一种时间序列数据的动态密度聚类算法

陈 皓†, 冀敏杰, 郭紫园, 夏 雨

(西安邮电大学 计算机学院, 陕西 西安 710121)

摘要: 传统的聚类算法多是针对某个时间片上的静态数据集合进行的聚类分析, 但事实上大部分数据存在时间序列上的连续动态演变过程. 本文对时间序列数据及其类结构的演变过程进行了分析, 发现在一定条件下相邻时间片间的数据集间存在较强的关联性, 并且类簇结构间则存在一定的继承性. 故本文得出新的思想, 在前一时间片聚类结果的基础上, 通过对部分变化数据的计算和类簇结构的局部调整就有望获得对后一时间片上数据进行完全聚类相同的效果, 且运算量会显著下降. 基于此思想提出了一种时间序列数据的动态密度聚类算法(DDCA/TSD). 仿真实验中使用6种数据集对所提出算法进行了实验验证. 结果显示DDCA/TSD在保证聚类准确性的基础上相对传统聚类算法有明显的时间效率提升, 并能更有效地发现数据点的属性变化及类簇结构的演变过程.

关键词: 时间序列数据; 数据关联性; 动态密度聚类; 类继承性

引用格式: 陈皓, 冀敏杰, 郭紫园, 等. 一种时间序列数据的动态密度聚类算法. 控制理论与应用, 2019, 36(8): 1304 – 1314

DOI: 10.7641/CTA.2019.80976

A dynamic density clustering algorithm for time series data

CHEN Hao†, JI Min-jie, GUO Zi-yuan, XIA Yu

(School of Computing Science, Xi'an University of Posts and Telecommunications, Xi'an Shaanxi 710121, China)

Abstract: The traditional clustering algorithms are an analysis method for static data sets on a certain time slice, but most of the data sets have a continuous dynamic evolution process on the time series. A high data correlation and cluster structure succession between adjacent time slice are found by the analysis of the successional process on time series data and its class structure. Consequently, based on the clustering results of the previous time slice, it is able to obtain the same effect as the result of completely clustering data on the latter time slice through calculating part changed data and adjusting partial cluster structure, meanwhile the whole computation will significantly decrease. Based on this idea, a dynamic density clustering algorithm for time series data (DDCA/TSD) is proposed. In simulations, six kinds of data sets are used to verify the proposed algorithm. The results show that DDCA/TSD has obvious time efficiency improvement compared with the traditional clustering algorithm on the basis of the cluster accuracy. Moreover, it is effective to find the change of data points' attribute and the evolution of cluster structure.

Key words: time series data; data correlation; dynamic density clustering; cluster succession

Citation: CHEN Hao, JI Minjie, GUO Ziyuan, et al. A dynamic density clustering algorithm for time series data. *Control Theory & Applications*, 2019, 36(8): 1304 – 1314

1 引言

时间序列数据是指按时间先后顺序排列的数据序列, 它潜在的表达了数据内部状态随时间变化的规律. 时间序列数据分析广泛应用于生物、医疗、教育和经济等各个领域[1], 其中聚类作为主要研究方法越来越受到学者的重视[2]. 目前, 一些时间序列聚类算法是将时间序列模拟为n维空间的某个点或提取时间序列数据的特征向量, 再利用静态聚类方法进行处理[3]. 此类方法缺乏对每个时间片状态以及其数据的分析, 无法挖掘出时间片之间数据的具体演变过程, 只是对整个时间序列采取整体或者抽象的处理方法. 另一些方法是对于每个时间片进行一次静态聚类[4]. 这种方法可以挖掘出时间片之间数据的演变过程, 但是无法有效利用相邻时间片间的数据进行分析, 且每次静态

收稿日期: 2018−12−14; 录用日期: 2019−04−17.

†通信作者. E-mail: chenhao@xupt.edu.cn; Tel.: +86 13720400364.

本文责任编委: 孙长银.

国家自然科学基金项目(61876138, 61203311, 61105064), 陕西省教育厅自然科学专项(17JK0701), 陕西省网络数据分析与智能处理重点实验室开放课题基金项目(XUPT–KLND(201804)), 西安邮电大学创新基金项目(103–602080016)资助.

Supported by the National Natural Science Foundation of China (61876138, 61203311, 61105064), the Scientific Research Program Funded by Shaanxi Provincial Education Department of China (17JK0701), the Science Foundation of the Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing (XUPT–KLND(201804)) and the Innovation Funds of Xi'an University of Posts and Telecommunications (103–602080016).

聚类操作对象都是全数据集. 当数据量比较大时, 计算量会随之增大, 时间消耗也会随之增加[5].

针对以上问题, 本文基于密度聚类过程, 提出一种利用时间片之间的关联性, 通过局部的计算和类簇结构调整达到全局聚类效果的动态聚类方法. 该方法充分利用了类结构在时间序列上的继承性, 不仅能够达到传统聚类模式的运算效果, 而且能够揭示类簇结构的变化和每个数据点的属性在时间序列上的变化过程, 同时显著降低整体计算量.

2　相关工作

根据数据和聚类的动态性, 聚类分析基本可分为4种类型[6]:

其一是数据是静态的, 集群过程也是静态的. 典型的如MacQueen[7]提出的k–means算法, 聚类速度快, 可以用于各种数据类型, 但初始点的选取对聚类效果有直接影响. 针对此问题Rodriguez A等人[8]提出的类中心都处在局部密度比较大的位置, 且距离具有比它更大的局部密度的对象相对较远. 基于层次聚类的代表BIRCH[9] 算法聚类效率高, 可识别噪声点, 但是不能发现任意形状的类. Ester等人[10]针对任意形状的类, 采用密度概念提出了基于密度的噪声应用空间聚类(density-based spatial clustering of applications with noise, DBSCAN)算法, 该算法能够发现任意形状并且能够识别噪声点, 有对输入参数敏感的缺点. 秦佳睿等人[11]则采用自动适应的半径来解决输入参数敏感问题.

其二是数据被视为静态数据, 但是集群的数量在每次新的计算时可动态变化. 如张阳等人[12]通过删除由信息动态变化而产生的冗余信息, 来减少动态聚类过程中的干扰, 但是对时间序列中集群数量变化的情况却不适用.

其三数据被视为动态的、随时间变化的, 但集群的数量是固定的. Agrawal等人[13]首次提出使用离散傅里叶变换将时间序列的时间域转换为频率域的特征表示方法. Benítez I等人[14]基于k–means算法采用hausdorff相似度的相似距离, 对时间序列的住宅用电量进行动态聚类分析. 王玲等人[15]通过选择一些蕴含重要信息的特征点, 降低时间序列维度, 但需保证原有时间序列趋势不变.

