一种新的基于网格的边界点检测算法

A Boundary Points Detector Based On Grid

邢　剑　（新疆大学信息科学与工程学院 乌鲁木齐 830046）
刘胜全　（新疆大学现代教育中心 乌鲁木齐 830046）
田　军　（新疆大学信息科学与工程学院 乌鲁木齐 830046）
田国忠　（新疆工业高等专科学校计算机工程 830091）

摘　要： 本文针对目前数据挖掘中边界点检测的效率低等问题，本文提出了基于网格中边界格和根据对象的邻域中数据分布特点进行边界点的检测算法 BPDG（a Boundary Points Detector based on Grid）。实验结果表明 BPDG 能在含有噪声点/孤立点的不同形状、大小的数据集上更加迅速而有效地检测出边界点。

关键词： 数据挖掘 边界点 边界格 邻域 密度

1　引言

信息技术的发展使得大量数据被收集，如何从这些数据中快速而有效的发现有用的信息是十分重要的工作。这些技术包括数据分类、数据聚类、孤立点分析、数据清理和数据预处理技术等[1]。

本文研究了数据挖掘中边界点检测问题。边界点是分布于稠密分布的数据边缘的数据点。在数据挖掘应用方面，因为边界点可能潜在的被错分类别，所以对聚类和分类这样的数据挖掘任务来说，边界点是有用的，而进一步提高边界点检测效率也具有积极的意义。虽然在数据挖掘应用中，检测聚类和分类的边界点很重要，但是目前国际国内涉及到的边界点研究算法并不多，主要有 BORDER[1] 算法和 BRIM[2] 算法。

参考文献[1]中 BORDER 边界点检测该算法有以下不足：

算法需要找出每个对象的 K－近邻，进而计算出每个对象的反向 K－近邻个数，算法的时间复杂度是 $O(KN^2)$，N 是数据规模的大小，因此算法的执行效率不高；

参考文献[2]中 BRIM 算法虽然可以在含有噪声的数据集中有效地检测出边界点，但是该算法仍然有些不足：

① 给定一个数据集，用户很难估计邻域半径 Eps 和边界度阈值 δ_0 的大小；

② 算法仍然是处理每一个对象，其算法的时间复杂度为 $O(N^2)$，N 为数据集的大小，因此算法的执行效率也不是很高；

针对以上算法的缺点，本文提出了利用网格技术来检测聚类边界点的算法。基于网格的方法不是考虑每一个点而是考虑网格单元，而且该算法不是处理所有的网格单元而是处理所有网格单元中的边界格内的对象，这样就除去了大量不用考虑的对象，从而大大地提高了算法效率。该算法可以在含有噪声的数据集上将边界点与聚类中的噪声点分离出来，比 BRIM 有更高的执行效率。最后通过仿真试验表明本方法是有效的可行的。

2　基本概念

边界点是分布于稠密分布的数据边缘的数据点。在数据挖掘应用方面，需要注意，边界点不同于孤立点。孤立点是分布于稀疏区域的点，而边界点是分布于稠密区域边缘的点[3]。

参考文献[2]是利用对象的邻域中数据的分布特点进行边界点的检测算法。并定义了密度吸引点。

定义 1[2]：对象 P 的密度吸引点 O 记作 Attractor（p）：如果满足 $o \in N_{Eps}(p)$，$N_{Eps}(p) = \{q \in D \mid dist(p, q) \leq Eps\}$ 其中 D 表示数据集，dist（p，q）表示对象 p，q 的欧式距离。

该算法的主要思想：

Step1：找出对象 p 的 Eps 邻域内密度吸引点 o，以向量 $\overrightarrow{po}$ 为始向量；

Step2：搜索对象 p 的 Eps 邻域内异与对象 p，o 的任意对象 q，判断向量 $\overrightarrow{po}$ 与 $\overrightarrow{pq}$ 的内积是否大于 0，若大于 0，对象 q 为正半邻域的对象，否则为负半邻域的对象，并统计各个邻域内对象个数；

