计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20170979
56(5):977-991, 2019

障碍空间中基于 Voronoi 图的不确定数据聚类算法

万 静 崔美玉 何云斌 李 松
(哈尔滨理工大学计算机科学与技术学院 哈尔滨 150080)
(wanjha@163.com)

Uncertain Data Clustering Algorithm Based on Voronoi Diagram in Obstacle Space

Wan Jing, Cui Meiyu, He Yunbin, and Li Song
(*College of Computer Science and Technology, Harbin University of Science and Technology, Harbin* 150080)

Abstract In order to solve the problem of the uncertain data clustering in obstacle space, the Voronoi diagram in computational geometry is introduced to divide the data space, and an uncertain data clustering algorithm based on Voronoi diagram in obstacle space is proposed. According to the properties of Voronoi diagram, four clustering rules are proposed. In order to consider the probability distribution between data, the KL distance is used as the similarity measure between data objects. Because obstacles can not always remain static in real life, and space obstacles often change dynamically. Then, according to whether the set of obstacles is changed, an uncertain data clustering algorithm in static obstacle environment and dynamic obstacle environment is proposed. Theoretical studies and experiments show that the uncertain refining clustering algorithm in the static obstacles environment(STAO_RVUBSCAN), the uncertain clustering algorithm of the dynamic increase of obstacles(DYNOC_VUBSCAN), the uncertain clustering algorithm of the dynamic reduction of obstacles(DYNOR_VUBSCAN) and the uncertain clustering algorithm of the dynamic movement of obstacles (DYNOM_VUBSCAN) have extremely high efficiency.

Key words static obstacles; dynamic obstacles; Kullback-Leibler divergence; uncertain data; Voronoi diagram

摘 要 为了有效解决障碍空间中的不确定数据聚类的问题,引入计算几何中的 Voronoi 图对数据空间进行划分,提出障碍空间中基于 Voronoi 图的不确定数据聚类算法.根据 Voronoi 图的性质,提出 4 项聚类规则.利用 KL 距离进行相似性度量.根据障碍集合是否发生变化,提出了静态障碍环境下和动态障碍环境下的不确定数据聚类算法.理论研究和实验表明:静态障碍物环境中的不确定精炼聚类算法(简称 STAO_RVUBSCAN 算法)、障碍物动态增加情况下的不确定聚类算法(简称 DYNOC_VUBSCAN 算法)、障碍物动态减少情况下的不确定聚类算法(简称 DYNOR_VUBSCAN 算法)和障碍物动态移动情况下的不确定数据聚类算法(简称 DYNOM_VUBSCAN 算法)都具有较高的效率.

关键词 静态障碍;动态障碍;KL 距离;不确定数据;Voronoi 图

中图法分类号 TP311.13

收稿日期:2017-12-29;**修回日期**:2018-08-14

基金项目:国家自然科学基金项目(61872105);黑龙江省教育厅科技研究项目(1253lz004);黑龙江省留学归国人员科学基金(LC2018030)
This work was supported by the National Natural Science Foundation of China (61872105), the Science and Technology Research Project of Heilongjiang Provincial Education Department (1253lz004), and the Scientific Research Foundation for Returned Scholars Abroad of Heilongjiang Province of China (LC2018030).

近年来,随着人们对信息技术的不断了解,数据的聚类[1]问题在许多应用中不断出现,如基于传感器网络、基于位置的服务等领域.由于在数据采集过程中,受周围环境的影响,或者是采集仪器精度的限制,因此数据具有不确定性.不确定数据[2-3]的分析与挖掘技术已成为最近几年的研究热点,不确定数据的聚类问题,就是如何将不确定数据集合划分成若干个簇,使最终的结果簇内元组间相似性较大,簇与簇之间元组相似性较小.

现有的聚类算法大多解决的是确定数据的聚类问题,对于不确定数据的聚类算法研究中,主要的研究方法是对传统的面向确定数据的聚类方法进行扩展,即在传统的聚类算法[4-5]如基于划分的聚类、基于密度的聚类、基于网格的聚类以及基于层次和模型的聚类方法等基础上对不确定数据进行建模,使用合适的相似性度量方法,进而提出不确定数据聚类算法.

国内外学者研究了基于划分的不确定聚类算法UK-means[6-7],算法是用最小边界矩形(minimum bounding rectangle, MBR)来表示数据的不确定性,MBR描述了数据点可能出现的区域位置,但UK-means算法只能发现球状簇,并且易受参数 k 的影响.为了解决大多数聚类划分算法发现簇形状单一的缺陷,Hao等人提出采用基于密度[8]的聚类算法.Kriegel等人根据经典DBSCAN(density-based spatial clustering of applications with noise)算法提出了FDBSCAN(a fast DBSCAN algorithm)[9]算法,Han等人根据OPTICS(ordering points to identify the clustering structure)算法提出了FOPTICS[10]算法,FDBSCAN算法采用距离分布函数作为数据间相似性度量的标准,而FOPTICS算法在处理大型数据集方面更有效.文献[11]提出了FDBSCAN算法的改进算法,多核M-FDBSCAN(multicore-FDBSCAN)方法是通过半聚类发生分裂和合并来构造最终的簇.基于密度的聚类算法大多需要依赖Eps和Minpts这2个参数,陆亿红等人[12]提出了OLUC(optimal k-nearest neighbors and local density-based clustering algorithm for uncertain data)算法,采用动态自适应的最近 k 近邻结构,计算局部相对密度,降低了参数的选择和密度等级的敏感性.由于网格结构能够提高聚类速度,因此韩利钊等人[13]提出基于区域划分的DBSCAN多密度聚类算法,利用网格相对密度差把数据空间划分成密度不同的区域,使用网格与密度相结合的方法,提高了聚类效率.Kao等人[14]利用Voronoi图和R树的性质,提出剪枝策略来减少期望距离的计算.以上算法仅考虑了确定元组间的距离,没有考虑元组间的不确定因素和数据间的概率分布,因此采用KL距离作为数据对象间的相似性度量标准.

选址问题在物流、生产生活方面有着广泛的应用,如工厂、垃圾处理中心、物流中心的选址等.随着电商的发展,网上购物的人越来越多,物流发展迅速,物流中心选址的好坏直接影响到服务质量、服务的效率和成本,从而影响到利润和市场竞争力,合适的选址会给人民生活带来便利.由于现实生活中存在一些地理条件的限制(如江河、湖泊、建筑、车辆等),使得聚类分析不可能都在理想的欧氏空间中进行,因此在进行相似性度量时,需要考虑障碍[15]的存在.然而现实生活中的障碍物不可能一直静止不变,空间障碍物往往会发生动态改变,障碍物的改变可能使得原有聚类结果发生改变,因此研究静态障碍空间和动态障碍空间中的不确定数据聚类问题有较大的意义.

近几年来,带有障碍的空间聚类问题受到越来越多国内外研究者的关注,曹科研等人[16]提出了一种障碍空间中的不确定数据聚类算法,分别基于R树、最近距离区域和Voronoi图的性质,设计了高效的剪枝策略,并较高程度地实现了不确定数据的聚类算法.文献[17]提出的带障碍约束空间聚类算法比遗传K-Medoids[18]具有较高的收敛速度.在聚类分析时,考虑障碍的约束,实用价值更高.

在障碍空间中,现有的数据聚类研究成果大多都是针对确定数据的聚类问题.为了解决障碍空间中的不确定数据聚类问题,利用Voronoi图对数据空间进行划分.根据障碍集合是否发生变化,提出了静态障碍环境下的不确定数据聚类算法和动态障碍环境下的不确定数据聚类算法,其中动态障碍物分3种情况处理,分别是障碍物动态增加时的不确定聚类算法DYNOC_VUBSCAN、障碍物动态减少时的不确定聚类算法DYNOR_VUBSCAN和障碍物动态移动情况下的不确定数据聚类算法DYNOM_VUBSCAN.理论研究和实验表明:所提算法具有较高的准确性.

1 定义与性质

定义1. 不确定数据点集[19].不确定数据集为

$X=\{X_1,X_2,\cdots,X_n\}$,X_i 表示一个不确定数据点.$X_i=\{(x_1,p_1),(x_2,p_2),\cdots,(x_d,p_d)\}$,每个 x_i 表示 X_i 的属性值,d 为维度数.为每一条数据的每一维属性值生成[0,1]区间内的概率值 p_i,以元组的形式表示,元组由数据值和概率组成.

定义 2. 障碍集.一个障碍用一个多边形表示,记为 O_i,其中障碍的顶点集用 $V=\{v_1,v_2,\cdots,v_k\}$ 表示,障碍的边集用 $E=\{e_1,e_2,\cdots,e_k\}$ 表示,记为 $O_i(V,E)$.障碍集用 $O=\{O_1,O_2,\cdots,O_m\}$ 表示.

定义 3. Voronoi 图[20].给定一组生成点 $X=\{X_1,X_2,\cdots,X_n\}\subset\mathbb{R}^2$,其中 $2<n<\infty$,且当 $i\neq j$ 时,$X_i\neq X_j$.其中 $i,j\in\{1,2,\cdots,n\}$.由 X_i 所决定的区域称为 Voronoi 单元 VX,Voronoi 图构成为 $VD(X)=\{VX(X_1),VX(X_2),\cdots,VX(X_n)\}$,其中 $VX(X_i)$ 表示的是 X_i 所在的 Voronoi 单元.

