基于轨迹点局部异常度的异常点检测算法

刘良旭[1)] 乐嘉锦[2)] 乔少杰[3)] 宋加涛[1)]

[1)](宁波工程学院电子与信息工程学院 浙江 宁波 315016)

[2)](东华大学计算机科学与技术学院 上海 200051)

[3)](西南交通大学信息科学与技术学院 成都 610031)

摘 要 随着大量的定位数据被收集在应用服务器,如何从大量定位轨迹数据挖掘异常信息已逐渐成为一个令人关注的研究课题.针对当前流行的、以轨迹片段表示局部特征的异常点检测算法存在的问题,文中提出了以轨迹点表示局部特征的异常点检测算法 TraLOD.该算法不仅提出了将每个轨迹点赋予一个 0～1 的值来表示其局部异常程度,而且还引入了相对距离来计算轨迹片段之间的不匹配性.此外,针对数据挖掘算法效率低的缺点,TraLOD 引入了 R-Tree 和距离特征矩阵来提高算法效率.性能分析和实验都证明了 TraLOD 的有效性.

关键词 轨迹数据;异常点检测;局部异常度;距离特征矩阵;R 树索引

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01966

Trajectory Outliers Detection Based on Local Outlying Degree

LIU Liang-Xu[1)] LE Jia-Jin[2)] QIAO Shao-Jie[3)] SONG Jia-Tao[1)]

[1)](*School of Electron and Information Engineering, Ningbo University of Technology, Ningbo, Zhejiang* 315016)

[2)](*School of Computer Science and Technology, Donghua University, Shanghai* 200051)

[3)](*School of Information Science and Technology, Southwest Jiaotong University, Chengdu* 610031)

Abstract Along with more and more location data being collected in application servers, how to mine outliers from these trajectory datasets is becoming an interesting topic. Aiming to the problems that were brought by using segment as local feature in trajectory outlier detection, the paper not only presents the concept of local outlying degree to express the outlying degree of local feature, but also introduces relative distance to compute the dismath between two segments. Moreover, to fast the performance, R-Tree and distance feature matrix are introduced. Finally, both of detailed experiments and performance analysis prove the algorithm available.

Keywords trajectory data; outlier diction; local outlier degree; distance feature matrix; R-tree index

1 引 言

随着 GPS 定位、传感器网络和无线通信等应用的日益普及,越来越多的轨迹数据被收集和保存在应用服务器.如何快速地从这些轨迹数据集中发现有效信息日益成为一个令人关注的研究课题.在传统异常点挖掘算法中,对象通常由几个相互独立的

收稿日期:2011-07-18;最终修改稿收到日期:2011-08-19.本课题得到国家自然科学基金(60972163,61070031,61100045)、浙江省自然科学基金(Y1100598,Y1080123)、教育部人文社会科学研究青年基金项目(10YJCZH117)、中国博士后科学基金项目(20090461346)、中国博士后科学基金特别资助项目(201104697)、中央高校基本科研业务费专项资金科技创新项目(SWJTU09CX035)和宁波市自然科学基金(2009A610090,2010A610106,2011A610175)资助.**刘良旭**,男,1974 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为数据库与数据挖掘.E-mail: luransh@gmail.com. **乐嘉锦**(通信作者),男,1951 年生,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为数据库与数据仓库、软件工程理论与实践.E-mail: lejiajin@dhu.edu.cn. **乔少杰**,男,1981 年生,博士,讲师,主要研究方向为数据库与数据挖掘. **宋加涛**,男,1966 年生,博士,教授,主要研究领域为图像识别.

属性组成，对象之间的距离一般通常以各个属性差值的加权和表示[1-5]；而在轨迹数据中，独立对象是由若干个相互联系的点组成的轨迹，因此，现有的轨迹数据异常点检测都是针对异常轨迹的检测. Knorr 等人[1]将轨迹转变成一个由位置、方向和速度等 3 个独立属性组成的对象. 然后运用传统距离函数计算轨迹间距离以检测异常轨迹. 该方法的优点是可以利用传统的异常点检测算法，但它的缺点在于：仅比较了轨迹的整体特性，而忽略了其局部特征. 而随着轨迹越来越长，这种方法的缺点也日益明显. 为了解决这个问题，Li 等人[6]提出了基于 motifs 的异常检测算法，它使用 k-means 算法从滑动窗口中收集 k-代表模式(motif)，然后使用分类器思想来检测异常轨迹. 其缺点在于：实际应用领域很难找到标准的训练数据集.

Lee 等人[7]提出了 TRAOD 算法. 该算法先将轨迹划分成线段，以线段表示轨迹局部特征(本文称之为基本比较单元)，然后使用线段 Hausdorf 距离计算每两个线段之间的距离以表示其间的不匹配性，从而确定异常轨迹. 此外，为了提高算法效率，TRAOD 中还给出了一种二阶段划分方法来提高算法效率，即先将轨迹划分成粗粒度的轨迹片段，同时用轨迹片段的起始点和终点的连线表示其局部特征，寻找异常的粗粒度轨迹片段；然后再将异常的粗粒度轨迹片段进行线段划分，寻找其中的异常线段. TRAOD 很好地解决了长轨迹之间不匹配性的计算，但它存在以下几个问题：首先，随着轨迹变得越来越长，线段并不能有效地表示轨迹的局部特征. 例如，应用 1 中的两条轨迹 Tr_1 和 Tr_2 尽管拥有不同的运动特征，但构成其的线段集合却极其相似；其次，TRAOD 使用线段的 CPU 时间 Hausdorff 距离表示两个线段之间的不匹配性，这可能使得算法存在如应用 2 那样描述的问题；第三，二阶段划分方法的局限性在于要求粗粒度轨迹片段内每个线段的方向要和该片段的开始点和终止点连接而成的线段方向基本保持一致. 显然，对于复杂轨迹来说，这个条件往往很难得到满足.

应用 1.　给定两条轨迹 $Tr_1=\{l_1,l_2,l_3,l_4,l_5,l_6,l_7,l_8\}$ 和 $Tr_2=\{l'_2,l'_4,l'_1,l'_5,l'_7,l'_6,l'_3,l'_8\}$，其中构成 Tr_2 的 l'_i 表示其拥有与 Tr_1 的 $l_i(0<i<9)$ 相同的长度和方向(即除了位置稍微有点不同之外，l'_i 拥有与 l_i 相同的特征)(如图 1 所示). 从图中可以看到，尽管两条轨迹拥有完全不同的运动特征，但如果用线段集合来表示轨迹，却发现线段集合拥有几乎相同的运动特征，这显然不是我们所期望的结果. 因此，对于更长更复杂的轨迹来说，用线段表示其局部特征可能会产生与现实完全不同的实验结果.

