ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.5, May 2006, pp.1009−1016
DOI: 10.1360/jos171009

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于划分的孤立点检测算法*

孙焕良[1,2]，鲍玉斌[1+]，于 戈[1]，赵法信[1]，王大玲[1]

[1](东北大学 信息科学与工程学院,辽宁 沈阳 110006)

[2](沈阳建筑大学 信息与控制工程学院,辽宁 沈阳 110015)

An Algorithm Based on Partition for Outlier Detection

SUN Huan-Liang[1,2], BAO Yu-Bin[1+], YU Ge[1], ZHAO Fa-Xin[1], WANG Da-Ling[1]

[1](School of Information Science and Engineering, Northeastern University, Shenyang 110006, China)

[2](School of Information and Control Engineering, Shenyang Jianzhu University, Shenyang 110015, China)

+ Corresponding author: Phn: +86-24-83683113, Fax: +86-24-23975654, E-mail: baoyb@mail.neu.edu.cn, http://www.neu.edu.cn

Sun HL, Bao YB, Yu G, Zhao FX, Wang DL. An algorithm based on partition for outlier detection. *Journal of Software*, 2006,17(5):1009−1016. http://www.jos.org.cn/1000-9825/17/1009.htm

Abstract: Outliers are objects that do not comply with the general behavior of the data. The method of partition divides data space into a set of non-overlapping rectangular cells by partitioning every dimension into equal length. Statistical information of cells is used to find knowledge in datasets. There exists very large data skew in real-life datasets, so partition will produce many empty cells, which affects the efficiency of the algorithms. An efficient index structure called CD-Tree (cell dimension tree) is designed for indexing cells. Moreover, to guide partition and to optimize the structure of CD-Tree, the concept of SOD (skew of data) is proposed to measure the degree of data skew. Finally, the CD-Tree-based algorithm is designed for outlier detection based on CD-Tree and SOD. The experimental results show that the efficiency of CD-Tree-based algorithm and the maximum number of dimensions processed increase obviously comparing with the Cell-based algorithm on real-life datasets.

Key words: data mining; outlier detection; partition; CD-tree (cell dimension tree); cell-based algorithm

摘 要: 孤立点是不具备数据一般特性的数据对象.划分的方法是通过将数据集中的数据点分布的空间划分为不相交的超矩形单元集合,匹配数据对象到单元中,然后通过各个单元的统计信息来发现孤立点.由于大多真实数据集具有较大偏斜,因此划分后会产生影响算法性能的大量空单元.由此,提出了一种新的索引结构——CD-Tree(cell dimension tree),用于索引非空单元.为了优化 CD-Tree 结构和指导对数据的划分,提出了基于划分的数据偏斜度(skew of data,简称 SOD)概念.基于 CD-Tree 与 SOD,设计了新的孤立点检测算法.实验结果表明,该算法与基于单元的算法相比,在效率及有效处理的维数方面均有显著提高.

* Supported by the National Natural Science Foundation of China under Grant Nos.60473073, 60573090, 60173051 (国家自然科学基金); the Foundation of Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institution of China (国家教育部高等学校优秀青年教师教学和科研奖励基金); the Natural Science Foundation of Liaoning Province of China under Grant No.20052006 (辽宁省自然科学基金); the Key Technologies Plan of Liaoning Education Department of China under Grant No.05L354 (辽宁省教育厅攻关计划基金)

Received 2004-06-26; Accepted 2005-05-23

关键词: 数据挖掘;孤立点检测;划分;CD-Tree(cell dimension tree);基于单元的算法
中图法分类号: TP181 文献标识码: A

孤立点检测是数据挖掘领域一个重要的研究方向,用于发现不具备数据一般特性的数据对象.孤立点检测可以应用到许多领域,如网络入侵检测、电信和信用卡欺骗、气象预报、客户分类等.孤立点检测方法主要有基于距离的方法[1−3]、基于密度的方法[4]、基于偏离的方法[5]等.其中 Konrr 与 Ng[2,3]提出了一种基于距离的孤立点定义,设计了基于单元的孤立点检测算法(cell simplified,简称 CS).该算法将数据空间的每个维进行等间隔划分,生成不交叠的超矩形单元并匹配数据点到相应单元中,借助于单元结构的性质去发现孤立点.该算法的时间复杂度为 $O(mt^k k^{k/2}+kN)$,其中 k 为数据维数,N 为数据点数,m 是划分后生成的单元总数,t 是一个常数.CS 算法在低维情况下较为有效,但它存在以下不足:

首先,基于单元的算法对空间划分时,生成的单元数与维数呈指数增长.例如,对于一个 5 维的数据集,将每维划分为 100 个间隔段,则生成的单元数将为 100^5,大量的单元数极大地降低了算法的效率.实际上,划分后会生成较多无数据点落入的空单元.如果算法只存储非空单元,则单元间的位置关系将被破坏,使基于单元的算法需要执行的近邻查询,每次都要遍历一次所有的单元,算法的效率难以保证.为了实现近邻查询,本文试图采用传统的多维索引结构(如 R^*-tree[6],SR-tree[7],X-tree[8]等)来存储非空单元.但是,基于单元的算法所执行的近邻查询与传统的近邻查询不同,其查询范围为一个超立方体的区域,而不是传统索引结构上可查询的球形区域.因此,传统的多维索引结构难以实现超立方体区域的近邻查询.所以,本文设计了一种新的索引结构——CD-Tree(cell dimension tree),只存储了非空单元,同时保持单元间的邻居关系,使近邻查询易于实现.

另外,基于单元的算法并未给出如何确定划分粒度的方法.如果采用粗粒度划分,不能体现出算法的优势;如果采用细粒度划分,生成的单元数较多,使得算法的性能下降.为了确定算法允许的最细划分粒度,本文提出了基于划分的数据偏斜度(skew of data,简称 SOD)概念,讨论了数据偏斜度与数据划分的关系,并且利用数据偏斜度来指导划分.另外,数据偏斜度可用于优化 CD-Tree 结构,提高算法的效率.

基于上述结构和概念,提出了基于 CD-Tree 的孤立点检测算法,该算法利用 CD-Tree 结构,只保存非空的单元.与基于单元算法使用的单元数 m 相比,基于 CD-Tree 的算法需要处理的单元数大为减少.在时间复杂度方面,基于 CD-Tree 的孤立点检测算法的时间复杂度为 $O((1-SOD)mt^k k^{k/2}+kN)$,而基于单元的算法的复杂度为 $O(mt^k k^{k/2}+kN)$.在空间与时间代价方面,基于 CD-Tree 的算法均优于基于单元的算法,尤其在维度的适应性方面有较大提高.同时,SOD 可用于优化和调整划分的粒度,提高了算法的效率.实验表明,基于 CD-Tree 的算法与基于单元的算法相比,在效率及有效处理的维数方面均有显著提高.

1 CD-Tree

假设 $A=\{A_1,A_2,...,A_k\}$是有界、全部有序域的集合,$S=A_1\times A_2\times\ldots\times A_k$ 是一个 k 维数值空间,其中 $A_1,A_2,\ldots,A_k$ 表示 S 的 k 个维或 k 个属性域.$X=\{x_1,x_2,...,x_N\}$表示 S 上的 N 个点的集合,其中,$x_i=\langle x_{i1},x_{i2},\ldots,x_{ik}\rangle$表示一个数据点,$x_{ij}$ 为数据点 x_i 的第 j 维的值.

定义 **1**(数据集 ***X*** 的划分 ***P***). 将数据集 X 所分布空间的每个维划分为相等的间隔段,生成不相交超矩形单元的集合 P,它覆盖整个数据集 X 所分布的空间.每个单元 C 的空间位置表示为$\{c_1,c_2,\ldots,c_k\}$,其中 $c_i=[l_i,h_i)$ 对应于第 i 个维的一个右开的间隔段,一个单元也可以表示为$(cNO_1,cNO_2,\ldots,cNO_k)$,其中 cNO_i 是区间$[l_i,h_i)$对应的间隔序号,间隔号从 1 到间隔段总数,称 P 为数据集 X 的一个划分.

由定义 1 可知,单元的总数 $m=m_1\times m_2\times\ldots\times m_k$,其中 m_i 为第 i 维的间隔段数量.

