文章编号:1674-2974(2020)02-0134-07 DOI:10.16339/j.cnki.hdxbzkb.2020.02.018

不确定数据流上的离群点检测处理

朱斌†,钟毓灵,王习特,白梅
(大连海事大学 信息科学技术学院,辽宁 大连 116026)

摘 要:提出了一种快速不确定数据流上的离群点检测算法.采用分层次划分思想给出了适用于流式数据的索引构建方法,并为索引结构中的叶子结点增加了部分存储信息,使得在数据更新时新流入的数据点可以利用中间结果信息直接完成批量过滤,降低计算成本.通过分析离群概率值求解的递推规律,给出了一种全新的离群概率值求解方案,该方案可以最大可能地避免全近邻集合的迭代计算,减少了大量的非离群点计算代价,从而加快处理速度.实验结果表明,快速不确定数据流上的离群点检测算法能够有效地提高检测效率.

关键词:离群点;不确定数据流;滑动窗口;过滤策略;分层次划分
中图分类号:TP391 **文献标志码**:A

Outlier Detection on Uncertain Data Streams

ZHU Bin†,ZHONG Yuling,WANG Xite,BAI Mei
(College of Information Science and Technology,Dalian Maritime University,Dalian 116026,China)

Abstract:This paper proposed a Fast Outlier Detection algorithm Over Uncertain Data Streams (FOD_OUDS). Firstly,an index inspired by hierarchical ideas was designed to manage uncertain data stream,and some storage information was added for the leaf nodes in the index structure,so that the newly inflowed data points can directly perform batch filtering by using the intermediate result information when the data is updated,thereby achieving the purpose of reducing the calculation cost. Secondly,by analyzing the recursive rules of outlier probability values in calculation,a novel outlier probability value solution scheme was presented,which can avoid as much as possible the calculating cost of nearest neighbor set,reduce the processing cost of a large number of inliers,thus speeding up processing. At last,a large amount of experiments show that the FOD_OUDS algorithm can effectively improve the detection efficiency.

Key words:outliers;probabilistic data stream;sliding window;filtering strategy;hierarchical division

离群点检测是数据管理领域的热点问题之一[1],广泛应用于工业损毁、金融诈骗和环境监测等应用场景中,离群点被认为是数据集合中显著区分于其他数据点的数据对象[2].目前,因为基于距离的离群

* **收稿日期**:2019-03-08
基金项目:国家自然科学基金资助项目(61602076,61702072,61976032),National Natural Science Foundation of China(61602076,61702072,61976032);中国博士后科学基金资助项目(2017M611211,2017M621122),China Postdoctoral Science Foundation Funded Projects(2017M611211,2017M621122);辽宁省自然科学基金资助项目(20180540003),Natural Science Foundation of Liaoning Province(20180540003);赛尔网络下一代互联网技术创新项目(NGII20190902),CERNET Innovation Project(NGII20190902);国家重点研发计划项目(2017YFC1404606),National Key R&D Program of China (2017YFC1404606);中央高校基本科研业务费专项基金资助项目(3132019202),Fundamental Research Funds for the Central Universities(3132019202)
作者简介:朱斌(1969—),男,辽宁大连人,大连海事大学副教授
† 通讯联系人,E-mail:dlmuzb@dlmu.edu.cn

点定义[3]能够直观反映离群点本质而得到广泛的应用,其具体描述为:对于数据集合中任意数据点 p,若 p 在半径 r 范围内的邻居个数少于 k 个,那么 p 被认为是离群点.

近年来,数据以高速度高容量的流式形式应用于工业生产、社会生活中,在这规模庞大、速度极快的流式数据里面,不确定性数据广泛存在于其中[4].数据的不确定性主要分为属性级不确定与存在级不确定,本文主要关注存在级不确定数据[5].目前,传统的离群点检测算法尚无法满足诸多现实需求,以气象监测系统为例,传感器不间断地采集局部气温、气压和紫外线指数等环境信息并以流的形式传输到数据库中,实时识别出离群点(异常气象信息),可以有效地防范自然灾害.但是,受到传感器精度及周围环境等因素影响,产生的数据流具有流速较快、规模较大及不确定性等数据特点,使得传统解决方案无法直接应用到上述问题中[5].因此,设计出一种高效的不确定数据流上的离群点检测算法成为本文的主要研究目标.