应用 2.　如图 1 所示，某个邻域内 3 条轨迹 Tr_1，Tr_2 和 Tr_3. 从图中可以发现，Tr_1 和 Tr_3 拥有较相似的运动特征. 而 Tr_2 处在离 Tr_1 更近的位置. 如果采用 TRAOD 中的线段 Hausdorff 距离那样的绝对距离，就会得到 Tr_2、Tr_1 具有更高的相似度. 这显然与实际意义不符.

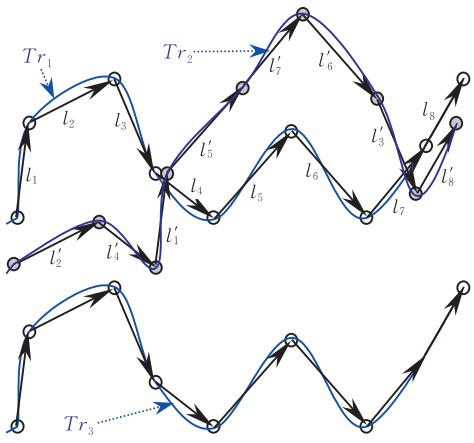


图 1　线段表示轨迹局部特征存在的问题

针对这些问题，本文提出了一种通过确定每个轨迹点的局部异常程度来确定轨迹中异常的新算法——基于局部异常度的轨迹异常点检测算法(Trajectory Outlier Detection Based on Local Outlier Degree，TraLOD). TraLOD 主要创新点在于：(1)提出了局部异常度的概念，即赋予每个轨迹点一个 0～1 的值以表示该点相对于其邻域内轨迹点的异常程度；(2)引入相对距离的概念. 参照基于平移的 Hausdorff 距离的思想，定义了基于相对距离的轨迹之间的距离函数，并在此基础上定义了轨迹点的局部异常度；(3)提出了基于 R-Tree 和距离特征函数的算法优化方法以提高算法效率.

2　距离函数和局部异常度的定义

随着定位技术和用户需求的日益提高，数据挖掘算法所面对的轨迹数据集也在不断地发生变化. 一方面，每条轨迹的时间和空间区间不断变大. 例如，飓风经过海洋、陆地、岛屿等区域时会出现不同的运动规律；另一方面，用户和系统对轨迹的精度要求也在不断提高. 这些都会导致轨迹变得越来越复

杂.针对这种现象,本文引入了局部异常度的概念.这里的局部包含两层含义:一是每个轨迹点仅和同一条轨迹上的若干个轨迹点有关联,即轨迹间的比较单位是若干个相邻的轨迹点组成;二是每个轨迹点的异常程度的计算仅涉及到其邻域内的轨迹点构成的基本比较单元,即基本比较单元仅和其邻域范围内的基本比较单元相比较,而不是整个数据空间.而 Lee 采用线段表示轨迹的局部特征,然后根据 MLHD 来计算轨迹线段之间的距离,但是对于复杂轨迹,线段很难有效表示轨迹的局部特征.本文采用的基本比较单元是长度为 k 的轨迹片段.这使得两个基本比较单元就变成了两个点数为 k 的点集.单从数据构成上看,两个基本比较单元之间的距离计算类似于图像识别中的距离计算,但是轨迹数据具有一些不同于图像数据的特征.首先,轨迹检测算法需要检测大量的基本比较单元之间的距离,Hausdorff 距离的计算复杂度太高;其次,Hausdorff 距离是无序的,这符合图像匹配的特征,然而轨迹点是有序的.最后,按照轨迹特征,基本比较单元仅与其邻域内的基本比较单元相比较,而不是整个数据空间.

接下来,我们按照基于平移的最小 Hausdorff 距离的思想,并结合轨迹数据的特点来分析轨迹点在其所在基本比较单元中的局部异常度.假设轨迹数据集 $S=\{S_i|1\leq i\leq n\}$,$|S_i|=m_i$,其中 S_i 表示任一轨迹,n 表示数据集包含的轨迹数目,m_i 表示轨迹 S_i 包含的轨迹点数目.$S_{ip}=\{s_{ip},\cdots,s_{i(p+k-1)},p+k-1\leq m_i\}$,$S_{jq}=\{s_{jq},\cdots,s_{j(q+k-1)},i\neq j,p+k-1\leq m_j\}$分别表示轨迹 S_i 和 S_j 任一基本比较单元.首先,由于轨迹点是有序的,且基本比较单元长度相同,因此,我们可以将 $dist(s_{i(p+t)},s_{j(q+t)})(1\leq t\leq k)$作为轨迹点 $s_{i(p+t)}$ 与基本比较单元 S_{jq} 的距离,这不仅可保证每个轨迹点有序这个特征,而且其平移变量 t 可以用其所有轨迹点 $s_{i(p+t)}$ 到其对应点 $s_{j(q+t)}$ 的平均差值 $\bar{t}=1/k\sum_{q=0}^{k-1}(s_{i(p+t)},s_{j(q+t)})$ 表示.因此,轨迹点 $s_{i(p+t)}$ 在基本比较单元 S_{ip} 与 S_{jq} 的异常度 $LO_k(S_{i(p+t)}|S_{ip},S_{jq})$ 定义如下:

$$LO_k(s_{i(p+t)}|S_{ip},S_{jq})=dist\left((s_{i(p+t)}-s_{j(q+t)})-1\Big/k\sum_{r=0}^{k-1}(s_{i(p+r)}-s_{j(q+r)})\right) \quad (1)$$

其中,$dist()$ 表示一种距离函数,如 L_1,L_2…(本文采用 L_2).

显然这个距离是基于平移的最小 Hausdorff 距离的简化,它可以消除相似轨迹之间的公共偏差,但它仍然无法检测出方向相反的异常轨迹.为了解决这个问题,下面引入了邻域点集的概念.

定义 1. 给定一个邻域阈值 ω,一个轨迹点 s_{ip},对于任一其它轨迹上的点 s_{jq},如果$(s_{ip}-s_{jq})\leq\omega$,那么 s_{jq} 包含在 s_{ip} 的 ω-邻域点集,表示为 $s_{jq}\in N_\omega(s_{ip})$.

定义 2. 对于两个基本比较单元 S_{ip} 和 S_{jq},如果 S_{jq} 所有轨迹点都满足 $s_{j(q+t)}\in N_\omega(s_{i(p+t)})(1\leq t\leq k-1)$,那么称 S_{ip} 和 S_{jq} 是相近基本比较单元,(S_{ip},S_{jq})被称为相近的基本比较单元对.