其四数据是动态的且集群的数量在每次迭代中都是动态变化的. Luczak等人[16]考虑动态时间弯曲及扩展动态时间方法对不等长时间序列进行聚类的优势, 提出了一种新的距离度量标准, 能够较好的通过不同时间序列的距离度量实现时间序列的聚类. 文献[17]利用Haar小波变换表示时间序列数据, 采用k–means算法和欧氏距离对新特征空间中的数据进行聚类. 文献[11] 描述了一种称为泛函模糊C均值或实用模糊C均值 (functional fuzzy C–means, FFCM)的算法, 它是模糊C均值(fuzzy C–means, FCM)的时间序列推广. Izakian和Pedrycz[18]提出了对时间序列数据进行模糊聚类的欧氏距离的增广式, 但是复杂度比较高. 谢福鼎等[3]通过等长处理时间序列后, 基于欧氏距离及模糊C均值聚类算法实现时间序列聚类, 计算复杂度低, 容易实现. 但存在要求时间序列长度一致且对时间轴变化敏感等问题. 孙雅与李志华[19]基于动态弯曲距离提出了一种时间序列层次聚类的算法, 但该算法相似性度量计算复杂度较高, 影响了算法效率. 刘琴等人[20]结合滑动窗口及k–means聚类算法提出了一种基于滑动窗口不等长时间序列的短时间序列距离的聚类算法, 能够解决不等长时间序列聚类的问题, 但最佳聚类个数难以确定. Shukri S等人[21]基于一个受自然启发的元启发式算法, 能够自动检测集群数量, 但是对高维数据聚类效果不好.

本文所讨论问题属于第4类问题. 上述研究基本是将时间序列模拟为n维空间的某个点或者提取时间序列数据的特征向量, 其本质依然是利用静态聚类方法处理时间序列数据, 并没有挖掘出每个数据的变化过程, 同时缺乏连续时间片间数据关联性的分析.

3　时间序列数据及聚类过程分析

3.1　时间序列上数据的关联性和类结构的继承性

不失一般性, 一个时间片上的静态聚类过程可描述如下: 数据集ξ由n个点组成, $\xi=\{X_1, X_2, \cdots, X_n\}$, 且每个点可用一个$d$维向量表示$X_i=\{X_{i1}, X_{i2}, \cdots, X_{id}\}$. 给定数据集$\xi$, 算法通过找到一组簇类集合$C=\{C_1, C_2, \cdots, C_k\}$使得簇间结构尽可能不同, 而簇内结构尽可能相似. 其中每个簇的质心点由CE_j表示, $j=1, 2, \cdots, k$. 与之不同的是, 时间序列数据的聚类是一个连续且动态的过程, 其中两个相邻时刻的数据及簇类变化可表示为: t_i时刻集合ξ^i中n_i个元素构成了k_i个类. 而在t_{i+1}时刻, 集合ξ^{i+1}中元素数量变为n_{i+1}且构成k_{i+1}个类. 在两个相邻时刻间, 元素将具有消失、新增、位移或不变4种状态, 从而影响t_{i+1}时刻的数据集ξ^{i+1}及簇类C^{i+1}. 以下是上述几种元素状态的定义:

定义 1　元素消失. 元素X_j在t_i时刻有$X_j^i\in\xi^i$, 而t_{i+1}时刻有$X_j^{i+1}\notin\xi^{i+1}$.

定义 2　元素新增. 元素X_j在t_i时刻有$X_j^i\notin\xi^i$, 在t_{i+1}时刻有$X_j^{i+1}\in\xi^{i+1}$.

定义 3　元素位移. 元素X_j在t_i时刻有$X_j^i\in\xi^i$, 在t_{i+1}时刻元素$X_j^{i+1}\in\xi^{i+1}$, $X_j^i\neq X_j^{i+1}$.

定义 4　元素不变. 元素X_j在t_i时刻有$X_j^i\in\xi^i$, 在t_{i+1}时刻元素$X_j^{i+1}\in\xi^{i+1}$, $X_j^i=X_j^{i+1}$.

元素的变化首先会导致类结构的变化, 从而进一步影响两个相邻时间片上类结构的变化. 假设t_{i+1}时刻相对t_i时刻有w_1个元素不变且其概率为p_1, w_2个元

素消失, 概率为p_2; w_3个元素新增, 概率为p_3; w_4个元素位移, 概率为p_4. 那么这种情况下从ξ^i变化到ξ^{i+1}的概率α为

$$\alpha = p_1^{w_1} p_2^{w_2} p_3^{w_3} p_4^{w_4}. \tag{1}$$

为了便于分析, 上述4种情况可分为元素发生变化和元素不发生变化两种. 设元素位移、消失、增加3种元素变化的概率为p. 由于元素变化和元素不变具有互补性, 则元素不变的概率为$1-p$, 则从ξ^i变化为ξ^{i+1}的概率α为

$$\alpha = (1-p)^{w_1} p^{w_2+w_3+w_4}. \tag{2}$$

由于t_{i+1}时刻元素数量为n_{i+1}, 而$n_{i+1}=w_1+w_2+w_3+w_4$, 则式(2)可写为

$$\alpha = (1-p)^{w_1} p^{n_{i+1}-w_1}. \tag{3}$$

显然, t_{i+1}时刻数据集的结构与w_1有直接关系. 故对式(3)中w_1求偏导数有

$$\frac{\partial \alpha}{\partial w_1} = (1-p)^{w_1} p^{n_{i+1}-w_1} [\ln(1-p) - \ln p]. \tag{4}$$

当式(4)为0时, $p=0.5$是一个极值点, 且$p<0.5$时, 函数随w_1增大递增, 当$p>0.5$时则相反.

结论 1 当$p<0.5$时, 相邻时间片上的数据集间具有较强的关联性.

显然, 在当$p<0.5$的条件下, 随着w_1的增大, 相邻时间片上数据集间的关联性会产生C^{i+1}与C^i间明显的继承性. 那么元素消失、元素增加或元素位移这几种变化模式会对类结构的继承性产生什么影响呢? 下面将分别进行分析.

1) 改变元素以消失元素为主.

当w_3和w_4较小时, 可认为发生改变的元素以消失为主体, 即$w_2 \approx n_{i+1}-w_1$. 这种情况下, 主要讨论元素在每个子类中消失的数量与整体类结构稳定性之间的关系. 以下是元素在集合中消失的两种极端情况: 其一, 消失元素相对均匀的分布在每个子类中. 这种情况下任意子类中的元素相对消失的数量是最少的; 其二, 消失元素主要集中于某个子类中. 这种情况下该子类的结构会受到较严重的破坏. 综合而言, 一个子类中个体消失数量越少, 其质心发生移动的概率就越低. 故可认为情况一导致t_{i+1}时刻整个类结构发生变化的概率较低, 而情况二则相反.

在$p<0.5$的条件下, 下一时刻新簇类结构的变化会主要受上述两种情况的哪种影响呢? 假设消失元素较集中分布在某个子类中的概率为π_1, t_i时刻类的总数为k, 子类C_k^i中有n_j^i $(1 \leqslant n_j^i \leqslant n_{i+1}-w_1)$个元素, 则$\pi_1$的概率是每次选中某个子类的概率$\beta = \dfrac{1}{k}$, 乘以该子类全部元素消失的概率$\gamma = \prod\limits_{y=0}^{n_j^i-1} \dfrac{1}{n_j^i - y}$, 然后乘以$n_{i+1}-w_1$个元素变化的概率$p$, 即$\pi_1$的概率为

$$\pi_1 = \Big(\prod_{y=0}^{n_j^i-1} \frac{1}{(n_j^i - y)k}\Big) p^{n_{i+1}-w_1}, \tag{5}$$

其中y指某个子类中元素消失的个数, 消失的个数每次增加1.

