相对邻域与剪枝策略优化的密度峰值聚类算法

纪 霞 [1, 2] 姚 晟 [1, 2] 赵 鹏 [1, 2]

摘 要 针对 *Science* 发表的密度峰值聚类 (Density peaks clustering, DPC) 算法及其改进算法效率不高的缺陷, 提出一种相对邻域和剪枝策略优化的密度峰值聚类 (Relative neighborhood and pruning strategy optimized DPC, RP-DPC) 算法. DPC 聚类算法主要有两个阶段: 聚类中心点的确定和非聚类中心点样本的类簇分配, 并且时间复杂度集中在第 1 个阶段, 因此 RP-DPC 算法针对该阶段做出改进研究. RP-DPC 算法去掉了 DPC 算法预先计算距离矩阵的步骤, 首先利用相对距离将样本映射到相对邻域中, 再从相对邻域来计算各样本的密度, 从而缩小各样本距离计算及密度统计的范围; 然后在计算各样本的 δ 值时加入剪枝策略, 将大量被剪枝样本 δ 值的计算范围从样本集缩小至邻域以内, 极大地提高了算法的效率. 理论分析和在人工数据集及 UCI 数据集的对比实验均表明, 与 DPC 算法及其改进算法相比, RP-DPC 算法在保证聚类质量的同时可以实现有效的时间性能提升.

关键词 聚类算法, 密度峰值, 相对邻域, 剪枝策略

引用格式 纪霞, 姚晟, 赵鹏. 相对邻域与剪枝策略优化的密度峰值聚类算法. 自动化学报, 2020, **46**(3): 562−575

DOI 10.16383/j.aas.c170612

Relative Neighborhood and Pruning Strategy Optimized Density Peaks Clustering Algorithm

JI Xia[1, 2] YAO Sheng[1, 2] ZHAO Peng[1, 2]

Abstract In order to overcome the low efficiency defect of density peaks clustering (DPC) algorithm published in *Science* and its improvement algorithms, a new relative neighborhood and pruning strategy optimized DPC (RP-DPC) algorithm is proposed in this paper. There are two main phases in DPC: determination of cluster centers and cluster assignation for remaining samples. The time complexity of DPC is determined by the first phase, so the improvements for the determination of cluster centers are proposed in this paper. Firstly, the RP-DPC algorithm maps samples to their relative neighborhoods, then computes the local density of every sample on the basis of relative neighborhood. This method shrinks the range of distance computing and density counting of every sample, thus avoiding a lot of unnecessary distance calculations. Secondly, the pruning strategy is led into the δ value computing of every sample, which restricts the δ computing of massive pruned samples to within their own neighborhoods, so as to greatly improve the efficiency. We demonstrate that: our RP-DPC algorithm can improve the time performance significantly on the basis of having same clustering quality compared with the DPC algorithm and its improvement algorithms through the theory analysis and the experiments on several popular test cases that include both synthetic and real-world data sets from the UCI machine learning repository.

Key words Clustering algorithm, density peaks, relative neighborhood, pruning strategy

Citation Ji Xia, Yao Sheng, Zhao Peng. Relative neighborhood and pruning strategy optimized density peaks clustering algorithm. *Acta Automatica Sinica*, 2020, **46**(3): 562−575

聚类分析简称聚类, 是根据样本间的相似性将样本集划分成合理类簇的过程, 聚类结果使得同一类簇中的样本具有较高的相似性, 而不同类簇之间的样本相似性较低[1−3]. 聚类是数据挖掘中的基本技术, 能够从数据中发现潜藏的知识和模式, 已广泛应用于社会网络分析、图像模式识别、智能商务等

收稿日期 2017-11-06 录用日期 2018-08-28
Manuscript received November 6, 2017; accepted August 28, 2018
国家自然科学基金 (61602004, 61672034), 安徽省重点研究与开发计划 (1804d8020309), 安徽省自然科学基金 (1708085MF160, 1908085MF188), 安徽省高等学校自然科学研究重点项目 (KJ2016A041, KJ2017A011), 安徽大学信息保障技术协同创新中心公开招标课题 (ADXXBZ201605) 资助
Supported by National Natural Science Foundation of China (61402004, 61672034), Key Research and Development Program of Anhui Province (1804d8020309), Natural Science Foundation of Anhui Province (1708085MF160, 1908085MF188), Natural Science Foundation of Anhui Higher Education Institutions (KJ2016A041, KJ2017A011), and Public Bidding Project of Co-Innovation Center for Information Supply and Assurance Technology of Anhui University (ADXXBZ201605)
本文责任编委 辛景民
Recommended by Associate Editor XIN Jing-Min

1. 安徽大学计算机科学与技术学院 合肥 230601 2. 安徽大学计算智能与信号处理教育部重点实验室 合肥 230601
1. School of Computer Science and Technology, Anhui University, Hefei 230601 2. Key Laboratory of Intelligent Computing and Signal Processing of the Ministry of Education, Anhui University, Hefei 230601

众多领域.大数据背景下, 数据的海量、多样和复杂使得具有自动理解、处理和概括数据的高效聚类算法研究迫在眉睫[4].

聚类算法大致可以分为划分式聚类方法[5−6]、层次聚类方法[7]、基于网格的聚类方法[8] 和基于密度的聚类方法[9] 等. 其中, 基于密度的聚类方法可以发现任意形状的类簇, 对噪音数据不敏感, 且聚类时不需要事先知道类簇的个数, 是数据挖掘技术中广泛使用的一类方法.

快速搜索和发现样本密度峰值聚类 (Density peaks clustering, DPC) 算法是 Rodriguez 等[10] 近年在 *Science* 发表的一种新型聚类算法. 与其他聚类算法不同, DPC 算法能自动确定类簇数和类簇中心点, 并进行高效的非中心点样本分配和离群点剔除, 因而吸引了众多学者对它进行深入研究. Wang 等[11] 针对 DPC 算法对输入参数 dc (密度计算的截断距离) 敏感, 并且没有有效的设定准则的问题, 在采用核函数计算密度的情况下, 结合数据域的概念提出了一种自动计算 dc 的方法. 同时, 强调在大数据量情况下算法效率是 DPC 算法亟待研究的关键问题. Zhang 等[12] 针对 DPC 算法不能解决一个类簇中多密度峰值或者无密度峰值的情况, 提出一种扩展的 E_CFSFDP (Extended clustering by fast search and find of density peaks) 算法, 在 DPC 算法聚类完成之后, 多执行一个子类的合并步骤. 该扩展方法在无密度峰值的数据集上取得了更好的实验效果, 但是时间开销巨大, 作者也将降低时间消耗作为下一步研究的重点. 谢娟英等[13] 针对 DPC 算法中截断距离 dc 对聚类结果影响较大和样本分配策略可能会导致的 “多米诺骨牌” 效应的问题, 分别提出了 K 近邻优化和模糊加权 K 近邻优化的密度聚类方法 KNN-DPC (K-nearest neighbors optimized density peaks clustering) 和 FKNN-DPC (Fussy weighted KNN-DPC)[14], 并通过实验证明了改进方法的有效性. 但是并未给出关于引入参数 K 的有效设定方法, 同时由于额外 K 近邻的查找和复杂的类簇分配策略的引入, 使得 KNN-DPC 和 FKNN-DPC 算法的时间复杂性都要远高于 DPC, 极大地影响了算法的实用性. 因为 DPC 算法首先需要计算数据集中任意两个样本间的欧氏距离, 其时间复杂度为 $\mathrm{O}(m \times n^2)$ (m 为样本特征个数, n 为数据集样本个数), 当处理海量高维数据时,大量的高维欧氏距离计算会带来高额的时间开销, 严重影响了算法的实用性, 所以对 DPC 算法的效率改进展开研究具有重要的应用价值. 巩树凤等[15] 考虑了 DPC 算法效率不高的问题, 给出了分布式环境下的密度中心聚类算法 SDDPC (Simple distributed density peaks clustering), 并且结合 Voronoi 图提出了优化的 EDDPC (Efficient distributed density peaks clustering) 算法, 提高了分布式环境下 DPC 算法的效率, 但并没有涉及 DPC 算法本身的研究与改进.

