文章编号: 1001-0920 (2011) 01-0071-04

一种高效的基于联合熵的边界点检测算法

邱保志, 曹鹤玲

(郑州大学 信息工程学院, 郑州 450001)

摘 要: 为了快速有效地检测出聚类的边界点, 提出一种将网格技术与联合熵相结合的边界点检测算法. 该算法中网格技术用于快速查找数据集中聚类边界所在的网格范围, 联合熵用于在边界落入的网格范围内准确识别聚类的边界点. 实验结果表明, 该算法能够在含有噪声点/孤立点的数据集上, 有效地检测出聚类的边界, 运行效率高.

关键词: 边界点; 联合熵; 网格

中图分类号: TP311 **文献标识码:** A

An efficient boundary points detecting algorithm based on joint entropy

QIU Bao-zhi, CAO He-ling

(School of Information and Engineering, Zhengzhou University, Zhengzhou 450001, China. Correspondent: CAO He-ling, E-mail: shijing410@sina.com)

Abstract: In order to detect boundary points of clusters quickly and efficiently, a boundary points detecting algorithm(EDGE) is proposed, which employs grid technique and joint entropy. Grid technique is used to search the scope of grids which the boundary of clusters is located in and joint entropy is used to detect boundary points of clusters in these grids. The experimental results show that EDGE can detect boundary points of clusters in datasets with noises/outliers effectively and efficiently.

Key words: boundary point; joint entropy; grid

1 引 言

聚类分析是数据挖掘中非常有价值的研究方向之一, 它根据簇内相似度最大而簇间相似度最小的度量标准, 最大限度地将相似的对象聚成一簇, 而将不同的对象分开. 聚类分析已广泛应用于各个领域, 包括市场研究、模式识别、数据分析和图像处理[1]. 目前, 人们对聚类技术已有深入的研究, 并提出了多种聚类算法, 但对聚类边界的研究则刚刚起步.

聚类边界是指位于高密度边缘的一类数据对象, 它代表了游离在两个或多个类别之间的一类个体, 其归属并不明确, 通常具有两个或两个以上聚类的特征[2]. 由此可知, 边界点与聚类内部点不同: 边界点位于聚类的边缘, 聚类内部点位于聚类的内部. 有效地提取聚类的边界点, 不仅可以提高聚类的精度, 还可以研究聚类边缘的特性. 所以, 边界点研究非常重要.

2 相关工作

DBSCAN (density-based spatial clustering of applications with noise)[3]算法基于密度定义了聚类边界的概念: 若一个非核心数据点 p 是从核心点 q 出发直接密度可达的, 则数据点 p 为边界点. DBSCAN 所识别出聚类的边界与密度阈值 Minpts 密切相关, 给出不同的阈值, 会得到不同的聚类边界, 因而该算法不能识别多密度数据集中聚类的边界.

文献 [4] 采用限制性 k-近邻的技术来提取聚类的边界点, 以提高聚类的精度. 但该算法只考虑了聚类边界点落入低密网格中时如何将聚类的边界点与噪声分离, 没有讨论如何提取整个聚类的边界.

Xia 等人[2]提出的 BORDER 边界点检测算法利用反向 k-近邻的思想来判别聚类的边界点. 根据边界点反向 k-近邻个数小于聚类内部点反向 k-近邻个数这一特征, 将反向 k-近邻个数少的标记为边界点. 由于噪声点/孤立点的反向 k-近邻个数比聚类边界点的反向 k-近邻个数还少, BORDER 不能正确地区分噪声和聚类的边界点. 另外, 该算法还存在参数选择困难和运行效率低等问题.

BRIM[5]边界点检测算法利用正负半邻域关系来

收稿日期: 2009-11-12; **修回日期:** 2010-02-05.

基金项目: 国家自然科学基金项目(60673087); 河南省教育厅自然科学基金项目(2009A520028).

作者简介: 邱保志(1964−), 男, 副教授, 博士, 从事数据挖掘等研究; 曹鹤玲(1980−), 女, 硕士, 从事数据挖掘的研究.

判断聚类的边界点. 边界点的正向半邻域内分布着较多的点, 而负向半邻域内分布着较少的点, 两部分点的个数相差很大; 而孤立点和聚类内部点的正向、负向半邻域内的点相差不大. 利用这一特征标记边界点, BRIM 解决了 BORDER 不能区分噪声和聚类边界点的问题, 但仍然存在参数选择困难和将靠近聚类边界的噪声误判为边界点的问题.