文献[6]首次给出了存在级不确定数据中的离群点定义,并提出了 DPA 算法用以解决集中式环境中的离群点检测问题.随后,文献[7]在文献[6]的基础上将研究内容扩展至不确定数据流环境中,利用网格索引结构管理不确定数据,并采用动态规划思想来求解离群概率值用以避免可能世界的空间膨胀.但因该算法在批量过滤时不可避免地需要近邻空间的查询,这就使得在处理多维数据时具有一定的局限性,另外,由于其忽略了离群概率值求解的递推规律,使其在概率值求解中也无法避免冗余计算.文献[8]也关注于该研究问题并提出了 PCUOD 算法,该算法通过估算数据点的离群概率范围进行概率剪枝,从而减少了必要的计算成本.但是,由于 PCUOD 算法中的界限估算方法在近邻数目急剧增加时会产生失效的情况,从而也造成了一定的局限性.总之,目前相关解决方案中仍存在诸多不足,无法高效地满足现实应用的需求.

本文主要研究快速不确定数据流上的离群点检测算法(Fast Outlier Detection algorithm Over Uncertain Data Streams,FOD_OUDS),旨在提高算法的执行效率.主要贡献包括以下几个部分:

1)采用分层次划分思想给出了不确定数据流环境中索引的构建方法,利用这种索引结构可以克服传统索引对多维数据管理的局限性.与此同时,本文通过对索引结构中的叶子子块增加部分存储信息,可以快速地完成新到达数据点的批量过滤,极大地减少了数据更新过程中的计算代价.

2)通过深入分析离群概率值求解的递推规律后,提出了一种新的离群概率值求解方法.该方法尽最大可能地避免了全近邻集合的迭代计算,从而极大地减少了冗余计算.

3)利用大量的对比实验,验证本文所提出的 FOD_OUDS 算法的有效性.

1　不确定数据流离群点检测算法

1.1　问题描述

本文主要研究不确定数据流环境中基于距离的离群点检测问题.首先,给出不确定数据流中基于距离的离群点定义;然后,简要描述在基于计数的滑动窗口上的处理流程.表 1 列出了本文使用的符号及其含义.

表 1　符号列表

Tab.1　List of symbols

符号	含义	符号	含义
r	查询半径	$P(\ )$	存在概率
k	查询邻居个数	$P_{\text{Outlier}}(\ )$	离群概率
Threshold	查询阈值	$N(\ )$	近邻集合
p	不确定数据点	NewN( )	后近邻集合
p.label	p 的时间戳	SubN( )	近邻子集合
P	不确定数据集	Outlier( )	离群点集合
DS	不确定数据流	Inlier( )	安全点集合
DS_S	滑动窗口中的数据集	Candidate( )	候选点集合

DS 表示具有 d 维属性的不确定数据流,在 DS 中任意数据点 p 都有一个存在概率 $P(p)(0<P(p)<1)$,其中,$p[i](1\leqslant i\leqslant d)$是点 p 在第 i 个维度上的可度量值.那么,点 p_1、p_2 之间的距离表示为:

$$\text{dist}(p_1,p_2)=\sqrt{\sum_{i\in[1,d]}(p_1[i]-p_2[i])^2} \tag{1}$$

定义 1　(r–近邻)给定数据集 P 和查询半径 r,点 p 在 P 内的近邻集合是 p 在 r 范围内包含的所有点的集合,即 $N(P,p)=\{p'\mid p'\in P,\text{dist}(p,p')<r\}$.

定义 2　((r,k)–离群点)给定数据集 P 和查询邻居个数 k,若点 p 是 P 内的(r,k)–离群点,则 p 在半径 r 范围内的邻居个数小于 k,即$|N(P,p)|<k$.

在数据集 P 中每个可能世界 W 都是 P 的子集.W 的存在概率为:

$$P(W)=\prod_{p\in W}P(p)\times\prod_{p\notin W}(1-P(p))$$

若 Ω 是所有 W 的集合，则有 $\sum_{W\in\Omega}P(W)=1$. 那么,点 p 在 P 中离群的概率表示为:

$$P_{\text{Outlier}}(p)=\sum_{W\in\Omega,p\in\text{Outlier}(W)}P(W) \tag{2}$$

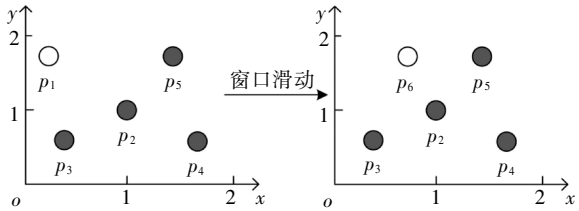
定义 3 (Threshold-离群点)给定查询阈值 Threshold,若点 p 的离群概率 $P_{\text{Outlier}}(p)>\text{Threshold}$,那么 p 是 Threshold-离群点.

