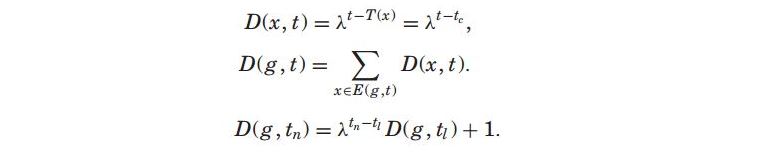
在基于网格的聚簇方法中，存在格子g，在t为0-30之间，各时刻格子中数据的个数和格子的密度，如下图所示，其中格子的密度公式为（与论文相同）：

**图1**



**图2**

其中，密度阈值是dl=0.2



在t为0-9时，每次格子都接受到数据，所以密度一直上升，直到t=9时，达到最大密度D=3.24。

在t为10-16时，格子没有接受数据，密度一直下降，直到t=16时达到最小密度D=0.27。

在t为17-19时，格子接受三个数据，密度上升，达到2.28。

在t为20-26时，格子没有接受数据，密度一直下降，直到小于临界值dl=0.2，此时格子被删除。

从中可以看出，使得格子密度下降的因素是因为中间有段时间t为10-16时没有接受数据，此时格子中的数据被分为新旧两种数据。合理的方案应该是，删除旧数据，保留新数据，而不是全部删除。

### （三）各个簇具有独立的进化行为。

因为数据流在随着时间变化，所以它的分布特征也在变化，即各个簇保存的数据个数，簇的中心与半径等信息都在随着时间不断变化，而且各个簇的变化还不一致。有的变化快，有的变化慢。对于变化快的簇，我们应该保留较多的信息，变化慢的少一些。

### （四）噪音点降低聚类精度。

在聚类过程中，绝大多数的数据都能被现存的簇或者潜在的微簇所吸收。因为噪音点无法被现有簇“吸收”，算法会为噪音点创建许多新的微簇。同时现有微簇必须相应合并减少或删除，所以降低了精度。

## 二、改进的流聚类算法概述

基于以上四点观察，我们提出一种改进的数据流聚类算法，并对上述四个问题提出解决办法。

数据

网格

微簇

类

匹配

密集格子

删除的格子

新增数据

**图3**

我们的思路如图3所示，其中网格指网格列表，负责格子的建立、更新和删除。微簇是格子的集合，并且微簇内的格子相互连通。对网格列表的一次扫描，使得相互连通的密格形成微簇，最终形成微簇列表。微簇列表是聚类结果的阶段性表示，也是一种粗糙的表示。类表负责形成簇。它从微簇列表中读取新数据，从网格列表中读取应该删除的格子，通过这两种方式更新自己，能够及时表现现阶段的数据分布。类表不仅可以表现最近一段时间的聚类结果，还可以体现出数据流分布特征的变化情况。类表是全局的结果，也是精确的结果。

下面介绍流程的各个过程。

### （一）数据形成网格列表。

每个数据都属于空间中的一个格子。格子在不同时间段接收不同数量数据后，就会形成密格、过渡格、稀疏格和离散格。随着算法的进行，密度很低的格子中的旧数据会被删除，某些长时间不接受数据的格子也会被从网格列表中删除。在网格列表中中，密格和过渡格的密度较大，稀疏格和离散格密度虽小但是保留了较新的数据，旧的数据已被删除。密格，接收数据的个数较多，而且数据之间的时间间隔较小，足以表现此时间段的空间中数据聚集的区域。所以我们利用密格形成初步的聚类。

### （二）网格列表形成微簇。

数据流中的簇被划分成网格后，一般情况下保持着网格连通、数据密集的特征。我们利用CF树的思想，将相互连通的网格快速连接在一起，形成微簇。最终，当网格列表被扫描结束后，现阶段所有的密格被分配到微簇中。这个行为，每经过gap时间就会进行一次。每个微簇都代表着空间中一处数据密集的区域。这是初步的聚类。但是随着时间的流逝、数据的分布模式会发生改变，出现数据特征的“漂移”现象。微簇列表是现阶段的聚类结果，无法表现出数据特征的变化情况。所以我们必须利用类表对它进行优化。

### （三）类表的形成和更新。

类表中每个类都是一个簇，为了便于区分，称之为类。变化，即是两者的对比。类表必须记录以前的聚类结果和现阶段的聚类结果。类表中的每个类也有自己产生，增长，停滞，消失的过程。

当类表为空时，扫描微簇列表时，它会把每个微簇当作类，并把相互连通的微簇合并在一起。这是类的产生。

当经过gap时间，再次扫描微簇列表时，每个微簇会寻找和自己相互连通或者中心距离小于某个阈值的类，并将类中没有的数据和网格添加到这个类中，实现类中数据的更新。这可以使得类能够实时表现数据流的分布特征。这是类的增长。

当类所表现的空间区域很少有数据到达时，随着密度的下降，它所包含的网格就会逐渐被从网格列表中删除。但是类中的对应网格和匹配的数据并没有被删除，只是被标记为无效格子。这个时候，类中的数据并没有减少，也没有增加，这是类的停滞。