针对 DPC 算法和现有改进算法在效率方面的不足, 本文提出一种基于相对邻域和剪枝策略的密度峰值快速搜索聚类 (Relative neighborhood and pruning strategy optimized density peaks clustering, RP-DPC) 算法. RP-DPC 的主要贡献包括: 1) 改变原 DPC 算法的流程, 不再预先计算样本两两之间的距离, 改为在聚类过程中计算必要的样本间距离, 从而避免了大量冗余距离的计算; 2) 借助双基准点映射的相对邻域来大致衡量样本之间的亲疏关系, 从而只需要对相对 “亲密” 的样本进行距离计算和密度统计; 3) 在计算样本的斥群值 (与更高密度样本之间距离的最小值[15]) 时加入剪枝策略, 极大地缩小被剪枝样本的斥群值查找范围, 从而加快了算法的效率; 4) 理论分析和在多个数据集上的对比实验均表明, RP-DPC 算法具有和 DPC 算法同样的聚类效果, 时间性能却大大优于已有的 DPC 算法及其改进算法.

1 DPC 算法

快速搜索和发现样本密度峰值的聚类算法 DPC[10] 能够自动发现数据集样本的类簇中心, 实现任意形状数据集样本的聚类. 该算法的设计基于以下假设: 1) 聚类中心点的密度较大, 被密度不超过它的样本点包围; 2) 聚类中心点与其他密度更大的点 (另一个类簇的中心点) 的距离相对较远. 为了找到同时满足上述条件的类簇中心, DPC 算法引入了样本 x_i 的密度 ρ_i 和斥群值 δ_i, 其定义如式 (1) 和式 (2) 所示:

$$\rho_i = \sum_j \chi(d(x_i, x_j) - d_c) \tag{1}$$

其中, $\chi(x)$ 是一个函数, 当 $x < 0$ 时, $\chi(x) = 1$, 否则 $\chi(x) = 0$; $d(x_i, x_j)$ 为样本 x_i 和 x_j 间的欧氏距离, d_c 为截断距离, 即样本 x_i 的密度 ρ_i 是与样本 x_i 的距离小于 d_c 的点的个数.

$$\delta_i = \min(d(x_i, x_j) | \rho_i > \rho_j) \tag{2}$$

斥群值 δ_i 代表密度比样本 x_i 大且距离样本 x_i 最近的点的距离. 当某个点 x_i 的密度是样本集中最大的, 那么设定点 x_i 的 $\delta_i = d_{\max}$ ($d_{\max}$ 为已有样本间距离的最大值).

DPC 算法将每个数据点的ρ值和δ值表示在一个 2 维决策图 (Decision graph) 上. 用户根据决策图的分布情况, 选定聚类中心点, 接下来再将所有剩下的点分配到比其密度更高且最近的样本点所属的类簇中. DPC 算法构造样本距离相对于样本密度的 2 维决策图, 能够展示任意维度数据集的类簇中心点, 实现对任意维度数据的可视化聚类分析.

文献 [10] 使用大量实验证明了 DPC 算法在聚类质量上的优良性能,但是该算法也存在一些不足, 包括质量和效率两个方面, 本文主要关注其效率方面. 对样本规模为n, 属性个数为m的数据集, DPC 算法中时间复杂度主要来自 3 部分: 1) 计算两两样本间的距离, 其时间复杂度为$\mathrm{O}(m \times n^2)$; 2) 计算每个样本的密度ρ, 其时间复杂度为$\mathrm{O}(n^2)$; 3) 计算每个样本的δ值, 其时间复杂度也是$\mathrm{O}(n^2)$. 所以 DPC 算法总的时间度杂度为$\mathrm{O}(m \times n^2)$. 当处理海量高维数据时, 算法的实用性受到了严重的影响.

进一步分析 DPC 算法的时间复杂度, 其最高复杂度来自于样本间距离矩阵的计算, 后续的样本ρ值和δ值都是在此距离矩阵上展开. 但仔细分析算法中ρ值和δ值的含义, 可以发现每个样本的ρ值和δ值计算并不需要全部用到该样本与其他$n-1$个样本的距离, 因此 DPC 算法预先计算的距离矩阵存在大量冗余距离. 本文提出的改进方法是将样本间距离的计算过程后移, 与样本的ρ值和δ值计算过程结合在一起. 对在ρ值和δ值计算中明确需要用到的样本间距离才进行计算, 从而避免大量冗余距离的计算, 提高算法的时间效率.

2 RP-DPC 算法及其分析

本文从缩减ρ值计算时样本间距离计算量和δ值计算时距离比较范围两个方面对原 DPC 算法进行改进, 提出了一种基于相对邻域和剪枝策略优化的密度峰值聚类算法: RP-DPC 算法. 算法主要思想是: 首先, 去掉 DPC 算法中复杂度最高的距离矩阵预计算步骤; 其次, 在ρ值计算中, 将相对距离的思想引入到 DPC 算法, 通过相对邻域来刻画对象之间的相对位置关系, 以此缩小样本ρ值计算所需要的距离计算量; 然后, 在计算各样本的δ值时, 加入剪枝策略, 将被剪枝样本点的δ值计算范围从样本集缩小到其自身邻域以内, 因此剪枝策略可以进一步提高算法的时间性能. 本节内容安排如下: 第 2.1 节对 RP-DPC 算法做一个整体描述, 第 2.2 节和第 2.3 节将分别给出 RP-DPC 算法对样本ρ值和δ值的改进计算方法, 最后, 第 2.4 节给出 RP-DPC 算法复杂度分析.