可知，所有满足定义 3 的点组成了数据集 P 中的离群集合 $\text{Outlier}(P)=\{p\mid p\in P,P_{\text{Outlier}}(p)>\text{Threshold}\}$.

在不确定数据流 DS 上,采用基于计数的滑动窗口模型管理数据，严格按照数据点 p 到达窗口的先后次序标记 p 的时间戳 p.label. 当窗口大小是 S 时，窗口内的数据集记作 DS_S,窗口内的点的生存周期是 $[p.\text{label},p.\text{label}+S]$. 同时，窗口中保存且仅保存最近到达的 S 个数据点，因此每当窗口中扩充一个新的点 p_{new} 时将对应一个旧的点 p_{old} 消失.

具体描述为，滑动窗口中的不确定离群点查询就是返回当前窗口中所有离群概率大于阈值的数据点的集合,就是 $\text{Outlier}(\text{DS}_S)=\{p\mid p\in\text{DS}_S,P_{\text{Outlier}}(p)>\text{Threshold}\}$.

例 1 图 1(a)(b)分别给出了当前时刻与下一时刻滑动窗口内的数据点集,图 1(c)给出了每个点的存在概率. 假设窗口大小 S=5,查询半径 r=3,查询邻居个数 k=3 和查询阈值 Threshold=0.6，数据点按照 $p_1\sim p_6$ 的次序到达. 以点 p_2 为例，根据上述定义,当前时刻 p_2 的离群概率 $P_{\text{Outlier}}(p_2)\approx0.63$,可知 p_2 是离群点. 下一时刻，随着窗口的滑动点 p_6 到达而点 p_1 消失,p_2 的离群概率变为 $P_{\text{Outlier}}(p_2)\approx0.42$,可知,随着窗口的滑动,下一时刻 p_2 将变为非离群点.



(a)当前时刻 (b)下一时刻

$P(p_1)$=0.3	$P(p_2)$=0.8	$P(p_3)$=0.7
$P(p_4)$=0.6	$P(p_5)$=0.2	$P(p_6)$=0.9

(c)存在概率

图 1 处理流程示例

Fig.1 The example of process flow

1.2 不确定数据流上的离群点检测处理

首先，采用分层次划分索引结构管理不确定流数据;然后,提出了全新的过滤方法;最后,给出了离群点查询动态维护的更新方法.

1.2.1 索引模型

采用分层次划分索引结构管理不确定流数据. 这种索引结构:一方面,可以克服传统索引对多维数据管理的局限性;另一方面,能够避免过多空白子块的产生,减少了存储空间的浪费. 同时,利用划分子块内不确定数据点的特性，可以快速批量过滤数据点,从而加速最终结果的查询.

在文献[9]工作的基础上,采用相同的划分策略构建索引结构. 为便于后续批量过滤,将每个划分子块 b 内的数据点按照存在概率由大到小的顺序排序，并记录块内数据点个数 b.num 和块内空间最大距离 b.dis.

1.2.2 过滤方法

首先,给出了空间数据点的批量过滤方法,利用这种方法可以在遍历划分子块的过程中，通过快速估算出子块内数据点整体的离群概率上界限值来完成批量过滤操作;然后,提出了一种新的离群概率值计算方法用以减少离群概率值的计算代价，该方法尽最大可能地避免了全近邻集合的迭代运算，从而减少了大量的运算成本,提高了算法的运算效率.

批量过滤方法具体的理论依据由引理 1 给出.

引理 1[9] 给定查询半径 r. 在 b.dis < r 的划分子块 b 中，若点 p_1 的存在概率小于点 p_2 的存在概率，那么在 b 中 p_1 的离群概率一定小于 p_2 的离群概率.

批量过滤时,利用引理 1,按照存在概率值由大到小的顺序计算出数据点在划分子块 b 内的离群概率,若某一数据点的离群概率小于 Threshold,那么在子块 b 中存在概率不大于该点的点均为非离群点,由此可以完成划分子块 b 中数据点的批量过滤操作.

离群概率值计算方法通过深入分析离群概率值求解的递推规律后给出了一种新的解决方案，该方案可以最大可能地避免全近邻集合的迭代计算从而减少运算成本的消耗. 具体的理论依据由定理 1 给出.

