基于密度的 Top-n 局部异常点快速检测算法

刘 芳 [1] 齐建鹏 [1] 于彦伟 [1] 曹 磊 [2] 赵金东 [1]

摘 要 局部异常检测 (Local outlier factor, LOF) 能够有效解决数据倾斜分布下的异常检测问题, 在很多应用领域具有较好的异常检测效果. 本文面向大数据异常检测, 提出了一种快速的 Top-n 局部异常点检测算法 MTLOF (Multi-granularity upper bound pruning based top-n LOF detection), 融合索引结构和多层 LOF 上界设计了多粒度的剪枝策略, 以快速发现 Top-n 局部异常点. 首先, 提出了四个更接近真实 LOF 值的上界, 以避免直接计算 LOF 值, 并对它们的计算复杂度进行了理论分析; 其次, 结合索引结构和 UB_1、UB_2 上界, 提出了两层的 Cell 剪枝策略, 不仅采用全局 Cell 剪枝策略, 还引入了基于 Cell 内部数据对象分布的局部剪枝策略, 有效解决了高密度区域的剪枝问题; 再次, 利用所提的 UB_3 和 UB_4 上界, 提出了两个更加合理有效的数据对象剪枝策略, UB_3 和 UB_4 上界更加接近于真实 LOF 值, 有利于剪枝更多数据对象, 而基于计算复用的上界计算方法, 大大降低了计算成本; 最后, 优化了初始 Top-n 局部异常点的选择方法, 利用区域划分和建立的索引结构, 在数据稀疏区域选择初始局部异常点, 有利于将 LOF 值较大的数据对象选为初始局部异常点, 有效提升初始剪枝临界值, 使得初始阶段剪枝掉更多的数据对象, 进一步提高检测效率. 在六个真实数据集上的综合实验评估验证 MTLOF 算法的高效性和可扩展性, 相比最新的 TOLF (Top-n LOF) 算法, 时间效率提升可高达 3.5 倍.

关键词 异常检测, 局部异常检测, Top-n, 剪枝策略

引用格式 刘芳, 齐建鹏, 于彦伟, 曹磊, 赵金东. 基于密度的 Top-n 局部异常点快速检测算法. 自动化学报, 2019, **45**(9): 1756−1771

DOI 10.16383/j.aas.c180425

开放科学 (资源服务) 标识码 (OSID):

A Fast Algorithm for Density-based Top-n Local Outlier Detection

LIU Fang[1] QI Jian-Peng[1] YU Yan-Wei[1] CAO Lei[2] ZHAO Jin-Dong[1]

Abstract Local outlier factor (LOF) effectively addresses the problem of outlier detection in skewed datasets, which has been shown remarkable detection performance in variety of applications. In this paper, we propose an efficient Top-n local outlier detection algorithm, called MTLOF (Multi-granularity upper bound pruning based top-n LOF detection), for fast detecting top-n local outliers in large-scale datasets. First, we propose four LOF upper bounds that are closer to the real LOF value to avoid the computations of LOF values, and analyze their computational complexity theoretically. Second, by combining with index structure and the upper bounds UB_1 and UB_2, we propose a two-layer Cell pruning strategy, not only adopting the global Cell pruning strategy, but also introducing a local pruning strategy based on the internal data objects of Cells, which effectively prunes the high-density regions. Third, we propose two more reasonable and effective data object pruning strategies using the proposed upper bounds UB_3 and UB_4. UB_3 and UB_4 are closer to the real LOF value, which benefits to pruning more data objects. On the other hand, the upper bound calculation method based on computation reuse greatly reduces the computational cost. Finally, we optimize the selecting method of initial Top-n local outliers leveraging the established index structure. Specifically, we select the initial Top-n local outliers in sparse regions, which is conducive to selecting the data objects with a larger LOF value as the initial local outliers. Experimental study on six real-world datasets demonstrates the efficiency and scalability of our proposed MTLOF — up to 3.5 times faster than the state-of-the-art TOLF (Top-n LOF) method.

Key words Outlier detection, local outlier detection, Top-n, pruning strategy

Citation Liu Fang, Qi Jian-Peng, Yu Yan-Wei, Cao Lei, Zhao Jin-Dong. A fast algorithm for density-based Top-n local outlier detection. *Acta Automatica Sinica*, 2019, **45**(9): 1756−1771

近年来, 随着各类智能移动设备的广泛普及, 社

收稿日期 2018-06-13 录用日期 2018-09-10
Manuscript received June 13, 2018; accepted September 10, 2018
国家自然科学基金 (61773331, 61403328, 61703360), 山东省自然科学基金 (ZR2016FM42), 山东省高等学校科技计划 (J17KA091) 资助
Supported by National Natural Science Foundation of China (61773331, 61403328, 61703360), Shandong Provincial Natural Science Foundation (ZR2016FM42), and A Project of Shandong Province Higher Educational Science and Technology Program (J17KA091)
本文责任编委 黎铭
Recommended by Associate Editor LI Ming
1. 烟台大学计算机与控制工程学院 烟台 264005 中国 2. 麻省理工学院计算机科学与人工智能实验室 剑桥 MA 02139 美国
1. School of Computer and Control Engineering, Yantai University, Yantai 264005, China 2. CSAIL, Massachusetts Institute of Technology, Cambridge MA 02139, USA

交网络、网上购物、移动支付、位置服务等新兴应用不断涌现, 各类海量大数据被采集和处理, 而面向这些大数据的挖掘分析服务已俨然成为一大独具特色的新兴产业. 异常检测作为数据挖掘最重要的任务之一, 在网络监测、信用卡欺诈、电信诈骗、金融证券服务、电子商务等各种应用领域都被认为是至关重要的内容. 因此, 异常检测在学术界与工业界都受到了越来越多的关注, 在大数据与人工智能应用中, 异常检测也发挥了越来越重要的作用, 例如在北京, 通过对地铁和公交乘车记录的异常检测分析可帮助安保系统识别出潜在的小偷[1].

异常检测旨在从海量数据中识别出与大多数数据样本差异较大的数据对象. 目前已存在很多异常检测研究工作[2−4], 如基于距离的异常检测[5−7]、基于邻居的异常检测[8−10]、基于分布的异常检测[11−13]和基于聚类的异常检测[14−15] 等. 然而, 这些异常检测方法都无法处理数据倾斜分布下的异常检测问题, 因为在倾斜分布的数据中, 不同区域的异常数据可能具有不同的数据特征, 而上述方法都采用全局的异常标准来处理数据对象. 基于密度的 LOF[16] 有效解决了在数据倾斜分布下的异常检测问题. 局部异常检测利用每个数据对象相对于其周围邻居的相对密度衡量异常因子, 这样的相对密度反映了局部的数据分布, 也就是说, 对异常的检测是相对于局部数据的, 因此可以处理倾斜分布下的异常检测问题. 在实际应用中, 尤其是在大数据系统中, 数据的分布往往是倾斜的, 基于 LOF 的局部异常检测方法相比其他检测方法在很多应用领域都表现出了较好的异常检测效果[17−18].

局部异常检测方法[16] 需要计算每个数据对象的 LOF 异常因子, 然后通过排序找出 LOF 值较大的数据点作为异常对象, 而 LOF 定义了相对密度, 要求查找每个数据的 k 近邻及可达距离, 因此检测计算成本非常高, 很难满足对大规模数据的异常检测效率需求. 最新研究[19] 针对 Top-n 局部异常点检测, 提出了一种基于 LOF 上界剪枝的检测算法 (Top-n LOF, TOLF), 利用 Cell 索引和 LOF 上界对数据对象进行剪枝, 较大地提升了局部异常点检测的效率. 尽管如此, TOLF 在 Cell 剪枝中仅采用了一个全局的两点间最小距离 $cp_{\min}$ 进行剪枝, 当全局 $cp_{\min}$ 较小时, 将严重影响 Cell 剪枝效果, 甚至失效, 此外, 针对数据对象剪枝的 LOF 上界相比真实 LOF 值较大且计算复杂度较高, 使得剪枝效果也有限.

针对这些问题, 本文提出了一种改进的 Top-n 局部异常点检测算法 MTLOF, 融合索引结构和多层 LOF 上界设计剪枝策略, 实现了高效的局部异常点检测. 1) 为避免直接计算 LOF 值, 提出了四个更接近真实 LOF 值的 LOF 上界 ($UB_1 - UB_4$), 并对它们的计算复杂度进行了理论分析; 2) 利用索引结构和 UB_1、UB_2 上界, 提出了两层的 Cell 剪枝策略, 不仅采用全局 Cell 剪枝策略, 还引入了基于 Cell 内部数据对象分布的局部剪枝策略, 有效解决了高密度区域的剪枝问题; 3) 利用提出的 UB_3 和 UB_4 上界, 提出了两个更加合理有效的数据对象剪枝策略, UB_3 和 UB_4 上界更加接近于真实 LOF 值, 有利于剪枝更多数据对象, 而基于计算复用的 LOF 上界计算方法, 大大降低了计算成本; 4) 优化了初始候选 Top-n 局部异常点的选择方法, 利用均匀区域划分和建立的索引结构, 在数据分布的稀疏区域选择初始局部异常点, 有利于选择 LOF 值较大的数据对象作为初始局部异常点, 有效提升初始临界值 (Cutoff threshold) ct, 使得在初始阶段剪枝掉更多的 Cell 和数据对象. 5) 在六个不同维度的真实数据集上的综合实验评估验证了 MTLOF 算法的高效性和可扩展性, 相比最新的 TOLF 算法, 提升的效率可高达 3.5 倍.

本文结构如下: 第 1 节讨论相关工作; 第 2 节定义基本概念和问题; 第 3 节给出详细的检测算法; 第 4 节进行实验验证和分析; 第 5 节总结全文.

1 相关工作

Breunig 等[16] 最早提出了局部异常因子 LOF 的概念, 相对基于距离的异常检测[5−6] 和 KNN 异常检测[9], LOF 采用相对密度衡量每个数据对象的异常程度, LOF 越高表示数据对象相对于其邻居的密度差异较大, 异常的可能性也就越高. 异常通常只是数据集的极少部分, 因此 Jin 等[20] 首次提出了 Top-n 局部异常的概念, 从数据集中选取 n 个 LOF 值最大的数据对象, 即为 Top-n 局部异常. 传统 LOF 挖掘方法[16] 主要分为两步: 首先计算出所有数据对象的 LOF 值, 然后对所有数据按 LOF 值降序排序, 前 n 个数据对象即为 Top-n 局部异常. 文献 [20] 首先利用 Birch 算法[21] 对数据对象进行聚类, 利用聚簇的半径和聚簇间的距离关系计算每个聚簇的 LOF 界值, 对聚簇进行排序, 然后在最可能包含异常的聚簇中检测 Top-n 局部异常点. 虽然该方法通过聚类方法剪枝掉了部分数据对象, 但是数据预处理计算成本昂贵, 不适用于大数据的处理. 此外, 算法的剪枝策略也具有较大局限性, 并没有表现很好的剪枝效果.