定义 4. 邻接多边形[21].共享相同边的 Voronoi 多边形称为邻接多边形,它们的生成点被称为邻接生成点.Voronoi 单元中存在几条边,则就会有几个邻接多边形.如图 1 所示,X_{11} 的 Voronoi 单元中有 5 条边,对应的邻接生成点为 $X_1,X_2,X_{10},X_{12},X_{13}$.$X_{11}$ 的邻接多边形为以上 5 个邻接生成点所在的 Voronoi 单元.

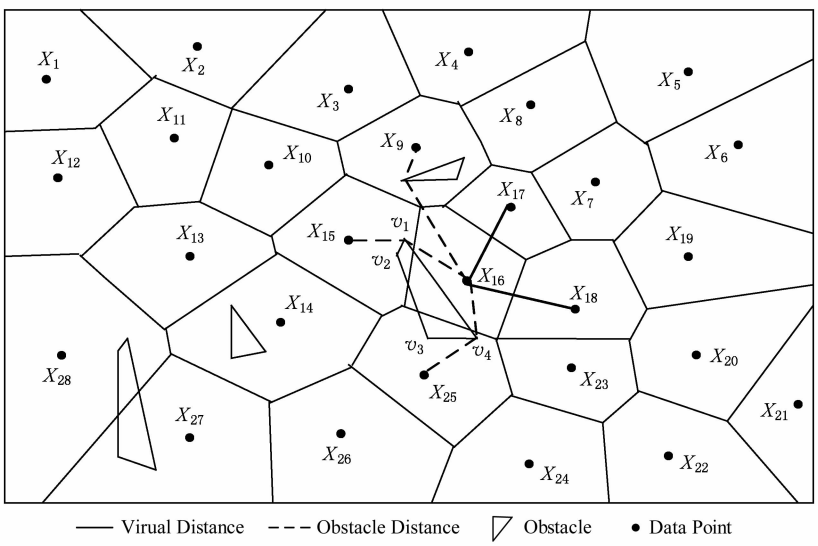


Fig. 1　An example of obstacle distance

图 1　障碍距离的示例

根据 Voronoi 图的结构和定义可以得出 2 个基本性质.

性质 1. 任意 2 个多边形不存在公共区域[18].Voronoi 图将不确定数据对象集合中的数据按照其最近邻特性将空间进行划分,生成互不重叠的区域.

性质 2. 临近特性[20].生成点 X_i 与邻接多边形中的邻接生成点距离最近.

定义 5. Voronoi 图的 k 级邻接生成点[20].给定一组生成点 $X=\{X_1,X_2,\cdots,X_n\}\subset\mathbb{R}^2$.$X_i$ 的 k 级邻接生成点定义如下:

1) 一级邻接生成.$AG_1(X_i)=\{X_j\mid VX(X_i)$ 和 $VX(X_j)$ 有公共边$\}$.

2) $k(k\geqslant 2)$ 级邻接生成点.$AG_k(X_i)=\{X_j\mid VX(X)$ 和 $VX(X_j)$ 有公共边,$X\in AG_{k-1}(X_i)\}$.

定义 6. 覆盖圆半径 R_c.半径 R_c 的选取,需要考虑的是整个不确定数据空间,半径 R_c 过小,覆盖圆内包含的不确定数据点的个数就越少,相应的点的密度就越小,使聚类迭代次数增加.半径 R_c 过大,容易将孤立点包含在覆盖圆中,因此 R_c 的确定是重要的.

定义 7. KL 距离[22].KL 距离也叫作相对熵,KL 距离衡量的是相同事件空间里的 2 个概率分布的差异情况,KL 距离函数表达式为

$$D(f\parallel g)=\int_D f(x)\ln\frac{f(x)}{g(x)}\mathrm{d}x. \tag{1}$$

在定义域 D 中,将不确定对象作为一个服从一定概率分布的随机变量.在连续的定义域中,不确定数据用概率密度函数(probability density function,PDF)[23]表示,概率密度函数用高斯核密度估计:

$D(f\parallel g)=\int_D f(x)\ln\frac{f(x)}{g(x)}\mathrm{d}x$,根据大数定律 $D(X\parallel Y)=\lim\limits_{s\to\infty}\frac{1}{s}\sum\limits_{i=1}^{s}\ln\frac{X(x_i)}{Y(y_i)}$,其中 s 表示数据样本数量,根据文献[22],KL 距离表达式为

$$D(X \| Y) = \frac{1}{s}\sum_{i=1}^{s} \ln \frac{X(x_i)}{Y(y_i)}. \quad (2)$$

在离散的情况下,不确定数据用概率质量函数(probability mass function, pmf)表示. 根据文献[22],KL 距离表达式为

$$D(f \| g) = \sum_{x \in D} f(x) \ln \frac{f(x)}{g(x)}. \quad (3)$$

KL 距离有 2 个基本性质:

性质 3. 非对称性[22]. KL 距离从直观上是个度量或距离函数,但它并不是一个真正的度量或者距离,因为它不具有对称性,一般从分布 P 到分布 Q 的距离通常不一定等于分布 Q 到分布 P 的距离. 即 $D(P \| Q) \neq D(Q \| P)$.

性质 4. 非负性[22]. KL 距离值非负,即 $D(P \| Q) \geqslant 0$. 当且仅当 2 个分布相同时,$D(P \| Q) = 0$.

定义 8. 聚类半径 R[22]. 聚簇网格内所有不确定数据点与代表点 C_i 的 KL 距离的均值,即为不确定聚类半径,其中聚簇网格内所有不确定数据点集合为 $X = \{X_1, X_2, \cdots, X_r\}$. 聚类半径 R 表达式为

$$R = \frac{1}{r}\sum_{i=1}^{r} f(X_i) \ln \frac{f(X_i)}{g(X)}, \quad (4)$$

其中,r 表示聚簇内不确定数据点的总个数.

定义 9. 不确定数据点之间的可视性[16]. 给定障碍集 O 和数据点 X_i 和 X_j,不确定数据点 X_i 和 X_j 是可视的当且仅当 X_i 和 X_j 之间的连线不会穿过障碍集 O 中任何障碍的内部,也不会与任何边集相交. 数据点 X_i 和 X_j 是不可视时,X_i 和 X_j 之间的连线会穿过障碍集 O 的内部或有边集相交. 将存在障碍的 Voronoi 单元标记为阻塞 Voronoi 单元 VX'.

定义 10. 不确定数据点之间的障碍距离[16]. 给定障碍集 O 和数据点 X_i 和 X_j,当 X_i 与 X_j 之间互为可视时,X_i 与 X_j 的障碍距离为 KL 距离,表示为 $D(X_i \| X_j)$. 当 X_i 与 X_j 之间不可视时,X_i 与 X_j 的障碍距离表示为 $dist(X_i, X_j)$,数据点之间的障碍距离是绕过障碍的路径最小距离. X_i 与 X_j 之间的障碍距离:

$$dist(X_i, X_j) = \min\{D(X_i \| v_i) + D(v_i \| X_j), D(X_i \| v_j) + D(v_j \| X_j)\}. \quad (5)$$

图 1 给出了不确定数据点之间的障碍距离,其中图 1 中的虚线表示障碍距离,粗实线表示无障碍时的直接距离.

如图 1 所示,根据式(5)障碍距离的计算,不确定数据点 X_{15} 与 X_{16} 之间的障碍距离为

$$dist(X_{15}, X_{16}) = \min\{D(X_{15} \| v_1) + D(v_1 \| X_{16}), D(X_{15} \| v_4) + D(v_4 \| X_{16})\}.$$

定义 11. 不确定数据点的密度[24]. 空间中的任意一个不确定数据点 X_i,$1 \leqslant i \leqslant n$. 以不确定数据点 X_i 为中心,R_c 为半径的覆盖圆区域内的不确定数据生成点个数,称为不确定数据点的密度,记作 $\rho(X_i)$.

定义 12. 核心对象. 根据给定阈值 ε,最大覆盖圆半径 R_c,若不确定数据对象 X_i 的最大覆盖圆中包含的不确定数据对象个数 $\rho \geqslant \varepsilon$ 时,则称 X_i 为核心对象.

2 障碍空间中的不确定数据聚类

随着数据规模越来越大,使用基于密度的聚类算法能发现任意形状的聚簇,并且对孤立点数据不敏感. 使用 Voronoi 图将数据空间划分成若干个空间单元,根据 Voronoi 图的邻接特性和所提出的规则对不确定数据进行聚类,使最终的聚类结果有较高的执行效率和准确性.

2.1 静态障碍空间下不确定数据聚类算法

静态障碍环境指的是空间障碍物集合不会发生改变,包括障碍的位置、障碍的点集和边集都不发生变化. 对此时的空间不确定数据实现聚类算法,并提出了 STAO_VUBSCAN 算法.

STAO_VUBSCAN 算法的主要思想:在一定距离内,可根据 Voronoi 单元与其邻接的 Voronoi 的级数大小,判定局部密度大小. 找出核心对象,以不确定数据点 X_i 为中心,以 R_c 为半径形成一个覆盖圆,计算覆盖圆内所含的不确定数据点的个数 ρ. 若 $\rho > \varepsilon$,则说明不确定数据点的数量较多,表示 X_i 所在覆盖圆内的局部密度越大. 根据局部密度的中心进而判断邻接 Voronoi 单元的聚类情况,最终实现静态障碍环境下的不确定数据聚类.

定理 1. 若不确定数据点 X_i 在 Voronoi 图中的所有一级邻接生成点都被当作孤立点,则此时的不确定数据点 X_i 也同样的视为孤立点.