定理 1 给定不确定数据点 p 和点 p 的近邻集合 $N(p)$,如果 n_MaxSubN(p)是近邻集合 $N(p)$中存在概率最大的 n 个近邻点组成的子集合，那么在所有由 $N(p)$中 n 个近邻点组成的子集合里,点 p 在子集合 n_MaxSubN(p)中成为离群点的概率值最小.

证明:给出查询邻居个数 k,不确定数据点 p 和

点 p 的近邻集合 $N(p)$. 其中,$n_\mathrm{SubN}(p)$是由近邻集合 $N(p)$中 n 个点组成的近邻子集合.

当 $n<k$ 时,点 p 在近邻子集合 $n_\mathrm{SubN}(p)$中成为离群点的概率值等于点 p 自身的存在概率值,即 $P_{\mathrm{Outlier}}(p,n\_\mathrm{SubN}(p))=P(p)$.

当 $n=k$ 时,点 p 在近邻子集合 $n_\mathrm{SubN}(p)$中成为离群点的概率值就等于 $P(p)\times\left\{1-\prod_{p'\in n\_\mathrm{SubN}(p)}P(p')\right\}$.

易知,当 $n=k$ 时,若 $n_\mathrm{SubN}(p)$是由 $N(p)$中存在概率最大的 k 个点组成的子集合, 则点 p 在子集合 $n_\mathrm{SubN}(p)$中成为离群点的概率值最小.

当 $n>k$ 时,假设点 p 在由 $N(p)$中存在概率最大的 n 个点组成的子集合 $n_\mathrm{MaxSubN}(p)$中成为离群点的概率值最小. 那么当 $n_\mathrm{MaxSubN}(p)$中扩充一个数据点 $p'(p'\in N(p)\wedge p'\notin n_\mathrm{MaxSubN}(p))$时,点 p 在新的近邻子集合中成为离群点的概率值为:

$$P(p)\times\left\{\sum_{a=0}^{k-1}P(a\text{-}n\_\mathrm{MaxSubN}(p))\text{-}P(p')\times P((k\text{-}1)\text{-}n\_\mathrm{MaxSubN}(p))\right\}$$

其中:$P(a\text{-}n_\mathrm{MaxSubN}(p))$是子集合 $n_\mathrm{MaxSubN}(p)$中 a 个数据点发生的概率. 由此可见,不确定数据点 p'的存在概率越大, 点 p 在新扩充的子集合中成为离群点的概率值越小. 综上,可证明结论成立.

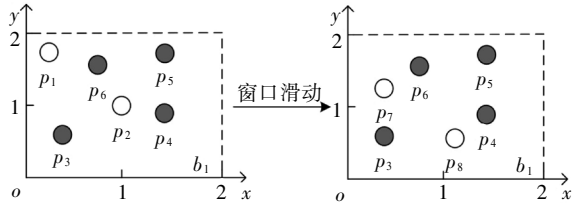
证毕.

由定理 1 可知, 在求解数据点 p 的离群概率值时, 按照近邻集合中近邻点存在概率值由大到小顺序来计算, 可以保证每一次的计算中点 p 在当前近邻集合中成为离群点的概率值都是最小的. 那么,为了快速判定点 p 是否为非离群点,进一步给出引理 2.

引理 2[9]　给定数据点 p 和 p 的近邻集合 $N(p)$, p 的离群概率随着 $N(p)$中点的个数的增加而减少.

由引理 2 可知, 若数据点 p 在其近邻子集合中成为离群点的概率值小于查询阈值, 那么点 p 将是一个非离群点. 也由此可知,根据定理 1 按照近邻集合中近邻点存在概率值由大到小的顺序来求解点 p 的离群概率值, 若点 p 是非离群点则可以在最少的迭代计算中判定出来. 具体示例由例 2 所示.

例 2　图 2 展示了 $b_1.\mathrm{dis}<r$ 的划分子块 b_1 内的数据点集. 假设 $r=3$,$k=3$ 和 Threshold=0.6,按照存在概率由大到小的顺序计算数据点在 b_1 中的离群概率. 首先, 根据定理 1 可以求得 p_4 在近邻子集合 $\mathrm{SubN}(p_4)=\{p_5,p_6,p_2\}$中是非离群点. 否则,按照传统方法, 最坏情况需要计算所有的近邻点才可求得结果. 然后, 根据引理 1 可判定 b_1 中的点均为非离群点. 由此,减少了大量的必要计算代价,并快速完成了批量过滤.