另外一类相关工作就是 LOF 的变种方法, Tang 等[22] 提出了一种基于连通性的异常检测方法 (Connectivity-based outlier factor, COF), COF 首先利用最小生成树衡量每个数据对象与它 k 近邻的连通度, 然后与 LOF 相似, 利用相对 k 近邻连通

度定义每个数据对象的 COF. 为了优化基于最小生成树聚类的异常检测, 朱利等[23] 提出了一种快速构建最小生成树的优化方法, 可同时检测基于距离的全局异常和基于密度的局部异常, 但是这类基于扫描树的聚类方法无法处理大规模数据, 如文献 [23] 中性能验证, 在处理 1 800 个数据点时已消耗近 60 秒. Papadimitriou 等[24] 提出了一种局部相关度方法 (Local correlation integral, LOCI), LOCI 采用一个区域半径 r 定义数据对象的本地邻居区域, 以代替 k 近邻. 虽然 LOCI 相比 LOF 具有较低计算复杂度, 但是在数据倾斜分布下, 通过固定区域半径定义局部异常并不合理, 可能导致稀疏区域的数据对象没有邻居点而高密度区域的数据对象包含过多的邻居点. 杨宜东等[25] 为了减少计算时间, 还提出了一种动态网格划分的数据流下的 LOCI 异常检测方法. Zhang 等[26] 提出了一种基于距离的局部异常点检测 (Local distance-based outlier factor, LDOF), LDOF 采用数据对象到其 k 近邻距离的均值定义数据对象的局部密度, 然后相对 k 近邻计算相对密度获得异常因子. 与文献 [26] 相似, Krieget 等[27] 则使用数据对象到其 k 近邻距离的平方和的均值来定义局部密度. 最近, Schubert 等[28−29] 又提出了一种更为简单的 LOF 变种方法, 称为 Simplified-LOF, 该方法直接使用 k-距离代替可达距离, 也就是说, 直接使用 k-距离的倒数定义局部密度. 该方法虽然简化了 LOF 方法, 但是仅考虑每个数据对象到第 k 个近邻的距离, 将导致异常检测效果严重依赖于参数 k 的选取. 此外, Liu 等[30] 将局部异常检测扩展到了不确定数据领域上, 研究了在概率密度表示的不确定数据模型上的异常检测方法. 最近 Cao 等[31−32] 还考虑了在属性级不确定数据上的 Top-n 局部异常点检测方法.

Liu 等[33−34] 提出一种基于隔离树的快速异常检测算法 (Isolation forest, Iforest), 该方法首先随机采样 m 个数据对象样本, 构建多棵 Itree 隔离树 (Isolation trees), 然后对每个数据对象来遍历这些 Itree, 根据数据对象经过的平均路径长度来判断是否为异常对象. Iforest 虽具有线性的时间复杂度, 但并不适用于特别高维数据, 因为每次切割数据空间都是随机选取一个维度, 建立隔离树后仍有大量维度信息没有使用, 导致算法可靠性降低. 此外, Iforest 算法也仅对全局异常敏感, 不擅长处理局部的相对稀疏点, 即本文处理的局部异常点. 为了加快 Top-n 局部异常点检测, 最新研究[19] 提出了一种基于 Cell 索引和 LOF 上界剪枝的 TOLF 算法, TOLF 首先将数据集划分成多个相对均匀的区域, 并对相对密集的区域建立 Cell 索引, 然后利用 Cell 索引和 LOF 上界对 Cell 和数据对象进行剪枝. 尽管 TOLF 有效提升了 Top-n 局部异常点的检测效率, 但是采用全局最小距离的 Cell 剪枝策略具有一定局限性, 当存在较多过密区域时 (最小距离非常小), 将严重影响 Cell 剪枝效果, 甚至失效. 此外, TOLF 所采用的面向单个数据对象剪枝的 LOF 上界相比数据对象真实 LOF 值较大, 导致数据对象的剪枝空间有限.

2 问题定义

本节首先介绍 LOF[16] 相关定义, 然后给出 Top-n 局部异常点检测的问题定义.

基于密度的局部异常根据每个数据对象本地的密度与它 k 近邻的密度的比值判断该数据对象是否是一个局部异常点. 对于两个数据对象 p 和 q, 本文使用 $dist(p,q)$ 表示它们间的距离. 所有数据对象集合表示为 D.

定义 1 (k 近邻). 给定数据对象 p 和任意正整数 k, 数据对象 p 的 k 近邻集合由到 p 距离最近的 k 个数据对象组成, 表示为 $\mathcal{N}_k(p)$.

定义 2 (k-距离). 给定数据对象 p 和任意正整数 k, 距离 p 第 k 个最近的数据对象记为 q_k, p 的 k-距离则是 p 到 q_k 的距离, 记为 $dist_k(p)$.

也就是说, 对于任意 $q \in \mathcal{N}_k(p)$, $dist(p,q) \leq dist_k(p)$.

定义 3 (可达距离). 给定数据对象 p 和 $q \in \mathcal{N}_k(p)$, 数据对象 p 相对于 q 的可达距离 $dist_r(p,q)$ 定义如下: $dist_r(p,q) = \max\{dist(p,q), dist_k(q)\}$.

根据定义 3, 如果对象 p 也是 q 的 k 近邻, 也就是说 $p \in \mathcal{N}_k(q)$, 则 p 相对于 q 的可达距离就是 $dist_k(q)$; 反之, 则为两对象的距离 $dist(p,q)$.

定义 4 (k 近邻可达距离和). 给定数据对象 p, 数据对象 p 的 k 近邻可达距离和 $dist_{kr}(p)$ 定义如下: $dist_{kr}(p) = \sum_{q\in\mathcal{N}_k(p)} dist_r(p,q)$.

定义 5 (局部可达密度). 数据对象 p 的局部可达密度 (Local reachability density, LRD) 表示为:

$$LRD(p) = \left[\frac{\sum\limits_{q\in\mathcal{N}_k(p)} dist_r(p,q)}{|\mathcal{N}_k(p)|}\right]^{-1} = \left[\frac{dist_{kr}(p)}{|\mathcal{N}_k(p)|}\right]^{-1}$$

从定义 5 可知, $LRD(p)$ 就是数据对象 p 相对于其 k 近邻的可达距离的均值的倒数, 也就是说, 局部可达密度主要通过数据对象相对于其 k 近邻的可达距离来估计它的局部密度. 数据对象相对于其 k 近邻的可达距离均值越小, 也就是说相对越密集, 它的局部可达密度就越高. 接下来根据 $LRD(p)$ 来定

义数据对象的局部异常因子 LOF.

定义 6 (局部异常因子). 数据对象 p 的局部异常因子表示为:

$$LOF(p)=\frac{\sum\limits_{q\in\mathcal{N}_k(p)}\frac{LRD(q)}{LRD(p)}}{|\mathcal{N}_k(p)|}$$

很明显, $LOF(p)$ 就是数据对象 p 的所有 k 近邻的局部可达密度与 p 的局部可达密度比值的均值. 因此, $LOF(p)$ 越接近于 1, 说明对象 p 与其 k 近邻的局部密度越接近, p 的异常程度越小; $LOF(p)$ 小于 1 时, 则说明对象 p 处于一个高密度区域. 相反, $LOF(p)$ 值越大, 则表示 p 是异常点的可能性越大. 所以本文的关注点在于快速检测出具有较高 LOF 的数据对象, 下面给出本文所要解决的 Top-n 局部异常点检测的问题定义.

问题 1 (Top-n 局部异常点检测). 给定数据集合 D, 对象近邻数 k, 和异常点数量 n, Top-n 局部异常点检测则是返回数据集 D 中 LOF 值最大的前 n 个数据对象.

3 基于密度的 Top-n 局部异常点检测

本节将详细介绍本文所提出的 Top-n 局部异常点检测的优化算法, 首先介绍基于临界值剪枝的检测算法, 然后介绍优化算法用到的四个 LOF 上界, 接着介绍基于 LOF 上界的剪枝策略, 最后给出优化的检测算法.

3.1 基于临界值剪枝的检测算法

为了获取 LOF 值最大的前 n 个数据对象, 即 Top-n 局部异常点, 传统的基本方法将计算出所有数据对象的 LOF 值, 然后按照 LOF 值排序, 返回前 n 个数据对象. 然而, 对于 Top-n 局部异常点, 并不需要计算出所有数据对象的 LOF 值, 我们仅关心 LOF 值最大的 n 个数据对象, 因此, 只需要维护 LOF 值最大的 n 个数据对象即可.

给定数据集 D, 从中随机选取 n 个数据对象, 并计算它们的 LOF 值, 选取最小的 LOF 值作为临界值 (Cutoff threshold, ct). 可以发现, 对于数据集 D 中任意的数据对象 p, 如果 $LOF(p)<ct$, 则 p 不可能是 Top-n 局部异常点, 可以直接被剪枝掉; 如果 $LOF(p)>ct$, 则 p 可能是 Top-n 局部异常点, 将 p 放入维护队列, 剔除队列中 LOF 最小的数据对象, 并更新临界值 ct. 依次遍历一遍数据集, 即可获取到 Top-n 局部异常点. 这种剪枝检测方法我们称之为基于临界值剪枝的检测算法.

从定义 6 可知, 直接计算数据对象 LOF 值的成本较高, 为了更高效地检测出 Top-n 局部异常点, 我们将从四个方面优化基于临界值剪枝的检测方法: 1) 如何快速获取到 LOF 值较大的 n 个数据对象作为初始维护的候选异常对象; 2) 选取更合理有效的 LOF 值上界进行剪枝判断, 以避免直接计算数据对象的 LOF 值; 3) 如何使用数据对象的这些 LOF 值上界, 使得计算量尽量小, 且被剪枝掉的数据对象数量尽量多; 4) 如何结合索引技术和 LOF 上界快速剪枝掉高密度区域的所有数据对象.

3.2 LOF 上界

本小节将首先介绍数据对象 LOF 值的四个上界, 然后分析四个上界的计算时间复杂度.

3.2.1 四个 LOF 上界

根据定义 5 和定义 6 可以得出:

$$\begin{aligned}LOF(p)=&\frac{\sum\limits_{q\in\mathcal{N}_k(p)}LRD(q)}{|\mathcal{N}_k(p)|\cdot LRD(p)}=\\&\frac{1}{LRD(p)}\cdot\frac{\sum\limits_{q\in\mathcal{N}_k(p)}LRD(q)}{k}=\\&\frac{1}{LRD(p)}\cdot\sum_{q\in\mathcal{N}_k(p)}\frac{1}{\sum\limits_{o\in\mathcal{N}_k(q)}dist_r(q,o)}=\\&\frac{dist_{kr}(p)}{|\mathcal{N}_k(p)|}\cdot\sum_{q\in\mathcal{N}_k(p)}\frac{1}{dist_{kr}(q)}\end{aligned}$$

$LOF(p)$ 被表示成了两个部分的乘积, 第一部分表示 p 的局部可达密度的倒数, 第二部分表示 p 所有 k 近邻的 k 近邻可达距离和的倒数和. 下面由定理 1 引出数据对象 p 的第一个 LOF 上界 $UB_1(p)$.

定理 1 (LOF 的上界一). 给定数据集 D 中的一个数据对象 p, p 的 LOF 值满足以下上界:

$$LOF(p)\le UB_1(p)=\frac{dist_{kr}(p)}{|\mathcal{N}_k(p)|\cdot cp_{\min}}$$

其中, $cp_{\min}$ 表示 D 中所有数据对象间最小的距离.

证明. 由于 $cp_{\min}$ 是 D 中所有数据对象间最小的距离, 因此 $\forall p,q$, $dist_r(p,q)\ge cp_{\min}$; $\sum_{o\in\mathcal{N}_k(q)}dist_r(q,o)\ge|\mathcal{N}_k(q)|\cdot cp_{\min}$, 所以 $\sum_{q\in\mathcal{N}_k(p)}[\sum_{o\in\mathcal{N}_k(q)}dist_r(q,o)]^{-1}\le\sum_{q\in\mathcal{N}_k(p)}[|\mathcal{N}_k(q)|\cdot cp_{\min}]^{-1}=[cp_{\min}]^{-1}$; 因此, $LOF(p)=dist_{kr}(p)/|\mathcal{N}_k(p)|\cdot\sum_{q\in\mathcal{N}_k(p)}[\sum_{o\in\mathcal{N}_k(q)}dist_r(q,o)]^{-1}\le dist_{kr}(p)/|\mathcal{N}_k(p)|\cdot cp_{\min}=UB_1(p)$ 成立. □

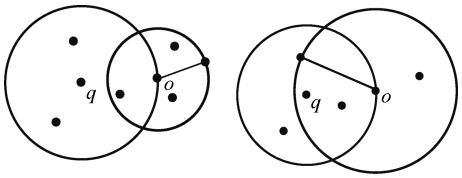
定理 2 (LOF 的上界二). 给定数据集 D 中的一个数据对象 p, p 的 LOF 值满足以下上界:

$$LOF(p)\le UB_2(p)=\frac{dist_{kr}(p)}{|\mathcal{N}_k(p)|}\cdot\frac{1}{\overline{dist}}$$

其中 $\overline{dist}=\min_{q\in\mathcal{N}_k(p)}\sum_{o\in\mathcal{N}_k(q)}dist(q,o)/|\mathcal{N}_k(p)|$.

