

基于宽度优先搜索的 K -medoids 聚类算法

颜宏文¹, 周雅梅¹, 潘 楚^{1,2*}

(1. 长沙理工大学 计算机与通信工程学院, 长沙 410114; 2. 湖南大学 信息科学与工程学院, 长沙 410082)

(* 通信作者电子邮箱 panchu2012@163.com)

摘 要:针对传统 K -medoids 聚类算法对初始值敏感、中心点随机选择以及聚类精度不够高等缺点,在粒计算有效初始化的基础上,提出中心点宽度优先搜索策略。首先,利用粒计算初始化获取 K 个有效粒子,遴选该 K 个粒子所对应的 K 个中心点作为 K 个初始中心点;然后,根据对象间的相似性分别对 K 个粒子中的对象建立以中心点为根节点的相似对象二叉树,通过宽度优先搜索遍历二叉树迭代出最优中心点,同时采用簇间距离和簇内距离优化准则函数。实验结果表明,所提算法在 UCI 中 Iris 和 Wine 标准数据集中测试,在有效缩短迭代次数的同时保证了算法聚类准确率。

关键词: K -medoids 聚类算法; 粒计算; 相似对象二叉树; 宽度优先搜索; 适应度函数

中图分类号: TP301 **文献标志码:** A

Novel K -medoids clustering algorithm based on breadth-first search

YAN Hongwen¹, ZHOU Yamei¹, PAN Chu^{1,2*}

(1. School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha Hunan 410114, China;
2. College of Computer Science and Electronic Engineering, Hunan University, Changsha Hunan 410082, China)

Abstract: Due to the disadvantages such as sensitivity to the initial selection of the center, random selection of centers and poor accuracy in traditional K -medoids clustering algorithm, a breadth-first search strategy for centers was proposed on the basis of granular computing effective initialization. The new algorithm selected K granules firstly using granular computing and selected their corresponding centers as the K initial centers. Secondly, according to the similarity between objects, the proposed algorithm set up binary tree of similar objects separately where the corresponding initial centers were taken as the root nodes, and then used breadth-first search to traverse the binary tree to find out K optimal centers. What's more, the fitness function was optimized by using within-cluster distance and between-cluster distance. The experimental results on standard data set Iris and Wine in UCI show that this proposed algorithm effectively reduces the number of iterations and guarantees the accuracy of clustering at the same time.

Key words: K -medoids clustering algorithm; granular computing; binary tree of similar object; breadth-first search; fitness function

0 引言

K -medoids 聚类算法是一种基于划分方法的聚类算法^[1],在处理含有异常数据和噪声数据的数据集时,具有很好的鲁棒性,在聚类算法中得到广泛的应用^[2]。但是该算法依然存在不少问题,如对初始中心点的选择比较敏感,同时该算法中中心点的生成是在数据集中剩余的所有非中心点对象中进行随机选择,从而导致算法迭代次数比较高而聚类准确率不高。因此,如何进行有效初始化和降低算法计算量,同时提高聚类准确率是改进 K -medoids 聚类算法的重要研究方向。

目前,已有一些可行的改进模型。文献[3]通过计算对象的密度信息来产生初始中心点,文献[4]提出一种选择核心要素的有利位置的子集作为初始值,文献[5]利用粒计算法来初始化聚类中心点,但由于它们基于传统的中心点搜索策略使得算法的性能没能得到很好的提升。文献[6-7]分

别使用子空间密度聚类和局部更新密度聚类进行数据流聚类,有效解决了非球形数据聚类问题。文献[8]通过一次性计算对象间的距离生成矩阵,根据矩阵初始化中心点,并提出局部中心点更新策略。文献[9]通过粒计算有效初始化聚类中心点,同时提出中心点粒度迭代搜索策略;在有效初始化基础上提出局部搜索策略,在降低迭代次数的同时提高算法性能。

