基于密度与近邻传播的数据流聚类算法

张建朋 [1] 陈福才 [1] 李邵梅 [1] 刘力雄 [1]

摘 要 针对现有算法聚类精度不高、处理离群点能力较差以及不能实时检测数据流变化的缺陷, 提出一种基于密度与近邻传播融合的数据流聚类算法. 该算法采用在线/离线两阶段处理框架, 通过引入微簇衰减密度来精确反映数据流的演化信息, 并采用在线动态维护和删减微簇机制, 使算法模型更符合原始数据流的内在特性. 同时, 当模型中检测到新的类模式出现时, 采用一种改进的加权近邻传播聚类 (Weighted and hierarchical affinity propagation, WAP) 算法对模型进行重建, 因而能够实时检测到数据流的变化, 并能给出任意时间的聚类结果. 在真实数据集和人工数据集上的实验表明, 该算法具有良好的适用性、有效性和可扩展性, 能够取得较好的聚类效果.

关键词 数据流挖掘, 近邻传播, 基于密度聚类, 变化检测

引用格式 张建朋, 陈福才, 李邵梅, 刘力雄. 基于密度与近邻传播的数据流聚类算法. 自动化学报, 2014, **40**(2): 277−288

DOI 10.3724/SP.J.1004.2014.00277

Data Stream Clustering Algorithm Based on Density and Affinity Propagation Techniques

ZHANG Jian-Peng[1] CHEN Fu-Cai[1] LI Shao-Mei[1] LIU Li-Xiong[1]

Abstract For the accuracy of the existing clustering algorithm is not high, and the ability of dealing with outliers is poor and unable to detect the real-time changes of data stream, a data stream clustering algorithm based on density and affinity propagation is proposed. The algorithm adopts an online/offline two-stage processing framework and it introduces the micro-cluster decay density to reflect the evolution of the data stream accurately. In the meantime, it uses the mechanism of online dynamic maintenance and deletion of the micro-cluster, which makes the algorithm′s model more consistent with the intrinsic characteristics of the original data streams. Simultaneously, it also takes an improved WAP (weighted and hierarchical affinity propagation) algorithm to reconstruct the models when detecting a new emerging class model. Thus it can detect the changes of the data stream in real time, and give the clustering results at any time. Experiments on real data sets and artificial data sets show that the algorithm has good applicability, efficiency, and scalability, thus it can achieve better clustering results.

Key words Data stream mining, affinity propagation, density-based clustering, change detection method

Citation Zhang Jian-Peng, Chen Fu-Cai, Li Shao-Mei, Liu Li-Xiong. Data stream clustering algorithm based on density and affinity propagation techniques. *Acta Automatica Sinica*, 2014, **40**(2): 277−288

随着信息技术的迅猛发展和广泛应用, 数据流挖掘技术在国内外得到广泛研究. 它在网络监控、证券交易分析、电信记录分析等方面有着巨大的应用前景. 针对数据流具有快速连续到达、持续增长、动态演化等特点, 传统聚类算法无法适用于数据流对象, 数据流对聚类算法提出了如下新的要求[1−3]: 1) 对自然簇个数无须假设; 2) 发现任意形状的簇; 3) 处理离群点的能力; 4) 实时检测数据流变化. 因此数据流的聚类问题得到了学术界和工业界的广泛关注, 如何有效地分析和处理这些数据, 发现其中隐藏的知识显得尤为重要.

近年来数据流环境下的聚类研究工作广泛展开, 文献 [4] 提出了进化数据流聚类框架 CluStream (High-dimensional projected data stream clustering), 它首次采用了两阶段处理框架: 在线微聚类和离线宏聚类两个部分. 在线阶段算法负责在线统计数据流特征向量, 生成概要信息; 离线阶段算法负责相应用户请求, 根据概要数据生成最终较为精确的聚类结果. 该算法灵活的扩展性是其受到广泛关注的主要原因. CluStream 同样存在一些缺点: 首先更适合对超球形分布数据进行聚类; 其次对于含有大量噪声的原始数据适应性较差; 最后需要预先指定类簇数量, 极大地影响了原始数据聚簇的形态分布. 针对以上问题, Aggarwal 等进一步提出了 HPStream 算法等[5], 采用高维投影和衰减函数来

收稿日期 2013-01-16 录用日期 2013-05-03
Manuscript received January 16, 2013; accepted May 3, 2013
国家高技术研究发展计划 (863 计划) (2011AA010603, 2011AA010605) 资助
Supported by National High Technology Research and Development Program of China (863 Program) (2011AA010603, 2011AA010605)
本文责任编委 封举富
Recommended by Associate Editor FENG Ju-Fu
1. 国家数字交换系统工程技术研究中心 郑州 450002
1. National Digital Switching System Engineering and Technological Research and Development Center, Zhengzhou 450002

更好地支持高维数据流的聚类分析, 但是仍然需要用户预先指定平均聚类维数; 文献 [6] 提出的 DenStream 算法, 在 CluStream 双层架构上引入了潜在核心微簇和孤立点微簇等概念, 能够支持任意形状的聚类. 但由于采用全局一致的绝对密度作参数, 使聚类结果对参数值的选取非常敏感; 文献 [7] 提出的 D-Stream, 将输入的数据映射到相应的网格, 算法引入了带衰减因子的密度计算法, 能够实时调整数据流的进化特征, 实现高效网格密度聚类, 但网格方法丢失了空间位置信息, 处理边缘能力较差, 聚类效果受网格粒度影响较大; 文献 [8] 在 D-Stream 框架上引入网格之间的吸引力的概念, 有效解决了空间位置信息丢失问题, 同时算法复杂度大大增加, 使得算法运行效率急剧下降.

在上述经典算法的指引下提出了很多新颖的方法. 文献 [9] 针对倾斜分布数据聚类效果不佳的问题, 提出了局部密度阈值随时间自适应变化的目标关联覆盖算法 (Target associate coverage algorithm, TACA), 但仍无法处理任意形状分布数据; 文献 [10] 和文献 [11] 分别使用子空间密度聚类 (High-dimensional data stream, HDDStream) 和局部更新密度聚类 (On-line density-based clustering algorithm for spatial data stream, OLDStream) 进行数据流聚类, 有效解决了非球形数据聚类问题, 但仍无法反映数据流的演化信息; Tang[12] 针对微簇交叠情况下改进 DenStream 算法提出了 DenStream2, 引入重叠因子的概念, 有效解决了交叠微簇的归属问题, 并在任意形状分布数据取得了良好的聚类效果; Zhang 等[13] 在顶级会议 PKDD (Principles and Practice of Knowledge Discovery in Databases) 首次提出了 StrAP (Stream data streaming with affinity propagation) 算法, 此算法将新到达的数据点与当前模型匹配, 匹配成功则动态更新微簇, 否则将可能是噪声点的数据点放入暂存盒中, 同时当检测到数据分布的变化时, 则重新构建聚类模型以适应数据分布的变化, 此算法能快速检测数据流变化并能给出任意时间的聚类结果, 但是聚类质量依赖于设定滑动窗口的大小, 并且没有考虑数据到达时间对聚类结果的影响.

针对现有聚类算法聚类质量不高、处理离群点能力较差以及不能有效检测数据流变化的缺陷, 本文在 StrAP 算法的基础上提出一种密度与近邻传播融合的高效数据流聚类算法 StrDenAP (Data stream base on density and affinity propagation clustering), 该算法借鉴 CluStream 算法的在线/离线双层框架, 在线聚类层持续将新到达的数据点映射到相应的微簇区域, 进而更新微簇概要结构信息, 并以微簇特征向量的形式记录这些信息, 并引入微簇衰减密度区分 "陈旧" 微簇和 "最新" 微簇, 来体现数据流的演化信息. 为了减少内存消耗, StrDenAP 算法提出在线动态删除机制对微簇进行维护, 使算法模型更加准确. 同时, 当检测到有新的类模式出现时, 本文通过改进的加权近邻传播聚类 (Weighted and hierarchical affinity propayation, WAP) 算法对聚类模型重建, 能够实时检测到数据流的变化, 后端离线层则是利用存储的历史数据的快照信息生成特定时间内更为精确的聚类结果或对数据流进行演化分析. 因而算法既能给出任意时间的聚类结果, 又能对历史数据进行演化分析. 实验结果表明, StrDenAP 算法能够取得较高的聚类质量, 并能给出任意时间的聚类结果, 能够满足实时监控系统的应用需求.