文献 [6] 采用网格技术提取聚类的边界点, 该算法将网格梯度值大于给定阈值的网格标记为边界网格, 认为边界网格中的点即为聚类的边界点. 事实上边界网格中的点未必是边界点, 该算法没有将聚类的噪声和边界点分离, 处理精度不高.

本文提出的 EDGE (an efficient boundary points detecting algorithm based on joint entropy) 算法引入了网格的处理技术, 克服了 BORDER 和 BRIM 算法执行效率低的缺点. 同时引入了联合熵的概念来进一步判定边界网格中的点是否为边界点, 克服了文献 [6] 中算法处理精度不高的问题.

3 算 法

给定一个 d-维数据集 D, 其属性 $(A_1, A_2, \cdots, A_d)$ 均是有界的, 不妨设第 i 维上的值在区间 $(l_i, h_i]$ $(i = 1, 2, \cdots, d)$ 中, 则 $s = (l_1, h_1] \times (l_2, h_2] \times \cdots \times (l_d, h_d]$ 即为 d-维数据空间. 将每一维分割成 k 个长度相等且不相交的区间段, 每个区间段均为左开右闭等长区间, 这样将数据空间分割成 k^d 个网格单元. 网格单元在第 i 维上的长度为 $\delta_i = (h_i - l_i)/k$, 第 i 维上的第 j 个区间段可表示为

$$I_{ij} = (l_i + (j-1)\delta_i, l_i + j\delta_i],$$
$$j = 1, 2, \cdots, k.$$

一个网格单元的邻居是指与其有共同边界或有共同点的那些网格单元, 每个网格单元的邻居数为 $3^d - 1$ (处于数据空间边界的单元除外).

3.1 相关定义

定义 1 (网格密度) 一个网格单元 C 中包含的数据点个数称为网格密度, 记为 den(C). 最大网格密度为 $\mathrm{MaxDen} = \max\{\mathrm{den}(C) \mid C \text{ 是一个网格单元}\}$. 网格平均密度为 verage $= N/\mathrm{ga}$. 其中: N 代表数据集的点数, ga 代表非空网格个数.

定义 2 (点密度) 任意一点 $X = (x_1, x_2, \cdots, x_d)$ 的点密度 PointDen(X) 是以 X 为中心、网格边长为边长的网格内的数据点个数.

定义 3 (边界网格) 给定一个密度阈值 Minpts, 如果一个网格单元的密度大于等于 Minpts, 则称此网格为稠密网格; 否则称为稀疏网格. 边界网格是指一个网格单元的邻居网格中含有稀疏网格的稠密网格, 或者邻居网格中含有稠密网格的稀疏网格.

这里定义的边界网格可能是稠密网格, 也可能是稀疏网格, 因为聚类的边界点可能落入稠密网格单元, 也可能落入稀疏网格单元. 同文献 [6] 中的方法相比, 显然本文在确定聚类边界的范围时更加合理, 不会造成聚类边界点的丢失.

定义 4 (自信息量)[7] 随机事件的自信息量定义为该事件发生概率的对数的负值, 设事件 x_i 的概率为 $p(x_i)$, 则其自信息量为

$$I(x_i) = -\log I(x_i) = \log \frac{1}{p(x_i)}.$$

定义 5 (n 维随机变量的联合熵) n 维随机变量 $X = (x_1, x_2, \cdots, x_n)$ 的联合熵定义为联合自信息的数学期望, 它是 N 维随机变量 X 的不确定性的度量, 即

$$H(X) = \sum_{i=1}^{m} p(X_i) I(X_i) = -\sum_{i=1}^{m} p(X_i) \log p(X_i), \tag{1}$$

其中: 概率 $p(X_i)$ 表示点 X_i 的密度 PointDen(X_i) 与以点 X 为中心所划网格内所有点密度之和的比值.

