基于 Delaunay 三角网的 CBDT 聚类算法研究

李 静,陈立潮,成洪静,聂跃光
(太原科技大学 计算机科学与技术学院,山西 太原 030024)

摘 要:聚类分析是空间数据挖掘的重要方法之一。Delaunay 三角网具有良好的空间邻近特性,应用于空间聚类分析具有独特的优势,提出了一种基于 Delaunay 三角网的聚类算法——CBDT 算法,该算法采用了将 Delaunay 三角剖分得到的三角形划分为小三角形、狭长三角形和大三角形的聚类模型,通过一定规则分别以小三角形、狭长三角形为基准进行扩展从而实现聚类。CBDT 算法相对于 AUTOCLUST 算法能识别密度渐变的簇,而且计算量要比 AUTOCLUST 小得多。经实验验证,证明了该算法的有效性。

关键词:空间聚类;聚类算法;Delaunay 三角网
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2009)01-0021-04

Study of Spatial Clustering Algorithm Based on Delaunay Triangulation

LI Jing, CHEN Li-chao, CHENG Hong-jing, NIE Yue-guang
(Dept. of Computer Science & Technology, Taiyuan University of Science & Technology, Taiyuan 030024, China)

Abstract: *Clustering analysis is one of main methods of spatial data mining. Delauany triangulation has the particular property of proximity* that used for clustering analysis. In this paper, a new algorithm - CBDT (clustering of based on Delaunay triangulation) has been proposed. By dividing the Delaunay triangulation into smaller-triangles, long and narrow-triangles and bigger-triangles, obtain the clustering model, and expand these triangles respectively according some rules to clustering. CBDT has the advantages that can not only recognize the cluster of density-changing and also expend smaller compute time than AUTOCLUST. The algorithm that had been experimented is feasible.

Key words: spatial clustering; clustering algorithm; Delaunay triangulation

0 引 言

从二维空间数据中发现隐含模式或有意义的子群的聚类有很多应用,比如资源规划、市场分析、图像处理、动物群体迁徙分析、疾病扩散分析等。Voronoi 图和其对偶图 Delaunay 三角网是两个被普遍接受和采用的分析研究区域离散数据的有利工具,在地理信息系统、地学分析、计算机视觉、有限元分析和表面对象重建等领域有广泛的应用。在分析研究区域离散数据时,都可以尝试一下采用 Delaunay 三角网和 Voronoi 图的分析途径[1,2]。Delauany 三角网具有两个非常重要的性质:保证最邻近的点构成三角形,即三角形的边长之和尽量最小,且每个 Delaunay 三角形的外接圆不包含面内的其他任何点,称之为 Delaunay 三角网的空外接圆性质;最大最小角度性质:在由点集 P 中所能形成的三角网中,Delaunay 三角网中三角形的最小内角尽量最大,即三角形尽量接近等边三角形。由于这两个性质,决定了 Delaunay 三角网具有极大的应用价值。同时,它也是二维平面三角网中唯一的、最好的。Delaunay 三角网作为一种构建数据集拓扑关系的方法,广泛应用于计算机图形学和实体建模中。Delaunay 三角网能很好地表达空间数据之间的邻近关系,笔者借鉴了 AUTOCLUST 算法的思想,提出了根据 Delaunay 三角剖分后所得三角形的形状、大小聚类的思想。据此提出了 CBDT (Clustering of Based on Delaunay Triangulation)算法,该算法与 AUTOCLUST 算法相比能实现同一簇中密度逐渐变化的簇。

收稿日期:2008-04-26
基金项目:山西省自然科学基金(200501044)
作者简介:李 静(1974-),女,山东邹平人,硕士研究生,研究方向为数据挖掘;陈立潮,教授,博士,研究方向为数据仓库与数据挖掘、智能软件技术。

1 相关问题的研究

Delaunay 三角网和 Voronoi 图应用在空间聚类中

的算法有 AMOEBA、AUTOCLUST、ANDC 等。比较有代表性的算法是 AUTOCLUST 算法。该算法的优点是聚类质量高,能识别不同密度、不同形状的簇,且不需要用户输入参数等优点。