消失元素较均匀分散在每个子类中的概率为π_2, 显然, π_2和π_1是相对互补的, 则可认为

$$\pi_2 = 1 - \pi_1. \tag{6}$$

由式(5)可知, 当$(n_j^i-y)k$大于0时, π_1递减. 故当$(n_j^i-y)k$取最小值时, π_1取最大值. 由实际情况可知, 子类C_k^i中元素个数$n_j^i \geqslant 1$, 子类数目$k \geqslant 1$, $0 \leqslant y \leqslant n_j^i-1$. 由此可得, 当$n_j^i=1$, $k=1$, $y=0$时, $(n_j^i-y)k$取最大值1. 故式(5) 最大值为

$$\pi_1 = p^{n_{i+1}-w_1}. \tag{7}$$

式(7)中, 当$0 \leqslant n_{i+1}-w_1$时, π_1递减. 由实际情况可知$1 \leqslant n_{i+1}-w_1$. 故在$p<0.5$的前提下, π_1最大值趋近0.5. 又结合式(6)可得$\pi_1 \leqslant \pi_2$.

此外, 当子类元素n_j^i或类数目k较大时, π_1更小而π_2更大, 即消失元素全部集中在某一个子类中的概率更小而消失元素均匀的分布在每个子类中的概率更大. 由此可得, 元素的改变以消失为主时相邻时刻的类结构具有较强的继承性.

2) 改变元素以新增元素为主体.

当w_2和w_4较小时, 可认为发生改变的元素以新增为主体, 即$w_3 \approx n_{i+1}-w_1$. 这种情况下, 主要讨论元素在每个子类中新增的数量与整体类结构稳定性之间的关系. 以下是两种极端情况: 其一, 新增元素分布在原有子类中, 这种情况下又分为两种小情况: ① 新增元素相对均匀的增加在原有子类中, 这种情况质心不会发生很大移动, 原有类结构相对稳定. ② 新增元素较为集中的增加在某个子类中, 这种情况质心可能会有较大位移, 从而导致类结构破坏. 此情况导致类结构破坏的过程如图1中(a)所示, 新增元素为灰色图形, 原有的两个类将合并为一个类. 但是, 导致这种情况必须有大量的新增元素. 在$p<0.5$的条件下其发生概率较低. 其二, 新增元素都不在每个子类中, 这种情况下子类整体结构会受到较严重的破坏. 这种情况导致类结构破坏如图1中(b)所示会新增一个子类, 新增元素为灰色图形.

由上述情况分析可知, 某子类中新增元素的数量越少, 类结构被破坏的概率越低. 另外, 新增元素相对均匀的增加在每个子类中, 类结构被破坏的概率较低. 相反, 只有新增元素集中于某个子类中时, 整体类结

构被破坏的概率较高. 下面分析新增元素相对均匀的增加在每个子类中的概率与新增元素集中在一个子类中的概率.

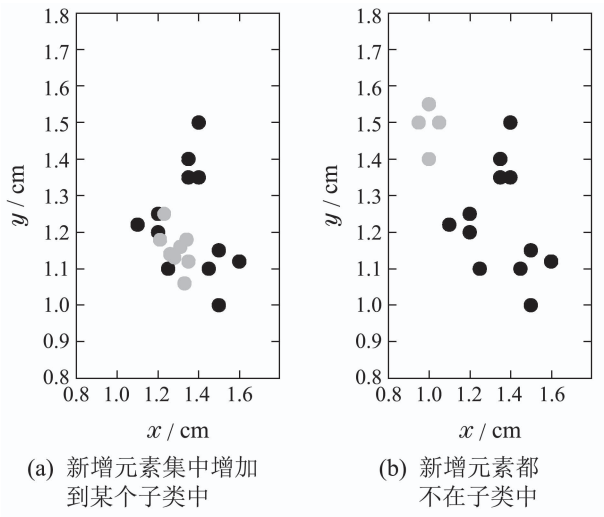


图 1 新增元素可能导致类结构破坏的情况

Fig. 1 The added elements may cause damage to the class structure

假设新增元素集中在一个子类中的概率为π_3, 而新增元素属于这一新增子类的概率为q, 则π_3为

$$\pi_3 = p^{n_{i+1}-w_1}\, q^{n_{i+1}-w_1}. \tag{8}$$

同理, 新增元素均匀分布在原有子类中的概率为n_4, 而π_4则相对互补于π_3, 即

$$\pi_4 = 1 - \pi_3. \tag{9}$$

由式(8)可知, 当$0 \leqslant n_{i+1}-w_1$时, π_3递减, 而$1 \leqslant n_{i+1}-w_1$, 故π_3的最大值为pq . 在p, q <0.5的前提下可得$\pi_3 \leqslant \pi_4$. 综上可见, 在p <0.5的条件下, 当改变元素以新增为主时相邻时刻类结构间依然具有较好的继承性.

3) 改变元素以位移元素为主体.

元素位移的本质是同一个元素在旧位置消失, 然后在新位置新增. 故元素位移可认为是消失和新增两种情况的叠加, 由前述分析可得, 在p, q <0.5的条件下, 当改变元素以位移为主时相邻时刻类结构间具有一定的继承性.

结论 2　在p<0.5条件下, 消失、新增或位移几种元素变化情况都会使C^{i+1}相对C^i具有明显的继承性.

3.2　经典算法在时间序列上的聚类过程分析

时间序列数据的关联性及其类结构的继承性为动态聚类过程提供了基础, 那么划分型聚类、层次型聚类以及密度聚类哪种更适合呢? 下面本文将以元素消失为例针对k–means算法, BIRCH算法和DBSCAN算法进行讨论. 图2中(a)和(b)分别为两个相邻时刻的数据集合及k = 2时k–means聚类的结果, 其中元素F在t_{i+1}时刻消失. 聚类结果中: 黑色表示类1, 灰色表示类2, “×”表示类的质心. 显然, 元素F消失后, 类2的质心会向左偏移. 由于质心发生了位移, 需要对剩余元素重新划分类的归属, 而元素G距离类2的质心比较近, 故元素G将归属于类2. 可见, 当元素的改变引起类质心移动时, k–means算法往往需要一个完全计算过程来获得一个新的类结构.

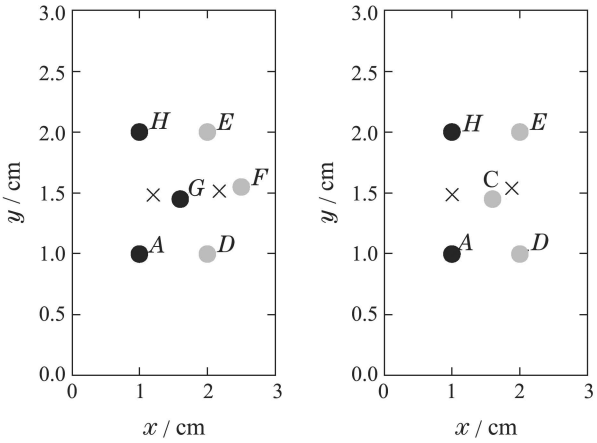


图 2　t_i时刻和t_{i+1}时刻k–means聚类结果

Fig. 2 The k–means clustering results of t_i and t_{i+1}

使用BIRCH算法对元素F消失的情况进行分析, 图3中(a)与(b)为t_i时刻和t_{i+1}时刻采用BIRCH算法聚类的结果. 圈中的数字代表每次合并类的顺序. BIRCH聚类算法的合并过程是每次选取距离最近的两个元素进行合并. 由图3中(a)可知, 当元素F消失后, 第2次合并的结果将发生变化, 元素E将和元素G, D进行合并. 显然, 元素的消失影响了元素间的距离矩阵, 这意味着只有重新构造距离矩阵才可完成新的聚类, 因此无法通过局部的计算来实现动态聚类过程.

