第 34 卷 第 17 期 计 算 机 工 程 2008 年 9 月
Vol.34 No.17 Computer Engineering September 2008
·软件技术与数据库· 文章编号：1000—3428(2008)17—0059—03 文献标识码：A 中图分类号：TP311

基于局部偏离因子的孤立点检测算法

谭 庆，张瑞玲
(洛阳师范学院信息技术学院，洛阳 471022)

摘 要：孤立点检测是知识发现中的一个活跃领域，如信用卡欺诈、入侵检测等。研究孤立点的异常行为能发现隐藏在数据集中更有价值的知识。该文提出基于局部偏离因子(LDF)的孤立点检测算法，利用每个数据点的 LDF 衡量该数据点的偏离程度。实验结果表明，该算法能有效检测孤立点，其效率高于 LSC 算法。

关键词：孤立点；*k* 距离邻居；局部偏离因子

Outlier Detection Algorithm Based on Local Deviation Factor

TAN Qing, ZHANG Rui-ling
(Academy of Information Technology, Luoyang Normal University, Luoyang 471022)

【Abstract】 Outlier detection is a hot research field in knowledge discovery in databases, such as credit card fraud, and intrusion detection, etc. Finding the rare abnormal behaviors or the outliers can be more interesting than finding the common patterns. This paper proposes a new outlier detection algorithm based on Local Deviation Factor(LDF). This algorithm counts the number of each point's LDF to reflect its isolation degree. The experimental results show that this algorithm can efficiently detect outliers and has higher efficiency than outliers detection algorithm LSC.

【Key words】 outlier; *k*-distance neighbors; local deviation factor

1 概述

孤立点检测又称为孤立点分析、异常检测、例外挖掘、小事件检测、挖掘极小类、偏差检测。孤立点是数据集中与众不同的数据，它们不符合惯常的数据模式，其产生机制与一般数据不同。

孤立点检测是数据挖掘的重要研究内容之一，被用于发现数据集中比聚类小的模式，即数据集中明显不同于其他数据的对象。孤立点检测已被应用在电信和信用卡欺骗、贷款审批、气象预报、金融领域、网络入侵检测和客户分类等领域。孤立点检测在国外得到广泛研究[1-3]，并被越来越多的国内学者注意。机器学习领域的一些学者指出，异常分析将成为一个重要的研究方向。

Hawkins 对孤立点的本质性做如下定义：在数据集中与众不同的数据，使人怀疑这些数据并非随机偏差，而是产生于完全不同的机制。随后，研究者根据对异常存在的不同假设，提出了很多孤立点检测算法，主要可分为基于统计的算法、基于距离的算法、基于密度的算法、基于偏差的算法等。

2 基于局部偏离因子的孤立点检测算法

2.1 相关概念

本文在分析基于密度的聚类算法和基于密度的孤立点算法的基础上，提出基于局部偏离因子(Local Deviation Factor, LDF)的孤立点检测算法。

定义 1(对象 p 的 k 距离[4]) 对于任何一个正数 k，对象 p 的 k 距离，即 k-distance(p)，$d(p,o)$在对象 p 与对象 $o\in D$ 之间必须满足以下条件($d(p,o)$是 p 与 o 之间的欧式距离)：至少有 k 个对象 $o'\in D\backslash\{p\}$，$d(p,o')\leq d(p,o)$；至多有 k-1 个对象 $o'\in D\backslash\{p\}$，$d(p,o')<d(p,o)$。

定义 2(对象 p 的 k 距离邻居[4]) 给定一个对象 p 的 k 距离，p 的 k 距离邻居包括所有与 p 的距离小于 k 距离的对象，即

$$N_{k\text{-distance}(p)}=\{q\in D\backslash\{p\}|d(p,q)\leq k\text{-distance}(p)\} \quad (1)$$