证明. 设 X_j 为 X_i 的所有一级邻接生成点,$1 \leqslant j \leqslant 6$[20]. 若 X_j 为孤立点,说明所有 X_j 所在覆盖圆的数据密度都较小,不能视为核心对象. 则以 X_i 为中心,R_c 为半径的覆盖圆内的数据密度也较小. 由于 X_j 为孤立点被剪枝,此时所有 X_j 的中心 X_i 也必定被视为孤立点处理.

图 2 中,不确定数据点 X_1 为圆心的覆盖圆中,覆盖圆内的 X_1 最大有 2 级生成点,所以在进行相

似性度量时,只考虑 2 级生成点以内的不确定数据和障碍情况,2 级以外的邻接生成点和障碍物可先不作考虑,这会简化 KL 距离的计算以及障碍的判断.判断每个核心对象所在的覆盖圆内数据点聚类情况,使得聚类过程更加简便有效.

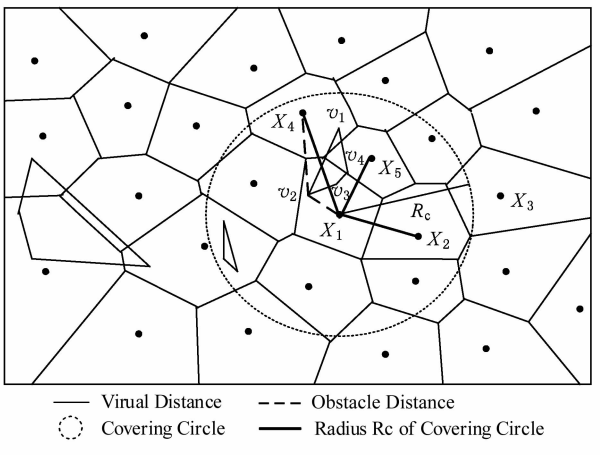


Fig. 2　An example based on Theorem 1

图 2　基于定理 1 的示例

算法 STAO_VUBSCAN 的主要步骤:首先选择任一未划分的不确定数据对象,根据不确定数据点密度大小,先对密度大的生成点设置为核心对象.将核心对象根据密度从大到小降序排序,存放在 Hash 表 L 中,$L=\{X_1,X_2,\cdots,X_t\}$,$1\leqslant i\leqslant t$.根据核心对象在 Hash 表中的顺序来依次聚类,将判断过的不确定数据从 Hash 表中删除,直到 Hash 表为空为止.使用 KL 距离作为相似性度量标准,寻找所有可能被聚到此类的不确定数据点,并将这些数据点标记为一类,并根据定理 1 进行部分孤立点的处理.重复以上步骤,直到所有不确定数据对象划分完毕.

基于以上讨论,进一步提出静态障碍环境下的不确定数据聚类算法 STAO_VUBSCAN,如算法 1 所示.

算法 1. *STAO_VUBSCAN*(X,O,ε,R_c).

输入:不确定数据点集 X、障碍集合 O、覆盖圆半径 R_c,ε,R;

输出:障碍空间下的不确定数据聚类结果 W.

① for $i=1$ to n do
② 　*Calculate_den*(X_i);
③ end for
④ if $\rho\geqslant\varepsilon$
⑤ 　$L\leftarrow$*Quick_Sort*(X_i); /* 降序排序 */
⑥ end if
⑦ for ($i=1$ to t)且 $L\neq\varnothing$ do
⑧ 　if X_i 与 X_j 可视
⑨ 　　*Calculate_KL*(); /* 式(2)或式(3) */
⑩ 　　if $D(X_i\parallel X_j)\leqslant R$
⑪ 　　$W\leftarrow X_i,X_j$;
⑫ 　　end if
⑬ 　　$L\leftarrow$*delete*(X_i,X_j);
⑭ 　end if
⑮ 　else if X_i 与 X_j 不可视
⑯ 　　*Calculate_OKL*(); /* 式(5) */
⑰ 　　if $dist(X_i,X_j)\leqslant R$
⑱ 　　$W\leftarrow X_i,X_j$;
⑲ 　　end if
⑳ 　　$L\leftarrow$*delete*(X_i,X_j);
㉑ 　end if
㉒ end for
㉓ return W.

首先算法 1 在创建 Voronoi 时,构建所需的时间复杂度为 $O(n\lg n)$.算法执行第 1 个 for 循环,遍历不确定数据点所需的时间复杂度是 $O(n)$.由于 n 的个数是有限的,所以此步骤是可终止的.在对核心对象进行快速排序时所需的时间复杂度为($m\lg m$),其中 m 表示的是核心对象的个数,$m\ll n$.算法执行第 2 个 for 循环所需的时间复杂度为 $O(t)$,因为覆盖圆内核心对象 t 的数量有限,所以此步骤也是可终止的.由以上算法的核心步骤的时间复杂度分析可知,算法总的时间复杂度为 $O(n\lg n)$.算法 1 的计算代价主要受空间数据集的规模、计算无障碍情况下距离度量的计算量和存在障碍时的障碍距离计算量的影响.

2.2　静态障碍空间中不确定数据精炼算法

算法 1 已经解决了静态障碍空间中的不确定数据聚类问题.但是算法 1 只考虑覆盖圆内不确定数据点密度,根据覆盖圆内不确定数据点密度大小,定义核心对象,而没有考虑到核心对象所在覆盖圆内的障碍的多少.针对覆盖圆内静态障碍物的分布情况,进一步提出 STAO_RVUBSCAN 算法,算法利用 Voronoi 图的邻接特性和提出的规则,设计了高效的聚类方法.

规则 1. 如果最大覆盖圆内不确定数据点密度较大,并且满足 $\rho\geqslant\varepsilon$,但存在的障碍较多时,则对应的不可视集合 S_2 中的不确定数据点就会较多,在进行相似性度量时,计算的是障碍距离.若 $dist(X_i,X_j)>R$,则 X_i 所在覆盖圆内的不确定数据点 X_j 不能加入到 X_i 所在的簇中.

只考虑核心对象的密度是不够的,还需要考虑核心对象最大覆盖圆内静态障碍的密度,障碍密度越大,说明不可视集合 S_2 内的不确定数据个数就越多,判断集合 S_1 中可视数据密度是否满足 $\rho \geqslant \varepsilon$. 如果满足,执行过程跟算法 1 一致. 否则,说明了核心对象周围的障碍较多,不可视集合 S_2 内的不确定个数较多. 在进行相似性度量时计算的是障碍距离,因为每个障碍距离都大于没有障碍时的可视距离,因此需要考虑核心对象是否被当作孤立点处理.

如图 3 所示,以不确定数据点 X_1 为圆心,R_c 为半径的最大覆盖圆内不确定数据集合 S 为$\{X_1, X_2, X_4, X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, X_{12}, X_{13}\}$,此时不确定数据点密度较大. 覆盖圆内存在 O_1, O_2, O_3, O_4, O_5 共 5 个障碍,障碍将不确定数据分为 2 个集合,分别为可视集合 S_1 和不可视集合 S_2. 根据分析得出,可视集合 $S_1 = \{X_7, X_8, X_{10}, X_{13}\}$,不可视集合 $S_2 = \{X_2, X_4, X_5, X_6, X_9, X_{11}, X_{12}\}$.

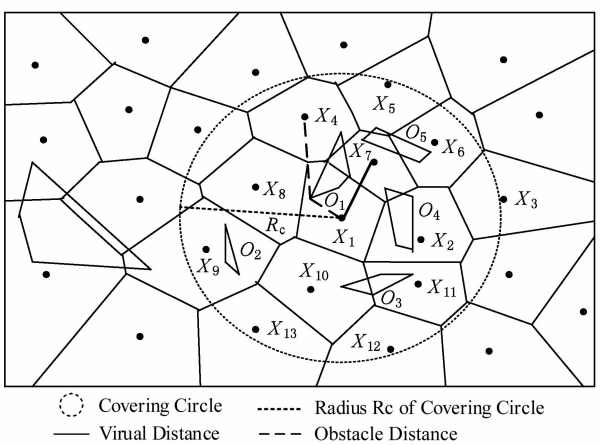


Fig. 3 An example based on regulation 1

图 3 基于规则 1 的示例

若 S_1 中不确定数据密度远小于 ε,或者 S_2 中不确定数据密度$\geqslant \varepsilon$,说明覆盖圆内不确定数据点 X_i 的周围存在较多的障碍,此时 X_i 不能作为核心对象处理. 连续域环境下,可视集合内的距离使用式(2). 离散域环境下,可视集合内的距离公式使用式(3),不可视集合内的障碍距离用式(5)度量. 规则 1 的提出,减少了障碍距离的判断,使得算法的效率较大程度地提高.

首先判断核心对象所在覆盖圆内障碍的情况,如果核心对象内的静态障碍不影响聚类结果,则聚类结果跟算法 1 相同. 如果核心对象所在覆盖圆内障碍的密度较大,只需判断不可视集合 S_2 中数据的聚类情况. 最后根据相似性度量标准,将覆盖圆内的不确定数据划分到合适的聚簇中. 基于以上分析,进一步提出了静态障碍环境下的不确定数据聚类精炼算法 STAO_RVUBSCAN,如算法 2 所示.

算法 2. $STAO\_RVUBSCAN(X, O, \varepsilon, R_c, L)$.

输入:$X, O, R_c, \varepsilon, R, L$;

输出:聚类结果 W.

