基于欧拉核的数据流聚类算法

朱颖雯[1,2] 杨 君[1]

(三江学院计算机科学与工程学院 南京 210012)[1]

(南京航空航天大学计算机科学与技术学院 南京 210016)[2]

摘 要 随着云计算、物联网的快速发展,数据采集变得更加快捷和自动化。许多新型的应用领域中,诸如实时监控系统、车辆交通监控系统、电力消耗记录以及网络流量监控等,每时每刻都在产生大量的流数据,对数据流挖掘的研究成为了热点问题。聚类分析作为数据流挖掘领域的一个重要问题,在近期被高度重视并得到广泛研究。不同于传统的静态数据聚类问题,数据流聚类受到有限内存、一遍扫描、实时响应和概念漂移等许多约束。为此,文中基于欧拉核提出了一种针对数据流的聚类算法。首先通过欧拉核显式地将数据映射到相同维度的复数特征空间,然后在特征空间中基于 GNG 模型进行聚类。欧拉核依赖于非线性鲁棒的 cosine 度量,故对野值低敏感;显式的映射避免了一般的核聚类算法需要使用核技巧而无法处理数据流的问题。实验数据表明,基于欧拉核的数据流聚类算法不仅表现出了较好的聚类性能,还识别了数据的结构信息。

关键词 GNG,数据流聚类,欧拉核,核方法

中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/jsjkx.190600158

Euler Kernel-based Data Stream Clustering Algorithm

ZHU Ying-wen[1,2] YANG Jun[1]

(School of Computer Science and Engineering, Sanjiang University, Nanjing 210012, China)[1]

(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]

Abstract With the advance of both cloud computing and internet of things, many applications generate huge amounts of data streams at fast speed. Examples include real-time surveillance systems, vehicle traffic monitoring systems, electricity consumption recording, and network traffic monitoring. Data stream mining has become a hot research topic. Its goal is to extract hidden knowledge/patterns from continuous data streams. Clustering, one of the most important problems in stream mining, has been highly explored. Different from traditional data clustering algorithm where given datasets are generally static and can be repeatedly read and processed, clustering data streams face more challenges due to having to satisfy such constraints as bounded memory, single-pass, real-time response and concept-drift detection. This paper presented a new clustering algorithm for data streams, called EG-Stream, by combining the Euler kernel method with the Growing Neural Gas(GNG) model. It can not only maintain the benefit of nonlinear modeling using kernel function, but also significantly solve the large scale computational problem in kernel-based clustering. Euler kernel is relying on a nonlinear and robust cosine metric that is less sensitive to outliers. More important, it intrinsically induces an empirical map which maps data onto a complex space of the same dimension, and it takes these advantages to measure the similarity between data in a robust way without increasing the dimensionality of data, which avoids the problem that other kernel clustering algorithms can not deal with data streams. Although this method is embarrassingly simple just by incorporating the Euler kernel into GNG, the experimental results on variety of UCI datasets indicate that this method can still achieve comparable or even better performance than G-Stream algorithm, and identify the structural information from stream data.

Keywords GNG, Data stream clustering, Euler kernel, Kernel method

---

到稿日期:2019-05-26 返修日期:2019-08-12 本文受江苏省三江学院校科研项目(2018SJKY026),江苏省普通高校自然科学研究资助项目(17KJD520007),江苏省高等学校自然科学研究面上项目(18KJB520042)资助。

朱颖雯(1982—),女,硕士,讲师,主要研究方向为机器学习,E-mail:83782796@qq.com(通信作者);杨 君(1982—),女,博士,副教授,主要研究方向为云计算。

1　引言

随着云计算、物联网的快速发展，尤其是小型无线传感器设备的广泛应用，数据采集变得更加快捷和自动化。许多新型的应用领域中，诸如实时监控系统、气象卫星遥感、网络通信检测以及电力供应网等，每时每刻都在产生大量的数据。这些数据并不事先存放在存储介质中，而是像水流一样不断出现，它们具有快速（High Speed）、时序（Temporally Ordered）、海量（Massive）等特征，被称作数据流（Data Stream）。数据流的大量产生和应用需求的不断增加，使得数据流挖掘研究变得炙手可热。挖掘数据流[1-7]的目的是从这些连续不断的流数据中提取隐藏的知识结构。数据流挖掘技术包括数据流分类、数据流聚类、数据流上的关联规则挖掘等。数据流聚类是数据流学习中的一项重点任务，它是将数据对象集合中的相似对象划分为一个或多个组（称为“簇”，Cluster）的过程，划分后，同一簇中的元素彼此相似，但相异于其他簇中的元素。不同于传统的静态数据聚类问题，数据流聚类面临许多约束：1）有限内存（Bounded Memory），数据流中的数据是海量的，因此不可能在内存及硬盘上存储整个数据流集；2）一次扫描（single-pass），同样地，因为数据量巨大，传统的多遍扫描数据的方法不再适用，对其进行挖掘的过程应该是一个单遍扫描的过程，且对流中数据元素的访问只能是单次线性的（Linear Scan），即只能按照流入顺序依次读取一次，随机访问是不现实的；3）实时响应（Real-time Response），多数应用要求很快的响应速度，并且挖掘应该是一个连续、在线的过程；4）概念漂移检测（Concept-drift Detection），数据分布可能会随着时间的推移而发生变化。

目前，数据流聚类算法研究已经在学术界和工业界得到广泛的关注，许多相关算法已被提出[8-22]。然而，现有方法均只能解决线性聚类问题，且对噪声和野值点极其敏感，鲁棒性较差。核方法是一类强有力的非线性方法，其目的是将数据由低维特征空间映射到高维特征空间，以进一步增强传统线性学习器的非线性学习能力。鉴于 KPCA[23] 和 KLDA[24] 等核代入算法的成功应用，在聚类分析中将核方法代入传统聚类算法，形成了一系列的经典核聚类算法，如核模糊 C-均值（Kernel Fuzzy C-Means，KFCM）[25-26]、核 K-均值（Kernel K-Means，KKM）[27] 等。这些核化的聚类算法通常具有更好的性能，但因计算复杂度和存储开销较高而仅适用于规模较小的数据集。文献[28]提出了欧拉 PCA 方法来解决大数据处理问题。文献[29]使用欧拉核解决了基于核 K-means 的硬划分聚类问题。鉴于数据流聚类面临的约束，少有研究者将核方法应用于数据流聚类。

本文的主要贡献如下：

1）本文提出了一种基于欧拉核的数据流聚类算法，首先通过欧拉核显式地将数据映射到相同维度的复数特征空间，然后在特征空间中使用 GNG 模型进行数据流聚类；

2）将欧拉核应用于数据流聚类，充分利用欧拉核函数的非线性鲁棒 cosine 度量及其对野值低敏感等特性，通过显式映射避免了一般的核聚类算法需要使用核技巧而无法处理数据流的问题；

3）在多个模拟和真实数据集上进行实验，实验结果验证了本文提出的数据流聚类算法在大规模数据流分析任务中具有较好的性能。

2　相关工作

设数据流 $\mathcal{DS}$ 为一个带有时间戳（Time Stamp）的多维数据点集合，$\mathcal{DS}=\{x_1,x_2,\cdots,x_n\}$（实际应用中 n 的取值可以为无限大），其中每个数据点 $x_i=(x_i^1,x_i^2,\cdots,x_i^d)$ 是一个 d 维的数据记录，其到达时间为 t_i。数据流本身具有无限、时序、动态等特性。与传统的聚类挖掘算法相比，由于数据流面临有限内存、一次扫描、实时响应、概念漂移检测等约束，数据流挖掘算法仅能产生近似的结果（Approximate Results）。