Step3：再依据边界度 δ（正半邻域的对象个数与负半邻域的对象个数之比乘以两个邻域内的对象个数正差值）是否大于预先设定的阈值 δ_0 来进行边界点的检测。

3 BPDG 算法

3.1 BPDG 算法概念

设 $A=\{A_1,A_2,\cdots,A_d\}$ 是一个有界的域的集合，不妨设第 i 维上的值在区间 $[m_i,n_i]$ 中，$i=1,2,\cdots,d$，则 $S=[m_1,n_1]\times[m_2,n_2]\times\cdots,[m_d,n_d]$ 就是 q 维空间。A_i 是第 i 维的属性。

定义 2[4]：设第 i 维上的值在区间 $[m_i,n_i]$ 中，$i=1,2,\cdots,d$，将第 i 维分成 ε 个等长的区间段，则第 i 维上的第 j 个区间段记为 $(m_{ij},K_{i(j+1)})$，且任意 $(m_{i(j-1)},K_{i(j)})\cap(m_{i(k-1)},K_{ik})\equiv\varphi$（φ 表示空，$j,k=1,2,\cdots\varepsilon$，且 $j\neq k$），那么数据空间被分成的网格单元数 Gird number $=\varepsilon^d$.

定义 3[5]：一个网格单元的相邻单元是那些与该单元有相邻边界的单元格或有相邻点的那些单元格。

定义 4：第 i 个单元格内对象的个数记为 $number_i$，$i=1,2,\cdots\varepsilon^d$。若 $number_i\geq minpts$，则称第 i 单元格为稠密单元格；若 $number_i\geq minpts$ 且 $number_i\neq 0$，则称第 i 单元格为稀疏单元格；若 $number_i\geq minpts$ 且 $number_i=0$，则称 i 单元格为空单元格。

定义 5：一个格是边界格必须满足以下条件：

（1）它必须是稠密单元格。

（2）至少满足下列一个条件：

- 相邻的格中仅存在稀疏单元格。
- 相邻的格中仅存在空单元格。
- 该格具有数据空间边界。

定义 6：网格单元中样品点的中心坐标是该单元格中所有样品点各个属性求平均值 $Pointscenter_i=\frac{1}{n}\sum_{i=1}^{n}X_{ij}$，其中，n 是该单元格中样品的个数，j 表示第 j 维 $1\leq j\leq d$，X_{ij} 表示第 i 个样品的第 j 个属性，距离度量使用欧几里德距离。

定义 7：格 $grid_i$ 坐标表示为 $(x_{i1},x_{i2}\cdots x_{ir},\cdots x_{id})$ 与格 $grid_j$ 坐标表示为 $(y_{j1},y_{j2}\cdots y_{jr},\cdots y_{jd})$ 的相邻面满足以下条件：

（1）格 $grid_i$ 与格 $grid_j$ 是相邻格。

（2）格 $grid_i$ 与格 $grid_j$ 的坐标中有且仅有一个 r 满足 $x_{ir}=y_{jr}(1\leq r\leq d)$。

3.2 BPDG 算法描述

输入：D，mipts，ε，num，d，其中，D 为数据集，minpts 为密度阈值，ε 为每一维需要分的段数，num 是删除域值，d 是维数。

输出：聚类的边界点

Step1：把 d 维数据空间分成 ε^d 个不相交的网格单元，并记录每一维上的区间长度。

Step2：将数据映射到每个网格单元中，并统计每个网格单元中点的个数；如果一个单元格中点的个数大于 Minpts，就标记该单元格为稠密单元格；否则若点的个数等于 0，就标记该单元格为空单元格，点的个数不等于 0 就标记该单元格为稀疏单元格。