① for (i=1 to t)且 $L \neq \varnothing$ do /* 规则 1 */

② $S \leftarrow Count\_\rho(X_i)$;

③ if $Circle\_R(X_i)\ \exists O_i$ /* 覆盖圆中存在障碍时 */

④ $S_1 \leftarrow Visiable\_X(X_i, X_j)$; /* $1 \leqslant j \leqslant S$ */

⑤ $S_2 \leftarrow NVisiable\_X(X_i, X_j)$;

⑥ end if

⑦ if $Count\_S_1 \geqslant \varepsilon$

⑧ $Calculate\_KL()$; /* 式(2)或式(3) */

⑨ if $D(X_i \parallel X_j) \leqslant R$

⑩ $W \leftarrow X_i, X_j$;

⑪ end if

⑫ $Calculate\_OKL()$; /* 式(5) */

⑬ if $dist(X_i, X_j) \leqslant R$

⑭ $W \leftarrow X_i\ X_j$;

⑮ end if

⑯ $L \leftarrow delete(X_i, X_j)$;

⑰ end if

⑱ if $Count\_S_1 < \varepsilon$ 或 $Count\_S_2 \geqslant \varepsilon$

⑲ $L \leftarrow delete(X_i, X_j)$;

⑳ end if

㉑ end for

㉒ return W.

算法 2 在创建 Voronoi 时,构建所需的时间复杂度为 $O(n \lg n)$. 算法执行只有一个 for 循环,所需的时间复杂度为 $O(t)$,因为覆盖圆内核心对象 t 的数量有限,所以此步骤也是可终止的. 由以上时间复杂度分析可知,算法 2 总的时间复杂度为 $O(n \lg n)$.

算法 2 的聚类效率优于算法 1,由于规则 1 的提出,过滤了大量的不确定数据点的判断,减少了大量的计算,也使得最终的聚类结果更准确.

2.3 动态障碍增加情况下不确定数据聚类算法

算法 1 和算法 2 研究的是静态障碍物环境下的不确定数据聚类算法,但是现实生活中的空间障碍物往往会发生动态改变,如空间障碍物的位置发生改变,障碍物的点集和边集发生改变,因此提出动态障碍情况下的不确定数据聚类算法. 根据障碍物动态变化的不同情况,分别提出障碍物动态增加、障碍

物动态减少和障碍物动态移动 3 种情况的不确定数据的聚类算法.

算法 3 研究的是障碍物动态增加时的聚类方法,由于空间障碍物的数量可能发生变化,空间障碍物的增加会对原始的聚类结果产生影响. 如图 4 所示,当障碍物增加时,障碍集则为 O_{new},其中 O_i' 为新增加的障碍物. 在障碍物增加之前,可视点集合为 S_1,不可视集合为 S_2. 在障碍物增加之后,设定可视点集合为 S_1',不可视点集合为 S_2'.

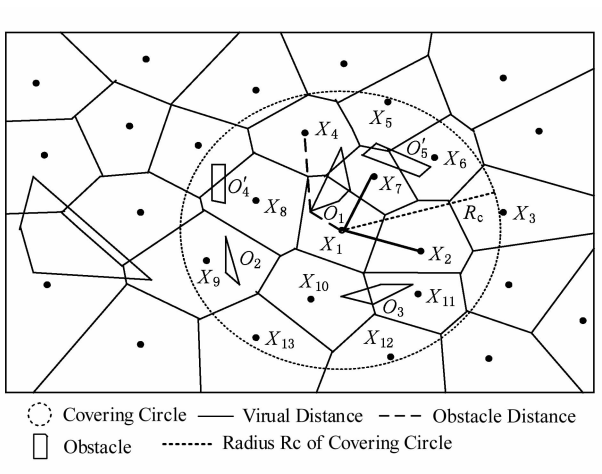


Fig. 4　An example of dynamic increase in obstacles

图 4　障碍动态增加的示例

根据新增加的障碍物的位置分析,障碍物增加对聚类可能没有影响,也可能有影响.

规则 2. 首先判断新增加障碍的位置,然后计算障碍所在 Voronoi 图中不确定数据点的最大覆盖圆范围. 如果新增加的障碍对聚类结果无影响,覆盖圆范围内不影响聚类的结果,则聚类结果与算法 2 的结果相同. 若新增障碍对不确定结果有影响,覆盖圆范围内的新增障碍影响了 Voronoi 图中不确定数据点的聚类,则对此覆盖圆内的不确定数据点重新聚类. 重新聚类可能引起其他类的聚类,因此根据 Voronoi 图的邻接特性,需将可视性改变的不确定数据点加入到离其最近的邻接核心对象所在的聚簇中.

如图 4 所示,覆盖圆内原始障碍集合为 $\{O_1, O_2, O_3\}$. 不可视集合 $S_2=\{X_4, X_9, X_{11}, X_{12}\}$,可视集合 $S_1=\{X_7, X_8, X_{10}, X_{13}, X_2, X_5, X_6\}$. 增加障碍 O_4' 和 O_5' 之后,覆盖圆内障碍集为 $\{O_1, O_2, O_3, O_4', O_5'\}$,此时的可视集合 $S_1'=\{X_7, X_8, X_{10}, X_{13}, X_2\}$,不可视点集合 $S_2'=\{X_4, X_5, X_6, X_9, X_{11}, X_{12}\}$. 障碍 O'_4 的加入,并没有使得不确定数据点 X_1 和 X_8 之间不可视,O_4' 的加入对覆盖圆内的不确定数据并没有影响,X_8 依然存在于可视集合中. 因此对新增加的障碍进行判断是有必要的.

由以上分析可知,障碍物的增加对不确定数据聚类的影响是局部的,分情况判定处理会避免较多不必要的距离计算,因此提出规则 2 对动态增加的障碍进行具体分析,进一步提出了障碍物动态增加时的不确定数据聚类算法 DYNOC_VUBSCAN,如算法 3 所示.

算法 3. *DYNOC_VUBSCAN*(X, O, ε, R_c).

输入:$X, \varepsilon, R, R_c, O$,新增障碍 O_i';

输出:聚类结果 W.

① $O_{new} \leftarrow O+O_i'$;

② for $i=1$ to n do

③　*Obtain* S_1', S_2';

④　*Calculate_R*(X_i);

⑤　*Judge_O*(O_i'); /* 规则 2 */

⑥　if $Count\_S_1' \geqslant \varepsilon$ 且 $Count\_S_2' < \varepsilon$

⑦　　$W \leftarrow STAO\_RVUBSCAN()$; /* 调用算法 2 */

⑧　else if $Count\_S_1' < \varepsilon$ 且 $Count\_S_2' \geqslant \varepsilon$

⑨　　*Calculate_OKL*(S_2');

⑩　　if $dist(X_i, X_j) \leqslant R$

⑪　　$W \leftarrow X_i, X_j$;

⑫　　end if

⑬　　$L \leftarrow delete(X_i, X_j)$;

⑭　end if

⑮ end for

⑯ return W.

首先算法 3 在创建 Voronoi 时,构建所需的时间复杂度为 $O(n\lg n)$. 算法 3 只有一个 for 循环,遍历不确定数据点所需的时间复杂度是 $O(n)$. n 代表的是不确定数据的总数量,并且数量是有限的,所以算法是可终止的. 当算法 3 满足步骤⑥所需的条件时,调用算法 2. 算法 2 的时间复杂度已经分析得出是 $O(n\lg n)$. 由以上分析可知,算法 3 总的时间复杂度为 $O(n\lg n)$.

2.4　动态障碍减少情况下不确定数据聚类算法

数据空间的障碍物可根据实际情况动态变化,当障碍物减少时也很有可能对原来的聚类结果产生影响.

规则 3. 当减少障碍物时,没有改变最后的聚类结果,此情况的聚类结果跟静态障碍环境下的不确定聚类算法结果相同,使用算法 2 完成聚类即可. 当减少的障碍物能够引发最终的聚类结果改变,则对减少的障碍所在的 Voronoi 单元的最大覆盖圆内的

不确定数据重新进行相似性度量计算. 重新聚类可能引起其他类的聚类,因此根据 Voronoi 图的邻接特性,需将不可视变成可视时集合中的不确定数据点加入到离其最近的邻接核心对象所在的聚簇中.

如图 5 所示,图 5 中用虚线表示的障碍为动态减少的障碍. 原始障碍集合为 $\{O_1,O_2,O_3\}$,此时覆盖圆内的可视点集合为 $S_1=\{X_7,X_8,X_{10},X_{13},X_2,X_5,X_6\}$,不可视集合 $S_2=\{X_4,X_9,X_{11},X_{12}\}$. O_2 和 O_3 为减少的障碍物. 此时的障碍集合中只有 O_1,减少障碍会引发不确定数据点之间数据变为可视状态,可视点集合的不确定数据会增多,不可视集合中的不确定数据会减少. 覆盖圆内的可视点集合为 $S_1'=\{X_7,X_8,X_{10},X_{13},X_2,X_5,X_6,X_9,X_{11},X_{12}\}$,不可视点集合为 $S_2'=\{X_4\}$. 障碍的减少,使得 X_9,X_{11},X_{12} 可能原本是其他簇中的不确定数据划分到 X_1 所在的簇中. 因此,空间障碍物的减少也会影响聚类结果的准确性.