到目前为止，现有的数据流聚类算法均由传统聚类算法扩展而来。根据被扩展的传统算法的不同，可以将已有数据流聚类算法分为 5 类：基于划分的方法（STREAM[8]）、基于层次的方法（CluStream[9]，HPStream[10]，SWClustering[11]，E-Stream[12]，REPSTREAM[13]）、基于密度的方法（DenStream[14]，OPTICS-Stream[15]，incPre-Decon[16]）、基于网格的方法（D-Stream[17]，MR-Stream[18]，CellTree[19]）、基于模型的方法（SWEM[20]，GCPSOM[21]，G-Stream[22]）。表 1 分别基于如下特性对现有的方法进行了总结：1）基算法（Basic Clustering Algorithm it Depends on）；2）使用的计算策略（在线学习或 2 步学习）；3）聚簇个数是否自适应；4）是否可挖掘拓扑结构；5）是否可以检测概念漂移；6）是否适合高维数据（High Dimensional Data）。

表 1　数据流聚类算法的比较

Table 1　Comparison of various data stream clustering algorithms

算法	基算法	在线学习	自适应聚簇个数	发掘拓扑结构	检测概念漂移	适合高维数据
STREAM[8]	k-medians	√	×	×	×	×
CluStream[9]	BIRCH	×	√	×	√	×
HPStream[10]	BIRCH	×	√	×	√	√
SWClustering[11]	BIRCH	×	√	√	×	×
E-Stream[12]	BIRCH	√	√	×	√	×
REPSTREAM[13]	CHAMELEON	√	√	×	√	×
DenStream[14]	DBSCAN	×	√	×	√	×
OPTICS-Stream[15]	OPTICS	×	√	×	√	×
incPre-Decon[16]	PreDecon	√	√	×	×	√
D-Stream[17]	DENCLUE	×	√	×	√	×
MR-Stream[18]	STING	×	√	×	√	×
CellTree[19]	STING	×	√	×	√	×
SWEM[20]	EM	√	√	×	√	×
GCPSOM[21]	SOM	√	√	√	√	×
G-Stream[22]	GNG	√	√	√	√	×

基于划分的数据流聚类方法相对简单且易于实现，但需要预先定义聚类簇个数，然而由于流数据分布未知，因此聚簇个数无法得到。此外，该方法无法检测到概念漂移。例如，STREAM 算法[8] 采用聚集运算（统计每一块已处理数据的聚类中心和划分到各类的数据个数）成功地将批处理聚类推广到数据流聚类。事实上，STEAM 算法也可被看作一种特殊的层次聚类算法，即对最初 n 个数据划分块，并对每一数据块采用 K-means 聚类得到 O(TK) 个聚类中心（T 为划分的数据块数，K 为每一块的聚类类数），然后重复上述过程，直至得到最终 O(K) 个中心为止。

基于层次的数据流聚类方法虽然能够发现有意义的聚簇结构，但通常具有较高的计算代价，而且对流数据到达的顺序

敏感。例如，CluStream 算法[9]沿用了 BIRCH 算法的思想，利用数据流时序变化性特点，动态聚类部分数据，而非对整体数据流聚类，从而使其具有良好的扩展性且能产生高性能的聚类结果。该框架由联机(Online)和脱机(Off-line)两部分组成。联机部分通过微聚类(Micro-clusters)统计数据流信息；脱机部分根据用户需求，利用所得统计信息输出最终的聚类。同时，为了体现数据流的时序变换过程，进一步引入了倾斜时间窗口框架，以增加现有数据的数据时间粒度(权重)，从而逐步降低历史数据的权重。HPStream 算法[10]沿用了 CluStream 的脱机和联机框架，但不同的是：1)HPStream 将数据空间投影到相关维的子空间，以达到有效聚类高维数据的目的；2)针对数据流的特点，HPStream 利用用户可调的衰减因子，为每个数据赋予一个随时间进行指数衰减的权重，从而体现了数据流的动态变换性，并有效地降低了内存需求。

基于密度的数据流聚类方法可以发现任意形状的聚簇，但是算法预设的参数太多。例如，DenStream 算法[14]提高了数据流聚类算法聚类任意形状数据的能力，并成功地将基于密度的聚类思想引入到数据流聚类框架中。实质上，DenStream 同样延续了 CluStream 算法的处理框架，不同之处在于其联机部分采用 DBSCAN 算法来搜集数据统计信息，由此在聚类大规模数据集的同时可聚类任意形状的数据集，但它同样也保留了 DBSCA 仅适用于聚类均匀分布数据集的缺陷。D-Stream 算法[17]是在 DenStream 基础上衍生出的新算法，不同于 DenStream 直接利用数据原空间聚类，其借鉴了基于网格聚类算法的优点，首先将数据投射到网格量化空间，再在此量化空间中合并稠密度区域以实现最终聚类。由此，D-Stream 成功融合了基于密度的方法可聚类任意形状数据和基于网格的聚类算法可推广到大规模数据的优势。

基于网格的数据流聚类方法的运行速度很快，可以发现任意形状的聚簇，但是其聚类质量取决于选取的网格粒度。

基于模型的数据流聚类算法包含了很多领域知识，其强依赖于假设模型。例如，SWEM 算法基于 EM 模型，GCPSOM 算法基于 SOM 模型，G-Stream 算法基于 GNG 模型。从表 1 可以发现，在线(联机)算法是处理数据流聚类的一种很好的策略，在线学习可以解决数据流约束中的一次扫描、实时响应和有限内存问题。STREAM[8]，REPSTREAM[13]，incPre-Decon[16]，SWEM[20]，GCPSOM[21]和 G-Stream[22]都是在线算法，但只有 REPSTREAM，SWEM，GCPSOM 和 G-Stream 可以处理概念漂移问题，即这些算法能够随着数据的流动更新新来的概念并移除旧的概念。GCPSOM 和 G-Stream 不仅可以应对数据流挖掘中的各类约束，而且可以发现数据的拓扑结构，它们分别基于 SOM(Self-organizing Maps)和 GNG(Growing Neural Gas)模型，但均无法处理非线性学习问题。因此，本文引入核方法，提出了一种基于 GNG 模型核化的数据流聚类算法。

3 欧拉核(Euler)函数

欧拉核是一类特殊的核函数，即通过欧拉公式将实数空间的数据显式地映射到复数空间，核空间数据与原始数据的维数相同。

令 $X=\{x_i,i=1,2,\cdots n\}$，$x_i\in R^d$ 是一个样本，定义映射 $\Phi=\{\phi(x_1),\cdots,\phi(x_n)\}$ 将数据由低维输入空间 X 映射到高维特征空间 F。通过 $\phi(\cdot)$，核矩阵 $\boldsymbol{K}$ 被定义为 $\boldsymbol{K}=\boldsymbol{\Phi}^{\mathrm{H}}\boldsymbol{\Phi}$[28]，因此 $K(x_j,x_q)$ 可以表示为：

$$K_{jq}=\frac{1}{2}\sum_{c=1}^{d}\cos(\alpha\pi(x_j(c)-x_q(c)))-\mathrm{i}\,\frac{1}{2}\sum_{c=1}^{d}\sin(\alpha\pi(x_j(c)-x_q(c))) \tag{1}$$

