情报学报 ISSN 1000-0135
第29卷第4期579-585，2010年8月

JOURNAL OF THE CHINA SOCIETY FOR SCIENTIFIC AND TECHNICAL INFORMATION ISSN 1000-0135
Vol.29 No.4，579-585 August 2010

doi:10.3772/j.issn.1000-0135.2010.04.001

一种基于主成分和密度的改进型动态数据流聚类算法[1)]

琚春华 梅 铮 许翀寰
（浙江工商大学计算机与信息工程学院，杭州 310018）

摘要 本文主要研究了在有限资源约束下的数据流聚类方法。针对海量，高速的数据流，现有聚类方法在有界内存和有界时间的限制下，难以快速有效地进行聚类，设计了一种基于主成分和密度的动态数据流聚类算法，PDStream算法.它采用滑动窗口管理数据流；首先使用主成分模型作为前置系统，它负责对基本窗口内的源数据进行属性转换，起到了降维的作用；然后使用密度聚类模型作为后置系统进行聚类操作；最后对系统中生成的概要数据进行简化的二次聚类并更新聚类簇。通过实验表明，PDStream算法有效克服了STREAM算法使得聚类受控于历史数据的缺点，显现出处理海量数据的优越性以及聚类质量高的特点。

关键词 数据流聚类 主成分分析 密度 滑动窗口

An Improved Clustering Algorithm for Dynamic Data Streams Based on Principal Component Analysis and Density

Ju Chunhua, Mei Zheng and Xu Chonghuan
(*Information College of Zhejiang Gongshang University, Hangzhou* 310018)

Abstract The data stream clustering method in the constraints of limited resources is investigated in this paper. In view of massive, high-speed data streams, the existing clustering methods are difficult to carry out rapid and effective clustering with bounded memory and time, an improved clustering PDStream algorithm for dynamic data streams based on principal component analysis and density is designed. It adopts sliding window to manage data streams. First, the pre-system makes use of principal component model to convert properties of the source data in the basic window, which plays a role of dimensionality reduction; Second, the post-system chooses the density model to execute clustering operation; Finally, the summary date generated in the aforementioned steps is required to execute simply second clustering and update the clustering result. Experiments show that PDStream algorithm effectively overcomes the shortcomings of the STREAM algorithm controlled by historical data and has the superiority of handling mass data and the characteristics of high-quality clustering.

Keywords data stream, principal component analysis, density, sliding window

收稿日期：2009年10月10日

作者简介：**琚春华**，男，1962年生，博士，教授，博士生导师，研究领域：人工智能，智能信息处理，电子商务，物流技术。**梅铮**，男，1986年生，硕士，研究领域：商务智能，数据挖掘。E-mail:meizheng3994239@163.com。**许翀寰**，男，1983年生，助教，研究领域：人工智能，数据挖掘，电子商务。

1) 资助项目：国家自然科学基金（编号：70671094）；浙江省自然科学基金重点项目（编号：Z1091224）；浙江省自然科学基金（编号：Y1090617）；浙江省科技计划项目（编号：2009C13G2050020）。

引 言

近年来，随着计算机软、硬件和网络的飞速发展，通信业、金融业、电子商务等领域正产生着源源不断的连续数据，人们正面临着如何处理大量流数据的问题，数据流挖掘正逐渐成为相关人员的研究热点。在国外，形成了很多数据流研究小组，在开发多个研究项目和原型系统，涉及了数据流的管理、查询、挖掘等各方面的内容。而国内，目前还没有大规模研究数据流问题的研究小组。对于数据流聚类，目前基于传统静态数据库或数据仓库的聚类研究已经比较成熟，出现了一些经典的聚类算法，比如基于划分的 k 平均和 k 中心点聚类，基于密度的 DBSCAN 聚类等。但将现有聚类算法用于具有海量、高速、动态特征的数据流聚类显然是不合适的，因为对数据流的扫描只有一次或有限的几次，它不可能像处理静态数据源那样可以进行重复扫描操作。因此面对有限内存和有限处理时间，如何设计高效、准确的数据流聚类算法是当前迫切需要解决的问题。