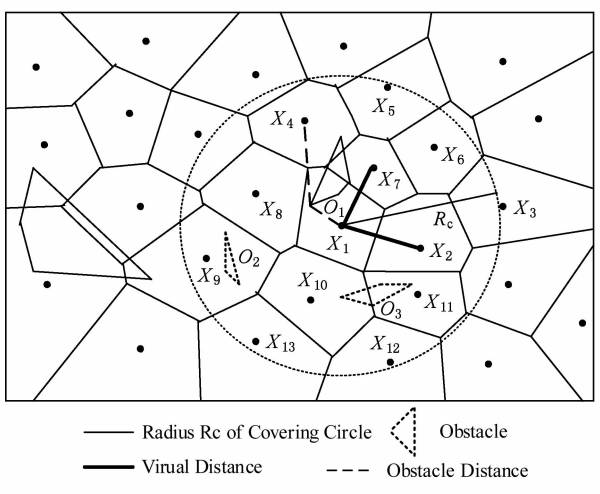


Fig. 5 An example of dynamic reduction in obstacles

图 5 障碍动态减少的示例

根据减少的障碍物的具体位置,减少的障碍可能对聚类影响,也可能对聚类结果没有影响,因此需要分析讨论. 算法的主要思想:首先得到新的障碍集合 O_{new},判断减少的障碍物的位置. 然后根据规则 3 分析减少的障碍对聚类是否有影响,最后得到障碍动态减少情况下的不确定数据聚类结果.

基于以上讨论,具体地给出动态障碍减少情况下的不确定数据聚类算法 DYNOR_VUBSCAN,如算法 4 所示.

算法 4. $DYNOR\_VUBSCAN(X,O,\varepsilon,R_c)$.

输入:X,O,ε,R_c,减少的障碍 O_i';

输出:聚类结果 W.

① $O_{new}\leftarrow O-O_i'$;

② for $i=1$ to n do

③ $Obtain\ S_1,S_2,S_1',S_2'$;

④ $Calculate\_R(X_i)$;

⑤ $Judge\_O(O_i')$; /* 规则 3 */

⑥ $S'\leftarrow S_2-S_2'$;

⑦ $q\leftarrow Count\_S'()$;

⑧ if $S'=\varnothing$ 或 $q=0$

⑨ $W\leftarrow STAO\_RVUBSCAN()$; /* 调用算法 2 */

⑩ else if $S'\neq\varnothing$

⑪ for $i=1$ to q do

⑫ $Calculate\_KL(S')$;

⑬ if $D(X_i\parallel X_j)\leqslant R$

⑭ $W\leftarrow X_i,X_j$;

⑮ end if

⑯ end for

⑰ end if

⑱ end for

⑲ return W.

首先算法 4 在创建 Voronoi 时,构建所需的时间复杂度为 $O(n\lg n)$. 算法执行最外层的 for 循环,遍历不确定数据点所需的时间复杂度是 $O(qn)$. n 代表的是不确定数据的总数量,并且数量是有限的,q 是障碍减少时,由不可视变成可视的数据点个数,数量也是有限的,所以算法是可终止的. 当算法 4 满足步骤⑧所需的条件时,调用算法 2. 算法 2 的时间复杂度已经分析得出是 $O(n\lg n)$. 由以上分析可知,算法 4 总的时间复杂度为 $O(n\lg n)$.

由于算法根据障碍将数据划分成可视集合与不可视集合,只需重新判断障碍减少时,由不可视变成可视的不确定数据点聚类情况. 此方法大大减少了计算量,使得局部的进行相似性度量比重新划分聚类更有效.

2.5 障碍物动态移动情况下的不确定数据聚类

本节研究的是障碍物动态移动情况下的不确定数据聚类,其中障碍物的动态移动指的是障碍物的点集和边集不发生改变,只是障碍物的位置发生改变.

规则 4. 障碍物的动态移动可以分解成障碍物动态减少和障碍物动态增加 2 部分,障碍物动态移动之前标记为 $O_{i\,beg}$,障碍物移动之后标记为 $O_{i\,end}$. 假设障碍物由位置 P 动态移动到位置 Q,在位置 P 时,相当于障碍物 $O_{i\,beg}$ 动态减少,因此需要分析障碍物动态减少时,由不可视集合移动到可视集合中

的不确定数据点的聚类情况.在位置 Q 时,相当于障碍物 $O_{i\text{end}}$ 的动态增加,因此需要分析障碍物动态增加时,由可视集合移动到不可视集合的不确定数据点的聚类情况.

如图 6 所示,虚线箭头表示的是障碍物动态移动前后位置的变化情况,根据障碍物动态移动的前后位置,移动后的障碍可能对聚类结果有影响,也可能对聚类结果没有影响,因此可以分为 4 种情况分析讨论:

1) 障碍物 O_i 由覆盖圆内部移动到覆盖圆内部,如图 6 障碍物 O_3 移动到 O_2 的情况.

2) 障碍物 O_i 在核心对象 X_i 所在的覆盖圆内部移动到覆盖圆外部,如图 6 障碍物 O_3 移动到 O_4 的情况.

3) 障碍物 O_i 在核心对象 X_i 所在的覆盖圆外部移动到覆盖圆内部,如图 6 障碍物 O_1 移动到 O_3 的情况.

4) 障碍物 O_i 在覆盖圆外移动,如图 6 障碍 O_1 移动到 O_4 的情况.

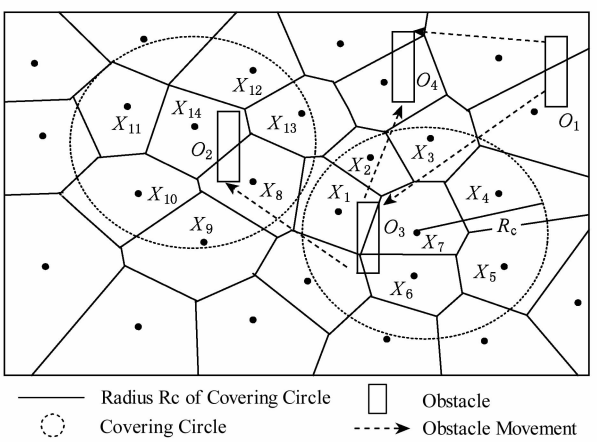


Fig. 6　An example of dynamic movement of obstacles

图 6　障碍物动态移动的示例

由于覆盖圆的圆心都为核心对象,因此不确定数据的分布是密集的.情况 1 中的障碍 O_i 在覆盖圆内移动,动态移动后的障碍集为 $O_{\text{new}}=O-O_{i\text{beg}}+O_{i\text{end}}$;情况 2 中的障碍 O_i 由覆盖圆内移动到覆盖圆外,表明障碍 $O_{i\text{end}}$ 所在区域的不确定数据较稀疏,障碍移动后的位置不影响核心对象的聚类结果,因此 $O_{\text{new}}=O-O_{i\text{beg}}$;情况 3 中的障碍 O_i 由覆盖圆外移动到覆盖圆内,表明障碍 $O_{i\text{end}}$ 所在区域的不确定数据分布是密集的,障碍 $O_{i\text{end}}$ 移动后的位置会影响核心对象的聚类结果,所以 $O_{\text{new}}=O-O_{i\text{beg}}+O_{i\text{end}}$;情况 4 中的障碍 O_i 在覆盖圆外移动,表明障碍 $O_{i\text{beg}}$ 和障碍 $O_{i\text{end}}$ 所在区域的不确定数据分布是稀疏的,障碍动态移动之后的位置不影响核心对象的聚类结果,因此 $O_{\text{new}}=O-O_{i\text{beg}}$.

首先标记动态移动的障碍物在移动之前的位置和移动之后的位置,并判断符合以上哪种情况,进而得到 O_{new}.根据规则 4,通过调用 DYNOR_VUBSCAN 算法和 DYNOC_VUBSCAN 算法,最终实现对动态障碍物移动时的不确定数据进行聚类.

基于以上的分析,给出障碍物动态移动情况下的不确定数据聚类 DYNOM_VUBSCAN 算法,如算法 5 所示.

算法 5. *DYNOM_VUBSCAN*(X,O,O_i).

输入:不确定数据集 X、障碍集 O、移动障碍 O_i;

输出:聚类结果 W.

① *Judge_* $O(O_{i\text{beg}},O_{i\text{end}})$;

② if $O_{i\text{beg}},O_{i\text{end}}$ in *Circle_R*()

③　$O_{\text{new}}\leftarrow O-O_{i\text{beg}}+O_{i\text{end}}$;

④　$W_1\leftarrow$*DYNOR_VUBSCAN*();

　　/* 调用算法 4 */

⑤　$W_2\leftarrow$*DYNOC_VUBSCAN*();

　　/* 调用算法 3 */

⑥　$W\leftarrow W_1\bigcup W_2$;

⑦ end if

⑧ if $O_{i\text{end}}$ in *Circle_R*()且 $O_{i\text{beg}}$ not in *Circle_R*()

⑨　$O_{\text{new}}\leftarrow O-O_{i\text{beg}}+O_{i\text{end}}$;

⑩　$W\leftarrow$*DYNOC_VUBSCAN*();

　　/* 对障碍物 $O_{i\text{end}}$ 调用算法 3 */

⑪ end if

⑫ if $O_{i\text{beg}}$ in *Circle_R*()且 $O_{i\text{end}}$ not in *Circle_R*()

⑬　$O_{\text{new}}\leftarrow O-O_{i\text{beg}}$;

⑭　$W\leftarrow$*DYNOR_VUBSCAN*();

　　/* 对障碍 $O_{i\text{beg}}$ 调用算法 4 */

⑮ end if

⑯ if $O_{i\text{beg}},O_{i\text{end}}$ not in *Circle_R*()

⑰　$O_{\text{new}}\leftarrow O-O_{i\text{beg}}$;

⑱　$W\leftarrow$*STAO_RVUBSCAN*();

　　/* 调用算法 2 */

⑲ end if

⑳ return W.