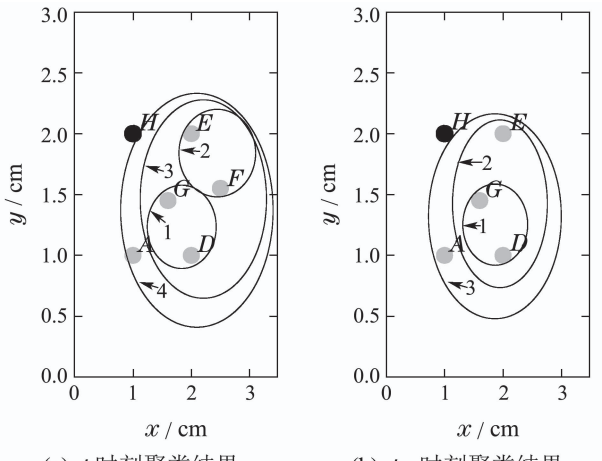
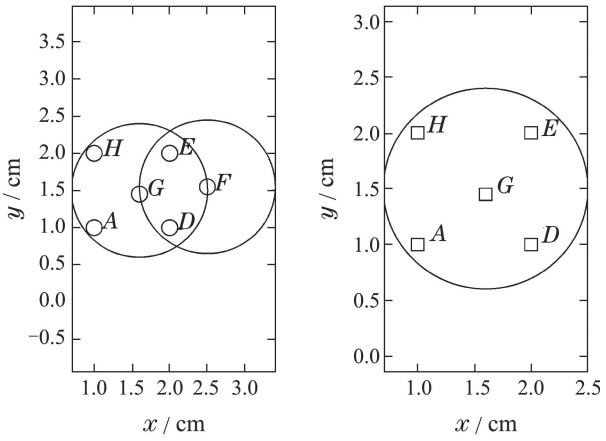


图 3　t_i时刻和t_{i+1}时刻BIRCH聚类结果

Fig. 3 The BIRCH clustering results of t_i and t_{i+1}

在密度聚类中, Eps表示空间中以任意一点为圆心的半径长度, 而MinPts表示给定的阈值, 是一个由用户输入的参数. 若某一点的给定邻域Eps内的元素数目超过给定的阈值MinPts, 那么该点就是核心点. 其边界点落在某个核心点的Eps邻域内, 而噪声点是不属于核心点和边界点的任何点. 此外, 任意两个足够靠近(在给定的Eps邻域之内)的核心点将属于同一个类中, 任何与核心点足够靠近的边界点也与该核心点属于同一类. 若使用DBSCAN算法对前一例子进行计算, 其结果将如图4所示.



(a) t_i时刻聚类结果 (b) t_{i+1}时刻聚类结果

图 4 t_i时刻和t_{i+1}时刻DBSCAN聚类结果

Fig. 4 The DBSCAN clustering results of t_i and t_{i+1}

当元素F消失后, 由密度聚类性质可知, 元素F的Eps邻域元素G, D, E可能受到影响. 按照密度聚类的概念, 可能受到影响的元素G, D, E将重新进行判断属性以及类别, 在此基础上就可达到图4中(b)所示的结果. 从这个简单的过程中可发现, 从t_i时刻到t_{i+1}时刻聚类结果的过程中, 仅仅改变局部相关元素的属性就可以达到全局数据聚类的效果.

综上所述, 基于密度聚类过程能够通过局部的少量计算重新完成类结构的动态构建, 这为时间序列数据中关联性分析提供了较好的途径.

4 时间序列数据的动态密度聚类算法

在上述分析的基础上, 本文提出了一种以密度聚类为基础的时间序列数据动态聚类方法, 称之为时间序列数据的动态密度聚类算法(dynamic density clustering algorithm for time series data, DDCA/TSD), 该算法包括了元素消失(elements remove, ER)、元素新增(elements add, EA)及元素位移(elements displace, ED)3种元素改变形式下的簇类结构局部调整操作. 以下是算法步骤.

DDCA/TSD:

输入: ξ^i: t_i时刻数据集合;

Queue: t_i时刻到t_{i+1}时刻元素变化队列;

Eps: 密度聚类半径参数;

MinPts: 密度聚类最小数据阈值参数.

输出: index: 聚类后所有对象的类标签;

attribute: 聚类后所有对象的属性标签.

1) QueueRemove = null; /*元素消失队列*/,
QueueAdd = null; /*元素新增队列*/,
QueueDisplace = null; /*元素位移队列*/;
2) while Queue! = null do;
3) data = Queue.pop();
4) if data是元素消失,
5) add data to QueueRemove;
6) elseif data是元素新增,
7) add data to QueueAdd;
8) elseif data是元素位移,
9) add data to QueueDisplace;
10) end if
11) end while
12) ER (ξ^i, QueueRemove);
13) EA (ξ^i, QueueAdd);
14) ED (ξ^i, QueueDisplace).

算法将发生变化的数据按其类型存储在3个队列中, 然后调用3个过程进行批量处理, 以减少调用次数多, 提高计算效率.

4.1 元素消失处理过程

当一个元素消失后, 由Eps邻域的概念可知, 每次变化的元素能够对以该点为中心的Eps邻域内的元素产生直接影响. 因此, 消失元素能够直接影响Eps半径内元素的属性, 被影响的Eps半径内的元素将可能传递地影响它们的Eps半径内的元素. 如图5是二维空间下消失元素对周围元素影响的实例.

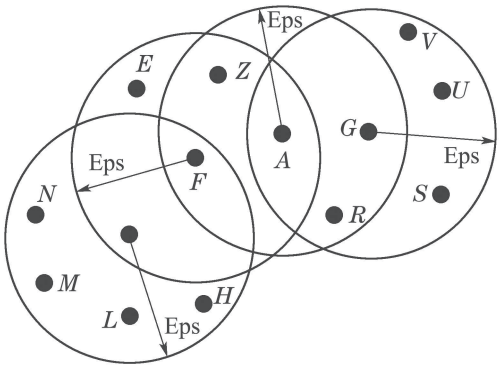


图 5 元素消失或元素新增对周围数据影响

Fig. 5 The influence of surrounding elements in ER or ED process

设元素A是消失元素, 圆表示某元素的Eps邻域, 假设MinPts = 4, 由于元素A的消失, 元素F由核心点

变为边界点, 元素E和元素Z受到传递影响, 由边界点变为噪声点. 原本一个大类被分为由D 为核心点和G为核心点的两个新类. 因此, 在元素消失的过程中, 主要思想为: 删除该元素, 并且查看该点邻域内点的属性是否变化, 如果变化, 则改变对应的属性, 对属性改变的点再查看该点邻域内点的属性是否变化, 直到邻域内点的属性都不再变化为止.

ER过程:

输入: ξ^i: t_i时刻数据集合;

QueueOne: 消失元素队列.

输出: ξ^{i+1}: 元素消失后的集合;

index: 元素消失后对象的类标签;

attribute: 元素消失后对象的属性标签.