1 问题描述及研究现状

1.1 数据流管理模型

动态数据流由一系列按序到达的数据组成，也可看作是信息传输过程经编码处理的数字信号串。若令 t 表示任一时间戳，a_t 表示在 t 时刻到达的数据元素，则数据流可以表示为无限集合 $\{\cdots, a_{t-1}, a_t, a_{t+1}, \cdots\}$。在描述数据流集合中的元素 a_t 时，一般采用 Turnstile 模型[1]。

目前，数据流处理技术包括抽样、直方图、小波、滑动窗口及一些统计技术。本文采用滑动窗口对数据流进行管理，其基本思想：它就像用一个不变的窗口，数据随时间的推移经过窗口，出现在窗口内的数据就是被计算的数据集合，滑动窗口模型所处理的数据是数据流中最新到达的滑动窗口内的数据，而不是处理目前滑动窗口中所有数据。再结合主成分分析中的降维技术，可以有效减少内存的需求。

设 n 表示当前时间戳，s, e 分别是两个已知的时间戳。描述的是数据流中最新的 W 个数据，其查询范围是 $\{a_{n-W+1}, \cdots, a_n\}$，随着数据的不断到达，窗口中旧数据从窗口一端移出，新数据从窗口另一端移入。目前应用比较广泛的一种计算框架是将滑动窗口划分成若干基本窗口，每次更新一个基本窗口的数据序列，图 1 给出了滑动窗口与基本窗口的原理图。通过把将大小为 w 的滑动窗口按照时间次序划分成 k 个等宽的子窗口，称为基本窗口。每个基本窗口包含 $b = w/k$ 个元组。

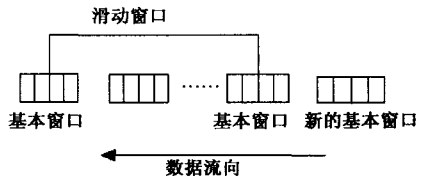

图1 滑动窗口与基本窗口

1.2 研究现状

目前数据流聚类[1~5]主要由 Guha、Aggarwal 等提出并被不断改进的。如数据流聚类的 LOCALSEARCH 算法，它是基于分治的思想使用一个不断的迭代过程实现有限空间对数据流进行 K-means 聚类。在 LOCALSEARCH 算法基础上改进得到的基于单遍扫描的 STREAM 算法，并证明了 STREAM 算法比层次聚类的 BIRCH 算法更优。后来又提出了一个有效解决数据流聚类问题的框架，即基于演化分析的 CluStream 算法。国内也出现了一些优秀的算法：如常建龙等[6]提出的一种基于滑动窗口的进化数据流聚类，朱蔚恒等[7]提出的基于数据流的任意形状聚类算法等。

STREAM 算法[8]是一个单遍扫描的基于 K-means 的流聚类算法。它使用有限的空间和时间，然而，它既不考虑数据的演化也不考虑时间粒度的变化，聚类可能受控于历史数据。在实际应用中，虽然它能够有效地克服噪声数据带来的影响，但它只能提供对当前数据流的一种描述，而不能反映流数据的变化情况。CluStream 算法[9]是一种基于用户指定的、联机聚类查询的演化数据流聚类算法。它把聚类过程分为联机和脱机两部分。首先：使用一个在线的微聚类 *micro-cluster* 过程对数据流进行初级聚类，并按一定的时间跨度将 *micro-cluster* 的结果按一种称为金字塔型时间窗口 pyramid time frame 的结构储存下来。同时，使用另一个离线的宏聚类 *macro-cluster* 过程，根据用户的具体要求对 *micro-cluster* 聚类结果进行再分析。但对于非球形的聚类，*micro-clusterr* 过程不能给出一个很好的描述。由于 *micro-cluster* 过程得到的结果是对数据流进行一个初步的描述，所以，如果这个描述本身不精确，那么后续的 *macro-cluster* 过程将不能得到一个

较好的结果。另外 CluStream 最大不足之处在于，聚类的半径随着数据的流入而不断增大，由于没有在线淘汰“老数据”，最后导致数据量越来越大，增加了处理代价。此外，由于数据流算法在时间和空间上的限制，设计一个高效的数据流聚类是一个艰巨的任务。