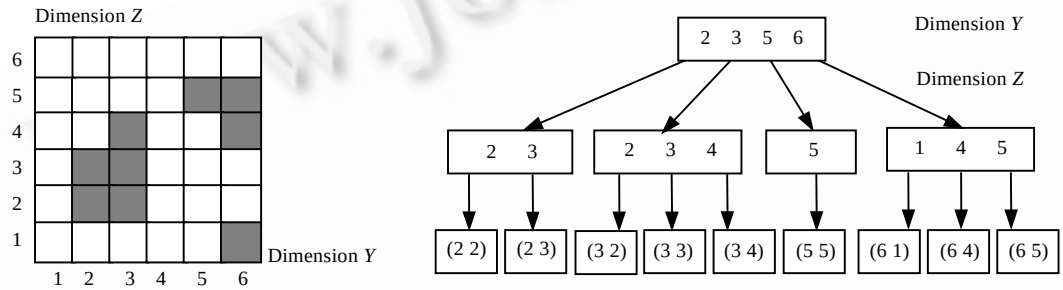
1.1 CD-Tree的结构

数据集 X 划分后生成了单元的集合,只把非空单元存入 CD-Tree 中.CD-Tree 的定义如下.

定义 **2**(**CD-Tree**). 数据集 X 在划分 P 下生成的 CD-Tree 结构如下:

1. 它有一个根结点,共有 k+1 层,其中 k 是数据集的维数.
2. 数据集的一维对应于 CD-Tree 的一层,第 k+1 层对应所有非空的单元.
3. 除了第 k+1 层,第 i 层(非叶结点层)中的结点包含形如(*cNO*,*pointer*)的内部结点,其中 *cNO* 是一个关键字,也是一个单元在第 i 维上的间隔号;*Pointer* 是一个指针,在第 k 层,它指向一个叶结点.在其他层的内部结点中,它指向下一层结点,该结点包含与本单元对应下一维的所有相异非空单元的序号.
4. 从根结点到叶结点的一条路径对应一个单元.

图 1(a)是一个划分下的单元结构,共有两维,每维等分成 6 个间隔段,编号为 1~6.黑色的单元表示非空的单元.例如 Cell(2,3)表示两维的编号分别为 2 和 3 的单元.图 1(b)表示了与这个单元结构相对应的 CD-Tree 结构.这个 CD-Tree 共有 3 层,前两层分别对应数据的两个维 Y 和 Z,最后一层为叶结点层,对应所有单元.第 1 维有 6 个间隔段,但只有 2,3,5 和 6 包含数据对象,这样,根结点有 4 个内部结点.根结点的第 1 个内部结点 2 指向的第 2 层结点有两个间隔号 2 和 3,说明在第 2 维上与第 1 维的间隔号 2 相对应有两个单元非空,分别是(2,2)和(2,3).从根结点到叶结点的一条路径为一个单元.在叶结点上要存储这个单元上的一些信息,如落入该单元的数据点数等.CD-Tree 的构建过程是,计算数据对象落入单元的坐标值,然后在相应的层次的结点中,以关键字按升序插入结点.



(a) Structure of a cell (a) 一个单元的结构

(b) A CD-Tree (b) 一棵 CD-Tree

Fig.1 An example of a CD-Tree

图 1 一棵 CD-Tree 的实例

1.2 CD-Tree上的近邻查询

基于单元的孤立点检测算法使用了一种近邻的结构检测孤立点.其定义如下:

定义 **3**(单元 ***C*** 的 ***R***-近邻). 在空间划分的结构中,单元 C 的 R-近邻是指以 C 为中心,以 R 为厚度的超立方体区域内所包含的所有单元的集合.其中,R 是一个自然数,表示近邻的厚度.

如图 2 所示为一个空间划分后的单元结构.单元 C(黑色)的 1-近邻是指其外层 1 倍单元厚度范围内包含的所有单元,在图中用灰色表示,共 8 个;2-近邻是指 2 倍单元厚度范围所包含的所有单元,共 24 个单元.CD-Tree 上的近邻查询是指在 CD-Tree 上获得 R-近邻的查询.对于二维数据集,CD-Tree 上的近邻查询的范围为一个正方形的区域;大于三维的数据集近邻查询的范围是一个超立方体结构.基于 CD-Tree 的近邻查询算法见算法 1.

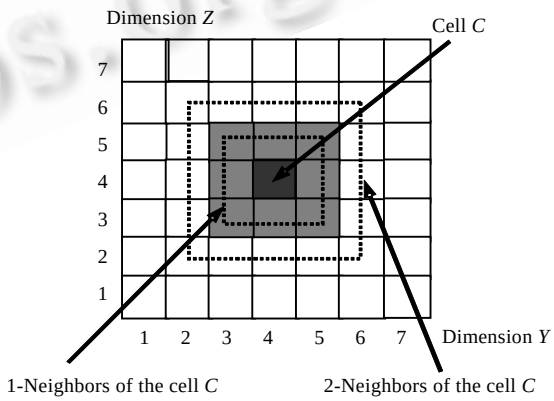


Fig.2 R- Neighbors search

图 2 R-近邻查询

算法 **1**. RangeQuery(*Current*,*CellCoo*,*Radius*,*Level*,*Result*).