证明. 因为上界 $UB_2(p)$ 的第一部分与 $LOF(p)$ 的第一部分相同, 本证明只需证明第二部分即可. 由于 $dist_r(q,o) = \max\{dist(q,o), dist_k(o)\}$, 所以 $dist(q,o) \leq dist_r(q,o)$; 如图 1 所示, 设定 $k=4$, 图 1 (a) 中, $dist_k(o) < dist(q,o)$, 所以 $dist_r(q,o) = dist(q,o)$, 图 1 (b) 中, $dist_k(o) > dist(q,o)$, 所以 $dist_r(q,o) = dist_k(o) > dist(q,o)$. 因此, $\sum_{o\in\mathcal{N}_k(q)} dist(q,o) \leq \sum_{o\in\mathcal{N}_k(q)} dist_r(q,o)$. 因为 $\min_{q\in\mathcal{N}_k(p)} \sum_{o\in\mathcal{N}_k(q)} dist(q,o)$ 表示在数据对象 p 的所有 k 近邻对象中, 到其 k 近邻的距离和的最小值, 所以 $\forall q \in \mathcal{N}_k(p)$, $dist_{kr}(q) \geq \min_{q\in\mathcal{N}_k(p)} \sum_{o\in\mathcal{N}_k(q)} dist(q,o)$, 即 $[dist_{kr}(q)]^{-1} \leq [\min_{q\in\mathcal{N}_k(p)} \sum_{o\in\mathcal{N}_k(q)} dist(q,o)]^{-1}$. 因此, $\sum_{q\in\mathcal{N}_k(p)} [dist_{kr}(q)]^{-1} \leq |\mathcal{N}_k(p)| / \min_{q\in\mathcal{N}_k(p)} \sum_{o\in\mathcal{N}_k(q)} dist(q,o) = [\min_{q\in\mathcal{N}_k(p)} \sum_{o\in\mathcal{N}_k(q)} dist(q,o)/|\mathcal{N}_k(q)|]^{-1} = \overline{dist}^{-1}$. □

很明显, $\overline{dist}$ 表示数据对象到其 k 近邻的距离和的均值, 因此, $\overline{dist} \geq cp_{\min}$, 所以 $LOF(p) \leq UB_2(p) \leq UB_1(p)$, 也就说, $UB_2(p)$ 相比 $UB_1(p)$ 更加接近于 p 的 LOF 值.



(a) $dist_k(o) < dist(q,o)$ (b) $dist_k(o) > dist(q,o)$

图 1 $dist_r(q,o)$ 与 $dist(q,o)$ 的关系示例

Fig. 1 The relationships between $dist_r(q,o)$ and $dist(q,o)$

定理 3 (LOF 的上界三). 给定数据集 D 中的一个数据对象 p, p 的 LOF 值满足以下上界:

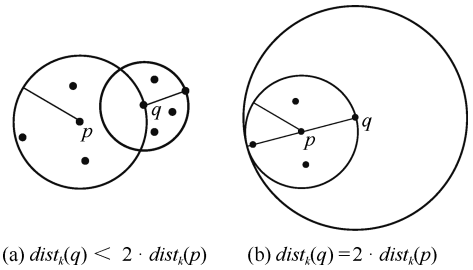
$$LOF(p) \leq UB_3(p) = 2\cdot dist_k(p)\cdot \sum_{q\in\mathcal{N}_k(p)} \frac{1}{\sum\limits_{o\in\mathcal{N}_k(q)} dist(q,o)}$$

证明. 由定理 2 的证明可知, $\sum_{o\in\mathcal{N}_k(q)} dist(q,o) \leq \sum_{o\in\mathcal{N}_k(q)} dist_r(q,o)$, 所以, $\sum_{q\in\mathcal{N}_k(p)} [\sum_{o\in\mathcal{N}_k(q)} dist_r(q,o)]^{-1} \leq \sum_{q\in\mathcal{N}_k(p)} [\sum_{o\in\mathcal{N}_k(q)} dist(q,o)]^{-1}$.

接下来只需证明 $dist_{kr}(p)/|\mathcal{N}_k(p)| \leq 2\cdot dist_k(p)$ 即可. 设定 $direct_{\max}(p) = \max\{dist_r(p,q)|q\in\mathcal{N}_k(p)\}$[16], 所以 $direct_{\max}(p)$ 大于等于对象 p 的 k 近邻可达距离和的均值, 即 $dist_{kr}(p)/|\mathcal{N}_k(p)| \leq direct_{\max}(p)$, 所以只需证明 $direct_{\max}(p) \leq 2\cdot dist_k(p)$ 即可.

$direct_{\max}(p)$ 是取对象 p 相对于它的 k 近邻最大的可达距离, 根据定义 3 可知, 这等价于求解 $\forall q \in \mathcal{N}_k(p)$, 返回 $dist(p,q)$ 和 $dist_k(q)$ 的最大值, 所以只需证明 $\max_{q\in\mathcal{N}_k(p)} dist(p,q) \leq 2\cdot dist_k(p)$ 和 $\max_{q\in\mathcal{N}_k(p)} dist_k(q) \leq 2\cdot dist_k(p)$ 即可.

由定义 2 可知, $\forall q \in \mathcal{N}_k(p)$, $dist(p,q) \leq dist_k(p)$, 所以 $\max_{q\in\mathcal{N}_k(p)} dist(p,q) \leq 2\cdot dist_k(p)$; 对于 $q \in \mathcal{N}_k(p)$ 来说, 它的 k 近邻分布可分为图 2 (a) 和图 2 (b) 两种情况, 一种情况为 q 附近有很多数据对象, q 的 k-距离小于 $2\cdot dist_k(p)$, 甚至小于 $dist_k(p)$, 如图 2 (a) 所示; 另一种情况为 q 附近存在较少的数据对象, 在最极端情况下, 不存在任何数据对象, 但是, 当 q 搜索邻居的范围扩大到 $2\cdot dist_k(p)$ 时, 一定能够涵盖 p 以及 p 的 k 近邻, 如图 2 (b) 所示, 也就说, $dist_k(q) \leq 2\cdot dist_k(p)$. 因此, $\max_{q\in\mathcal{N}_k(p)} dist_k(q) \leq 2\cdot dist_k(p)$. □



(a) $dist_k(q) < 2\cdot dist_k(p)$ (b) $dist_k(q) = 2\cdot dist_k(p)$

图 2 $dist_k(p)$ 与 $dist_k(q)$ 的关系示例

Fig. 2 The relationships between $dist_k(p)$ and $dist_k(q)$

接下来, 我们给出最后一个更加接近于 LOF 值的上界 $UB_4(p)$.

定理 4 (LOF 的上界四). 给定数据集 D 中的一个数据对象 p, p 的 LOF 值满足以下上界:

$$LOF(p) \leq UB_4(p) = \frac{dist_{kr}(p)}{|\mathcal{N}_k(p)|}\cdot \sum_{q\in\mathcal{N}_k(p)} \frac{1}{\sum\limits_{o\in\mathcal{N}_k(q)} dist(q,o)}$$

证明. $UB_4(p)$ 的第一部分 $dist_{kr}(p)/|\mathcal{N}_k(p)|$ 与 $LOF(p)$ 一致, 第二部分 $\sum_{q\in\mathcal{N}_k(p)} [\sum_{o\in\mathcal{N}_k(q)} dist(q,o)]^{-1} \geq \sum_{q\in\mathcal{N}_k(p)} [dist_{kr}(q)]^{-1}$ 已经在定理 3 中证明, 因而, 定理 4 得证. □

由定理 3、定理 4 及它们证明过程可以得出, $LOF(p) \leq UB_4(p) \leq UB_3(p)$.

3.2.2 时间复杂度分析

根据定理 1∼4, 四个上界的大小顺序为: $LOF(p) \leq UB_4(p) \leq UB_2(p) \leq UB_1(p)$ 和 $LOF(p) \leq UB_4(p) \leq UB_3(p)$. 由于 $UB_3(p)$ 的第一部分相对于 $UB_1(p)$、$UB_2(p)$ 的第一部分变

大, 而第二部分相对变小, 因此, 不能确定 $UB_3(p)$ 和 $UB_1(p)$、$UB_2(p)$ 的相对大小.

设定数据集 D 中数据对象总数为 N, 一般情况下, $k \ll N$, 查询每个数据对象 k 近邻的时间复杂度为 $\mathrm{O}(N\log k)$, 计算 $dist_{kr}(p)$ 时, 首先查询到 p 的 k 个近邻, 时间复杂度为 $\mathrm{O}(N\log k)$, 然后求出所有 k 近邻的 k 近邻, 时间复杂度为 $\mathrm{O}(k\cdot N\log k)$, 求得 k 个可达距离并相加, 时间复杂度为 $\mathrm{O}(k)$, 所以计算 $dist_{kr}(p)$ 的复杂度为 $\mathrm{O}[(k+1)N\log k+k]$. 计算 $\overline{dist}$ 和 $\sum_{q\in\mathcal{N}_k(p)}[\sum_{o\in\mathcal{N}_k(q)}dist(q,o)]^{-1}$ 时, 同样, 首先获得 p 的 k 近邻, 然后求每个 k 近邻 q 的 k 近邻, 并计算每个 q 与其近邻的距离和, 时间复杂度为 $\mathrm{O}[(k+1)\cdot N\log k+k]$, 最后, 求取 k 个距离和中的最小值或倒数和, 时间复杂度为 $\mathrm{O}(k)$, 因此, $\overline{dist}$ 和 $\sum_{q\in\mathcal{N}_k(p)}[\sum_{o\in\mathcal{N}_k(q)}dist(q,o)]^{-1}$ 的时间复杂度为 $\mathrm{O}((k+1)N\log k+2k)$.

1) $UB_1(p)$: 获取全局 $cp_{\min}$ 的时间复杂度为 $\mathrm{O}(N^2)$; 但整个检测算法仅需计算一次, 所以平均到每个数据对象的计算时间为 $\mathrm{O}(N)$; 计算 $dist_{kr}(p)$ 的时间复杂度为 $\mathrm{O}((k+1)N\log k+k)$, 因此, 计算 $UB_1(p)$ 的平均复杂度为 $\mathrm{O}(UB_1(p))=\mathrm{O}(N+(k+1)N\log k+k)$.

2) $UB_2(p)$: 根据 $dist_{kr}(p)$ 和 $\overline{dist}$ 的时间复杂度知, $UB_2(p)$ 的时间复杂度为 $\mathrm{O}(2(k+1)N\log k+3k)$.

3) $UB_3(p)$: 根据 $dist_k(p)$ 和 $\sum_{q\in\mathcal{N}_k(p)}[\sum_{o\in\mathcal{N}_k(q)}dist(q,o)]^{-1}$ 的时间复杂度知, $UB_3(p)$ 的时间复杂度为 $\mathrm{O}((k+2)N\log k+2k)$.

4) $UB_4(p)$: 根据 $dist_{kr}(p)$ 和 $\sum_{q\in\mathcal{N}_k(p)}[\sum_{o\in\mathcal{N}_k(q)}dist(q,o)]^{-1}$ 的时间复杂度知, $UB_4(p)$ 的计算时间复杂度为 $\mathrm{O}(2(k+1)N\log k+3k)$.

5) $LOF(p)$: 获取 $dist_{kr}(p)/|\mathcal{N}_k(p)|$ 的复杂度为 $\mathrm{O}((k+1)N\log k+k)$, 所以计算 $LRD(p)$ 的复杂度为 $\mathrm{O}((k+1)N\log k+k+1)$; 计算 p 每个 k 近邻 q 的 LRD 的时间复杂度为 $\mathrm{O}(k\cdot N\log k+k+1)$; 因此, 计算 $LOF(p)$ 的时间复杂度为 $\mathrm{O}((k+1)N\log k+k+1)+k\cdot\mathrm{O}(k\cdot N\log k+k+1)+\mathrm{O}(k)\approx\mathrm{O}((k^2+k+1)N\log k+k^2+3k)$.