1 基本的近邻传播算法及相关扩展

1.1 近邻传播算法 (Affinity propagation, AP)

近邻传播聚类算法是一种基于近邻信息传播的聚类算法[14], 该算法的目的是找到最优的类代表点集合使得所有数据点到最近的类代表点的相似度之和最大. 该算法的简要流程如下: 算法的输入是所有 N 个数据点两两之间的相似度组成的相似性矩阵 $S_{N\times N}$, 算法起始阶段将所有的样本都看作是潜在的聚类中心. 同时, 为每个样本点建立与其他样本点的吸引程度的信息定义如下:

吸引度 (Responsibility): $r(i,k)$ 用来描述点 k 适合作为数据点 i 的聚类中心的程度;

归属度 (Availability): $a(i,k)$ 用来描述点 i 选择点 k 作为其聚类中心的适合程度.

为了找到合适的聚类中心 x_k, AP 算法不断地从数据样本中搜集证据 $r(i,k)$ 和 $a(i,k)$. $r(i,k)$ 和 $a(i,k)$ 的迭代公式为

$$r(i,k) \leftarrow s(i,k) - \max_{k'\neq k}\left[a(i,k') + s(i,k')\right] \tag{1}$$

$$a(i,k) \leftarrow \begin{cases} \min\{0, r(k,k) + \\ \quad \sum\limits_{\substack{i'\neq i,\\ i'\neq k}} \max[0, r(i',k)]\}, & i \neq k \\ \sum\limits_{i'\neq k}^{n} \max[0, r(i',k)], & i = k \end{cases} \tag{2}$$

AP 算法中主要根据式 (1) 和式 (2) 不断循环迭代从而更新证据. 迭代更新的快慢可以通过调节阻尼系数 λ 实现. $r(i,k)$ 与 $a(i,k)$ 越强, 则 k 点作为聚类中心的可能性就越大, 并且 i 点隶属于以 k 点为聚类中心的聚类的可能性也越大. AP 算法通过迭代过程不断更新每一个点的吸引度和归属度值, 直到产生 m 个高质量的聚类代表点, 同时将其余的

数据点分配到相应的聚类中.

1.2 加权近邻传播聚类 (WAP)

WAP 算法原理与 AP 算法极为相似, 是 AP 算法在大规模数据集上的扩展[13]. WAP 算法的目标是选取最优的类代表点集合, 描述为寻找一个目标函数 $E'[c]$, 使得 $E'[c] = \sum_{i=1}^{N} S'(i, c_i) - \sum_{i=1}^{N} \sigma_i(c)$ 取最大值. 解决该问题的过程与 AP 算法一致, 在不改变所传递的信息量的前提下, 对临近点进行整合, 使得整合过的点的聚类结果与原始点的聚类结果相同. 不同之处仅在于 WAP 使用新定义的相似度, 数据点间的相似度和偏好度被重新定义为

$$S'(i,k) = \begin{cases} n_i S(i,k), & i \neq k \\ p(i) + (n_i - 1)\varepsilon_i, & i = k \end{cases} \tag{3}$$

其中, n_i 为权重, 取以 i 作为其类代表点的数据个数, 是以 i 为类代表点的类内平均相似度. 从 WAP 算法的计算流程知, 每个聚类代表点的个数被考虑, 影响该点传递信息的量. 因此在这些权值点上用 WAP 聚类, 就等价于用 AP 对所有数据点的聚类. WAP 算法的最坏时间复杂度为 $\mathrm{O}(N^{\frac{3}{2}})$, 当数据规模 N 很大时, 时间复杂度相对于 AP 算法 ($\mathrm{O}(N^2)$) 明显降低, 因此, WAP 算法更适合解决大规模的数据流聚类问题.

2 StrDenAP 算法框架及相关概念

2.1 算法框架

StrDenAP 算法主要分为初始聚类、在线聚类和离线处理 3 个过程. 初始聚类利用最初数据初始化整个在线处理过程; 在线聚类则动态更新微簇衰减密度并实时检测新到达数据流分布的变化, 给出在线的聚类结果; 离线过程由用户调用, 并依据 Pyramid time 框架[4−5,7] 选取特定时刻的中间结果, 针对用户的查询给出最终挖掘结果. 通过在线/离线两个协调运作实现动态、快速地处理流式数据, 很好地满足用户对数据流分析的需求, 挖掘模型如图 1 所示.

2.2 相关概念及拓展

定义 1 (衰减函数). 随着数据流的进化, 数据流中的每个样本点的权重是随时间衰减的[6−10], 假设样本点 X 到达时间为 t, 初始权重为 1, 本文定义一个衰变函数: $f(t) = 2^{-\lambda t} \lambda \in (0,1)$ 作为权值来表示历史数据的衰减程度, λ 是衰减因子, λ 越大, 历史数据的衰减程度就越高.

定义 2 (元组衰减密度). 对数据流中在 t_0 时刻到达的数据 X, 则 t 时刻该元组的衰减密度 $N_x(x,t)$ 定义为: $N_x(x,t) = 2^{-\lambda(t-t_0)}$, $\lambda \in (0,1)$.

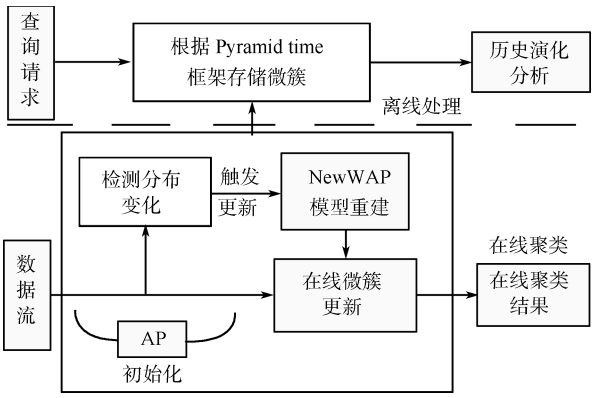


图 1　StrDenAP 数据流聚类模型

Fig. 1　Data stream clustering mode of StrDenAP

定义 3 (微簇衰减密度). 令 t 时刻微簇 mc 所包含的 n 个元组集合为 $M(\mathrm{mc}, t)$, 则微簇 mc 的衰减密度定义为[7]: $N_{\mathrm{mc}}(t) = \sum_{i=1}^{n} N_x(x_i, t) = \sum_{i=1}^{n} 2^{-\lambda(t-t_{0i})}$.

定理 1. 所有微簇 mc 的衰减密度之和 $N_{\mathrm{mc}}(t)$ 最大值不会超过 $\dfrac{1}{1-2^{-\lambda}}$, 任一微簇密度平均值不会超过 $\dfrac{1}{N(1-2^{-\lambda})}$, N 为当前微簇的数目.

证明. 设 $X(t)$ 中从开始时刻到 t 时刻所到达的所有数据项为: 一组带有时间标签 $T_{il}, \cdots, T_{in}$ 的 d 维数据点集 $X_{il}, \cdots, X_{in}$, 则:

$$N_{\mathrm{mc}}(t) = \sum_{t'=0}^{t} 2^{-\lambda(t-t')} = \frac{1-2^{-\lambda(t+1)}}{1-2^{-\lambda}}$$

因此, 微簇密度的最大值为

$$N_{\mathrm{mc}}\max = \lim_{t\to\infty} N_{\mathrm{mc}} = \lim_{t\to\infty} \frac{1-2^{-\lambda(t+1)}}{1-2^{-\lambda}} = \frac{1}{1-2^{-\lambda}}$$

□

由定理 1 可知, 在任何时刻所有微簇 mc 的衰减密度之和不会超过 $\dfrac{1}{1-2^{-\lambda}}$, 设 N 为微簇的数目, 则任一微簇的平均密度不会超过 $\dfrac{1}{N(1-2^{-\lambda})}$.