2.1 RP-DPC 算法描述

DPC 算法与 RP-DPC 算法的流程图如图 1 所示, 两个算法的不同之处已用虚线框标出. 由图 1 可以清楚地看出, 与 DPC 算法相比, RP-DPC 算法去掉了距离矩阵的计算步骤, 但在样本的ρ值和δ值计算过程中增加了相对邻域映射模型和剪枝策略. RP-DPC 算法的具体描述见算法 1.

算法 1. RP-DPC 算法

输入. 数据集$S = \{x_i\}_{i=1}^n$, 截断距离d_c.

输出. 聚类结果C.

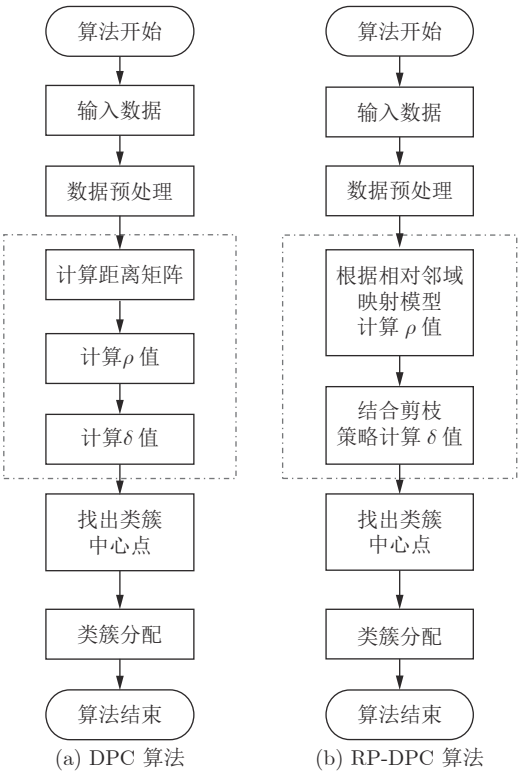
步骤 1. 数据预处理: 数据归一化;

步骤 2. 采用相对邻域映射模型计算每个样本的密度ρ_i;

步骤 3. 采用剪枝策略计算样本的斥群值δ_i;

步骤 4. 按照 DPC 中的操作从由ρ和δ 构成的决策图中选出类簇中心点集合CI;

步骤 5. 按照 DPC 的方法对非类簇中心点的样本进行类簇分配.



(a) DPC 算法 (a) DPC algorithm (b) RP-DPC 算法 (b) RP-DPC algorithm

图 1 算法流程图

Fig. 1 The algorithm flowchart

2.2 相对邻域映射模型

在密度ρ值计算中, 原 DPC 算法首先需要计算每个样本与其他$n-1$个样本的距离, 找到该样本

的邻域, 然后再统计其邻域内的样本数量得到密度, 这一步骤的时间复杂度为$O(n^2)$. 对于任意样本x_i来说, 计算其密度需要知道x_i与其他$n-1$个样本的精确距离, 但若是将某个样本从x_i邻域中排除则只需要大致的距离范围就可以了. 基于上述分析, 本文采用排除法. 通过样本间相对位置获取样本间大致的距离范围来排除大部分不可能属于彼此邻域的样本, 从而大大降低密度计算的复杂度. 为了实现这一目标, 受文献 [16] 的启发, 本文将相对距离的思想引入到 RP-DPC 算法.

文献 [16] 在计算样本邻域时, 根据各样本到基准点的距离将每一个样本映射到不同的桶中, 然后在计算每个样本的邻域时最多只要从相邻 3 个桶的样本中寻找, 而不是从整个样本集, 从而提高算法效率. 具体做法如下:

设给定样本集$S=\{x_i\}_{i=1}^n$, 每个样本有m维特征, 第i个样本的第j个特征的值记为v_{ij}, 截断距离记为d_c.

文献 [16] 中对基准点和映射函数的定义如式 (3) 和 (4) 所示.

$$x_0=(v_{0j})_{j=1}^m \tag{3}$$

其中, $v_{0j}=\min((v_{ij})_{i=1}^n)$. 基准点$x_0$由$S$中所有样本在$m$个特征上的最小值所构成.

根据各样本到基准点的距离将样本集S中的样本映射到不同的集合, 记为$B_0,\cdots,B_p$, 映射函数定义为

$$B_p=\left\{x_i\in S\left|\left\lceil\frac{d(x_i,x_0)}{d_c}\right\rceil=p\right.\right\} \tag{4}$$

文献 [16] 的方法对于靠近基准点的样本邻域计算效率提升较为明显, 但是对于距离基准点较远的样本, 计算效率提升有限. 原因在于随着映射距离的增大, 距离基准点越远的桶包含的样本数量越多, 样本的分布也越分散. 针对这个问题, 本文 RP-DPC 算法改进了文献 [16] 的方法, 采用双基准点映射模型. 在第 1 个基准点x_0的基础上, 新增基准点x_{n+1}.

定义 1. 样本集$S=\{x_i\}_{i=1}^n$, 第i个样本的第j个属性值为v_{ij}, 构造基准点x_{n+1}如下:

$$x_{n+1}=\begin{cases}(v_{(n+1)j})_{j=1}^{\lfloor\frac{m}{2}\rfloor}, & v_{(n+1)j}=\max((v_{ij})_{i=1}^n)\\ (v_{(n+1)j})_{j=\lfloor\frac{m}{2}\rfloor+1}^{m}, & v_{(n+1)j}=\min((v_{ij})_{i=1}^n)\end{cases} \tag{5}$$

基准点x_{n+1}也包含m个特征, 它的前一半特征值是由S中所有样本在这些特征上的最大值构成, 而后一半特征值是由S中全部样本在这些特征上的最小值构成.

定义 2. 根据式 (4) 的映射函数, 由x_0基准点映射得到的桶记为$B(x_0)_0,\cdots,B(x_0)_p$, 由基准点x_{n+1}映射得到的桶记为$B(x_{n+1})_0,\cdots,B(x_{n+1})_q$. $x_i\in S$, 如果$x_i\in B(x_0)_k$, $x_i\in B(x_{n+1})_t$, 则样本x_i的相对邻域定义为

$$RN_i=B(x_0)_k\cap B(x_{n+1})_t \tag{6}$$

在密度计算过程中, 还需要用到邻域的定义, 这里一并给出.

定义 3. 样本x_i的邻域定义为

$$N_i=x_j|d(x_i,x_j)\le d_c \tag{7}$$

显然, $\rho_i=|N_i|$. 密度ρ_i记录了与样本i距离小于截断距离的样本个数, 而邻域则记录了这些样本的编号.

下面给出相对邻域映射模型的相关性质, 这些性质说明了双基准点模型是如何在样本密度 值的计算中提升算法执行效率的.

定理 1. $\forall x_i$, $x_j\in S$, 已知$d(x_i,x_j)\le d_c$. 假设x_i和x_j的相对邻域分别为$RN_i=B(x_0)_k\cap B(x_{n+1})_l$, $RN_j=B(x_0)_s\cap B(x_{n+1})_t$, 则 $|k-s|\le 1$, $|l-t|\le 1$.