根据以上时间复杂度分析, $LOF(p)$ 的时间复杂度远大于四个上界的时间复杂度, 因此, 计算上界的时间远小于计算 LOF 值的时间.

3.3 融合索引和 LOF 上界的剪枝方法

尽管利用 LOF 上界剪枝数据对象可减少计算成本, 但是对于每个数据对象仍需计算 k 近邻可达距离和. 为了快速剪枝掉高密度区域的数据对象, 下面介绍融合 Cell 索引和 LOF 上界的剪枝方法, 无需对每个数据对象进行计算即可剪枝掉高密度区域内的所有数据对象.

3.3.1 基于 Cell 的全局剪枝

对于某一高密度区域内的数据对象, 如果能够保证所有数据对象的 LOF 值上界 UB_i 小于临界值 ct, 则该区域内的所有数据对象都可以直接被剪枝掉. 给定边长 len_{side}, 将整个数据空间按照 len_{side} 为单位长度划分, 得到的每个子空间划分称为一个 Cell, 如图 3 所示, 包括了 9 个 Cell, 中间 Cell 记为 C.

考虑使用上界 $UB_1(p)$, 给定一个高密度的 Cell C, 如果对于 $\forall p\in C$, $UB_1(p)<ct$, 则该 Cell 中所有的数据对象都可以直接被剪枝掉.

引理 1 (基于 Cell 的全局剪枝). 给定一个 Cell, 记为 C, LOF 剪枝临界值 ct, 如果 C 包含的数据对象多于 k 个, 并且其边长 $len_{\mathrm{side}}\le ct\cdot cp_{\min}/(2\sqrt{d})$ (d 为数据的维度), 那么 C 中所有的数据对象可以直接被剪枝.

证明. 由定理 1 可知, $LOF(p)\le UB_1(p)=dist_{kr}(p)/|\mathcal{N}_k(p)|\cdot cp_{\min}$, 只需证明 $\forall p\in C$, $UB_1(p)\le ct$ 即可, 也就是证明 $dist_{kr}(p)/|\mathcal{N}_k(p)|\le ct\cdot cp_{\min}$.

由于 C 包括多于 k 个对象, 所以对于任一对象 $p\in C$ 都可以在 Cell 对角线 $\sqrt{d}\cdot len_{\mathrm{side}}$ 范围内找到 k 近邻, 即 $dist_k(p)\le\sqrt{d}\cdot len_{\mathrm{side}}$; 对于 p 的 k 近邻, 在最坏情况下, 都可以在 $2\sqrt{d}\cdot len_{\mathrm{side}}$ 范围内找到 k 近邻, 即 $\forall q\in\mathcal{N}_k(p)$, $dist_k(q)\le 2\sqrt{d}\cdot len_{\mathrm{side}}$, 如图 3 所示, 当 p 处于 C 的右上角时 (极端情况), 假设 C 中仅包含 $k+1$ 个数据点, p 的 k 近邻可能会取到 C 外的 q 点, 在最坏情况下, q 仍能在 $2\sqrt{d}\cdot len_{\mathrm{side}}$ 范围内找到 k 个近邻. 因此, $dist_{kr}(p)/|\mathcal{N}_k(p)|\le direct_{\max}(p)\le 2\sqrt{d}\cdot len_{\mathrm{side}}$.

如果 $len_{\mathrm{side}}\le ct\cdot cp_{\min}/(2\sqrt{d})$, 则 $2\sqrt{d}\cdot len_{\mathrm{side}}\le ct\cdot cp_{\min}$, 那么 $dist_{kr}(p)/|\mathcal{N}_k(p)|\le ct\cdot cp_{\min}$. □

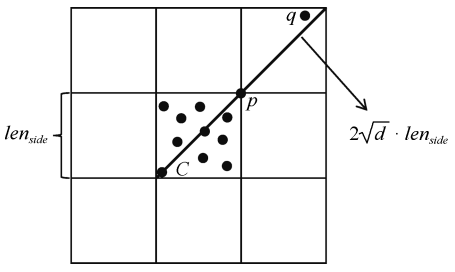


图 3　基于 Cell 索引的剪枝示例

Fig. 3　An example of pruning based on Cell index

传统 Cell 划分方法将整个数据空间按照全局的边长划分, 从引理 1 可知, 高密度区域的剪枝条件除

了与 Cell 内的数据对象数量有关, 还要求 Cell 的边长不大于 $ct \cdot cp_{\min}/(2\sqrt{d})$. 很明显, 该边长条件与 $cp_{\min}$ 有关, 当全局 $cp_{\min}$ 较小时, 将严重影响被剪枝掉的高密度区域的数量.

基于上述考虑, 本文采用文献 [19] 提出的均匀区域生成方法, 首先将整个数据集按照数据对象分布划分成几个相对独立的数据分布相对均匀的区域, 每个区域独自处理数据对象, 即分区自治. 具体的划分方法分为两步, 1) 首先将整个数据空间看成根节点, 然后按照二叉树迭代地划分数据空间, 直到每个叶子节点至少包括 k 个数据对象且不可再分; 2) 从叶子节点向上合并节点, 如果两个子节点内部数据对象间最小的距离 $cp^1_{\min}$ 和 $cp^2_{\min}$ 的大小比例小于 $diff$, 即 $\max\{cp^1_{\min}, cp^2_{\min}\}/\min\{cp^1_{\min}, cp^2_{\min}\} < diff$, 则合并这两个子节点, 直到不能再向上合并, 一个独立的区域被生成. 通过设定适当的比例 $diff$, 可以将两个分布相似的子节点合并, 因此, 可以得到相对分布均匀的区域. 如图 4 所示, 根据数据密度分布生成 4 个均匀区域, 每个区域内即可采用一个 $cp^{A_i}_{\min}$ 执行基于 Cell 的全局剪枝策略.

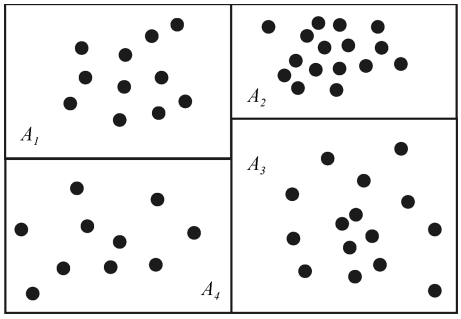


图 4 区域划分示例

Fig. 4 An example of area partition

虽然基于 Cell 索引方法可以用于剪枝掉高密度的区域, 但 Cell 索引采用统一的边长划分数据空间, 虽然实现简单, 但是限制了大块的高密度区域的剪枝, 除此之外, 固定的边长也不够灵活, 不便于检验不同密度的高密度区域, 例如, 给定边长下的某个 Cell 内少于 k 个数据对象, 但是若适当增加边长 (仍满足剪枝限定条件), 该 Cell 即可包含多于 k 个数据对象. 因此, 本文在每个数据对象较多的区域内 ($|A_i| \geq t \cdot k$), 建立一颗 Rtree 索引, 使用简单的层次索引 Rtree 代替 Cell 索引, Rtree 索引从上向下索引数据对象, 每个叶子节点包含小于等于 k 个数据对象. 虽然 Rtree 索引的节点区域为矩形, 但是可以按矩形中较长的边等同于 Cell 的边长来处理, 上述的引理 1 依然可以适用. Rtree 索引为层次索引结构, 节点的矩形大小不固定, 这样, 我们就可以从上向下快速剪枝掉最大块的符合边长条件的高密度区域. 为了便于描述, 在下文仍采用 Cell 来表述 Rtree 索引中的一个树节点.

3.3.2 基于 Cell 的局部剪枝

从引理 1 可以看出, 对于被剪枝掉的 Cell 需要满足全局条件 $len_{\text{side}} \leq ct \cdot cp_{\min}/(2\sqrt{d})$. 虽然该方法只需计算出 $cp_{\min}$, 但是仅考虑到上界 $UB_1(p)$, 忽略了 Cell 内部数据对象的分布情况, 下面给出考虑到 Cell 内数据对象分布的局部剪枝方法.

引理 2 (基于 Cell 的局部剪枝). 给定一个 Cell, 记为 C, LOF 剪枝临界值 ct, 如果 C 包含的数据对象多于 k 个, 并且其边长 $len_{\text{side}} \leq ct \cdot \overline{kd}_{\min}/(2\sqrt{d})$, 那么 C 中所有的数据对象可以直接被剪枝, 其中 $\overline{kd}_{\min} = \min_{p \in C} \sum_{q \in \mathcal{N}_k(p)} dist(p,q)/|\mathcal{N}_k(p)|$.

证明. 与引理 1 相似, 只需证明 $\forall p \in C$, $LOF(p) \leq UB_2(p) \leq ct$ 即可, 也就是证明 $dist_{kr}(p)/|\mathcal{N}_k(p)| \leq ct \cdot \overline{dist}$; 参见引理 1 证明, 得出 $dist_{kr}(p)/|\mathcal{N}_k(p)| \leq direct_{\max}(p) \leq 2\sqrt{d} \cdot len_{\text{side}}$.

由于 $\overline{dist} \geq \overline{kd}_{\min}$, 如果 $len_{\text{side}} \leq ct \cdot \overline{kd}_{\min}/(2\sqrt{d})$, 则 $2\sqrt{d} \cdot len_{\text{side}} \leq ct \cdot \overline{kd}_{\min} \leq ct \cdot \overline{dist}$, 那么 $dist_{kr}(p)/|\mathcal{N}_k(p)| \leq ct \cdot \overline{dist}$. □

可以看出, 引理 1 采用了上界 $UB_2(p)$ 对 Cell 进行剪枝处理, 虽然剪枝的边长条件放松了, 但是该剪枝需要计算 Cell 内部所有数据对象的 k 近邻距离和的均值. 因此, 本文设计了一个两层的 Cell 剪枝策略, 首先使用边长条件 $len_{\text{slide}} \leq ct \cdot cp_{\min}/(2\sqrt{d})$ 进行剪枝, 若不能被剪枝掉, 再使用边长条件 $len_{\text{slide}} \leq ct \cdot \overline{kd}_{\min}/(2\sqrt{d})$ 进一步地进行剪枝判断.

3.4 基于 LOF 上界的数据对象剪枝

经过两层 Cell 的剪枝检测后, 对于没有被剪枝掉的 Cell, 为了避免直接计算 LOF 值, 将首先对每个数据对象进行基于上界的剪枝判断, 若不能被剪枝, 再计算 LOF 值, 最后判断是否为 Top-n 异常候选项, 若是, 则更新临界值 ct.

基于 $UB_3(p)$ 和 $UB_4(p)$ 的数据对象剪枝: 根据引理 1 和引理 2, 被剪枝的 Cell 内的数据对象一般满足小于等于上界 $UB_1(p)$ 和 $UB_2(p)$, 因此, 对于不满足 Cell 剪枝条件的数据对象, 需要采用更加小的上界来进行剪枝. 根据 3.2.2 节分析可知, $UB_4(p) \leq UB_2(p) \leq UB_1(p)$, $UB_4(p) \leq UB_3(p)$, 可使用 $UB_4(p)$ 进一步对数据对象进行剪枝检测, 但是, 计算 $UB_4(p)$ 的时间复杂度是 $UB_3(p)$ 的近两倍. 此外, 计算每个数据对象 p 的 $UB_3(p)$ 时, 可直接复用之前对该 Cell 进行局部剪枝时已计算过的 $\overline{kd}_{\min}$, 也就是说, 可直接复用之前计算过的每个数据对象的 k 近邻距离和结果直接获得该 Cell 内每个对象的 $UB_3(p)$. 因此, 我们先使用上界 $UB_3(p)$ 对

数据对象进行剪枝检测, 若不满足剪枝条件, 再使用更小的上界 $UB_4(p)$ 进行剪枝检测.