按照前面的分析,基本比较单元只和其邻域范围内的其它基本比较单元做比较,因此,轨迹点 $s_{i(p+t)}$ 在相近的基本比较单元对 s_{ip} 与 s_{jq} 的局部异常值定义如下.

定义 3. 给定不同轨迹的两个基本比较单元 S_{ip} 和 S_{jq},S_{ip} 的轨迹点 s_{ir} 在这个相近基本比较单元对(S_{ip},S_{jq})的局部异常值 $LOD_{k,\omega}(S_{i(p+t)}|S_{ip},S_{jq})$ 定义如下.

$$LOD_{k,\omega}(s_{ir}|S_{ip},S_{jq})=\begin{cases}dist(s_{ir}-s_{j(q+r-p)})-1\Big/k\sum_{t=0}^{k-1}(s_{i(p+t)}-s_{j(q+t)}), & \text{当 } S_{ip}\text{ 和 }S_{jq}\text{ 相近}\\ 0, & \text{当 } S_{ip}\text{ 和 }S_{jq}\text{ 不相近}\end{cases} \quad (2)$$

其中,$dist()$ 表示向量的距离值,如 L_1,L_2,…(本文采用 L_2).

定义 3 给出了轨迹点在其所属的基本比较单元与其相近的基本比较单元的局部异常程度,如果这两个基本比较单元不是相近的,那么该轨迹点的局部异常值在这两个基本比较单元是没有意义的(定义 3 中用 0 代替其值,然后再结合定义 4、定义 5 实现这个思想).

定义 4. 假设轨迹点 s_{ir} 表示轨迹 S_i 中的第 r 个轨迹点,$S_{ip}=\{s_{ip},\cdots,s_{i(p+k-1)}|\max(0,r-k+1)\leq p\leq r\}$ 表示包含轨迹点 s_{ir} 的任一基本比较单元,$S_{jq}=\{s_{jq},\cdots,s_{j(q+k-1)}|\ j\neq i,s_{j(q+t)}\in N_\omega(s_{i(p+t)},0\leq t\leq k-1\}$ 表示与 S_{ip} 相近的某一基本比较单元,N_u 表示与 S_{ip} 相近的基本比较单元 S_{jq} 的个数.那么轨迹点 s_{ir} 在基本比较单元 S_{ip} 的局部异常值为

$$LOD_{k,\omega}(s_{ir}|S_{ip})=\begin{cases}\min(LOD_{k,\omega}(s_{ir}|S_{ip},S_{jq})), & \text{当 } N_u\neq 0\\ 0, & \text{当 } N_u=0\end{cases} \quad (3)$$

定义 5. 设 s_{ir} 表示轨迹 S_i 的第 r 个轨迹点,

$S_{ip}=\{s_{ip},\cdots,s_{i(p+k-1)}\mid \max(0,r-k+1)\leq p\leq r\}$表示任一包含轨迹点 s_{ir} 的基本比较单元，N_p 表示与 S_{ip} 具有相近的基本比较单元的轨迹数目. 那么轨迹点 s_{ir} 局部异常值为

$$LOD_{k,\omega,\lambda}(s_{ir})=\begin{cases}\sum\limits_{p=\max(1,r-k+1)}^{r} LOD_{k,\omega}(s_{ir}|S_{ip})/(N_p\times\max(1,r-k+1)), & \text{当 } N_p\geq\lambda\\ \left(N_p\times\omega+\sum\limits_{p=\max(1,r-k+1)}^{r} LOD_{k,\omega}(s_{ir}\mid S_{ip})\right)/\lambda, & \text{当 } N_p<\lambda\end{cases}\tag{4}$$

其中，相近阈值 λ 表示与轨迹点所在的基本比较单元的轨迹数目.

按照定义 4，当该点所在的基本比较单元都找不到一个相近的基本比价单元时，该轨迹点的局部异常值为 ω，即它具有最高的局部异常度. 由于相近的两个基本比较单元的任何一对轨迹点之间的距离都小于邻域阈值 ω. 那么轨迹点在这个相近基本比较单元对的局部异常值总是满足 $LOD_{k,\omega}(S_{i(p+t)}\mid S_{ip},S_{jq})\leq\omega$，因此，我们可以推出 $LOD_{k,\omega}(S_{ir})$ 的取值范围为$[0,\omega]$. 当 $LOD_{k,\omega}(S_{ir})=\omega$ 则表示该轨迹点所在基本比较单元无相近的基本比较单元或是该点在包含该点的所有相近的基本比较单元对的局部异常值都为 ω；当 $LOD_{k,\omega}(S_{ir})=0$ 则表示该点在包含其的所有相近基本比较单元对的局部异常值都为 0，即对于每个相近的基本比较单元对，平移之后的该点与其对应点相重合. 将等式(2)进行归一化处理，就可以得到归一化的局部异常度：

$$LOD_k(s_{ir}\mid S_{ip},S_{jq})=\begin{cases}(1/\omega)dist\left((s_{ir}-s_{j(r+q-p)})-(1/k)\sum\limits_{t=0}^{k-1}(s_{i(p+t)}-s_{j(q+t)})\right), & \text{当 } S_{ip}\text{和}S_{jq}\text{相近}\\ 0, & \text{当 } S_{ip}\text{和}S_{jq}\text{不相近}\end{cases}\tag{5}$$

轨迹点在其所在基本比较单元的局部异常度：

$$LOD_k(s_{ir}\mid S_{ip})=\begin{cases}1/N_u\min(LOD_{k,\omega}(s_{ir}\mid S_{ip},S_{jq})), & \text{当 } N_u\neq0\\ 0, & \text{当 } N_u=0\end{cases}\tag{6}$$

轨迹点的局部异常度为

$$LOD_{k,\lambda}(s_{ir})=\begin{cases}\left(\sum\limits_{p=\max(1,r-k+1)}^{r} LOD_{k,\omega}(s_{ir}\mid S_{ip})\right)/(N_p\times\max(1,r-k+1)), & \text{当 } N_p\geq\lambda\\ \left(N_p+\sum\limits_{p=\max(1,r-k+1)}^{r} LOD_{k,\omega}(s_{ir}\mid S_{ip})\right)/\lambda, & \text{当 } N_p<\lambda\end{cases}\tag{7}$$