证明. 反证法. 先证$|k-s|\le 1$.

现假设 $|k-s|>1$, 由式 (4) 可以得到, $|\lceil d(x_i,x_0)/d_c\rceil-\lceil d(x_j,x_0)/d_c\rceil|>1$, 进一步简化得到$|d(x_i,x_0)-d(x_j,x_0)|>d_c$. 下面分两种情形进行证明.

1) 当x_0、x_i和x_j三点可以组成三角形时, 三角形的三条边分别是$d(x_i,x_0)$、$d(x_j,x_0)$和$d(x_i,x_j)$, 由三角形的边长不等式可以得到 $|d(x_i,x_0)-d(x_j,x_0)|<d(x_i,x_j)$, 又由假设条件 $|d(x_i,x_0)-d(x_j,x_0)|>d_c$, 两式联解可以得到$d_c<d(x_i,x_j)$, 这与已知条件$d(x_i,x_j)\le d_c$矛盾.

2) 当x_0、x_i和x_j三点不能组成三角形时, 即三点在一条直线上, 由点x_0的构造方法可知x_i和x_j都在x_0的右上一侧, 此时可以得到下面的等式 $|d(x_i,x_0)-d(x_j,x_0)|=d(x_i,x_j)$, 同样将假设条件 $|d(x_i,x_0)-d(x_j,x_0)|>d_c$代入可以得到, $d_c<d(x_i,x_j)$, 这与已知条件$d(x_i,x_j)\le d_c$矛盾.

综合上述两种情形可知, 假设不成立, 也即 $|k-s|\le 1$成立.

$|l-t|\le 1$的证明过程类似,这里不再赘述. □

定理 1 说明若两个样本x_i和x_j之间满足邻域关系 (即$d(x_i,x_j)\le d_c$), 则它们的相对邻域是相邻的. 定理 1 保证了用相对邻域映射模型能够保持样本之间的邻域关系. 基于定理 1, 可以得到推论 1. 推论 1 给出样本间的邻域关系在双基准点相对邻域

模型中的保持范围.

推论 1. 已知样本集 S 由 x_0 基准点映射得到的桶记为 $B(x_0)_0,\cdots,B(x_0)_p$，由基准点 x_{n+1} 映射得到的桶记为 $B(x_{n+1})_0,\cdots,B(x_{n+1})_q$. $\forall x_i$，$x_j\in S$，$RN_i=B(x_0)_k\cap B(x_{n+1})_l$, 如果 $d(x_i,x_j)\le d_c$, 则下列性质成立:

1) 当 $k\in\{1,2,\cdots,p-1\}$, $l\in\{1,2,\cdots,q-1\}$ 时,

$$RN_j\subseteq(B(x_0)_{k-1}\cup B(x_0)_k\cup B(x_0)_{k+1})\cap (B(x_{n+1})_{l-1}\cup B(x_{n+1})_l\cup B(x_{n+1})_{l+1})$$

2) 当 $k=0, l\in\{1,2,\cdots,q-1\}$ 时,

$$RN_j\subseteq(B(x_0)_k\cup B(x_0)_{k+1})\cap(B(x_{n+1})_{l-1}\cup B(x_{n+1})_l\cup B(x_{n+1})_{l+1})$$

3) 当 $k\in\{1,2,\cdots,q-1\}$，$l=0$ 时,

$$RN_j\subseteq(B(x_0)_{k-1}\cup B(x_0)_k\cup B(x_0)_{k+1})\cap(B(x_{n+1})_l\cup B(x_{n+1})_{l+1})$$

4) 当 $k=0$，$l=0$ 时,

$$RN_j\subseteq(B(x_0)_k\cup B(x_0)_{k+1})\cap(B(x_{n+1})_l\cup B(x_{n+1})_{l+1})$$

5) 当 $k=p$，$l\in\{1,2,\cdots,q-1\}$ 时,

$$RN_j\subseteq(B(x_0)_{k-1}\cup B(x_0)_k)\cap(B(x_{n+1})_{l-1}\cup B(x_{n+1})_l\cup B(x_{n+1})_{l+1})$$

6) 当 $k\in\{1,2,\cdots,q-1\}$，$l=q$ 时,

$$RN_j\subseteq(B(x_0)_{k-1}\cup B(x_0)_k\cup B(x_0)_{k+1})\cap(B(x_{n+1})_{l-1}\cup B(x_{n+1})_l)$$

7) 当 $k=p$，$l=q$ 时,

$$RN_j\subseteq(B(x_0)_{k-1}\cup B(x_0)_k)\cap(B(x_{n+1})_{l-1}\cup B(x_{n+1})_l)$$

证明. 性质 1) 的证明.

现欲证明

$$RN_j\subseteq(B(x_0)_{k-1}\cup B(x_0)_{k+1})\cap(B(x_{n+1})_{l-1}\cup B(x_{n+1})_l\cup B(x_{n+1})_{l+1})$$

即 $\forall x\in RN_jB$ 都存在

$$x_j\in B(x_0)_{k-1}\cup B(x_0)_k\cup(x_0)_{k+1}\text{且}$$

$$x_j\in B(x_{n+1})_{l-1}\cup B(x_{n+1})_l\cup B(x_{n+1})_{l+1}$$

先来证明 $x\in B(x_0)_{k-1}\cup B(x_0)_k\cup B(x_0)_{k+1}$.

根据式 (4) 和式 (6)，由 $RN_j=B(x_0)_k\cap B(x_{n+1})_l)$ 可得 $\lceil d(x_i,x_0)/d_c\rceil\le k$. 这里用反证法. 假设 $x_j\notin B(x_0)_{k-1}\cup B(x_0)_k\cup B(x_0)_{k+1}$，如 $x_j\in B(x_0)_{k+2}$，则根据式 (4) 可得 $\lceil d(x_j,x_0)/d_c\rceil\le k+2$, 将两个不等式联合起来可以得到 $d(x_j,x_0)/d_c-d(x_i,x_0)/d_c>1$，化简可得 $d(x_j,x_0)-d(x_i,x_0)>d_c$, 又已知三角形不等式 $|d(x_j,x_0)-d(x_i,x_0)|<d(x_i,x_j)$, 联合两个不等式可以得到 $d(x_i,x_j)>d_c$，与已知条件矛盾, 故假设不成立.

$x_j\in B(x_{n+1})_{l-1}\cup B(x_{n+1})_l\cup B(x_{n+1})_{l+1}$ 的证明过程相同.

当 $x_j\in B(x_0)_{k-2}$ 和 $x_j\in B(x_0)_{l-2}$, 证明方法是类似的.

同样, 当 $x_j\in B(x_0)_{k\pm r}$ 和 $x_j\in B(x_{n+1})_{l\pm r}$，$r>2$ 时, 证明同上. 综合以上, 性质 1) 得证.

