第 46 卷　第 11 期
Vol. 46　No. 11

计　算　机　工　程
Computer Engineering

2020 年 11 月
November 2020

·先进计算与数据处理·　文章编号:1000-3428(2020)11-0132-07　文献标志码:A　中图分类号:TP391

引入局部向量点积密度的数据流离群点快速检测算法

毛亚琼[1],田立勤[1,2],王　艳[1],毛亚萍[3],王志刚[1]
(1. 青海师范大学 计算机学院,西宁 810008; 2. 华北科技学院 计算机学院,北京 065201; 3. 青海省基础测绘院,西宁 810000)

摘　要:现有数据流离群点检测算法在面对海量高维数据流时普遍存在运算时间过长的问题。为此,提出一种引入局部向量点积密度的高维数据流离群点快速检测算法。以保存少量中间结果的方式只对窗口内受影响的数据点进行增量计算,同时设计 2 种优化策略和 1 条剪枝规则,减少检测过程中各点之间距离的计算次数,降低算法的时空开销,从而提高检测效率。理论分析和实验结果表明,该算法可以在保证检测准确性的情况下有效提高数据流的离群点检测效率,并且可扩展至并行环境进行并行加速。

关键词:离群点检测;高维数据流;局部向量点积密度;增量计算;剪枝规则

开放科学(资源服务)标志码(OSID):

中文引用格式:毛亚琼,田立勤,王艳,等. 引入局部向量点积密度的数据流离群点快速检测算法[J]. 计算机工程,2020,46(11):132-138,147.

英文引用格式:MAO Yaqiong,TIAN Liqin,WANG Yan,et al. Fast outlier detection algorithm in data stream with local density of vector dot product[J]. Computer Engineering,2020,46(11):132-138,147.

Fast Outlier Detection Algorithm in Data Stream with Local Density of Vector Dot Product

MAO Yaqiong[1],TIAN Liqin[1,2],WANG Yan[1],MAO Yaping[3],WANG Zhigang[1]
(1. School of Computer,Qinghai Normal University,Xining 810008,China;
2. School of Computer,North China Institute of Science and Technology,Beijing 065201,China;
3. Qinghai Basic Surveying and Mapping Institute,Xining 810000,China)

【Abstract】 Existing outlier detection algorithms are generally time-consuming to deal with massive high-dimensional data streams. To address the problem,this paper proposes a Fast outlier detection algorithm in data stream with Local Density of Vector dot Product(FASTLDVP). It carries out incremental calculation only for the affected data points in the window,and keeps a small amount of intermediate results. Meanwhile,two optimization strategies and one pruning rule are designed to reduce the number of distance calculation times and the space-time overhead of the algorithm,so as to improve the detection efficiency. Theoretical analysis and experimental results show that this algorithm can effectively improve the detection efficiency of outliers in data stream while ensuring the detection accuracy,and can be extended to parallel environments for parallel acceleration.

【Key words】 outlier detection; high-dimensional data stream; Local Density of Vector dot Product(LDVP); incremental calculation; pruning rule
DOI:10. 19678/j. issn. 1000-3428. 0056453

0　概述

在离群数据中可能隐含有价值的知识,并且其价值甚至可能超过正常数据,因此,离群点检测(也称为异常点检测)是数据挖掘领域一项重要的研究课题。离群点检测最主要的目的是检测数据集中不

基金项目:国家重点研发计划(2017YFC0804108,2018YFC0808306);中央高校基本科研业务费专项资金(3142019043);河北省重点研发计划(19270318D);青海省物联网重点实验室资助项目(2017-ZJ-Y21);青海省应用基础研究项目(2017-ZJ-752);河北省物联网监控工程技术研究中心项目(3142016020)。

作者简介:毛亚琼(1991—),女,硕士研究生,主研方向为数据挖掘;田立勤(通信作者),教授、博士生导师;王　艳,博士;毛亚萍,硕士;王志刚,博士。

收稿日期:2019-10-30　**修回日期**:2020-01-04　**E-mail**:maoyaqiong1@ sina. cn

满足数据一般行为或模式的数据对象[1],其被广泛应用于网络入侵检测、WSN 异常检测、医疗与电网监控[2]等领域。目前传统的离群点检测技术有基于统计、基于深度、基于距离、基于聚类、基于密度、基于分类等多种方法[3]。随着模式识别、机器学习和人工智能技术的不断发展,空间数据、高维数据以及时间序列数据的离群点检测被研究者关注,更多新颖且有效的离群点检测方法被提出,如基于密度偏倚抽样的离群点检测方法[4]、基于分区的离群点检测方法[5]、基于角度分布的离群点检测方法[6]、基于向量点积密度的离群点检测方法[7]、结合模糊粗糙集的检测技术以及结合自组织映射方法的离群点检测技术[8]等。