上述对象成为对象 p 的 k 距离邻居，简记为 $N_k(p)$。

定义 3(对象 p 的局部偏离率) 给出对象 p 的 k 距离邻居，以 p 为圆心，以 k 距离为半径得到一个包含所有 k 距离邻居的圆，计算 k 距离邻居的质心 p'，可得对象 p 的局部偏离率 $LDR_k(p)$为

$$LDR_k(p)=\frac{dist(p,p')}{|N_{k\text{-distance}}(p)|} \quad (2)$$

其中，分子是对象 p 与质心 p'的欧式距离；分母是对象 p 的 k 距离邻居的总数。对象 p 的局部偏离率反映了在以 p 为圆心、k 距离为半径的圆内对象集对对象 p 的影响。如果 LDR 的值很小，说明在对象 p 周围的数据点分布较均匀，则 p 成为孤立点的概率很小。如果 LDR 的值很大，表明在对象 p 局部范围内的数据点对于 p 的分布是不相关的，则 p 成为孤立点的概率很大。如果 LDR 值为零，说明在对象 p 周围的数据点均匀分布，则 p 成为孤立点的概率很小。

由于最终的孤立点判断度量是依靠局部偏离因子，而不是 LDR，因此在计算对象的局部偏离因子时，可以对 LDR 值为零的对象进行剪枝，以节省算法的计算时间，但这将对精度造成一定影响。在考虑时间效果重于精度效果的条件下，还可以对 LDR 值为零的对象的 k 距离邻居进行剪枝，因为从概率上而言，上述对象很大程度上属于聚类中的核心对象，其 k 距离邻居在很大程度上都应该是聚类中的点。

定义 4(对象 p 的局部偏离影响率) 给定对象 p 的 k 距离

基金项目：河南省科技攻关基金资助项目(0524220059)
作者简介：谭 庆(1977－)，男，讲师、硕士，主研方向：数据挖掘，程序设计；张瑞玲，副教授
收稿日期：2008-05-12 **E-mail：**lytqemail@163.com

邻居和局部偏离率，p 的局部偏离影响率 $LIR_k(p)$为

$$LIR_k(p)=\frac{\sum_{o\in N_k(p)} LDR_k(o)}{|N_{k-\text{distance}}(p)|} \quad (3)$$

其中，分子是对象 p 的 k 距离邻居集中对象的 LDR 和。对象 p 的局部偏离影响率反映了 p 的 k 距离邻居集中的对象对 p 的偏离影响。

定义 5(对象 p 的局部偏离因子) 给定对象 p 的局部偏离率和局部偏离影响率，p 的局部偏离因子 $LDF_k(p)$为

$$LDF_k(p)=\frac{LDR_k(p)}{LIR_k(p)} \quad (4)$$

对象 p 的局部偏离因子反映了 p 的 k 距离邻居内邻近对象分散程度。高 LDF 值说明对象 p 周围是一个稀疏的区域，此对象很可能是一个孤立点，低 LDF 值说明该对象周围是一个密集区域，它不可能是孤立点。

2.2 算法描述

本算法的设计思想如下：先计算数据集中每个对象 p 与数据集中所有其他数据之间的距离，从这些距离中选出最小的 k 个距离，再从这 k 个最小距离中选出其中的最大者作为对象 p 的 k 距离，得到对象 p 的 k 距离邻居。然后计算对象 p 的 k 距离邻居的质心 p'，求出对象 p 与质心 p'的欧式距离，根据式(2)计算对象 p 的局部偏离率 $LDR_k(p)$。通过剪枝过程，得到孤立点的候选集。再对孤立点的候选集，根据式(3)计算对象 p 的局部偏离影响率 $LIR_k(p)$，根据式(4)计算对象 p 的局部偏离因子 $LDF_k(p)$。最后的关键是从数据集中选出前 n 个具有最大 LDF 值的对象作为孤立点。