3　Naive 算法

假设轨迹数据集 $S=\{S_i\mid 1\leq i\leq n\}$，$|S_i|=m_i$ 表示每条轨迹的长度，S_{ip} 为 S_i 的任一基本比较单元，S_{jq} 为 S_j 的任一基本比较单元. Naive 算法就是将每条轨迹的每个基本比较单元与其它轨迹的每个基本比较单元作比较，计算其每个轨迹点在这两个基本比较单元的局部异常度，最后求其平均值就是该轨迹点的局部异常度. 图 2 给出 Naive 算法的伪代码. 实现思想就是对每条目标轨迹 S_i，逐一计算 S_i 的每个基本比较单元的局部异常度(即与其它轨迹的基本比较单元逐一比较，数组 *outlier* 用来存放目标轨迹每个点的累计局部异常度值和该点对基本比较单元对的局部异常度个数)(第 2～8 行)；接着，根据 *outlier* 和定义计算轨迹点的局部异长度(第 9～14 行).

```
NaiveCompLOD(S, k, ω)
1.  for each trajectory S_i do
2.      Initialize array outlier[m_i]
3.      for each trajectory S_j(i≠j) do
4.          for each (S_ip , S_jq)
5.              if S_ip is closed to S_jq then
6.                  for each point s_i(p+t) in S_ip do
7.                      outlier[p+t].val+=LOD_k(s_ir)|S_ip ,S_jq);
8.                          outlier[p+t].num ++;
9.      for each point s_it in S_i do
10.         if outlier[t].num!=0 then
11.             LOD_k(s_it)=outlier[t].val/outlier[t].num;
12.         else
13.             LOD_k(s_it)=1;
14.     OutputLODToFile(s_it ,LOD_k(s_it))
```

图 2　Naive 算法的伪代码

下面以距离计算为主要代价分析其计算复杂度. 对于一条轨迹 S_i，其所包含的长度为 k 的基本比较单元数目为 m_i-k+1，那么对于两个轨迹 S_i 和 S_j，需要比较的基本比较单元次数为 $(m_i-k+1)\cdot(m_j-k+1)$，而比较两个基本比较单元的计算复杂度为 $u(1\leq u\leq k)$，因此，对于两个轨迹 S_i 和 S_j，用 Naive 算法检测其是否全局匹配的计算复杂度 $cost(S_i,S_j)$ 为

$$cost(S_i,S_j)=(m_i-k+1)\times(m_j-k+1)\times u \quad (8)$$

轨迹挖掘的数据量通常比较大，我们不考虑缓存先前计算得到的距离. 因此，Naive 算法的总计算复杂度为

$$cost(S)=\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n}(m_i-k+1)\times(m_j-k+1)\times u \quad (9)$$

假设每条轨迹长度相同，即 $m_i=m$，那么等式(9)转化为

$$cost(S_i,S_j)=n(n-1)(m-k+1)^2\times u \quad (10)$$

4 基于 R-Tree 的异常轨迹检测算法

对于数据量及其庞大的轨迹数据集来说，Naive 算法的计算代价是昂贵的. 在传统数据挖掘领域，引入索引是常用的算法优化方法. 但是，空间索引算法 R-Tree 并不适合用来索引时空数据. 尽管很多研究者提出了一些有效的索引结构(如 TPR*-Tree[9]，LUR-Tree[10])，但当前还不存在一种公认有效的时空数据索引结构. 因此，直接对时空数据建立索引并不可行. TRAOD[6] 引入的二阶段划分尽管加速异常轨迹的检测，但这种方法的局限性在于它必须保证粗粒度的轨迹片段的方向与其内部的细粒度轨迹片段的方向保持一致，否则算法的准确性将明显下降. 但这个局限性使得二阶段划分并不适合在复杂轨迹数据集使用. 此外，由于 TraLOD 中轨迹片段是相互重合的，二阶段划分方法会导致某些轨迹片段丢失.

由于在 TraLOD 中，每个轨迹点的局部异常度仅取决于同一条轨迹的相邻轨迹点以及其对应的邻域点集内的轨迹点相比较，即仅与目标轨迹点的邻域内的轨迹相关. 而且，我们并不考虑轨迹点的绝对时间，即该轨迹数据集可以看成空间数据集. 因此，除了将轨迹数据集按照轨迹顺序存放之外，TraLOD 框架还引入 R-Tree 来索引所有轨迹点以实现快速地找出所有可能相近的基本比较单元对. TraLOD 主要实现过程分为两个阶段：首先，算法利用 R-Tree 和两条轨迹的距离特征矩阵找出相近的基本比较单元对，然后确定其中每个轨迹点在这个基本比较单元对的局部异常度.

4.1 数据存储结构

TraLOD 采用的数据结构主要由三个部分构成：存储轨迹点的数据文件、根据轨迹号索引所有轨迹点的 B^+-Tree 和按空间位置索引所有轨迹点的 R-Tree. 即所有轨迹数据按照轨迹编号和点的顺序存放在一个二进制文件中. 系统在轨迹编号和点序号上建立 B^+-Tree 索引，并为了实现对 R-Tree 的支持，对轨迹点的经纬度坐标建立 R-Tree 索引. 相对于 Naive 算法，这种数据结构多了一个 R-tree 索引(作为空间数据索引技术的典型算法，R-Tree 通常都被会存储轨迹数据的数据库支持). 此外，由于 R-tree 是一个以矩形框为单元的索引算法，而轨迹点的 ω-邻域区域是一个圆形区域，因此，进行 R-tree 查询时，必须经过一定的距离计算来确定轨迹点的领域点集.

4.2 距离特征矩阵

为了方便描述，在描述 TraLOD 实现过程之前，先引入距离特征矩阵的概念. 假设给定两条轨迹 $A=\{a_1,\cdots,a_n\}$ 和 $B=\{b_1,\cdots,b_n\}$，那么 A、B 间的距离特征函数可以用一个由元素 (a_i,b_j) 组成的矩阵来表示. 对于给定阈值 ω，将矩阵中 $dist(a_i,b_j)>\omega$ 的元素置成 0，那么，对于大多数复杂轨迹来说，它们之间的距离特征矩阵通常都是稀疏矩阵. 下面给出正对角序号和正对角相邻的概念.

定义 6. 矩阵中元素 (a_i,b_j) 的正对角序号是其行号和列号之差，表示为 $pssn(a_i,b_j)=i-j$.

定义 7. 假设矩阵的两个元素 (a_i,b_j) 和 (a_x,b_y) 的正对角序号相同，如果满足 $|i-x|=1$，那么称 (a_i,b_j) 和 (a_x,b_y) 是正对角相邻.

按照上面的定义，两个相近的基本比较单元其实就是这两条轨迹构成的距离特征矩阵中 k 个正对角相邻的非 0 元素.