计算复用: 除了计算 $UB_3(p)$ 时可直接复用之前的计算结果, 在计算 $UB_4(p)$ 同样可以复用之前计算 $UB_3(p)$ 和 $\overline{kd}_{\min}$ 时的结果. 对于最终未能被上界剪枝的数据对象, 在计算真实 LOF 值时, 也可以复用之前已经计算得到所有数据对象的 k 近邻和 k-距离. 因此, 本文的检测算法对每个数据对象最多仅执行一次 k 近邻查询.

3.5　选取初始候选局部异常点

第 3.2 节～3.4 节详细描述第 3.1 节提出的优化方法, 第 3.2 节回答了可利用哪些 LOF 上界进行剪枝判断, 而避免直接计算 LOF 值, 第 3.3 节回答了如何结合索引技术和 LOF 上界快速剪枝高密度区域内的所有数据对象, 第 3.3 和第 3.4 节共同回答了如何使用 LOF 上界, 使得上界的计算尽量小, 而剪枝掉的数据对象尽量多, 接下来本节来回答第一个优化问题: 如何快速获取到 LOF 值较大的 n 个对象作为初始维护的候选异常对象?

初始 Top-n 候选异常对象的选取严重影响着检测算法的执行效率, 当选择的初始数据对象的 LOF 值偏大时, 可快速剪枝掉大量的 Cell 区域或数据对象, 但是, 当选择的初始数据对象包括高密度区域的数据对象时, 将导致初始临界值 ct 非常低, 初始阶段将几乎没有 Cell 或数据对象被剪枝. 因此, 在选择初始 Top-n 候选异常对象时应尽量避免选取高密度区域的数据对象, 显然随机选择方法并不适合, 因为在 LOF 异常检测场景中, 通常情况下异常对象是极少数的, 随机选择方法更容易选取到非异常对象或高密度区域内的数据对象.

本文针对采用的区域划分和索引结构选取初始 Top-n 候选异常对象, Rtree 索引利用空间区域索引数据对象, 每个节点记录了所包含的数据对象及覆盖的区域. 对于高密度区域, 通常节点包括数据对象较多, 且覆盖区域较小, 而局部异常点通常分散在稀疏区域. 因此, 本文首先在区域划分中, 选择 $cp_{\min}^{A_i}$ 较大且数据对象较少的区域, 在这些区域内随机选取初始 Top-n 局部异常点; 如果这些区域不足 n 个数据对象, 接着在 Rtree 索引的叶子节点中, 根据节点的覆盖区域, 选择区域最大的 Top-$\lceil \frac{n}{k} \rceil$ 个节点, 在这些区域和叶子节点内部随机选取 n 个节点作为初始 Top-n 局部异常点.

3.6　MTLOF: Top-n 局部异常点检测算法

本节给出本文提出的基于多粒度上界剪枝的 Top-n 局部异常点检测算法 (Multi-granularity upper bound pruning based top-n LOF detection, MTLOF), MTLOF 算法的伪代码如算法 1 所示.

算法 1. MTLOF 算法

输入. Dataset D, the number of nearest neighbors k, top outliers n, parameters $diff$ and t.

输出. Top-n LOF outliers.

1) $Set_{\text{area}} \leftarrow GenerateArea(D, diff)$.
2) **for** $A_i \in Set_{\text{area}}$ **do**
3)　　**if** $|A_i| \geq t \cdot k$
4)　　$rtree \leftarrow Rtree(A_i, k)$; **then**
5)　　$Set_{\text{Rtree}} \leftarrow Set_{\text{Rtree}} \cup \{rtree\}$;
6)　　**else if** $cp_{\min}^{A_i} \geq \max_{A_j \in Set_{\text{area}} \setminus Set_{\text{ini}}} \{cp_{\min}^{A_j}\}$
7)　　$Set_{\text{ini}} \leftarrow Set_{\text{ini}} \cup A_i$
8) **if** $|Set_{\text{ini}}| \geq n$ **then**
9):　　$Top_n, ct \leftarrow Random(Set_{\text{ini}}, n)$;
10) **else**
11)　　$Set_{\text{ini}} \leftarrow Set_{\text{ini}} \cup MaxLeaf(Set_{\text{Rtree}})$
12)　　$Top_n, ct \leftarrow Random(Set_{\text{ini}}, n)$;
13) **for** $p \in Set_{\text{ini}} - Top_n$ **do**
14)　　**if** $UB_3(p) \leq ct$ or $UB_4(p) \leq ct$ **then**
15)　　$p.st \leftarrow prune$;
16) **else if** $LOF(p) \leq ct$ **then**
17)　　$p.st \leftarrow prune$;
18)　　**else**
19)　　$Top_n.replace(p)$; update ct;
20) **for** $rtree \in Set_{\text{Rtree}}$ **do**
21)　　**for** $node \in rtree$ from top to bottom
22)　　**if** $node.st == prune$
23)　　continue;
24)　　**else if** $|node| > k \wedge node.len_{\text{side}} \leq ct \cdot cp_{\min}/(2\sqrt{d})$ **then**
25)　　$node.st \leftarrow prune$;
26)　　$node.allchilden \leftarrow prune$;
27) **else if** $node.len_{\text{side}} \leq ct \cdot \overline{kd}_{\min}/(2\sqrt{d})$
28)　　$node.st \leftarrow prune$;
29)　　$node.allchilden \leftarrow prune$;
30)　　**else**
31)　　**for** $p \in node$
32)　　**if** $UB_3(p) \leq ct$ or $UB_4(p) \leq ct$
33:　　$p.st \leftarrow prune$;
34)　　**else if** $LOF(p) \leq ct$ **then**
35)　　$p.st \leftarrow prune$;
36)　　**else**
37)　　$Top_n.replace(p)$; update ct;
38) return Top_n.

首先, 采用均匀区域划分方法将数据空间划分成多个区域 (如行 1) 所示), 对于内部数据对象数量大于等于 $t \cdot k$ 的区域, 建立一颗 Rtree 索引 (如行 3)～5) 所示), 否则, 如果该区域的 $cp_{\min}^{A_i}$ 为

$Set_{\text{area}} \setminus Set_{\text{ini}}$ 的最大者, 将区域内的数据对象放入初始候选异常点集合 Set_{ini} (如行 6) ∼ 7) 所示). 在数据对象较多的区域内建立 Rtree 的目的有两个: 一是为了进行基于 Cell 的剪枝判断, 二是快速查找每个数据对象的 k 近邻.

然后, 在 Set_{ini} 集合中随机选择 n 个数据对象作为初始 Top-n 局部异常点, 如果 Set_{ini} 内少于 n 个数据对象, 则从 Rtree 集合中选择覆盖区域最大的叶子节点, 从中再随机选择初始 Top-n 局部异常点, 并获取初始临界值 ct, 过程如行 8) ∼ 12) 所示.

之后, 开始遍历所有数据对象. 首先遍历 Set_{ini} 剩余的数据对象, 这是因为这些数据对象被 Cell 剪枝的可能性较小. 如行 13) ∼ 19) 所示, 首先利用上界 $UB_3(p)$ 和 $UB_4(p)$ 进行剪枝判断, 若不行, 再计算 LOF 值, 若不能被剪枝, 则将 p 放入 Top_n 取代 LOF 值最小的数据对象, 并更新临界值 ct. 接下来开始遍历 Rtree 集合, 对于每棵 Rtree, 从上向下遍历节点. 根据引理 1 和引理 2, 如果节点内包含多于 k 个数据对象, 并且其边长 $len_{\text{side}} \leq ct \cdot cp_{\min}/(2\sqrt{d})$ 或者 $len_{\text{side}} \leq ct \cdot \overline{kd}_{\min}/2\sqrt{d}$, 则该节点可被剪枝掉, 因此, 它的所有子节点都可以被剪枝掉 (行 21) ∼ 29) 所示). 对于不能被两层 Cell 剪枝策略剪枝的节点, 则需要对内部每个数据对象进行基于 LOF 上界的剪枝判断, 如行 31) ∼ 37) 所示, 首先执行两层 LOF 上界 $UB_3(p)$ 和 $UB_4(p)$ 的剪枝判断, 若不能通过上界剪枝, 再计算 LOF 值, 若仍大于临界值 ct, 则更新 Top_n 和 ct. 最后, 返回最终的 Top-n 局部异常点集合.

4 实验评估

本节在 6 个真实数据集 (第 4.1 节) 上对所提算法进行了综合评估. 首先, 在第 4.2 节评估了所提算法的总体效率, 然后, 在第 4.3 节分别评估了基于 Cell 剪枝和基于数据对象剪枝的效率, 在第 4.4 节分别证明了所提的四个上界在剪枝方面的有效性以及初始化优化方法的有效性, 之后, 在第 4.5 节评估了 LOF 类算法 (MTLOF、LOF、MC、TOLF) 与 Iforest、Simplified-LOF 算法的异常检测准确率和效率, 在第 4.6 节分析了参数 k 和 n 对所提的 MTLOF 及对比算法的效率影响, 最后, 在第 4.7 节验证了所提算法在多维数据集上的有效性.

4.1 实验数据与测试方法

实验平台采用 Intel Xeon E5-2660 处理器, 8 核, 48 GB 内存, Windows sever 2012 操作系统. 所有算法采用 Java 实现, MTLOF 源代码已在 Github[1] 公开.

[1] https://github.com/LiuFang0812/TopNDetection

实验数据. 实验采用了 6 个真实数据集, 详细统计信息如表 1 所示.

1) Mobike: 通过摩拜单车平台上[2] 爬取的北京市六环内所有摩拜单车的真实 Gps 位置数据, 单车数量多达五万辆, 该数据集仅提取了一天的数据.

2) Gowalla[35]: 该数据集来自移动社交网站 Gowalla, 包括了 19.6 万用户在 2009 年 2 月到 2010 年 10 月的签到位置数据, 共计 644 万条位置信息, 本实验提取了在北美范围内的数据, 约 510 万条位置信息.

表 1 实验数据集统计信息

Table 1 The statistical information of experimental data sets

数据集	数据对象数量	数据维度	数据大小 ($\times 10^6$)
Mobike	1 082 732	2	43.1
Gowalla	5 127 211	2	376
Geolife	11 065 399	2	851
Massachusetts	31 136 410	2	1 228
Skinseg	245 057	3	12
Forestcover	581 012	10	71.6
Subforestcover	283 402	10	28.3

3) Geolife[36]: 该数据集来自微软亚洲研究院的 Geolife 项目, 收集了 182 名用户在 2007 年 4 月至 2012 年 8 月间的 Gps 轨迹数据, 每条 Gps 轨迹由带有时间戳的经纬度位置点序列组成.

4) Massachusetts (Mass)[37]: 该数据集通过 Openstreemap[3] 采集, 在美国马萨诸塞州范围内所有的建筑物的地理位置, 数据集中的每一行代表一栋建筑物, 地理位置采用经度和纬度两个属性表示.

5) Skinseg[38]: 该数据集从不同年龄组 (青年、中年、老年)、不同种族 (白人、黑人、亚洲人) 以及不同性别的人脸照片中随机采样的 B、G、R 数值的数据集, 包含三维特征属性.

6) Forestcover[39]: 该数据集来自科罗拉多北部罗斯福国家森林的四个荒野森林 (Neota、Rawah、Comanche Peak 和 Cache La Poudre), 共包括 58 万多个 30×30 平方米的区域, 每个区域包含了各种树种的海拔高度、数量以及坡度等十个属性.

7) Subforestcover[39]: 该数据集选取了 Forestcover 数据集中所有带标签的数据, 共包括 28 万多个对象. Rawah 和 Comanche Peak 森林主要生长的物种为黑松, 它们的物种和特征变量的范围 (如海

[2] https://api.mobike.com/

[3] https://www.openstreetmap.org

拔范围等) 都比较典型, 于是把标记为黑松的数据记录视为正常对象, 标记类别为 2, Cache La Poudre 森林主要生长黄松、花旗松和杨木/柳树, 由于相对较低的海拔和物种组成, 该森林相比其他森林更为独特, 数据集中将标记为杨木/柳树的数据记录视为异常对象, 类别标记为 4, 异常对象数量的比例为 Subforestcover 数据的 7%.