定理 2. 设微簇 mc 在时刻 t_n 接收了一个新数据, 该微簇上一次接收数据的时间为 t_l ($t_l < t_n$), 则此微簇的密度可以由下面公式进行更新:

$$N_{\mathrm{mc}}(\mathrm{mc}, t_n) = 2^{-\lambda(t_n - t_l)} N(\mathrm{mc}, t_l) + 1 \tag{4}$$

它使微簇不必每时每刻对其密度值进行更新, 仅在吸收新数据时更新其密度值, 也不必过多地存储单

元中数据到达时刻的相关信息, 很大程度上增加了算法的执行效率并减少了所需存储空间.

证明. 设在时刻 t_l 微簇 mc 中的数据有 $X_{il},\cdots,X_{in}$, 则 $N_{\rm mc}(t_l)=\sum_{i=1}^{n}N_x(x_i,t_l)$. 根据

$$N_x(x_i,t_n)=2^{-\lambda(t_n-t_l)}N_x(x_i,t_l)+1, i=1,2,\cdots,n$$

则

$$N_{\rm mc}({\rm mc},t_n)=\sum_{i=1}^{n}N_x(x_i,t_n)+1=\sum_{i=1}^{n}2^{-\lambda(t_n-t_l)}N_x(x_i,t_l)+1$$

□

定义 4 (密集微簇与稀疏微簇). 在 t 时刻, 假设微簇 mc 中存在一组带有时间标签 $T_{il},\cdots,T_{in}$ 的 d 维数据点集 $X_{il},\cdots,X_{in}$, $N_{\rm mc}(t)$ 为微簇在 t 时刻密度; 对于给定的阈值 $(\mu>1)$, 若微簇密度满足

$$N_{\rm mc}(t)\geq\frac{\mu}{N(1-2^{-\lambda})}=N_{\rm thred}$$

则该微簇为密集微簇; 否则, 为稀疏微簇.

定义 5 (微簇聚类特征表示). 由于数据流无限、快速到达, 难以将数据维护在有限内存中进行精确聚类, 因此, 引入一种概要表示 —— 微簇 (mc) 来提供近似聚类结果, 此时可由六元组聚类特征 $CF\{E_i,N_{{\rm mc}i},D_i,R_i,T_0,T_n\}$ 来定义, 其中, E_i 为微簇的聚类代表点的坐标向量; $N_{{\rm mc}i}$ 为微簇衰减密度最近一次更新时的密度值; D_i 为聚类代表点 e_i 到所属元组 x_i 的加权距离和, $D_i(t)=\sum_{i=1}^{n}N_x(x_i,t)dist^2(e_i,x_i)$; R_i 为微簇的平均半径, $R_i=D_i/N_{{\rm mc}i}$; T_0 为微簇创建时间; T_n 为微簇最后更新时间.

3 StrDenAP 算法的详细描述

3.1 初始聚类

我们对数据流中最初的 $N_{\rm init}$ 个数据点集 P 运用 AP 算法, 以初始化整个在线处理过程. 通过扫描数据集 P 来获取最初的一组聚类微簇. 对集合 P 中的每一个点 p, 若其微簇内数据点的总密度大于密集微簇的阈值 $D_{\rm thred}$, 则依据点 p 及其邻居数据点创建一个密集微簇, 以所有密集微簇的平均半径作为最初的 ζ.

3.2 在线微簇维护

由于进化数据流的动态特性, 新簇不断出现, 而旧簇不断退化消亡. 新簇需要长时间积累才能发展成密集微簇, 而大部分新样本点属于现存的稀疏微簇. 当有新数据点到来时, 将其优先放入离其最近的密集微簇或稀疏微簇中; 如果无法被当前模型吸收, 再将其放到暂存盒 Reservoir 缓存中; 当暂存盒中检测到出现新的类模式, 重新聚类将暂存盒里的数据点加入到模型中去. 执行操作见算法 1.

算法 1. Process(x_t)

新到达数据点 x_t;
  For (每个密集微簇 rc)
    If ($d(x_t,{\rm rc}_i)\leq\zeta$)
      把 x_t 加入到离它最近的密集微簇 rc 中;
  For (每个稀疏微簇 oc)
    If ($d(x_t,{\rm oc}_i)\leq\zeta$)
      把 x_t 加入到离它最近的稀疏微簇 oc 中;
      If ($N_{\rm mc}(t)\geq N_{\rm thred}$)
      此 oc 簇已发展成密集微簇, 把 oc 簇从稀疏微簇中;
      移出, 加入到密集微簇 rc 中;
    Else
      点 x_t 不能被现有的微簇所吸收,
      将点 x_t 放入暂存盒缓存中,
      说明点 x_t 可能是离群点噪声也可能是新微簇的种子.
  End

随着时间的推移, 密集微簇和稀疏微簇数量将会越来越多而消耗大量内存容量. 为了缓解内存容量问题, 本文提出微簇维护机制.

3.2.1 密集微簇的蜕化检测

对于现存的密集微簇 rc, 随着时间的滑动, rc 的密度将不断地衰减, 即如果长时间内微簇没有吸收新的样本点, 则其密度将不断衰减, 一旦权值低于指定阈值 $N_{\rm thred}$, 这就意味着 rc 已经退化为离群点噪声. 我们将其删除并释放内存空间, 以供新的密集微簇使用. 因此, 要定期检查密集微簇的衰减密度值, 以便有效地利用内存. 对于任意密集微簇退化成稀疏微簇的最短时间跨度为

$$T_p=\left[\frac{1}{\lambda}\log\left(\frac{N_{\rm thred}}{N_{\rm thred}-1}\right)\right]\tag{5}$$

由等式 $2^{-\lambda T_p}N_{\rm thred}+1=N_{\rm thred}$, 我们每隔 T_p 对每个簇进行权值检查, 判断是否将其删除.

3.2.2 稀疏微簇的删除

随着数据流不断到达, 稀疏微簇的数量迅速增加, 增大系统的负载. 理想的删除策略应考虑两种情况. 一种是陈旧的微簇, 表明该聚类代表的模式已经过时, 不能描述当前数据; 文献 [4] 采用了一种假设数据流到达时间呈正态分布的策略来删除陈旧微簇的策略. 然而大量研究表明, 流式数据到达近似为泊松过程[15], 即到达时间间隔服从负指数分布. 我们利用时齐泊松过程模型, 对同一微簇中的样本的到达时刻进行建模, 相邻两样本的到达时之差应服从

参数为 λ 的负指数分布. 利用 y 的均值作为 $1/\lambda$ 的无偏估计, 可得

$$\frac{1}{\lambda}=\bar{y}=\frac{\sum_{i=1}^{n}y_n}{N}=\frac{T_n-T_0}{N}$$

对每个稀疏微簇进行删除判断: 如果 $T-T_n>\theta(T_n-T_0)/N_{\mathrm{mc}i}$, 那么 oc 属于过期稀疏微簇, 应删除稀疏微簇; θ 是截断阈值, 经验设定为 $\theta=-\ln(0.001)\approx 7$. 另一种是密度值过小的微簇, 表明该微簇很可能是由噪声或野值引入的, 不是主要模式; 由于现存的稀疏微簇要想成长为密集微簇, 就必须提供给稀疏微簇充足的时机发展, 同时及时删除那些真正噪声点的稀疏微簇. 但这样的策略要耗用大量内存, 因而, 我们引入了最低权值机制[6], 周期性地将真正噪声点的稀疏微簇删除. 权值下限定义为

$$\xi(T_c,T_0)=\frac{2^{-\lambda(T_c-T_0+T_p)}-1}{2^{-\lambda T_p}-1} \tag{6}$$

就意味着从当前角度来看, 该离群微簇不太可能成长为一个密集微簇, 很可能是由噪声或野值引入的. 对每个稀疏微簇每隔 T_p 进行一次检查. ξ 是一个当前时间 T_c 和稀疏微簇创建时间 T_0 的函数. 当 $T_c=T_0$ 时, 即在稀疏微簇刚创建的时候, $\xi(t_c,t_0)=1$. 随着时间的流逝逐渐增长, 且 $\lim_{T_c\to\infty}\xi(T_c)=N_{\mathrm{thred}}$, 离群微簇存在的时间越长, 其期望权值越高. 由此根据第 3.2.1 节、第 3.2.2 节可知执行操作如算法 2 所述.