对基于 LDF 的孤立点检测算法描述如下：

输入：数据集，k, n(k 是对象的最近邻居数，n 是前 n 个孤立点的个数)。

输出：前 n 个对象是从数据集选出的具有最大的 LDF 值。

对每个对象 p，本文算法包含如下 7 个步骤：(1)利用定义 1 计算 p 的 k 距离。(2)利用定义 2 计算 p 的 k 距离邻居。(3)根据定义 3 计算 p 的局部偏离率。(4)通过剪枝过程，得到孤立点的候选集。可以根据不同需要采用 2 种剪枝技术：1)剪枝 LDR 为零的数据对象；2)剪枝 LDR 为零的数据对象及其 k 距离邻居对象。(5)对于候选集，根据定义 4 计算 p 的局部偏离影响率。(6)对于候选集，根据定义 5 计算 p 的局部偏离因子。(7)通过局部偏离因子，降序排列数据集，前 n 个数据对象就是需要挖掘的孤立点。

由于孤立点个数远小于数据集规模，因此可以在步骤(7)中，遍历孤立点的候选集数据，每次得到最大的 LDF 值，然后输出，以提高时间效率。

3 实验与分析

本文实验环境是 P4 2.4 GHz CPU，512 MB 内存，Windows XP 专业版操作系统。算法在 Visual C++ 6.0 环境下用 C++语言实现。

对本文算法和基于局部稀疏系数(Local Sparsity Coefficient, LSC)的孤立点检测算法进行实验对比。Malik Agyemang 提出的基于 LSC 的孤立点检测算法的主要思想如下：先对数据集中每个对象，计算离它最近 k 个对象的距离，从中选出最大的距离作为该点的 k 距离，对数据集中每个对象计算与其距离不大于该对象 k 距离的邻近对象形成一个集合。然后计算每个对象与其对应集合中的所有对象之间平均距离的反比，即局部稀疏率。最后计算集合内所有对象的局部稀疏率之和与该点的局部稀疏率比值的平均比率，即 LSC，根据每个对象的 LSC 值从大到小的顺序排列整个数据集，并把前 n 个对象作为孤立点。

3.1 算法性能分析

为了验证算法的正确性，本文先用文献[5]中的二维数据进行实验。该数据集有 1 270 个记录，1 个大簇和一些分布不均匀的孤立点数据。此实验最近邻居个数 k 都设为 50，检测的孤立点个数设为 50，实验部分结果如表 1 所示。图 1 是 LSC 算法识别的孤立点结果，图 2 是本文算法识别的孤立点结果，其中，检测出的孤立点加粗表示。由实验结果可以看出，LSC 算法能识别的孤立点，本文算法也能识别。LSC 算法在图 1 左上角的大簇中识别了一些点作为孤立点，这是错误的。而本文算法没有将这个大簇中的点误识为孤立点。因此，本文算法是有效的，可以发现，随数据的 LDF 值从高到低，孤立点的分布逐渐向大簇靠拢，体现了局部概念。

表 1 本文算法和 LSC 算法检测出的孤立点比较

排列顺序	本文算法		LSC 算法	
	记录号	LDF 值	记录号	LSC 值
1	48	5.871 26	567	3.938 94
2	50	4.639 00	890	2.363 75
3	567	3.797 57	1 147	2.320 81
4	56	2.549 21	1 138	2.115 71
5	9	2.528 74	158	2.069 81

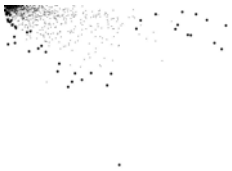

图 1 LSC 算法识别的孤立点

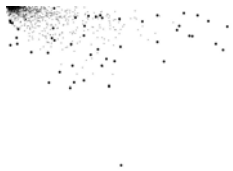

图 2 本文算法识别的孤立点

使用文献[6]一个较复杂的二维综合数据集，该数据集共有 2 178 个数据点。此实验最近邻居个数 k 都设为 100，检测的孤立点个数设为 50，实验部分结果如表 2 所示。图 3 是 LSC 算法识别的孤立点结果，图 4 是本文算法识别的孤立点结果，其中，检测出的孤立点加粗表示。由实验结果可以看出，LSC 算法无法发现该数据集的孤立点，本文算法在发现复杂分布的孤立点时，比 LSC 算法更有效。