随着物联网技术的兴起,数据量和维度均呈上升趋势,高速、无限且动态的高维数据流在许多领域产生,同时数据流中数据的分布特征也随时间动态变化。这些特征要求数据流离群挖掘算法在有限内存下检测时仅能用一次或较少次数遍历数据,而传统针对静态数据集的方法需要多次扫描数据集才能完成检测,很难扩展到数据流的离群点挖掘[9]。现有的一些针对数据流的检测方法[10-12]由于"维度灾难"问题而对高维数据流的检测效果欠佳。基于角度[13]和基于局部向量点积密度的算法在高维空间中比基于距离的方法更稳定,但计算复杂度高达$O(d^2n^2)$甚至$O(d^2n^3)$(d为维度,n为数据点个数),计算开销太大。因此,提高高维数据流离群点检测的准确性和计算效率是当前研究的热点。

本文提出一种引入局部向量点积密度的数据流离群点快速检测(Fast outlier detection in data stream with Local Density of Vector dot Product,FASTLDVP)算法。当滑动窗口内有数据更新时,设计一种增量算法,在原有计算结果的基础上,只对受到影响的数据点进行增量更新,从而避免对全部数据点进行重新计算。此外,还设计2种优化策略和1条剪枝规则来减少r-邻域更新时窗口内各点之间距离计算的次数。本文拟通过设计FASTLDVP算法,以较小的计算代价达到与LDVP-OD算法相同的检测效果。

1 相关工作

文献[14]针对静态环境提出一种基于密度的局部离群点检测算法,其通过引入局部异常因子(Local Outlier Factor,LOF)表示每个数据对象的局部异常程度,即数据点的异常程度与其一定范围内邻居(k_{dist}-邻域内的点)分布有关,k_{dist}-邻域密度越大,该点成为离群点的可能性越小。算法中最耗时的是数据集中各数据点之间距离的计算,复杂度为$O(d^2n^2)$。文献[15]通过引入滑动窗口的思想提出n-INCLOF算法,仅更新当前滑动窗口内受影响数据点的LOF值。文献[16]提出一种基于加权聚类的数据流离群点检测算法。该算法对数据流的数据分布特征适应性较强,准确度较高,但是阈值设定较多,可能会对算法的有效性产生较大影响。上述算法能适应分布复杂的数据流,但是当数据流的维度或者k_{dist}值不断增大时,需要进行大量的数据点之间距离的计算。

研究表明,在高维空间中角度比距离更加稳定[17]。文献[6]引入角度方差异常因子(Angle-based Outlier Factor,ABOF)的概念,提出一种基于角度的离群点检测(Angle-based Outlier Detection,ABOD)算法。文献[13]将ABOD算法扩展至动态数据流应用中,提出基于角度的数据流离群点快速检测算法FASTDSABOD,其通过增量计算的方法动态更新每个数据点的ABOF,从而将时间复杂度从$O(d^2n^3)$降至$O(d^2n^2)$。虽然复杂度有所下降,但是其在有限的内存空间中依然无法处理大规模的数据集,并且对数据分布不规则的数据集识别效果欠佳。文献[18]在ABOD算法的基础上提出一种随机投影技术和乘积域AMS Sketch相结合的近似估计角度方差的方法FastVOA。该方法计算复杂度近似线性,但没有考虑任何权重因素的近似理解,虽然降低了计算复杂度,但是不能保证离群检测结果的准确性。

文献[7]提出度量数据离群程度的局部向量点积密度算法LDVP-OD。该算法同时结合密度、距离和角度的离群度量方式的优点,在数据流离群点检测上有较高的准确性,但存在计算复杂度过高的问题(复杂度为$O(d^2n^2)$)。

现有数据流局部离群点检测方法的主要区别在于离群程度度量方法和r-邻域确定方法的差异[19]。上述算法均存在对索引数据结构及邻域方法存在依赖性、重复计算和检测结果准确性依赖于参数设定和计算复杂度过高的问题,难以保证较高的运行效率。本文算法的改进主要是优化邻域的确定方法和离群因子的计算方法。

2 预备知识

局部向量点积密度[7](LDVP)的设计思想为:LDVP越大的数据点离常规簇越近且邻域点数越多,表明所在区域的数据分布越密集;相反,LDVP越小的数据点离常规簇越远且邻域点越少,表明数据分布越异常。给定数据集$D^d \subseteq \mathbb{R}^d$,$|D| = n$,LDVP的相关定义如下:

定义1[7] r-邻域(r-neighborhood)

D^d中数据点o的r-邻域由与o点之间距离小于r的数据点组成,即:

$$N_r(o) = \{o' \in D^d \mid \mathrm{dist}(o,o') \leqslant r\} \tag{1}$$

本文运用基于斜率和最大距离的思想,实时确定滑动窗口中数据点邻域半径的值。设k_{dist}为数据

点 o 与其第 k 个最近邻点的距离,所有数据点的 k_{dist} 降序排列,取“谷底点”对应的 k_{dist} 为邻域半径 r,在一般情况下,k 确定为 4。当滑动窗口向前移动时,旧数据移除同时新数据流入,滑动窗口内数据点的邻域半径 r 变为 r',r'相对于 r 会出现 $r'>r$、$r'<r$ 和 $r'=r$ 3 种情况。

定义 2[7] 向量点积均值(Mean of Vector dot Product,MVP)