性质 2) ~ 7) 的证明与性质 1) 类似, 这里不再赘述. □

推论 1 给出了样本密度计算时的样本间距离的计算范围. 根据基准点 x_0 和 x_{n+1}，将整个样本集 S 映射到这样一个方形区域, 具体如图 2 所示. S 中的任何一个样本 x_i 根据其到基准点的距离都可以被映射到一个相应的相对邻域 RN_i 中, 样本 x_i 的邻域可以被看作一个以 x_i 为球心, 以截断距离 d_c 为半径的超球体. 根据 x_i 在相对邻域中的位置, 样本密度的计算范围就从整个样本集缩小到了如图 2 中浅色样本区域, 从而提高了算法的时间性能. 相对邻域映射模型下样本的密度计算过程详细描述如算法 2.

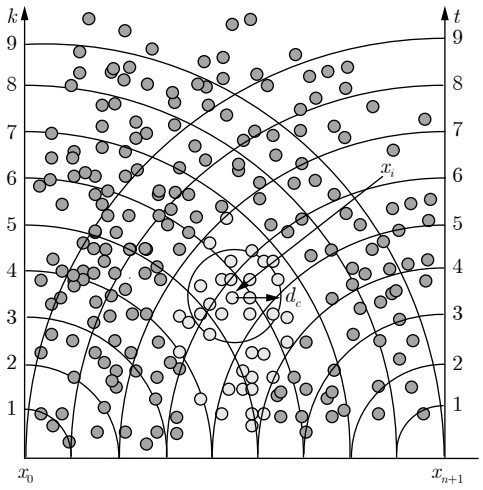


图 2 相对邻域映射模型

Fig. 2 Relative neighbor mapping model

算法 2. 密度计算过程 $R-\rho(S,d_c,)$ 算法

输入. 样本集 $S=\{x_i\}_{i=1}^n$，截断距离 d_c.

输出. 各样本的密度 $\rho=\{\rho_i\}_{i=1}^n$，距离矩阵 $D=(d_{ij})_{n\times n}$，各样本的邻域 $N=\{N_i\}_{i=1}^n$.

步骤 1. 初始化 $\{\rho_i=0\}_{i=1}^n$，$(d_{ij}=-1)_{n\times n}$，$\{N_i=\emptyset\}_{i=1}^n$;

步骤 2. 根据式 (3) 构造基准点 x_0，根据定义 1 构造基准点 x_{n+1};

步骤 3. 根据式 (4) 和定义 2 计算各样本的相

对邻域;

步骤 4. 对样本集中每一个样本x_i执行下面的操作:

1) 根据推论计算x_i样本间距离计算范围;

2) 对x_i样本间距离计算范围内的每一个样本x_j执行以下操作:

a) 计算$d(x_i, x_j)$, 并将该距离填入距离矩阵中d_{ij};

b) 如果$d(x_i, x_j) \le d_c$:

i) $\rho_i = \rho_i + 1$; $\rho_j = \rho_j + 1$;

ii) $N_i = N_i \cup \{x_j\}$; $N_j = N_j \cup \{x_i\}$.

算法 2 的复杂度分析如下: 步骤$1 \sim 3$的时间复杂度都是$\mathrm{O}(n)$. 假设n个样本在x_0基准点维度上平均映射到p个相对邻域, 则x_0基准点维度上每个样本需要比较的样本规模为$3n/p$. 在x_{n+1}基准点维度上平均映射到q个相对邻域, 则每个样本需要比较的样本规模为$3n/q$. 则步骤 4 的复杂度为$\max(\mathrm{O}(mn^2/p), (mn^2/q))$. 因为算法 2 主要时间复杂度集中在步骤 4, 所以算法 2 的复杂度为$\max(\mathrm{O}(mn^2/p), (mn^2/q))$, p和q是直接与样本规模n相关的. 通常n越大, p和q也越大.

2.3　δ 值的剪枝计算

样本x_i的邻域可以被看作一个以x_i为球心、以截断距离d_c为半径的超球体. 由δ值的含义可以得到以下结论: 对x_i来说, 如果其邻域内存在一个样本x_j密度大于x_i, 则在整个样本集S中满足密度大于x_i且距离x_i最近的样本一定是x_j, 也即可以得到$\delta_i = d_{ij}$且$\delta_i < d_c$. 若样本x_i的邻域内存在多个样本密度大于x_i, 则计算δ_i时也只需要从这多个样本中寻找距离x_i最近的, 而不需要考虑x_i邻域之外的样本. 基于以上分析, 本文在δ值的计算中引入d_c剪枝策略.

定理 2 (d_c 剪枝). 对于任一个样本点, 如果其邻域内存在密度更大的样本点, 就知道了该样本点δ值的上确界β, 且$\beta \le d_c$, 则该样本点被剪枝.

证明. $\forall x_i$, 设$x_j \in N_i$, 且$\rho_j > \rho_i$. 由定义 1 可知$d_{ij} < d_c$. 下面分两种情况作进一步证明:

1) 若样本点x_i邻域内还存在另一个密度更大的样本点x_k, 且$d_{ik} < d_{ij}$. 则由δ值的含义可知$\delta_i \le d_{ik}$. 因此$\delta_i \le d_{ij}$. 存在多个高密度点的情况证明类似, 此处不再重复.

2) 若样本点x_i邻域内除了x_j之外不存在另外密度更大的样本点, 即其他的高密度点都在x_i的邻域之外. $\forall x_k \notin N_i$, 即$d_i k > d_c$, 且$\rho_k > \rho_i$. 由式 (2) 可以知道$\delta_i = d_{ij}$.

综合上述两种情况可以得到$\delta_i \le d_{ij}$. 又已知$d_{ij} \le d_c$, 两式联立可得$\delta_i \le d_c$. □

定理 2 说明无论一个样本的更高密度样本分布如何, 只要其邻域内存在一个更高密度的样本, 则其δ值的计算就可以局限在邻域范围以内. 由定理 2 可以进一步得出推论 2.

推论 2. 任意一个样本不被剪枝, 当且仅当它在自己邻域范围密度最高.

证明. 1) 充分性. $\forall x_i \in S$, 假设x_i是其邻域N_i中密度最高的样本点, 则密度高于x_i的样本一定处于N_i之外. 假设离x_i最近且密度高于x_i的样本点为x_k, 那么可得$\delta_i = d(x_k, x_i) > d_c$, 则$x_i$不被剪枝;

2) 必要性. $\forall x_i \in S$, 当x_i不被剪枝, 根据定理 2 可知$\delta_i > d_c$, 即密度高于x_i的最近的样本点在x_i的邻域之外, 也就是说, x_i的邻域内所有样本密度都低于x_i, x_i在自己邻域范围密度最高. □

结合d_c剪枝策略的δ值计算过程详细描述如算法 3.