1) Array=[·], num=QueueOne.length, $n = 0$;
2) while QueueOne! = null do
3) A = QueueOne.pop(·);
4) if n<QueueOne.length
5) delete A from ξ^i;
6) end if
7) 将点A为圆心Eps为半径内的点加入Array;
8) for $i = 0$ to Array.length do
9) if Array[i]的属性发生变化且Array[i]没有标记
10) 改变Array[i]的属性;
11) add Array[i] to QueueOne;
12) 标记Array[i];
13) delete Array[i] from Array;
14) end if
15) end for
16) n++; /*只把QueueOne最初的数据删除*/
17) end while

4.2　元素新增处理过程

元素新增是元素消失的逆过程. 一个新增元素能够直接影响Eps半径内的数据点的属性, 被影响的Eps半径内的元素将可能传递地影响它们的Eps半径内的元素. 设图5中点A是新增元素, 圆表示某元素的Eps邻域, MinPts =4. 由于点A的新增, 点F由边界点变为核心点, 点E由噪声点变为边界点. 原来的两个类也因点A的加入而合并为一个大类.

元素新增和元素消失过程类似, 都是查找变化元素的Eps邻域中属性变化的点, 对属性改变的点再查看该点邻域内点的属性是否变化, 直到邻域内点的属性都不再变化为止.

EA过程:

输入: ξ^i: t_i时刻数据集合;

QueueOne: 新增元素队列.

输出: ξ^{i+1}: 元素新增后的集合;

index: 元素新增后对象的类标签;

attribute: 元素新增后对象的属性标签.

1) Array=[·], num=QueueOne.length, $n = 0$;
2) while QueueOne! = null do
3) A =QueueOne.pop();
4) if n<QueueOne.length
5) add A to ξ^i;
6) end if
7) 将点A为圆心Eps为半径内的点加入Array;
8) for $i = 0$ to Array.length do
9) if Array[i]的属性发生变化且Array[i]没有标记
10) 改变Array[i]的属性;
11) add Array[i] to QueueOne;
12) 标记Array[i];
13) delete Array[i] from Array;
14) end if
15) end for
16) n++; /*只新增QueueOne原始的数据*/
17) end while

4.3　元素位移处理过程

元素位移可视为元素消失和元素新增两个过程的叠加, 故ED过程可由ER过程和EA过程组合而成. 图6用来具体说明元素位移中每个数据以及簇内结构的变化过程. 假设图5是t_i时刻数据集, 元素A按照图6中的箭头方向移动到新的位置形成t_{i+1}时刻数据集. 采用DDCA/TSD算法对变化后数据集进行聚类, 具体的元素属性以及类别变化过程如表1和表2所示. 其中类别中不同的数字代表不同的类. 在属性描述中, 1代表核心点, 0代表边界点, –1代表孤立点, Nan代表该元素消失.

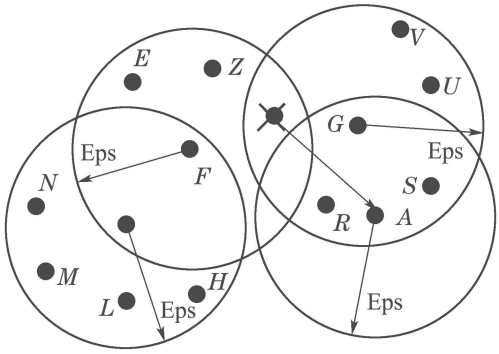


图 6　素消位移对周围数据影响

Fig. 6　The influence of surrounding elements in ED process

由表1和图5可知, 在t_i时刻所有元素都为同一个大类. 经过位置变化以后, 元素A, G, R, S, U, V变为

类别2, 元素E和元素Z变为噪声点类, 一个大类分成3个小类. 由表2可知, 在t_i时刻元素A, D, F, G是核心点, 元素A经过ER操作之后, 元素F由核心点变为边界点, 元素E和元素Z由边界点变为噪声点. 然后元素A经过EA操作之后, 其他元素的属性以及类别没有发生改变, 说明元素A的新增对簇内结构没有造成影响. 最终形成图6所示t_{i+1}时刻聚类的结果. 因此, DDCA/TSD算法可以清楚的显示簇内结构每个元素属性以及类别的变化过程, 而这些变化过程是传统聚类算法挖掘不到的.

表 1 元素在t_i时刻和t_{i+1}时刻的类别

Table 1 The category of the elements on t_i and t_{i+1}

元 素	A	D	E	F	G	H	L	M	N	R	S	U	V	Z
t_i时刻类别	**1**	1	**1**	1	**1**	1	1	1	1	**1**	**1**	**1**	**1**	**1**
t_{i+1}时刻类别	**2**	1	**–1**	1	**2**	1	1	1	1	**2**	**2**	**2**	**2**	**–1**

表 2 元素在t_i时刻, 元素消失后和t_{i+1}时刻的属性

Table 2 The attribute of the element on t_i and t_{i+1} after element A disappear

元 素	A	D	E	F	G	H	L	M	N	R	S	U	V	Z
t_i时刻类别	**1**	1	**0**	**1**	1	0	0	0	0	0	0	0	0	**0**
元素A消失后属性	**Nan**	1	**–1**	**0**	1	0	0	0	0	0	0	0	0	**–1**
t_{i+1}时刻类别	**0**	1	**–1**	**0**	1	0	0	0	0	0	0	0	0	**–1**

5 实验

在PC (2.2 GHzi3, 2 GB RAM, Windows 7) 上使用MATLAB 2014a实现算法并使用经典聚类代表的真实数据集Seeds[22]、规模较大的数据集Mushroom[22]和Anuran Calls (采用梅尔频率倒谱系数提取特征, Mel-frequency cepstral coefficients, MFCCs)[22]、SEQUO IA 2000数据库进行测试. Seeds, Mushroom和Anuran Calls(MFCCs)数据集均来自机器学习UCI数据库中的数据. 数据集Seeds由波兰科学院农业物理研究所创建的, 数据集代表了3种不同类型的小麦. Mushroom 数据集包括23种蘑菇的假设样品的描述, 共有8124个实例、两个类别和22个属性. Anuran Calls(MFCCs)数据集从Anuran的音节中提取的声学特征, 该数据集包含4个类别、7195个实例和22个属性. 表3为上述数据集的具体信息. 数据库SEQ[10]是公开的SEQUO IA 2000数据的一个子集, 同时包含多个任意形状的、不同密度的数据簇以及重叠子簇.

表 3 真实数据集信息

Table 3 The information of real data set

数据集	实例个数	属性个数	类别数目	类实例个数
Seeds	210	7	3	70–70–70
Mushroom	8124	22	2	4208–3916
Anuran Calls (MFCCs)	7195	22	4	68–542–2165–4420

本文对真实数据集和标准数据集进行了时间序列化处理. 处理过程为每次在数据集范围内随机消失、改变及位移数据集中30%的数据, 一共进行5次这样的操作, 形成5个时间片上连续的时间序列数据集.