设数据点 $o'\in N_r(o)$,$|N_r(o)|=n'$,另随机选取一对数据点 $p_i,p_j\in N_r(o)\setminus\{o'\}$,点 o'与 p_i、p_j 构成以 o'为顶点的两个向量 $\boldsymbol{o'p_i}$,$\boldsymbol{o'p_j}$。点 o'相对于 $N_r(o)$的向量点积均值定义为:

$$\mathrm{MVP}(o')=\frac{1}{\frac{1}{2}(n'-1)(n'-2)}\times\sum_{p_i,p_j\in N_r(o)\setminus\{o'\}}(\cos(\angle p_io'p_j\cdot\|\boldsymbol{o'p_i}\|\|\boldsymbol{o'p_j}\|))=\frac{1}{\frac{1}{2}(n'-1)(n'-2)}\sum_{p_i,p_j\in N_r(o)\setminus\{o'\}}(\boldsymbol{o'p_i}\cdot\boldsymbol{o'p_j})\tag{2}$$

定义 3[7] 局部向量点积密度(LDVP)

数据点 o 的局部向量点积密度定义为:

$$\mathrm{LDVP}(o)=\sum_{o'\in N_r(o)}\mathrm{e}^{-\mathrm{MVP}(o')}\tag{3}$$

3 FASTLDVP 算法

3.1 设计思想

在 LDVP-OD 算法中,每个数据点的 LDVP 值都与所处的 r-邻域有关,当滑动窗口向前移动一位,旧的数据点被移除,新的数据点流入,窗口内数据点的 r-邻域就会发生变化,相关数据点的异常程度也会受到影响。LDVP-OD 算法对此的处理方式是:一旦窗口中有数据被移除或者新数据流入,重新查询窗口内所有数据点的 r-邻域并重新计算 LDVP。然而,由于 LDVP 的局部特性,滑动窗口的更新只会影响到少数数据点的 LDVP 值,而不会影响所有数据点的 LDVP 值。因此,本文提出一种增量算法,只对窗口内受影响数据点的 LDVP 值进行更新,并且受影响的数据点也不需要代价高昂地重新计算,只需在原有结果(包括 r-邻域更新和局部向量点积密度计算)的基础上进行更新查询或累加计算即可。此外,在 r-邻域查询时不可避免地会出现大量的距离计算,尤其当数据量很大或数据维度很高时,计算开销很大。为此,本文在 r-邻域查询时增加 2 个优化策略和 1 条剪枝规则来减少窗口内各点之间距离计算的次数。FASTLDVP 算法流程如图 1 所示。为方便理解,本文将算法分成 r-邻域更新和局部向量点积密度 LDVP 计算两部分分别进行论述。

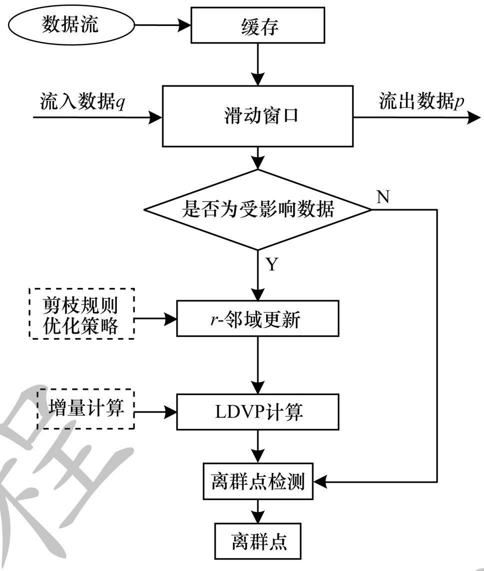

图 1 FASTLDVP 算法流程

Fig. 1 Procedure of FASTLDVP algorithm

3.2 *r*-邻域更新

r-邻域更新的计算量来源于数据点之间的距离计算,本文使用欧式距离作为距离度量的方式。在该阶段,只对滑动窗口内有影响的数据点进行增量更新,并且使用 2 种优化策略缩小数据点邻域查询的范围,使用 1 个剪枝规则减少数据点之间距离计算的次数。

3.2.1 相关定义

定义 4 欧式距离

设 d 维空间中任意两点 $o=(o_1,o_2,\cdots,o_d)$,$o'=(o'_1,o'_2,\cdots,o'_d)$,两点间的欧式距离可定义为:

$$\mathrm{dist}(o,o')=\sqrt{\sum_{k=1}^{d}(o_k-o'_k)^2}\tag{4}$$

定义 5 数据向量的模

数据集中每一个数据点映射到空间中可以看作 1 个 d 维向量,设 $o=(o_1,o_2,\cdots,o_d)$为数据集 D 中的数据点,则点 o 的模定义为:

$$M(o)=\sqrt{\sum_{k=1}^{d}o_k^2}\tag{5}$$

3.2.2 算法描述

以窗口内受影响数据点 o 为例讨论 $N_r{}'(o)$的更新,即判断窗口内某点 $o'(o'\in D)$是否属于 $N_r{}'(o)$ $(o\in D)$。算法主要步骤如下:

步骤 1 移除旧数据点 p 对窗口内数据的影响。仅点 o 的邻域 $N_r(o)$ $(o\in N_r(p),p\in D)$受影响。

步骤 2 将窗口内已有数据点相互之间的影响分为以下 3 种情况:

1)当 $r'>r$ 时,点 o 更新后的邻域为 $N_r{}'(o)=N_r(o)+o'$,$o'\notin N_r(o)$,$o'\in D$。利用优化策略 1 缩小点 o'查询范围,同时使用剪枝规则 1 减小距离计算次数。

优化策略1 新增点 o'分布在 $N_r(o)$ 中这些数据点的周围,即 o'一定分布在 $N_r(o)$ 中数据点的邻域内。因此,只需在 $N_r(o)$ 中数据点的邻域内搜索,即可找到点 o 邻域新增的数据点。

剪枝规则1 由定理1可以得出 $\mathrm{dist}(o,o') \geqslant |M(o)-M(o')|$,则当 $|M(o)-M(o')| > r'$时,必定有 $o' \notin N_r'(o)$,此时可直接对 o'剪枝而无需计算 $\mathrm{dist}(o,o')$。只有当 $|M(o)-M(o')| \leqslant r'$时才需要计算 $\mathrm{dist}(o,o')$,通过 $\mathrm{dist}(o,o') \leqslant r'$进一步判断是否属于邻域。对于 $|M(o)-M(o')|$,$M(o)$ 已知,$M(o')$可以在数据点 o'流入滑动窗口时计算(每个数据点只需计算一次,计算复杂度为 $O(d^2)$,$d \ll n$)。

定理1 设 d 维数据集 D 中任意两个数据点 o、o',$\langle \cdot, \cdot \rangle$表示两向量的点积,则 o、o'之间的距离 $\mathrm{dist}(o,o')$满足:

$$\mathrm{dist}(o,o') \geqslant |M(o)-M(o')| \tag{6}$$

证明:

展开欧氏距离公式(见定义4):

$$\mathrm{dist}(o,o') = \sqrt{\sum_{k=1}^{d}(o_k - o_k')^2} = \sqrt{\sum_{k=1}^{d} o_k^2 - 2\sum_{k=1}^{d} o_k \times o_k' + \sum_{k=1}^{d} o_k'^2}$$

将其代入式(5),得到:

$$\mathrm{dist}(o,o') = \sqrt{M(o)^2 - 2\langle o,o' \rangle + M(o')^2}$$

又由:

$$M(o) \times M(o') = \sqrt{\sum_{k=1}^{d} o_k^2 \times \sum_{k=1}^{d} o_k'^2} = \sqrt{\sum_{k,j=1,k\neq j}^{d}(o_k \times o_j')^2 + \sum_{k=1}^{d}(o_k \times o_k')^2} \geqslant \sum_{k=1}^{d} o_k \times o_k'$$

可得:

$$\sqrt{M(o)^2 - 2(o-o') + M(o')^2} \geqslant \sqrt{M(o)^2 - 2(M(o) \times M(o')) + M(o')^2} = \sqrt{(M(o)-M(o'))^2}$$

因此:

$$\mathrm{dist}(o,o') \geqslant |M(o)-M(o')|$$

证毕。

2)当 $r' < r$ 时,点 o 更新后的邻域 $N_r'(o) = N_r(o) - o'$,$o' \in N_r(o)$。利用优化策略2缩小点 o'查询范围,同时使用剪枝规则1减少距离计算次数。

优化策略2 当邻域半径变小时,$N_r(o)$以外的所有数据点都不可能出现在 $N_r'(o)$内,因此,只需将 $N_r(o)$中满足 $\mathrm{dist}(o,o') > r'$的点 o'删除,即为$N_r'(o)$。

3)当 $r' = r$ 时,窗口中任何数据点的邻域都不受 r'影响。

步骤3 新数据 q 流入时对窗口内数据的影响及自身邻域查询。由于 o 和 q 互为邻域,因此更新 $N_r'(o)$,$o \in D$ 同时可得到 $N_r'(q)$。在距离计算时同样使用剪枝规则1。

r-邻域更新算法代码如下:

输入 D,旧数据 p,新数据 q、r

输出 $N_r'(o)$

```
for each o(o∈D) do
//步骤1,滑动窗口移除旧数据p
for each o∈Nr(p) do
Nr'(o)←Nr(o) - p
Nr'(p)←Nr(p) - o
end for
//步骤2,将窗口内已有点的相互影响分为3种情况
if r' > r do//邻域增大
for each o'∉Nr(o),o'∈D do
if |M(o) - M(o')|≤r' do
dist(o,o') = sqrt(Σ_{k=1}^{d}(o_k - o'_k)^2)
if dist(o,o')≤r' do
Nr'(o)←Nr(o) + o
Nr'(o')←Nr(o') + o//两点互为邻域
end if
end if
end for
end if
if r' < r do//邻域变小
for each o'∈Nr(o),o'≠o do
if |M(o) - M(o')| > r' do
Nr'(o)←Nr(o) - o'
Nr'(o')←Nr(o') - o//两点互为邻域
end if
if |M(o) - M(o')|≤r' do
dist(o,o') = sqrt(Σ_{k=1}^{d}(o_k - o'_k)^2)
if dist(o,o') > r' do
Nr'(o)←Nr(o) - o'
Nr'(o')←Nr(o') - o//两点互为邻域
end if
end if
end for
end if
if r' = r do//邻域不变
nothing need to do
end if
//步骤3,滑动窗口流入数据点q
if |M(o) - M(q)|≤r' do
dist(o,q) = sqrt(Σ_{k=1}^{d}(o_k - q_k)^2)
if dist(o,q)≤r' do
Nr'(o)←Nr(o) + q
Nr'(q)←Nr(q) + o//一个循环,同时可得到新增点
//p的邻域
end if
end if
end for
```