(a)当前时刻　　(b)下一时刻

$P(p_1)=0.5$	$P(p_2)=0.6$	$P(p_3)=0.6$	$P(p_4)=0.8$
$P(p_5)=0.7$	$P(p_6)=0.7$	$P(p_7)=0.7$	$P(p_8)=0.9$

(c)存在概率

图 2　批量过滤示例

Fig.2　The example of batch filtering

1.2.3　*更新方法*

为节省窗口滑动时数据更新所带来的计算成本, 本小节中首先分析了不确定数据流中离群点的性质,并将滑动窗口内的数据进行归类,用以避免对部分数据点的重复计算. 然后,对当前窗口中的叶子划分子块增加了部分存储信息, 使得新到达窗口中的数据点可以利用存储信息直接完成批量过滤,从而达到减少计算成本的目的.

首先, 给出定理 2 用以确定窗口中不可能成为离群点的数据点,以此避免重复计算.

定理 2　给定不确定数据点 p, 若点 p 在后近邻集合 $\mathrm{NewN}(p)$中的离群概率值小于查询阈值,那么点 p 不可能成为离群点.

证明　根据引理 2 可知,若点 p 在近邻子集合中的离群概率值小于查询阈值, 那么点 p 的离群概率值一定小于查询阈值. 又因为 $\mathrm{NewN}(p)$中的近邻点到达窗口都比点 p 晚, 所以若点 p 在后近邻集合 $\mathrm{NewN}(p)$中的离群概率值小于查询阈值,则点 p 将不可能成为离群点.

证毕.

由此更新维护时, 对于满足定理 2 的点将永远不可能成为离群点,也就不需要被更新计算.

具体地,可将当前窗口内的数据 DS_S 分为以下 3 类集合:1)当前窗口内的离群点的集合 $\mathrm{Outlier}(\mathrm{DS}_S)$. 2)当前窗口内是非离群点但随着窗口滑动可能成为离群点的候选集合 $\mathrm{Candidate}(\mathrm{DS}_S)$. 3)所有满足定理 2 的安全点的集合 $\mathrm{Inlier}(\mathrm{DS}_S)$.

然后, 对当前窗口中的叶子子块增加部分存储信息并利用这些存储信息来直接完成新到达数据点的批量过滤. 与此同时,给出了划分子块中批量过滤

的动态维护过程.

对于叶子划分子块将增加部分存储信息,包括 3 个部分:1) 记录 b 内是非离群点且存在概率最大的点 p 的存在概率 b.temp;2) 记录包括点 p 和点 p 按照定理 1 满足它在 b 中的近邻子集合中是非离群点的近邻子集合的集合 b.SubN;3)b.SubN 中最早消失的数据点的时间戳 b.label.

下面,主要介绍批量过滤的动态维护,包括处理失效数据点 p_{old} 和处理新插入数据点 p_{new}.

1) 失效数据点 p_{old} 的处理. 对于 p_{old} 映射到的划分子块 b,若 p_{old} 属于 b.SubN,则需要更新 b 的记录信息并更新 b 中的批量过滤. 若 p_{old} 不属于 b.SubN,则直接删除 p_{old}.

2)新插入数据点 p_{new} 的处理. 检测 p_{new} 映射到的划分子块 b,如果 $P(p_{new})<b$.temp 那么 p_{new} 是非离群点并加入到候选集;如果 b.temp$<P(p_{new})$,那么需要更新 b 的记录信息并更新 b 中的批量过滤, 用以过滤更多数据点.

例 3 展示了利用划分子块的存储信息完成新到达数据点的批量过滤并给出了动态维护的过程.

例 3 图 2(a)(b)分别展示了当前时刻与下一时刻划分子块 b_1 内的数据点集. 假设 $r=3$,$k=3$ 和 Threshold=0.6. 当前时刻 b_1 的记录信息 b.label = 2、b.temp=0.8 和 b.SubN=$\{p_4,p_5,p_6,p_2\}$. 根据引理 1,b_1 内的点均为非离群点,其中,p_1 是安全点,其他点是候选点. 下一时刻,b_1 中点 p_6 和 p_7 到达而点 p_1 和 p_2 消失. 首先处理消失点:当 p_1 消失时,不会影响 b_1 中的过滤; 当 p_2 消失时将重新计算 b_1 中的记录信息,有 b.label=3、b.temp=0.8 和 b.SubN=$\{p_4,p_5,p_6,p_3\}$,此时 b_1 中各点均为候选点. 然后处理新插入点:当插入 p_7 时,因 $P(p_7)<b$.temp 可直接判定 p_7 是候选点;当插入 p_8 时,因 b.temp$<P(p_8)$需要更新 b_1 的记录信息,经计算 b.label=6、b.temp=0.9 和 b.SubN=$\{p_8,p_7,p_6\}$. 此时,b_1 中各点均是非离群点, 其中 p_8、p_7 和 p_6 是候选点而其他点是安全点.