以上文献中,文献[3-5]提出的初始化方法,克服了传统 K -medoids 聚类算法对初始化的敏感性,具有很好的借鉴意义。文献[6-9]提出局部更新策略可取。综上所述,对数据的初始化处理就是寻找一种有效的数据规约方法;中心点区域更新策略在本质上就是把中心点的搜索更新范围锁定在一个有效的小区域内,但此时中心点的选择还是具有随机性。鉴于此,本文首先采用粒计算进行有效初始化,克服算法对初始化敏感;然后提出一种中心点宽度优先搜索策略,对中心点

收稿日期:2014-12-05;修回日期:2015-01-12。

基金项目:国家自然科学基金资助项目(51277015);湖南省研究生科研创新项目(CX2014B386)。

作者简介:颜宏文(1968-),女,湖南株洲人,教授,博士,主要研究方向:数据挖掘、电网数据通信;周雅梅(1989-),女,湖南娄底人,硕士研究生,主要研究方向:数据挖掘;潘楚(1986-),男,湖南怀化人,博士研究生,主要研究方向:数据挖掘、复杂网络。

进行精确搜索,避免中心点搜索随机性,进一步降低算法迭代次数;同时利用平衡簇内距离和簇间距离的准则函数来提高评价聚类标准,从而保证高准确率。

1 预备知识

1.1 K-medoids 算法

传统 K-medoids 聚类算法选取数据集中实际对象来代表簇,随机选择 K 个代表对象,剩余的对象被分配到与其最为相似的代表对象所在的簇中。划分方法是基于最小化所有对象 p 与其对应的代表对象 o_i 之间的相异程度之和的原则来进行划分。使用一个绝对误差标准函数^[1]来衡量聚类效果,定义为式(1):

$$E(w) = \sum_{i=1}^K \sum_{p \in c_i} |p - o_i|^2 \quad (1)$$

其中: p 为簇 c_i 中的对象, o_i 为聚类中心, p 和 o_i 是多维的。

算法首先在数据集中随机选择 K 个对象作为初始聚类中心点 o_1, o_2, \dots, o_K , 然后把剩余对象分配到离其距离最近的中心点 o_i 所在簇中, 得到一组聚类划分后计算 $E(w)$; 接着在数据集中重新随机选择 K 个中心点 o_1', o_2', \dots, o_K' , 然后围绕该 K 个中心点进行重新划分, 得到新一组聚类划分后, 计算 $E'(w)$, 然后比较 $\Delta E = E'(w) - E(w)$, 若 $\Delta E < 0$, 则 o_1', o_2', \dots, o_K' 替换 o_1, o_2, \dots, o_K 成为新的中心点; 否则不变。这样不断地进行中心点迭代更新, 直到聚类质量不可能被任何替换提高为止。

1.2 粒计算

粒计算(Granular Computing, GrC)是美国学者 Lin^[40]于1997年提出的。聚类分析是以“最优”相似度函数为基础, 在所有可能的粒度中, 求出一个“最优”粒度^[11]。粒度和聚类有着千丝万缕的关联, 将粒概念融入聚类算法能够更有力地执行聚类操作^[12]。结合粒和数据挖掘的相关概念, 基于粒计算的初始化具体表述如下。

定义1 对象相似度及对象平均相似度。设 $T = (U, B)$ 为聚类空间, U 为对象集, B 为属性集合。对象相似度 $S(x_i, x_j)$ 定义如式(2)所示:

$$S(x_i, x_j) = 1 / (1 + \sum_{l=1}^{|B|} w_l |x_{il} - x_{jl}|) \quad (2)$$

其中 w_l 为属性分辨能力, 定义如式(3)所示:

$$w_l = \sum_{i=1}^m |x_i|^2 / |U|^2 \quad (3)$$

属性 l 值被划分成区间块 $\{x_1, x_2, \dots, x_m\}$, $|x_i|$ 为 x_i 划分区间的对象个数, m 为划分块数, $|U|$ 为对象总数。若对象个数为 n , 则对象平均相似度定义如式(4)所示:

$$\bar{d} = \sum_{i,j=1}^n S(x_i, x_j) / n^2 \quad (4)$$

定义2 粒子密度及粒子的平均密度。设 U 上的 n 个粒子划分为 $\{X_1, X_2, \dots, X_n\}$, 则其中粒子 X_i 的密度定义如式(5)所示:

$$gd(X_i) = |X_i| / |U| \quad (5)$$

其中 $|X_i|$ 表示第 i 个区间里的对象个数, 则 n 个粒子的平均密度定义如式(6)所示:

$$\overline{GD} = \left(\sum_{i=1}^n gd(X_i) \right) / n \quad (6)$$

定义3 粒子中心。设第 i 个粒子包含 N 个对象, 分别为 $x_{i1}, x_{i2}, \dots, x_{iN}$, 则该粒子的中心 o_i 定义如式(7)所示:

$$o_i = \left\{ x_{ij} \left| \min_{j=1}^N \left| x_{ij} - \frac{1}{N} \sum_{k=1}^N x_{ik} \right| \right. \right\} \quad (7)$$

粒计算初始化具体步骤如下:

步骤1 根据式(2)、(4), 分别计算对象间的相似度以及对象平均相似度。

步骤2 设定一个阈值 d , 其中 d 的范围为 \bar{d} 上下浮动 10% 左右, 具体大小根据实验效果设定。如果 $S(x_i, x_j) \geq d$, 则 $M(i, j) = 1$; 否则 $M(i, j) = 0$, 其中 M 表示对象间的模糊相似矩阵。

步骤3 将对象按矩阵 M 进行归类得到对象的粗粒集 G , 即记录每个对象的相似对象的编号。每个对象与其相似的所有对象的集合称为一个粒子, 在归类过程中采用一个样本(已经编号)只跟其后的样本逐个比较, 从而避免了样本重复归类, 得到更好的粗粒集。其次在此粗粒集的基础上进行去重处理, 即去除粒子中样本个数、属性值都相同的多余粒子, 得到样本个数按大小排列且不重复的细粒集 \bar{G} 。

步骤4 根据式(5)计算每个粒子的密度, 按式(6)计算细颗粒集 \bar{G} 的平均密度, 选择 $gd(x_i) \geq \overline{GD}$ 的粒子作为有效粒子放入集合 I 中, 称 I 为有效粒子集。

步骤5 按式(7)计算有效粒子集 I 中每个粒子的中心, 并计算任意两颗粒子间的欧氏距离记录在 D 中。

步骤6 在 I 中取最大密度粒子的中心作为第 1 个聚类中心 o_1 , 该粒子记为 Q_1 ; 选择距最大密度粒子最远且密度最大的粒子的中心点作为第 2 个聚类中心 o_2 , 该粒子记为 Q_2 ; 对于 I 中剩余粒子, 根据 D , 分别求出其中心点到 o_1, o_2, \dots, o_n 的距离为 $d_{i1}, d_{i2}, \dots, d_{in}$, 取 $d_i = \min(d_{i1}, d_{i2}, \dots, d_{in})$, 然后计算 $d = \max(d_i)$, 对应的粒子中心为 o_i , 该粒子记为 Q_i , 以此类推计算 o_k , 对应粒子记为 Q_k 。