3.3 局部向量点积密度计算

局部向量点积密度(LDVP)计算的过程中存在

大量重复计算。本文通过设计一种增量算法,将 MVP 增量计算公式(见定义 6)中重复计算的部分保存为中间结果,方便下次直接使用。该算法并不改变定义 2 中 MVP 的计算结果,核心思想是用运算规则的分配率合并同类项来避免重复计算,对计算过程做一种简单而有效的优化。

3.3.1 相关定义及推理

定义 6 MVP 增量计算公式

设 d 维数据集 $D^d \subseteq \mathbb{R}^d$,$N_r(o) \in D^d$,$|D^d| = n$,$N_r(o) = n'(n' << n)$,数据点 o',$p_1, p_2, \cdots, p_j \cdots, p_{n'-1} \in N_r(o)$。$N_r(o)$ 中数据点的 d 维向量形式表示为$(p_{j1}, p_{j2}, p_{j3}, \cdots, p_{jd})$,其中,$p_{jk}$ 表示该点向量的第 k 维度。点 o' 相对于 $N_r(o)$ 的 MVP 增量计算公式定义为:

$$\text{ACC-MVP}(o') = \frac{1}{\frac{1}{2}(n'-1)(n'-2)} \sum_{i=2}^{n'} \sum_{k=1}^{d} \left[(o'_k - p_{ik}) \sum_{j=1}^{i-1} (o'_k - p_{jk}) \right] \quad (7)$$

其中,$p_i, p_j \in N_r(o) \setminus \{o'\}$,$p_i \neq p_j$。

推理 1 增量计算公式 ACC-MVP 的计算结果与向量点积均值 MVP 完全等价。

通过给出从 MVP 公式(定义 2)到 ACC-MVP 公式(定义 6)等价的推导过程,可验证此定理,具体如下:

$$\frac{1}{\frac{1}{2}(n'-1)(n'-2)} \sum_{p_i, p_j \in N_r(o) \setminus \{o'\}} (\boldsymbol{o'p}_i \cdot \boldsymbol{o'p}_j) =$$

$$\frac{1}{\frac{1}{2}(n'-1)(n'-2)} \sum_{p_i \in N_r(o) \setminus \{o', p_j\}} \sum_{p_j \in N_r(o) \setminus \{o', p_i\}} \left[(o'_1 - p_{i1}), (o'_2 - p_{i2}), \cdots, (o'_k - p_{ik}) \right] \left[(o'_1 - p_{j1}), (o'_2 - p_{j2}), \cdots, (o'_k - p_{jk}) \right] =$$

$$\frac{1}{\frac{1}{2}(n'-1)(n'-2)} \sum_{p_i \in N_r(o) \setminus \{o', p_j\}} \sum_{p_j \in N_r(o) \setminus \{o', p_i\}} \left[(o'_1 - p_{i1})(o'_1 - p_{j1}) + (o'_2 - p_{i2})(o'_2 - p_{j2}) + \cdots + (o'_k - p_{ik})(o'_k - p_{jk}) \right] =$$

$$\frac{1}{\frac{1}{2}(n'-1)(n'-2)} \sum_{i=2}^{n'} \sum_{k=1}^{d} \left[(o'_k - p_{ik}) \cdot \sum_{j=1}^{i-1} (o'_k - p_{jk}) \right]$$

3.3.2 算法描述

表 1 列出了 FASTLDVP 算法在连续检测期间用于存储中间结果的数据结构。

表 1 增量计算数据结构

Table 1 Data structure for incremental calculation

数据结构	说明
tempart1[n',k]	存储受影响邻域内数据点的中间结果 tempart1 值
tempart[n']	存储受影响邻域内数据点的中间结果 tempart 值
acc-mvp[n']	存储受影响邻域内数据点的 ACC-MVP 值
ldvp[n]	存储所有数据点的 LDVP 值

将 MVP 累加计算公式(见定义 6)中重复计算部分存储在中间结果数据结构中,如式(8)和式(9)所示:

$$\text{tempart1} = \sum_{j=1}^{i-1} (o'_k - p_{jk}) \quad (8)$$

$$\text{tempart} = \sum_{i=2}^{n'} \sum_{k=1}^{d} [(o'_k - p_{ik}) \times \text{tempart1}] \quad (9)$$