Step3：扫描所有的单元格，依据定义标记出边界格，并分别记录与边界格具有相邻面的单元格和相邻的单元格，分别统计具有相邻面的稀疏单元格个数。

Step4：扫描所有的单元格，若是边界格，处理具有相邻面的稀疏单元格，计算稀疏单元格中未标记的点到该边界格中样品中心的欧几里德距离，如果该距离小于所有维中的最大区间长，就把该点并到该边界格中，并标记该点。

Step5：考虑边界格中的所有点，对每一个点 x，在该边界格内和与该边界格相邻的单元格内找出点 x 的密度吸引点 y，对 $\forall z\in x$ 的 Eps 邻域，如果向量 $\overrightarrow{xz}$ 和向量 $\overrightarrow{xy}$ 之间的夹角在 $[0,90°]$ 内，则点 z 属于点 x 的稠密邻域，如果向量 $\overrightarrow{xz}$ 和向量 $\overrightarrow{xy}$ 之间的夹角在 $[90°,180°]$ 内，则点 z 属于点 x 的稀疏邻域。最后将点 x 的稠密邻域点的个数与稀疏邻域点的个数之间的比值记为点 x 的删除度，若 x 的删除度大于删除阈值 num，则 x 为边界点。

4　实验与分析

下面用二维的综合数据集来验证算法的有效性，用不同规模的数据集来验证算法的效率。实验的环境均为：Pentium Ⅳ 2.4G Hz CPU，内存为 256M，操作系统为 window XP professional，算法的编写使用 Visual C++6.0。

4.1　算法的有效性

为了验证算法的有效性，我们对多个综合数据进行了测试，由于篇幅的限制，我们从中选出几个比较典型的数据集，图 1 数据集包含 22180 个对象，图 2 数据集包含 7832 个对象，图 3 数据集包含 12101 个对象。图 4 数据集包含 16220 个对象。实验所使用的各种参数见表 1。

表 1　实验中所使用的参数

名称	参数	图 1	图 2	图 3	图 4
BRIM 算法	Eps	40	60	35	35
	δ	62	100	42	42
BPDG 算法	minpts	2	1	2	1
	ε	60	30	60	90
	num	1.5	2	1.5	2

图 1 是 BPDG、BRIM 算法在不含有噪声点的数据集上检测的边界点结果，从两个图中看出：在不含有噪声空间数据集上，两个算法都能正确检测出聚类的边界点。图 2 是在含有变化密度的聚类且噪声的数据分布比较稀疏的数据集上边界点检测，图 3 是在含有不同形状、大小的聚类且噪声的数据分布比较稠密的数据集上的边界点检测。算法 BPDG 与 BRIM 实验结果对比：BPDG 比 BRIM 更能识别出含有稠密噪声点的聚类的边界点。图 4 在含有不同形状、大小的聚类且噪声的数据分布比较稀疏的数据集上的边界点检测算法 BPDG 与 BRIM 实验结果对比。比较结果：BPDG 比 BRIM 更能识别出含有不同形状、大小且噪声点稀疏的聚类的边界点。

BPDG 比 BRIM 检测的边界点更真实地反映了各聚类的大小、形状。由此可以得出结论：BPDG 能比较好地在含噪声数据集上检测出边界点，从而验证了算法的有效性。

4.2　算法的复杂度及效率分析

表 2 是 BRIM 与 BPDG 的速度对比结果。图 5 是 2 个算法的时间对比图。

表 2　BRIM 与 BPDG 的速度对比

名称	图 1 样本时间(s)	图 2 样本时间(s)	图 3 样本时间(s)	图 4 样本时间(s)
BRIM 算法	83	10	25	47
EDGE 算法	38	6	15	28