2 算法的改进

2.1 中心点宽度优先搜索

传统 K-medoids 聚类算法的两大模块分别为中心点的更新和重新聚类划分。其中, 中心点的更新策略是在剩余 $n - K$ 个对象中进行随机搜索选择, 由于中心点大范围搜索的随机性, 从而导致算法迭代次数较高而且准确率不高。文献[9]提出的粒度迭代搜索策略, 把中心点的搜索范围锁定在 K 个有效的粒子内, 在一定程度上降低算法迭代次数, 但由于有效粒子的不纯性(粒子中的对象不全属于同一簇), 搜索过程中还是带有随机性。为了提高算法的性能, 本文在获取 K 个有效粒子的基础上, 基于有效粒子的高密度和高相似性原则, 判定与初始中心点较近的对象处于同一簇, 进一步判定这些与初始中心点较近的对象是最优中心点的最佳候选对象。所以, 我们为每个粒子建立以初始中心点为根节点的相似对象二叉树 T_1, T_2, \dots, T_k 。在相似对象二叉树中, 左节点与根节点的相似性比右节点与根节点相似性高, 上一层节点与根节点的相似性比下一层节点与根节点相似性高。这样做是为了让中心点 o_1, o_2, \dots, o_k 的更新不再是随机搜索, 而是在对应的相似对象二叉树 T_1, T_2, \dots, T_k 上进行宽度优先搜索, 在遍历对象的同时进行对象的动态组合, 即每棵 T_i 每遍历一步得到的对象都要与其他树在这之前遍历得到的对象进行一个动态组

合,从而得到新的中心点。

2.2 准则函数

聚类的核心思想是要求簇内距离最小、簇间距离最大。根据这一原则,本文也采用簇间距离和簇内距离共同优化准则函数。当改进的准则函数达到最大值,算法将得到最优聚类结果。

设类间距离 $O(w)$ 为不同簇中心点的距离,定义为:

$$O(w) = \sum_{i,j=1}^K |o_i - o_j|^2 \quad (8)$$

改进的准则函数可以表示为:

$$F(w) = \frac{\sqrt{O(w)}}{\sqrt{E(w)}} \quad (9)$$

2.3 改进算法的具体描述

输入 包含 n 个对象的数据集 $Data$,簇的数目 K 。

输出 K 个簇类中心及聚类准确率。

步骤 1 利用粒计算初始化 K 个聚类中心 o_1, o_2, \dots, o_K , 同时记录中心点所对应的粒子为 Q_1, Q_2, \dots, Q_K , 为每个粒子建立以初始中心点为根节点的相似对象二叉树 T_1, T_2, \dots, T_K 。

步骤 2 把剩余对象指派给离对象最近的中心点所代表的簇,记为 w ,并计算 $F(w)$ 。

步骤 3 分别在对应的 T_1, T_2, \dots, T_K 中进行一次宽度优先搜索遍历对象。

步骤 4 每遍历一棵树的对象的同时进行对象间的动态组合,形成新的中心点 o_1', o_2', \dots, o_K' 代替之前对应粒子中的代表对象 o_1, o_2, \dots, o_K , 并重新聚类划分, 记为 w' , 计算 $F(w')$ 。

步骤 5 计算 $\Delta F = F(w') - F(w)$; 若 $\Delta F > 0$, o_i' 替换 o_i , 此时记 w' 为 w ; 否则不变。

步骤 6 重复执行步骤 4 ~ 步骤 5, 直到遍历所得的所有对象完成动态组合的所有可能组合对为止。

步骤 7 重复执行步骤 3 ~ 步骤 6, 直到簇集合中对象不再发生改变,结束算法。

3 实验结果与分析

实验环境: AMD Dual-Core CPU T4400 ++ @ 2.31 GHz, 2 GB 的内存, 操作系统 Windows 7, 集成开发软件 VC++ 6.0, Matlab 2009。实验采用的数据集: 本文选用 UCI 中的 Iris、Wine 标准数据集。

为了验证本文算法的可行性和高效性, 采用 PAM (Partitioning Around Medoid) 算法、快速 K-medoids 算法、文献 [9] 算法及本文算法, 分别在以上数据集上进行聚类测试, 每种算法运行 100 次计算其平均值。本文在 Iris、Wine 标准数据集特征如表 1 所示。

表 1 实验中涉及的数据集

数据集	样本数目	属性维度	类别个数	每类个数
Iris	150	4	3	50, 50, 50
Wine	178	13	3	59, 71, 48

对于 PAM 算法和快速 K-medoids 算法, 设置标准数据集 Iris、Wine 的簇个数均为 $K = 3$ 。针对文献 [9] 算法, 根据各数据集 \bar{d} 的不同和数据集不同, 通过实验验证得出参数设置如