对比算法. 为了验证所提 MTLOF 算法的有效性, 实验结果与 LOF 算法[16]、MC 算法[20] 以及最新的 TOLF 算法[19]、Iforest 算法[33−34]、Simplified-LOF [28−29] 进行了对比分析.

评估指标. 针对算法的效率评估, 测量了每个算法总的检测时间, 同时还分别测量了数据预处理和检测计算的 CPU 时间. 为了更好地评估算法性能, 实验部分还对算法的剪枝效果进行了评价.

4.2　总体效率评估

首先, 在表 1 所示的四个二维数据集上, 评估了我们提出的 MTLOF 的效率, 并与 LOF、MC、TOLF 算法进行了对比分析.

实验结果如图 5 所示, 参数 n 取 $0.001\% \cdot |D|$, 由于各数据集的数据对象数量及分布不同, 在 Mobike、Gowalla、Geolife 和 Mass 数据集, k 分别取 6、20、20 和 30, 与 TOLF 算法设置一致, 固定 $diff$ 为 10, t 为 6.

从图 5 可以看出, MTLOF 算法在四个数据集上都表现出了最好的检测效率, 相比 LOF、MC 和 TOLF 算法, 分别平均提升了 30、18 和 2.6 倍的效率. 特别是在大规模的 Geolife 数据集, 相比最新的 TOLF, MTLOF 算法提升的效率高达 3.5 倍. MTLOF 和 TOLF 都采用了均匀区域划分的预处理方法, 还对区域内数据对象建立了索引结构, 极大地加快了 k 近邻的搜索, 减少了数据预处理的时间. 虽然 MC 算法也通过聚类的方法对数据进行预处理, 在聚类结果上进行相应的剪枝, 但是数据对象聚类所消耗的预处理时间远高于建立索引的预处理时间. 最新的 TOLF 在划分的区域内建立 Cell 网格索引, 利用 $cp_{\min}$ 进行一次 Cell 剪枝, 然后利用较大的两个上界进行两次数据点剪枝, 然而当区域内的 $cp_{\min}$ 较小时, 几乎没有 Cell 被剪枝掉, 而且执行数据点剪枝的两个上界相比真实 LOF 值较大且计算量较高. 而本文所提的 MTLOF 采用了两层 Cell 剪枝策略, 除了使用全局的 $cp_{\min}$ 剪枝, 还利用基于 Cell 内部数据对象分布的局部剪枝策略, 当 $cp_{\min}$ 较小时,

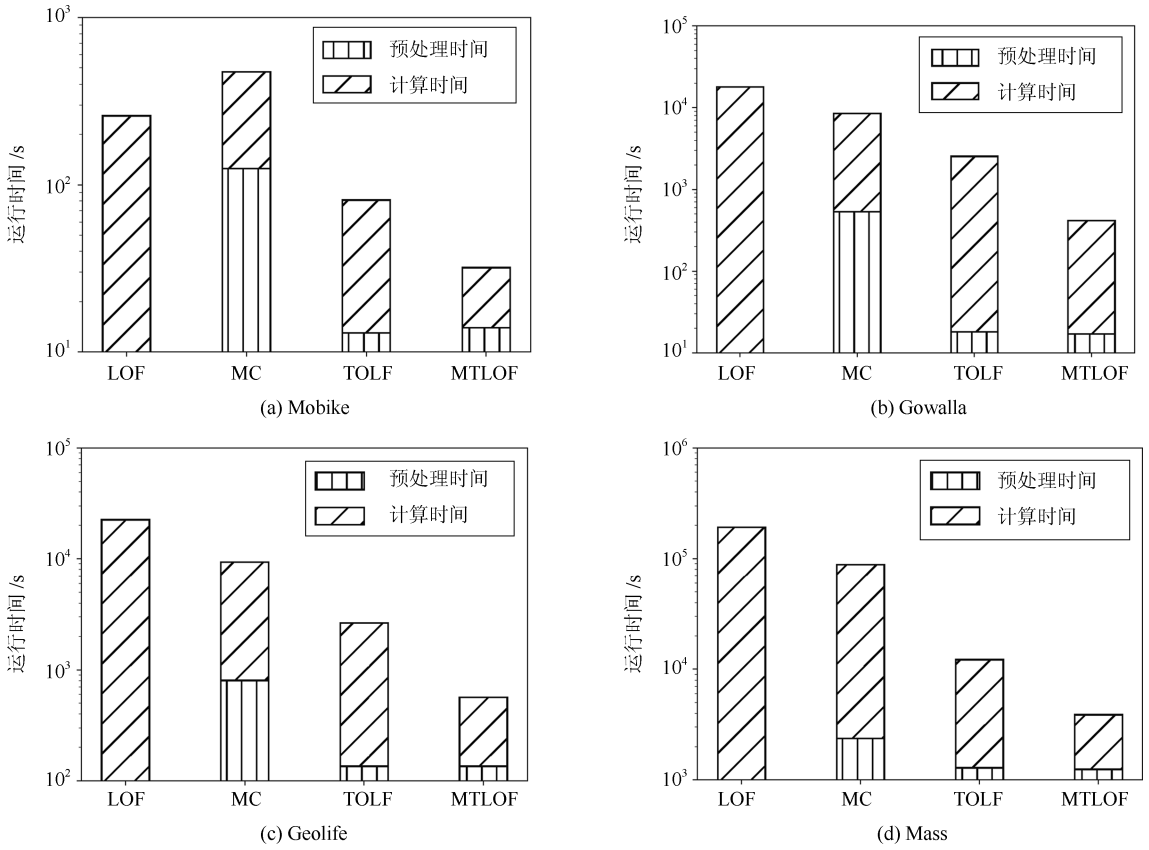


图 5　总体效率对比评估

Fig. 5　Comparison evaluation of overall efficiency

仍能通过 $\overline{kd}_{\min}$ 剪枝掉高密度的 Cell, 此外, 我们使用层次的 Rtree 代替 Cell 网格索引, 可以更快速地剪枝掉较大块的高密度节点. 对于不能被剪枝的节点内的数据对象, 我们采用了两个更加接近 LOF 值的上界进行剪枝判断, 从而使得被剪枝掉的数据对象更多, 同时, 基于我们的复用计算优化, 对 $UB_3(p)$ 和 $UB_4(p)$ 的计算完全可复用 Cell 剪枝过程中的计算, 有效减少了检测计算成本. MTLOF 效率远高于 TOLF 的另外一个重要原因还在于, 我们优化了对初始局部异常点的选取, 利用预处理过程的区域划分和数据索引, 优先在稀疏区域和 Rtree 中覆盖区域较大的叶子节点内部选择初始候选局部异常点, 相比随机选择方法, 大大提升了初始异常点的 LOF 值, 相应地, 也提升了初始临界值 ct, 使得更多的树节点在初始阶段被快速剪枝.

4.3 剪枝效率评估

为了验证算法剪枝的有效性, 在上一个实验的同时, 我们还在四个数据集上统计了 MTLOF 和 TOLF 算法的 Cell 剪枝及对象剪枝中被剪枝的数据对象数量. MTLOF 算法的统计结果如表 2 所示, TOLF 算法的统计结果如表 3 所示.

表 2 MTLOF 剪枝数量 (%)
Table 2 The pruning number of MTLOF (%)

	Mobike	Gowalla	Geolife	Mass
Cell 剪枝	26.9	20.9	42.1	30.2
数据对象剪枝	71.5	40.9	43.8	68.7
总剪枝数量	98.4	61.8	85.9	98.9

表 3 TOLF 剪枝数量 (%)
Table 3 The pruning number of TOLF (%)

	Mobike	Gowalla	Geolife	Mass
Cell 剪枝	0	0	9.2	5.9
数据对象剪枝	62.3	19.5	52.4	80.1
总剪枝数量	62.3	19.5	61.6	86

从表 2 和表 3 可以看到, 在四个数据集上, MTLOF 的总剪枝的数据对象数量都远多于 TOLF 的总剪枝数量. 在 Mobike 和 Mass 数据集上, MTLOF 算法直接剪枝掉的数据对象总量高达 98 % 以上, 在 TOLF 剪枝较少的 Gowalla 数据集上, MTLOF 也直接剪枝了 61.8 % 的数据对象. 更为明显地是, MTLOF 的 Cell 剪枝的数据量比例都在 20 % 以上, 甚至高达 40 % 以上, 而 TOLF 的 Cell 剪枝比例相对较少, 在 Mobike 和 Gowalla 数据集上, TOLF 的 Cell 剪枝甚至为 0 %, 这是因为在 Mobike 和 Gowalla 数据集的高密度区域内, 数据对象间的最小距离往往非常小甚至为零, 在 TOLF 的 Cell 剪枝中, 始终采用 $ct \cdot cp_{\min}/(2\sqrt{d})$ 为 Cell 边长, 当每个区域的 $cp_{\min}$ 都很小时, 就使得 Cell 剪枝失效. 而本文所提的 MTLOF 除了基于 Cell 的全局剪枝, 还引入了基于 Cell 的局部剪枝, 当区域内的 $cp_{\min}$ 较小时, 仍能通过每个节点 (Cell) 内数据对象的 $\overline{kd}_{\min}$ 进行 Cell 剪枝. 此外, 在同一数据集上, MTLOF 的数据对象剪枝的数量比例也高于 TOLF 的数据对象剪枝比例, 这是因为 MTLOF 采用的两个上界 $UB_3(p)$ 和 $UB_4(p)$ 更加接近于真实 LOF 值, 因此, 被剪枝掉的数据对象更多, 这也说明了本文所提的 LOF 上界更加合理有效. 优化的初始局部异常点选择方法也是 MTLOF 算法具有较高的剪枝比例的重要原因, 初始临界值 ct 越高, 被剪枝掉 Cell 和数据对象数量也就越多.

4.4 优化有效性评估

为了评估 UB_1、UB_2、UB_3 和 UB_4 四个上界在剪枝方面的有效性, 本节在四个二维数据集上, 分别统计了 UB_1、$UB_1 + UB_2$、$UB_1 + UB_2 + UB_3$ 和 $UB_1 + UB_2 + UB_3 + UB_4$ 四种组合情况下的被剪枝的对象数量, 实验结果如表 4 所示. 第二行表示在每个数据集上仅利用上界 UB_1 剪枝的对象数量, 第三行表示同时采用 UB_1 和 UB_2 剪枝的对象数量, 第四行表示通过 UB_1、UB_2 和 UB_3 总共剪枝的对象数量, 最后一行表示同时使用四个上界时的总剪枝数量. 可以看到, 随着采用上界个数的增加, 表中每行剪枝数量比例都有所增加, 这证明了每个上界剪枝都是有效的. 利用上界 UB_1 的 Cell 全局剪枝, 虽然计算成本较小, 但是剪枝的数量也相对有限. 基于 UB_2 的 Cell 局部剪枝, 虽然需要计算本地所有数据对象的 k 近邻距离和的均值, 但剪枝掉的数量也相对较多. 上界 UB_3 和 UB_4 主要用于剪枝 Cell 过程中未被剪枝的对象. 根据第 3.2.2 节分析可知, $UB_4(p) < UB_3(p)$, 但是 UB_3 的时间复杂度更小, 并且可以复用之前计算过的每个对象的 k-距离, 可直接获得每个对象的 UB_3, 于是首先利用 UB_3 来进行对象剪枝, 平均可剪枝掉 41.1 % 的数据对象, 最高可达 61.5 %. 虽然 UB_4 最接近真实的 LOF 值, 但平均只剪枝 15.2 % 的对象, 这是因为上界 UB_1、UB_2 和 UB_3 已经剪枝掉了大部分的数据对象.

此外, 我们还在 Mobike 和 Gowalla 数据集上统计了 MTLOF 在采用初始 Top-n 异常对象优化选取方法和没有采用优化选取方法时分别所需的运行时间, 实验结果如图 6 所示. 相比 TOLF 算法, 没有采用初始化优化方法的 MTLOF 在两个数据集上分别提升 2.1 和 2.5 倍, 而采用初始化优化的 MTLOF 分别提升 2.6 和 3.4 倍. 也就是说, 本文

所提的剪枝策略在两个数据集上提升 2.1 和 2.5 倍效率, 而初始化优化方法又进一步提升 1.3 倍和 1.4 倍.

