·先进计算与数据处理· 文章编号:1000-3428(2018)05-0025-08 文献标志码:A 中图分类号:TP391

一种面向轨迹信息的时序数据流异常检测算法

高嘉伟[1,2],刘建敏[1]

(1. 山西大学 计算机信息与技术学院,太原 030006;
2. 计算智能与中文信息处理教育部重点实验室,太原 030006)

摘 要: 针对传统聚类算法多数无法对时序数据流进行聚类的问题,提出一种基于密度和网格的聚类算法。引入动态划分网格的方法,通过当前数据块内数据的特征动态地设置网格划分、网格密度阈值等参数并自适应地生成网格,将其转化为不同类型的图并分别进行聚类。分析某一个体的轨迹,采取按时间段的个体轨迹划分方法检测个体异常轨迹。实验结果表明,该算法可根据用户的需求得到不同时间段内数据的聚类结果,适用于处理轨迹信息等时序数据流的异常检测问题。

关键词: 数据流;数据块;聚类;动态划分;异常检测

中文引用格式: 高嘉伟,刘建敏. 一种面向轨迹信息的时序数据流异常检测算法[J]. 计算机工程,2018,44(5):25-32,46.

英文引用格式: GAO Jiawei, LIU Jianmin. An Anomaly Detection Algorithm for Time-series Data Flow Oriented to Trajectory Information[J]. Computer Engineering,2018,44(5):25-32,46.

An Anomaly Detection Algorithm for Time-series Data Flow Oriented to Trajectory Information

GAO Jiawei[1,2], LIU Jianmin[1]

(1. School of Computer Information and Technology, Shanxi University, Taiyuan 030006, China;
2. Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Taiyuan 030006, China)

【Abstract】 Aiming at the problem that most traditional clustering algorithms cannot cluster time-series data flow, a clustering algorithm based on density and grid is proposed. It introduces the method of dynamic grid partitioning, dynamically set parameters such as grid partition and grid density thresholds and adaptively generates grids through the characteristics of data within the current data block, convert it into different types of maps and performs clustering separately. The trajectory of an individual is analyzed, and the individual trajectory is detected by the individual trajectory division method according to the time period. Experimental results show that the algorithm can obtain the clustering results of data in different time periods according to the user's requirements, and is suitable for the problem of abnormal detection of time-series data flow such as trajectory information.

【Key words】 data flow; data block; clustering; dynamic partitioning; anomaly detection

DOI: 10.19678/j.issn.1000-3428.0046381

0 概述

轨迹信息是一种时序数据流,具有无限性、快速性和无约束性等特点[1]。对轨迹信息进行有效分析,提取数据中的有效信息,及时发现异常轨迹,已成为目前学术界和工业界广泛关注的领域之一。

传统聚类算法无法对轨迹这类时序数据流进行聚类,为此,国内外学者关于时序聚类算法开展了研究并取得了一些成果。Aggarwal 等人于 2003 年提出了一种时序数据流聚类算法——CluStream[2]。该算法通过在线和离线 2 个过程并利用金字塔时间框架对不同时间段内的数据进行聚类。由于该数据流聚类算法不能处理任意形状的簇,因此文献[3]提出 D-stream 聚类算法,其主要思想是将数据空间预先划分成一系列网格,通过将时序数据映射到相应网格,对网格处理得到聚类结果。

然而,D-stream 聚类算法需要用户预先设置较多参数且精度较低。为此,许多学者基于该算法做

基金项目: 国家自然科学基金(61573229,41401521);山西省自然科学基金(201601D202036);2016 年山西省高等学校大学生创新创业训练计划项目(201610108011)。

作者简介: 高嘉伟(1980—),男,讲师、博士研究生,主研方向为机器学习、软件设计;刘建敏,本科生。

收稿日期: 2017-03-15 **修回日期:** 2017-05-17 **E-mail:** 530511416@qq.com

了进一步改进。NDD-stream 算法[3]通过计算网格单元的密度和簇的数目,动态地调整网格密度阈值,有效地避免了用户对密度阈值设置的主观性,但由于该算法仅对稠密网格及其边界的过渡网格或稀疏网格进行聚类,忽视了未处于稠密网格边界的过渡网格和稀疏网格。因此,从聚类结果来看,其聚类精度仍较低。文献[4]考虑到数据的时态特征和空间倾斜分布,定义了时态密度和自适应的密度阈值函数,使得更多的网格参与聚类,提高了算法的聚类精度,但该算法对网格进行了预先划分,由于划分参数的不确定性,使得所划分的网格不能很好地自适应当前数据,影响最终的聚类结果。文献[5]定义了样本分布的局部密度,利用类内密度有序性搜索聚类边界并通过圈定聚类边界来获取聚类结果,但该算法通过不断地改变搜索半径来处理数据分布疏密度不一的问题,增加了算法时间复杂度。此外,这些算法以及之后一些改进的算法[6-9],仅能获取当前时段内数据的聚类结果,而不能根据用户需求获取任意时段内数据的聚类结果。

综上所述,目前关于时序数据的密度聚类算法仍存在 3 类问题:关于解决数据分布疏密不一的方法有待改进;基于网格的聚类方法网格都被预先划分;未能有效地获取任意时段内数据的聚类结果。

此外关于轨迹异常检测方法已有很多,比如文献[10-11]提出对出租车异常轨迹检测的算法,其主要特征是按区域对轨迹划分,对区域内的所有个体的轨迹进行分析,但针对某一个体的轨迹,该算法并未给予有效的异常检测的方法。

本文从个体的轨迹为着眼点,针对以上3 类问题对已有的时序数据流聚类算法进行改进,提出一种基于密度和网格的聚类算法,并将其应用于时序数据流异常检测。

1 基本概念

1.1 定义

定义 1(数据单元) 数据单元即拥有 d 个属性值的数据,任意一个数据单元 $x_i=(a_{i1},a_{i2},\cdots,a_{id})$[3]。

定义 2(数据流) 数据流即按一定时间顺序到达的数据单元的集合,数据流 $X=(x_1,x_2,\cdots,x_i,\cdots,x_N)$[12]。

1.2 密度网格

由于数据流具有无限性,如果对数据一一处理,时间复杂度较高。为此,可以通过将数据空间划分成多个子数据空间,先将数据映射到相应的某个数据子空间中,之后再对所有数据子空间进行处理,从而提高数据处理效率。

对于拥有 d 维属性的数据流,每个数据 $x_i=(a_{i1},a_{i2},\cdots,a_{id})$ 都可存储到相应的 d 维数据空间 $S=S_1\times S_2\times\cdots\times S_d$。