4.3 算法实现

按照上一节的分析，搜索相近的基本比较单元就转变为搜索目标轨迹与其它轨迹的距离特征矩阵中所有 k 个正对角相邻的非 0 元素. 因此，TraLOD 的实现思想如下(伪代码如图 3 所示)：首先，针对每条目标轨迹，借助于 R-tree 找出该轨迹中所有轨迹点的邻域点集 $\{(s_{ip},s_{jq})\mid dist(s_{ip},s_{jq})\leq\omega\}$，将其放入一个按(比较轨迹 S_j 编号，正对角序号，点在轨迹中位置)排序的堆栈 *stackNeigh*(当该堆栈中元素过多时，部分元素会保存在磁盘文件)(第 1 行，由函数 *QueryRTree* 实现). 接着，由于距离特征矩阵中正对角相邻的元素在堆栈中的存放位置肯定是相邻的，所以，要逐一从堆栈中取出栈顶元素，判断其和之前取出的元素是否处在同一个距离特征矩阵、是否是正对角相邻，并作相应的处理，计算轨迹点在相近的基本比较单元对的局部异常度(第 3～15 行)；

最后，按照前面计算的轨迹点在相近基本比较单元对的局部异常度计算每个轨迹点的局部异常度（第16～21行）.

```
ComputingLOD(k,ω)
1. For each Si, do
2.    Initialize outlier;
3.    ComputingLODofOne(Si,k,ω);
ComputingLODofOne(Si,k,ω)
1. stackNeigh=QueryRTree(Si,ω);
2. cur_traj=-1, cur_pos=-1, cur_pssn=-MAXVAL;
3. while (stackNeigh isn't NULL) do
4.    curEn=stackNeigh.RevTop();
5.    If (curEn.traj!=cur_traj) or (curEn.pssn!=
        cur_pssn) or (curEn.pos-1!=curPos) then
6.       if (CloseUnitSet.Size()≥k) then
7.          CompLODofSeg(CloseUnitSet,outlier,k,ω);
8.       if (curEn.traj!=cur_traj) then
9.          cur_traj=curEn.TrajID;
10.         cur_pssn=curEn.pssn;
11.      else if (curEn.pssn!=cur_pssn) then
12.         cur_pssn=curEn.pssn;
13.      CloseUnitSet.Size()=0;
14.   CloseUnitSet.addEntry(curEn);
15.   curPos=curEn.pos;
16. for each point sir in outlier do
17.      if (outlier[r].num!=0)
18.         LOD(sir)=outlier[r].val/outlier[r].num;
19.      else
20.         LOD(sir)=1;
21.   OutputLODToFile(sit,LODk(sit));
CompLODofSeg(CloseUnitSet,outlier,k,ω)
1. closeUnitArray=GetCloseUnit(CloseUnitSet,k);
2. for each entry CloseUnit in CloseUnitArray do
3.    CompLODofUnit(CloseUnit,outlier,k,ω);
```

图 3　基于 R-Tree 算法的伪代码

算法中，函数 *CompLODofSeg* 用来计算其中目标轨迹点在相近基本比较单元对的局部异常值，其包含 4 个输入参数：相邻元素数组 *CloseUnitSet* 用来存放正对角相邻的基本比较单元对；两个参数 k，ω；局部异常度数组 *outlier*. 该函数从 *CloseUnitSet* 取出相近基本比较单元对（个数为 *CloseUnitSet* 数组个数$-k+1$，每个相近基本比较单元对对应 k 个正对角相邻的数组元素）；然后对每个相近的基本比较单元对，按照等式(5)计算每个轨迹点在该基本比较单元对的局部异常度，并保存其的最小局部异常度值（由函数 *CompLODofUnit* 实现）.

4.4　性能分析

本节将以轨迹点之间的距离计算作为主要代价来分析算法性能. 由于在搜索过程仅有 R-Tree 搜索需要距离计算，因此我们仅考虑 R-Tree 搜索和计算阶段的计算代价. 各个参数说明如表 1 所示. 假设轨迹数据集 $S=\{S_i \mid 1\leq i\leq n\}$，$|S_i|=m_i$ 表示每条轨迹的长度，轨迹数据空间的整个区域为 $L\times L$，且轨迹点在整个数据空间均匀分布，则轨迹 S_i 每个点的单位区域内且不属于 S_i 的轨迹点数目为

$$N_i=\sum_{j\neq i,j=1,\cdots,n} m_j/L^2 \tag{11}$$

表 1　参数说明

变量名	说明
$S_{\omega\text{-}ER}$	轨迹 S_i 中每个轨迹点的 ω-邻域园外接矩形（Enclosing Rectangle）的面积.
$S_{\omega\text{-}IR}$	轨迹 S_i 中每个轨迹点的 ω-邻域园内接矩形（Inscribing Rectangle）的面积.
N_{tpi}	包含在目标轨迹点的 ω 邻域区域的外接矩形和内接矩形之间的轨迹点数目（S_i 的轨迹点除外）
N_i	单位面积中包含的轨迹点数目（不包含轨迹 S_i）
N_{total_point}	总的轨迹点数目
N_{total_bcu}	总的基本比较单元对数目

首先，ω-邻域是圆形区域，而 R-Tree 是矩形区域搜索，因此，对于每个轨迹点，算法必须检查其 ω-圆形邻域的内接矩形和外接矩形之间的点是否处在 ω-邻域之内，则搜索 S_i 的每一轨迹点的计算复杂度为

$$N_{tpi}=(S_{\omega\text{-}ER}-S_{\omega\text{-}IR})/L^2\times N_i=2\omega^2/L^2\sum_{j\neq i,j=1\cdots n} m_j \tag{12}$$

搜索 R-Tree 的计算复杂度为

$$cost_q=N_{total\_point}\times N_{tpi}=\sum_{i=1}^{n} m_i\times\frac{2\omega^2}{L^2}\sum_{j\neq i,i=1}^{n} m_j$$
$$=\sum_{i=1}^{n}\sum_{j\neq i,i=1}^{n}\frac{2\omega^2}{L^2}m_j m_i \tag{13}$$

其次，在计算阶段，由于每个非 S_i 的轨迹点处在 S_i 的某个轨迹点的 ω-邻域之内的概率为 $\pi\omega^2/L^2$，S_i 的某个基本比较单元 S_{ip} 存在一条相近的基本比较单元的概率为 $(\pi\omega^2/L^2)^k$. 而总的基本比较单元对的数目为

$$N_{total\_bcu}=\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n}(m_i-k+1)\times(m_j-k+1) \tag{14}$$