表 2 本文算法和 LSC 算法检测出的孤立点比较

排列顺序	本文算法		LSC 算法	
	记录号	LDF 值	记录号	LSC 值
1	1 787	2.265 33	1 787	3.193 02
2	1 754	2.233 90	1 754	3.187 78
3	1 858	2.181 55	1 858	3.124 84
4	2 109	2.036 56	2 169	3.087 53
5	2 061	2.023 53	735	3.003 79

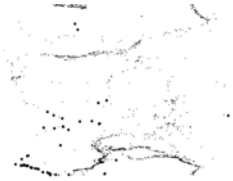

图 3 LSC 算法识别的孤立点

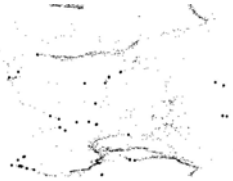

图 4 本文算法识别的孤立点

3.2 算法复杂度分析

基于 LDF 的孤立点检测算法第 1 步的时间复杂度为 $O(kN^2)$，第 2 步为 $O(N^2)$，第 3 步为 $O(kN)$，第 4 步为 $O(N)$，第 5 步为 $O(kM)$，第 6 步为 $O(M)$，第 7 步为 $O(nM)$，其中，k 是每个数据对象的最近邻居数；n 是所求的前 n 个孤立点的

个数；N 是数据集的规模；M 是候选集的规模。完成剪枝过程后，M 通常远小于 N，因此，本文算法总的时间复杂度为 $O(kN^2)$。算法的时间复杂度与每个对象的最近邻居个数成线性关系，与数据集的规模呈平方关系。LSC 算法的总的时间复杂度也为 $O(kN^2)$。

使用文献[6]的二维综合数据集，进行如下 2 个实验：

(1)k 值的执行时间。使用 2 178 条记录，实验需要用户提供 k 值和要求的前 n 个孤立点数。此实验使用了 5 个不同的 k 值，即 20, 40, 60, 80, 100。前 n 个孤立点数都设定为 10，2 个算法在不同最近邻居个数下的执行时间如图 5 所示，可以看出，对于不同 k 值，本文算法的效率都高于 LSC 算法。随着 k 的增长，本文算法的执行时间增长缓慢，而 LSC 算法的执行时间增长较快。2 个算法的执行时间近似线性增长，验证了上述关于时间复杂度的分析。

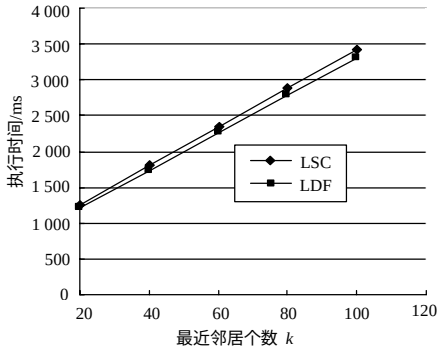

图 5　不同最近邻居个数情况下本文算法和 LSC 算法的执行时间

(2)不同数据规模的执行时间。此实验使用 2 178 条记录中的 2 100 条。数据量大小为 300 条~2 100 条记录，每次递增 300 条。前 n 个孤立点个数设为 10，每个对象的最近邻居数 k 设为 30。采用剪枝方法仅剪去 LDR 为零的数据对象。实验结果如图 6 所示，可以看出，2 个算法的执行时间近似为抛物线，验证了上述关于 2 个算法的时间复杂度与数据规模呈平方关系的分析。由图 6 可以看出，本文算法运行时间低于 LSC 算法的运行时间，符合上述时间复杂度分析。当在剪枝过程中也剪去 LDR 为零的数据对象的 k 邻居时，运行时间必然比 LSC 算法更低。随着数据集规模的增大，在发现孤立点的正确率和执行时间方面，本文算法比 LSC 算法更高效。