下: 其中 Iris 数据集参数: $d = 0.730$, $K = 3$; Wine 数据集参数: $d = 0.052$, $K = 3$ 。

首先, 给出 Iris、Wine 数据集的初始化结果, 如表 2、3 所示。其中在表 2、3 中的加下划线的对象编号即为与中心点属于同一簇的对象编号。

表 2 Iris 数据集初始化结果

粒子编号	对象个数	中心点编号	有效粒子中包含的对象编号
$Q_1 = 2$	69	2	2, 6, 9, 15, 17, 18, 19, 21, 22, 23, 24, 30, 31, 32, 34, 35, 36, 37, 38, 39, 47, 49, 50, 51, 52, 53, <u>60, 62</u> , 65, 66, 67, 68, <u>75, 77</u> , 78, 79, 80, 81, 82, 84, 90, 92, 93, 95, 96, 97, 98, <u>105, 106, 107, 108, 109, 110</u> , 111, 112, 114, <u>120, 121, 122, 123, 124</u> , 128, 129, <u>135, 139</u> , 141, 142, 143, 144
			<u>21, 23, 24, 35, 36, 37, 38, 39, 49, 50, 51, 52, 53, 54, 65, 66, 67, 68, 69, 75, 78, 80, 81, 82, 84, 95, 96, 97, 98, 108, 110, 111, 112, 113, 114, 122, 124, 125, 128, 129, 140, 141, 142, 143</u>
$Q_2 = 21$	44	112	28, 29, 40, 41, 42, 43, 44, 55, 56, 58, 70, 71, 72, 74, 85, 86, 87, 88, 89, 100, 101, 102, 103, 115, 116, 117, 118, 119, 131, 132, 133, 134, 145, 146, 147, 148, 149
$Q_3 = 28$	37	43	

表 3 Wine 数据集初始化结果

粒子编号	对象个数	中心点编号	有效粒子中包含的对象编号
$Q_1 = 61$	50	91	61, 63, 64, 66, 67, 71, 72, 75, 76, 77, 79, 83, 84, 85, 86, 90, 91, 92, 94, 97, 99, 102, 103, 106, 107, 113, 114, 115, 116, 118, 121, 122, 123, 124, 125, 127, 131, 132, 137, 142, 143, 146, 150, 151, 152, 156, 160, 165, 170, 171
			<u>2, 6, 11, 13, 16, 17, 26, 27, 30, 33, 34, 37, 42, 45, 49, 50, 51, 52, 55, 57, 58</u>
$Q_2 = 2$	21	26	4, 19, 20, 21, 25, 39, 40, 43, 60, 65, 68, 69, 78, 81, 88, 98, 100, 104, 109, 130, 135, 136, 141, 145, 153, 155, <u>158, 161, 163, 166, 167, 168, 172, 173, 174, 175, 176</u>
$Q_3 = 4$	37	155	

表 2 中所遴选得到的中心点 o_1, o_2, o_3 分别为: 2, 112, 43, 它们对应地位于 Iris 标准数据集中的第 0 簇、1 簇、2 簇。与之对应的粒子 Q_1, Q_2, Q_3 的编号分别为: 2, 21, 28, 它们也对应地分别位于 Iris 标准数据集中的第 0 簇、1 簇、2 簇。同样, 表 3 所遴选得到的中心点 o_1, o_2, o_3 分别为: 91, 26, 155, 它们对应地位于 Wine 标准数据集中的第 1 簇、0 簇、2 簇。与之对应的粒子 Q_1, Q_2, Q_3 的编号分别为: 61, 2, 4, 它们对应分别位于 Wine 标准数据集中的第 1 簇、0 簇、0 簇。在 Wine 数据集中, 所遴选出来的粒子 Q_3 与中心点 o_3 不属于同一簇, 但仔细观察 Q_3 中所包含的对象, 其中有 18 个对象与中心点属于同一簇, 所以小误差不影响后续的工作。观察表 2、表 3, 可以获知粒子中大部分的对象与中心点属于同一簇。基于有效粒子的高密度和高相似性, 建立以中心点为根节点的相似对象二叉树是有根据的。