基于以上不足，本文提出一种基于主成分和密度的数据流聚类模型。对于 PDStream 模型，它首先采用处理数据流常用的滑动窗口模型管理数据。然后使用主成分分析模型作为前置系统，密度聚类模型作为后置系统对基本窗口内的数据流进行处理，最后在概要数据的基础上进行简化的二次聚类并更新聚类簇，具体过程：在特定时间段内，将流入基本窗口 k 中的数据作为静态数据进行主成分分析，即对源数据进行了降维；再对转化后的数据进行基于密度的聚类，即得到了关于窗口 k 的概要数据；最后再结合数据流的特征通过概要数据不断更新聚类簇。通过理论与实验分析，PDStream 模型在执行时间、存储空间和聚类质量等性能上有所改进，它能够符合数据流聚类的要求。

2 主成分和密度融合的数据流聚类模型

2.1 主成分分析模型（PCA principal component analysis）

维数约简一直是模式识别、机器学习、多元数据分析等领域的重要研究课题之一。特别是随着信息时代的到来，人们获取大数据量、高维数、非结构化的数据变得越来越容易，这就使数据约简变得更为迫切。

主成分分析[10]的主要目的就是对原变量加以“改造”，在不致损失原变量太多信息的条件下尽可能地降低原变量的维数，有效达到数据约简的目的。以商业数据流为例，数据通常少至几十维多至成千上万维，面对这样的高维数据，只有通过有效的降维，才能达到比较好的分析效果。降维就是通过对特征属性的提取和选择，在所有特征中求出最重要的特征，放弃一些次要特征，达到降维目的，高维数据流通过降维后，才能更好的进行数据流聚类分析，

主要原理：设 $X = (X_1, X_2, \cdots, X_p)^{\mathrm{T}}$ 为 p 维随机向量，设线性组合：

$$Y_1 = a_1^{\mathrm{T}} X = a_{11} x_1 + a_{12} x_2 + \cdots + a_{1p} x_p ,$$

如果要用 Y_1 尽可能多地保留原始的 X 的信息，经典的办法是使 Y_1 的方差尽可能大，这需要对线性变换的系数 a_1 加限制，一般要求它是单位向量，即 $a_1^{\mathrm{T}} a_1 = 1$。其他的各 Y_i 也希望尽可能多地保留 X 的信息，但前面的 $Y_1, Y_2, \cdots, Y_{i-1}$ 已保留的信息就不再保留，即要求 $\mathrm{Cov}(Y_i, Y_i) = 0, (j = 1, \cdots, i-1)$，同时对 a_i 也有 $a_i^{\mathrm{T}} a_i = 1$ 的要求，在这样的条件下使 $\mathrm{Var}(Y_i)$ 最大。

注意到加大 a 的长度必定使 Y_1 的方差变大。事实上，对于任意常数 $c > 0$，有 $D(cy_1) = c^2 a^{\mathrm{T}} Va$，因此，如果对 a 不加任何限制，问题就变得毫无意义了，一个自然的限制是令 a 的长度为 1，即 $a_1^{\mathrm{T}} a_1 = 1$，于是问题就归结为求 a，使 $\begin{cases} \max: & a^{\mathrm{T}} Va \\ \text{s.t.} & a^{\mathrm{T}} a = 1 \end{cases}$，问题转化为求一个条件极值问题，利用拉格朗日(Lagrange)乘数法，即可求得 a 值，以及相应特征根对应的单位化特征向量。在用拉格朗日数乘法在 $a^{\mathrm{T}} a = 1$ 条件下求 a 值的过程中，我们发现对于动态数据流而言，常常出现大量零数据及空值，导致构造协方差矩阵时出现大型稀疏矩阵，由于稀疏矩阵的数据关联性较小，弱化了主成分分析的前提，为此我们需要对大型稀疏矩阵进行压缩，以便于后面产生更加合理的聚类簇。稀疏矩阵的压缩存储方式有很多，对特殊的稀疏矩阵，如三角矩阵、对称矩阵等，它们的压缩存储处理相对地比较简单；而对于随机稀疏矩阵，由于其非零元分布不规则，因此，除了存储非零元的值之外，还必须同时记下它所在行或列的位置，以此实现稀疏矩阵的压缩。接着我们使用 Davidson 算法求解稀疏矩阵的最大(或最小)特征值及相应的特征向量[11]。