算法 2. Update(mc)

If (T_c mod T_p)
  For (每个密集微簇 rc)
    If ($N_{\mathrm{mc}}(t)<N_{\mathrm{thred}}$)
    此密集微簇已经变为离群点噪声, 删除此密集微簇 rc;
  For (每个稀疏微簇 oc)
    $\xi(T_c,T_0)=\dfrac{2^{-\lambda(T_c-T_0+T_p)}-1}{2^{-\lambda T_p}-1}$
    If ($N_{\mathrm{mc}}(t)<\xi(T_c,T_0)$)
    此稀疏微簇不大可能成为密集微簇, 删除此稀疏微簇 oc;
    If ($T_c-T_n>\theta(T_n,T_0)/N_{\mathrm{mc}i}$)
    那么 oc 属于过期稀疏微簇, 应删除稀疏微簇 oc;
End

3.2.3 基于改进的 WAP 算法的模型重建

暂存盒 Reservoir 的点有可能出现新的类模式, 很有必要在适当的时候将它们加入到模型中去. 本文采用两种标准重新建立 StrDenAP 模型: 一是给定暂存盒已满至 $R_{\max}$, 二是 Page-Hinkley (PH)[13] 检测变化的方法. 当满足这两个标准之一时, 本文提出一种改进的 WAP (NewWAP) 对现有模型中的类代表点和暂存盒中的点一起进行聚类, 从而得到新的模型. 这样既考虑了数据的变化, 又降低了算法的复杂度, 可以满足大规模流数据的实时处理.

当暂存盒已满至 $R_{\max}$ 或 PH 检测大于阈值 χ, 模型需要重建, 将暂存盒里新的类模式加入到当前模型中. 文献 [13] 采用 WAP 算法进行重建聚类, 然而 WAP 算法按式 (3) 修改权值点的相似度和偏好度, 在权值较大时, 会导致权值点成为孤立微簇, 以及出现代表点只能从权值点中产生的问题. 本文试图从消息传递机制的角度改进 WAP 算法. 经过分析我们给出如下命题.

命题 1. 两点之间的相似度 $s(i,k)$ 只与两点的特征向量有关, 而与权值大小无关.

命题 2. 吸引度 $r(i,k)$ 的计算与权值大小无关. 考虑式 (1) 中吸引度 $r(i,k)$ 的计算式中 $\max_{k'\neq k}[a(i,k')+s(i,k')]$, 它只需选择具有最大值的一个数据点 k', 无须考虑 k' 权值大小.

命题 3. 归属度 $a(i,k)$ 的计算与权值大小有关. 考虑式 (2) 中归属度 $a(i,k)$ 和自我归属度 $a(k,k)$ 的计算式中 $\sum_{i'\neq k}^{n}\max[0,r(i',k)]$, 它的值为候选代表点 k 的所有正的支持和, 因此需要考虑支持数据 i' 权值大小. 因此本文给出一种改进的 WAP (NewWAP) 算法, 根据命题 1 不改变数据点的相似度, 其相似度定义为

$$S'(i,k)=\begin{cases}-dist(i,k)^2, & i\neq k\\ s*+D_i, & i=k\end{cases} \tag{7}$$

根据命题 2 和命题 3 可知, 吸引度计算与权值点多少无关, 归属度公式变为

$$a(i,k)\leftarrow\begin{cases}\min\{0,r(k,k)+\\ \quad\sum\limits_{\substack{i'\neq i,\\ i'\neq k}}\max[0,n(i')r(i',k)]\}, & i\neq k\\ \sum\limits_{i'\neq k}^{n}\max[0,n(i')r(i',k)], & i=k\end{cases} \tag{8}$$

相比于 WAP 等处理带权值数据的算法, NewWAP 算法在精度上最接近原始 AP 算法, 在时间耗费和精度上达到了一个很好的平衡.

当模型需要重建时, $E=\{(E_i,N_{\mathrm{mc}i})\}\cup\{(E'_j,1)\}$, 其中 $(E_i,N_{\mathrm{mc}i})$ 表示当前模型微簇特征向量, E'_j 是暂存盒里的离群点, 根据式 (7) 输入相似度矩阵, 并根据 NewWAP 算法进行聚类, 得到新的聚类代表点 $E=\{(F_i,N'_{\mathrm{mc}i})\}$, 并将暂存盒 Reservoir 清空, 其中 F_i 为包含先前代表点 $E_1,E_2,\cdots,E_m$ 和 m' 个暂存盒数据点而形成新的

聚类代表点, 微簇特征向量更新为

$$\begin{aligned}
&E_i \leftarrow F_i\\
&N_{\mathrm{mc}i} \leftarrow N_{\mathrm{mc}1} + N_{\mathrm{mc}2} + \cdots + N_{\mathrm{mc}m} + m'\\
&D_i \leftarrow \sum_{i=1}^{m}(n_i dist(F_i, E_i)^2 + D_i) + \sum_{i=1}^{m'} dist(F_i, E_i')^2\\
&R_i \leftarrow \frac{D_i(t)}{N_{\mathrm{mc}i}(t)}\\
&N_i \leftarrow N_1 + N_2 + \cdots + N_m + m'
\end{aligned} \tag{9}$$

3.3 离线阶段处理

为了实现离线演化分析, 首先需要定义如下概念.

定义 6 (直接密度可达). 如果两个微簇 mc_i, mc_j 距离满足 $dist(e_i, e_j)^2 \le R_i + R_j$, 且 e_i, e_j 是微簇代表点, 则称微簇 mc_i 到 mc_j 是直接密度可达的.

定义 7 (密集微簇相连). 如果存在两个核心微簇 mc_i 和 mc_j 满足如下条件:

条件 1. $1 - E \le D_i/D_j \le 1 + E$, E 为设定的阈值;

条件 2. 满足密度可达条件, 即 $dist(e_i, e_j) \le R_i + R_j$. 则称 mc_i 到 mc_j 是两个体积相近、相邻的密集微簇相连.

由此可得:

1) 由某一时刻的快照得到精确类簇, 从任意微簇出发遍历找到与其密度相连的密集微簇, 利用本文改进的基于密度 NewDBSCAN (New density-based spatial clustering of applications with noise) 的算法形成更为准确的聚类微簇, 具体步骤如算法 3 所示.

2) 获得历史某一时间段的数据流聚簇结果, 实现对数流演化分析, 具体描述如算法 4 所示.

3.4 StrDenAP 算法整体流程

记动态数据流为 DS, 衰减系数为 λ, 密度阈值为 D_{thred}, 减枝策略周期为 T_p, 暂存盒大小为 $R_{\max}$, PH 检测阈值为 χ. 聚类算法描述如下:

1) 初始化微簇. 应用 AP 算法生成初始模型, 得到初始密集微簇.

当新的数据点到达时, 被当前模型吸收.

2) 更新模型. 执行算法 1 Process (x_t).

3) 微簇衰减. 一段时间内没有新的数据到达时, 其特征按照衰减函数 $f = 2^{-\lambda\Delta t}$ 变化, 否则更新聚类微簇的特征.

4) 动态微簇维护. 执行算法 2 Update (mc).

若新到数据点无法被当前模型吸收, 则放入暂存盒 Reservoir 中.

5) 如果检测到 Reservoir 分布情况变化, 则采用 NewWAP 算法重建模型, 并按式 (9) 更新微簇特征.

6) 转向第 2) 步重复执行直到结束.

7) 离线处理. 执行离线聚类算法.

算法 3. 精确聚类算法NewDBSCAN(mc)

输入: 微簇快照 mc
输出: 某一时刻的精确类簇
Do
  Get (mc);//得到一个未处理的密集微簇
  If (mc 是密集微簇) then
    找出所有到 mc 微簇密度相连的微簇;
    利用它们形成一个自然簇;
  End if
Until 所有的微簇都被处理.

算法 4. 演化聚类算法OffMerge(t_1, t_2)

输入: t_1, t_2 两个时刻的密集微簇快照
输出: (t_1, t_2) 这一时间段所到来的数据点的聚簇情况
步骤

1) T_{2-1}:(t_1, t_2) 这一时间段所到来的数据点的中间微簇.