那么, 根据以上实验结果且结合 2.1 节的分析, 本文为每个

粒子构建以中心点为根节点的相似对象二叉树 T_1, T_2, \dots, T_K 。构建的 T_1, T_2, \dots, T_K 如图1、图2所示,这里只列出树的前3层。

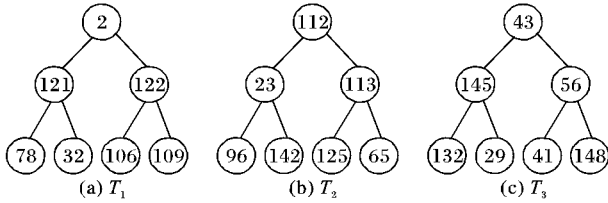


图1 Iris 数据集的相似对象二叉树 T_1, T_2, T_3

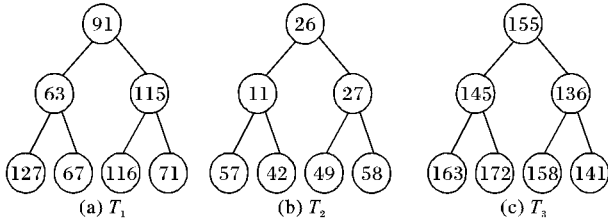


图2 Wine 数据集的相似对象二叉树 T_1, T_2, T_3

分别对比表2、图1和表3、图2中的数据,分析得知离根节点最近的几层节点都与根节点属于同一簇,验证了2.1节中的与初始中心点较近的对象处于同一簇的推断。

通过实验比较各算法在 Iris、Wine 数据集中准确率,实验数据如表4所示。

表4 各类聚类算法在数据集上的准确率比较 %

数据集	PAM 算法	快速 K-medoids 算法	文献[9]算法	本文算法
Iris	77.56	89.32	96.13	97.23
Wine	52.87	70.79	84.26	84.26

表4的实验结果表明本文算法在两组不同数据集中测试,其聚类准确率都有很大的提升。对于 Iris 数据集,本文算法聚类准确率最高能达到98.00%,收敛于121、112、43,而这一组中心点都是处于 T_1, T_2, T_3 第0、1层的对象;同样对于 Wine 数据集,本文算法最优聚类准确率也能达到86.45%,收敛于115、11、145,同样这一组中心点都是处于 T_1, T_2, T_3 第1层的对象。结果显示,本文算法能在有限的迭代次数中搜索到最优解或者近似最优解。

最后,鉴于准则函数一致,本文比较文献[9]算法与本文算法的收敛速度,即搜索到最优中心点的迭代次数,如图3所示。

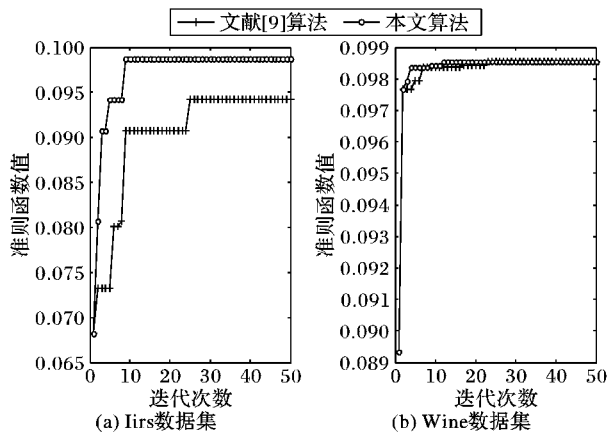


图3 搜索最优中心点的迭代次数

可以看出,两种算法的准则函数起始值相同,在算法准确率差不多的情况下,与文献[9]小范围的随机搜索策略相比较,本文提出的中心点宽度优先搜索策略,精确搜索新的中心点,所以迭代次数更小,收敛速度更快,证明该搜索策略的高