算法 3. 斥群值计算过程$P - \delta_i(\rho, N, D)$

输入. 各样本的密度$\rho = \{\rho_i\}_{i=1}^n$, 距离矩阵$D = (d_{ij})_{n \times n}$, 各样本的邻域$N = \{N_i\}_{i=1}^n$.

输出. 各样本的斥群值$\delta = \{\delta_i\}_{i=1}^n$.

步骤 1. 初始化$\delta_i = \{d_{\max}\}_{i=1}^n$;

步骤 2. 对样本集按照密度降序排序;

步骤 3. 从前往后对样本集中的每一个样本x_i执行下面的操作:

1) 若样本x_i未被剪枝, 则执行下面的操作:

a) $j = 1, \cdots, i-1$, 如果$d_{ij} = -1$, 则计算$d(x_i, x_j)$并赋值给d_{ij};

b) 找到样本x_i与这$i-1$个更高密度样本之间距离的最小值, 赋值给δ_i;

2) $\forall x_j \in N_i$, 如果$\rho_j < \rho_i$, 执行以下操作:

a) 对x_j剪枝;

b) 如果$d(x_i, x_j) < \delta_i$, 则$\delta_i \leftarrow d(x_i, x_j)$.

如果样本点被剪枝, 即其δ值的计算仅限于邻域范围之内, 由算法 2 可知需要用到的样本间距离都已经计算过了, 而若样本点未被剪枝, 即该样本的δ值计算需要在密度高于该点的样本间去查找, 则有可能需要用到的距离尚未被计算, 故算法 3 步骤 3 中的 1-a) 步进行了相应距离的计算.由此可见, 本文将 DPC 算法中作为基础条件的距离矩阵计算取消了, 转变为在算法 2 和算法 3 中对必要距离的计算, 以此避免了大量冗余距离的计算, 算法时间效率得到有效提升.

算法 3 没有增加新的空间复杂度. 时间复杂度集中在步骤 3. 步骤 3 的操作 1) 是对未被剪枝的样

本计算δ值; 操作 2) 是用来剪枝. 未被剪枝的样本δ值计算过程与文献 [10] 相同, 被剪枝样本则只需从邻域内比自己密度大的样本中找出距离最近的样本. 用$\overline{\rho}$表示平均密度, 假设k个样本被剪枝, 总的计算量可以估算为$O(n(n-k)+\overline{\rho}k)$. k越大, $O(n(n-k)+\overline{\rho}k)$越小.

我们可以进一步对被剪枝样本数量k作一个简单估算. 根据推论 2,只有在自己邻域范围密度最高的样本不会被剪枝. 假设样本集简单地划分为$n/\overline{\rho}$个互不相交的邻域超球体, 每个邻域超球体中只有密度最高的样本点未被剪枝, 则共有$n/\overline{\rho}$个样本未被剪枝. 因此总的计算量为$O(n^2/\overline{\rho}+\overline{\rho}(n-n/\overline{\rho}))$. 按照文献 [10] 的截断距离设定方案,一般邻域大小为整个样本集规模的1 % ~ 2 %. 这里取 1.5 % 计算, 最终总的计算量为$O(3n^2/200)$. 原 DPC 算法在计算斥群值时复杂度为$O(n^2)$, 可见加入剪枝策略以后, 斥群值δ 的计算效率有极大提升.

2.4 RP-DPC 算法分析

本文 RP-DPC 算法引入相对邻域映射和剪枝策略来计算样本密度ρ和斥群值δ. 对样本规模为n的数据集, DPC 算法存储距离矩阵的空间复杂度为$O(n^2)$, 这也是该算法主要的空间复杂度. RP-DPC 算法增加了存储每个样本的邻域空间, 但增加的空间复杂度不超过$O(\overline{\rho}n)$, $\overline{\rho}$表示平均密度. 因此 RP-DPC 算法的空间复杂度与 DPC 算法相同.

RP-DPC 算法中时间复杂度集中在步骤 2 密度ρ计算和步骤 3 斥群值δ计算. 步骤 2 和步骤 3 的复杂度分别为$\max(O(mn^2/p), O(mn^2/q))$和$O(n(n-k)+\overline{\rho}k)$, 因此 RP_DPC 算法的时间复杂度为$\max(\max(O(mn^2/p), O(mn^2/q)), O(n(n-k)+\overline{\rho}k))$. 而 DPC 算法总的时间复杂度为$O(mn^2)$, 因此本文 RP-DPC 算法在时间性能上要明显优于 DPC 算法.

DPC 算法的核心思想是寻找密度峰值点采用启发式准则的决策图和非类簇中心点类别分配采用就近高密度的分配准则, 本文 RP-DPC 算法保留了 DPC 算法的核心思想, 因而两种算法的聚类质量是完全一样的. 但本文 RP-DPC 与 DPC 的不同在于: 决策图构建参数ρ和δ的计算过程中分别引入了加速策略, 极大地降低了时间复杂度. 加速策略的引入使得 RP-DPC 算法相对于 DPC 算法来说, 在保证聚类质量的同时能有效地提升聚类效率, 因而更加适用于大规模数据集的聚类处理.

3 实验结果与分析

实验采用人工数据集和真实数据集对 RP-DPC 算法进行测试和评价, 数据集详细描述见表 1. 其中人工数据集来源于同类研究的参考文献 [17–19], 真实数据集来源于 UCI 机器学习数据库[20]. 这些数据集是测试聚类算法性能非常经典的数据集, 从样本规模、特征个数到类簇数目都有较大的变化, 能够很好地比较和验证 RP-DPC 算法在时间上的优良性能. 实验部分的对比算法为 DPC 算法[10]、DF-DPC (Density peaks clustering with data field) 算法[11]、E-DPC 算法[12]、KNN-DPC 算法[13]和 FKNN-DPC 算法[14]. 本文 RP-DPC 算法及其他对比算法采用 Python3.6.0 实现. 实验环境为 Windows XP 操作系统, Intel(R) Core(TM) i7 CPU, 1.73 GHz.

3.1 实验结果与分析

下面的实验分两个部分, 第 1 部分对 RP-DPC 和对比算法进行聚类质量的展示和分析, 第 2 部分对 6 种算法的效率进行比较和分析.

3.1.1 聚类质量

表 2 给出了 6 种算法在人工个数据集上的聚类结果. 图 3 ~ 6 展示了 6 种算法在二维人工数据集上的聚类结果和对应的类簇中心点.