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

a) 对于 T_1, T_2 中共有的微簇

T_2 微簇对应的统计信息 $-T_1$ 中的; 然后再添加到 T_{2-1};

b) T_1 中有, T_2 中没有的微簇

如果在 T_2 时刻的某一微簇中合并过 T_1 时刻的此微簇, 则相减后添加到 T_{2-1};

否则, 此微簇是由噪声点形成的, 因此没有必要添加, 将其剔掉;

c) T_2 中有, T_1 中没有的微簇

很显然, 这样微簇内的点是在这一时间段内出现的, 直接添加到 T_{2-1} 中.

3) 在中间知识库 T_{2-1} 上进行如上 NewDBSCAN(mc) 算法即得结果.

4 实验评估

4.1 实验环境

实验平台: AMD 双核 1.8 GHz CPU, 主频 2.0 GHz, 2 级缓存 4 MByte, 内存 1 GByte, Window XP 操作系统, 算法采用 Matlab2008b 编写.

实验所用数据分为真实数据与仿真数据两部分. 真实数据为 KDDCUP′99 网络入侵检测数据集[16] 和 KDDCUP′98 慈善捐赠数据集[6], KDDCUP′99 来自于麻省理工学院林肯实验室连续两个星期的网络流检测. 数据集共包含近 500 万条 TCP (Transmission control protocol) 连接记录, 包含正常连接类型在内共有 5 类, 其余 4 类 (DoS (Dennial-

of service), R2L (Remote-to-login), U2R (User-to-root), Probe) 为攻击类型, 可细分为 22 种攻击行为, 从其 42 个有效属性中抽取 34 个连续属性来构成本节的实验数据, 取其 10 % 的样本数据集进行测试, 共有 494 021 条记录. KDDCUP′98 慈善捐赠数据集记录了关于慈善捐赠的信息, 共 95 412 条. 对该数据集进行聚类分析可用于揭示具有相似捐赠行为的捐助群体, 我们从 481 个字段中抽取 56 个字段, 所有实验数据都经过标准化处理至 [0, 1] 区间.

仿真数据所采用的实验数据是文献 [6] 使用的进化数据流 EDS, 通过 10 次随机选择 DS1、DS2、DS3 数据集得到的进化数据流 EDS. DS1、DS2、DS3 的数据量都是 20 000, 数据分布如图 2 所示. 为进一步测试算法 StrDenAP 的可扩展性, 我们生成维数和簇数可变化的人工数据集, 人工数据集采用随机径向基函数生成器[3] 产生的随机数据流. 可以产生数量 B 从 3 K 到 300 K 个数据点, 簇的个数 C 介于 8 到 128 之间, 维度 D 变化范围是 10 至 40. 每隔 2 K 个数据点就以概率 pc 改变数据流高斯分布的均值和方差. 且采用如下标记描述人工数据集: B3kC3D4 表示数据集包含 3 000 个数据点, 分别属于 3 个不同的簇, 各数据点的维数为 4.

实验对 StrDenAP 聚类效果进行评估, 同时与 StrAP[13] 和经典的 DenStream[6] 及其改进算法 DenStream2[12] 进行性能比较. 除特殊说明外, 初始化点数 $N_{\text{init}} = 2\,000$, 衰减因子 $\lambda = 0.25$, 阈值参数 $\mu = 3$, 其余参数与文献 [9] 的 StrAP 算法采用同样的设置, DenStream 和 DenStream2 参数设置与文献 [12] 相同, 其中离群点阈值 $\beta = 0.2$, 邻域 $\varepsilon = 16$, 重叠因子 $o = 3$. 同时定义数据流速度 v 为单位时间内处理的实例个数, 窗口长度 H 是创建宏聚类回溯的时间窗长.

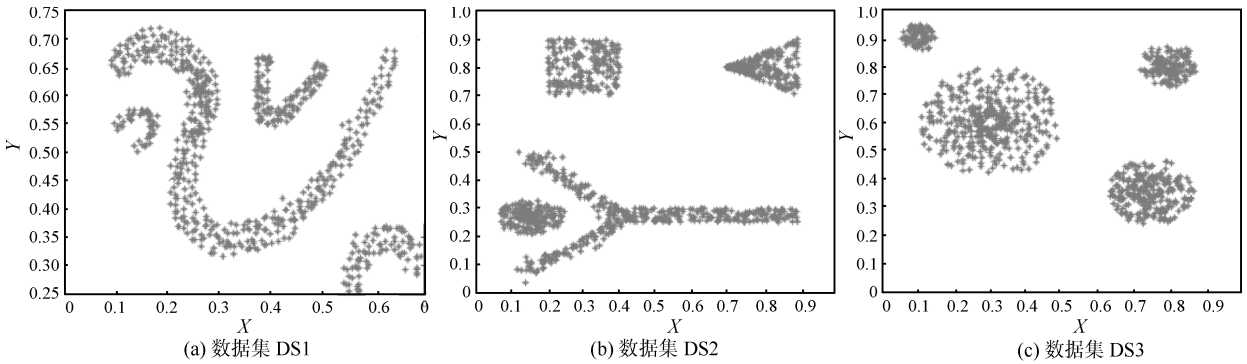
4.2　NewWAP 算法聚类质量

首先对本文所提的 NewWAP 算法进行验证, 采用失真度 (Distortion) 作为聚类评价指标. 具体定义为: 每个点和它所属类的中心点之间距离的平方和, 即 Distortion $= \sum_{i=1}^{N} d^2(x_i, e_i)$, 其中 e_i 是点 x_i 所属类的中心. 实验过程如下[13]: 1) 将人工数据集 B3kC3D4 随机地分为若干个相同的子集 (子集取 $L = 10$), 并分别在每个子集上运行 AP 算法, 从而得到生成各自类代表集合 $Et = \{e_j, n_j\}$, 其中 e_j 是类的代表点, n_j 是以 e_j 作为其类代表点的数据个数; 2) 将 L 个子集的类代表点集合 Et 合并成新的数据集分别运行 NewWAP 算法和 WAP 算法得到最终的聚类中心, 通过调整偏向参数 p 来改变聚类数目[14], 实验结果如图 3 所示.

很明显, Distortion 越小, 就说明相似的点都聚在一个类中, 因此聚类的效果就越好. 当然, Distortion 的比较是要基于相同的类个数 n, n 越大, Distortion 会越小. 所以, 从图 3 中可以看出, 在相同的 n 条件下, NewWAP 相对于 WAP 有更小的 Distortion. 由此可见, 本文所提 NewWAP 算法能够得到更为准确的聚类结果, 实验结果更接近原始 AP 算法, 尤其在有噪声的情况下性能更优, 验证了算法的有效性.

4.3　StrDenAP 聚类质量评价

现在对 StrDenAP 算法采用结果簇的聚类纯度 (Clustering purity) 作为评价指标, 其中聚类纯度定义为: $Pur = [\sum_{i=1}^{K} \frac{|C_i^d|}{|C_i|}/K]$, 其中 K 表示簇的个数, $|C_i^d|$ 表示在簇 i 中具有该簇最主要类标号的数据点数, $|C_i|$ 表示簇 i 中包含的所有数据点的个数. 全局聚类纯度是指在所有窗口内聚类纯度的平均值[15], 算法的全局聚类纯度如表 1 所示.