输入:CD-Tree,单元的坐标为 *CellCoo*,查询厚度为 *Radius*,CD-Tree 的当前层为 *Level*.

输出:*Result* //查询结果.

1. *Size*=当前结点的内部关键字的个数;
   DimCoo=*CellCoo.CurrentDimensionKey*; //取单元的当前维坐标值
2. **if** (*DimCoo*+*Radius*+1>*Size*) **then**
   BeginPosition=*Size*−1;
3. else
   BeginPosition=*DimCoo*+*Radius*; //否则定位于 *DimCoo*+*Radius*
4. **for** (int *i*=*BeginPosition*; *i*>−1; *i*−−) //从后向前搜索
   a. **if** (*DimCoo*−*Radius*>当前关键字的值) **then** goto 5**;**
   b. **else**
      if ((*DimCoo*+*Radius*)<当前关键字的值) **then** goto 4;
   c. **if** (*Level*≠*Dim*+1) **then** //如果存在下一层,则向下递归
      RangeQuery(*Current.NextNode*,*CellCoo*,*Radius*,*Level*+1,*Result*);
   d. **else**
      Result.Add(*Current*, *i*);
5. return;

2 数据偏斜度(SOD)

由于划分后非空单元数量与数据偏斜程度有关,本文试图找到一种度量来估计划分后的非空单元数,进而指导划分.传统的衡量数据偏斜的度量,如方差,不能用来估计非空单元数.在图 3(a)中,方差小的数据集划分后生成的非空单元数较少;在图 3(b)中,方差大的数据集划分后生成的非空单元数也较少;在图 3(c)中,方差大小介于前两者之间的数据集,划分后生成的非空单元数最多.这是因为划分后生成的非空单元所占的比例表示数据填充数据空间的程度,而方差衡量的是数据与其均值相对应的偏斜程度,与划分无关.本节介绍一种新的衡量数据偏斜程度的度量,用于衡量数据空间划分后无数据点落入的空单元所占比例,进而指导对数据的划分及优化 CD-Tree 结构.

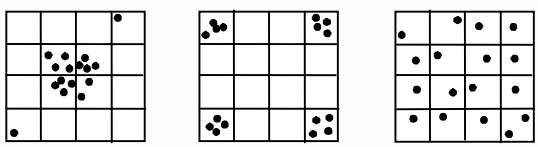


Fig.3 SOD in different partitions

图 3 不同划分下的偏斜度

定义 **4**(数据集 ***X*** 在划分 ***P*** 下的偏斜度 ***SOD***(***X***,***P***)). 数据集 *X* 在划分 *P* 下的偏斜度 *SOD*(*X*,*P*)定义为

$$\frac{1}{2N}\left(\sum_{i=1}^{m}\left|p_i-\mu\right|\right).$$

其中,*N* 为数据点数,p_i 为第 *i* 个单元的数据点数,*m* 为单元个数,$\mu=N/m$ 是分布在单元格中的平均数据点数.

SOD(*X*,*P*)表示空间被数据填满的程度,有如下性质:

性质 **1**. 偏斜度 *SOD*(*X*,*P*)∈[0,1]表示数据集 *X* 在 *P* 划分下的偏斜程度.当 *SOD*(*X*,*P*)=0 时,无偏斜;*SOD*(*X*,*P*)越接近于 1,其偏斜程度越大.

定理 **1**. 一个数据集 *X* 在 *P* 划分下的偏斜度等于 $\frac{1}{N}\left(\sum_{j=1}^{v}\left(p_j-\mu\right)\right)$,其中 p_j 为数据点数大于平均数据点数μ的单元的数据点数(证明略).

定理 1 说明,当计算偏斜度时,不必访问所有单元,只需访问数据点数大于平均点数μ的单元,这样简化了偏斜度的计算.

定义 **5**(子划分(**sub-partition**)). 给定数据集 X 的两个划分 P_1 和 P_2,如果$\forall C_1\in R_1,\exists C_2\in P_2$,使得 $C_1\subset C_2$,即对于 C_1 所有维上的间隔段$[l_{1i},h_{1i}]$,在 C_2 中均存在间隔段$[l_{2j},h_{2j}]$满足$[l_{1i},h_{1i}]\subset[l_{2j},h_{2j}]$,则称 P_1 为 P_2 的子划分,表示为 $P_1\subset P_2$.