定义 6 (二维随机变量的联合熵)[7] 二维随机变量 XY 的概率空间表示为

$$\begin{bmatrix} XY \\ P(XY) \end{bmatrix} = \begin{bmatrix} x_1y_1 & \cdots & x_iy_j & \cdots & x_my_m \\ p(x_1y_1) & \cdots & p(x_iy_j) & \cdots & p(x_my_m) \end{bmatrix},$$

其中 $p(x_iy_j)$ 满足概率空间的非负性和完备性, 且有

$$0 \leqslant p(x_iy_j) \leqslant 1, \ \sum_{i=1}^{m}\sum_{j=1}^{m} p(x_iy_j) = 1.$$

二维随机变量 XY 的联合熵定义为联合自信息的数学期望, 它是二维随机变量 XY 的不确定性的度量, 即

$$H(XY) = \sum_{i=1}^{m}\sum_{j=1}^{m} p(x_iy_j) I(x_iy_j) = -\sum_{i=1}^{m}\sum_{j=1}^{m} p(x_iy_j) \log p(x_iy_j),$$

其中概率 $p(x_iy_j)$ 表示点 (x_iy_j) 的密度与以点 XY 为中心所划网格内所有点密度之和的比值.

定义 7 (边界点) 若边界网格内点 $X = (x_1, x_2, \cdots, x_d)$ 的密度 PointDen(X) 大于等于给定阈值 Minpts, 且联合熵 $H(X)$ 小于给定阈值 β, 则称边界网格内的点 X 为边界点.

密度阈值 Minpts 主要用于区分稀疏网格和稠密网格以及边界点. 经过大量实验表明, 当 Minpts 大于等于平均网格密度时, 区分效果比较差, 选择 Minpts 小于平均网格密度; $\mathrm{Minpts} = \mathrm{Average} - \sqrt{\mathrm{MaxDen}}$ 是

一个合适的值, 在本算法中是自动生成的, 不再作为一个参数.

联合熵是信息论中的重要概念, 其大小反映了数据集中点的分布均匀程度. 联合熵越大, 表明分布越均匀; 联合熵越小, 表明分布越不均匀. 本文引入联合熵主要是用来判断边界网格中数据点分布的均匀程度. 聚类内部数据点的分布通常是均匀的, 而边界处是非均匀的, 所以联合熵值的大小反映了该点是否为聚类边界的可能性. 一个点的联合熵值越小, 说明该点是聚类边界点的可能性越大.

联合熵阈值 β 可根据最大熵定理[7]计算出最大值为 logMaxDen, β 越大取出的边界点越多; β 越小取出的边界点越少.

3.2　算法描述

EDGE 算法的主要思想是: 将数据集中数据对象映射到网格单元中, 根据密度阈值将网格分为稠密网格和稀疏网格, 并判定其是否为边界网格; 根据边界网格中点的联合熵判断边界网格中的点是否为边界点.

EDGE 算法描述如下:

输入: k (为每一维划分的区间个数) 和 β (为联合熵阈值);

输出: 聚类的边界点集合.

Step 1: 生成网格. 根据 k 值将 d- 维数据空间每一维分为 k 个长度相等、不相交的区间段, 每个区间段为左开右闭等长区间, 从而将数据空间划分成 k^d 个网格单元.

Step 2: 密度计算. 将数据集 D 中的数据点映射到网格单元中, 计算非空网格个数 ga; 计算每个网格的密度和网格平均密度 Average (Average $= N/$ga, N 为数据集的点数). 密度最大的网格密度记为 MaxDen, 根据 Minpts $=$ Average $-\sqrt{\text{MaxDen}}$, 计算密度阈值 Minpts.

Step 3: 标记边界网格. 根据密度阈值 Minpts, 将密度大于等于 Minpts 的网格标记为稠密网格, 否则为稀疏网格. 根据定义 3 标记边界网格.

Step 4: 聚类边界点检测. 根据定义 2, 计算每个边界网格内各个点的密度 PointDen(X); 根据式 (1), 计算边界网格内点的联合熵 $H(X)$; 根据定义 7, 判别该点是否为边界点.

Step 5: 输出聚类的边界点.

本算法中网格划分 k 按照文献 [8] 取值. $k = \sqrt{N}$ (N 为数据集的点数) 是一个较好的取值, 或者取此值附近的值.