由此,点 o' 的向量点积均值通过以下公式计算:

$$\text{ACC-MVP}(o') = \frac{1}{\frac{1}{2}(n'-1)(n'-2)} \times \text{tempart} \quad (10)$$

对于邻域内任意点 o' 的向量点积均值,当邻域内有数据点被删除时,点 o' 的中间结果只更新与删除点 p_i 有关的计算。同样地,当邻域内有数据点新增时,点 o' 的中间结果只更新与新增点 p_j 有关的计算。更新中间结果后使用式(10)计算向量点积均值,最后使用定义 4 对邻域中各数据点的局部向量点积密度进行计算。算法伪代码中没有提及的还有新增点的向量点积密度计算,新增点的计算通过定义 2 和定义 3 进行常规计算。FASTLDVP 算法代码如下:

输入 旧数据 p_i,新数据 p_j;

输出 LDVP(o)

1. for each $o'(o' \in N_r'(o))$
2. if point p_i deleted from $N_r'(o)$ do
3. tempart1 $-= (o'_k - p_{jk})$
4. tempart $-= \sum_{k=1}^{d} [(o'_k - p_{ik}) \cdot \text{tempart1}]$
5. end if
6. If point p_j insert into $N_r'(o)$ do
7. tempart1 $+= (o'_k - p_{jk})$
8. tempart $+= \sum_{k=1}^{d} [(o'_k - p_{ik}) \cdot \text{tempart1}]$
9. end if
10. ACC-MVP$(o') = 1/\frac{1}{2}(n'-1)(n'-2)$ tempart
11. LDVP(o) $+= e^{-\text{ACC-EMP}(o')}$
12. end for

3.4 复杂度分析

3.4.1 r-邻域更新算法复杂度

本文分别从移除旧数据点 p 对窗口内数据的影响、窗口内已有数据点互相之间的影响、新数据 q 流入时对窗口内数据的影响及自身邻域查询 3 个方面

进行 r-邻域更新算法复杂度分析。当 $r' > r$ 时,算法复杂度为 $O(vn) + O(\frac{pn + p'd^2}{2}) + O(n + ud^2)$;当 $r' < r$时,算法复杂度为 $O(vn) + O(\frac{qn + q'd^2}{2}) + O(n + ud^2)$;当 $r' = r$ 时,算法复杂度为 $O(vn) + O(n + ud^2)$。$O(vn)$ 表示更新窗口内受移除点影响的数据点邻域的复杂度;$O(\frac{pn + p'd^2}{2})$ 表示 $r' > r$ 时更新窗口内受影响数据点邻域的复杂度;$O(\frac{qn + q'd^2}{2})$表示 $r' < r$ 时更新窗口内受影响数据点邻域的复杂度;$O(n + ud^2)$ 表示新数据流入时更新窗口内受影响数据点及其自身邻域的复杂度。其中,n 表示滑动窗口大小,d 表示数据的维度,v 代表移除点邻域内的数据点个数,p、q 表示使用优化策略缩小后的查询范围,p'、q'、u 分别表示经过剪枝后需要计算距离的数据点的个数。

当 $v, p, q, p', q', d, n' << n$ 时,上述 3 种情况下算法复杂度均保持近似线性。

3.4.2 LDVP 计算算法复杂度

滑动窗口内所有数据点的 LDVP 计算算法复杂度为 $O(wn'd) + O(n'^2)$,其中,n'表示邻域内数据点个数,w 表示受影响数据点个数。

综上所述,本文利用优化策略缩小受影响数据点邻域查询范围,使得 $v, p, q, p', q', d, w, n' << n$,利用剪枝规则减少距离计算次数,通过推理 1 避免原始 LDVP 计算公式中的重复计算。由此,算法只需扫描数据一次或少量几次即可完成离群检测,同时具有较高的运行效率。

4 实验与结果分析

理论推导本文算法可以在不影响准确率的情况下提高算法效率。本节主要针对 FASTLDVP 算法的运行效率,利用 3 组实验对其进行验证。实验环境为 Pentium® Dual-core CPU,主频 2.2 GHz,内存 4 GB,操作系统为 Window7。实验数据包括仿真数据集和真实数据集。算法采用 Java 语言实现。用文献[20]中所提到的数据集产生方法生成 5 000 条维度为(10,50,100,200)的仿真数据集,分析算法中剪枝过程对不同维度数据的影响,同时使用真实数据集分析算法中剪枝过程对大数据量数据执行效率的影响,并分别对 LDVP-OD[7]、FASTDSABOD[13] 以及文献[16]算法的运行效率进行对比。真实数据集选用 KDD CPU99 数据集中的 5×10^4 条数据。为使评估结果更准确可信,对 FASTLDVP 算法的实验结果取 100 次运行结果的均值。

4.1 剪枝策略和优化策略对算法的影响

本组实验主要探讨 r-邻域更新环节中增加的优化策略和剪枝策略(简称剪枝)对算法带来的影响。实验对比了相同维度不同数据量,以及相同数据量不同维度的数据在增加剪枝过程前后 FASTLDVP 算法执行效率的变化情况。