定义 3(网格单元) 在数据空间 S 中,对任意一个网格 g_u,选取每一维的划分为 $H_{g_uj}(1\leqslant j\leqslant d)$,则对于任意一个网格单元 $g_u=H_{g_u1}\times H_{g_u2}\times\cdots\times H_{g_ud}$[3]。

定义 4(网格群) 设数据块内所有数据单元形成了 R 个网格,则由这 R 个网格形成了一个网格群 $G=(g_1,g_2,\cdots,g_R)$。

网格的位置由生成网格的数据单元所决定,网格之间可能会存在重叠的现象,因此,本文定义了邻近网格。

定义 5(邻近网格) 对于网格群内任意 2 个网格 g_o 和 g_w,若 $g_o\cap g_w\neq\varnothing$,$g_o\not\subset g_w$ 且 $g_w\not\subset g_o$,则 g_o 与 g_w 为邻近网格,表示为 $g_o\sim g_w$。

定义 6(网格单元密度) 设任意一网格单元 g_u 在时刻 t,网格内数据的个数为 r_u,则此时网格的密度为[3]:

$$D_{g_u}(t)=r_u \tag{1}$$

即网格单元密度等于网格内数据的个数。

定义 7(网格单元中心点) 对于任意一个网格 g_u,为其定义一个中心点 $foucus_{g_u}=(\varphi_{g_u1},\varphi_{g_u2},\cdots,\varphi_{g_uj},\cdots,\varphi_{g_ud})$。其中,$\varphi_{g_uj}$ 表示 g_u 网格的中心点在 j 维的属性值,其数值上等于网格内所有数据单元在该维属性上取值的均值,即:

$$\varphi_{g_uj}=\frac{\sum_{z=1}^{r_u}a_{zj}}{r_u},1\leqslant j\leqslant d \tag{2}$$

其中,r_u 表示网格单元 g_u 内数据单元的个数。

定义 8(网格单元结构体) 网格单元结构体有 3 个变量和 2 个数组,用来存储该网格单元内数据的概要信息。当数据单元 x_i 映射到某一网格时,更新该网格单元结构体。设某一时刻某一网格单元 g_u,其结构体表示为:

$$\begin{aligned}GSTR_{g_u}=\{&D_{g_u},lable_{g_u},focus_{g_u},\\&\{\min\{a_{zj}\}\mid 1\leqslant j\leqslant r_u,1\leqslant z\leqslant d\},\\&\{\max\{a_{zj}\}\mid 1\leqslant j\leqslant r_u,1\leqslant z\leqslant d\}\}\end{aligned} \tag{3}$$

其中,D_{g_u}、$lable_{g_u}$、$focus_{g_u}$、$\{\min\{a_{zj}\}\mid 1\leqslant j\leqslant r_u,1\leqslant z\leqslant d\}$、$\{\max\{a_{zj}\}\mid 1\leqslant j\leqslant r_u,1\leqslant z\leqslant d\}$ 分别表示某一时刻网格单元 g_u 的密度、类别、中心点与由该网格内数据在每一个属性上取值的最小值和最大值所构成的数组。

1.3 网格单元密度阈值

为解决数据分布密度不一的问题,本文引入了网格密度阈值,用于生成不同类型的网格。网格密度阈值参数的取值,对算法中格簇的形成以及聚类的结果有较大的影响。为此,本文定义了数据块,通过获取数据块内所有网格的特征信息,采用文献[1]提出的平均密度的思想,动态地确定网格单元密度阈值。

定义 9(数据块) 数据块用于暂存某时间段内的数据单元,其长度为 $n(1\leqslant n\leqslant N)$。

当数据块内数据单元的个数等于 n 时,即数据块达到饱和状态,对数据块内的 n 个数据单元进行处理并清空数据块内的数据单元,数据块继续接收新的数据单元,以此类推。从数据单元 x_ρ 开始,数据块 $Xb=(x_\rho,x_{\rho+1},\cdots,x_{\rho+n-1})$,其中,$\rho=1+n\lambda$,$0\leqslant\lambda\leqslant\lceil\frac{N-n}{n}\rceil$,且 λ 为非负整数。

以数据块为数据处理单元,将数据块内所有的数据单元映射到网格中,设在某一时刻 t 由数据块内数据单元所形成的所有网格的平均密度为 $D_{\text{avg}}(t)$,网格最小密度为 $D_{\min}(t)$,网格最大密度为 $D_{\max}(t)$:

$$D_{\text{avg}}(t)=\frac{\sum_{u=1}^{R}D_{g_u}(t)}{R} \tag{4}$$

其中,$D_{g_u}(t)$表示 t 时刻第 u 个网格的密度,R 表示该数据块内网格的数量。

稠密网格阈值为[1]:

$$DG_{\text{Th}}(t)=\frac{D_{\text{avg}}(t)+D_{\max}(t)}{2} \tag{5}$$

稀疏网格阈值为[1]:

$$SG_{\text{Th}}(t)=\frac{D_{\text{avg}}(t)+D_{\min}(t)}{2} \tag{6}$$

则在时刻 t,对于任意一个网格单元 g_u,其密度为 $D_{g_u}(t)$,有如下定义:

稠密网格:

$D_{g_u}(t)\geqslant DG_{\text{Th}}(t)$

过渡网格:

$SG_{\text{Th}}(t)<D_{g_u}(t)<DG_{\text{Th}}(t)$

稀疏网格:

$D_{g_u}(t)\leqslant SG_{\text{Th}}(t)$

1.4　格簇

由于本文采用网格划分数据空间的方法,将数据单元先映射到网格中,因此可以通过将有联系的网格聚在一起形成多个格簇,从而得到聚类结果。

为了确定各网格之间的联系,本文引用了图论中图的概念。对于一个网格群 $G=(g_1,g_2,\cdots,g_R)$,将网格群中的所有网格的中心点作为图中的顶点。若该网格群内的某 2 个网格邻近,则认为这2 个网格具有一定的联系并为其对应的顶点赋予一条边,边的权重为网格中心点间的距离,从而形成了一个无向有权图。