定理 **2**. 给定数据集 X 的两个划分 P_1 和 P_2,如果 $P_1\subset P_2$,则 $SOD(X,P_1)\geq SOD(X,P_2)$(证明略).

由定理 2 可知,划分的粒度越细,偏斜度越大.这样,可以通过计算粗粒度划分下的偏斜度来估计细粒度划分下的偏斜度:首先,计算相对粗粒度下的偏斜度(因为粗粒度易于划分);然后通过粗粒度与细粒度下偏斜度之间的关系来选择一个合适粒度的划分.假设 P_1 和 P_2 是对数据集 X 的两个划分,且 $P_1\subset P_2$,令 n_1,n_2 分别为 P_1 和 P_2 下的非空单元数.因为偏斜度近似等于空单元的比例,则有 $n_1\approx(1-SOD(X,P_1))\times m_1,n_2\approx(1-SOD(X,P_2))\times m_2$.其中,$m_1,m_2$ 分别为 P_1 和 P_2 划分下的单元总数.根据定理 2,$SOD(X,P_1)\geq SOD(X,P_2)$,则 $n_1\leq(1-SOD(X,P_2))\times m_1$,即 n_1 的上限为$(1-SOD(X,P_2))\times m_1$.这样就得到了细粒度下生成单元数的上限,根据现有数据集的大小、内存的使用情况来选择一个合理的划分.

定理 **3**. 给定数据集 X 及其一个划分 P,当 $m\geq N$ 时,$SOD(X,P)$等于划分 P 下空单元数占单元总数的比例.其中,N 为数据对象个数,m 为单元总数.

证明:由定义 4, $SOD(X,P)=\dfrac{1}{2N}\left(\sum_{i=1}^{m}\left|p_i-\mu\right|\right)$,设 v 为选择度大于平均选择度μ的单元数,s 为选择度小于μ且大于 0 的单元数,q 为划分后空单元所占总单元数的比例.那么,

$$SOD(X,P)=\frac{1}{2N}\left(\sum_{j=1}^{v}\left(p_j-\mu\right)-\sum_{i=1}^{s}\left(p_i-\mu\right)+\sum_{i=1}^{qm}\mu\right).$$

由定理 1 可知,$SOD(X,P)=\dfrac{1}{N}\left(\sum_{j=1}^{v}\left(p_j-\mu\right)\right)$,则 $SOD(X,P)=\dfrac{1}{2}SOD(X,P)-\dfrac{1}{2N}\sum_{i=1}^{s}\left(p_i-\mu\right)+\dfrac{1}{2N}qm\mu$.因此,

$$SOD-q=\frac{1}{N}\left(\sum_{j=1}^{s}\left(\mu-p_j\right)\right),\ \text{如果 } m\geq N,\text{则}\mu\leq 1.$$

由于选择度为大于等于 1 或等于 0 的值,则 $s=0$,即 $SOD-q=0$. □

由定理 3 可知,利用偏斜度估计的空单元数与实际空单元数的误差为 $m(SOD-q)$.当 $m\geq N$ 时,$m(SOD-q)=0$,误差为 0.

3 基于 CD-Tree 的孤立点检测算法

当划分的粒度变细或维数增高时,基于单元的算法效率显著下降,甚至无法工作.另外,基于单元的算法对划分的粒度粗细程度没有明确的指导,不便于使用.基于 CD-Tree 的孤立点检测算法的基本思想与基于单元的算法相同(见算法 2),其总体步骤如下:

1. 通过预计算数据集在粗粒度划分下的偏斜度及维偏斜度,估计合理划分(步骤 1),具体方法见第 2 节.
2. 动态生成有点的单元、匹配单元及其中的数据对象到 CD-Tree 中(步骤 2).
3. 在 CD-Tree 上通过近邻查询及相关性质来检测孤立点(步骤 3、步骤 4).

步骤 3 计算每个单元中的数据点数,如果单元中的点数大于 M,则将该单元标记为 *red*,并将其第 1 层邻居的所有单元标记为 *pink*,表示这两类单元中的点均不是孤立点,在下一步的考察中不予考虑.步骤 4(a)统计 *white* 单元 C_w 与其第 1 层邻居中的数据点数 $Count_{c_2}$,如果 $Count_{c_2}$ 大于 M,也将该单元标记为 *pink*(步骤 4(b)),下一步的考察中也不予考虑.当 $Count_{c_2}$ 不大于 M 时,进一步计算 $Count_{c_2}$ 与 C_w 第 2 层邻居之和 $Count_{c_3}$ (步骤 4(c)).如果 $Count_{c_3}$ 小于 M,说明这个单元中所有的点均为孤立点(步骤 4(c)(1)).对于不满足以上条件的单元,需要将该单元中的对象与其第 2 层邻居中的所有对象逐一进行比较(步骤 4(c)(2)).