(a) 数据集 DS1　(b) 数据集 DS2　(c) 数据集 DS3

(a) Data set DS1　(b) Data set DS2　(c) Data set DS3

图 2　三类数据集

Fig. 2　Three kinds of data sets

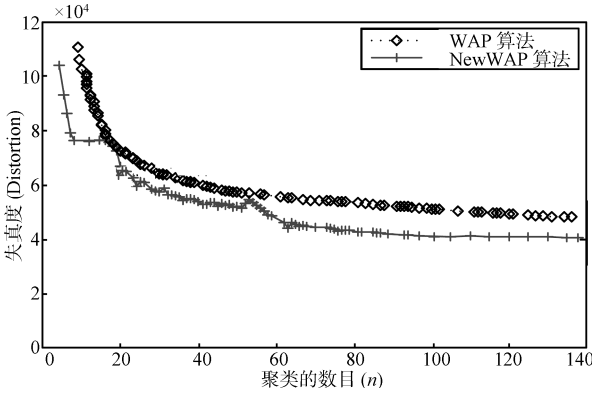


图 3 失真度随聚类数目的变化 n 的比较

Fig. 3 Distortion with clustering number change

我们还在一些代表性的时间戳上进行实验. 为了使结果更准确, 算法各执行 5 次并计算平均聚类纯度作为结果值, 图 4 为 StrDenAP 法对 EDS 数据流在不同速率、不同窗口上的平均聚类纯度指标. 当流速 $v = 1\,000$、$H = 1$ 时, StrDenAP 算法对 EDS 数据流聚类纯度指标最高, StrAP 算法与 DenStream2 算法次之, StrAP 算法受滑动窗口 Δ 的影响较大, 而 DenStream2 算法由于引入了重叠因子解决了微簇重叠时信息丢失的问题, 聚类性能好于 DenStream 算法, 但仍低于本文算法, 我们还将 EDS 数据流流速 $v = 100$、$H = 10$ 时进行实验, 结果如图 4 (b) 所示, 显示了与图 4 (a) 类似的聚类结果, 可见 StrDenAP 算法在较低速率、较大窗口情况下同样能获取较高质量的聚类结果, 说明了本文算法的优越性.

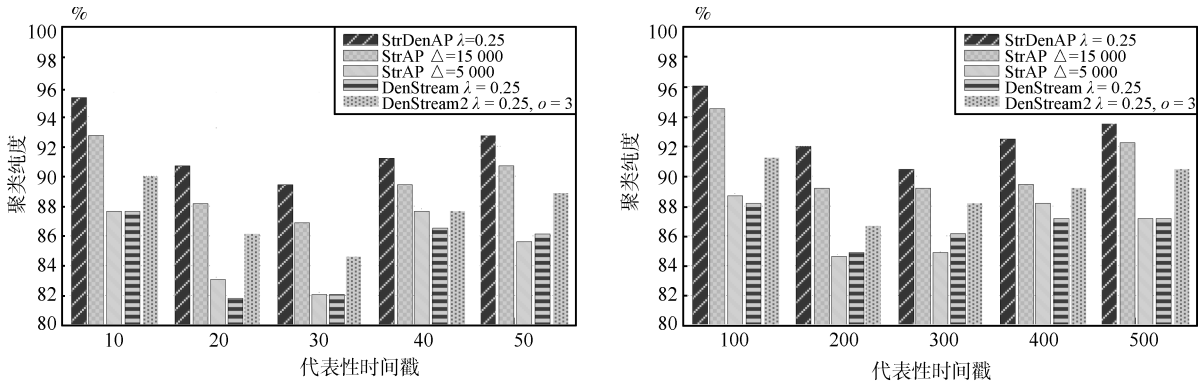
同样, 我们还对 StrDenAP 算法在标准数据集 KDDCUP′99 上特定时刻但窗口 H 大小不同的情况下的聚类纯度进行比较, 通常取入侵开始的时刻 (如表 2 所示). 从图 5 可以看出, 无论窗口大小, StrDenAP 算法在特定时刻的聚类平均纯度值均比其余算法的要好, 说明了本文算法所得到的聚类比较紧凑, 质量较高.

StrDenAP 算法高质量的聚类结果得益于本文所提概要数据结构更能准确捕获当前记录的分布状况, 这将使得 StrDenAP 算法相对于 StrAP 算法更加精准, 同时还得益于它有效的在线微簇维护策略, 在删除离群点噪声的同时维护了数据流中的潜在微簇. 然而在 StrAP 中, 历史过期微簇都被考虑进来, 当且仅当模型重建时才聚类合并微簇, 当噪声点很多时, 将耗费大量的内存来存储那些无用的噪声点, 且误合并了不同的微簇, 导致聚类质量不够理想.

表 1 全局聚类纯度

Table 1 Global clustering purity

数据集	窗口设置	StrDenAP (%) ($\lambda = 0.25$)	StrAP (%) ($\Delta = 15\,000$)	StrAP (%) ($\Delta = 5\,000$)	DenStream (%) ($\lambda = 0.25$)	DenStream2 (%) ($\lambda = 0.25$, $o = 3$)
EDS 进化流	H=1, v=1 000	91.49	87.67	84.34	84.60	85.41
	H=10, v=100	92.33	88.12	85.73	84.98	88.04
KDDCUP′99	H=5, v=1 000	97.89	94.81	92.09	92.75	93.97
	H=10, v=500	98.06	95.79	93.87	93.44	95.23
KDDCUP′99	H=5, v=1 000	92.89	89.01	84.89	88.31	85.31
	H=10, v=500	93.23	89.93	85.92	86.42	89.12



(a) ESD 数据集 ($H = 1$, $v = 1\,000$)

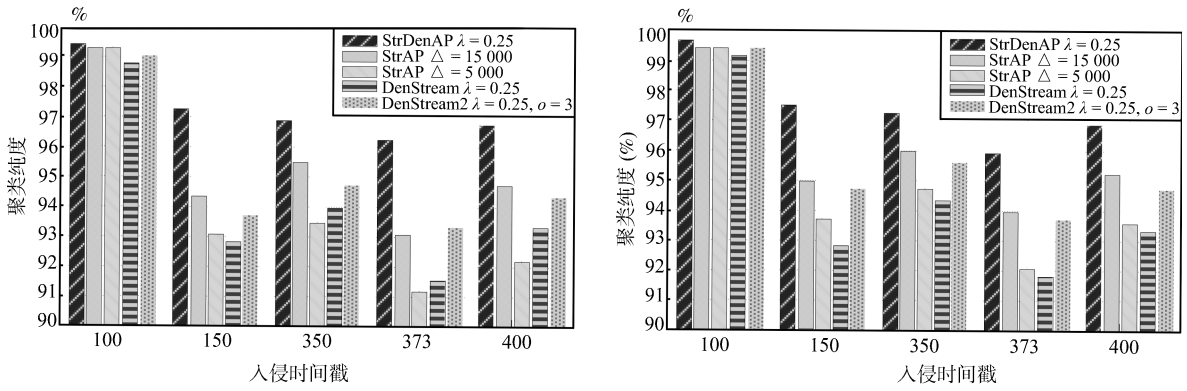
(a) ESD data sets ($H = 1$, $v = 1\,000$)

(b) ESD 数据集 ($H = 1$, $v = 100$)

(b) ESD data sets ($H = 1$, $v = 100$)

图 4 在 EDS 进化数据集上的聚类纯度

Fig. 4 Clustering purity on EDS data sets



(a) 网络入侵数据集 ($H = 1$, $v = 1\,000$)
(a) KDDCUP′99 data sets ($H = 1$, $v = 1\,000$)

(b) 网络入侵数据集 ($H = 5$, $v = 1\,000$)
(b) KDDCUP′99 data sets ($H = 5$, $v = 1\,000$)

图 5　在 KDDCUP′99 数据集上的聚类纯度

Fig. 5　Clustering purity on KDDCUP′99 data sets

StrDenAP 能够更及时地删除无潜力升级的离群点和噪声点, 又提高了数据流的聚类质量. 以上结果也验证了理论分析的正确性.

表 2　入侵时间戳各类别数量

Table 2　Number of categories on invasion time

各攻击类型数量	入侵时间戳 ($v = 1\,000$, $H = 5$) 100	150	350	373	400
Normal	4 000	4 004	4 097	892	406
Satan	0	380	0	0	0
Bufoverflow	0	7	1	2	0
Teardrop	0	99	99	383	0
Smurf	1 000	143	0	819	2 988
Ipsweep	0	52	182	0	0
Ioadmodule	0	6	0	0	1
Rootkit	0	1	0	0	1
Warezclient	0	307	0	0	0
Multihop	0	0	0	0	0
Neptune	0	0	618	2 688	1 603
Pod	0	0	1	99	0
Portsweep	0	0	1	117	1
Land	0	0	1	0	0
总数	5 000	5 000	5 000	5 000	5 000

4.4　有噪声StrDenAP聚类质量评价

为了进一步验证 StrDenAP 算法在有噪声环境下的优越性, 我们对 DS1、DS2、DS3 组成的 EDS 数据集中分别加入了一些随机噪声点. 图 6 分别显示了 StrDenAP 在具有 1 % 随机噪声 EDS 数据集和具有 5 % 随机噪声 EDS 数据集上的聚类结果, 图中矩形部分 ST1、ST2、ST3 分别为 DS1、DS2、DS3 过渡状态的聚类纯度, 不同簇间重叠交叉导致在过渡态的纯度低于其他时刻. 在矩形时间之外, 宏聚类纯度均在 95 % 以上. 可见, StrDenAP 在有噪声环境下同样能取得更好的聚类质量. 结果表明改进后的算法在有噪声环境下同样能取得更好的聚类质量, 尤其在数据流含有大量噪声时, 改善效果更加明显.