其中，$x_j(c)$ 表示 x_j 的第 c 个分量。不同于已经存在的 Mercer 核，核矩阵 $\boldsymbol{K}$ 被定义在复数空间中，并且满足 $\boldsymbol{K}^{\mathrm{H}}=\boldsymbol{K}$，因为 $K_{qj}=\overline{K_{jq}}$($\overline{\cdot}$ 表示复共轭算子)。

实际上，欧拉核引入了一个显式的映射 $\phi(x_j)$，可以将数据 x_j 从一个 d 维实数空间 R^d 映射到一个 d 维的复数 RKHS 空间 C^d，即 $R^d\to C^d$。$\phi(x_j)$ 如式(2)所示：

$$\phi(x_j)=\frac{1}{\sqrt{2}}\mathrm{e}^{\mathrm{i}\alpha\pi x_j}=\frac{1}{\sqrt{2}}(\cos(\alpha\pi x_j)+\mathrm{i}\sin(\alpha\pi x_j)) \tag{2}$$

其中，i 为虚数单位。当 $0<\alpha<2$ 时，此映射为双射。

基于式(1)，可以在 RKHS 中计算映射 $\phi(x_j)$ 和 $\phi(x_q)$ 的欧氏距离，如式(3)所示：

$$\begin{aligned}d(\phi(x_j),\phi(x_q))&=\|\phi(x_j)-\phi(x_q)\|^2\\&=(\phi(x_j)-\phi(x_q))^{\mathrm{T}}(\phi(x_j)-\phi(x_q))\\&=\sum_{c=1}^{d}(1-\cos(\theta_j(c)-\theta_q(c)))\end{aligned} \tag{3}$$

其中，$\theta_j(c)$ 表示 $\alpha\pi x_j$。

式(3)的证明如下：

$$\begin{aligned}&d(\phi(x_j),\phi(x_q))\\&=\|\phi(x_j)-\phi(x_q)\|^2\\&=(\phi(x_j)-\phi(x_q))^{\mathrm{T}}(\phi(x_j)-\phi(x_q))\\&=\sum_{c=1}^{d}\frac{1}{\sqrt{2}}(\cos\theta_j(c)+\mathrm{i}\sin\theta_j(c)-\cos\theta_q(c)-\mathrm{i}\sin\theta_q(c))^{\mathrm{T}}\\&\quad\frac{1}{\sqrt{2}}(\cos\theta_j(c)+\mathrm{i}\sin\theta_j(c)-\cos\theta_q(c)-\mathrm{i}\sin\theta_q(c))\\&=\sum_{c=1}^{d}\frac{1}{2}((\cos\theta_j(c)-\cos\theta_q(c))+\mathrm{i}(\sin\theta_j(c)-\sin\theta_q(c)))^{\mathrm{T}}((\cos\theta_j(c)-\cos\theta_q(c))+\mathrm{i}(\sin\theta_j(c)-\sin\theta_q(c)))\\&=\sum_{c=1}^{d}\frac{1}{2}((\cos\theta_j(c)-\cos\theta_q(c))-\mathrm{i}(\sin\theta_j(c)-\sin\theta_q(c)))((\cos\theta_j(c)-\cos\theta_q(c))+\mathrm{i}(\sin\theta_j(c)-\sin\theta_q(c)))\\&=\sum_{c=1}^{d}\frac{1}{2}(M-\mathrm{i}N)(M+\mathrm{i}N)=\sum_{c=1}^{d}\frac{1}{2}(M^2+N^2)\\&=\sum_{c=1}^{d}\frac{1}{2}((\cos\theta_j(c)-\cos\theta_q(c))^2+(\sin\theta_j(c)-\sin\theta_q(c))^2)\\&=\sum_{c=1}^{d}\frac{1}{2}(\cos^2\theta_j(c)+\cos^2\theta_q(c)-2\cos\theta_j(c)\cos\theta_q(c)+\sin^2\theta_j(c)+\sin^2\theta_q(c)-2\sin\theta_j(c)\sin\theta_q(c))\\&=\sum_{c=1}^{d}\frac{1}{2}(2-2\cos\theta_j(c)\cos\theta_q(c)-2\sin\theta_j(c)\sin\theta_q(c))\\&=\sum_{c=1}^{d}(1-1\cos\theta_j(c)\cos\theta_q(c)-1\sin\theta_j(c)\sin\theta_q(c))\end{aligned}$$

$$=\sum_{c=1}^{d}(1-\cos M\cos N-\sin M\sin N)$$

$$=\sum_{c=1}^{d}(1-\frac{1}{2}(\cos(M+N)+\cos(M-N)+$$

$$\frac{1}{2}(\cos(M+N)-\cos(M-N)))$$

$$=\sum_{c=1}^{d}(1-\cos(M-N))=\sum_{c=1}^{d}(1-\cos(\theta_j(c)-\theta_q(c)))$$

可以看出，虽然核映射 ϕ 和核矩阵 $\boldsymbol{K}$ 都是被定义在复数空间中，但是距离函数 $d(\cdot,\cdot)$ 是一个实数，因而此函数可以作为点与点之间的相似度度量。cosine 函数是一个周期函数，它的值域为[－1，1]，因此式(3)对野值低敏感。通过式(2)，可以将输入数据直接显式地映射到 RKHS 空间。

4　基于欧拉核的数据流聚类算法

基于欧拉核在非线性学习中的优良性质，本文提出了一种基于欧拉核的数据流聚类算法 EG-Stream。首先，通过欧拉核显式地将数据映射到相同维度的复数特征空间；然后，在特征空间中基于 GNG (Growing Neural Gas) 模型对数据进行聚类。EG-Stream 算法生成了一个包含一系列神经元节点(Nodes)的图结构，其中神经元节点 c 代表聚类(Cluster)，且每个神经元节点均包含一个相关的权值向量 $w_c=(w_c^1, w_c^2,\cdots,w_c^d)$、一个距离阈值 δ_c 和一个误差变量 $error(c)$。

EG-Stream 首先使用式(2)将每个到达的数据点 x_j 映射到 d 维的复数空间 $\phi(x_j)$，然后利用 GNG 模型挖掘聚类。在挖掘过程中，EG-Stream 根据先到达的一些数据点(如 20 个)随机选取 c_1 和 c_2 作为初始的神经元节点，当一个新的数据点 $\phi(x_j)$ 到达时，EG-Stream 利用式(3)找到离 $\phi(x_j)$ 最近和次近的神经元节点 s_1(标记为获胜点)和 s_2。如果 $\phi(x_j)$ 到最近的神经元节点 s_1 的距离超过了当前 s_1 的距离阈值 δ_{s1}，则 EG-Stream 算法将这个数据点放入缓冲区(Reservoir)；否则将该数据点归入神经元节点 s_1，并更新获胜神经元 s_1 节点的权值和误差变量，再更新与 s_1 节点直接相连的邻居神经元节点的权值。