1.3 算法描述

FOD_OUDS 算法描述:

输入:滑动窗口数据集 DN_S,查询阈值 Threshold,查询邻居个数 k,查询半径 r,待删除点 p_{old},待插入点 p_{new};

输出:离群集合 Outlier(DN_S)

1. WHILE p_{new} 插入到当前窗口中 DO
2. IF 滑动窗口已满 DO //处理失效点 p_{old}
3. 删除待消失数据点 p_{old};
4. IF p_{old} 在 b 记录的集合 b.SubN 中 THEN
5. 更新 b 的记录信息,并更新 b 中的批量过滤;
6. ENDIF
7. 集合 D←近邻集合 $N(p_{old})$中未被处理更新的点;
8. FOR 遍历集合 D 中的数据点 p DO
9. 计算属于候选集中的点 p 的离群概率,如果 p 的离群概率大于阈值,那么将 p 移入到离群集中.
10. ENDFOR
11. ENDIF

//处理插入点 p_{new}

12. 根据 $P(p_{new})$将其插入到所映射的划分子块 b 中;
13. IF $P(p_{new})$大于 b 的 b.temp THEN
14. 更新 b 的记录信息,并更新 b 中的批量过滤;
15. ENDIF
16. 集合 D←近邻集合 $N(p_{new})$中未被处理更新的点;
17. FOR 遍历集合 D 中的数据点 p DO
18. 计算属于候选集或离群集中的点 p 的离群概率,若 p 的离群概率小于阈值,根据定理 2,将其加入到候选集或安全集中;
19. ENDFOR
20. IF p_{new} 未被 b 中过滤 THEN
21. 计算 p_{new} 的离群概率,若 p_{new} 的离群概率小于阈值,则将 p_{new} 加入到候选集,否则加入到离群集;
22. ENDIF
23. ENDWHILE

在检测过程中,首先,判断当前滑动窗口内数据是否已满,若是,则每当有新的点 p_{new} 到达窗口时都将对应一个旧的点 p_{old} 失效(算法中行 2),并考虑删除 p_{old} 后对它近邻点和对它映射到划分子块的批量过滤的影响, 其近邻集合中的某个原来属于候选集的点,有可能变为离群点(算法中行 3~行 11). 然后,对于新插入的点 p_{new},一方面需要考虑 p_{new} 的到达对它近邻点和它映射到划分子块的批量过滤的影响, 并做出相应的调整. 另一方面, 检测 p_{new} 是否能被批量过滤,若不能则计算它的最终结果(算法中行 12~22).

2 实验分析

实验采用 C++编程语言实现不确定数据流上的离群点检测算法. 环境配置为 Inter Core i5 3230 2.6 GHz CPU,6 GB 内存,Winsows10 操作系统.

在对比实验中,对本文提出的 FOD_OUDS 算法与 WDPA (Dynamic Programming Algorithm for Window)算法[7]和 PCUOD(Probability Pruning for Continuous Uncertain Outlier Detection)算法[8]分别在真实数据集和人工模拟数据集中进行性能对比. 其中,真实数据集采用的是森林环境监测数据,共包含 120 000 个数据点和 4 个属性维度,其中,每一个属性值均被

映射在0~100 范围内. 由于真实数据并非是概率数据，所以对每一个数据点随机生成一个存在概率值来增加概率属性. 实验中主要对比的是查询时间,表 2 展示了对比实验的实验结果.

表 2　实验结果

Tab.2　Experimental result

参数	查询时间/s		
	FOD_OUDS	PCUOD	WDPA
k=20,r=4	589	781	881
k=20,r=6	435	579	711
k=20,r=8	233	312	655
k=30,r=8	319	401	506

表 2 展示了真实数据集中 3 种算法的性能对比,其中,由于 WDPA 算法采用网格索引结构管理不确定数据并在批量过滤时不可避免地需要近邻空间查询,因而在 4 维数据中的检测代价相对较高. 而对于 PCUOD 算法,由于其在近邻数目较多时过滤性能将会减弱,因而过滤性能相对较低,从而导致需要精确计算的数据点增多使得查询较为缓慢. 相比之下,由于 FOD_OUDS 算法采用分层次划分索引,因而能够较好地管理多维数据，并且在过滤方法中避免了近邻空间的查询，也最大可能地避免了全近邻集合迭代计算,因而拥有较好的处理性能.