定义 10(格簇) 通过采用深度优先算法对所形成的无向有权图进行遍历并获取对应的连通分支,将该过程所产生的连通分支定义为格簇,连通分支的个数即为格簇的个数。

定义 11(格簇关键点) 对于任意一个格簇 k_f,将格簇中密度最大网格的中心点定义为该格簇的关键点 $keyp_{k_f}$,$keyp_{k_f}=(\varphi_{k_f1},\varphi_{k_f2},\cdots,\varphi_{k_fj},\cdots,\varphi_{k_fd})$,其中 φ_{k_fj}表示格簇 k_f的第 j 个属性平均值。

定义 12(稠密格簇、过渡格簇、稀疏格簇) 稠密格簇为由稠密网格所形成的连通图的点的集合;过渡格簇为由过渡网格所形成的连通图的点的集合;稀疏格簇为由稀疏网格所形成的连通图的点的集合。

定义 13(格簇阈值) 格簇阈值即为该格簇可容纳其他格簇内数据点的最大邻域半径。对于不同类型的格簇,其阈值的定义也不同。设某一格簇 k_f,其关键点为 $keyp_{k_f}$且有 v 个网格,则该格簇内所有网格的中心点到该格簇关键点距离的最大值 $e_{\max}$、平均值 e_{avg}和最小值 $e_{\min}$为:

$$e_{\max}=\max\left\{\sqrt{\sum_{j=1}^{d}(\varphi_{g_1j}-\varphi_{k_fj})^2},\sqrt{\sum_{j=1}^{d}(\varphi_{g_2j}-\varphi_{k_fj})^2},\cdots,\sqrt{\sum_{j=1}^{d}(\varphi_{g_vj}-\varphi_{k_fj})^2}\right\} \tag{7}$$

$$e_{\text{avg}}=\frac{1}{v}\sum_{u=1}^{v}\sqrt{\sum_{j=1}^{d}(\varphi_{g_uj}-\varphi_{k_fj})^2} \tag{8}$$

$$e_{\min}=\min\left\{\sqrt{\sum_{j=1}^{d}(\varphi_{g_1j}-\varphi_{k_fj})^2},\sqrt{\sum_{j=1}^{d}(\varphi_{g_2j}-\varphi_{k_fj})^2},\cdots,\sqrt{\sum_{j=1}^{d}(\varphi_{g_vj}-\varphi_{k_fj})^2}\right\} \tag{9}$$

在式(7)~式(9)中,φ_{g_uj}表示网格 g_u 中心点的第 j 个属性平均值,φ_{k_fj}表示格簇 k_f的第 j 个属性平均值。

本文所有的距离均采用欧式距离,下文不再赘述。

对于任意一个格簇 k_f,其阈值 $Th(k_f)$为:

$$Th(k_f)=\begin{cases}e_{\max},\text{格簇 }k_f\text{ 为稠密格簇}\\(e_{\max}+e_{\text{avg}})/2,\text{格簇 }k_f\text{ 为过渡格簇}\\(e_{\min}+e_{\text{avg}})/2,\text{格簇 }k_f\text{ 为稀疏格簇}\end{cases} \tag{10}$$

为了判断某 2 个格簇内数据点的分布形态变化是否相同,采用文献[5]判断局部散布形态是否发生突变的方法。首先通过主成分分析提取每个格簇内数据点的特征,然后利用最大特征值和次大特征值的比值来确定该格簇内数据点的分布形态变化,并引入格簇内数据点分布形态相似阈值 $\beta(\beta\geqslant1)$来判断某 2 个格簇内数据点的分布形态变化是否相同。对于一个格簇集合 $K=(k_1,k_2,\cdots,k_L)$内的任意2 个格簇 k_p 和 k_q,k_p 内数据点的分布形态变化为 $change_{k_p}$,关键点为 $keyp_{k_p}$且阈值为 $Th(k_p)$,k_q 内数据点的分布形态变化为 $change_{k_q}$,关键点为 $keyp_{k_q}$且阈值为 $Th(k_q)$。格簇 k_p的关键点 $keyp_{k_p}$与格簇 k_q关键点 $keyp_{k_q}$间的距离为:

$$dist(keyp_{kp},keyp_{kq})=\sqrt{\sum_{j=1}^{d}(\varphi_{k_pj}-\varphi_{k_qj})^2} \quad (11)$$

定义 14(相似格簇) 若格簇 k_p 和 k_q 相似,当且仅当满足式(12)和式(13)2 个条件[5]:

$$\frac{1}{\beta}\leqslant\frac{change_{k_p}}{change_{k_q}}\leqslant\beta \quad (12)$$

$$dist(keyp_{k_p},keyp_{k_q})\leqslant Th(k_p)+Th(k_q) \quad (13)$$

定义 15(格簇结构体) 为每一个格簇构建一个结构体,用来存储格簇的概要信息。设某一格簇 k_f,其结构体表示为:

$$LCSTR_{k_f}=\{keyp_{k_f},Th(k_f),lable_{k_f},change_{k_f}\} \quad (14)$$

其中,$keyp_{k_f}$、$Th(k_f)$、$lable_{k_f}$、$change_{k_f}$分别表示格簇的关键点、格簇的阈值、格簇的类型和格簇内点集分布的形态变化。

2 本文算法

针对前文所述目前关于时序数据的密度聚类算法仍存在 3 类问题,本文提出一种基于密度和网格的聚类算法,并应用于时序数据流异常检测问题。针对现有时序数据流聚类算法存在的问题和异常轨迹检测算法的局限性,本文主要针对某一个体的轨迹进行分析并提出了一种基于密度和网格的聚类算法。采取按时间段对个体轨迹划分的方法对个体异常轨迹进行检测,通过获取当前数据块内数据的特征动态地设置网格密度阈值、网格划分等参数。针对目前时序数据流存在的问题,本文算法做了如下改进:针对数据分布密度不一的问题,通过对稠密网格、过渡网格和稀疏网格形成的稠密图、过渡图和稀疏图分别进行聚类,并利用自适应阈值对所得结果进行再次聚类。针对网格被预先划分的问题,本文算法引入了动态生成网格的方法。根据当前在线处理过程内数据点的特征,以某些数据点作为网格中心,自适应地重新生成网格。针对获取任意时段内数据的聚类结果,通过获取该段时间内的格簇进行再次聚类,并利用自适应阈值对所得结果进行再次聚类,得到最终的聚类结果。

2.1 设计方法

在传统的异常检测算法中,训练集中必须要有异常轨迹参与训练,否则学习器无法识别异常轨迹。然而,提前获取用户异常轨迹信息可能比较困难,因而本文采用聚类思想解决该问题。由于用户的轨迹信息是一种时序数据流,因此本文算法采用时序数据聚类算法,根据用户的运动规律将训练的轨迹信息划分时间段,并对相应时间段内的轨迹数据进行聚类,其聚类结果作为该用户的训练集。测试轨迹首先对相应时间段内的轨迹信息进行聚类,得到聚类结果作为测试集。若测试集中存在某一类与训练集中的任何一类都不相似,则认为该测试轨迹异常。