AUTOCLUST算法实现聚类的基本思想是:在 Delaunay 三角网中,与簇边界点相连的边的长度变化比与簇内部点相连的边的长度变化大得多。这是因为处于簇的边界上的点既有长边与之相连,也有短边与之相连,因此它的边的长度变化比较大。与簇内部点相连的边都是短边,边的长度变化小。AUTOCLUST 是通过区分长边、短边来划分簇的。该算法主要使用统计变量 Local_ mean(p_i)、Local_ St_ Dev(p_i)、Mean_ St_ Dev(P) 来设定长边、短边集合。其定义如下[3]:

定义 1 与点 p_i 相连的边的平均值。$d(p_i)$ 表示与 p_i 相连的边的数量

$$\text{Local\_ Mean}(p_i) = \frac{\sum_{j=1}^{d(p_i)} |e_j|}{d(p_i)}$$

定义 2 与点 p_i 相连的边的标准方差

$$\text{Local\_ St\_ Dev}(p_i) = \sqrt{\frac{\sum_{j=1}^{d(p_i)} (\text{Local\_ Mean}(p_i) - |e_j|)^2}{d(p_i)}}$$

定义 3 所有点的标准方差的平均值

$$\text{Mean\_ St\_ Dev}(P) = \frac{\sum_{i=1}^{n} \text{Local\_ St\_ Dev}(p_i)}{n}$$

定义 4 短边集合,e_j 表示与点 p_i 相连的边

$$\text{Short\_ Edges}(p_i) = \{e_j \| e_j < \text{Local\_ Mean}(p_i) - \text{Mean\_ St\_ Dev}(P)\}$$

定义 5 长边集合,e_j 表示与点 p_i 相连的边

$$\text{Long\_ Edges}(p_i) = \{e_j \| e_j > \text{Local\_ Mean}(p_i) + \text{Mean\_ St\_ Dev}(P)\}$$

定义 6 其他边集合

$$\text{Other\_ Edges}(p_i) = N(p_i) - (\text{Short\_ Edges}(p_i) \cup \text{Long\_ Edges}(p_i))$$

AUTOCLUST 是一个三阶段的算法,每一阶段是一个边的修正过程。第一阶段构建 Delaunay 三角网,并删除所有在短边集合和长边集合中的边,形成各个簇的粗糙边界。第二阶段 AUTOCLUST 恢复短边集合中的边并进行调整,主要处理簇之间的桥问题。第三阶段扩大邻近点的范围,扩充到与 p_i 路径长度不大于 2 的所有点,再使用 Local_ Mean2,$G(p_i)$ 进行进一步的调整,寻找不同区域之间的连接边。可见该算法的计算量比较大,特别是在第三阶段将邻近点的范围扩大到路径长度不大于 2 的点的范围,并且该算法不能实现渐变密度簇的识别。

2 CBDT 算法

Delaunay 三角剖分得到的三角网,在点密集的区域,三角形比较小,各边比较短;在点稀疏的区域三角形比较大,各边比较长;在密度均匀的区域,三角形接近等边三角形,在密度变化比较大的区域三角形比较狭长。基于这些特点,笔者提出了 CBDT 算法,该算法使用了将 Delaunay 三角剖分得到的三角形划分为小三角形、狭长三角形和大三角形三类聚类模型,通过一定规则分别以小三角形、狭长三角形为基准进行扩展。CBDT 算法相对于 AUTOCLUST 算法能识别密度渐变的簇,而且计算量要比 AUTOCLUST 小得多。

2.1 CBDT 算法基本概念

定义 7 Mean(T) 为三角网 T 中所有边的平均值。

定义 8 Mean_ St_ Dev(T) 为三角网 T 中所有边的标准方差的平均值。

定义 9 Max(t_i) 表示 t_i 三角形中的最大边。

定义 10 Min(t_i) 表示 t_i 三角形中的最小边。

定义 11 小三角形:Max(t_i) < Mean(T) - Mean_ St_ Dev (T)) 且 Max(t_i)/Min(t_i) < K

定义 12 狭长三角形:Max(t_i)/Min(t_i) ≥ K 且 Min(t_i) < (Mean(T) + Mean_ St_ Dev(T))