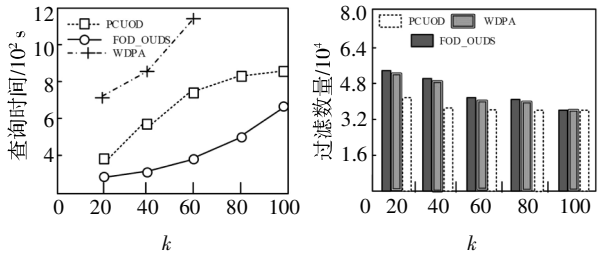
在人工模拟数据集的对比实验中，默认的测试数据具有 5 个维度属性，每个维度属性值被映射到 0~1 000 内,并对每个数据点随机生成一个存在概率值. 实验中主要考察查询邻居个数 k、查询半径 r、数据维度以及窗口大小 S 变化对查询时间和过滤数量的影响. 其中,固定设置查询阈值为 0.6,具体参数如表 3 所示.

表 3　参数设置

Tab.3　Parameter setting

参数	默认值	变化范围
k	20	20~100
r	60	20~100
Window Size	8×10^4	$4 \times 10^4 \sim 12 \times 10^4$
数据维度	5	4 ~ 8

图 3 为查询邻居个数 k 对算法性能的影响. 随着 k 值的增大,3 种算法都需要消耗更多的查询时间并且过滤数量都相应减少. 主要是因为 k 值的增大导致离群点数目增多，使得算法的计算成本相对增加. 通过对比发现,FOD_OUDS 算法的查询时间明显低于另外 2 种算法,这主要是因为 FOD_OUDS 算法的离群概率值计算可以尽最大可能地避免全近邻集合的迭代计算，从而有利于非离群点的高效过滤使得整体查询时间较短.

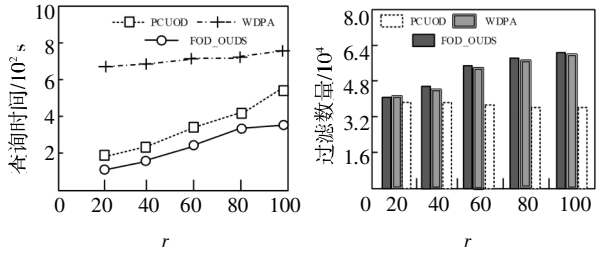


(a)k 与查询时间　　(b)k 与过滤数量

图 3　参数 k 对算法的影响

Fig.3　The effect of k for the algorithm

图 4 为查询半径 r 对算法性能的影响. 随着 r 值的增大,几种算法都需要消耗更多的查询时间. 在过滤数量上,随着 r 值增大,PCUOD 算法的过滤数量逐渐减少,FOD_OUDS 算法和 WDPA 算法的过滤数量逐渐增多. 对于 PCUOD 算法，由于其过滤性能的减弱将直接导致其计算成本增大;对于 WDPA 算法,由于在批量过滤时不可避免地需要近邻空间查询,所以利用网格索引结构维护多维数据会产生非常高昂的空间查询代价. 相对来说,FOD_OUDS 算法利用分层次划分索引在空间查询代价相对较低且批量过滤时并不需要近邻空间查询,所以性能相对较好.



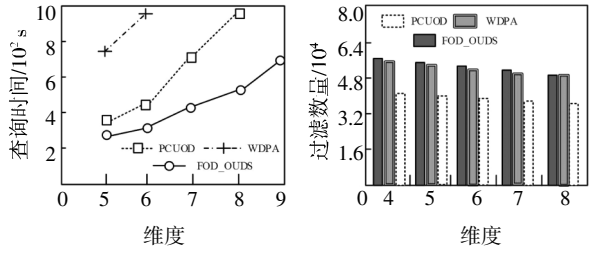
(a)r 与查询时间　　(b)r 与过滤数量

图 4　参数 r 对算法的影响

Fig.4　The effect of r for the algorithm

图 5 为数据维度变化对算法性能的影响. 随着维度的增大,算法的查询时间都明显增大,过滤性能也都减弱. 主要是因为随着维度的增大,空间搜索和索引更新都将更加耗时,但是 FOD_OUDS 算法的处理性能相对较优,主要是因为 FOD_OUDS 算法所采用的索引在多维数据中的搜索能力和更新性能相对较好,并且通过增加索引结构中的部分存储信息,在一次索引映射中就可以直接完成数据点的批量过

滤,也使得其查询时间大幅减少.