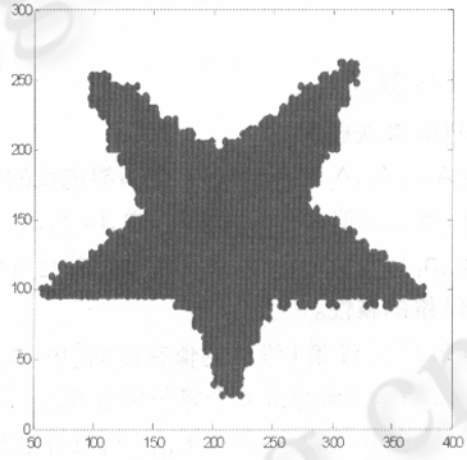

图 1　(原始数据)

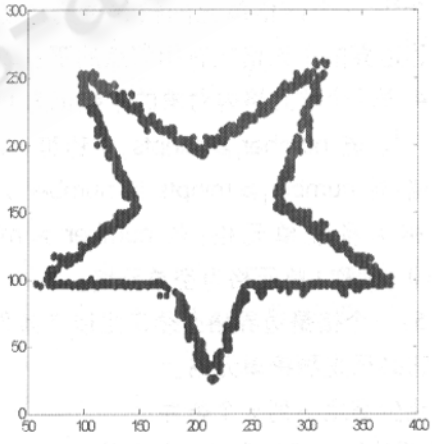

图 1　α BRIM(Eps=40;δ=62)