定义 13 大三角形:Min(t_i) ≥ (Mean(T) + Mean_ St_ Dev(T))

2.2 CBDT 算法描述

CBDT 算法主要分两步实现:第一步构建 Delaunay 三角网,计算 Mean(T)和 Mean_ St_ Dev(T) 的值。第二步分别选择小三角形、狭长三角形为基准三角形,然后按一定规则对其扩展,实现聚类。在 CBDT 算法中涉及到参数 k 值的确定问题。k 的取值范围根据经验通常为 1.2~2.5, 构建 Delauany 三角网是本算法的一个预处理。

2.2.1 Delaunay 三角网生成算法

目前,Delaunay 三角网的生成算法[4~7]研究的比较成熟。根据实现过程,把生成 Delaunay 三角网的各种算法分为三类:分治算法、逐点插入法、三角网生长法[5]。上述三类算法中,三角网生长法在 20 世纪 80 年代中期以后就很少用到,较常见的是分治算法和逐点插入法。而这两类算法又各有其优缺点。逐点插入法实现过程相对简单,所需内存较小,但它的时间复杂度高;分治算法时间复杂度虽然低,但由于算法中存在

递归,它需要较大内存空间。本算法采用了一种合成算法,将逐点插入法植入到了分治算法中,互相取长补短,从而达到了较好的时空性能。

2.2.2　CBDT 聚类算法

在 Delaunay 三角网生成后,计算出 Mean(T) 和 Mean_ St_ Dev(T) 的值,选择参数 k 的值,然后先以小三角形为基准三角形进行扩展,再以狭长三角形为基准进行扩展,最终实现聚类。在本算法中,基准三角形的任意一边都可以作为扩展边,由扩展边得到的三角形为扩展三角形,与扩展边相对的顶点为扩展顶点;各三角形的初始标记为 0,被扩展后标记为 1;各顶点的初始标记为 0,经过聚类以后,顶点的标记值表示该顶点属于第几个簇,f 与顶点的标记相关联,它是表示第几个簇的变量,初值为 1。经过聚类后,顶点标记值相同的为同一个簇,其中顶点标记值为 0 的为孤立点。在对三角形进行扩展的时候一定要先对小三角形进行扩展,然后再对狭长三角形进行扩展,注意先后顺序,否则得不到正确结果。以下是算法的主要步骤。

步骤 1:以小三角形为基准进行扩展。具体如下:

(1) 选取任意一标记为 0 的小三角形为起始三角形,并标记该三角形为 1; 该三角形三个顶点都标记为 f,以该三角形为基准三角形,其三边分别作为扩展边进行扩展。

(2) if (扩展三角形标记为 1) 不扩展;

(3) else

{将扩展三角形标记为 1;

if (扩展三角形 $\mathrm{Max}(t_i)/\mathrm{Min}(t_i) < k$)

{将扩展顶点标记为 f;

该三角形作为基准三角形,对其边分别进行扩展;}

else

不扩展;

}

(4) 返回步骤(2) 直至再无扩展边可以扩展。

(5) $f = f + 1$; 返回步骤(1) 直至再无小三角形可以扩展。

为了更好说明此过程,图 1 表示了小三角形 a 的扩展过程。a 的三边分别扩展为 b、c、d,b 又扩展为 e、f, c 扩展为 g、h,d 扩展为 j、i。在扩展的过程中如果遇到扩展边的扩展三角形已标记为 1,则停止该扩展边的扩展。如果三角形 c 不满足 $\mathrm{Max}(t_i)/\mathrm{Min}(t_i) < k$ 的扩展条件,三角形 c 只是将扩展标记修改为 1,该三角形不再扩展。