表 4　MTLOF 每个上界剪枝数量 (%)

Table 4　The pruning number of each upper bound in MTLOF (%)

	Mobike	Gowalla	Geolife	Mass
UB_1	6.1	9.4	25.3	16.8
UB_1+UB_2	26.9	20.2	42.1	30.2
$UB_1+UB_2+UB_3$	88.4	50.2	70.5	74.7
总剪枝数量	98.4	61.8	85.9	98.9

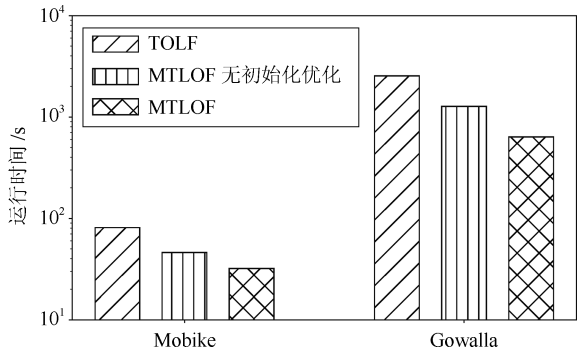


图 6　初始化优化方法有效性评估

Fig. 6　Effectiveness evaluation of initialization optimization

4.5　正确性评估

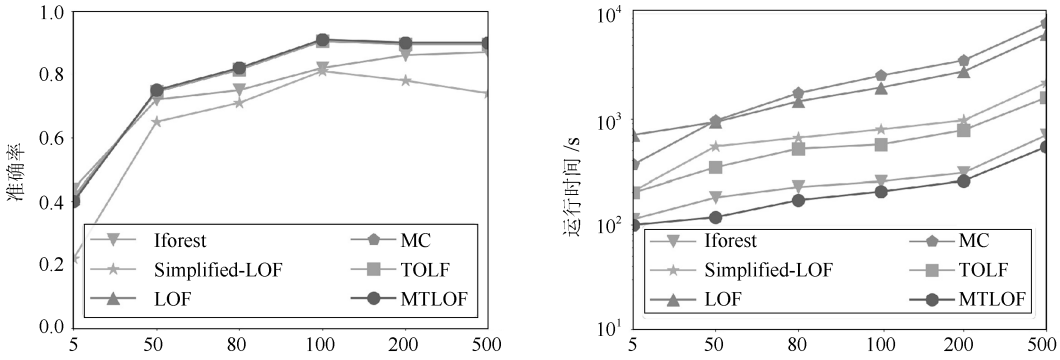
本节在 Subforestcover 数据集上评估了 MTLOF 算法以及对比算法 LOF、MC、TOLF、Iforest、Simplified-LOF 的准确率和效率. 准确率 $=(R\cap D)/R$, 其中 D 表示数据集中真实的异常对象集合, R 指检测算法发现的异常对象集合.

实验结果如图 7 所示, 对于 MTLOF、LOF、MC、TOLF 和 Simplified-LOF 算法, 横坐标表示参数 k 值, 对于 Iforest 算法, 横坐标表示为训练 Itree 而采样的对象子集数量 m. 图 7 (a) 展示了所有算法检测结果的准确率, 由于 MTLOF、MC 和 TOLF 都是通过计算对象的 LOF 值来检查 Top-n 异常的, 所以这三种算法的准确率与 LOF 算法一致, 同时, 实验结果也验证了本文所提剪枝策略的正确性. 从图中还可以看出, LOF 类算法 (MTLOF、LOF、MC、TOLF) 的准确率在 $k(m)\geq 50$ 以后一直优于 Iforest 和 Simplified-LOF 算法. 当 k 取 100 时, LOF 类算法的准确率达到最大的 93 %. 而对于 Iforest 算法, 准确率最高仅达到 86 %. Simplified-LOF 算法的准确率相对小, 最大值仅达到 81 %, 之后不断增大 k 值, 准确率反而急剧下降, 这是因为 Simplified-LOF 直接用 k-距离代替可达距离, 当 k 值较大时, 采用 k-距离所表示的局部密度将变的不准确.

图 7 (b) 展示了所有算法在参数变化下的检测时间, 可以发现, MTLOF 的运行时间一直优于所有对比算法. 随着 $k(m)$ 值增加, 所有算法所需的运行时间都有所增加, 这是因为随着 k 的不断增大, 搜索 k 近邻的时间不断增加. 尽管如此, MTLOF 在所有测试参数下的运行时间都少于 Iforest 所消耗的时间. MTLOF 相比 Simplified-LOF 和 Iforest 算法, 平均分别提升 3.7 和 1.5 倍效率.

4.6　参数敏感性分析

本节在 Mobike 和 Mass 数据集上评估重要参数 k 和 n 的变化对所提 MTLOF 算法以及对比算法的效率影响.



(a) 准确率　(a) Precision　　(b) 效率　(b) Efficiency

图 7　准确率和效率评估

Fig. 7　Evaluation of precision and efficiency

4.6.1 参数 k 影响评估

固定参数 $n = 0.001\% \cdot |D|$, 从 1 到 80 变化参数 k, 图 8 展示了四个算法在两个数据集上随着参数 k 变化的总检测时间. 如图 8 所示, MTLOF 的检测效率在两个数据集上的所有测试都一直优于三个对比算法, 相比 LOF、MC 和 TOLF 算法分别平均提升了 28, 19 和 2.7 倍. 随着 k 值的增大, 所有算法消耗的总检测时间越来越长, 这是因为 k 值的增加, 使得所有数据对象的 k 近邻查询越来越费时, 导致总处理时间直线增加. 尽管如此, 随着参数 k 的增加, MTLOF 比 LOF、MC 和 TOLF 算法节省的 CPU 时间越来越多, 优势越来越明显, 这是因为 k 值的增加使得 Rtree 索引的叶子节点不断增大, 同时所提的上界也更加接近于真实的 LOF 值, 致使 MTLOF 剪枝掉更多的节点和数据对象, 相对对比算法, 效率优势越来越明显.

4.6.2 参数 n 影响评估

图 9 展示了算法随参数 n 变化的总检测时间, 在 Mobike 数据集上固定 $k = 5$, 在 Mass 数据集上固定 $k = 10$, 从 1 到 1 000 变化参数 n. 如图 9 所示, MTLOF 算法在所有参数下的检测效率都优于三个对比算法, 相比最新的 TOLF 算法, 平均提升了 2.75 倍. 所有算法的检测时间相对于 n 的变化并不明显. 基本的 LOF 算法检测 Top-n 局部异常分为两个步骤, 首先计算所有数据对象的 LOF 值, 然后排序获取前 n 个异常对象, 随着 n 的不断变化, 排序所花费的时间也会不断地增加, 但是这个排序的时间相比计算 LOF 的时间要小的多, 使得总的检测时间并没有明显增加. MC 算法首先采用 Birch 聚类获得聚类簇, 然后按照聚簇的半径和距离关系排序聚类簇, 仅需在最可能包含异常的簇内检测异常, 因此, 随着 n 的增加, 需要检测的聚簇个数越多, 消耗的总检测时间也越长, 但是 MC 在 n 较小时剪枝掉的数据数量都相对较少, 因此, 总消耗的检测时间也没有明显增加. TOLF 和 MTLOF 算法的检测时间随着 n 的增加而缓慢增加, 这是因为 n 的增加使得初始需要计算 LOF 值的数据对象增加, 尽管如

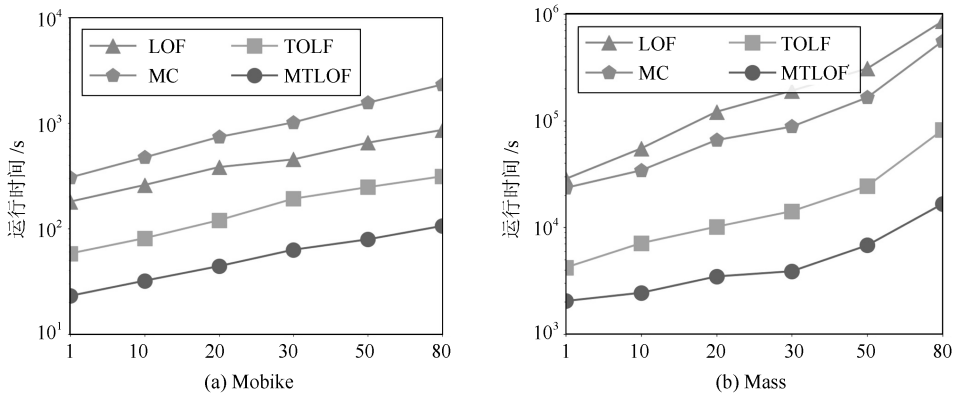


图 8 参数 k 对检测时间的影响

Fig. 8 Impact of parameter k on detection time

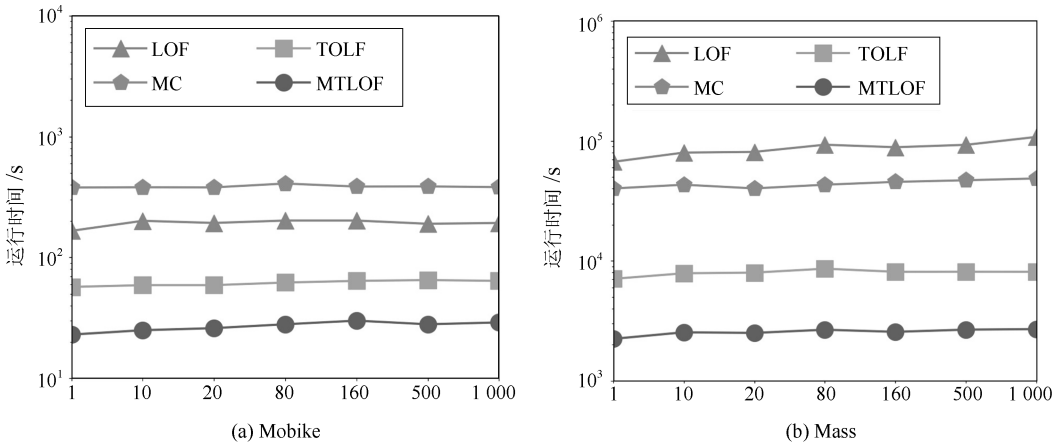


图 9 参数 n 对检测时间的影响

Fig. 9 Impact of parameter n on detection time

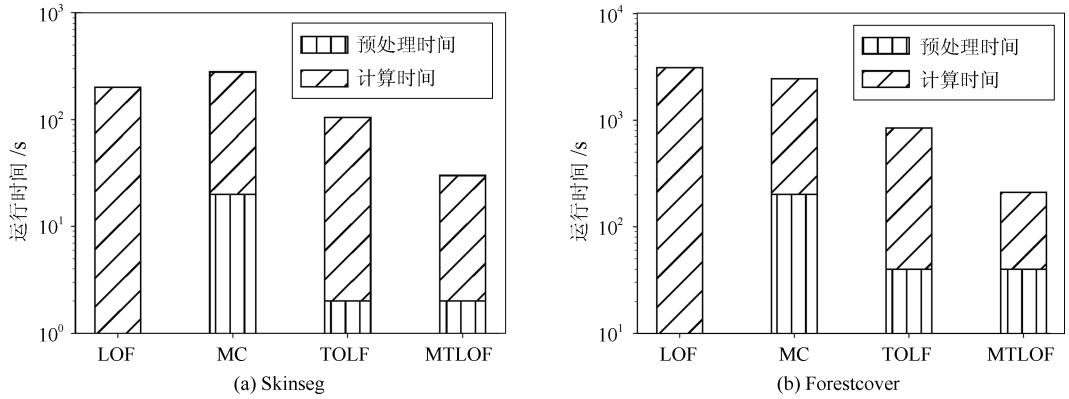


图 10　多维数据集上的效率评估

Fig. 10　Efficient evaluation on multi-dimensional datasets

此, 由于两个算法都对数据进行了均匀区域划分并在区域内建立了索引结构, 即使 n 不断增加, 仍然可以通过索引快速查找 k 近邻, 并通过 LOF 上界执行剪枝, 因此, 总检测时间也没有急剧增加. 随着 n 的增加, 我们 MTLOF 能够通过提出的 $UB_3(p)$ 和 $UB_4(p)$ 上界剪枝掉更多数据对象, 相比 TOLF, 节省更多检测时间.