5.1 评价指标

本文采用准确率 (accuracy, ACC)、纯度 (purity, PR)、召回率 (recall, RE)、兰德指数 (adjusted rand index, ARI)、运行时间(T, 单位ms)和参与计算数据个数(numbers, NUM)共6个评价指标对所提算法进行了有效性评价. ACC表达了单个时间片上聚类正确的比例, PR表示单个时间片上预测为正的样本中有多少是对的, RE表示单个时间片上样本中的正例有多少被预测正确, ARI用来衡量单个时间片上两个数据分布的吻合程度, T表达了单个时间片上算法运行的时间, NUM表达了动态时间序列数据基于前一时间片聚类结果上, 参加本次时间片聚类数据点的数目. 其中: ACC, PR, RE, ARI, T为某时间片上的静态指标, T用来评价聚类时间效率. NUM为时间序列数据动态参数, 用来评价计算开销. 如果聚类结果越接近数据集的真实划分, 那么ACC, PR, RE, ARI的值就越大, 最大值为1. 也就是说, 这4个评价标准的值越大, 聚类质量越好, 算法越有效. NUM最大值为数据集样本总数的值, T越小说明聚类速度越快, NUM越少说明聚类计算开销越小. 以下是上述指标的定义:

$$\mathrm{ACC} = \frac{1}{n} \max_{j_1,\, j_2, \cdots,\, j_k \in S} \sum_{i=1}^{k} n_{ij_i}, \tag{10}$$

$$\mathrm{PR}=\frac{1}{k}\sum_{i=1}^{k}\frac{n_{ij^*{}_i}}{b_i},\tag{11}$$

$$\mathrm{RE}=\frac{1}{k}\sum_{i=1}^{k}\frac{n_{ij^*{}_i}}{c_i},\tag{12}$$

$$\mathrm{ARI}=\frac{\sum_{ij}\binom{n_{ij}}{2}-[\sum_{i}\binom{b_i}{2}\sum_{j}\binom{c_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_{i}\binom{b_i}{2}+\sum_{j}\binom{c_j}{2}]-[\sum_{i}\binom{b_i}{2}\sum_{j}\binom{c_j}{2}]/\binom{n}{2}},\tag{13}$$

其中: n为数据集ξ的总数; k为聚类结果类的总数; b_i和c_j分别是B_i和C_j中对象的个数; $n_{ij}=|B_i\cap C_j|$, $n_{ij}{*_i}$表示正确分到第i类中的对象个数.

5.2　算法性能比较

本文选择Python科学计算库中的sklearn模块的DBSCAN算法、改进的自适应快速算法(adaptive and fast density-based spatial clustering of applications with noise, AF–DBSCAN)[23]和文献[24]中提出的算法作为实验对比, 并将结果与DDCA/TSD 的结果进行对比分析. 本文首先采用上述聚类算法对经典的小规模数据集Seeds和规模较大的数据集Mushroom和Anuran Calls(MFCCs)进行实验分析, 其次对SEQUOIA 2000数据库中的数据规模从小到大进行时间效率的实验.

上述实验的3种算法的实验过程如下: 对于每一个时间片, 3种算法分别进行独立的静态聚类并输出结果和相关指标. 与之不同, DDCA/TSD算法的实验中, t_1时间片是DBSCAN算法进行完全聚类的结果, 其后的4个时间片则由DDCA/TSD算法进行动态聚类并输出相关指标. 在t_1时间片DDCA/TSD算法需要DBSCAN形成初始聚类结果, 其后的4个时间片则由DDCA/TSD算法基于初始时刻结果进行各个时间片上的动态聚类, 所以t_1时间片DDCA/TSD算法没有进行聚类.

针对经典的小规模数据集Seeds, 上述3种算法在5个时间片上的实验结果分别如表4所示. 针对较大规模数据集Mushroom和和Anuran Calls (MFCCs), 实验结果分别如表5和表6所示.

表 4　在5个时间片上Seeds数据集的各种算法指标对比

Table 4　The comparison of various algorithm for Seeds data sets on 5 time slices

时间片	算法	ACC	PR	RE	ARI	NUM	T
t_1	DBSCAN	0.867	0.869	0.846	0.832	210	21.8
	AF–DBSCAN	0.886	0.902	0.894	0.880	210	**8.0**
	文献[24]	**0.905**	0.916	0.911	0.898	210	8.6
t_2	DBSCAN	**0.871**	**0.875**	**0.869**	**0.858**	210	21.5
	AF–DBSCAN	0.881	0.896	0.889	0.880	210	7.8
	文献[24]	**0.920**	0.935	0.927	0.906	210	8.7
	DDCA/TSD	**0.871**	**0.875**	**0.869**	**0.858**	**68**	**7.2**
t_3	DBSCAN	**0.862**	**0.871**	**0.859**	**0.875**	210	21.6
	AF–DBSCAN	0.895	0.907	0.899	0.881	210	7.9
	文献[24]	**0.910**	0.911	0.906	0.897	210	8.8
	DDCA/TSD	**0.862**	**0.871**	**0.859**	**0.875**	**66**	**7.1**
t_4	DBSCAN	**0.881**	**0.894**	**0.887**	**0.878**	210	21.7
	AF–DBSCAN	0.890	0.906	0.894	0.879	210	8.0
	文献[24]	**0.900**	0.913	0.905	0.887	210	8.6
	DDCA/TSD	**0.881**	**0.894**	**0.887**	**0.878**	**71**	**7.4**
t_5	DBSCAN	**0.886**	**0.902**	**0.894**	**0.880**	210	21.6
	AF–DBSCAN	0.895	0.907	0.899	0.881	210	7.9
	文献[24]	**0.924**	0.933	0.925	0.914	210	8.5
	DDCA/TSD	**0.886**	**0.902**	**0.894**	**0.880**	**69**	**7.3**

表 5 在5个时间片上Mushroom数据集的各种算法指标对比

Table 5 The comparison of various algorithm for Mushroom data sets on 5 time slices

时间片	算法	ACC	PR	RE	ARI	NUM	T
t_1	DBSCAN	0.923	0.933	0.926	0.914	8124	5729.7
	AF–DBSCAN	0.935	0.941	0.938	0.928	8124	**3264.2**
	文献[24]	**0.948**	0.960	0.950	0.943	8124	3746.3
t_2	DBSCAN	**0.922**	**0.930**	**0.925**	**0.913**	8124	5734.6
	AF–DBSCAN	0.934	0.939	0.937	0.927	8124	3266.8
	文献[24]	**0.949**	0.966	0.952	0.944	8124	3745.8
	DDCA/TSD	**0.922**	**0.930**	**0.925**	**0.913**	**2451**	**1696.4**
t_3	DBSCAN	**0.923**	**0.934**	**0.926**	**0.914**	8124	5740.0
	AF–DBSCAN	0.935	0.942	0.938	0.928	8124	3268.5
	文献[24]	**0.949**	0.965	0.952	0.944	8124	3746.2
	DDCA/TSD	**0.923**	**0.934**	**0.926**	**0.914**	**2513**	**1700.3**
t_4	DBSCAN	**0.922**	**0.929**	**0.925**	**0.913**	8124	5738.1
	AF–DBSCAN	0.936	0.944	0.940	0.929	8124	3265.3
	文献[24]	**0.948**	0.964	0.950	0.943	8124	3747.6
	DDCA/TSD	**0.922**	**0.929**	**0.925**	**0.913**	**2439**	**1695.7**
t_5	DBSCAN	**0.923**	**0.933**	**0.925**	**0.913**	8124	5471.5
	AF–DBSCAN	0.935	0.942	0.938	0.927	8124	3263.9
	文献[24]	**0.949**	0.964	0.952	0.943	8124	3745.2
	DDCA/TSD	**0.923**	**0.933**	**0.925**	**0.913**	**2427**	**1694.8**