类似于 GNG 算法，EG-Stream 算法使用边管理策略(Edge Management Procedure)生成网络图拓扑结构。如果 s_1 和 s_2 相连，则把相连边的年龄设置为 0($age=0$)，否则创建连接 s_1 和 s_2 的边。当 s_1 被标记为获胜点时，将所有与 s_1 相连的边的年龄增加 1，并利用式(4)对边的年龄进行衰减。

$$age=2^{-\lambda_2(t-t_0)} \tag{4}$$

其中，λ_2($\lambda_2>0$)表示随着时间变化的年龄的增长率；t 和 t_0 分别代表当前时刻和这条边的创建时间。如果某条边的年龄 age 大于给定值 $a_{\max}$，则删除这条边。当训练步数到达一个给定的常数 β 时，插入新神经元节点。除了对边的年龄衰减，EG-Stream 算法也根据式(5)对神经元节点的权值进行衰减。

$$weight(c)=\sum_{i=1}^{m}2^{-\lambda_1(t-t_{i0})} \tag{5}$$

其中，m 为 t 时刻神经元节点 c 中包含的数据点个数；λ_1($\lambda_1>0$)代表随着时间变化的权值的延迟率；t 和 t_{i0} 分别代表当前时刻和数据点 x_i 的到达时间。如果神经元节点的权值低于给定值 $weight_{\min}$，则这个节点被认为是过于陈旧的节点，将该节点及其相连的边一并删除。最后，误差变量与 δ 相乘进行更新。EG-Stream 算法如算法 1 所示。

算法 1　EG-Stream 算法

输入：$\mathscr{DS}=\{x_1, x_2,\cdots, x_n\}$

输出：神经元节点集合 $C=\{c_1, c_2, c_3,\cdots\}$ 及其权值 $W=\{w_{c_1}, w_{c_2}, w_{c_3},\cdots\}$

1. 随机选择两个数据点初始化神经元节点 c_1 和 c_2 及它们的权值 w_{c_1} 和 w_{c_2}；
2. for each x_i
3.　　根据式(2)计算 $y_i=\phi(x_i)$；
4.　　G-Stream-Onesample(y_i, C, W)；
5. end for

算法 2　G-Stream-Onesample

输入：y_i，神经元节点集合 $C=\{c_1, c_2, c_3,\cdots\}$ 及其权值 $W=\{w_{c_1}, w_{c_2}, w_{c_3},\cdots\}$

输出：神经元节点集合 $C=\{c_1, c_2, c_3,\cdots\}$ 及其权值 $W=\{w_{c_1}, w_{c_2}, w_{c_3},\cdots\}$

1.　选出距离 y_i 最近的获胜神经元节点 s_1 和次近节点 s_2；

$$s_1=\min_{w_{ci}\in W}(\|y_i-w_{ci}\|)\wedge s_2=\min_{w_{ci}\in W\setminus\{s_1\}}(\|y_i-w_{ci}\|)$$

2.　if $\|y_i-s_1\|>\delta_{s1}$
3.　　将 y_i 送入缓冲区；
4.　　if 缓冲区满
5.　　将缓冲区中的数据点移至数据 $\mathscr{DS}$ 最前面，并立即处理；
6.　　end if
7.　else
8.　　将 y_i 归于节点 s_1 并记录它的时间戳；
9.　　计算神经元节点 s_1 的误差变量：
　　$error(s_1)=error(s_1)+\|w_{s1}-y_i\|^2$
10.　　更新 s_1 节点及其直接拓扑邻居节点的权值：
　　$w_{s1}=w_{s1}+\epsilon_b(y_i-w_{s1})$
　　$w_{sn}=w_{sn}+\epsilon_n(y_i-w_{sn})$
11.　　增长 s_1 节点连接边的年龄，即 age=age+1，并根据式(4)进行衰减；
12.　　if s_1 节点和 s_2 节点有边相连
13.　　设置对应边的年龄 age=0；
14.　　else
15.　　创建这两个节点的连接边，记录时间戳，并设置其年龄 age=0；
16.　　end if
17.　　if 连接边的 age>$age_{\max}$
18.　　删除该连接边；
19.　　end if
20.　　删除没有连接边的节点；
21.　　if 输入节点 y_i 的个数是 β 的整数倍
22.　　/* 按照以下要求创建 3 个新的神经元节点 */
23.　　找到当前误差变量最大的神经元节点 q；
24.　　找到神经元节点 q 的邻居中误差变量最大的节点 f；
25.　　在神经元节点 q 和 f 中间创建一个新的节点 r：
　　$w_r=(w_q+w_f)/2$
26.　　插入连接 r 和 q 以及 r 和 f 的连接边，并删除 q 和 f 的原始连接边；
27.　end if
28.　根据式(5)衰减每个神经元节点，并删除陈旧(weight<$weight_{\min}$)或没有连接边的神经元节点；
29.　更新每个神经元节点的误差变量：error=error*δ
30.　end if

5 实验与结果

为了验证本文所提算法的有效性，在 7 个数据集上将其与现有的数据流聚类算法 G-Stream 进行了比较。实验使用的计算机配置为 Intel Core i5-3470 3.2 GHz 处理器和 16 GB 内存，Windows 7 操作系统，所有比较程序均在 MATLAB R2013a 上设计和运行。

5.1 聚类评价指标

为了对各种聚类算法的精度进行评价，引入了 3 项评价指标[22]：1) Accuracy(Acc)；2) Normalized Mutual Information (NMI)；3) Rand index(RI)。

1) Accuracy(Acc)[30]

$$Acc=\frac{\sum_{i=1}^{K}\frac{|C_i^d|}{|C_i|}}{K}\times 100\% \tag{6}$$

其中，K 表示聚簇个数，$|C_i^d|$ 表示在聚簇 i 中的样本点数，$|C_i|$ 表示聚簇 i 中真实的样本个数。因此，Acc 度量了聚簇的纯度，其取值范围为 0～1，值越大表明聚类精度越高。

2) Normalized Mutual Information(NMI)[31]

归一化互信息 NMI 是一种量化两个分布之间共享统计信息的对称策略。当聚簇标签和真实样本类别一对一映射时，NMI 值到达最大值 1.0。给定真实聚簇 $A=\{A_1,A_2,\cdots,A_k\}$ 和某聚类算法得到的聚簇 $B=\{B_1,B_2,\cdots,B_h\}$，混淆矩阵 $\boldsymbol{C}$ 中的元素 C_{ij} 表示既在 A_i 又在 B_j 中的样本个数。NMI 的计算式如下：

$$NMI(A,B)=\frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B}C_{ij}\log(C_{ij}N/C_{i.}C_{.j})}{\sum_{i=1}^{C_A}C_{i.}\log(C_{i.}/N)+\sum_{j=1}^{C_B}C_{.j}\log(C_{.j}/N)} \tag{7}$$

其中，C_A(C_B)表示 A(B)中样本的个数，$C_{i.}$($C_{.j}$)表示 C 中 i 行元素之和，N 表示样本个数。

3) Rand Index(RI)[32]