4.7　多维数据上的有效性评估

最后, 在多维数据集 Skinseg 和 Forestcover 上评估了所提算法的有效性. 图 10 展示了四个算法在 Skinseg 和 Forestcover 上的总检测时间. 参数 n 固定为 $0.001\,\% \cdot |D|$, k 固定为 6. 在 Skinseg 数据集上, MTLOF 仍然能比最新的 TOLF 算法快了近 2.5 倍. 虽然 Skinseg 数据集较小, 但本文提出的多粒度的剪枝策略在这个数据集上明显优于 TOLF 算法. 在 10 维的 Forestcover 数据集上, MTLOF 算法也展现了所提剪枝策略的优势, 相比 LOF、MC 和 TOLF 算法分别提升了 15 倍、12 倍和 3 倍的检测效率, 这也说明了 MTLOF 在高维数据集上具有更高的可扩展性.

5　结论

本文提出了一个面向大数据的高效的 Top-n 局部异常点检测算法 MTLOF. 首先, 为了避免直接计算数据对象的 LOF 值, 提出了四个计算复杂度更低并且更接近于真实 LOF 值的上界. 其次, 结合索引结构和 LOF 上界, 引入了两层的 Cell 剪枝策略. 然后, 针对未被剪枝的 Cell 内部数据对象, 利用 $UB_3(p)$ 和 $UB_4(p)$ 上界提出了两个更加合理有效的剪枝策略. 此外, 还利用均匀区域划分和建立的索引结构, 优化了初始候选局部异常点的选取方法, 使得 LOF 值较大的数据对象被选取为初始局部异常点. 最后, 在六个真实数据集上的综合实验评估验证了所提 MTLOF 算法的有效性, 相比 LOF、MC 和 TOLF 算法, 检测效率可分别提升 30、18 和 2.6 倍.

下一步工作将考虑借助分布式计算平台, 设计分布式异常检测算法以进一步提升检测效率, 此外, 还计划面向不断快速增长的海量高速数据集, 研究实时的 Top-n 局部异常点检测方法.

References

1 Du B W, Liu C R, Zhou W J, Hou Z S, Xiong H. Catch me if you can: detecting pickpocket suspects from large-scale transit records. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, California: ACM, 2016. 87−96

2 Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. *ACM Computing Surveys (CSUR)*, 2009, **41**(3): Article No. 15

3 Gupta M, Gao J, Aggarwal C C, Han J W. Outlier detection for temporal data: a survey. *IEEE Transactions on Knowledge & Data Engineering*, 2014, **26**(9): 2250−2267

4 Aggarwal C C. Outlier analysis. *Data Mining*. Switzerland: Springer, 2015. 237−263

5 Knorr E M, Ng R T. Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the 24th International Conference on Very Large Data Bases. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1998. 392−403

6 Knorr E M, Ng R T, Tucakov V. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 2000, **8**(3−4): 237−253

7 Wang Xi-Te, Shen De-Rong, Bai Mei, Nie Tie-Zheng, Kou Yue, Yu Ge. BOD: an efficient algorithm for distributed outlier detection. *Chinese Journal of Computers*, 2016, **39**(1): 36−51
(王习特, 申德荣, 白梅, 聂铁铮, 寇月, 于戈. BOD: 一种高效的分布式离群点检测算法. 计算机学报, 2016, **39**(1): 36−51)

8 Bay S D, Schwabacher M. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, D.C.: ACM, 2003. 29−38

9 Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Dallas, Texas, USA: ACM, 2000. 427−438

10 Angiulli F, Pizzuti C. Fast outlier detection in high dimensional spaces. In: European Conference on Principles of Data Mining and Knowledge Discovery. Berlin, Heidelberg: Springer, 2002. 15−27

11 Solberg H E, Lahti A. Detection of outliers in reference distributions: performance of Horn′s algorithm. *Clinical Chemistry*, 2005, **51**(12): 2326−2332

12 Aggarwal C C, Yu P S. Outlier detection for high dimensional data. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. Santa Barbara, California, USA: ACM, 2001. 37−46

13 Aggarwal C C, Yu P S. Outlier detection with uncertain data. In: Proceedings of the 2008 SIAM International Conference on Data Mining. Atlanta, Georgia, USA: SIAM, 2008. 483−493

14 Yu D T, Sheikholeslami G, Zhang A D. FindOut: finding outliers in very large datasets. *Knowledge and Information Systems*, 2002, **4**(4): 387−412

15 He Z Y, Xu X F, Deng S C. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 2003, **24**(9−10): 1641−1650

16 Breunig M M, Kriegel H P, Ng R T, Sander J. LOF: identifying densitybased local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Dallas, Texas, USA: ACM, 2000. 93−104

17 Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J. A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the 2003 SIAM International Conference on Data Mining. San Francisco, CA, USA: SIAM, 2003. 25−36

18 Campos G O, Zimek A, Sander J, Campello R J G B, Micenková B, Schubert E, et al. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 2016, **30**(4): 891−927

19 Yan Y Z, Cao L, Rundensteiner E A. Scalable top-n local outlier detection. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Halifax, NS, Canada: ACM, 2017. 1235−1244

20 Jin W, Tung A K H, Han J W. Mining top-n local outliers in large databases. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, California: ACM, 2001. 293−298

21 Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. Montreal, Quebec, Canada: ACM, 1996. 103−114

22 Tang J, Chen Z X, Fu A W C, Cheung D W. Enhancing effectiveness of outlier detections for low density patterns. In: Proceedings of the 2002 Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin, Heidelberg: Springer, 2002. 535−548

23 Zhu Li, Qiu Yuan-Yuan, Yu Shuai, Yuan Sheng. A fast kNN-based MST outlier detection method. *Chinese Journal of Computers*, 2017, **40**(12): 2856−2870
(朱利, 邱媛媛, 于帅, 原盛. 一种基于快速 k-近邻的最小生成树离群检测方法. 计算机学报, 2017, **40**(12): 2856−2870)

24 Papadimitriou S, Kitagawa H, Gibbons P B, Faloutsos C. Loci: fast outlier detection using the local correlation integral. In: Proceedings of the 19th International Conference on Data Engineering. Bangalore, India: IEEE, 2003. 315−326

25 Yang Yi-Dong, Sun Zhi-Hui, Zhu Yu-Quan, Yang Ming, Zhang Bo-Li. A fast outlier detection algorithm for data streams based on dynamic grids. *Journal of Software*, 2006, **17**(8): 1796−1803
(杨宜东, 孙志挥, 朱玉全, 杨明, 张柏礼. 基于动态网格的数据流离群点快速检测算法. 软件学报, 2006, **17**(8): 1796−1803)

26 Zhang K, Hutter M, Jin H D. A new local distance-based outlier detection approach for scattered real-world data. In: Proceedings of the 2009 Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin, Heidelberg: Springer, 2009. 813−822

27 Kriegel H P, Kröger P, Schubert E, Zimek A. LoOP: local outlier probabilities. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management. Hong Kong, China: ACM, 2009. 1649−1652

28 Schubert E, Zimek A, Kriegel H P. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Mining and Knowledge Discovery*, 2014, **28**(1): 190−237

29 Schubert E, Zimek A, Kriegel H P. Generalized outlier detection with flexible kernel density estimates. In: Proceedings of the 2014 SIAM International Conference on Data Mining. Philadelphia, Pennsylvania, USA: SIAM, 2014. 542−550

30 Liu J, Deng H F. Outlier detection on uncertain data based on local information. *Knowledge-Based Systems*, 2013, **51**: 60−71

31 Cao K Y, Shi L X, Wang G R, Han D H, Bai M. Density-based local outlier detection on uncertain data. In: Proceedings of the 2014 International Conference on Web-Age Information Management. Macau, China: Springer, 2014. 67−71

32 Cao Ke-Yan, Luan Fang-Jun, Sun Huan-Liang, Ding Guo-Hui. Density-based local outlier detection on uncertain data. *Chinese Journal of Computers*, 2017, **40**(10): 2231−2244
(曹科研, 栾方军, 孙焕良, 丁国辉. 不确定数据基于密度的局部异常点检测. 计算机学报, 2017, **40**(10): 2231−2244)

33 Liu F T, Ting K M, Zhou Z H. Isolation forest. In: Proceedings of the 8th IEEE International Conference on Data Mining. Pisa, Italy: IEEE, 2008. 413−422

34 Liu F T, Ting K M, Zhou Z H. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2012, **6**(1): Article No. 3

35 Cho E, Myers S A, Leskovec J. Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego, California, USA: ACM, 2011. 1082−1090

36 Zheng Y, Xie X, Ma W Y. GeoLife: a collaborative social networking service among user, location and trajectory. *IEEE Data Engineering Bulletin*, 2010, **33**(2): 32−40

37 Haklay M, Weber P. OpenStreetMap: user-generated street maps. *IEEE Pervasive Computing*, 2008, **7**(4): 12−18

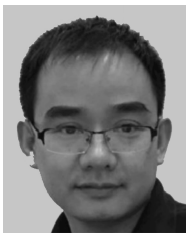
38 Bhatt R B, Sharma G, Dhall A, Chaudhury S. Efficient skin region segmentation using low complexity fuzzy decision tree model. In: Proceedings of the 2009 Annual IEEE India Conference. Gujarat, India: IEEE, 2009. 1−4

39 Blackard J A, Dean D J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 1999, **24**(3): 131−151

刘 芳 烟台大学计算机与控制工程学院硕士研究生. 2017 年获得烟台大学计算机科学与技术专业学士学位. 主要研究方向为数据挖掘.
E-mail: liufang0812@163.com
(**LIU Fang** Master student at the School of Computer and Control Engineering, Yantai University. She received her bachelor degree in computer science from Yantai University in 2017. Her main research interest is data mining.)

齐建鹏 烟台大学计算机与控制工程学院硕士研究生. 2015 年获得烟台大学计算机科学与技术专业学士学位. 主要研究方向为数据挖掘, 分布式计算.
E-mail: jianpengqi@126.com
(**QI Jian-Peng** Master student at the School of Computer and Control Engineering, Yantai University. He received his bachelor degree in computer science from Yantai University in 2015. His research interest covers data mining and distributed computing.)

于彦伟 烟台大学计算机与控制工程学院副教授, 美国宾夕法尼亚州立大学信息科学与技术学院博士后. 2014 年获得北京科技大学计算机科学与技术专业博士学位. 主要研究方法为数据挖掘, 机器学习, 分布式计算. 本文通信作者.
E-mail: yuyanwei@ytu.edu.cn
(**YU Yan-Wei** Associate professor at the School of Computer and Control Engineering, Yantai University, Postdoctor at the College of Information Sciences and Technology, Pennsylvania State University. He received his Ph. D. degree in computer science from University of Science and Technology Beijing in 2014. His research interest covers data mining, machine learning and distributed computing. Corresponding author of this paper.)



曹 磊 麻省理工学院计算机科学与人工智能实验室博士后. 2016 年获得伍斯特理工学院计算机科学专业博士学位. 主要研究方法为数据挖掘, 机器学习.
E-mail: lcao@csail.mit.edu
(**CAO Lei** Postdoctor at the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. He received his Ph. D. degree in computer science from Worcester Polytechnic Institute, USA in 2016. His research interest covers data mining and machine learning.)

赵金东 烟台大学计算机与控制工程学院副教授. 2012 年获得北京科技大学计算机科学与技术专业博士学位. 主要研究方法为无线传感器网络, 分布式计算.
E-mail: zhjdong@126.com
(**ZHAO Jin-Dong** Associate professor at the School of Computer and Control Engineering, Yantai University. He received his Ph. D. degree in computer science from University of Science and Technology Beijing in 2012. His research interest covers wireless sensor networks and distributed computing.)