效性。

4 结语

综上所述,本文算法在 Iris、Wine 的数据集上进行测试,能有效初始化初始聚类中心点,使其分别位于不同的簇中,避免了传统算法对初始化敏感的问题;提出中心点宽度优先搜索策略对中心点进行精确搜索更新,避免传统算法中心点搜索的随机性;同时改进准则函数,提高聚类评价体系,从而提高聚类质量。本文得到了理想的实验结果,验证了本文算法的可行性和有效性。

参考文献:

- [1] HAN J, KAMBER M, PEI J. Data mining: concepts and techniques[M]. FAN M, translated. Beijing: China Machine Press, 2012: 293 - 297. (HAN J, KAMBER M, PEI J. 数据挖掘: 概念与技术[M]. 范明, 译. 北京: 机械工业出版社, 2012: 293 - 297.)
- [2] XIA N, SU Y, QIN X. Efficient K-medoids clustering algorithm[J]. Application Research of Computers, 2010, 27(12): 4517 - 4519. (夏宁霞, 苏一丹, 覃希. 一种高效的 K-medoids 聚类算法[J]. 计算机应用研究, 2010, 27(12): 4517 - 4519.)
- [3] PARDESHI B, TOSHNIWAL D. Improved K-medoids clustering based on cluster validity index and object density[C]// Proceedings of the 2nd IEEE International Advance Computing Conference. Piscataway: IEEE, 2010: 379 - 384.
- [4] ADRIANO A P, MARIO A N, CAETANO T J. Using pivots to speed-up K-medoids clustering[J]. Journal of Information and Data Management, 2011, 2(2): 221 - 236.
- [5] MA Q, XIE J. New K-medoids clustering algorithm based on granular computing[J]. Journal of Computer Applications, 2012, 32(7): 1973 - 1977. (马箐, 谢娟英. 基于粒计算的 K-medoids 聚类算法[J]. 计算机应用, 2012, 32(7): 1973 - 1977.)
- [6] NTOUTSI I, ZIMEK A, PALPANAS T, et al. Density-based projected clustering over high dimensional data streams[C]// Proceedings of the 2012 SIAM International Conference on Data Mining. Piscataway: IEEE, 2012, 12: 987 - 998.
- [7] YU Y, WANG Q, KUANG J, et al. An on-line density-based clustering algorithm for spatial data stream[J]. Acta Automatica Sinica, 2012, 38(6): 1051 - 1058. (于彦伟, 王沁, 邝俊, 等. 一种基于密度的空间数据流在线聚类算法[J]. 自动化学报, 2012, 38(6): 1051 - 1058.)
- [8] PARK H S, JUN C H. A simple and fast algorithm for K-medoids clustering[J]. Expert Systems with Applications, 2008, 36(2): 3336 - 3341.
- [9] PAN C, LUO K. Improved K-medoids clustering algorithm based on improved granular computing[J]. Journal of Computer Applications, 2014, 34(7): 1997 - 2000. (潘楚, 罗可. 基于改进粒计算的 K-medoids 聚类算法[J]. 计算机应用, 2014, 34(7): 1997 - 2000.)
- [10] LIN T Y. Granular computing: from rough sets and neighborhood systems to information granulation and computing with words[C]// Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing. Dordrecht: Kluwer Academic Publishers, 1997: 1602 - 1606.
- [11] WANG G, ZHANG Q, HU J. An overview of granular computing[J]. CAAI Transactions on Intelligent Systems, 2007, 2(6): 8 - 26. (王国胤, 张清华, 胡军. 粒计算研究综述[J]. 智能系统学报, 2007, 2(6): 8 - 26.)
- [12] XU L, DING S. Research on granularity clustering algorithms[J]. Computer Science, 2011, 38(8): 25 - 28. (徐丽, 丁世飞. 粒度聚类算法研究[J]. 计算机科学, 2011, 38(8): 25 - 28.)