第 1 主成分：在约束条件 $a_1^{\mathrm{T}} a_1 = 1$ 下所确定的向量 a_1，使得 $\mathrm{Var}(Y_1) = a_1^{\mathrm{T}} \sum a_1$ 达到最大，即 $Y_1 = a_1 X$，其中 $\sum$ 为协方差矩阵。

第 k 主成分，若 $Y_1, Y_2, \cdots, Y_{k-1}$ 还不足以反映原变量的信息，则进一步构造 $Y_k = a_k^{\mathrm{T}} X = a_{k1} x_1 + a_{k2} x_2 + \cdots + a_{kp} x_p$。

在约束 $a_k^{\mathrm{T}} a_k = 1$ 及 $\mathrm{Cov}(Y_k, Y_i) = a_k^{\mathrm{T}} \sum a_i = 0 (i = 1, 2, \cdots, k-1)$ 下所确定的 a_k，使得 $\mathrm{Var}(Y_k) = a_k^{\mathrm{T}} \sum a_k$ 达到最大，即 $Y_k = a_k X$。

2.2 基于密度的聚类算法

经典的基于密度的聚类算法有 DBSCAN 算法[12]，它可用来过滤“噪声”孤立点数据，以发现任

意形状的簇，其主要思想是只要临近区域的密度(对象的个数)超过某个阈值则继续进行聚类，即对于给定簇中的每个对象，在一个特定范围的区域中必须至少包含某个数目的对象。

(1)给定对象半径 ε 内的区域称为该对象的 ε -领域。

(2)如果一个对象的 ε -领域至少包含最小数目 *MinPts* 个对象，则称该对象为核心对象。

(3)给定一个对象集合 D，如果 p 是在 q 的 ε- 领域内，而 q 是一个核心对象，则称对象 p 从对象 q 出发是直接密度可达的。

(4)如果存在一个对象链 $p_1,\cdots,p_n,p_1 = q,p_n = p$，对 $p_i \in D(1 \leqslant i \leqslant n)$，$p_{i+1}$ 是从 p_i 关于 ε 和 *MinPts* 直接密度可达的，则对象 p 是从对象 q 关于 ε 和 *MinPts* 密度可达的。

(5)如果对象集合 D 中存在一个对象 o，使得对象 p 和 q 是从对象 o 关于 ε 和 *MinPts* 密度可达的，那么对象 p 和 q 是关于 ε 和 *MinPts* 密度相连的。

2.3 PDStream 算法设计

2.3.1 基本思想

主成分分析是以数据流的数据量巨大、属性间存在一定的关联性为前提的，如果数据过少，导致数据离散化程度过高，则主成分分析将达不到预期效果。

定理 1[10] 构造出的方差大于零的主成分最多只有 p 个。

事实上，若有 $p+1$ 个主成分 $Y_1,\cdots,Y_{p+1}$，其方差均大于零，则由于 $a_1,a_2,\cdots,a_{p+1}$ 这 $p+1$ 个 p 维单位向量必线性相关，从而不妨设 $a_{p+1} = l_1a_1 + l_2a_2 + \cdots + l_pa_p$ 且 $l_1 \neq 0$，有 $\mathrm{Cov}(Y_{p+1},Y_1) = (l_1a_1 + l_2a_2 + \cdots + l_pa_p)^{\mathrm{T}}\sum a_1 = l_1a_1^{\mathrm{T}}\sum a_1 = l_1\mathrm{Var}(Y_1) \neq 0$，这与主成分的要求(要使 Y_{p+1} 和 Y_1 所反映的原变量的信息不相重叠，要求 Y_{p+1} 和 Y_1 不相关，即 $\mathrm{Cov}(Y_{p+1},Y_1) = \mathrm{Cov}(a_{p+1}^{\mathrm{T}}X,a_1^{\mathrm{T}}X) = a_{p+1}^{\mathrm{T}}\sum a_1 = 0$)相矛盾。所以构造出的方差大于零的主成分最多只有 p 个。