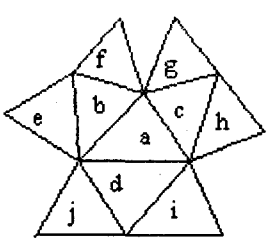


图 1　小三角形的扩展

步骤 2:以狭长三角形为基准进行扩展。

(1) 选取任意一标记为 0 的狭长三角形,将短边相连的顶点标记为 f,并标记该三角形为 1,将此三角形作为基准三角形,对其三边分别进行扩展。

(2) if (扩展三角形标记为 1) 不扩展;

(3) else

{将扩展三角形标记为 1;

if (扩展三角形是狭长三角形)

if (短边相连的两顶点有一顶点标记大于 0)

短边相连的两顶点标记为同一簇;

else if(短边相连的两顶点均为 0)

{$f = f + 1$;

短边相连的两顶点标记为 f;}

}

该三角形作为基准三角形,对其边分别进行扩展;

}

(4) 返回步骤(2) 直至再扩展边可以扩展。

(5) 返回步骤(1) 直至再无狭长三角形可以扩展。

图 2 表示了狭长三角形 A 的扩展过程。三角形 a 的两边分别扩展为 b 和 c,b 的一边扩展为 d,c 的一边扩展为 e,d 的两边分别扩展为 f 和 g,e 的一边扩展为 h,f 的一边扩展为 i,h 的一边扩展为 j,i 的一边扩展为 k, j 的两边分别扩展为 l 和 m。其中 b、e、m 为大三角形,其余为狭长三角形。图 2 中所示 1、2、3、4 的顶点聚为一簇, 5、6、7 顶点聚为一簇。

输出: 顶点标记值相同的为同一个簇, 标记为 0 的顶点为孤立点,总共输出 f 个簇。

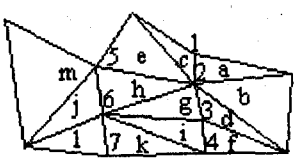


图 2　狭长三角形的扩展

3　仿真验证

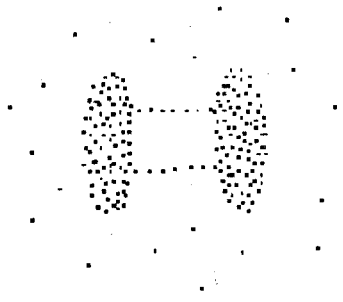
3.1　样本数据的准备

笔者在 AutoCAD 中设计了两个虚拟数据集如图 3(a)、图 4(a)所示,验证了文中提出的 CBDT 算法的有效性。

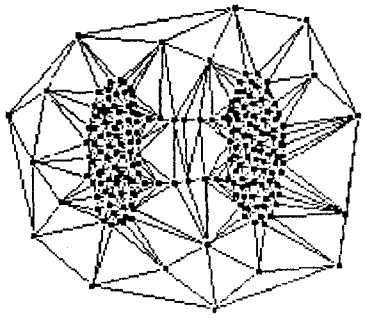
3.2　实验结果及讨论

文中用 CBDT 算法分别对图 3(a)和图 4(a)所示

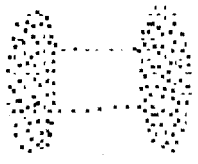
的样本数据集进行聚类。在对图 3(a)进行聚类时,先对其构网,得到图 3(b)所示的 Delauany 三角网,从图中可以清楚地看到小三角形、狭长三角形、大三角形的分布情况,小三角形往往位于簇的内部;狭长三角形、大三角形位于簇的边缘地带。显然由小三角形进行扩展,可以识别出如图 3(c)所示的两个椭圆图。在 CBDT 算法中,之所以要对狭长三角形单独进行扩展,是为了寻找有可能遗漏的单链簇,如图 3(b)中两个簇之间的桥。该数据集参数 k 设定为 1.2,得到图 3(c)所示的聚类结果(输出以不同的颜色来区分不同的簇,以下相同),其中有两个椭圆形簇和两个单链簇,其余的点为孤立点。同样,在对图 4(a)进行聚类时,先对其构网,得到图 4(b)所示的 Delauany 三角网,参数 k 设定为 1.3,聚类结果为图 4(c)所示的两个密度渐变的簇,其余的点为孤立点。从图 4(b)中可以看到三角网中三角形的分布情况,在这两个簇的内部三角形的大小在逐渐变化而三角形的形状基本一致,但是在这两个簇的边缘三角形的形状、大小与簇内部三角形的形状、大小具有显著不同,该算法能识别出密度渐变的簇。该算法采用扩展基准三角形的方法本身就扩大了