计算阶段的总代价为

$$cost_c=N_{total\_bcu}\times k\times(\pi\omega^2/L^2)^k$$
$$=\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n}(m_i-k+1)\times(m_j-k+1)\times k\times(\pi\omega^2/L^2)^k \tag{15}$$

根据等式(13)和等式(15)，该算法总的计算代价为

$$cost=cost_q+cost_c$$
$$=\sum_{i=1}^{n}\sum_{j\neq i,i=1}^{n}\frac{2\omega^2}{L^2}m_j m_i+\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n}(m_i-k+1)\times(m_j-k+1)\times k\times(\sqrt{\pi}\omega/L)^{2k} \tag{16}$$

假设每条轨迹的长度相同，即 $m_i=m$，又由于 $\omega\ll L$，且 $k\geq 2$，则等式(16)可以简化为

$$cost=n(n-1)(m^2)\frac{2\omega^2}{L^2}+$$

$$n(n-1)(m-k+1)^2k(\sqrt{\pi}\omega/L)^{2k}$$
$$=n(n-1)m^2\frac{\omega^2}{L^2}\left(2-k\left(\frac{m-k+1}{m}\right)^2(\sqrt{\pi}\omega/L)^{2k-2}\right)$$
$$\approx 2n(n-1)m^2\omega^2/L^2 \quad (17)$$

由于复杂轨迹长度 $m\gg k$，根据等式(10)和等式(17)得到 TraLOD 和 Naive 算法的性能之比

$$p=\frac{2n(n-1)m^2\frac{\omega^2}{L^2}}{n(n-1)(m-k+1)^2\times u}=\frac{2m^2}{(m-k+1)^2\times u}\frac{\omega^2}{L^2}\approx\frac{2\omega^2}{uL^2} \quad (18)$$

由于 $\omega\ll L$，$u\approx L$，所以 TraLOD 性能要大大优于 Naive 算法，下面实验也证实了该结果.

5 实验结果

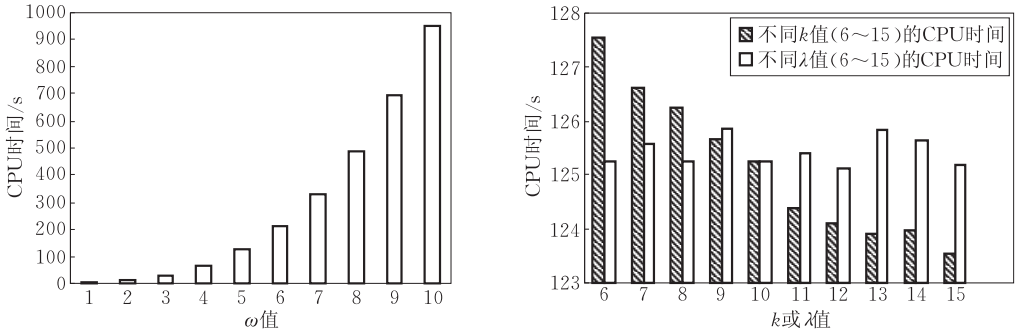
基于本文提出的轨迹检测算法，笔者开发出异常轨迹检测系统 TrajDetector. 该系统的开发环境是 Visual C++6.0，操作系统为 Windows XP. 实验硬件环境为：CPU 为 Centrino2 2.6 GHz，内存为 512 GB. R-Tree 采用的是 Beckmann[11] 提出的 R*-tree①. 实验数据集来自真实的飓风数据②，该数据包括飓风的经纬度位置信息、最大中心风力、中心压力等，本文抽取其中的经度、纬度的数据项作为实验数据. 本文采用的实验数据有两个：(1) 东太平洋飓风数据集. 该数据集记载了 1949 年到 2006 年的东太平洋上飓风移动数据，包括 35986 个点组成的 813 条轨迹；(2) 西太平洋飓风数据集. 该数据集记载了 1949 年到 2008 年的西太平洋上飓风留下的移动轨迹，包括 17691 个点组成的 565 条轨迹.

5.1 参数影响

TraLOD 涉及到 3 个参数：邻域阈值 ω、基本比较单元的长度 k 和相近阈值 λ. 下面我们来讨论这 3 个参数对算法性能和轨迹点的局部异常度 LOD 的影响. 实验数据集采用东太平洋飓风数据集.

5.1.1 参数对算法性能的影响

邻域阈值 ω、基本比较单元的长度 k 和相近阈值 λ 对算法性能的影响是不同的. 对算法性能影响最大的参数是邻域阈值 ω. 图 4 显示了随着 ω，k 或 λ 的变化(ω 从 1～10，k 从 6～15，λ 从 6～15)，算法所花费的 CPU 时间(默认参数值 $\omega=5$，$k=10$，$\lambda=10$). 从图中可以看到，影响最大的是参数 ω，因为它的变化会直接影响轨迹点的邻域点集中的点数. 当 ω 等于 1 时，算法的执行时间极少(仅有 5s)，这是因为此时每个轨迹点的邻域点集包含的点都很少，其所在基本比较单元的相近基本比较单元更少，从而导致算法的执行时间大大降低，但随着 ω 的增大，算法的执行时间快速增大，当 ω 等于 10 时，算法的执行时间已经接近 1000 s. 因为它要检索所有可能的基本比较单元. 参数 k 对算法性能的影响就比较少. 图 4(a)显示了改变参数 k 值($\omega=5$，$k=6\sim15$，$\lambda=10$)时，算法执行时间的变化情况. 显然，随着 k 的增加，算法的执行时间稍微有点减少，这是因为 k 的增大会导致每个轨迹点所在基本比较单元所对应的相近基本比较单元会减少，从而导致算法的计算阶段中的计算时间减少. 该值变化不大的原因是：如算法性能分析中所述的，算法的主要计算时间主要集中在 R-Tree 的查找阶段. 图 4(b)同样也显示了随着参数 k 变化(从 6～15，$\omega=5$，$k=10$，$\lambda=6\sim15$)算法执行时间的情况. 从图中可以发现，参数 λ 对算法性能几乎没有影响，这是因为 λ 的变化只是在最后计算 LOD 值时使用，它对算法中的 R-Tree 搜索和计算阶段都没有任何影响.



(a) 不同ω值的CPU时间　　(b) 不同k或λ的CPU时间

图 4 不同 ω(k 或 λ)值的 CPU 时间

① http://www.rtreeportal.org/
② http://weather.unisys.com/hurricane/index.html

5.1.2　参数对局部异常度的影响

轨迹点的局部异常度(LOD)值会随着邻域阈值 ω、基本比较单元的长度 k 和相近阈值 λ 3 个参数的变化而变化. 为了比较不同参数对 LOD 值的影响,我们从数据集中随机取 100 个轨迹点进行分析.