图 3 为 6 种算法对 flame 数据集的聚类结果显示. 除了类簇中心点不太一样, 6 种算法几乎都实现了正确聚类 (KNN-DPC 算法有 2 个样本的类别分配存在错误). 由图 4 可以看出, 在 pathbased 数据集上, 6 种算法都能找到正确的类簇中心, 但是在类

表 1 实验数据集

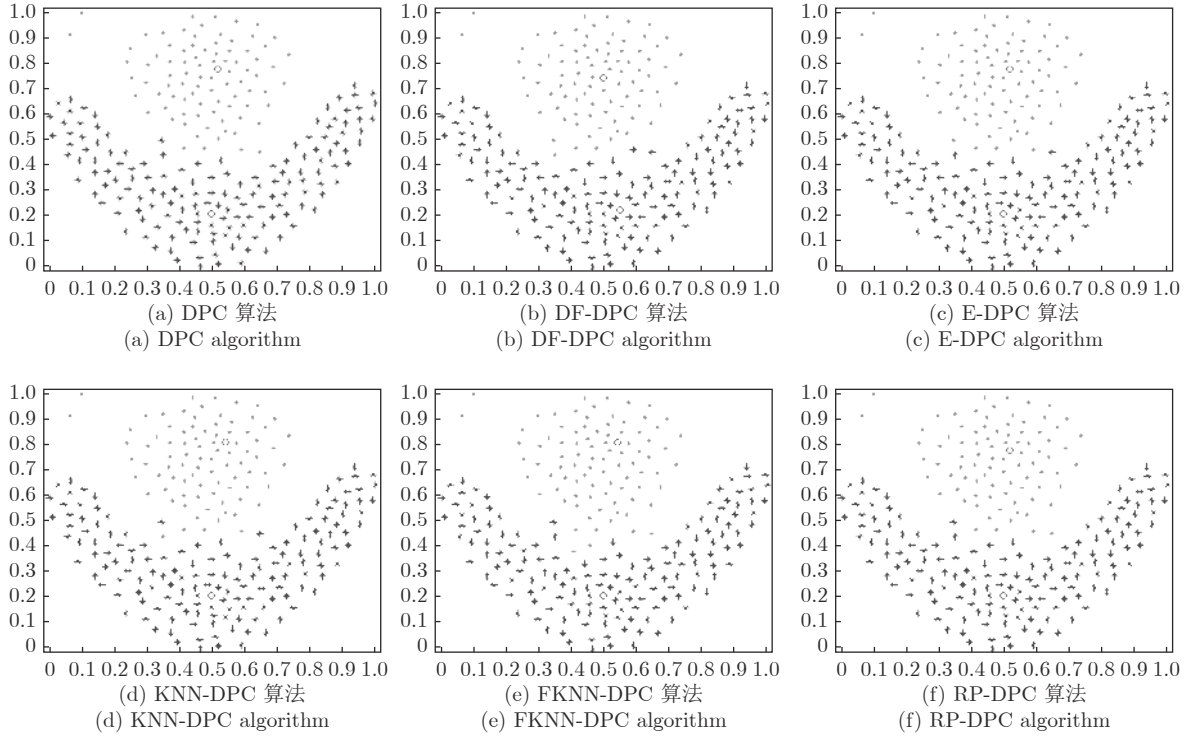
Table 1 Data sets used in experiments

数据集	样本数/特征数	类簇数/来源	数据集	样本数/特征数	类簇数/来源
flame	240/2	3/文献 [18]	iris	150/4	3/文献 [21]
pathbased	300/2	3/文献 [19]	wine	178/13	3/文献 [21]
spiral	312/2	3/文献 [19]	WDBC	569/30	2/文献 [21]
aggregation	788/2	7/文献 [17]	segmentation	2 310/19	7/文献 [21]
D31	3 100/2	31/文献 [20]	waveform	5 000/21	3/文献 [21]

表 2　6 种算法在人工数据集上的聚类结果

Table 2　Clustering results of six algorithms on synthetic datasets

数据集	算法	F/P	Par	Acc	AMI	ARI	算法	F/P	Par	Acc	AMI	ARI
flame	DPC	2/2	5	1.000	1.000	1.000	KNN-DPC	2/2	5	0.992	0.978	0.962
	DF-DPC	2/2	0.157	1.000	1.000	1.000	FKNN-DPC	2/2	5	1.000	1.000	1.000
	E-DPC	2/2	5	1.000	1.000	1.000	RP-DPC	2/2	5	1.000	1.000	1.000
pathbased	DPC	3/3	2	0.752	0.578	0.517	KNN-DPC	3/3	6	0.983	0.951	0.924
	DF-DPC	3/3	0.027	0.493	0.402	0.399	FKNN-DPC	3/3	6	0.983	0.951	0.924
	E-DPC	3/3	5	0.987	0.941	0.960	RP-DPC	3/3	2	0.752	0.578	0.517
spiral	DPC	3/3	2	1.000	1.000	1.000	KNN-DPC	3/3	6	1.000	1.000	1.000
	DF-DPC	3/3	0.148	1.000	1.000	1.000	FKNN-DPC	3/3	6	1.000	1.000	1.000
	E-DPC	3/3	5	1.000	1.000	1.000	RP-DPC	3/3	2	1.000	1.000	1.000
aggregation	DPC	7/7	4	0.999	0.998	0.996	KNN-DPC	3/3	6	0.999	0.997	0.995
	DF-DPC	7/7	0.084	0.999	0.998	0.996	FKNN-DPC	3/3	6	0.999	0.997	0.995
	E-DPC	7/7	4	0.999	0.998	0.996	RP-DPC	3/3	2	0.999	0.998	0.996
D31	DPC	31/31	1	0.968	0.937	0.956	KNN-DPC	31/31	12	0.954	0.903	0.939
	DF-DPC	31/31	0.039	0.968	0.937	0.956	FKNN-DPC	31/31	12	0.954	0.903	0.939
	E-DPC	31/31	1	0.968	0.937	0.956	RP-DPC	31/31	1	0.968	0.937	0.956



(a) DPC 算法 (a) DPC algorithm　(b) DF-DPC 算法 (b) DF-DPC algorithm　(c) E-DPC 算法 (c) E-DPC algorithm

(d) KNN-DPC 算法 (d) KNN-DPC algorithm　(e) FKNN-DPC 算法 (e) FKNN-DPC algorithm　(f) RP-DPC 算法 (f) RP-DPC algorithm

图 3　6 种算法在 flame 数据集的聚类结果

Fig. 3　Clustering results of six algorithms on flame dataset

簇分配上 6 种算法聚类效果差距较大. E-DPC 算法、KNN-DPC 算法和 FKNN-DPC 算法的聚类效果较好, 而 DPC 算法、DF-DPC 算法和本文的 RP-DPC 算法聚类效果较差. 因为 6 种算法的类簇中心点都是正确的, 所以问题出在非类簇中心点样本的类别分配上. DPC 算法、DF-DPC 算法和本文的 RP-DPC 算法采用的都是 DPC 算法的就近分配原则, 导致第 3 类样本中的大部分被误分到距离较近