由定理 1 可以知道，我们所构造出的主成分个数将一定比原数据流的属性变量少，这也是我们采用 PCA 模型作为前置系统的主要依据。

PDStream 算法对数据流实行增量式处理，随着数据流增加或删除，加入新生成的概要数据，不断更新聚类簇。概要数据的建立是通过对各个基本窗口分别进行聚类得到的，它是一种能够在一个远小于数据规模的内存空间里不断更新一个代表数据集特征的结构，挖掘算法即是对这种概要数据进行近似处理并实现在线分析的。

基于 PCA 和密度的 PDStream 算法采用滑动窗口对数据流进行管理，在特定时间段 T 内，首先将流入基本窗口 k 中的数据作为静态数据进行主成分分析，将高维数据结构转化为用另一组不同性质的低维数据结构表示，此步骤起到了降维作用，再对转化后的数据进行基于密度的聚类，此时即得到了关于窗口 k 的概要数据(窗口 k 的聚类簇)。在时间截止前，即 $T = |t_2 - t_1|$（t_1 为数据开始流入基本窗口 k 的时刻，t_2 为终止聚类操作的时刻)。采用相同的处理方法可得到各个基本窗口的概要数据，最后按预先设置的时间，终止概要数据的生成，并把前面生成的概要数据作为静态聚类源进简化二次密度聚类并更新簇，即得到按要求生成的 k 个聚类簇。算法流程如图 2 所示。

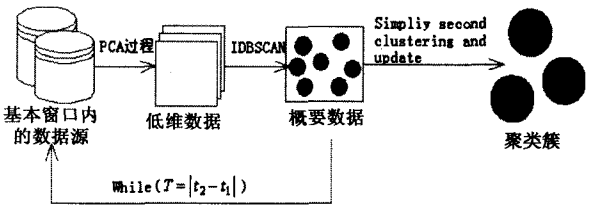

图 2 算法流程简图

2.3.2 算法设计

PDStream 算法的设计内嵌了 PCA 算法和 IDBSCAN 算法，下面分别予以描述。

①PCA 算法：

Step1：求出 $X = (X_1,X_2,\cdots,X_p)^{\mathrm{T}}$ 的协方差矩阵 $\sum$ 的 p 个特征值，并按大小顺序排列：$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_p \geqslant 0$，以及对应的正交单位化特征向量为 $e_1,e_2,\cdots,e_p$

Step2：求出 k 个主成分 $Y_k = e_k^{\mathrm{T}}X = e_{k1}X_1 + e_{k2}X_2 + \cdots + e_{kp}X_p, k = 1,2,\cdots,p$

Step3：第 k 个主成分 Y_k 对总信息的贡献率 $\lambda_k/\sum_{i=1}^{p}\lambda_i$，前 m 个主成分的累计贡献率 $\sum_{i=1}^{m}\lambda_i/\sum_{i=1}^{p}\lambda_i$

Step4：依据累计贡献率 $\sum_{i=1}^{m}\lambda_i/\sum_{i=1}^{p}\lambda_i \geqslant 85\%$ 确定 m 值，即主成分的个数。

②IDBSCAN 算法：

Step1：对最新流入基本窗口 k 的数据建立 R*-树，并找出 k-dist 曲线由陡峭变得平滑的点对应的距离，确定 *Eps* 合理取值；

Step2：从数据集中任意选取一点 p，并对其在基本窗口 k 内进行查询；

Step3：如果 p 是核心点，则寻找所有从 p 密度可达的点，最终形成一个包含 p 的簇；

Step4：否则，p 被暂时标注为噪声点；

Step5：如果基本窗口 k 所包含的对象集中存在无任何标记的点，则选取一点重复以上操作；

Step6：否则，得到初始聚类簇集。

③PDStream 算法描述：

输入：从任意时刻 t 开始流入到滑动窗口内的数据。

输出：k 个聚类簇 $\{A_1, A_2, \cdots, A_k\}$。

从 t 时刻开始，在特定时间内，随着数据不断流入滑动窗口，数据（向量表示为 $X = (x_1, x_2, \cdots, x_n)$）充满第 k 个基本窗口时，即对这数据量为 w、宽度为 k 的基本窗口进行等距抽样，并对得到的数据实施 PCA 过程，将数据 $X = (x_1, x_2, \cdots, x_n)$ 转化为 $Y = (y_1, y_2, \cdots, y_m)(m \ll n)$，然后在此第 k 个基本窗口内对新转化的数据实施 IDBSCAN 过程，得到聚类结果 $A^{(k)} = \{A_1^{(k)}, A_2^{(k)}, \cdots, A_i^{(k)}\}$。