2.2 时序数据聚类

该聚类分为在线处理数据和离线处理数据 2 个处理过程。在线处理数据过程对数据块内的数据进行处理,动态生成网格并将数据一一映射到相应的网格中,通过使用密度阈值对在线算法所产生的网格分成稠密网格、过渡网格和稀疏网格。根据对稠密网格、过渡网格和稀疏网格不同的处理方法对其进行处理,得到格簇集合并将所有的格簇集合存入数据库中。离线处理数据过程,根据用户的需求,取出某时间段内的格簇集合,通过判断任意 2 个格簇是否相似进行再次聚类,得到最终的聚类结果。

2.2.1 动态生成网格单元方法

传统的固定划分网格方法不能适应当前数据的特征,且网格一旦划分就不会改变。为了避免这种缺陷,本文引入了动态生成网格的方法,根据当前数据块的特征自适应地生成网格,且网格的划分会随着数据块的变化而改变。

当数据块已满即数据块内有 n 个数据单元时,采集该数据块内数据单元特征,通过计算数据单元每一维属性的均值 δ_j,动态地确定网格单元每一维空间划分的长度 h_j。每一维属性的均值 δ_j 为:

$$\delta_j=\frac{\sum_{i=1}^{n}a_{ij}}{n} \quad (15)$$

其中,a_{ij}表示第 i 个数据单元的第 j 个属性值。

网格单元 g_u每一维空间的划分长度 h_j为:

$$h_j=\frac{\sqrt{\sum_{i=1}^{n}(a_{ij}-\delta_j)^2}}{n} \quad (16)$$

根据数据块内数据单元的特征,选取 $R(1\leqslant R\leqslant n)$个数据单元作为网格的中心生成 R 个网格,对于任意一个网格 $g_u(1\leqslant u\leqslant R)$在每一维上的划分 H_{g_uj}表示为:

$$H_{g_uj}=[\omega_{uj}-h_j/2,\omega_{uj}+h_j/2] \quad (17)$$

其中,ω_{uj}表示所选取的第 u 个数据单元的第 j 个属性值,则所生成的网格单元可以表示为:

$$g_u=H_{g_u1}\times H_{g_u2}\times\cdots\times H_{g_ud}=\prod_{j=1}^{d}[\omega_{uj}-h_j/2,\omega_{uj}+h_j/2] \quad (18)$$

2.2.2 各类网格的处理方法

网格的处理方法主要有以下 2 种方法:

1)稠密网格处理方法

针对稠密网格,以网格中的中心点作为顶点,若任意 2 个网格相邻,则为其赋予一条边生成稠密无向有权图,通过深度优先算法获取有权图的连通分支从而得到稠密格簇集合 DC(Dense Cluster)。

2)过渡网格和稀疏网格处理方法

针对过渡网格,需先计算稠密格簇中每个稠密网格中所有点距该稠密网格中心点距离的标准差。设任意一个网格 $g_m(1\leqslant m\leqslant R)$,其标准差为:

$$SD_{g_m}=\frac{\sum_{z=1}^{r_m}\sqrt{\sum_{j=1}^{d}(a_{zj}-\varphi_{g_{mj}})^2}}{r_m} \tag{19}$$

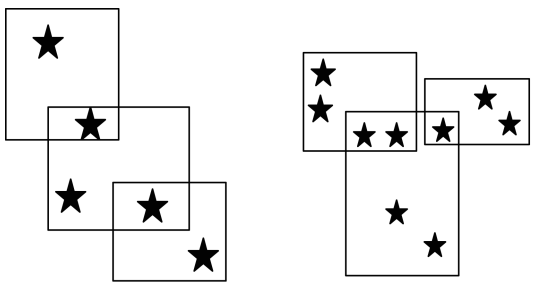
其中,r_m 为该稠密网格内数据单元的个数,a_{zj} 为第 z 个数据单元的第 j 个属性值,$\varphi_{g_{mj}}$ 为该网格中心点的第 j 个属性值,若过渡网格 g_l 的中心点与某一稠密网格的中心点的距离小于标准差 SDg_m,则将该过渡网格 g_l 归入该稠密网格所属的格簇中,并将 g_l 从过渡网格集合移除。继续计算归入稠密格簇的过渡网格的标准差 SDg_l,若剩余的某个过渡网格的中心点与归入稠密格簇的过渡网格的中心点的距离小于过渡网格 g_l 的标准差 SDg_l,则将该过渡网格归入过渡网格 g_l 所属的稠密格簇中,直到剩余的任意一个过渡网格都不可归入某个稠密格簇为止。之后,对剩余的过渡网格采用和处理稠密网格相同的方法生成过渡无向有权图,并得到相应的过渡格簇集合 TC(Transition Cluster)。对稀疏格簇采用与过渡网格相同的处理方法得到稀疏格簇集合 SC(Sparse Cluster)。

2.2.3　噪声的处理

在对用户的轨迹信息进行训练获取样本集时,可能会有异常轨迹参与训练。这些异常轨迹对于后续分析是很宝贵的资源,而在传统聚类方法中往往将其标注为噪声予以舍弃。因而本文将噪声数据独立生成异常类进行处理。

在时刻 t,若某个格簇内所有网格的数据个数之和依然小于稀疏网格阈值 $SG_{\mathrm{Th}}(t)$,则认为该格簇内所有的数据点为噪声,将该格簇从所生成的格簇集合中分离,并将其标记为异常类。

例如,不妨设 $SG_{\mathrm{Th}}(t)=8$,某 2 个格簇都包含 3 个网格,在这 2 个格簇中,一个格簇内所有数据点为噪声,而另一个格簇内所有数据点为非噪声的情况,噪声判别情况如图 1 所示。



(a)所有数据点为噪声　　(b)所有数据点为非噪声

图 1　噪声判别情况

由图 1(a)可知,图稀疏格簇内数据点的个数之和为 5,小于稀疏网格阈值 $SG_{\mathrm{Th}}(t)$,所以该格簇内所有的数据点为噪声;图 1(b)格簇内数据点之和为 9,大于稀疏网格阈值 $SG_{\mathrm{Th}}(t)$,因而该格簇内的数据为非噪声点。