如图 2 所示,在相同维度($d = 15$)不同数据量的情况下,数据量较小时剪枝前后算法运行时间相差不大,但随着数据量大规模增大,剪枝后算法的运行时间远少于剪枝前算法的运行时间。

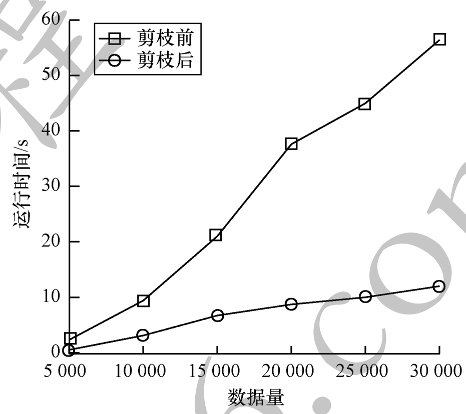

图 2 FASTLDVP 相同维度不同数据量下剪枝前后的运行时间

Fig. 2 Running times of FASTCDVP under different data sizes and same dimension before and after pruning

如图 3 所示,在相同数据量($n = 5\ 000$)不同维度的情况下,剪枝前维度较小算法执行速度快,随着维度增加,距离计算量增加,算法执行效率降低。增加剪枝后,由于剪枝了大量不必要的距离计算,因此运行效率得到提高。

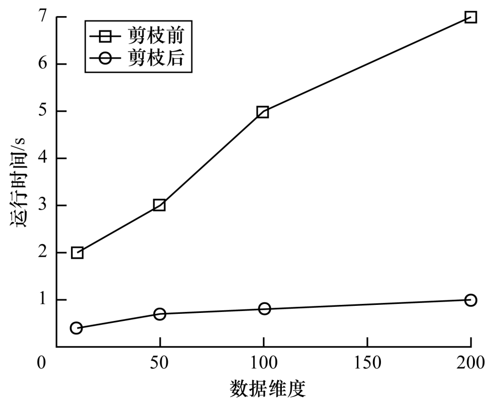

图 3 FASTLDVP 相同数据量不同维度下剪枝前后的运行时间

Fig. 3 Running times of FASTCDVP under same data size and different dimensions before and after purning

4.2 算法整体效率分析

FASTLDVP 算法与 LDVP-OD、FASTDSABOD、文献[16]算法在不同数据量下的运行时间对比如图 4所示。

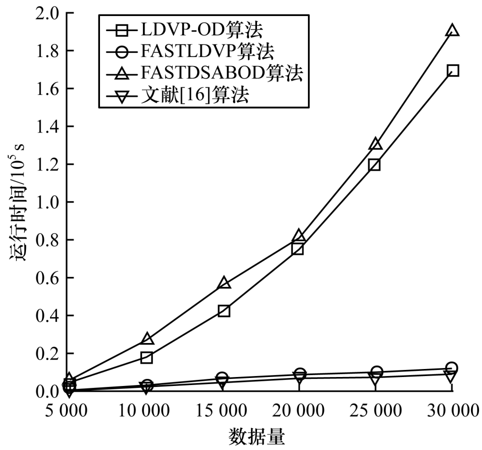

图 4　4 种算法的运行时间对比

Fig. 4　Comparison of running times of four algorithms

图 4 结果表明,当数据量较小时,FASTLDVP 算法和其他算法的运行效率相差不大,但是随着数据量的大规模增加,FASTLDVP 算法的运行效率逐渐高于 LDVP-OD 和 FASTDSABOD 算法。FASTDSABOD 与 LDVP-OD 算法的运行时间更多地消耗在数据点间的距离计算以及复杂公式的重复计算上。FASTLDVP 算法由于只计算受影响数据点,且增加了数据剪枝环节,因此避免了不必要的距离计算和向量点积计算,从而大幅降低了计算复杂度,并且较 LDVP-OD 和 FASTDSABOD 算法,这种优势将随数据量的增加而表现得更明显。FASTLDVP 和文献[16]算法运行时间相近,并且随着数据规模的增大,运行时间的增长也较为相近。但在不同数据规模下,文献[16]算法的运行效率略高于 FASTLDVP 算法。

5　结束语

本文提出一种针对动态数据流的离群点检测算法 FASTLDVP。利用局部离群检测的特点,设计一种增量计算方法,只对受影响数据点进行增量更新,同时通过优化策略和剪枝规则减少邻域查询时数据点之间距离计算的复杂度。理论分析和实验结果表明,该算法能以较小的计算代价达到与 LDVP-OD 算法相同的检测效果,适合于高维数据流的异常检测。由于数据流的数据分布是随时间变化的,因此如何在动态数据流中快速准确地捕捉到概念漂移,将是下一步的研究方向。

参考文献

[1] MANDHARE H C, IDATE S R. A comparative study of cluster based outlier detection, distance based outlier detection and density based outlier detection techniques[C]//Proceedings of 2017 International Conference on Intelligent Computing and Control Systems. Washington D. C., USA: IEEE Press, 2017: 931-935.