表 6 在5个时间片上Anuran Calls(MFCCs)数据集的各种算法指标对比

Table 6 The comparison of various algorithm for Anuran Calls (MFCCs) data sets on 5 time slices

时间片	算法	ACC	PR	RE	ARI	NUM	T
t_1	DBSCAN	0.941	0.952	0.945	0.937	7195	5578.6
	AF–DBSCAN	0.953	0.965	0.957	0.946	7195	**3178.4**
	文献[24]	**0.968**	0.978	0.970	0.964	7195	3155.1
t_2	DBSCAN	**0.940**	**0.949**	**0.944**	**0.936**	7195	5584.1
	AF–DBSCAN	0.954	0.968	0.958	0.947	7195	3175.6
	文献[24]	**0.968**	0.980	0.971	0.966	7195	3154.9
	DDCA/TSD	**0.940**	**0.949**	**0.944**	**0.936**	**2113**	**1523.4**
t_3	DBSCAN	**0.941**	**0.953**	**0.945**	**0.938**	7195	5571.9
	AF–DBSCAN	0.953	0.966	0.957	0.946	7195	3177.0
	文献[24]	**0.969**	0.983	0.972	0.968	7195	3156.2
	DDCA/TSD	**0.941**	**0.953**	**0.945**	**0.938**	**2254**	**1531.6**
t_4	DBSCAN	**0.941**	**0.952**	**0.945**	**0.938**	7195	5575.8
	AF–DBSCAN	0.953	0.965	0.957	0.945	7195	3180.3
	文献[24]	**0.968**	0.979	0.971	0.967	7195	3157.9
	DDCA/TSD	**0.941**	**0.952**	**0.945**	**0.938**	**2135**	**1525.7**
t_5	DBSCAN	**0.940**	**0.950**	**0.944**	**0.937**	7195	5574.5
	AF–DBSCAN	0.953	0.965	0.957	0.946	7195	3179.6
	文献[24]	**0.969**	0.982	0.972	0.968	7195	3156.4
	DDCA/TSD	**0.940**	**0.950**	**0.944**	**0.937**	**2179**	**1527.9**

首先对比DBSCAN, AF–DBSCAN和文献[24]算法为代表的3类算法在多个时间片上进行连续完全聚类的性能差异. 表4–6中指标ACC最高的是文献[24]的算法, 比DBSCAN平均高出3.1%, 比AF–DBSCAN平均高出1.9%, 说明文献[24]的算法聚类正确比例高. 指标PR最高的也是文献[24]的算法, 比DBSCAN平均高出3.3%, 比AF–DBSCAN平均高出2.0%, 说明文献[24]的算法正的样本数是数目比较多. 指标RE表示样本中的正例有多少被预测正确, 3种算法中文献[24]的算法最高, 比DBSCAN平均高出3.9%, 比AF–DBSCAN平均高出2.1%, 说明文献[24]的算法聚类的结果和真实结果最类似. 指标ARI用来衡量两个数据分布的吻合程度, 3种算法中文献[24]的算法最高, 比DBSCAN平均高出3.3%, 比AF–DBSCAN平均高出2.0%, 说明文献[24]的算法聚类结果与真实结果最接近. 但是, 3个数据集中指标T最小的是AF–DBSCAN算法, 说明完全聚类中AF–DBSCAN算法的时间效率最好. 上述结果显示, 在多个时间片上的连续完全聚类中, 文献[24]算法的聚类结果最好, AF–DBSCAN算法和DBSCAN算法与文献[24]算法差别不大. AF–DBSCAN算法聚类时间效率最高.

其次对比采用时间序列上连续动态聚类的DDCA/TSD算法与DBSCAN算法, AF–DBSCAN算法和文献[24]算法的性能差异. 可发现实验数据中DDCA/TSD算法和DBSCAN算法的ACC, PR, RE, ARI 4种指标是一样的. 说明了DDCA/TSD算法和DBSCAN算法聚类结果相同且聚类效果一致. 并且DBSCAN, AF–DBSCAN和文献[24]的算法在每个时间片上参加本次聚类的数目NUM是整个数据集数目, 而DDCA/TSD 算法参加数目NUM每次都远小于整个数据集数目. 这说明DDCA/TSD算法是基于前一个时间片的基础进行局部类结构调整. 它仅进行了小规模的局部计算, 却达到全局聚类的相同效果. 更重要的是, DDCA/TSD算法的指标T是最小的, 尤其在较大规模的数据集Mushroom和MFCC上, 比静态聚类中时间效率最好的AF–DBSCAN算法提高接近2倍, 这说明DDCA/TSD算法的运算模式能够显著提高时间序列上动态聚类过程的计算效率. 因此, DDCA/TSD算法的聚类效果和文献[24]的算法差别不大, 但是, DDCA/TSD算法采取局部聚类代替全局聚类的动态聚类思想在时间效率上有很高程度的提升.

为进一步说明DDCA/TSD算法相对DBSCAN算法, AF–DBSCAN算法和文献[24]算法的时间效率有很高程度的提升, 实验采用SEQUO IA 2000数据库进行进一步实验. 数据集采用的数据个数分别是200, 500, 1000, 2000, 5000, 10000, 15000. 与前一实验类似, 在数据集范围内依然随机消失、改变及位移数据集中30%的元素, 故改变的元素数量依次是60, 150, 300, 600, 1500, 3000, 4500. 实验采用指标T来统计上述4种算法的时间效率, T是SEQUO IA 2000数据集在5个时间片上的平均值, 图7是实验结果.

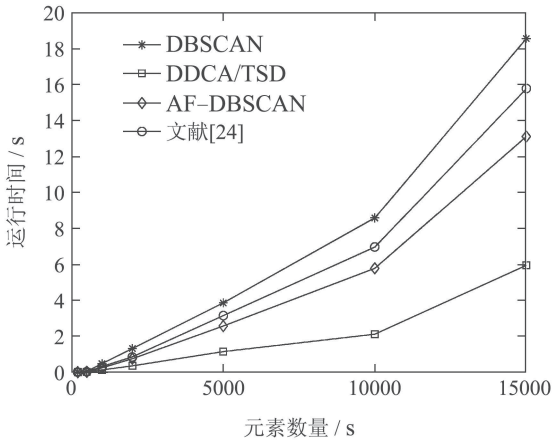


图 7　SEQUO IA 2000数据库测试时间结果

Fig. 7　The results for SEQUO IA 2000 database

由图7可知, 4种算法的运行时间都随着数据的增大而增大, 并且当元素总数增加较多时, 算法运行的时间增幅会比较大. 但DDCA/TSD算法的增幅要明显小于DBSCAN算法, AF–DBSCAN算法和文献[24]算法的增幅. 再者, DDCA/TSD算法在不同数量的数据集上的运行时间都约为DBSCAN算法的33%, 约为AF–DBSCAN算法的51%, 约为文献[24]算法的42%. DDCA/TSD算法通过动态的连续局部聚类调整能够显著降低聚类过程的计算开销, 尤其是数据集中元素个数多于5000时, DDCA/TSD算法的时间效率提升更为明显. 综上所述, DDCA/TSD算法非常适合处理时间序列上大规模数据的聚类问题.