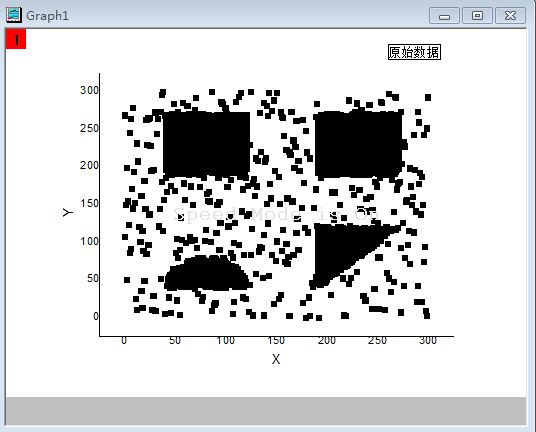
每隔gap时间，网格列表都会向类表提供这次被删除的网格名单，这样类中无效网格的数目不断增加。当无效网格在该类中所有网格中的比例到达某个阈值时，这个类就会被删除。类比对离散网格的处理，我们不会完全删除该类中所有网格，仍然保留有效网格，并使他们形成新的类，保留在类表中。这是类的消失。

## 三、模拟实验的结果

数据流的聚类要关注效率和聚类策略两个方面，以上的论述侧重于策略方面。上述方法将网格的方法，基于距离的方法和分层的方法结合起来，采用每种方法的优势，用一定的方法避免其缺陷。网格在高维时，格子个数太多，致使搜素效率降低，我们采用“中心格-周围格”的数据结构降低格子列表中的结点个数。基于距离的方法，不能识别非球形簇，我们将数据为处理单元改变为以网格为处理单元，并且利用网格的相邻、连通等特点识别非球形簇。分层的方法，一旦微簇合并则不能分离。这种方法容易形成局部的聚簇，但是局部的结果并不能代表全局结果。我们利用类表，采用无效格子这个指标，可以将合并的簇分开。

实验结果如下：

二维数据的横纵坐标范围是0-300，每个格子的长和宽是5，每个数据的坐标都是整数，共20,000个数据，其中衰减因子为0.995，密格和过渡格阈值时4.0，过渡格和稀疏格阈值是0.63。图1是数据原始图，程序读取时，先读取左边的列，每列先读取最下面的数据，即按照坐标大小依次读取。按照这种思路读取，那么左边的数据应该先形成簇，当右边的数据形成簇时，左边的数据逐渐消失，由此可以看出数据流的分布特征随着时间而变化。



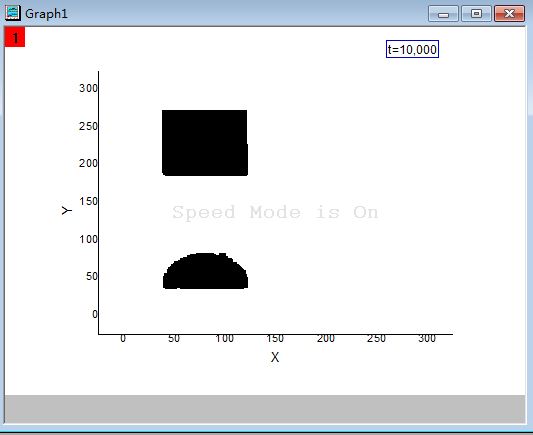
**图4**

图4中，这是原始数据，整体可分为四个簇。



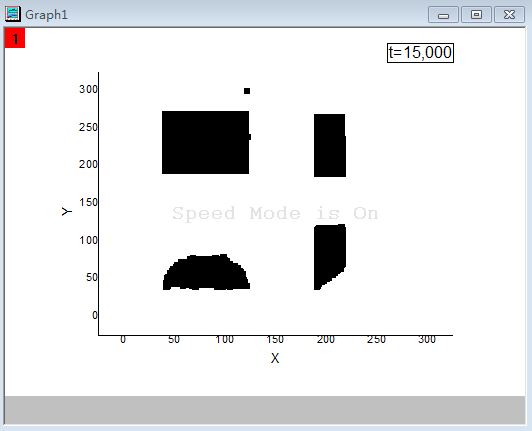
**图5**

图5中，t=5000时，程序读取5000个点，类表中有两个结点，形成两个簇。这两个簇处在生长期，类中的数据会不断增加。



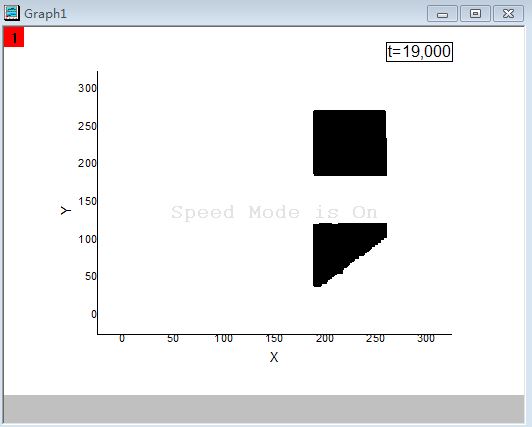
**图6**

图6，t=10,000时，程序接受10,000个数据，类表中有两个结点，形成两个簇。



**图7**

图7中，t=15,000时，程序接受15,000个数据，类表中增加两个结点，形成四个个簇。左边的两个类处于停滞期，无效格子的比例逐渐增大。右边的两个类，处于增长期，类中数据个数不断增加。



**图8**

图8中，t=19,000时，程序接受19,000个数据，类表中最先形成的两个类消失，新数据形成两个新簇。