算法 **2**. FindOutsCDTree.

输入:数据集 X,M.

输出:孤立点集.

1. 预计算数据集的偏斜度,估计 d,选择 CD-Tree 维次序.
2. 调用 CreateCDTree(),匹配数据对象到 CD-Tree 中.
3. **for** 对每个单元 C_q
   a. **if** $Count_q>M$ **then** 标记 C_q 为 *red*;
   b. RangeQuery($Root$,C_q,1,1, $Result_{L_1}$); //查询 C_q 的 1-邻居,即 C_q 的第 1 层邻居
   c. **if** $C_r\in Result_{L_1}$ 已标记为 *red* **then** 标记 C_r 为 *pink*
4. **for** 每个非空的 *white* 单元 C_w
   a. $Count_{c_2}=Count_c+\sum_{i\in L_1(Cw)}count_i\ \sum_{i\in L_1(Cw)}count_i$
   b. **if** $Count_{c_2}>M$ **then** 标记 C_w 为 *pink*
   c. **else** $Count_{c_3}=Count_{c_2}+\sum_{i\in L_2(C_w)}count_i$
      1. **if** $Count_{c_3}<=M$ **then** 输出 C_w 中的所有对象为孤立点
      2. **else for** 每个 *white* 单元 C_m
         i. RangeQuery($Root$,C_m,$\lceil 2\sqrt{k}\ \rceil$,1, $Result_{L_2}$); //查询 C_m 的$\lceil 2\sqrt{k}\ \rceil$-邻居
         ii. RangeQuery($Root$,C_m,1,1, $Result_{L_1}$);
         iii. $Cell_{L_2}=Result_{L_2}-Result_{L_1}$ //计算 C_m 第 2 层邻居
         iv. **for** 每个 $C_n\in Cell_{L_2}$
            1. **for** C_m 中每个数据对象 Q
               a. $Count_Q=Count_{w_2}$;
               b. **for** C_n 中每个数据对象 Q'
                  1. **if** $dist(Q,Q')<=d$ **then** $Count_P=Count_P+1$;
                     i. **if** $Count_Q>M$ **then** 标记 Q 为 *red*; goto 4(c)(2)(iv)(1).
5. 输出所有 *white* 单元中的点作为孤立点.

从空间代价方面考察,基于 CD-Tree 的孤立点检测算法利用 CD-Tree 存储非空的单元.如果基于单元的算法划分生成的单元数为 m,则保存到 CD-Tree 上的单元数为$(1-SOD)\times m$,其中 SOD 为数据集在一个划分下的偏斜度.又因为大多数真实数据集中的偏斜度较大(实验部分对真实数据集的偏斜度进行了计算),所以$(1-SOD)\times m<<m$.从时间复杂度方面考察,步骤 1、步骤 2 的时间代价为 $O(N)$及 $O(n)$,其中 n 为非空单元数;步骤 3 的时间代价为 $O(N/M)$;步骤 4 的代价是 $O(n(2\lceil 2\sqrt{k}\ \rceil+1)^k)=O(nc^kk^{k/2})$.那么,算法总代价为 $O(nc^kk^{k/2}+kN)=O((1-SOD)mc^kk^{k/2}+kN)$,而基于单元的算法的复杂度为 $O(mc^kk^{k/2}+kN)$[1].由于真实的数据集中数据偏斜的存在,导致$(1-SOD)\times m<<m$.

通过以上分析,基于 CD-Tree 的算法优于基于单元的算法,尤其在适应的维数方面有较大提高.另外,Knorr 与 Ng 设计了基于磁盘的算法,主要是解决数据量大时内存不足的问题.本文也设计了基于 CD-Tree 的磁盘算法,在数据量较大情况下同样有效,而在维数上有也所提高,详见第 4 节.