图 5 显示了不同 ω 值时($\omega=3,5,7,10$, $k=10$, $\lambda=10$),100 个随机轨迹点的 LOD 值的变化情况. 从图中可以看到,随着 ω 值的增大,轨迹点的 LOD 值逐渐减小,而且其趋势是越来越慢. 其原因是:随着 ω 值的增大,越来越多的轨迹点被包含在该点的邻域点集中,这使得相近的基本比较单元对逐渐增加,当拥有相近基本比较单元的轨迹数目少于 λ 时(由于不足的局部异常度是由 1 代替),局部异常度迅速减少,当其值大于等于 λ 时,局部异常度的变化开始变小. 从图中也可以看到大多数轨迹点在 $\omega=7$ 和 $\omega=10$ 的变化很小.

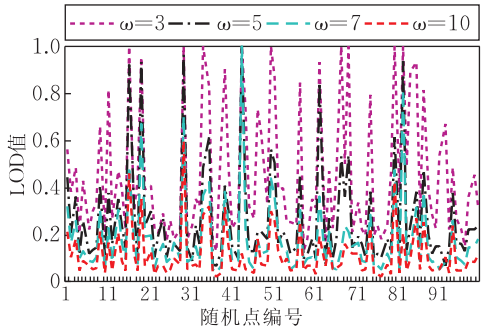


图 5　不同 ω 的 LOD 值

图 6 显示了不同 k 值时($\omega=5$, $k=5,7,10,15$, $\lambda=10$),100 个随机轨迹点的 LOD 值的变化情况. 尽管 k 的变化对算法性能影响不大,但对 LOD 的值影响还是挺大的. 从图中可以看到,随着 k 值的增大,轨迹点的 LOD 值逐渐增大,而且其趋势是越来越快. 其原因是:由于 ω 保持不变,那么每个轨迹点的邻域点集内的轨迹点数目保持不变,随着 k 值的增大,要求基本比较单元的长度增加,这导致对应的相近基本比较单元减少,从而使得 LOD 值慢慢增大. 当具有相近的基本比较单元的轨迹数少于 λ 时,由于与 ω 变化时同样的原因,LOD 值开始快速增大. 图中显示:当 $k=15$ 时,轨迹点的 LOD 值都开始快速向 1 靠近.

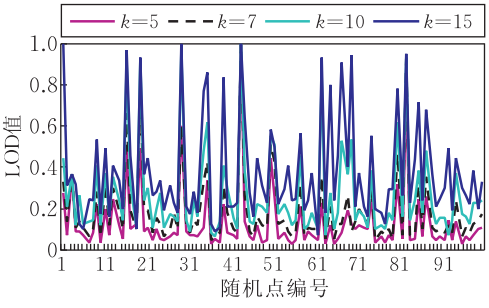


图 6　不同 k 的 LOD 值

图 7 显示了不同 λ 值时($\omega=5$, $k=10$, $\lambda=5,7,10$,或 15),100 个随机轨迹点的 LOD 值的变化情况. 从图中可以看到,λ 值的变化对大多数轨迹点的 LOD 值没有影响,只有在极个别轨迹点,其 LOD 值会随着 λ 值的增大而逐渐增大. 这是因为 λ 是用来限制拥有相近基本比较单元的最少轨迹数,即具有与轨迹点所在基本比较单元相近的基本比较单元的轨迹数目. 当该数目大于多于 λ,按照等式(7),其 LOD 值不会变化,否则,其 LOD 值会随着 λ 值的增大而增大.

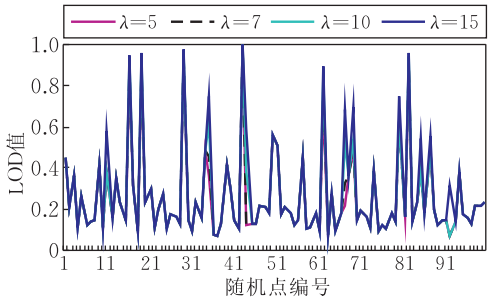


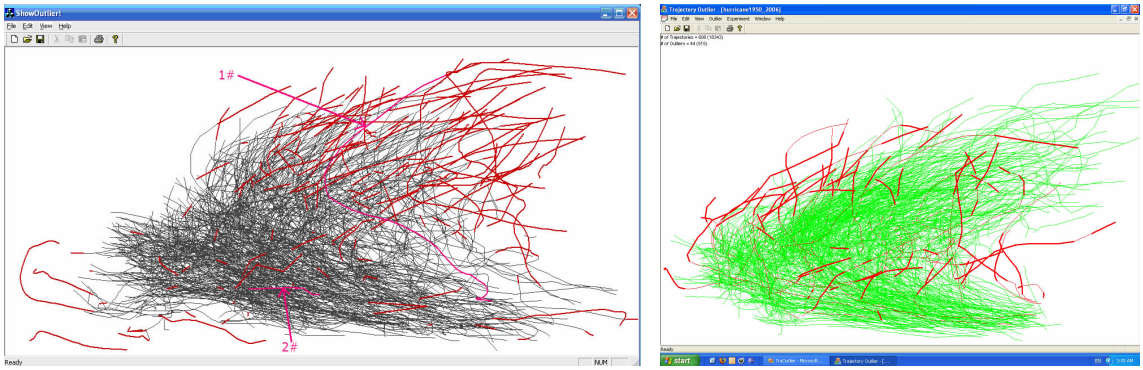
图 7　不同 λ 的 LOD 值

综上所述,最主要参数是 ω,它不仅明显地影响算法的执行时间,还会明显地改变轨迹点 LOD 值,参数 ω 的选择尤为重要. 其次,参数 k 对算法执行性能的影响并不明显,但其对轨迹点的 LOD 值的影响相当明显. 而参数 λ 对算法的影响,不管是在性能还是在 LOD 值,几乎可以忽略(仅当 LOD 值接近边缘值时,产生影响). 因此,对于用户来说,可以根据下面的方式来确定 3 个参数:首先根据应用邻域的特点确定 ω 值,然后再设定基本比较单元的长度参数 k;最后根据用户对稀疏区域轨迹点的局部异常度的要求来设定参数 λ.