(a)原始数据集(210个数据)

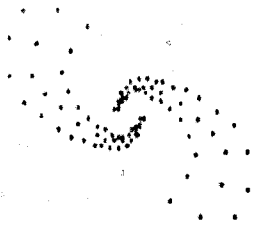


(b)生成Delaunay三角网

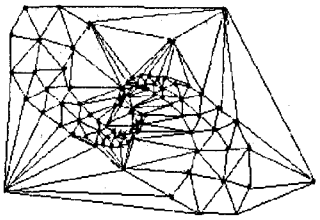


(c)聚类结果

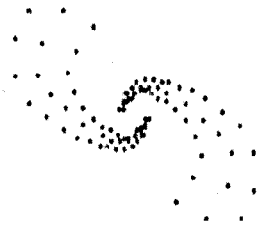
图 3 测试数据 1 的聚类示意图



(a)原始数据集(100个数据)



(b)生成Delaunay三角网



(c)聚类结果

图 4 测试数据 2 的聚类示意图

邻近点的范围,不用像 AUTOCLUST 那样考虑将邻近点的范围扩大到路径长度不大于 2 的点的范围,减少了计算量。

4 结束语

CBDT 算法是一种基于 Delaunay 三角网的聚类算法。本算法的核心是采用将 Delaunay 三角剖分所得三角形划分为小三角形、狭长三角形、大三角形,以此为聚类模型实现聚类的。该算法不但能实现 AUTOCLUST 算法所能识别的簇,而且能识别 AUTOCLUST 算法不能识别的密度渐变的簇,如图 4(a)所表示的数据集,AUTOCLUST 算法不能识别出如图 2(c)所示完整的簇,并且 CBDT 算法在时间复杂度上要优于 AUTOCLUST 算法。

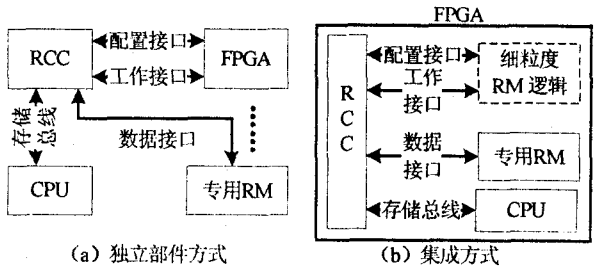
参考文献:

[1] 张 宏,温永宁,刘爱利. 地理信息系统算法基础[M]. 北京:科学出版社,2006.

[2] 彭仪普,刘文熙. Delaunay 三角网与 Voronoi 图在 GIS 中的应用研究[J]. 测绘工程,2002,3(11):39-41.

[3] Estivill-Castrov, LEE I J. AUTOCLUST: Automatic cluster-

(下转第 28 页)



(a) 独立部件方式 (b) 集成方式

图 3 CPU,RM 和 RCC 连接关系的实现结构

4 关键技术验证

RHS 关键技术的模拟验证包括以下内容:

(1) RCC 逻辑用 Verilog 语言进行设计,最小能仲裁 3 * 3 路 CrossBar 的同时访问,通过设定数目参数其能支持可变模块数的互连访问;RCC 的 testbench 中构造了两个 CPU、两个 RM 及一个全局 flash 的访问模型,通过模拟验证了 RCC 仲裁功能的正确性。