RI 比较$\frac{n\times(n-1)}{2}$个数据对，其中 n 为数据集中样本的个数，P_1 和 P_2 为两种聚类算法，n_{11} 为数据对(x_i,x_j)在 P_1 和 P_2 中被划分为同一类的数据对数，n_{00} 则为(x_i,x_j)隶属于不同类的数据对数。RI 错误率的计算式如下：

$$RI=\frac{n_{11}+n_{00}}{C_n^2} \tag{8}$$

由式(8)可得 $RI\in[0,1]$，当 P_1 与 P_2 划分完全一致时，$RI=1$。

5.2 数据集和参数设置

为了对 EG-Stream 算法的聚类有效性进行评价，实验使用了人工数据集和真实数据集，表 2 列出了数据集的相关信息。

DS1，letter4，Sea 和 HyperPlan 数据集为人工模拟数据集。HyperPlan 是一个含有概念漂移的数据流，包含 5 个类共100 000 个样本，每个样本 5 维。KddCup99，CoverType 和 ACT 均来自 UCI。KddCup99 数据集最早来源于 MIT 林肯实验室的一项入侵检测评估项目，记录了 9 周时间内 TCP 网络连接和系统审计数据，用于仿真各种不同的用户类型、网络流量和攻击手段。这些原始数据包含约 500 000 条连接记录的训练集。每个连接记录包含 41 个属性，这些连接记录含 1 种正常的标识类型 normal 和 22 种训练攻击类型，共 23 个类别。CoverType 数据集来源于 US Geological Survey (USGS) 和 US Forest Service (USFS) 对位于 Roosevelt 国家森林的 4 片荒野区域的观测。数据集中包含 581 012 条记录，这些记录最终被分为 7 种类型，每条观测记录包含 54 个地质学和地理学属性。ACT(The Daily and Sports Activities Data Set)数据集包含 45 个传感器在 5 min 内以 25 Hz 的采样频率收集的 19 项活动的数据。为了获得高维数据集，分别将 1 min 和 5 s 的活动数据处理为一个样本，结果得到了大小为 760 * 67 500 (ACT1)和 9 120 * 5 625(ACT2)的数据矩阵。

表 2 数据集

Table 2 Datasets

名称	样本量	样本维数	类别数
DS1[22]	9 153	2	14
letter4[22]	9 344	2	7
Sea[22]	60 000	3	2
HyperPlan[1)]	100 000	10	5
KddCup99[2)]	494 021	41	23
CoverType[3)]	581 012	54	7
ACT2[4)]	9 120	5 625	19
ACT1	760	67 500	19

如算法 1 所示，EG-Stream 算法需要设置一些参数，主要包括：ε_b，ε_n，β，λ_1，λ_2，$|windows|$(滑动窗口样本个数)和$|reservoir|$(缓冲区样本个数)。其中，所有数据集使用 $age_{\max}=250$(边年龄的最大值)，$weight_{\min}=2$(神经元节点权值的最小值)，并且每次插入新节点的个数 $NbNodesInserted=3$，其他参数的设置如表 3 所列。

表 3 参数设置

Table 3 Setting of parameter

Datasets	ε_b	ε_n	β	λ_1	λ_2	$\lvert windows\rvert$	$\lvert reservoir\rvert$
DS1	0.01	0.001	300	0.4	0.4	600	400
letter4	0.01	0.001	300	0.4	0.4	600	400
Sea	0.01	0.001	400	0.2	0.2	1000	300
HyperPlan	0.01	0.001	300	0.2	0.2	600	550
KddCup99	0.01	0.001	300	0.2	0.2	600	400
CoverType	0.01	0.001	300	0.2	0.2	600	400
ACT2	0.01	0.001	300	0.2	0.2	600	400
ACT1	0.01	0.001	60	0.2	0.2	60	50

5.3 聚类精度的比较

本节在给定数据集上进行对比实验。首先，从数据流中到达的前 20 个样本中随机选取 2 个作为初始化的神经元节点，重复实验 10 次并取平均值作为聚类结果。使用 Acc，NMI 和 RI 评价指标对实验进行判断，聚类精度如表 4－表 6 所列。可以看出，欧拉核化 EG-Stream 的聚类性能均优于非

1) http://www.cse.fau.edu/～xqzhu/stream.html

2) http://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

3) https://archive.ics.uci.edu/ml/datasets/Covertype

4) http://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities

核化的 G-Stream，特别是在高维的数据集 ACT1 上，EG-Stream 提升了 2%的性能，验证了欧拉核的有效性。

图 1 给出了 G-Stream 和 EG-Stream 算法的聚类精度随窗口增加的变化情况。一般来说，数据流是无限的，因此我们按照滑动窗口来处理数据流，每个窗口可以表示为 $W[i,j]=(x_i,x_{i+1},\cdots,x_j)$，其中 $i<j$，x_i 表示 i 时刻到达的样本。在当前时刻 t_c，EG-Stream 仅处理最近的 w 个样本，即 $W[t_c-w+1,t_c]$。图 1 中，横坐标 Training Epoch Number 表示滑动窗口数。可以看出，随着滑动窗口数的增加，EG-Stream 得到了比 G-Stream 更好的精度。图 2 给出了两种算法随着滑动窗口变化得到的神经元节点数。图 3 给出了两种算法随着滑动窗口变化得到的 RMS 误差，EG-Stream 算法的误差始终低于 G-Stream 的误差。

表 4　G-Stream 和 EG-Stream 在不同数据集上的 *Acc* 比较

Table 4　Comparison results of EG-Stream and G-Stream in terms of *Acc*

Datasets	G-Stream	EG-Stream
DS1	$0.9660+3.5871*10^{-4}$	$\underline{0.9680+1.0456*10^{-4}}$ ($\alpha=0.0001$)
letter4	$0.9759+2.6119*10^{-4}$	$\underline{0.9868+1.1194*10^{-4}}$ ($\alpha=0.0003$)
Sea	$0.8383+3.1955*10^{-6}$	$\underline{0.8389+4.4716*10^{-6}}$ ($\alpha=0.002$)
HyperPlan	$0.4222+6.6754*10^{-6}$	$\underline{0.4233+2.6593*10^{-6}}$ ($\alpha=0.02$)
KddCup99	$0.9814+1.5916*10^{-5}$	$\underline{0.9816+2.6865*10^{-6}}$ ($\alpha=0.001$)
CoverType	$0.5800+6.6396*10^{-5}$	$\underline{0.6012+4.7396*10^{-5}}$ ($\alpha=0.001$)
ACT2	$0.6691+1.5517*10^{-4}$	$\underline{0.6714+1.5640*10^{-4}}$ ($\alpha=0.0001$)
ACT1	$0.4724+0.0013$	$\underline{0.4964+0.0017}$ ($\alpha=0.001$)

表 5　G-Stream 和 EG-Stream 在不同数据集上的 *NMI* 比较

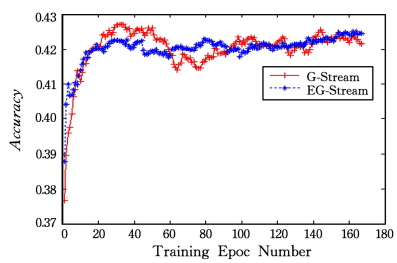
Table 5　Comparison results of EG-Stream and G-Stream in terms of *NMI*