当基本窗口 k 流出滑动窗口，要进行聚类操作的基本窗口是 l 时，同理，进行类似处理并得聚类结果 $A^{(l)} = \{A_1^{(l)}, A_2^{(l)}, \cdots, A_j^{(l)}\}$

……

时间结束后，把从各个基本窗口所得到的聚类结果作为第一次聚类后的概要数据，再在此基础上进行二次聚类，即按照簇边缘—簇核心—簇边缘的顺序将所有基本窗口的聚类合并、删除或者产生新的簇，最后即生成特定时间内整条数据流的聚类簇 $\{A_1, A_2, \cdots, A_k\}$。

2.4 PDStream 算法时间复杂度

下面分析每个时刻算法的时间复杂度，它取决于算法的三个阶段：主成分的确定、基本窗口内数据流的聚类、概要数据的二次聚类。算法第一阶段都是线性操作，时间复杂度为 $O(n)$；第二阶段由于采用 kd 树数据结构，使得可以有效地检索特定点给定距离的所有点，时间复杂度为 $O(n\log n)$；第三阶段实质是重复了第二阶段的操作，仅仅是数据量的改变，数据量是线性递增，故时间复杂度为 $O(n^2\log n)$。

2.5 算法分析

建立 R*-树和绘制 k-dist 曲线图都是非常耗费时间的工作，大规模数据库尤其如此。此外，用户须反复试验，选择合适的 k-dist 值以达到较好的聚类效果。所以在进行聚类之前应该对数据流进行预处理操作。

DBSCAN 算法虽然能够发现任意形状的聚类，但它致命的缺点是需要根据分析人员的主观判断给定参数 ε 和 MinPts 进行聚类，并且聚类结果对参数是非常敏感的。真实的高维数据经常分布不均，在聚类前就设定以上两个参数明显是不附实际的，不均匀的数据需要不断变化的参数，此时我们可以把参数 ε 和 MinPts 动态地设计成关于数据量和数据分布面积之比的函数，如：$\varepsilon = f_1(C/S)$，$\text{MinPts} = f_2(C/S)$，其中 C 为数据量，S 为围绕核心对象 o 的面积。

下面将 PDStream 算法与 STREAM 和 CluStream 算法进行对比分析。前面已指出 STREAM 所得的聚类可能受控于历史数据，屏蔽了新流入基本窗口数据的作用，而 PDStream 算法在设计过程中，它采用了各基本窗口的初始聚类，而后进行概要数据簇的二次聚类，随着数据的流入起到了实时更新的效果，有效解决了 STREAM 算法的此项不足。

对于 CluStream 算法，前面指出其对于非球形的聚类，*micro-cluster* 过程不能给出一个很好的描述，而 PDStream 算法后置系统是基于密度的聚类，其最大的优势是能够发现并描述任意开关的聚类，另外 CluStream 算法会由于没有在线淘汰“老数据”导致处理代价大大增加，而 PDStream 算法在进行二次聚类时，是以前面得到的概要数据为基础，更新后的簇与原簇相比，不会有数据量的增加，反而可能因为密度阈值偏小，将距离较远的几个聚类进行了合并，这样则会减少数据量，起到更佳的效果。

3 性能分析与测试

由于 PDStream 算法在前阶段使用 PCA 对数据流进行降维，并且其最后一步聚类过程是基于概要数据的，而传统数据流聚类算法是基于不断更新的原始数据。这样，PDStream 在扫描时间，聚类效率上较之 CluStream 算法都有所改进。下面结合具体数据流进行实证分析，比较对象是当前最好的数据