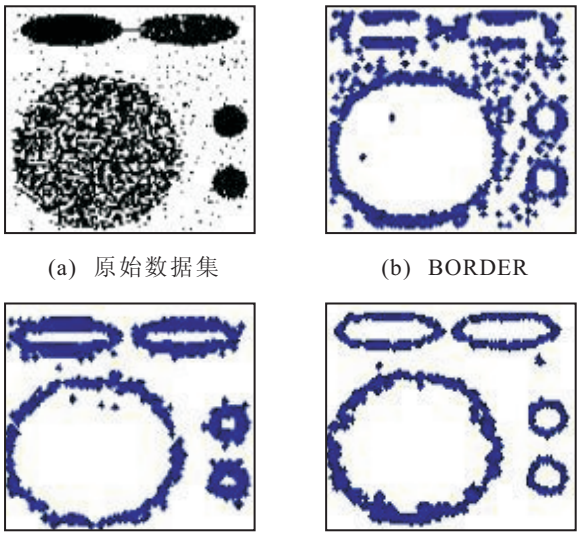
本算法对噪声点的处理分为两步: 1) 在标记边界网格时, 当稀疏网格周围均为稀疏网格时, 此稀疏网格中的点为噪声点; 2) 在边界网格内判别边界点, 当某点的密度小于给定阈值 Minpts 时, 此点为噪声点.

4　实验结果及分析

算法的实验环境: CUP 为 Intel(R) Pentium(R) processor 1.50 GHz, 内存为 512 M, 操作系统为 Microsoft Windows XP professional, 算法编写环境为 Microsoft Visual Studio 2005.

4.1　边界结果对比

1) 含有大量噪声的多密度数据集, 其形状如图 1(a) 所示[9]. 该数据集包含 5 034 个数据对象, 含有 3 个圆形和 2 个椭圆形的聚类, 半径最大的圆的密度远小于其他的 4 个聚类. 图 1(b) 是 BOEDER 算法检测的最佳结果, 使用参数为: 近邻个数 $k = 120$, 边界点个数 $n = 1\,100$. 图 1(c) 是 BRIM 算法检测出的最佳结果, 使用参数为: 邻域半径的平方 Eps $= 35$, 边界度阈值 $\delta = 55$, 检测到的边界点个数为 1 149. 图 1(d) 是 EDGE 算法的结果, 使用的参数为: $k = 90$, $\beta = 2.71$, 检测出的边界点个数为 680.



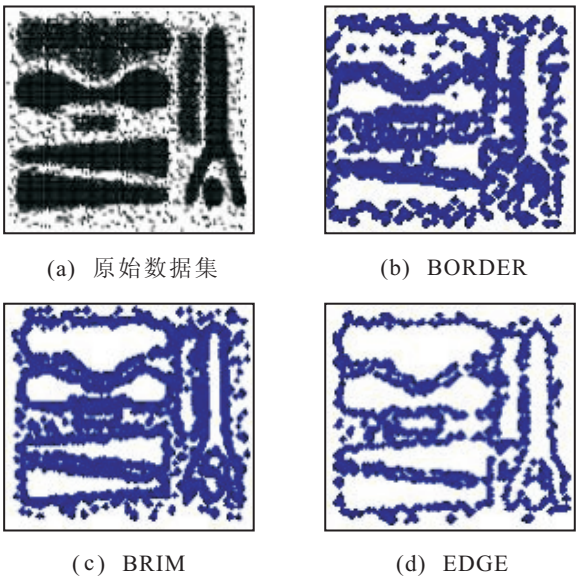
(a) 原始数据集　(b) BORDER　(c) BRIM　(d) EDGE

图 1　结果对比 (含有大量噪声的多密度数据集)

从实验结果可以看出, 在含有大量噪声的多密度数据集上, BOEDER 算法将噪声点/孤立点误认为边界点, 且丢失了 2 个椭圆形聚类的部分边界; BRIM 算法可以有效去除大量噪声, 但不能去掉靠近边界的噪声和低密度内部的个别点; EDGE 算法不但能有效去除孤立点, 而且可以有效消除噪声, 在此将聚类间的桥结线作为噪声点处理.

2) 含有不同形状、大小和密度且带大量噪声的数据集, 其形状如图 2(a) 所示[9]. 该数据集包含 21 878 个数据对象, 8 个不同形状、大小和密度的聚类, 且聚类之间距离很近. 图 2(b) 是 BOEDER 算法检测的最

佳结果, 使用参数为: $k = 70$, $n = 5\,000$. 图2(c)是BRIM算法的最佳结果, 使用参数为: Eps $= 35$, $\delta = 40$, 检测到的边界点个数为5 802. 图2(d)是EDGE算法的结果, 使用的参数为: $k = 100$, $\beta = 2.7$, 检测到的边界点个数为1 419.