(2) 利用 Xilinx 提供的 FFT、CODIC、LSFR 等 IP 核完成这些算法在 FPGA 上的加速。

(3) 采用流水化结构设计了一种专用 RM 结构,n 段功能单元能组成深度流水的处理结构,在其上映射数据密集型的 MAC(乘累加)算法,形成高速的流水处理结构。

(4) 利用部分重构设计流程生成包括 LEON3 微处理器核逻辑、IP 核算法处理单元在内的配置流,能完成可变 IP 核算法处理单元的控制和处理,进行了模拟验证。

(5) LEON3 CPU 、其它 CPU 访问模型、RCC、专用 RM、IP 核算法处理单元的设计逻辑在由 Xilinx XC2VP50 大容量 FPGA、存储芯片等构造的实验平台上进行了实现验证;在采用 SPARC 微处理器[6](与 LEON3 具有相同的接口)、CPLD 和 XC2V500 FPGA 构造的实验平台上实现了 IP 核算法处理单元和在线重构控制逻辑,验证了实现的可行性。

上述模拟或实现验证了 RHS 中的几项关键技术:RCC 结构及其实现、基于 FPGA 的 IP 核应用、在线重构控制实现及部分重构流程、RISC 结构的嵌入式处理器设计及应用、专用 RM 结构设计等。在相同的频率下,实验同一算法(FFT)在 CPU 上软件实现的执行时间是使用 FPGA 硬件加速执行时间的 20 多倍,这说明在上述关键技术的支撑下,带有不同处理模块的 RHS 能为具有不同计算特点的应用提供高效的处理平台。

5 结束语

RHS 能融合多种不同的计算资源,使系统中的某些资源能够很大限度地满足某种应用的模式和处理要求。系统中 RM 硬件功能的在线重构特性能实现硬件的功能改变,RCC 能有效支持自身的配置和对其它模块的重构控制,可在一定的配置设置下完成各模块间的灵活互连,这种可变性使系统能适应更大范围的应用需求,向一体化和高性能的方向发展。下一步的工作将整合各种技术,实现完整的可重构异构原型系统,并进行实际应用的功能划分和算法映射。

参考文献:

[1] Miyamori T, Olukotun K. A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications[C]// Proc. of IEEE Sym. on FCCM98. Napa, CA:[s. n.],1998:2 - 11.

[2] Singh H, Lee M, Lu G, et al. MorphoSys: case study of a reconfigurable computing system targeting multimedia Applications[C]// Proc. Design Automation Conference. Los Angeles, California:[s. n.], 2000:573 - 578.

[3] Margerm S. Reconfigurable Computing in Real - World Applications[J]. FPGA and Structured ASIC Journal,2006(5): 1 - 8.

[4] Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode[EB/OL]. [2005 - 09 - 10]. http://www.xilinx.com.

[5] Castillo J, Huerta P, López V, et al. A Secure Self - Reconfiguring Architecture based on Open - Source Hardware[C]// Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs. Puebla City, Mexico:[s. n.], 2005:10 - 17.

[6] 32 - bit SPARC V8 Embedded Processor. SAILING S698, User's Manual[EB/OL]. [2005 - 10 - 23]. http://www.orbita.com.

(上接第 24 页)

ing via boundary extraction for mining massive point_data sets [C]// Proceeding of International conference on Geo_computation,2000. Greenwich: University of Greenwich,2000:23 - 25.

[4] 武晓波,王世新,肖春生. Delaunay 三角网的生成算法研究[J]. 测绘学报,1999,28(1):28 - 35.

[5] 吴燕来,朱 莉. Delauany 三角网生成算法的研究与实现[J]. 计算机与信息技术,2007(3):21 - 22.

[6] Tsai V J D. Delaunay Triangulations in TIN Creation: an Overview and a Linear - time Algorithm[J]. Int . J. of GIS. 1993,7(6):501 - 524.

[7] Larkin B J. An ANSI C program to determine in expected linear time the vertices of the convex hull of a set of planar points [J]. Computer & Geosciences,1997,17(3):431 - 443.