流聚类算法之一 STREAM。

3.1 执行时间比较

实验环境为 Windows xp 个人版、2G 内存、Pentiun IV 平台，采用 matlab7.0 实现。算法采用 KDD-CUP'99 网络入侵数据流进行仿真，该数据集曾被多篇数据流聚类文章引用。如 Aggarwal 等[13]利用该数据集比较 CluStream 和 STREAM 算法；Chalaghan 等[14]用它来比较 STREAM 和 Birch 算法。该数据集由一个局域网中 TCP（transfer control protocol）连接的原始记录所构成.实验时从数据集中读取 20000 个数据，选取了其中 34 个数据值属性，并设定簇数为 8 进行对比试验。以间距为 10% 进行等比例随机抽样，即分别随机抽取原数据量的 10%、20%，并分别记录所耗费的时间，所得实验结果如图 3 所示。

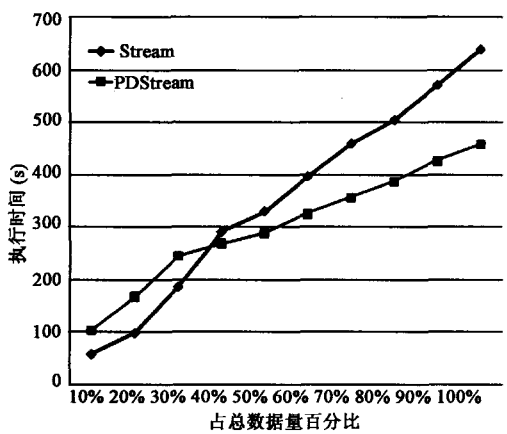

图 3 执行时间对比

从图 3 中可以看出，当数据量较小时，STREAM 算法运行时间较少，处理速度较快，但随着数据量的逐渐增大，PDStream 算法显现出自己的优势，时间并没有像前者那样呈直线上升而是很平缓地上升，处理速度快于前者。这说明 PDStream 算法对于处理数据流这种大数据理更加合适，有效克服了时间和内在的不足，得到了预期效果。

3.2 聚类质量比较

下面比较 PDStream 和 STREAM 在真实数据集上的聚类质量，采用的度量是 SSQ（sum of square distance）[9,15]。对于 PDStream，我们定义 SSQ：对于在核心对象所形成的簇中的数据点，该点属于该核心对象所在的簇；否则，该点属于离它最近的簇。这样，SSQ 就是所有点和该点所属簇的核心对象的距离平方和。从图 4 可以看出 PDStream 的聚类质量优于 STREAM，其原因正是 STREAM 算法过分依赖于历史数据，新数据所起的作用不明显，而 PDStream 引用概要数据很好地解决了这一点。

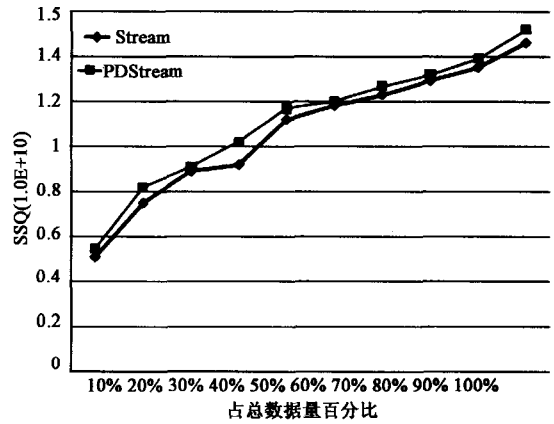

图 4 聚类质量比较

3.4 参数影响

由于 PDStream 算法涉及到的参数比较多，而且它们都对聚类质量和执行时间有一定影响，如确定主成分个数的累积贡献率大小和确定核心对象的半径大小等参数。在此只选取累积贡献率进行分析，算法中我们将其固定为 85%，此时为了进行比较，将其值不变调整，观察其对执行时间的影响，结果如图 5 所示。