[2] GENG Juncheng, ZHANG Xiaofei, ZHOU Qingjie, et al. A low voltage electricity theft identification method based on improved LOF[J]. Advances of Power System & Hydroelectric Engineering, 2019, 35(11): 30-36. (in Chinese)
耿俊成,张小斐,周庆捷,等. 基于局部离群点检测的低压台区用户窃电识别[J]. 电网与清洁能源, 2019, 35(11): 30-36.

[3] MAO Jiali, JIN Cheqing, ZHANG Zhigang, et al. Anomaly detection for trajectory big data: advancements and framework[J]. Journal of Software, 2017, 28(1): 17-34. (in Chinese).
毛嘉莉,金澈清,章志刚,等. 轨迹大数据异常检测:研究进展及系统框架[J]. 软件学报, 2017, 28(1): 17-34.

[4] FU Peiguo, HU Xiaohui. Anomaly detection algorithm based on the local distance of density-based sampling data[J]. Journal of Software, 2017, 28(10): 2625-2639. (in Chinese).
付培国,胡晓惠. 基于密度偏倚抽样的局部距离异常检测方法[J]. 软件学报, 2017, 28(10): 2625-2639.

[5] STONE-GROSS B, COVA M, CAVALLARO L, et al. Your botnet is my botnet: analysis of a botnet takeover[C]//Proceedings of 2009 ACM Conference on Computer and Communications Security. New York, USA: ACM Press, 2009: 635-647.

[6] KRIEGEL H P, SCHUBERT M, ZIMEK A. Angle-based outlier detection in high-dimensional data[C]//Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2008: 444-452.

[7] SHOU Zhaoyu, ZOU Fengbo, TIAN Hao, et al. Outlier detection based on local density of vector dot product in data stream[C]//Proceedings of International Conference on Security with Intelligent Computing and Big-data Services. Berlin, Germany: Springer, 2018: 1-5.

[8] STALMANS E, IRWIN B. A framework for DNS based detection and mitigation of malware infections on a network[C]//Proceedings of IEEE Information Security South Africa Conference. Washington D. C., USA: IEEE Press, 2011: 1-5.

[9] YU Liping, LI Yunfei, ZHU Shixing. Anomaly detection algorithm based on high-dimensional data stream[J]. Computer Engineering, 2018, 44(1): 51-55. (in Chinese)
余立苹,李云飞,朱世行. 基于高维数据流的异常检测算法[J]. 计算机工程, 2018, 44(1): 51-55.

[10] NI Weiwei, LU Jieping, CHEN Geng, et al. An efficient data stream outliers detection algorithm based on k-means partitioning[J]. Journal of Computer Research and Development, 2006, 43(9): 1639-1643. (in Chinese)
倪巍伟,陆介平,陈耿,等. 基于 k 均值分区的数据流离群点检测算法[J]. 计算机研究与发展, 2006, 43(9): 1639-1643.

(下转第 147 页)

（上接第 138 页）
[11] CAO Lei, YANG Di, WANG Qingyang, et al. Scalable distance-based outlier detection over high-volume data streams[C]//Proceedings of the 30th IEEE International Conference on Data Engineering. Washington D. C., USA: IEEE Press, 2014: 76-87.
[12] POKRAJAC D, LAZAREVIC A, LATECKI L J. Incremental local outlier detection for data streams[C]//Proceedings of 2007 IEEE Symposium on Computational Intelligence and Data Mining. Washington D. C., USA: IEEE Press, 2007: 1-10.
[13] YE H, KITAGAWA H, XIAO J. Continuous angle-based outlier detection on high-dimensional data streams[C]//Proceedings of the 19th International Database Engineering and Applications Symposium. New York, USA: ACM Press, 2015: 647-652.
[14] BREUNIG M M, KRIEGEL H P, NG R T, et al. LOF: identifying density-based local outliers[C]//Proceedings of ACM SIGMOD International Conference on Management of Data. New York, USA: ACM Press, 2000: 1-5.
[15] GAO K, SHAO F J, SUN R C. n-INCLOF: a dynamic local outlier detection algorithm for data streams[C]//Proceedings of International Conference on Signal Processing Systems. Washington D. C., USA: IEEE Press, 2010: 179-183.
[16] THAKRAN Y, TOSHNIWAL D. Unsupervised outlier detection in streaming data using weighted clustering[C]//Proceedings of International Conference on Intelligent Systems Design and Applications. Washington D. C., USA: IEEE Press, 2012: 947-952.
[17] KRIEGEL H P, KRÖGER P, ZIMEK A. Outlier detection techniques[C]//Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press, 2010: 1-5.
[18] PHAM N, PAGH R. A near-linear time approximation algorithm for angle-based outliner detection in high dimensional data[C]//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2012: 877-885.
[19] ZHANG Ji, GAO Qigang, WANG Hai. Outlier detection for high-dimensional data streams[C]//Proceedings of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2015: 37-46.
[20] HAN J W, MICHELINE K. Data mining: concepts and techniques[M]. 2nd edition. [S. l.]: Elsevier, 2006.

编辑　金胡考