算法 5 主要通过调用算法 2、算法 3 和算法 4 来实现,由于以上算法是可终止的,并且时间复杂度已经分析得出为 $O(n\lg n)$,进而得出算法 5 的时间复杂度为 $O(n\lg n)$.

2.6 障碍空间下的不确定数据聚类算法

通过分析定义域的情况,分别根据离散区域或者连续域对不确定数据进行聚类,离散域下的不确定数据用概率质量函数表示,连续域下的不确定数据用概率密度函数表示.采用核密度估计[20]方法得出概率密度函数,该方法不需要输入参数,它仅从数据本身对概率密度函数作出估计,并且可以估计任意形状的密度函数.由于概率质量函数在进行相似性度量计算时,比使用概率密度函数更简单,所以对数据域的情况分开处理是有意义的.对于障碍情况的判断,可以根据障碍集合是否变化,判断障碍物是静态的还是动态变化的,最终提出总算法.

根据前面提出的静态障碍物环境下和动态障碍物环境下的不确定数据聚类算法,本节提出 RO_VUBSCAN 算法,使得最终的算法可以高效地解决静态障碍空间环境下的不确定聚类、动态障碍物增加、减少和移动情况下的不确定聚类问题.

首先判断障碍集合是否变化,如果障碍集合不发生改变,说明解决的是静态障碍环境下的不确定数据聚类问题,可调用算法 2 完成聚类.若障碍集合发生变化,可根据障碍数量变化判断是障碍物动态增加、障碍物动态减少还是障碍物动态移动的,如果是障碍动态增加时,则调用算法 3 完成聚类.如果是障碍动态减少时,则调用算法 4 完成聚类.如果是障碍动态移动时,则调用算法 5 完成聚类.基于以上分析,进一步提出障碍空间下的不确定聚类算法 RO_VUBSCAN,如算法 6 所示.

算法 6. RO_VUBSCAN(X,O,ε,R_c).

输入:不确定数据集 X、障碍集 O,ε,R_c;

输出:障碍空间下的不确定数据聚类结果集 W.

① *Create_Vor*(X);
② *Add_ O*(O);
③ for i=1 to n do
④ *Calculate_den*(X_i); /* 计算 $\rho(X_i)$ */
⑤ *Judge_O*(O);
⑥ if O=O_{new}
⑦ W←*STAO_RVUBSCAN*();
⑧ else if O≠O_{new}
⑨ if *Count_O*<*Count_O*$_{new}$
⑩ W←*DYNOC_VUBSCAN*();
⑪ else if *Count_O*>*Count_O*$_{new}$
⑫ W←*DYNOR_VUBSCAN*();
⑬ else W←*DYNOR_VUBSCAN*();
⑭ end if
⑮ end if
⑯ end for
⑰ return W.

如果定义域是连续的,则不确定数据用概率密度函数表示,KL 距离的计算使用式(2),如果定义域是离散的,则不确定数据用概率质量函数表示,相似性度量计算使用式(3).在计算障碍距离时,则使用式(5)进行相似性度量.

算法 6 在创建 Voronoi 图所需的时间复杂度为 $O(n\lg n)$.算法执行 for 循环时,遍历不确定数据点所需的时间复杂度是 $O(n)$.n 代表的是不确定数据的总数量,并且数量是有限的,所以算法是可终止的.当算法满足步骤⑥所需的条件时,调用算法 2.算法 2 的时间复杂度已经分析得出是 $O(n\lg n)$.当算法满足步骤⑧所需的条件时,调用算法 3、算法 4 或者算法 5,以上算法的时间复杂度都为 $O(n\lg n)$.由以上分析可得出,算法 6 总的时间复杂度为 $O(n\lg n)$.

3 实验比较与分析

本节主要通过实验对所提出的算法进行性能分析与比较.根据已有的研究成果发现,在相似性度量方面,大多集中在几何距离的度量上,没有考虑数据间的概率分布.针对这些问题提出了 STAO_RVUBSCAN, DYNOC _ VUBSCAN, DYNOR _ VUBSCAN,DYNOM_VUBSCAN 等算法.由于现有的研究成果无法直接有效地处理动态障碍空间中的不确定数据的聚类问题,因此对文献[16]中提出的 OBS_UK_means 算法中动态添加障碍和动态减少障碍,进而分别与所提算法 DYNOC_VUBSCAN, DYNOR_VUBSCAN,DYNOM_VUBSCAN 进行实验比较与分析.

根据障碍情况,需要对 OBS_UK_means 算法做 4 种处理:

1) 障碍集 O=O_{new}时,则与所提的静态障碍物环境下的聚类情况相同,可利用 OBS_UK_means 算法与 STAO_RVUBSCAN 算法做实验对比.

2) 障碍集 O≠O_{new},并且 *Count_O*<*Count_O*$_{new}$ 时,表示障碍物动态增加,则在原有的聚类结果中添加新增障碍集,再重新对空间中所有的不确定数据点进行聚类.将这种障碍动态增加时调用 OBS_UK_means 的算法与提出的 DYNOC_ VUBSCA 的方法做实验对比.

3) 障碍集 $O\neq O_{new}$,并且 $Count\_O>Count\_O_{new}$ 时,表示障碍动态减少,则对障碍集为 O_{new} 时的空间中所有不确定数据点重新聚类.将这种障碍动态减少时调用 OBS_UK_means 的算法与提出的 DYNOR_VUBSCA 的方法做实验对比.

4) 障碍集 $O\neq O_{new}$,并且 $Count\_O=Count\_O_{new}$ 时,表示障碍物动态移动.对障碍集为 O_{new} 时的所有不确定数据点重新聚类,将这种障碍动态移动时调用 OBS_UK_means 的算法与提出的 DYNOM_VUBSCA 的方法做实验对比.

实验环境:Windows7 的 64 位操作系统,采用 Java 编程.实验中计算机的硬件配置:8 GB 内存,500 GB 硬盘,处理器:Intel® Core™ i3 处理器(主频为 2.30 GHz).

实验使用标准的 UCI 实验室数据集[25],详细情况如表 1 所示:

Table 1　UCI Laboratory Datasets

表 1　UCI 实验室数据集

Dataset	Sample Size	Dimensionality	Category
Iris	150	4	3
Wine	178	13	3
Haberman	306	3	2
Heart	270	13	2

UCI 实验室数据集中的数据表示的是确定的数据,为便于实验,在实验过程中对实验数据集进行了适当调整,生成不确定数据的过程具体有 5 个步骤:1)读取原始数据集的属性;2)遍历数据集的每条数据的每个属性;3)为每一条数据的每一维的属性值随机生成一个概率值,概率值在[0,1]范围之内,最终表达的形式为:$X_i=\{(x_1,p_1),(x_2,p_2),\cdots,(x_d,p_d)\}$,其中每个 x_i 表示 X_i 的属性值,d 为维度,X_i 表示一个不确定数据点;4)得到每条数据概率属性值;5)输出不确定数据集.

实验主要考虑 6 个方面因素:数据维度 d、数据基数 n、障碍物数量、障碍物分布、CPU 运行时间、聚类质量.利用以上 6 方面作为衡量算法的指标.

常用的聚类有效性评测有内部评价法、外部评价法和相关性测试评价.它们能对聚类结果进行评价,得出聚类结果是否最优.实验采用评测指标(silhouette)作为聚类内部有效性评测,实验采用 F-measure 熵[26]作为聚类外部评测标准.

分别对各个算法进行 50 次独立聚类实验,统计每次实验的结果,然后对每种算法求 50 次实验结果的平均值,对比算法的实验结果如表 2 所示.结果显示,对于以上 4 组数据集,RO_VUBSCAN 算法的 F-measure 指标平均值和 S 指标均值均高于 OBS_UK_means 算法和 FOPTICS 算法的评测指标.通过实验可看出,RO_VUBSCAN 算法表现出更好的聚类效果.

Table 2　Comparison of Effectiveness of Algorithms

表 2　算法评测有效性对比

Dataset	Algorithm	F-measure	Silhouette
Iris	RO_VUBSCAN	0.8901	0.7993
	FOPTICS	0.8780	0.8032
	OBS_UK_means	0.8530	0.7751
Wine	RO_VUBSCAN	0.7938	0.8189
	FOPTICS	0.6980	0.7126
	OBS_UK_means	0.7390	0.7845
Haberman	RO_VUBSCAN	0.7230	0.6926
	FOPTICS	0.6548	0.6145
	OBS_UK_means	0.6790	0.6440
Heart	RO_VUBSCAN	0.7482	0.7860
	FOPTICS	0.6903	0.7585
	OBS_UK_means	0.7128	0.7067

图 7 中 OBS_UK_means,RO_VUBSCAN,FOPTICS 这 3 个算法的 CPU 执行时间随着维度 d 的增加而增加,其中 RO_VUBSCAN 算法的 CPU 执行时间上升趋势较平缓.从实验结果看出,使用 KL 距离相似性度量方法比其他的距离度量更高效.由于考虑障碍的存在,随着维度的增加,OBS_UK_means 算法需要考虑障碍的存在,因此 CPU 执行时间大于 FOPTICS 算法.