Datasets	G-Stream	EG-Stream
DS1	$0.8208+1.5575*10^{-4}$	$\underline{0.8266+7.3710*10^{-5}}$ ($\alpha=0.0001$)
letter4	$0.6253+1.3868*10^{-4}$	$\underline{0.6410+1.1683*10^{-4}}$ ($\alpha=0.0003$)
Sea	$0.1374+5.0889*10^{-7}$	$\underline{0.1381+6.1428*10^{-7}}$ ($\alpha=0.002$)
HyperPlan	$0.0177+9.1363*10^{-7}$	$\underline{0.0183+2.0142*10^{-7}}$ ($\alpha=0.02$)
KddCup99	$0.6656+4.5222*10^{-4}$	$\underline{0.6857+9.8330*10^{-4}}$ ($\alpha=0.0001$)
CoverType	$0.1222+4.6306*10^{-5}$	$\underline{0.1291+1.2616*10^{-5}}$ ($\alpha=0.001$)
ACT2	$0.6093+4.0498*10^{-5}$	$\underline{0.6104+4.2298*10^{-5}}$ ($\alpha=0.0001$)
ACT1	$0.5664+2.6178*10^{-5}$	$\underline{0.5808+5.4523*10^{-4}}$ ($\alpha=0.001$)

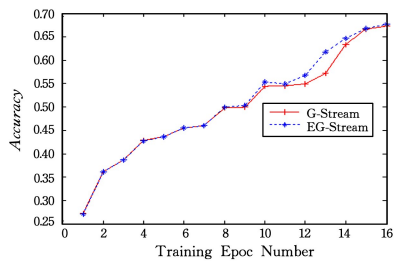
表 6　G-Stream 和 EG-Stream 在不同数据集上的 *RI* 比较

Table 6　Comparison results of EG-Stream and G-Stream in terms of *RI*

Datasets	G-Stream	EG-Stream
DS1	$0.9720+7.4450*10^{-6}$	$\underline{0.9729+3.8123*10^{-6}}$ ($\alpha=0.0001$)
letter4	$0.8154+9.0060*10^{-6}$	$\underline{0.8194+8.5060*10^{-6}}$ ($\alpha=0.0003$)
Sea	$0.4707+1.3053*10^{-8}$	$\underline{0.4707+8.5831*10^{-9}}$ ($\alpha=0.002$)
HyperPlan	$0.7043+3.6652*10^{-8}$	$\underline{0.7043+2.9010*10^{-8}}$ ($\alpha=0.02$)
KddCup99	$0.8375+0.0015$	$\underline{0.8601+0.0019}$ ($\alpha=0.0001$)
CoverType	$0.6225+6.5539*10^{-8}$	$\underline{0.6226+4.0416*10^{-8}}$ ($\alpha=0.001$)
ACT2	$0.9482+3.4593*10^{-7}$	$\underline{0.9484+4.2011*10^{-7}}$ ($\alpha=0.0001$)
ACT1	$0.9082+2.8901*10^{-4}$	$\underline{0.9134+1.8685*10^{-4}}$ ($\alpha=0.001$)



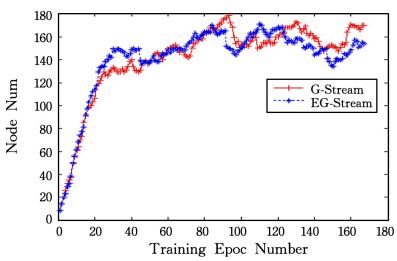
(a) HyperPlan



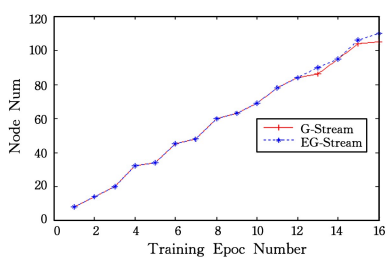
(b) ACT2

图 1　G-Stream 和 EG-Sream 聚类精度的比较

Fig. 1 Comparison of accuracy for G-Stream and EG-Stream



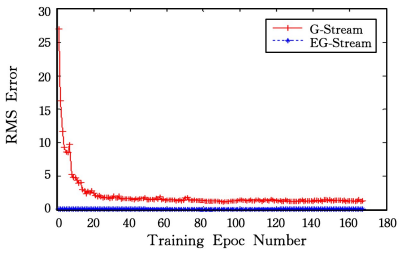
(a) HyperPlan



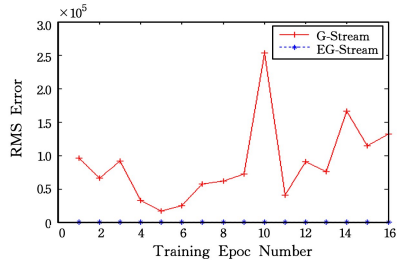
(b) ACT2

图 2　G-Stream 和 EG-Sream 生成的神经元节点数的比较

Fig. 2　Number of nodes created by G-Stream and EG-Stream



(a) HyperPlan



(b) ACT2

图 3 G-Stream 和 EG-Stream 误差的比较

Fig. 3 RMS error for G-Stream and EG-Stream

5.4 滑动窗口大小对算法的影响

本节讨论滑动窗口的重叠比对 EG-Stream 算法的影响。对于 t 时刻长度为 Q 的滑动窗口 $W[t-Q+1,t]$，其样本点可以表示为 $(x_1,x_2,\cdots,x_Q)$，x_i 为一个 d 维数据。随着样本点的陆续到达，两个连续的窗口 $window_t$ 和 $window_{t+1}$ 可以分别表示为：

$$Data^{window_t}=x_1,x_2,\cdots,x_{M-1},x_M,x_{M+1},x_{M+2},\cdots x_{Q-1},x_Q$$

$$Data^{window_{t+1}}=x_{M+1},x_{M+2},\cdots x_{Q-1},x_Q,x_{Q+1},\cdots,x_{Q+M-1},x_{Q+M}$$

这意味着 M 个旧的样本点离开了，M 个新的样本点进来了。

基于这两个滑动窗口，我们定义重叠比为 $M/Q(\%)$。表 7－表 9 列出了重叠比为 0%，25%，50%，75% 时 EG-Stream 算法的聚类结果。从表 7 中可以看出，随着重叠比的增加，聚类精度 Acc 普遍提高，特别是 EG-Stream 算法在大数据集 Kddcup99 和 CoverType 上的表现尤其明显。表 8－表 9 有类似的结果。

表 7 滑动窗口重叠比改变后 G-Stream 算法和 EG-Stream 算法的 Acc 比较

Table 7 Acc of G-Stream and EG-Stream while changing the overlap percentage of sliding windows

Datasets		0	25%	50%	75%
DS1	G-Stream	0.9659	0.9500	0.9795	0.9657
	EG-Stream(α=0.0001)	0.9786	0.9761	0.9804	0.9839
letter4	G-Stream	0.9480	0.9233	0.9860	0.9866
	EG-Stream(α=0.0003)	0.9893	0.9614	0.9903	0.9967
Sea	G-Stream	0.8416	0.8398	0.8364	0.8374
	EG-Stream(α=0.002)	0.8389	0.8388	0.8373	0.8413
HyperPlan	G-Stream	0.4258	0.4254	0.4230	0.4302
	EG-Stream(α=0.02)	0.4213	0.4197	0.4241	0.4333
KddCup99	G-Stream	0.9821	0.9821	0.9851	0.9827
	EG-Stream(α=0.0001)	0.9825	0.9852	0.9859	0.9838
CoverType	G-Stream	0.5824	0.5698	0.5664	0.5833
	EG-Stream(α=0.001)	0.5995	0.5908	0.6005	0.5906
ACT2	G-Stream	0.6817	0.6948	0.7155	0.6328
	EG-Stream(α=0.0001)	0.6764	0.6948	0.7152	0.6328
ACT1	G-Stream	0.4592	0.5434	0.5816	0.5289
	EG-Stream(α=0.001)	0.4816	0.5816	0.5805	0.5276