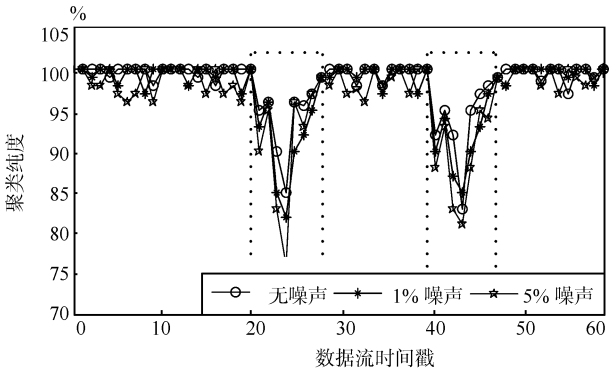


图 6　EDS 数据集在不同的噪声比例的聚类纯度 ($H = 5$, $v = 1\,000$)

Fig. 6　Clustering purity on EDS datasets of different noise ratio ($H = 5$, $v = 1\,000$)

4.5　执行时间测试

我们还对不同规模的 KDDCUP′99 网络入侵检测数据集和进化数据流 EDS 测试两种算法的执行时间. 为公平起见, 将重建算法均选取 NewWAP 算法, 图 7 显示了测试的结果. 由图 7 可以清晰看到, StrDenAP 和 StrAP 算法的执行时间随着数据集规模的增加而增长, 仅在 2 s～5 s 便可处理 1 万条记录, 基本满足实时聚类要求. 注意到, StrDenAP 所用时间的增长速率低于 StrAP, 这是因为 StrAP 保存了更多需要处理的历史过期数据, 没有及时有效地删除过期数据, 而 StrDenAP 根据微簇密度的变化自适应地调整聚类, 能够更及时地删除过期数据

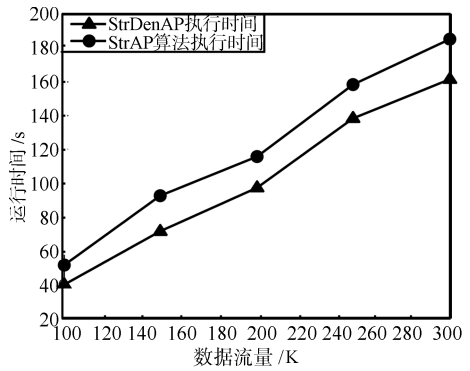
和噪声点, 大大减少了工作量, 提高了聚类速度.

4.6 伸缩性测试

为了进一步测试本文算法, 在维数不同和簇数不同的数据流上对 StrDenAP 的执行时间进行算法比较. 为了获取具有自然簇个数和数据对象维度的任意组合, 在生成人工数据集时变化维度 D 和簇数 C. 我们从 10 到 50 改变簇数 C, 并固定数据对象个数为 100 000 和维度 D. 图 8 (a) 显示 StrDenAP 算法处理数据流的时间与簇的个数基本呈线性增长关系. 同时可以看到, 执行时间增长并不显著, 这是由于密集微簇的数量随自然簇个数的增加并没有过多的增长. 例如, 对于维度 $D = 25$ 的人工数据集, 当簇的个数 C 从 10 增加到 50 时, 处理时间基本上呈线性增长. 另外, 如图 8 (b) 所示, 从 10 到 40 变化数据维数, 可以发现, 随着数据维度的增加, 处理时间也随之线性增长.

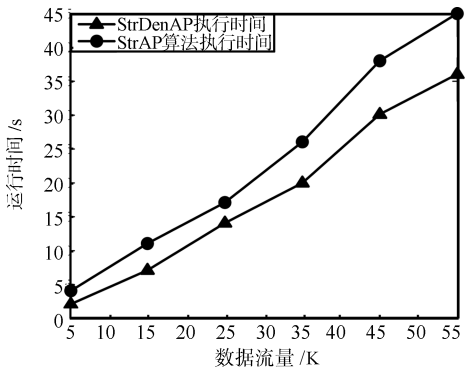
4.7 参数敏感性分析

下面对影响 StrDenAP 算法的两个重要参数衰减因子 λ 和离群点阈值 μ 进行敏感性分析. 衰减因子 λ 控制着历史数据对当前簇的重要程度. 此前实验设置值 $\lambda = 0.25$. 现将 λ 逐步增大来观察聚类质量的变化情况 ($0.05 \leq \lambda \leq 5.5$). 图 9 (a) 显示了实验结果. 当 λ 被设置为一个相对较小或较大的值时, 聚类质量相对较差. 特别地, 当 $\lambda = 0$ 时, 微簇密度并未衰减, 从而影响了聚类效果; 当 $\lambda > 4$ 时, 数据点在它们到达后迅速衰减, 只有少数近期数据对最终结果有贡献, 导致聚类结果也不理想. 可以看到, 若 λ 处于 0.2 到 0.8 之间时, 聚类质量非常理想和稳定, 且总是高于 95 %. 离群点阈值 μ 决定了密集微簇和稀疏微簇的比例. 将 μ 逐步增大来观察聚类质量的变化情况, 图 9 (b) 显示了实验结果. 当 μ 被设置在 (2, 6) 之间时, 聚类质量稳定且保持较高水平. 然而, μ 不宜被设置的过大, 可以看到随着 μ 的增大, 聚类质量显著降低. 这是因为离群点阈值 μ 设置过高导致密集微簇数量急剧减少, 大量稀疏微簇随着时间流逝未及时变为密集微簇而都被当成离群噪声删除, 导致聚类质量有所下降.



(a) 网络入侵数据集 ($H = 5$, $v = 1\,000$)

(a) Network intrusion datasets ($H = 5$, $v = 1\,000$)

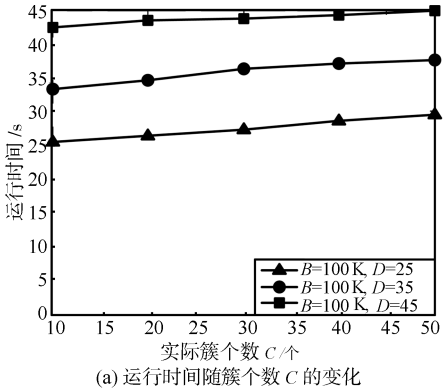


(b) EDS 进化数据集 ($H = 5$, $v = 1\,000$)

(b) EDS evolutionary data stream ($H = 5$, $v = 1\,000$)