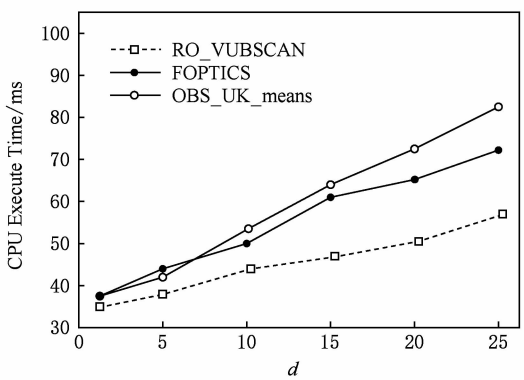


Fig. 7　The effect of dimension d for CPU execution time

图 7　维度 d 对 CPU 执行时间的影响

如图 8 所示,随着维度 d 的不断增加,算法的有效性的变化曲线也呈下降趋势.通过实验,得出在维

度大于 10 之后的曲线,算法的有效性变化曲线下降明显,但 STAO_RVUBSCAN 算法的有效性依然较高. 由于 DYNOR_VUBSCAN 算法和 DYNOR_VUBSCAN 算法处理的是动态障碍的情况,因此有效性较静态障碍空间下 STAO_RVUBSCAN 算法的有效性略低一些.

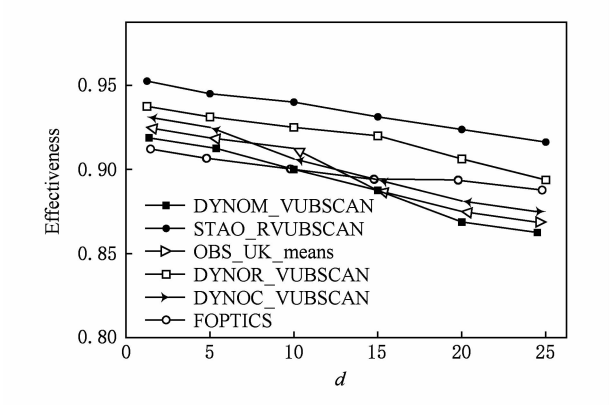


Fig. 8 The effect of dimension d on algorithm effectiveness

图 8 维度 d 对算法有效性的影响

图 9 给出了 3 个算法的 CPU 执行时间随着不确定数据样本数变化的对比结果. 随着不确定数据样本数的不断增大,FOPTICS 算法在数据量增大时,CPU 的执行时间急剧上升. 但在数据量较小的时刻,因为 FOPTICS 算法没有障碍的约束,FOPTICS 算法的 CPU 执行时间比 RO_VUBSCAN 算法的执行时间少. 在相同不确定数据样本数的情况下,RO_VUBSCAN 的 CPU 执行时间一直小于 OBS_UK_means 算法,因此通过实验得知 RO_VUBSCAN 算法更有效.

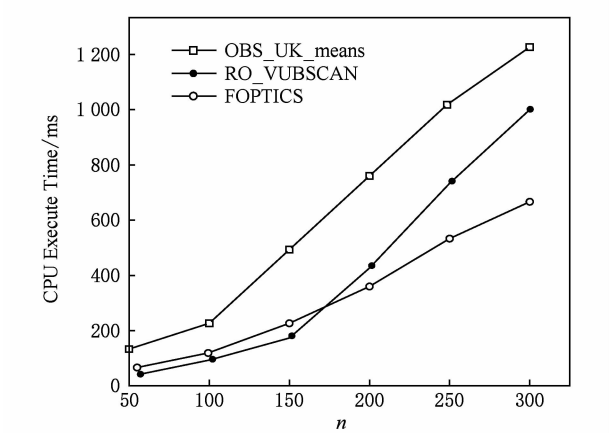


Fig. 9 The effect of sample number on CPU execution time

图 9 样本数对 CPU 执行时间的影响

图 10 给出了数据量从 5 000 增加到 30 000 的过程中,3 个算法对应的 CPU 执行时间的变化情况. 通过实验得出,RO_VUBSCAN 算法的 CPU 执行时间相对于 OBS_UK_means,FOPTICS 更少. 因此得出算法 RO_VUBSCAN 算法更有效.

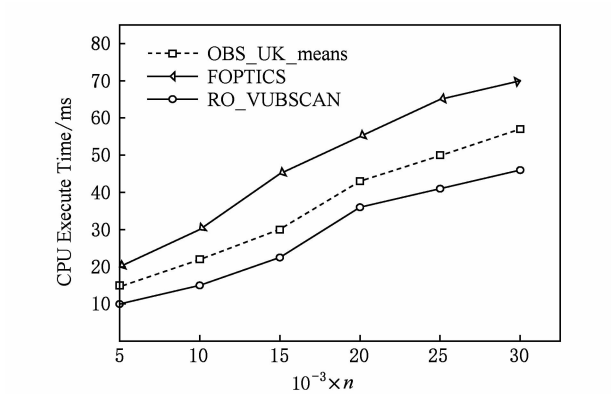


Fig. 10 The effect of data size on CPU execution time

图 10 数据量对 CPU 执行时间的影响

如图 11 所示,分析了数据量从 5 000 增加到 30 000 期间,随着数据量的增加,3 个聚类算法的有效性. 通过实验结果可看出,FOPTICS 算法的曲线平稳,并且效率一直较高. 相比于 RO_VUBSCAN 算法,由于障碍的存在,聚类的过程中,存在着一定的误差,在数据量增加到 20 000 期间有效性比 FOPTICS 略低,随着数据量的继续增加,算法的有效性比 FOPTICS 高. 相比于 OBS_UK_means 算法,RO_VUBSCAN 算法更有效.

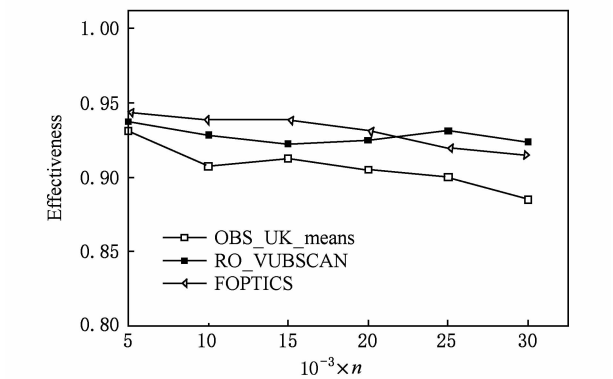


Fig. 11 The effect of data size on the effectiveness of algorithm

图 11 数据量对算法有效性的影响

表 3 给出了障碍物动态减少时 DYNOR_VUBSCAN 算法与对比算法的 CPU 执行时间的比较情况. 结果显示,DYNOR_VUBSCAN 算法的 CPU 执行时间均小于对比算法,当障碍物动态减少时,由于对比算法需要计算全部数据,因此 CPU 执行时间远大于 DYNOR_VUBSCAN 算法.

Table 3　The Effect of the Dynamic Reduction of Obstacles on the Execution Time of CPU

表 3　障碍物动态减少对 CPU 执行时间的影响

Algorithm	Number of Obstacles	CPU Time/s
DYNOR_VUBSCAN	100	10.7
OBS_UK_means	100	15.6
DYNOR_VUBSCAN	90	12.4
OBS_UK_means	90	19.8

表 4 给出了障碍物动态增加时 DYNOC_VUBSCAN 算法与对比算法的 CPU 执行时间的比较情况,在障碍物数量动态增加时,DYNOC_VUBSCAN 算法的 CPU 执行时间变化不大,而对比算法的 CPU 执行时间变化则较大.结果表明:所提算法优于对比算法.

Table 4　The Effect of the Dynamic Increase of Obstacles on the Execution Time of CPU

表 4　障碍物动态增加对 CPU 执行时间的影响

Algorithm	Number of Obstacles	CPU Time/s
DYNOC_VUBSCAN	100	11.1
OBS_UK_means	100	15.9
DYNOC_VUBSCAN	110	12.8
OBS_UK_means	110	21.0

表 5 给出了障碍物动态移动时 DYNOM_VUBSCAN 算法与对比算法的 CPU 执行时间的比较情况.结果显示 DYNOM_VUBSCAN 算法的 CPU 执行时间远小于对比算法的 CPU 执行时间.

Table 5　The Effect of the Dynamic Movement of Obstacles on the Execution Time of CPU

表 5　障碍物动态移动对 CPU 执行时间的影响

Algorithm	Obstacle	Number of Obstacles	CPU Time/s
DYNOM_VUBSCAN	O	100	12.9
OBS_UK_means	O	100	19.2
DYNOM_VUBSCAN	O_{new}	100	14.9
OBS_UK_means	O_{new}	100	24.7

表 6 显示了障碍物数量增加时 STAO_RVUBSCAN 算法与对比算法的 CPU 执行时间的比较情况,在障碍物数量不断增加时,STAO_RVUBSCAN 算法的 CPU 执行时间增加的趋势较缓慢,而对比算法的 CPU 执行时间变化较大.

表 7 给出了障碍物分布不同时 RO_VUBSCAN 算法与对比算法的 CPU 执行时间比较情况,障碍物的分布是随机的,结果显示 RO_VUBSCAN 算法的 CPU 执行时间变化不大,并且执行时间均小于对比算法.

Table 6　The Effect of the Different Number of Obstacles on the Execution Time of CPU

表 6　不同数量的障碍物对 CPU 执行时间的影响

Algorithm	Number of Obstacles	CPU Time/s
STAO_RVUBSCAN	100	13.2
	200	18.7
	300	26.5
	400	29.7
OBS_UK_means	100	18.6
	200	33.5
	300	48.2
	400	63.7

Table 7　The Effect of Different Location Distribution of Obstacles on the Execution Time of CPU

表 7　障碍物不同位置分布对 CPU 执行时间的影响

Algorithm	Obstacle	Number of Obstacles	CPU Time/s
RO_VUBSCAN	o	100	15.2
OBS_UK_means	o	100	19.8
RO_VUBSCAN	o_{new}	100	16.3
OBS_UK_means	o_{new}	100	21.4

通过以上实验分析可知,提出的静态障碍物环境下的不确定数据聚类算法 STAO_RVUBSCAN 与动态障碍物环境下的 DYNOC_VUBSCAN,DYNOR_VUBSCAN,DYNOM_VUBSCAN 不确定聚类算法均具有较高的效率.