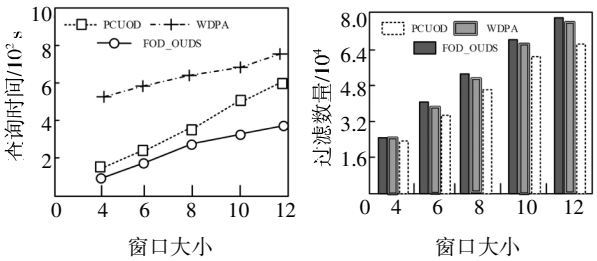


(a)维度与查询时间 (b)维度与过滤数量

图 5 维度对算法的影响

Fig.5 The effect of dimensionality for the algorithm

图 6 为窗口大小变化对算法性能的影响. 随着窗口增大,算法的查询时间都明显增多,这是因数据量的增多增加了计算成本. 同时,过滤数量也明显增多, 主要是因为随着数据量的增大导致非离群点数目逐渐增多,使得几种算法均容易满足过滤条件. 但是,整体性能上 FOD_OUDS 算法较优.



(a)窗口大小与查询时间 (b)窗口大小与过滤数量

图 6 窗口大小对算法的影响

Fig.6 The effect of window size for the algorithm

综上所述,FOD_OUDS 算法在针对不确定数据流环境中的离群点检测问题上的检测时间更短并且过滤性能更优, 从而验证了本文提出的 FOD_OUDS 算法的有效性与高效性.

3 结 论

本文针对不确定数据流环境中的离群点查询问题,提出了 FOD_OUDS 算法. 首先,采用分层次划分思想给出了索引构建策略, 使其具备良好的过滤性能. 然后,在分析了不确定数据点的离群概率值求解的递推规律后, 提出了优先过滤非离群点的概率值求解方法,从而加快了过滤速度. 其次,给出了动态维护的更新方法, 以减少更新过程中的必要计算代价,从而提高了算法的运算效率. 最后,通过实验验证了 FOD_OUDS 算法具有较高的查询效率与较好的过滤性能.

参考文献

[1] SADIK S,GRUENWALD L,LEAL E. Wadjet:Finding outliers in multiple multi-dimensional heterogeneous data streams[C]//2018 IEEE 34th International Conference on Data Engineering. Paris, France:IEEE,2018:1232—1235.

[2] HAWKINS D M. Identification of outliers [J]. Biometrics,1981,37(4):27—41.

[3] KNORR E M,NG R T. Algorithms for mining distance-based outliers in large datasets[C]// Proceedings of the 24th International Conference on Very Large Data Bases. New York:Springer,1998:392—403.

[4] 吴杰,衣枚玉,张金辉,等. 大数据下的结构性态监测信息管理系统设计与应用[J]. 湖南大学学报(自然科学版),2016,43(9):76—81.
WU J,YI M Y,ZHANG J H,*et al*. Design and application of an information management system for structural behavior monitoring based on big data technology [J]. Journal of Hunan University (Natural Sciences),2016,43(9):76—81.(In Chinese)

[5] 周傲英,金澈清,王国仁,等. 不确定性数据管理技术研究综述[J]. 计算机学报,2009,32(1):1—16.
ZHOU A Y,JIN C Q,WANG G R,*et al*. A survey on the management of uncertain data [J]. Chinese Journal of Computers,2009,32(1):1—16. (In Chinese)

[6] YU H,WANG B,XIAO G,*et al*. Distance-based outlier detection on uncertain data [J]. Journal of Computer Research and Development,2010,47(3):474—484.

[7] WANG B,YANG X C,WANG G R,*et al*. Outlier detection over sliding windows for probabilistic data streams [J]. Journal of Computer Science and Technology,2010,25(3):389—400.

[8] CAO K Y,WANG G R,HAN D H,*et al*. Continuous outlier monitoring on uncertain data streams [J]. Journal of Computer Science & Technology,2014,29(3):436—448.

[9] 钟毓灵,王习特,白梅,等. FODU:不确定数据集中快速离群点检测方法[J]. 计算机工程与应用,2019,55(19):105—114
ZHONG Y L,WANG X T,BAI M,*et al*. FODU:A fast outlier detection approach on uncertain data sets[J]. Computer Engineering and Applications,2019,55(19):105—114. (In Chinese)