通过分离最后得到 2 种格簇集合,即正常轨迹格簇集合 $N_{NLC}=(k_1,k_2,\cdots,k_{NL})$ 和异常轨迹格簇集合 $A_{\mathrm{ABNLC}}=(k_{AB1},k_{AB2},\cdots,k_{ABNL})$。

2.2.4　类相似

类实际上是格簇集合,那么对于任意 2 个类 C_P 和 C_Q,设 C_P 有 b_P 个格簇,C_Q 有 b_Q 个格簇,则 $C_P=(k_1,k_2,\cdots,k_{b_P})$,$C_Q=(k_1,k_2,\cdots,k_{b_Q})$。类 C_P 的中心点为 $c_P=(\alpha_{c_P1},\alpha_{c_P2},\cdots,\alpha_{c_Pd})$,类 C_Q 的中心点为 $c_Q=(\alpha_{c_Q1},\alpha_{c_Q2},\cdots,\alpha_{c_Qd})$。

$$\alpha_{C_Pj}=\frac{\sum_{z=1}^{b_P}\varphi_{k_zj}}{b_P},1\leqslant j\leqslant d \tag{20}$$

$$\alpha_{C_Qj}=\frac{\sum_{z=1}^{b_Q}\varphi_{C_Qj}}{b_Q},1\leqslant j\leqslant d \tag{21}$$

设类 C_P 和 C_Q 的距离为:

$$d_{\mathrm{cen}}(C_P,C_Q)=dist(c_P,c_Q)=\sqrt{\sum_{j=1}^{d}(\alpha_{C_Pj}-\alpha_{C_Qj})^2} \tag{22}$$

其中,式(20)中的 α_{C_Pj} 表示类 C_P 中心点 c_P 的第 j 个属性值,式(21)中的 α_{C_Qj} 表示类 C_Q 中心点 c_Q 的第 j 个属性值。

类间距离归一化处理:

设所求的类间距离最大值为 $DC_{\max}$,最小距离为 $DC_{\min}$。

$$D_{\mathrm{cen}}(C_P,C_Q)=\frac{d_{\mathrm{cen}}(C_P,C_Q)-DC_{\min}}{DC_{\max}-DC_{\min}} \tag{23}$$

设类间相似阈值为 $\mu(0<\mu<1)$,若 $D_{\mathrm{cen}}(C_P,C_Q)\leqslant\mu$,则认为这 2 个类相似;否则不相似。

2.2.5　在线处理数据过程描述

在线处理数据过程如下:

输入　数据块 $Xb=(x_\rho,x_{\rho+1},\cdots,x_{\rho+n-1})$

输出　格簇集合 $K=(k_1,k_2,\cdots,k_L)$

步骤 1　获取数据块 Xb 中的数据单元。

步骤 2　计算数据单元每一维度的平均值 δ_j 和维度划分的长度 h_j,并确定每个网格空间。

步骤 3　读入新数据单元 $x_v(i\leqslant v\leqslant i+n-1)$,如果数据 $x_v(i\leqslant v\leqslant i+n-1)$ 属于当前某一网格则该网格密度加 1;否则以数据单元 $x_v(i\leqslant v\leqslant i+n-1)$ 为中心生成一个新的网格。

步骤 4　计算时刻 t 稠密网格阈值 $DG_{\mathrm{Th}}(t)$ 和稀疏网格的阈值 $SG_{\mathrm{Th}}(t)$。

步骤 5　判断网格是稠密网格、过渡网格或稀疏网格。

步骤 6　对稠密网格、过渡网格和稀疏网格分别进行处理,并得到稠密格簇集合 DC,过渡格簇集合 TC 和稀疏格簇集合 SC。

步骤 7　输出格簇集合 $DC\cup TC\cup SC$。

2.2.6　离线处理数据过程描述

在该过程处理前,先根据用户所确定的时间段 T,从数据库中获取该时段内的格簇集合。

输入 时间段 T 内的格簇集合 $K=(k_1,k_2,\cdots,k_L)$,形态相似阈值 β

输出 聚类结果 $C=NC\cup ABNC$

步骤 1 确定每一个格簇的关键点。

步骤 2 判断格簇内的数据是否为噪声点,并得到相应的正常轨迹格簇 NLC 和异常轨迹格簇 $ABNLC$。

步骤 3 根据格簇相似的方法,对 NLC 和 $ABNLC$ 分别进行处理,将相似的格簇归为一类并得到正常轨迹类 NC 和异常轨迹类 $ABNC$。

步骤 4 输出最终的聚类结果 $C=NC\cup ABNC$。

本文聚类算法的流程如图 2 所示。

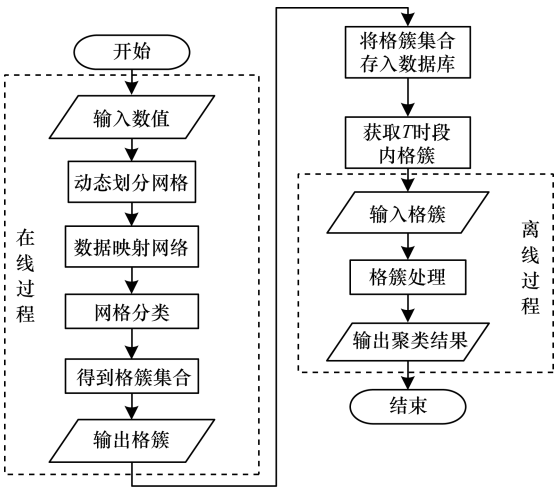


图 2 本文聚类算法流程

2.3 异常检测过程

针对某一用户可能在某段时间内常去一些固定地点,因此其轨迹具有一定的规律。同时,也可能在某段时间内去一些之前未去过的地点。若能及时发现用户在该时间段内出行轨迹的异常,对于用户人身安全防护,尤其是老人、儿童等弱势群体,则具有重要的理论意义和应用价值。

2.3.1 轨迹异常

若测试轨迹中的某一类与训练集中正常轨迹类的任何一类都不相似,则认为该测试轨迹异常。当测试轨迹中的某一类不仅与训练集中正常轨迹中的某类相似,而且也与异常轨迹中的某类相似时,若与异常轨迹类的距离小于正常轨迹类的距离,则认为该测试轨迹异常。

2.3.2 过程描述

异常检测过程如下:

输入 某段时间内的数据流 $X=(x_1,x_2,\cdots,x_i,\cdots,x_N)$,数据块的长度 n,类间相似阈值 μ

输出 轨迹是否异常

步骤 1 将数据流按数据块长度划分为多个数据块。

步骤 2 利用 2.2 节的聚类方法对每个数据块进行聚类。

步骤 3 对步骤 2 所得的类,利用 2.3.1 节的方法,判断轨迹是否异常。

步骤 4 输出最终判断结果。

3 实验结果与分析

本文实验以微软亚洲研究院 Geolife 项目收集的用户 GPS 轨迹信息[13]为研究对象。该项目收集了 178 位用户从 2007 年 4 月—2011 年 10 月的轨迹信息。该数据集的 GPS 轨迹表示时间戳点的序列,其中每一个包含纬度、经度和海拔高度的信息。

本文实验采用 2.5 GHz 处理器,4 GB 内存。算法利用 Eclipse4.4.2 和 Matlab2014a 编程实现。

由于有些用户的运动规律不明显,因此本文实验从 Geolife 项目中筛选了运动规律较为明显且轨迹信息较为完整的 50 名用户在 100 d 内的轨迹信息,并根据每个用户的运动规律对其轨迹信息按时间段进行划分。

3.1 功能比较

若类间相似阈值设置不合理,则可能会导致最终异常检测的判断结果。在实验过程中发现类间相似阈值与样本集中的经度之差(最大经度值与最小经度值之差)和纬度之差(最大纬度值与最小纬度值之差)2 个变量有关。因此,针对所选出 50 名用户,随机选取 40 名用户的轨迹信息进行了分析,并得到类间相似阈值。

由于类间相似阈值无量纲,而经度之差和纬度之差是有量纲的数据,因此需先对经度之差和纬度之差进行归一化处理。以经度之差为例,其归一化过程如下:

$$\Delta lon=\frac{\Delta Lon-min_{\Delta Lon}}{max_{\Delta Lon}-min_{\Delta Lon}} \tag{24}$$

其中,ΔLon 表示某一用户的实际经度之差,$min_{\Delta Lon}$ 为所有用户经度之差的最小值,$max_{\Delta Lon}$ 为所有用户经度之差的最大值,Δlon 为归一化后的经度之差。

利用 Matlab 拟合工具 cftool 对 40 名用户的类间阈值信息拟合得到类间相似阈值 $\mu(\Delta lon,\Delta lat)$ 与归一化后经度之差 Δlon,纬度之差 Δlat 的关系如下:

$$\begin{aligned}\mu(\Delta lon,\Delta lat)=&-0.048\,61+17.49\Delta lon-20.84\Delta lat-\\&162.7\Delta^2 lon-182.4\Delta lon\Delta lat+\\&516.8\Delta^2 lat-626.5\Delta^3 lon+\\&769\,9\Delta^2 lon\Delta lat-1.25\times\\&10^4\Delta lon\Delta^2 lat+410\,9\Delta^3 lat\end{aligned}$$

类间相似阈值拟合曲线如图 3 所示。

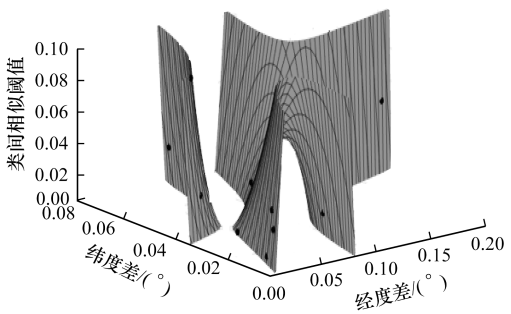


图 3 类间相似阈值拟合曲线

一条拟合曲线往往需要通过一些拟合参数来判断其拟合效果,该曲线的拟合参数值“误差平方和 SSE、相关系数 R-square、调整后的相关系数 Adjusted R-square 和标准差 RMSE”如表 1 所示。

表 1　拟合参数的取值

参数	参数值
SSE	0.000 337
R-square	0.951 100
Adjusted R-square	0.731 200
RMSE	0.012 980

若 SSE 越接近 0,R-square 越接近 1,则曲线拟合效果越好。从表 1 可见,该曲线的拟合参数 R-square 为 0.951 1,较接近于 1,表示该曲线能解释因变量 95.11% 的变化,SSE 误差平方和为 0.000 337,较接近于 0。因此,可以判断该曲线拟合效果较好,可用该曲线的拟合结果来确定类间相似阈值的大小。

本文算法检测异常轨迹情况以用户 4 在 00:00—05:00 时间段内的轨迹信息举例说明。利用用户 4 从 2008-08-05—2008-10-25 在 00:00—05:00 时间段内的数据生成样本。判断用户 4 在 2008-10-27、2008-10-28 和 2008-11-05 的 00:00—05:00 时间段内轨迹是否异常。最后得到用户 4 在 2008-10-27 和 2008-10-28 的 00:00—05:00 时间段内轨迹正常,在 2008-11-05 的 00:00—05:00 时间段内轨迹异常。