5.2　与 TRAOD 的对比和分析

由于 TraLOD 的距离定义与 TROAD 有较大不同,两种计算方法的计算量差别也很大. 因此,在算法性能上两者之间不具有直接的可比性. 因此,本文在实验最后部分仅对 TraLOD 与 TRAOD 在算法效果上进行简单对比和分析. 图 8(a)显示了 TraLOD 使用 LOD=0.7 检测出来的轨迹异常情况,粗线表示轨迹的异常片段. 图 8(b)显示了文献[6]在网上提供的实验结果(http://netfiles.uiuc.

edu/jaegil/www/icde08). 浅色线段表示正常的轨迹,粗线表示异常的轨迹片段. 从两个图中可以发现,TraLOD 不仅可以像 TRAOD 算法一样挖掘出那些明显偏离大多数轨迹运动规律的异常,而且还可以挖掘出那些虽然不明显偏离但具有明显异常的(如图 8 中 2# 异常片段)、运动规律与其局部区域相似但也与别的区域相似(如图 8 中 1# 轨迹片段)的异常部分. 其原因就是 TRAOD 用线段表示轨迹的局部距离,并使用绝对距离(线段 Hausdorff 距离)表示两个线段之间的不匹配性. 这使得 TRAOD 很难检测出像如图 8(b)中 1#,2# 类似的异常轨迹片段. 因此,TraLOD 要比 TROAD 算法更具有现实意义.



(a) TraLOD (b) TROAD

图 8 TraLOD 与 TRAOD 的实验结果对比

6 结束语

随着物联网、3G 技术的迅猛发展,对轨迹数据进行数据挖掘日益成为人们感兴趣的研究课题. 而异常点检测作为数据挖掘的一个重要研究方向,轨迹数据挖掘领域同样具有很重要的理论和实际意义. 为了克服当前流行的、以轨迹片段表示轨迹局部特征所存在的不足,本文提出了以轨迹点表示轨迹局部特征的异常轨迹点检测算法. 该算法不仅引入局部异常度来表示轨迹点的异常情况,而且在距离函数中引入了相对距离的思想,使得距离度量更加符合实际意义. 另一方面,在算法效率角度,TraLOD 采用 R-Tree 索引和局部特征矩阵(由两条轨迹的每对点之间距离组成)来提高算法效率. 实验和性能分析都证实了 TraLOD 不仅具有很高的计算效率,而且按照 LOD 值检测出来的异常轨迹片段具有很高的理论和实际意义.

参 考 文 献

[1] Knorr E M, Ng R T, Tucakov V. Distance-based outliers: Algorithms and applications. VLDB Journal, 2000, 8(3): 237-253

[2] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets//Proceedings of the 2000 ACM SIGMOD International Conference. Dallas, TX, USA, 2000: 427-438

[3] Breunig M M, Kriegel H P, Ng R T, Sander J. LOF: Identifying density-based local outliers//Proceedings of the 2000 ACM SIGMOD International Conference. Dallas, TX, USA, 2000: 93-104

[4] Papadimitriou S, Kitagawa H, Gibbons P B, Faloutsos C. LOCI: Fast outlier detection using the local correlation integral//Proceedings of the 19th International Conference on Data Engineering. Bangalore, India, 2003: 315-326

[5] Aggarwal C C, Yu P S. Outlier detection for high dimensional data//Proceedings of the 2001 ACM SIGMOD International Conference. Santa Barbara, CA USA, 2001: 37-46

[6] Li X, Han J, Kim S, Gonzalez H. ROAM: Rule- and motif-based anomaly detection in massive moving object data sets//Proceedings of the 7th SIAM International Conference on Data Mining. Minneapolis, Minnesota, 2007: 296-307

[7] Lee J, Han J, Li X. Trajectory outlier detection: A partition-and-detect framework//Proceedings of the 24th International Conference on Data Engineering. Cancún, México, 2008: 140-149

[8] Huttenlocher D P, Klanderman G A, Rucklidge W A. Comparing images using the hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993, 15(9): 850-863

[9] Tao Yufei, Papadias Dimitris, Sun Jimeng. The TPR*-tree: An optimized spatio-temporal access method for predictive queries//Proceedings of the 29th International Conference on Very Large Databases. Berlin, Germany, 2003: 790-801

[10] Xiong X, Mokbel M F, Aref W G. LUGrid: Update-tolerant grid-based indexing for moving objects//Proceedings of the 7th International Conference on Mobile Data Management. Nara, Japan, 2006: 13

[11] Beckmann N, Kriegel H P, Schneider R, Seeger B. The R*-Tree: An efficient and robust access method for points and rectangles//Proceedings of the 1990 ACM SIGMOD International Conference. Atlantic City, NJ, 1990: 322-331

LIU Liang-Xu, born in 1974, Ph.D., associate professor. His main research interests include database and data mining, knowledge discovery.

LE Jia-Jin, born in 1951, professor, Ph. D. supervisor. His research interests include database and data warehouse, software engineering theory and practice.

QIAO Shao-Jie, born in 1981, Ph. D., lecturer. His main research interests include database and data mining.

SONG Jia-Tao, born in 1966, Ph.D., professor. His main research interests include image analysis and computer vision.

Background

Along with rapidly development of GPS and Location Service, more and more trajectory data are collected and stored in application servers. How to mine valuable information becomes more and more popular topic in the researches of data mining. As a important researching direction of data mining, outlier detecting is one of important points in trajectory data mining. Current popular approach in trajectory outlier detecting is that trajectory segment is regarded as local feature to detect the outliers in trajectory dataset, such as the method proposed by Li in 2007. However, the length of segment is difficult to define in real application, because different application maybe needs to define different length. Aimed this defect, this paper presents a new framework of trajectory outliers detection, called Trajectory Outliers Detection based on Local Outlying Degree (TraLOD). TraLOD firstly presents the concept of local outlying degree, which expresses outlying degree of target point relative to the points within its neighbor region. Secondly, the distance between two segments is defined according to the idea of Hausdorff distance under translation. Finally, R-Tree and distance feature matrix is introduced to improve algorithm's performance. Detailed experimental results and analysis prove that TraLOD owns more efficient and effective than existing algorithms.

This work is supported in part by National Natural Science Foundation of China under Grant No. 60972163, 61070031, 61100045, Zhejiang Provincial Natural Science Foundation of China under grant Nos. Y1100589 and Y1080123, Youth Foundation for Humanities and Social Sciences of Ministry of Education of China under Grant No. 10YJCZH117, China Postdoctoral Science Foundation under Grant Nos. 20090461346, 201104697, Fundamental Research Funds for the Central Universities under Grant No. SWJTU09CX035, and Natural Science Foundation of Ningbo under Grant No. 2009A610090, No. 2010A610106, and No. 2011A610175.