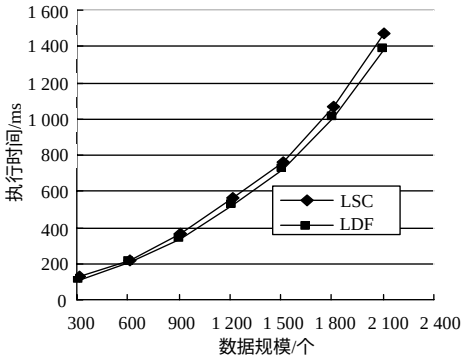

图 6　不同数据规模情况下本文算法和 LSC 算法的执行时间

4　结束语

本文提出基于 LDF 的孤立点检测算法，此算法能很好地检测出数据集中的孤立点。实验结果表明，它对孤立点的识别效果优于传统 LSC 算法。笔者的下一步工作是研究高维空间中孤立点的识别方法。

参考文献

[1] Lazarevic A, Srivastava J, Kumar V. PAKDD 2004 Tutorial: Data Mining for Analysis of RareEvents[EB/OL]. (2004-03-26). http://www.deakin.edu.au/~pakdd04/pdf/Tutorial2.pdf.

[2] Hawkins D. Identification of Outliers[M]. London, England: Chapman and Hall, 1980.

[3] Laurikkala J, Juhola M, Kentala E. Informal Identification of Outliers in Medical Data[C]//Proc. of the 15th Int'l Workshop on Intelligent Data Analysis in Medicine and Pharmacology. Berlin, Germany: [s. n.], 2000.

[4] 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. [S. l.]: ACM Press, 2000.

[5] Data Sets from the Math Forum@Drexel University. American Colleges and Universities[EB/OL]. (2000-01-29). http://mathforum.org/workshops/sum96/data.collections/datalibrary/data.set6.html.

[6] StatLib Datasets Archive. smoothmeth.[EB/OL]. (1996-03-13). http://stat.cmu.edu/datasets.

（上接第 58 页）

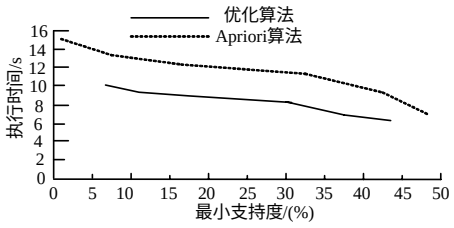

图 2　算法随支持度变化的执行时间比较

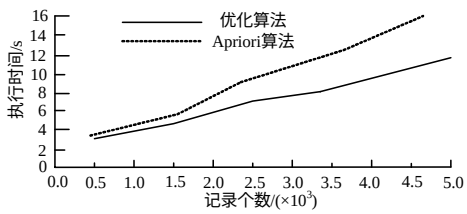

图 3　算法随记录数变化的执行时间比较

综上所述，基于两个矩阵的算法在执行效率上有独特优势，其所需存储空间极大减少，可一次调入内存进行执行，且具有支持并行计算的特点。

5　结束语

针对 Apriori 算法存在的不足，本文提出基于两个矩阵优化的关联规则挖掘算法。此算法无须重复扫描数据库，通过存储逻辑数据节省存储空间，利用逻辑运算提高运算速度。

参考文献

[1] 李新征. 一种新的高效 Apriori 算法[J]. 微计算机信息, 2006, 22(3): 193-194.

[2] 高宏宾, 潘　谷, 黄义明. 基于频繁项集特性的 Apriori 算法的改进[J]. 计算机仿真, 2007, 28(10): 73-74.

[3] 陈　凯, 冯全源. 最大频繁项集的高效挖掘[J]. 微电子学与计算机, 2005, 22(8): 22-25.