表 8 滑动窗口重叠比改变后 G-Stream 算法和 EG-Stream 算法的 NMI 比较

Table 8 NMI of G-Stream and EG-Stream while changing the overlap percentage of sliding windows

Datasets		0	25%	50%	75%
DS1	G-Stream	0.8187	0.8067	0.8026	0.8002
	EG-Stream(α=0.0001)	0.8193	0.8301	0.8093	0.7979
letter4	G-Stream	0.6041	0.6063	0.6381	0.5953
	EG-Stream(α=0.0003)	0.6354	0.6414	0.6226	0.5963
Sea	G-Stream	0.1361	0.1375	0.1354	0.1372
	EG-Stream(α=0.002)	0.1369	0.1388	0.1360	0.1377
HyperPlan	G-Stream	0.0199	0.0190	0.0172	0.0205
	EG-Stream(α=0.02)	0.0190	0.0177	0.0173	0.0209
KddCup99	G-Stream	0.6747	0.6979	0.6605	0.6482
	EG-Stream(α=0.0001)	0.6579	0.7198	0.6860	0.7149
CoverType	G-Stream	0.1230	0.1157	0.1153	0.1161
	EG-Stream(α=0.001)	0.1299	0.1247	0.1267	0.1297
ACT2	G-Stream	0.6152	0.6084	0.6185	0.5796
	EG-Stream(α=0.0001)	0.6148	0.6084	0.6182	0.5796
ACT1	G-Stream	0.5651	0.6060	0.6215	0.6023
	EG-Stream(α=0.001)	0.5695	0.6217	0.6307	0.6066

表 9 滑动窗口重叠比改变后 G-Stream 算法和 EG-Stream 算法的 RI 比较

Table 9 RI of G-Stream and EG-Stream while changing the overlap percentage of sliding windows

Datasets		0	25%	50%	75%
DS1	G-Stream	0.9708	0.9700	0.9673	0.9677
	EG-Stream(α=0.0001)	0.9701	0.9716	0.9695	0.9681
letter4	G-Stream	0.8112	0.8128	0.8202	0.8091
	EG-Stream(α=0.0003)	0.8160	0.8273	0.8142	0.8091
Sea	G-Stream	0.4703	0.4707	0.4706	0.4705
	EG-Stream(α=0.002)	0.4705	0.4707	0.4704	0.4706
HyperPlan	G-Stream	0.7042	0.7042	0.7044	0.7043
	EG-Stream(α=0.02)	0.7045	0.7040	0.7042	0.7043
KddCup99	G-Stream	0.8331	0.8597	0.8327	0.8265
	EG-Stream(α=0.0001)	0.8277	0.8629	0.8404	0.8620
CoverType	G-Stream	0.6225	0.6218	0.6223	0.6224
	EG-Stream(α=0.001)	0.6226	0.6219	0.6220	0.6219
ACT2	G-Stream	0.9490	0.9488	0.9493	0.9410
	EG-Stream(α=0.0001)	0.9486	0.9488	0.9493	0.9410
ACT1	G-Stream	0.9179	0.9338	0.9380	0.9191
	EG-Stream(α=0.001)	0.9097	0.9367	0.9371	0.9195

5.5 可视化验证

除了上述实验，我们还探索了 EG-Stream 聚类非平稳数据流的能力。在许多真实应用中，数据随着时间慢慢进入，其分布随时都在改变，体现为非平稳性。例如，第一个类的样本点全部到达后，第二个类的样本点再全部到达，依次类推。在这种情况下，旧的概念消失，同时随着数据点的到达，新的概念产生。根据定义 2，此时发生概念漂移。

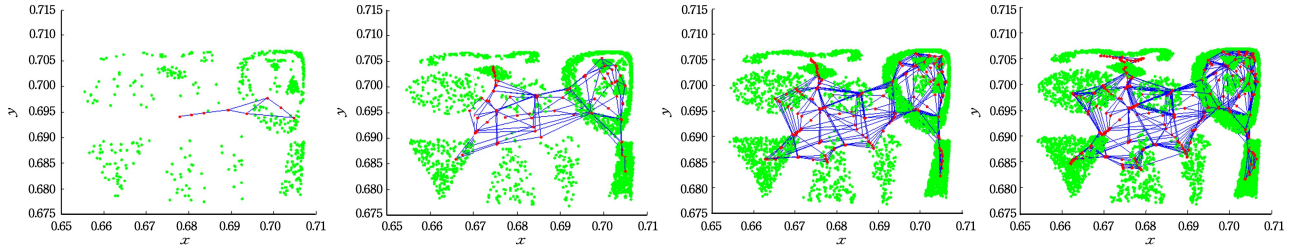
定义 1（概念，Concept） 产生数据流的过程可以被考虑为在随机变量 Y 和 $X=\{X_1,\cdots,X_n\}$ 上的联合分布，其中 $y\in dom(Y)$ 表示类别标签，$x_i\in dom(X_i)$ 表示属性值，$dom(\cdot)$ 表示随机变量的域。

在数据流背景下，概念随时间变化，t 时刻的概念可以表示为 $P_t(X,Y)$。

定义 2（概念漂移，Concept Drift） 当 t 时刻和 u 时刻的分布发生变化时，概念漂移发生。

$$P_t(X,Y)\neq P_u(X,Y)$$

图 4 和图 5 给出了 EG-Stream 算法在 DS1 和 letter4 数据集上的神经元节点生成过程。第一幅子图表示第一个滑动窗口的样本点到来后的情形，所有样本的 1/3 和 2/3 到达后的情形，最后一幅子图表示所有样本点都到达的情形。这些图说明：EG-Stream 可以准确地识别数据流的结构，并最优分割这些结构信息。

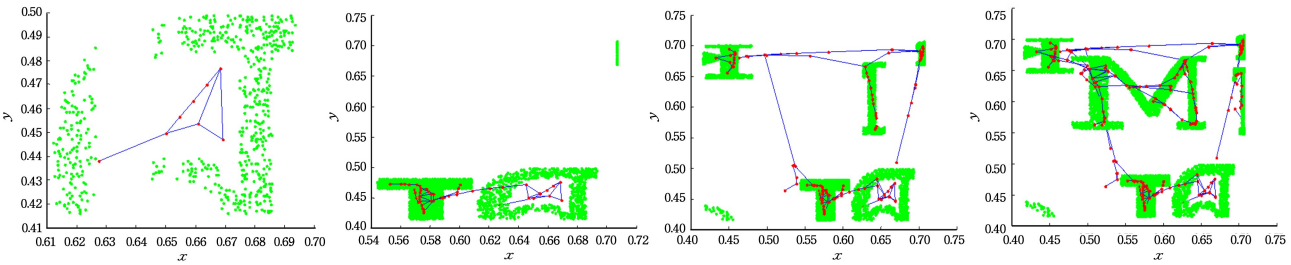


(a)1st windows　(b)After 1/3 of all widows　(c)After 2/3 of all widows　(d)Final Graph