用户 4 在 2008-10-27、2008-10-28 该时段内的正常轨迹,以及在 2008-11-05 该时段内的异常轨迹如图 4 ~ 图 7 所示。

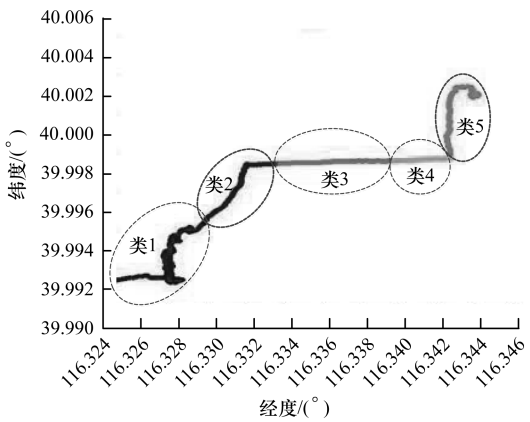


图 4　用户 4 从 2008-09-25—2008-10-25 在 00:00—05:00 时间段内的轨迹样本

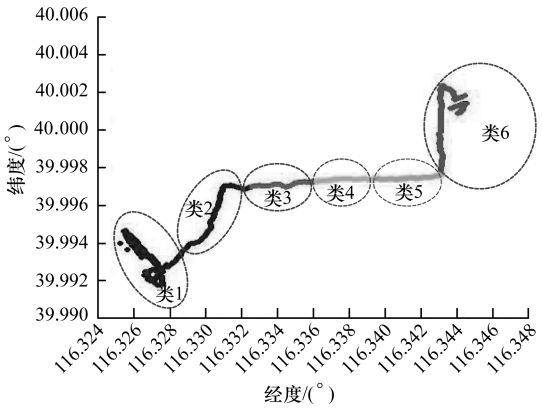


图 5　用户 4 在 2008-10-27 的 00:00—05:00 时间段内的正常轨迹

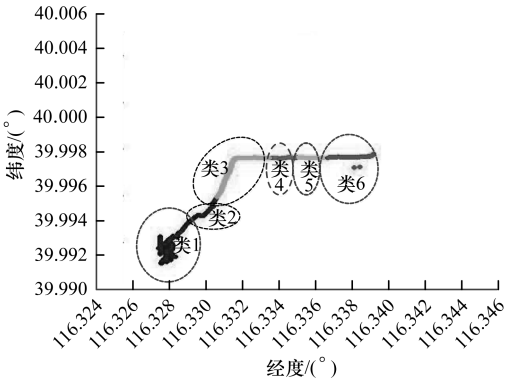


图 6　用户 4 在 2008-10-28 的 00:00—05:00 时间段内的正常轨迹

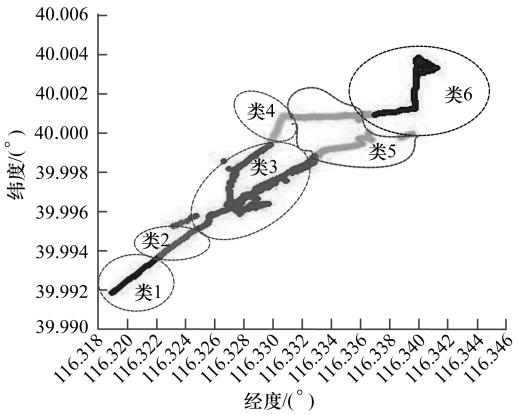


图 7　用户 4 在 2008-11-05 的 00:00—05:00 时间段内的异常轨迹

从图 7 可知,图 7 中有一部分轨迹在图 4 中没有出现,可以认为用户可能去了之前没有去过的地方,该轨迹异常。图 5 的轨迹和图 4 的轨迹相似,可以认为该轨迹正常。图 6 中的轨迹是图 4 轨迹中的一部分,因此该轨迹也正常。

对剩余的 10 名用户进行测试,该 10 名用户在原数据对应的编号见表 2。利用 10 名用户前 80 d 内的数据进行训练得到样本集,并用剩余的 20 d 数据进行测试,通过类间阈值拟合曲线求得每个用户轨迹信息的类间阈值。10 名用户后 20 d 内的轨迹统计情况见表 2。

表 2 10 名用户轨迹统计情况

用户（原数据编号）	实际异常轨迹数	实际正常轨迹数	总轨迹数
1(01)	3	17	20
2(02)	2	18	20
3(08)	10	10	20
4(09)	2	18	20
5(012)	2	18	20
6(013)	3	17	20
7(015)	3	17	20
8(016)	8	12	20
9(017)	4	16	20
10(022)	9	11	20
平均值	5	15	20

各算法检测正确率见表 3。其中文献[4-5]算法的参数值均取该文献所建议的最优参数值，而本文算法参数的取值，通过实验获得最优参数。在表 3 中，TNR 为真负率，ACC 为准确率，其计算方法与文献[14-15]TNR 和 ACC 的计算方法相同。从表 3 中可得，文献[4]检测异常轨迹的平均正确率仅为 38.4%，文献[5]检测异常轨迹的平均正确率为53.5%，本文算法检测异常轨迹的平均正确率为 72.2%，因此本文算法更适合于处理轨迹类数据信息并用来检测异常轨迹。

表 3 各算法检测正确率比较 %

用户	文献[4]算法（$\nu=100$ Kb/s，$\lambda=3$）		文献[5]算法（$r=1$，$\beta=3$）		本文算法（$n=100$，$\beta=3$）	
	TNR	ACC	TNR	ACC	TNR	ACC
1	33.3	80.0	66.7	80.0	100.0	100.0
2	0.0	80.0	50.0	85.0	50.0	95.0
3	40.0	60.0	50.0	75.0	60.0	80.0
4	50.0	75.0	50.0	80.0	50.0	95.0
5	50.0	0.0	50.0	85.0	100.0	100.0
6	33.3	75.0	33.3	75.0	60.0	80.0
7	33.3	70.0	66.7	85.0	66.7	95.0
8	50.0	60.0	62.5	80.0	60.0	80.0
9	50.0	80.0	50.0	80.0	75.0	90.0
10	44.4	55.0	55.6	85.0	100.0	100.0
平均值	38.4	63.5	53.5	81.0	72.2	93.0

3.2 时间效率比较

本文实验针对 10 名测试用户的轨迹信息对文献[4-5]和本文算法的时间效率进行了测试，其结果如表 4 所示。

表 4 算法运行时间比较 s

算法	用户 1	用户 2	用户 3	用户 4	用户 5	用户 6	用户 7	用户 8	用户 9	用户 10
文献[4]算法	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.01	0.01
文献[5]算法	2.99	3.18	3.44	3.85	4.73	4.02	9.92	8.06	5.46	5.14
本文算法	0.20	0.21	0.22	0.24	0.28	0.25	0.46	0.40	0.31	0.29

由表 4 可得，文献[4]的时间效率最高，文献[5]的时间效率最低，而本文算法的时间效率介于两者之间且更接近于文献[4]算法。由实验 3.1 节可知，文献[4]判断轨迹异常的平均 TNR 仅有 38.4%，因此虽然该方法的时间效率高，但仍不适合用于轨迹信息的处理。文献[5]判断轨迹异常的平均 TNR 为 53.5%，虽然比文献[4]高，但其时间复杂度远大于文献[4]和本文算法，因此，该算法也不适合于对轨迹信息的处理。本文算法虽然比文献[4]消耗的时间多，但判断轨迹异常的平均 TNR 为 72.2%，相当于文献[4]的 2 倍。

综上所述，本文算法更适用于处理个体轨迹信息等时序数据流的异常检测问题。

4 结束语

本文提出一种面向轨迹信息的时序数据流异常检测算法，算法主要分为训练和测试 2 个阶段，利用训练阶段得到的样本集并通过类间相似阈值求解方法确定阈值的大小，对测试阶段的测试集进行测试。对于 2 个阶段的聚类过程，引入了动态划分网格的方法以自适应当前数据的特征，并针对不同类型网格采用不同的方法进行处理，解决密度分布不均等问题。此外，根据用户的需求可得到不同时间段内数据的聚类结果。实验结果表明，本文算法能够较好地处理轨迹信息等时序数据流。由于本文算法仅能检测某时间段内用户的轨迹是否异常，而无法实时判断某一时刻用户所处的位置是否异常，因此如何判断用户某一时刻的位置是否异常和降低算法的时间复杂度，是下一步主要研究方向。

参考文献

[1] 胡 睿，林昭文，柯宏力，等. 一种基于密度和滑动窗口的数据流聚类算法[J]. 计算机科学，2011，38(5)：145-148.