BRIM 算法的最耗时部分是对每个对象进行邻域搜索和计算角度的过程，其算法的时间复杂度为 O(N^2)，如果采用了空间存储结构如树，其时间复杂度

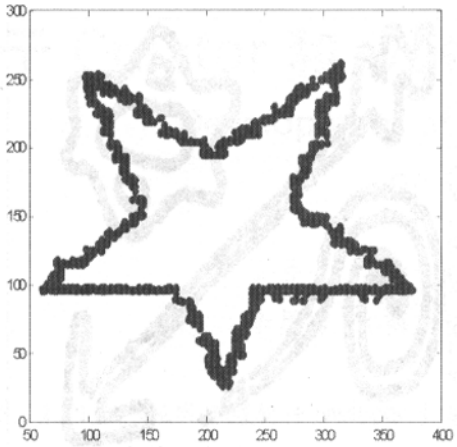

图 1　b BPDG（minpts = 2；ε = 60；num = 1.5）

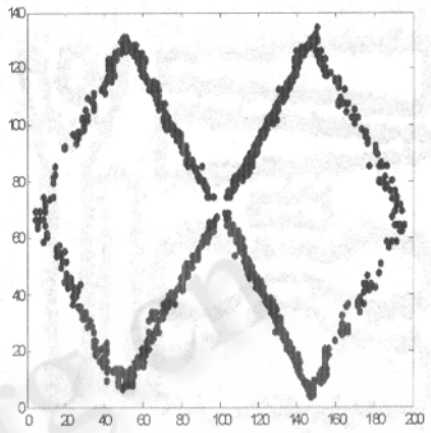

图 2　b BPDG（minpts = 1；ε = 30；num = 2）

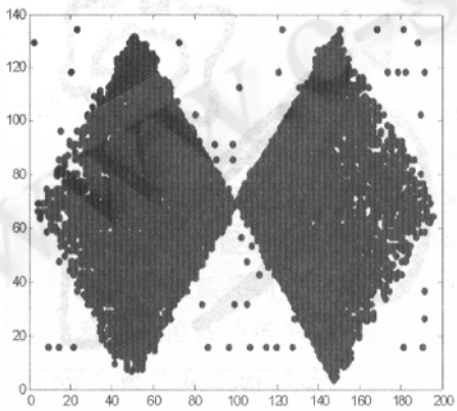

图 2　（原始数据）

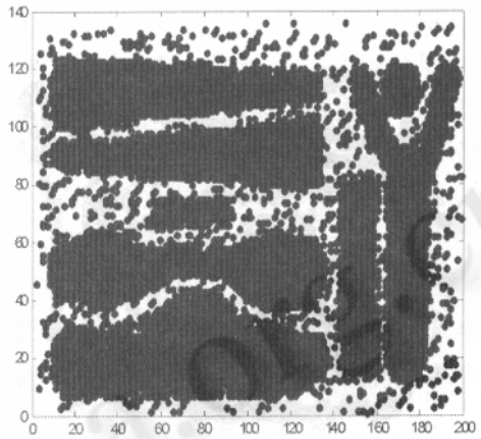

图 3　（原始数据）

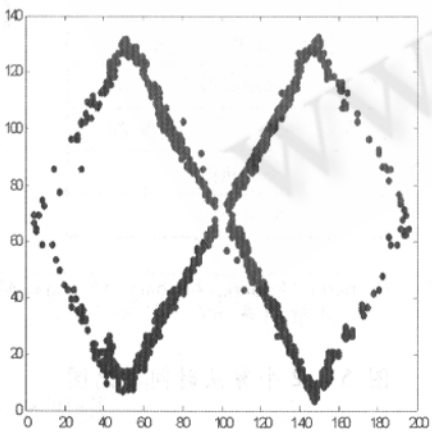

图 2　a BRIM（Eps = 60；δ = 100）

为 O（N log N），为数据集的大小；而 BPDG 利用基于网格的方法，该算法不是处理所有的网格单元而是处理边界格内的对象，这样就除去了大量不用考虑的对象，从而大大地提高了算法效率，因此，如果采用了空间存储结构如 SR 树，BPDG 算法的时间复杂度是 O（K log K），K 为边界格内数据集的大小。

5　结束语

边界点的研究已表明能够给数据挖掘带来好处，如：能够提高分类和聚类的精度。我们提出了一种新的算法 BPDG（a Boundary Points Detector based on Grid）该算法利用了基于网格中边界格和根据对象的邻域中数据分布特点进行边界点的检测思想。大量的

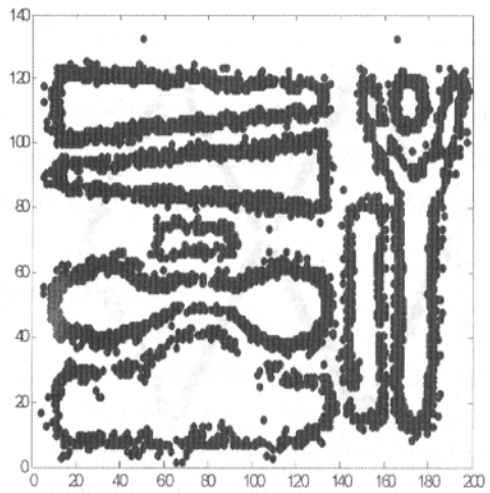

图 3　a BRIM(Eps = 35;δ = 42)

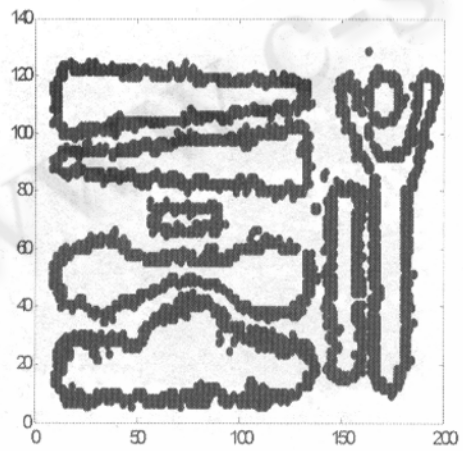

图 3　b BPDG (minpts = 2;ε = 60;num = 1.5)

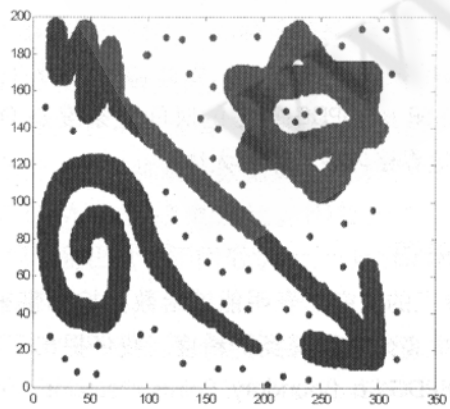

图 4　(原始数据)