4 实验分析

在这一节中,我们测试所提出算法的有效性及性能.实验的运行环境是:操作系统为 Windows 2000 Server,512MB 主存,CPU 2.5GHz.一种数据集为文献[1]使用的 NHL 数据集,该数据集是美国曲棍球联合会的数据集,记录所有曲棍球运动员的信息,如运动员的个人资料、每一场比赛的得分情况等;另一种为图像傅立叶系数数据集,该数据集广泛用于多维索引(如 X-Tree[8])的测试中.本文约定一些缩写符号表示实验中的不同算法:CS 是文献[1]中的基于单元的算法;NL(nested loop)是文献[1]中的 Nested-Loop 算法;CDT(cell dimension tree)是基于 CD-Tree 的算法;CS-d 是文献[1]中的基于单元的磁盘算法;CDT-d 是基于 CD-Tree 的磁盘算法.

4.1 偏斜度实验分析

本节计算了一些真实数据集(如 NHL 数据集[1]、NBA 数据集[3])的偏斜度.同时,利用图像数据集对数据集偏斜度规律进行了实验分析.为了保证每个维赋予同样的权重,把数值型的字段采用最小-最大规范化(min-max normalization)方法,规范化到(0,1)之间,每个维划分为 4 份.其中 *SOD*,*q* 分别表示数据集在一个划分下的偏斜度及空单元所占的比例.图 4 显示:当数据集大小不变、维数增加时,导致 *SOD* 及 *q* 增加.其中,维数为 3,数据集的大小为 50 000 元组.图 5 显示:当维数保持不变、数据集大小增加时,导致 *SOD* 及 *q* 降低.其中,维数为 5,每维划分的间隔数为 5.图 6 显示:划分的粒度越细,*SOD* 及 *q* 的值越大,*SOD* 与 *q* 的差值越小.其中,元组数为 50 000,维数为 3.

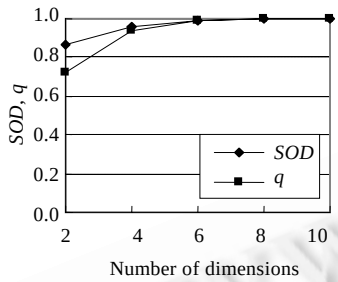


Fig.4 *SOD* and *q* in different dimensions

图 4 不同维数下的 *SOD*,*q* 值

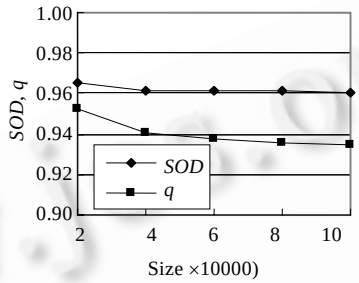


Fig.5 *SOD* and *q* in different dataset size

图 5 不同大小数据集下的 *SOD*,*q* 值

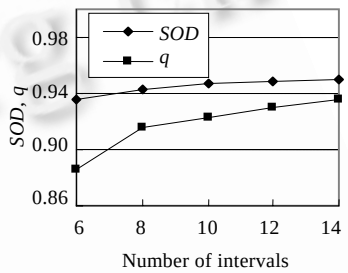


Fig.6 *SOD* and *q* in different interval number

图 6 不同划分间隔数下的 *SOD*,*q* 值

4.2 图像数据集算法性能比较

本文在相同的环境下实现了相关算法.如图 7 所示为内存算法在不同数据集大小下的测试结果,其中数据维数为 3,$M=(1-0.999)\times N$,N 为数据点数.假设调整 d 的值,以得出相同比例的数据作为孤立点.如对于有 100 000 个数据点的数据集,CDT 的时间开销是 CS 的 1/6,而 NL 的时间开销比 CS 的更大.对于上例中的数据集,由于偏斜度的存在,CDT 算法的单元数为 1 788,远小于 CS 的 195 112.因此,CDT 的性能优于 CS.图 8 给出了两个算法在维数变化情况下的执行时间(其余参数设置相同),CDT 的性能优于 CS.图 9 表示了基于磁盘的算法测试结果,实验的数据集是三维数据集,数据集大小从 20 万到 100 万条元组,读取的页面大小为 100 条元组,设置的参数相同($M=(1-0.9999)\times N=0.06$),CDT-d 算法的性能明显优于 CS-d 算法.