[2] AGGARWAL C C，YU P S，HAN J，et al. A framework for clustering evolving data streams[C]//Proceedings of International Conference on VLDB'03. Washington D. C.，USA：IEEE Press，2003：81-92.

（下转第 46 页）

集群宕机恢复过程的效率有了 28% 以上的提升,并增强了 Redis 集群的可靠性。

Redis 集群只能在宕机主节点数少于主节点总数一半时才能从故障中恢复,这是由 Redis 本身实现的选举算法决定的。下一步工作是通过对选举算法进行分析改进,使得多数主节点故障宕机时集群也能够成功恢复,提升 Redis 集群对宕机节点数目的容忍度。

参考文献

[1] Redislabs. Redis cluster specification[EB/OL]. [2017-03-13]. http://redis.io/topics/cluster-spec/. 2016 April.

[2] FITZPATRICK B. Distributed caching with memcached[EB/OL]. [2017-03-13]. http://www.linuxjournal.com/article/7451.

[3] SHARMA V, CARROLL J, KHUNE A. Scaling deep social feeds at pinterest[C]//Proceedings of IEEE International Conference on Social Computing. Washington D.C., USA: IEEE Press, 2013: 777-783.

[4] CIDON A, EISENMAN A, ALIZADEH M, et al. Dynacache: dynamic cloud caching[C]//Proceedings of the 7th USENIX Conference on Hot Topics in Cloud Computing. New York, USA: ACM Press, 2015: 19.

[5] 焦 健,李 岩. 基于 Redis 的 SVG 空间信息可视化数据库[J]. 小型微型计算机系统, 2015, 36(6): 1193-1198.

[6] 阿里云. 云数据库 Redis 版[EB/OL]. [2017-03-13]. https://help.aliyun.com/document_detail/26342.html.

[7] SHAO Zhiyuan, JIN Hai, CHEN Bin, et al. HARTs: high availability cluster architecture with redundant TCP stacks[C]//Proceedings of IEEE International Conference on Performance, Computing, and Communications. Washington D.C., USA: IEEE Press, 2003: 253-260.

[8] BASSEK C K, PIERRE S, QUINTERO A. Redundancy schemes for high availability computer clusters[J]. Journal of Computer Science, 2006, 2(1): 33-47.

[9] DERIS M M, RABIEI M, NORAZIAH A, et al. High service reliability for cluster server systems[C]//Proceedings of IEEE International Conference on Cluster Computing. Washington D.C., USA: IEEE Press, 2003: 281-288.

[10] JI Zhanlin, GANCHEV I, O´DROMA M, et al. A distributed Redis framework for use in the UCWW[C]//Proceedings of International Conference on Cyber-enabled Distributed Computing and Knowledge Discovery. Washington D.C., USA: IEEE Press, 2014: 241-244.

[11] 黄健宏. Redis 设计与实现[M]. 北京: 机械工业出版社, 2015.

[12] KERMARREC A M, VAN STEEN M. Gossiping in distributed systems[J]. Acm Sigops Operating Systems Review, 2007, 41(5): 2-7.

[13] 刘德辉,尹 刚,王怀民,等. 分布环境下的 Gossip 算法综述[J]. 计算机科学, 2010, 37(11): 24-28.

[14] ROBERTSON A. Linux-HA heartbeat system design[C]//Proceedings of the 4th Annual Linux Showcase & Conference. New York, USA: ACM Press, 2000: 20.

[15] 张小芳,胡正国,郑继川,等. 高可用性集群技术的研究和应用[J]. 计算机工程, 2003, 29(4): 26-27.

编辑 顾逸斐

(上接第 32 页)

[3] CHEN Y, TU L. Density-based clustering for real-time stream data[C]//Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Jose, USA: ACM Press, 2007: 133-142.

[4] 杨 宁,唐常杰,王 悦,等. 一种基于时态密度的倾斜分布数据流聚类算法[J]. 软件学报, 2010, 21(5): 1031-1041.

[5] 徐正国,郑 辉,贺 亮,等. 基于局部密度下降搜索的自适应聚类方法[J]. 计算机研究与发展, 2016, 53(8): 1719-1728.

[6] 米 源,杨 燕,李天瑞. 基于密度网格的数据流聚类算法[J]. 计算机科学, 2011, 38(12): 178-181.

[7] 于彦伟,王 欢,王 沁,等. 面向海量数据流的基于密度的簇结构挖掘算法[J]. 软件学报, 2015, 26(5): 1113-1128.

[8] 于彦伟,王 沁,邝 俊,等. 一种基于密度的空间数据流在线聚类算法[J]. 自动化学报, 2012, 38(6): 1051-1059.

[9] 马 帅,王腾蛟,唐世渭,等. 一种基于参考点和密度的快速聚类算法[J]. 软件学报, 2003, 14(6): 1089-1095.

[10] 朱 燕,李宏伟,樊 超,等. 基于聚类的出租车异常轨迹检测[J]. 计算机工程, 2017, 43(2): 16-20.

[11] ZENG Y, CAPRA L, WOLFSON O, et al. Urban computing: concepts, methodologies, and applications[J]. ACM Transactions on Intelligent Systems & Technology, 2014, 5(3): 38.

[12] 金澈清,钱卫宁,周傲英. 流数据分析与管理综述[J]. 软件学报, 2004, 15(8): 1172-1181.

[13] ZHENG Y, XIE X, MA W Y. GeoLife: a collaborative social networking service among user, location and trajectory[J]. Bulletin of the Technical Committee on Data Engineering, 2010, 33(2): 32-39.

[14] 高嘉伟,梁吉业. 非平衡数据集分类问题研究进展[J]. 计算机科学, 2008, 35(4): 10-13.

[15] 高嘉伟,梁吉业,刘杨磊,等. 一种基于 Tri-training 的半监督多标记学习文档分类算法[J]. 中文信息学报, 2015, 29(1): 104-110.

编辑 索书志