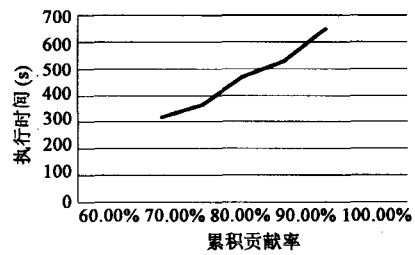

图 5 参数影响图

从图 5 中可以看出，随着累积贡献率不断变大，算法执行时间也不断增长，成正比例关系，与理论上一致。仔细分析可知，当累积贡献率变大时，其从原始数据流中构造的主成分将不断增多，正是由于主成分个数的变大，导致调入内存中的数据量增大，所以执行时间会变长。适当设置累积贡献率将会起到

更佳的聚类效果。

4 结束语

本文提出了一个基于主成分和密度的数据流聚类算法 PDStream。顺带提出了两个辅助算法 PCA 算法和 IDBSCAN 算法，MPCA 算法实质是对流入时间窗口的数据进行预处理，它有严格的数据证明并且能够得到良好的效果，而 MDBSCAN 是在空间密度聚类基础上改进而得到的。PDStream 算法采用主成分和密度两阶段模型，并加入二次聚类过程，成功解决了 STREAM 算法聚类受控于历史数据和 CluStream 算法难以描述非球形簇和难以在线淘汰“老数据”的不足。PDStream 能得到高质量的聚类结果而耗费少量的时间和内存。它是单遍扫描、增量更新的算法，能通过密度的参数值处理噪声和过时的数据。在真实数据集上的实验结果表明了该算法的有效性和准确性。如何精确设计滑动窗口中不断变化的概要数据以及简化 PDStream 算法的结构，更有效地提高算法的性能和减少算法复杂度将是后期研究的主要内容。

参 考 文 献

[1] Muthukrishnan S. Data streams algorithms and applications [C] // Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia: Society for Industrial and applied Mathematics, 2003: 413-423.

[2] Guha S, Koudas N. Approximating a data stream for querying and estimation: algorithms and performance evaluation [C] // Proceedings of the 18th International Conference on Data Engineering (ICDE). San Jose, California, USA, 2002: 567-576.

[3] Chalaghan L O, Mishra N, Meyerson A, Guha S. Streaming data algorithms for high-quality clustering [C] // Proceedings of the 18th Int'l Conf. on Data Engineering. San Jose, 2002: 685-694.

[4] Domingos P, Hulten C. Mining high-speed data streams [C] // Proceedings of the KDD, 2000.

[5] Aggarwal C C, Han J, Wang J, et al. A framework for projected clustering of high dimensional data streams [C] // Proceedings of the VLDB. Toronto, Morgan Kaufmann Publishers, 2004: 852-863.

[6] 常建龙，曹锋，周傲英. 基于滑动窗口的进化数据流聚类[J]. 软件学报，2007，18(4)：905-918.

[7] 朱蔚恒，印鉴，谢益煌. 基于数据流的任意形状聚类算法[J]. 软件学报，2006，17(3)：379-387.

[8] Guha S, Mishra N, Motwani R, et al. Clustering data streams [C] //FOCS, 2000: 359-366.

[9] Aggarwal C, Han J, Wang J, et al. A framework for clustering evolving data streams [C] // Proceedings of the 29th International Conference on Very Large Databases. Berlin, Germany: Morgan Kaufmann Publishers, 2003: 81-92.

[10] 梅长林，范金城，等. 数据分析方法[M]. 北京：高等教育出版社，2006：115-117.

[11] 王顺绪，戴华. 求解大型矩阵特征值问题的并行块 Davidson 方法[J]. 南京航空航天大学学报，2007，39(6)：814-818.

[12] 冯少荣，肖文俊. DSBCAN 聚类算法的研究与改进[J]. 中国矿业大学学报，2008，37(1)：105-111.

[13] Aggarwal C C, Han J, Wang J, et al. A framework for clustering evolving data streams [C] // Proceedings of the Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003: 81-92.

[14] Guha S, Meyerson A, Mishra N, et al. Clustering data streams: Theory and practice [J]. IEEE Trans. On Knowledge and Data Engineering, 2003, 3(15): 515-528.

[15] O'Callaghan L, Mishia N, Meyerson A, et al. Streaming-data algorithms for high-quality clustering [C] // Proceedings of the IEEE International Conference on Data Engineering. San Jose, California, USA: IEEE Computer Society, 2002: 685-699.

（责任编辑 马 兰）