注：红色代表节点，蓝色代表边，绿色代表样本点

图 4　EG-Stream 在 DS1 数据集上神经元节点的生长过程(电子版为彩色)

Fig. 4　Evolution of graph creation of EG-Stream on DS1



(a)1st windows　(b)After 1/3 of all widows　(c)After 2/3 of all widows　(d)Final Graph

注：红色代表节点，蓝色代表边，绿色代表样本点

图 5　EG-Stream 在 letter4 数据集上神经元节点的生长过程(电子版为彩色)

Fig. 5　Evolution of graph creation of EG-Stream on letter4

结束语　本文引入欧拉核数据流聚类算法来解决数据流聚类中无法处理的大规模计算和鲁棒聚类问题。在数据流聚类算法中引入欧拉核。欧拉核不同于已经存在的各种核，欧拉核实现了一个显式而非隐式的 d 维空间到 d 维空间的复数映射，并提供了数据点到聚类中心的鲁棒的 cosine 函数距离度量。显式地映射避免了一般的核聚类算法需要使用核技巧而无法处理数据流的问题。最后，在 UCI 数据集上进行实验，结果证明：基于欧拉核的数据流聚类算法表现出了较好的聚类性能，同时还识别了数据的结构信息。

参 考 文 献

[1] NGUYEN H L, WOON Y K, NG W K. A survey on data stream clustering and classification[J]. Knowledge and information systems, 2015, 45(3): 535-569.

[2] GABER M M, ZASLAVSKY A, KRISHNASWAMY S. Mining data streams: a review[J]. ACM Sigmod Record, 2005, 34(2): 18-26.

[3] GAMA J, RODRIGUESP P. An overview on mining data streams[M]// Foundations of Computational, Intelligence Volume 6. Berlin: Springer, 2009: 29-45.

[4] AGGARWAL C C. Data streams: An overview and scientific applications[M]// Scientific Data Mining and Knowledge Discovery. Berlin: Springer, 2009: 377-397.

[5] SILVA J A, FARIA E R, BARROS R C, et al. Data stream clustering: A survey[J]. ACM Computing Surveys (CSUR), 2013, 46(1): 13.

[6] LI Y, YANG G, HE H, et al. A study of large-scale data clustering based on fuzzy clustering[J]. Soft Computing, 2016, 20(8): 3231-3242.

[7] ZHANG P, SHEN Q. Fuzzy c-means based coincidental link filtering in support of inferring social networks from spatiotemporal data streams[J]. Soft Computing, 2018, 22(21): 7015-7025.

[8] O'CALLAGHAN L, MISHRA N, MEYERSON A, et al. Streaming-data algorithms for high-quality clustering[C]// Proceedings 18th International Conference on Data Engineering. IEEE, 2002: 685-694.

[9] AGGARWAL C C, HAN J, WANG J, et al. A framework for clustering evolving data streams[C]// Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29. VLDB Endowment, 2003: 81-92.

[10] AGGARWAL C C, HAN J, WANG J, et al. A framework for projected clustering of high dimensional data streams[C]// Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30. VLDB Endowment, 2004: 852-863.

[11] ZHOU A, CAO F, QIAN W, et al. Tracking clusters in evolving data streams over sliding windows[J]. Knowledge and Information Systems, 2008, 15(2): 181-214.

[12] UDOMMANETANAKIT K, RAKTHANMANON T, WAIYA-

MAI K. E-stream: Evolution-based technique for stream clustering[C]// International Conference on Advanced Data Mining and Applications. Berlin: Springer, 2007: 605-615.

[13] LÜHR S, LAZARESCU M. Incremental clustering of dynamic data streams using connectivity based representative points[J]. Data & Knowledge Engineering, 2009, 68(1): 1-27.

[14] CAO F, ESTERT M, QIAN W, et al. Density-based clustering over an evolving data stream with noise[C]// Proceedings of the 2006 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2006: 328-339.

[15] TASOULIS D K, ROSS G, ADAMS N M. Visualising the cluster structure of data streams[C]// International Symposium on Intelligent Data Analysis. Berlin: Springer, 2007: 81-92.

[16] KRIEGEL H P, KRÖGER P, NTOUTSI I, et al. Density based subspace clustering over dynamic data[C]// International Conference on Scientific and Statistical Database Management. Berlin: Springer, 2011: 387-404.

[17] CHEN Y, TU L. Density-based clustering for real-time stream data[C]// Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2007: 133-142.

[18] WAN L, NG W K, DANG X H, et al. Density-based clustering of data streams at multiple resolutions[J]. ACM Transactions on Knowledge Discovery from Data (TKDD), 2009, 3(3): 14.

[19] PARK N H, LEE W S. Statistical grid-based clustering over data streams[J]. Acm Sigmod Record, 2004, 33(1): 32-37.

[20] DANG X H, LEE V, NG W K, et al. An EM-based algorithm for clustering data streams in sliding windows[C]// International Conference on Database Systems for Advanced Applications. Berlin: Springer, 2009: 230-235.

[21] SMITH T, ALAHAKOON D. Growing self-organizing map for online continuous clustering[M]// Foundations of Computational Intelligence Volume 4. Berlin: Springer, 2009: 49-83.

[22] GHESMOUNE M, LEBBAH M, AZZAG H. A new growing neural gas for clustering data streams[J]. Neural Networks, 2016, 78: 36-50.

[23] SCHOLKOPF B, SMOLA A, MULLER K R. Nonlinear component analysis as a kernel eigenvalue problem[J]. Neural computation, 1998, 10(5): 1299-1319.

[24] MIKA S, RATSCH G, WESTON J, et al. Fish discriminant analysis with kernels[C]// Proceedings of IEEE Neural Networks for Signal Processing Workshop(NNSP). Madison, WI: IEEE Press, 1999: 41-48.

[25] ZHANG D Q. Kernel-Based Associative Memories, Clustering Algorithms and their Applications[D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2004. (in Chinese)
张道强. 基于核的联想记忆及聚类算法的研究与应用[D]. 南京: 南京航空航天大学, 2004.

[26] ZHANG D Q, CHEN S C. Kernel-Based fuzzy and possibilities C-Means clustering[C]// Proceedings of the 13th International Conference on Artificial Neural Networks. Istanbul, Turkey: Springer, 2003: 122-125.

[27] GIROLAMI M. Mercer-based clustering in feature space[J]. IEEE Transactions on Neural Networks, 2002, 13(3): 780-784.

[28] LIWICKI S, TZIMIROPOULOS G, ZAFEIRIOU S, et al. Euler principal component analysis[J]. International Journal of Computer Vision, 2013, 101(3): 498-518.

[29] WU J S, ZHENG W S, LAI J H. Euler Clustering[C]// Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. 2013: 1792-1798.

[30] CAO F, ESTERT M, QIAN W, et al. Density-based clustering over an evolving data stream with noise[C]// Proceedings of the 2006 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2006: 328-339.

[31] STREHL A, GHOSH J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions[J]. Journal of Machine Learning Research, 2002, 3: 583-617.

[32] RAND W M. Objective criteria for the evaluation of clustering methods[J]. Journal of the American Statistical Association, 1971, 66(336): 846-850.