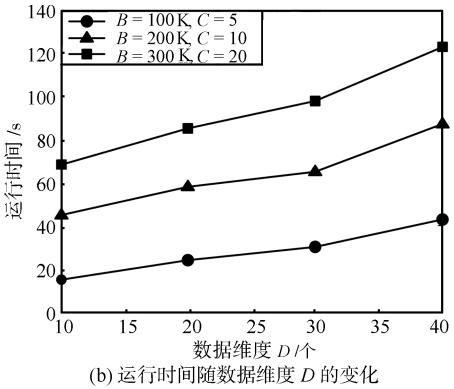
图 7 在不同数据集上算法执行时间比较

Fig. 7 Execution time on different data sets



(a) 运行时间随簇个数 C 的变化

(a) Execution time change with the number of clusters

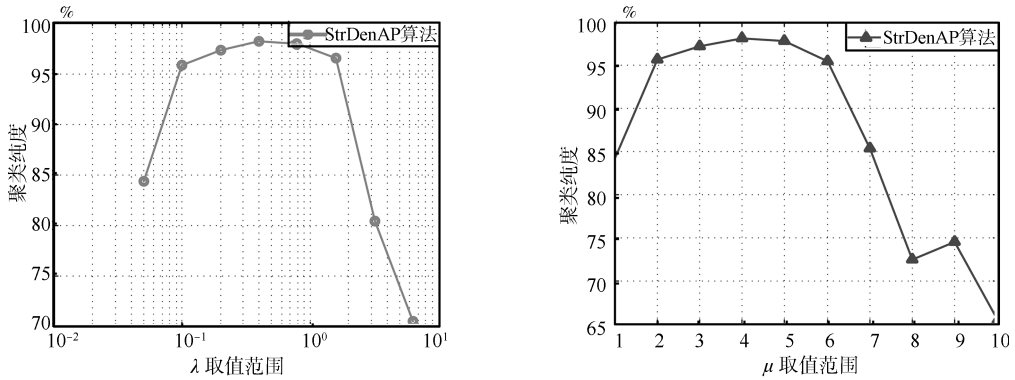


(b) 运行时间随数据维度 D 的变化

(b) Execution time change with the data dimension

图 8 运行时间随维度与簇数变化的比较

Fig. 8 Execution time on different dimensions and cluster number



(a) 衰减因子 λ 变化 ($0.05 \leq \lambda \leq 5.5$)
(a) Decay factor λ changes ($0.05 \leq \lambda \leq 5.5$)

(b) 离群点阈值 μ 变化 ($1 \leq \mu \leq 10$)
(b) Outliers threshold μ changes ($1 \leq \mu \leq 10$)

图 9　聚类质量随参数的变化

Fig. 9　Execution time on different parameters

5　总结

针对数据流聚类的难点问题, 本文提出了一种新的数据流聚类算法 StrDenAP. 与 StrAP 算法相比, StrDenAP 算法具有以下特点: 1) 考虑了数据流的时态特征, 引入数据点衰减密度来区分 "旧的" 数据和 "新的" 数据; 2) 提出了在线维护策略, 能够根据密度的动态变化, 适时剔除过期和孤立点微簇, 在噪声环境下具有较高的聚类纯度; 3) 聚类模型更加准确, 能较好地处理野值点噪声数据. 实验结果表明: 由于考虑了微簇衰减密度的变化, 该算法能够捕捉数据流的时态特征, 并且通过本文所提的 NewWAP 算法进行模型重建, 使聚类模型更加准确; 该算法具有较高的聚类能力和可伸缩性, 运行效率和精度达到了很好的平衡. 然而在实际应用中, 许多通信结构逐渐呈现出一种分布式特征, 如何有效地将 StrDenAP 算法扩展到分布式环境下, 则是我们进一步的研究方向.

References

1 Zhu Qun, Zhang Yu-Hong, Hu Xue-Gang, Li Pei-Pei. A double-window-based classification algorithm for concept drifting data streams. *Acta Automatica Sinica*, 2011, **37**(9): 1077−1084
(朱群, 张玉红, 胡学钢, 李培培. 一种基于双层窗口的概念漂移数据流分类算法. 自动化学报, 2011, **37**(9): 1077−1084)

2 Hassani M, Spaus P, Gaber M M, Seidl T. Density-based projected clustering of data streams. In: Proceedings of the 2012 Scalable Uncertainty Management, Berlin Heidelberg, Springer, 2012. 311−324

3 Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, Seidl T. MOA: massive online analysis, a framework for stream classification and clustering. *The Journal of Machine Learning Research*, 2010, **99**: 1601−1604

4 Aggarwal C C, Han J W, Wang J Y, Yu P S. A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases (Vol. 29), VLDB Endowment, 2003. 81−92

5 Aggarwal C C, Han J W, Wang J Y, Yu P S. A framework for projected clustering of high dimensional data streams. In: Proceedings of the 30th International Conference on Very Large Data Bases (Vol. 30), VLDB Endowment, 2004. 852−863

6 Cao F, Ester M, Qian W, Zhou A Y. Density-based clustering over an evolving data stream with noise. In: Proceedings of the 2006 SIAM International Conference on Data Mining, Bethesda, USA, 2006. 328−339

7 Chen Y, Tu L. Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, San Jose, California, 2007. 133−142

8 Tu L, Chen Y. Stream data clustering based on grid density and attraction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2009, **3**(3): 12−20

9 Yang Ning, Tang Chang-Jie, Wang Yue, Chen Yu, Zheng J L. Clustering algorithm on data stream with skew distribution based on temporal density. *Journal of Software*, 2010, **21**(5): 1031−1041
(杨宁, 唐常杰, 王悦, 陈瑜, 郑皎凌. 一种基于时态密度的倾斜分布数据流聚类算法. 软件学报, 2010 **21**(5): 1031−1041)

10 Ntoutsi I, Zimek A, Palpanas T, Kröger P, Kriegel H P. Density-based projected clustering over high dimensional data streams. *SIAM SDM*, 2012, **12**: 987−998

11 Yu Yan-Wei, Wang Qin, Kuang Jun, He Jie. An on-line density-based clustering algorithm for spatial data stream. *Acta Automatica Sinica*, 2012, **38**(6): 1051−1058
(于彦伟, 王沁, 邝俊, 何杰. 一种基于密度的空间数据流在线聚类算法. 自动化学报, 2012, **38**(6): 1051−1058)

12 Tang J. An Algorithm for Streaming Clustering [Ph. D. dissertation], Uppsala University, Sweden, 2011

13 Zhang X, Furtlehner C, Sebag M. Data streaming with affinity propagation. In: Proceedings of the 2008 Machine Learning and Knowledge Discovery in Databases, Berlin Heidelberg, Springer, 2008. 628−643

14 Wang Kai-Jun, Zhang Jun-Ying, Li Dan, Zhang Xin-Na, Guo Tao. Adaptive affinity propagation clustering. *Acta Automatica Sinica*, 2007, **33**(12): 1242−1246
(王开军, 张军英, 李丹, 张新娜, 郭涛. 自适应仿射传播聚类. 自动化学报, 2007, **33**(12): 1242−1246)

15 Huang De-Cai, Wu Tian-Hong. Density-based clustering algorithm for mixture data sets. *Control and Decision*, 2010, **25**(3): 416−421
(黄德才, 吴天虹. 基于密度的混合属性数据流聚类算法. 控制与决策, 2010 **25**(3): 416−421)

16 Zhao L, Kang H S, Kim S R. Improved clustering for intrusion detection by principal component analysis with effective noise reduction. In: Proceedings of the 2013 Information and Communicatiaon Technology, Berlin Heidelberg, Springer, 2013. 490−495

张建朋 国家数字交换系统工程技术研究中心博士研究生. 主要研究方向为电信网信息关防, 数据流挖掘. 本文通信作者. E-mail: zhangjianpeng@gmail.com
(**ZHANG Jian-Peng** Ph. D. candidate at the National Digital Switching System Engineering and Technological Research and Development Center. His research interest covers telecommunication network protection and data stream mining. Corresponding author of this paper.)

陈福才 国家数字交换系统工程技术研究中心研究员. 主要研究方向为电信网信息关防.
E-mail: chenfucai@ndsc.com.cn
(**CHEN Fu-Cai** Professor at the National Digital Switching System Engineering and Technological Research and Development Center. His research interest covers telecommunication network protection.)

李邵梅 博士, 国家数字交换系统工程技术研究中心讲师. 主要研究方向为图像处理与模式识别.
E-mail: lishaomei@ndsc.com.cn
(**LI Shao-Mei** Ph. D., lecturer at the National Digital Switching System Engineering and Technological Research and Development Center. Her research interest covers telecommunication network protection and data stream mining.)

刘力雄 博士, 国家数字交换系统工程技术研究中心副教授. 主要研究方向为图像处理与模式识别.
E-mail: lishaomei@ndsc.com.cn
(**LIU Li-Xiong** Ph. D., associate professor at the National Digital Switching System Engineering and Technological Research and Development Center. His research interest covers telecommunication network protection and data stream mining.)