6　结论

本文对时间序列数据动态变化过程进行了分析, 发现相邻时间片间具有数据的关联性和类结构的继承性, 对几类典型聚类算法的分析说明密度聚类更有利于发现数据点的属性变化及类簇结构的演变过程. 在此基础上提出了DDCA/TSD算法. 实验结果显示此方法对时间序列数据能以更高的效率揭示数据结构的变化结果及过程, 并更适合处理时间序列上的大规模数据聚类问题. 在下面的工作中, 如何将其应用于实际问题的解决是下一步研究的主要工作.

参考文献:

[1] FU T. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 2011, 24(1): 164 – 181.

[2] LI Hailin, GUO Chonghui. Survey of feature representations and similarity measurements in time series data mining. *Application Research of Computers*, 2013, 30(5): 1285 – 1291.
(李海林, 郭崇慧. 时间序列数据挖掘中特征表示与相似性度量研究综述. 计算机应用研究, 2013, 30(5): 1285 – 1291.)

[3] XIE Fuding, ZHAO Xiaohui, JI Min, et al. A time series dynamic clustering algorithm. *Application Research of Computers*, 2012, 29(10): 3677 – 3680.
(谢福鼎, 赵晓慧, 嵇敏, 等. 一种时间序列动态聚类的算法. 计算机应用研究, 2012, 29(10): 3677 – 3680.)

[4] HE Xiaoxu. *Research on key issues in time series data mining*. Hefei: University of Science and Technology of China, 2014.
(何晓旭. 时间序列数据挖掘若干关键问题研究. 合肥: 中国科学技术大学, 2014.)

[5] SHAN Zhongnan, WENG Xiaoqing, MA Chaohong. Review of time series semi-supervised classification. *Journal of the Hebei Academy of Sciences*, 2018, 35(2), 49 – 54.
(单中南, 翁小清, 马超红. 时间序列半监督分类综述. 河北省科学院学报, 2018, 35(2): 49 – 54.)

[6] OLIVEIRA J V D, PEDRYCZ W. *Advances in Fuzzy Clustering and Its Applications*. Hoboken: John Wiley & Sons, 2007:1 – 454.

[7] MACQUEEN J. Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley: University of California Press, 1967: 281 – 297.

[8] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks. *Science*, 2014, 344(6191): 1492 – 1496.

[9] ZHANG T, RAMAKRISHNAN R, LINVY M. BIRCH: an efficient data clustering method for very large databases. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, New York: Newsletter, 1996, 25(2): 103 – 114.

[10] ESTER M, KRIEGEL H P, XU X, et al. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon: AAAI Press, 1996: 226 – 231.

[11] QIN Jiarui, XU Weihong, MA Honghua, et al. Self-adaptive local eps DBSCAN. *Journal of Chinese Computer Systems*, 2018, 39(10): 2186 – 2190.
(秦佳睿, 徐蔚鸿, 马红华, 等. 自适应局部半径的DBSCAN聚类算法. 小型微型计算机系统, 2018, 39(10): 2186 – 2190.)

[12] ZHANG Yang, HE Li, ZHU Haodong. An improved k–means dynamic clustering algorithm. *Journal of Chongqing Normal University(Natural Science)*, 2016, 33(1): 97 – 101.
(张阳, 何丽, 朱颢东. 一种改进的k–means动态聚类算法. 重庆师范大学学报(自然科学版), 2016, 33(1): 97 – 101.)

[13] AGRAWAL R, FALOUTSOS C, SWAMI A. Efficient similarity search in sequence databases. *Proceeding of the 4th International Conference on Foundations of Data Organization and Algorithms*. Washington DC: IEEE Computer Society, 1993: 69 – 84.

[14] BENÍTEZ I, DÍEZ J L, QUIJANO A, et al. Dynamic clustering of residential electricity consumption time series data based on Hausdorff distance. *Electric Power Systems Research*, 2016, 140(6): 517 – 526.

[15] WANG Ling, MENG Jianyao, XU Peipei, et al. Similarity dynamical clustering algorithm based on multidimensional shape features for time series. *Chinese Journal of Engineering*, 2017, 39(7): 1114 – 1122.
(王玲, 孟建瑶, 徐培培, 等. 基于多维时间序列形态特征的相似性动态聚类算法. 工程科学学报, 2017, 39(7): 1114 – 1122.)

[16] LUCZAK M. Hierarchical clustering of time series data with parametric derivative dynamic time warping. *Expert Systems with Applications*, 2016, 62(12): 116 – 130.

[17] GENOLINI C, FALISSARD B. KmL: k–means for longitudinal data. *Computational Statistics*, 2016, 25(2): 317 – 328.

[18] IZAKIAN H, PEDRYCZ W. Agreement-based fuzzy C – means for clustering data with blocks of features. *Neurocomputing*, 2014, 127(127): 266 – 280.

[19] SUN Ya, LI Zhihua. Clustering algorithm for time series based on locally extreme point. *Computer Engineering*, 2015, 41(5): 33 – 37.
(孙雅, 李志华. 基于区域极值点的时间序列聚类算法. 计算机工程, 2015, 41(5): 33 – 37.)

[20] LIU Qin, WANG Kaile, RAO Weixiong. Non-equal time series clustering algorithm with sliding window STS distance. *Journal of Frontiers of Computer Science and Technology*, 2015, 9(11): 1301 – 1313.
(刘琴, 王恺乐, 饶卫雄. 不等长时间序列滑窗STS距离聚类算法. 计算机科学与探索, 2015, 9(11): 1301 – 1313.)

[21] SHUKRI S, FARIS H, ALJARAH I, et al. Evolutionary static and dynamic clustering algorithms based on multi-verse optimizer. *Engineering Applications of Artificial Intelligence*, 2018, 72(3): 54 – 66.

[22] BACHE K, LICHMAN M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

[23] ZHOU Zhiping, WANG Jiefeng, ZHU Shuwei et al. An improved adaptive and fast AF – DBSCAN clustering algorithm. *CAAI Transactions on Intelligent Systems*, 2016,11(1): 93 – 98.
(周治平, 王杰锋, 朱书伟, 等. 一种改进的自适应快速AF – DBSCAN聚类算法. 智能系统学报, 2016, 11(1): 93 – 98.)

[24] HAN Lizhao, QIAN Xuezhong, LUO Jing et al. Multi-density clustering algorithm DBSCAN based on region division. *Application Research of Computers*, 2018, 35(6): 1668 – 1671, 1685.
(韩利钊, 钱雪忠, 罗靖, 等. 基于区域划分的DBSCAN多密度聚类算法. 计算机应用研究, 2018, 35(6): 1668 – 1671, 1685.)

作者简介:

陈 皓 博士, 副教授, 硕士研究生导师, 主要研究领域为工程优化与数据挖掘, E-mail: chenhao@xupt.edu.cn;

冀敏杰 硕士研究生, 研究领域为数据挖掘和机器学习, E-mail: 873453014@qq.com;

郭紫园 硕士研究生, 研究领域为数据挖掘和进化计算, E-mail: 1159382619@qq.com;

夏 雨 硕士研究生, 研究领域为数据挖掘与医疗大数据分析, E-mail: xiayu32110@163.com.