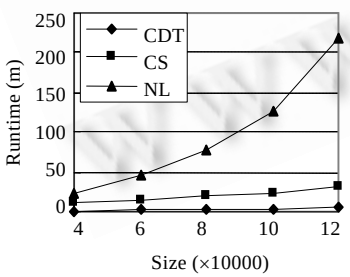


Fig.7 Performance comparison in different dataset size

图 7 不同数据集尺寸下的测试结果

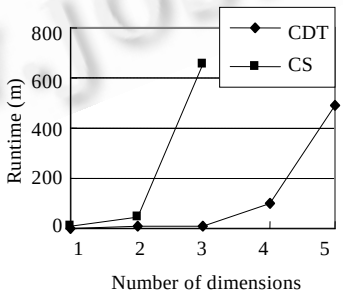


Fig.8 Performance comparison in different dimensions

图 8 不同维数下的测试结果

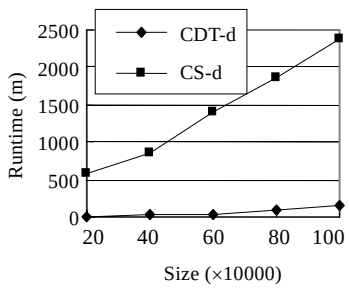


Fig.9 Query runtime for disk-based algorithms

图 9 基于磁盘的算法执行时间

5 结 论

孤立点检测是数据挖掘中一个重要的研究问题.基于单元的孤立点检测算法适用于低维情况,当维数增加以及划分粒度较细时,由于生成单元数过多,算法不能有效地工作.针对这个问题,我们设计了一种新的索引结构 CD-Tree,只保留有数据点落入的非空单元,且该结构保持了单元间的邻居关系,使近邻查询易于执行.

本文提出了基于划分的数据偏斜度的概念.偏斜度能够有效衡量划分下的数据偏斜程度,进而用于估计生成的非空单元数,指导数据的划分及优化 CD-Tree 结构.基于 CD-Tree 的算法在空间上需要存储$(1-SOD)m$个单元,而基于单元的算法需要存储 m 个单元.在时间复杂度方面,基于 CD-Tree 的算法的复杂度为$O((1-SOD)mt^kk^{k/2}+kN)$,而基于单元的算法的复杂度为$O(mt^kk^{k/2}+kN)$,其中 t 是一个常数.因此,对于偏斜度较大的数据集,基于 CD-Tree 的算法更有优势.实验表明,基于 CD-Tree 的孤立点检测算法较之基于单元算法,在效率及有效处理的维数方面均有显著提高.

References:

[1] Knorr E, Ng R. Algorithms for mining distance-based outliers in large data sets. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the VLDB Conf. New York: Morgan Kaufmann Publishers, 1998. 392−403.

[2] Knorr E, Ng R. Finding intensional knowledge of distance-based outliers. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the VLDB Conf. Edinburgh: Morgan Kaufmann Publishers, 1999. 211−222.

[3] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. In: Chen WD, Naughton JF, Bernstein PA, eds. Proc. of the ACM SIGMOD Conf. Dallas: ACM Press, 2000. 427−438.

[4] Breunig MM, Kriegel HP, Ng R, Sander J. LOF: Identifying density-based local outliers. In: Chen WD, Naughton JF, Bernstein PA, eds. Proc. of the ACM SIGMOD Conf. Dallas: ACM Press, 2000. 94−104.

[5] Arning A, Agrawal R, Raghavan P. A linear method for deviation detection in large databases. In: Simoudis E, Han JW, Fayyad UM, eds. Proc. of the KDD Conf. Portland: AAAI Press, 1996. 164−169.

[6] Beckmann N, Kriegel HP, Schneider R, Seeger B. The *R**-tree: An efficient and robust access method for points and rectangles. In: Hector GM, Jagadish HV, eds. Proc. of the ACM SIGMOD Conf. Atlantic: ACM Press, 1990. 322−331.

[7] Katayama N, Satoh S. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In: Peckham J, ed. Proc. of the ACM SIGMOD Conf. Tucson: ACM Press, 1997. 369−380.

[8] Berchtold S, Keim DA, Kriegel H. The X-tree: An index structure for high-dimensional data. In: Vijayaraman TM, Buchmann AP, Mohan C, Sarda NL, eds. Proc. of the 22nd VLDB Conf. Bombay: Morgan Kaufmann Publishers, 1996. 28−39.

孙焕良(1969−),男,黑龙江望奎人,博士,副教授,主要研究领域为数据仓库,数据挖掘.

赵法信(1974−),男,博士生,工程师,主要研究领域为数据挖掘,模糊数据库.

鲍玉斌(1968−),男,博士,副教授,CCF 高级会员,主要研究领域为数据仓库,数据挖掘.

王大玲(1962−),女,博士,教授,CCF 高级会员,主要研究领域为数据挖掘,Web 挖掘.

于戈(1962−),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库理论与技术.