4　结束语

传统的聚类算法大多是在欧氏空间进行,没有考虑障碍的存在.由于现实生活中,不确定数据点之间可能存在障碍,因此本文根据障碍物集合是否发生变化,把障碍大致分成 2 种情况,分别对静态障碍空间下和动态障碍空间下的不确定数据进行聚类分析.静态障碍空间中,提出了 STAO_VUBSCAN 算法和精炼算法 STAO_RVUBSCAN.精炼算法的提出,使得聚类结果更准确和高效.对于动态障碍,考虑了障碍物动态增加、障碍物动态减少和障碍物动态移动 3 种情况,分别对这 3 种情况进行分析,提出了 DYNOC_VUBSCAN 算法、DYNOR_VUBSCAN

算法和 DYNOM_VUBSCAN 算法. 理论研究和实验分析验证表明,提出的算法具有较高的性能. 由于没有对确定数据进行聚类分析,未来研究的重点主要是障碍物动态情况下的确定数据聚类问题.

参 考 文 献

[1] Zhang Xianchao, Liu Han, Zhang Xiaotong. Novel density-based and hierarchical density-based clustering algorithms for uncertain data [J]. Neural Networks, 2017, 93(9): 240-255

[2] Zhou Aoying, Jin Cheqing, Wang Guoren, et al. A survey on the management of uncertain data [J]. Chinese Journal of Computers, 2009, 32(1): 1-16 (in Chinese)
(周傲英, 金澈清, 王国仁, 等. 不确定性数据管理技术研究综述[J]. 计算机学报, 2009, 32(1): 1-16)

[3] Ngai W K, Kao B, Chui C K, et al. Efficient clustering of uncertain data [C] //Proc of the 6th Int Conf on Data Mining. Piscataway, NJ: IEEE, 2006: 436-445

[4] Wang Juntao, Su Xiaolong. An improved k-means clustering algorithm [C] //Proc of the 3rd Int Symp on Intelligent Information Technology and Security Informatics. Piscataway, NJ: IEEE, 2011: 63-67

[5] Wang Wei, Yang Jiong, Muntz R. STING: A statistical information grid approach to spatital data mining [C] //Proc of the 23rd Int Conf on Very Large Data Bases. San Francisco: Morgan Kaufmann, 1997: 186-195

[6] Xu Lei, Hu Qinghua, Zhang Xisheng, et al. AdaUK-Means: An ensemble boosting clustering algorithm on uncertain objects [C] //Proc of the 7th Chinese Conf on Pattern Recognition. Berlin: Springer, 2016: 27-41

[7] Liao Kuanteng, Liu Chuanming. An effective clustering mechanism for uncertain data mining using centroid boundary in UKmeans [C] //Proc of Int Computer Symp. Piscataway, NJ: IEEE, 2017: 300-305

[8] Hao Shengxuan, Zhou Xiaofeng, Hong Song. A new method for noise data detection based on DBSCAN and SVDD [C] //Proc of the 5th IEEE Int Conf on Cyber Technology in Automation, Control, and Intelligent Systems. Piscataway, NJ: IEEE, 2015: 784-789

[9] Kriegel H P, Pfeifle M. Hierarchical density-based clustering of uncertain data [C] //Proc of the 5th IEEE Int Conf on Data Mining. Piscataway, NJ: IEEE, 2005: 689-692

[10] Han Liu, Zhang Xianchao, Zhang Xiaotong, et al. Self-adapted mixture distance measure for clustering uncertain data [J]. Knowledge-Based Systems, 2017, 126(12): 33-47

[11] Erdem A, Gündem T I. M-FDBSCAN: A multicore density-based uncertain data clustering algorithm [J]. Turkish Journal of Electrical Engineering & Computer Sciences, 2014, 22(1): 143-154

[12] Lu Yihong, Xia Cong. Optimal k-nearest neighbors and local density-based clustering algorithm for uncertain data [J]. Control and Decision, 2016, 31(3): 541-546 (in Chinese)
(陆亿红, 夏聪. 不确定数据的最优 k 近邻和局部密度聚类算法[J]. 控制与决策, 2016, 31(3): 541-546)

[13] Han Lizhao, Qian Xuezhong, Luo Jing, et al. Multi-density clustering algorithm DBSCAN based on region division [J]. Application Research of Computers, 2018, 35(6): 1668-1671 (in Chinese)
(韩利钊, 钱雪忠, 罗靖, 等. 基于区域划分的 DBSCAN 多密度聚类算法[J]. 计算机应用研究, 2018, 35(6): 1668-1671

[14] Ben Kao, Lee S D, Lee F K F. Clustering uncertain data using voronoi diagrams and R-Tree index [J]. IEEE Transactions on Knowledge & Data Engineering, 2010, 22(9): 1219-1233

[15] Zhang Jun, Papadias D, Mouratidis K, et al. Spatial queries in the presence of obstacles [C] //Proc of the 9th Int conf on Extending Database Technology. Berlin: Springer, 2004: 366-384

[16] Cao Keyan, Wang Guoren, Han Donghong, et al. Clustering algorithm of uncertain data in obstacle space [J]. Journal of Frontiers of Computer Science & Technology, 2012, 6(12): 1087-1097 (in Chinese)
(曹科研, 王国仁, 韩东红, 等. 障碍空间中不确定数据聚类算法[J]. 计算机科学与探索, 2012, 6(12): 1087-1097)

[17] Zhang Xueping, Du Haohua, Yang Tengfeng, et al. A novel spatial clustering with obstacles constraints based on PNPSO and K-Medoids [G] //Advances in Swarm Intelligence. Berlin: Springer, 2010: 605-610

[18] Zhou Jin, Pan Yuqi, Chen C L P, et al. K-medoids method based on divergence for uncertain data clustering [C] //Proc of the 46th IEEE Int Conf on Systems, Man, and Cybernetics. Piscataway, NJ: IEEE, 2017: 2671-2674

[19] Pan Donghua, Zhao Lilei. Uncertain data cluster based on DBSCAN [C] //Proc of the 2nd Int Conf on Multimedia Technology. Piscataway, NJ: IEEE, 2011: 3781-3784

[20] Hao Zhongxiao. Spatial Databases Theoretical Basis [M]. Beijing: Science Press, 2013 (in Chinese)
(郝忠孝. 空间数据库理论基础[M]. 北京: 科学出版社, 2013)

[21] Zhang Liping, Liu Lei, Hao Xiaohong, et al. Voronoi-based group reverse k nearest neighbor query in obstructed space [J]. Journal of Computer Research and Development, 2017, 54(4): 861-871 (in Chinese)
(张丽平, 刘蕾, 郝晓红, 等. 障碍空间中基于 Voronoi 图的组反 k 最近邻查询研究[J]. 计算机研究与发展, 2017, 54(4): 861-871)

[22] Wang Jianrong. Research on clustering algorithm for uncertain data based on probability distribution similarity [D]. Xi'an: Xidian University, 2014 (in Chinese)

(王建荣. 基于概率分布相似性的不确定数据聚类算法研究[D]. 西安:西安电子科技大学,2014)

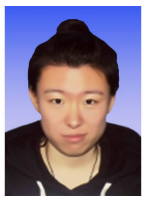
[23] Xu Lei, Hu Qinghua, Hung E, et al. Large margin clustering on uncertain data by considering probability distribution similarity [J]. Neurocomputing, 2015, 158(12): 81-89

[24] Ma Shuai, Wang Tengjiao, Tang Shiwei, et al. A fast clustering algorithm based on reference and density [J]. Journal of Software, 2003, 14(6): 1089-1095 (in Chinese)
(马帅,王腾蛟,唐世渭,等. 一种基于参考点和密度的快速聚类算法[J]. 软件学报,2003,14(6):1089-1095)

[25] He Yunbin, Zhang Zhichao, Wang Jing, et al. Research for uncertain data clustering algorithm: U-PAM and UM-PAM algorithm [J]. Computer Science, 2016, 43(6): 263-269 (in Chinese)
(何云斌,张志超,万静,等. 不确定数据聚类的 U-PAM 算法和 UM-PAM 算法的研究[J]. 计算机科学,2016,43(6):263-269)

[26] Cao Zhenli, Sun Ruizhi, Li Meng. A method for clustering uncertain data streams based on GMM [J]. Journal of Computer Research and Development, 2014, 51(Suppl Ⅱ): 102-109 (in Chinese)
(曹振丽,孙瑞志,李勐. 一种基于高斯混合模型的不确定数据流聚类方法[J]. 计算机研究与发展,2014,51(增刊Ⅱ):102-109)

Wan Jing, born in 1972. PhD. Professor. Her main research interests include database theory and application, spatial database, embedded system.



Cui Meiyu, born in 1994. Master candidate. Her main research interests include data mining and spatial data clustering. (254143488@qq.com)

He Yunbin, born in 1972. PhD. Professor. His main research interests include database theory and application. (hybha@163.com)

Li Song, born in 1977. PhD. Professor. His main research interests include database theory and application, data mining, data query. (lisongbeifen@163.com)