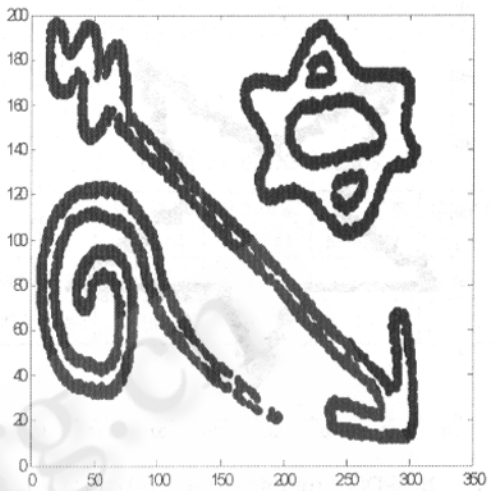

图 4　a BRIM(Eps = 35;δ = 42)

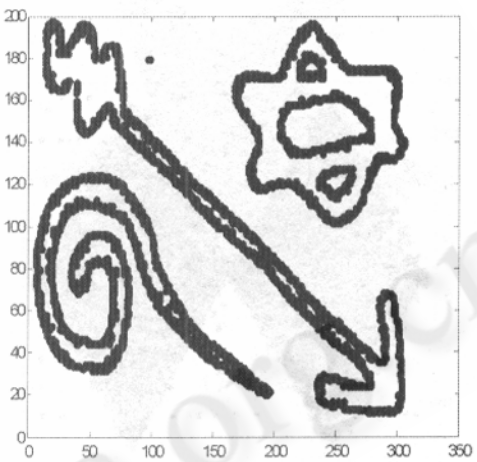

图 4　b BPDG (minpts = 1;ε = 90;num = 2)

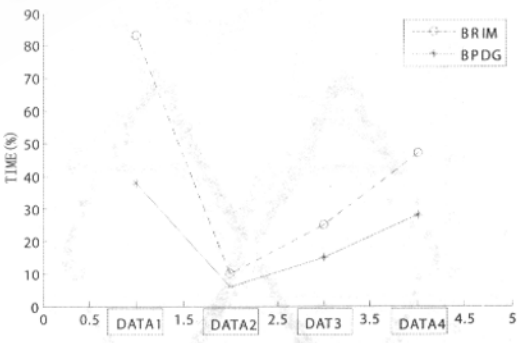

图 5　2 个算法时间对比图

(下转第 60 页)

(上接第 56 页)

实验结果证明本文提出的算法能在含有不同形状、大小噪声点的数据集上有效的、较准确的检测出边界点，在 BPDG 算法中时间复杂度明显地得到了降低。

参考文献

1 Xia C, Hsu W, Lee M L, Ooi B C. BORDER: Efficient Computation of Boundary Points. IEEE transaction on knowlBPDG and engineering [J]. 2006, 18 (3):289 - 303.

2 Qiu B Z, Yue F, Shen J Y. BRIM: A Efficient Boundary Points Detecting Algorithm [J]. PAKDD2007, LNAI4426, 761 - 768.

3 Korn F, Muthukrishnan S. Influence Sets Based on Reverse Nearest Neighbor Queries, roc. ACM SIGMOD [J], 2000:201 - 212.

4 Xia C., Lu H., Ooi B. C., Hu J. Gorder: An Efficient Method for KNN Join Processing [J]. VLDB 2004, pp:756 - 767.

5 邱保志、沈钧毅等，基于网格技术的高精度聚类算法[J]，计算机工程，2006，32(3):12 - 13.

6 Lindell Y, Pinkas B. Privacy Preserving Data Mining [c]. Advances in Cryptology - Crypto2000, Lecture Notes in Computer Science, 2000.