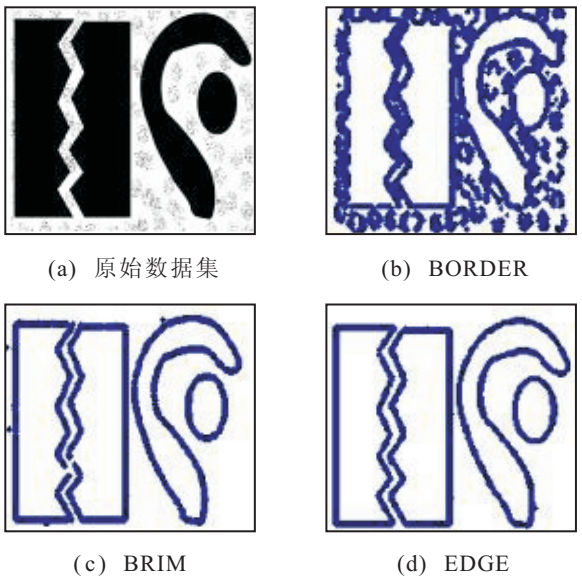
(a) 原始数据集 (b) BORDER

(c) BRIM (d) EDGE

图 2 结果对比(含有不同形状、大小和密度且带大量噪声)

从实验结果可以看出, 在含有不同形状、大小和密度且带有大量噪声、聚类之间距离很近的数据集上, BOEDER算法将大量噪声点看作是边界点, 并且将聚类的一些内部点误标记为边界点; BRIM算法将大量噪声误判为边界点; EDGE算法只是将靠近边界的少量噪声点误判为边界点.

3) 含有大量噪声的均匀密度数据集, 其形状如图3(a)所示[10]. 该数据集包含78 239个数据对象, 是含有不同形状、均匀密度的大型数据集. 图3(b)是BOEDER算法检测的最佳结果, 使用参数为: $k = 60$, $n = 7\,000$. 图3(c)是BRIM算法检测的最佳结果, 使用的参数为: Eps $= 16$, $\delta = 80$, 检测到的边界点个数为2 174. 图3(d)是EDGE算法的结果, 使用参数为: $k = 100$, $\beta = 2.9$, 检测出的边界点个数为2 728.



(a) 原始数据集 (b) BORDER

(c) BRIM (d) EDGE

图 3 结果对比(含有大量噪声的均匀密度数据集)

从实验结果可以看出, 在含有大量噪声的均匀密度数据集上, BOEDER算法将噪声点误认为边界点, 所得到的边界点图形中含有大量噪声点; BRIM算法可有效去除大量噪声点, 却不能去掉靠近边界的噪声点, 且丢失了个别边界; EDGE算法完全可以去除噪声点/孤立点, 并且边界非常清晰、完整.

4.2 算法的时间复杂度分析

EDGE算法中, Step 1的时间复杂度为$O(k^d)$; Step 2为$O(N)$, 其中N为数据集中点的个数; Step 3为$O(2k^d)$; Step 4为$O(3M)$, M为边界网格中的总点数, $M \ll N$; Step 5为$O(M)$. 所以, 整个算法的时间复杂度为$O(3k^d + N + 4M)$, 是数据点的线性函数. BORDER算法的时间复杂度为$O(kN^2)$, k为近邻的个数. BRIM算法的数据复杂度为$O(N \log N)$. 在不同规模数据集上进行实验, 执行时间对比结果如表1所示.

表 1 3种算法在不同规模数据集上运行时间对比

数据集/点数	EDGE/s	BORDER/s	BRIM/s	EDGE/BORDER/%	EDGE/BRIM/%
5 034	2	22	3	9	67
7 832	8	18	9	44	89
11 182	9	38	12	24	75
21 878	16	293	21	5	76
78 239	41	3216	821	1	5

由表1可见, EDGE算法在执行时间上远小于BORDER算法, 且数据集越大效果越明显; EDGE算法与BRIM算法相比, 在小数据集上二者相差不大, 但在大数据集上, EDGE算法比BRIM算法有显著的优势.

5 结 论

本文利用网格技术和联合熵相结合的方法, 提出了一种新的边界点检测算法. 该算法能够准确识别不同形状、大小和密度的数据集中聚类的边界, 可以有效去除噪声. 算法的时间复杂度是输入数据集点数的线性函数, 具有执行效率高、检测精度高等优点, 尤其在大型数据集上执行时间优势更加明显.

参考文献(References)

[1] Han J W, Kamber M. Data mining: Concepts and techniques[M]. 2nd ed. New York: Morgan Kaufmann, 2006: 384.

(下转第79页)