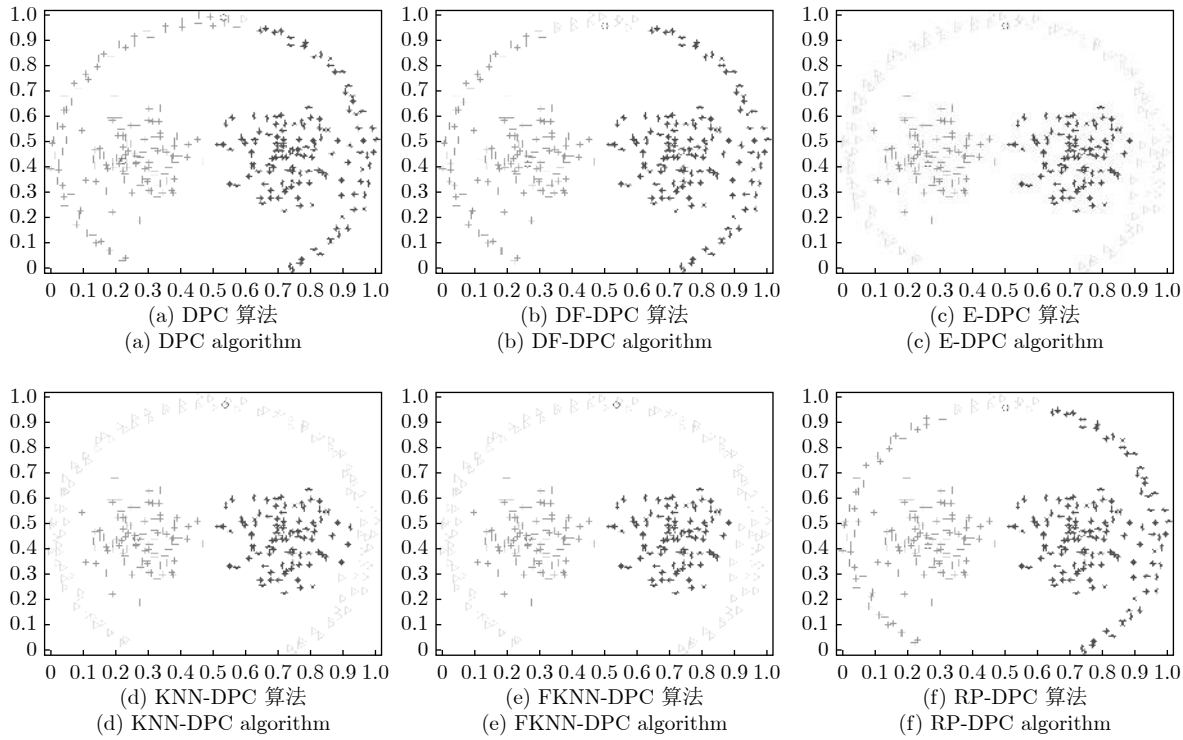


图 4 6 种算法在 pathbased 数据集的聚类结果

Fig. 4 Clustering results of six algorithms on pathbased dataset

且密度更高的第 1 类和第 2 类中.

图 5 所示的 6 种算法在 spiral 数据集上的聚类结果表明: 虽然各算法确定的类簇中心点不同, 但 6 种算法均实现了对 spiral 数据集的正确聚类. 图 6 所示的 6 种算法在 aggregation 数据集的聚类结果与 flame 数据集和 spiral 数据集上情况类似, 类簇中心点的选择略有不同, 但聚类结果都很好. 在 D31 数据集上, 6 种算法都找到了正确的类簇中心点, 并且聚类效果都较好.

从人工数据集上的聚类结果中可以发现, 本文 RP-DPC 算法与 DPC 算法聚类效果完全一致, 包括聚类中心点选取及最终类簇分布都是完全一致的. 表 2 中 DPC 算法与 RP-DPC 算法在人工数据集上的聚类结果指标也是完全一致的. 可见, RP-DPC 算法具有与 DPC 一样的聚类质量.

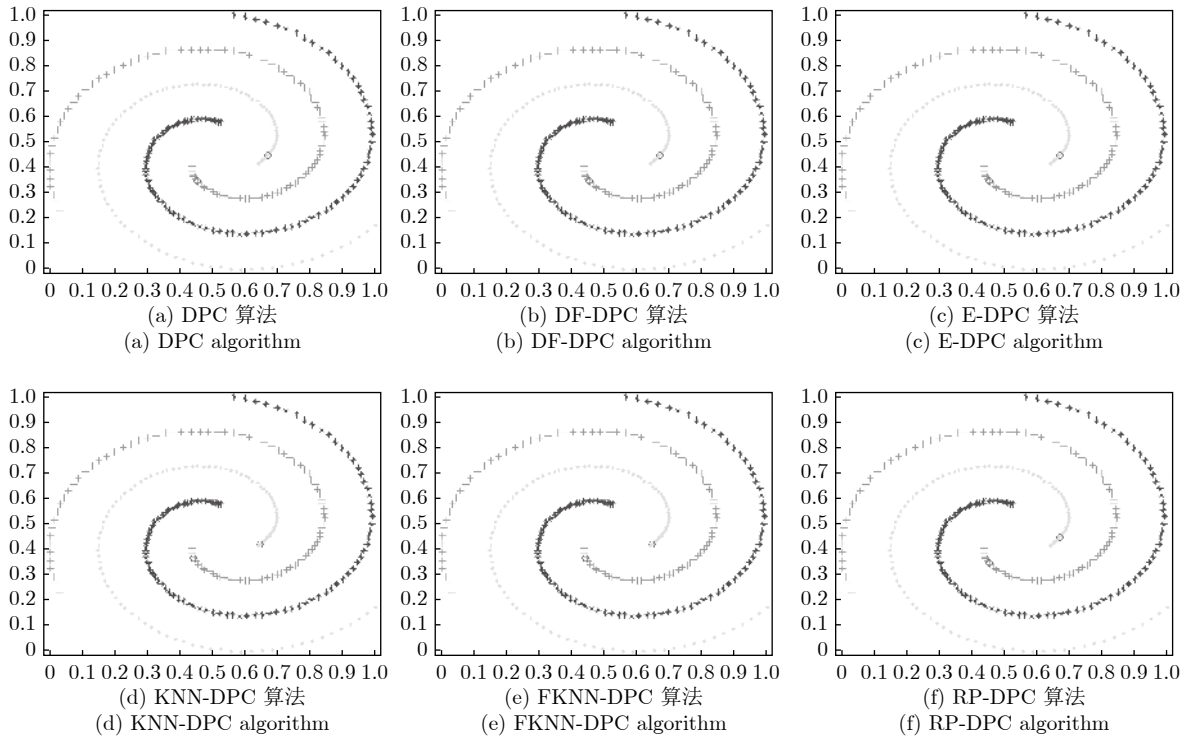
在 UCI 数据集上的聚类结果仍然用 Acc, AMI, ARI, F/P 四个评价指标来进行质量评价, 并且给出聚类结果对应的参数 Par. 6 种算法在真实数据集上的聚类结果如表 3 所示. 从 6 种算法在真实数据集上的聚类结果指标来看, KNN-DPC 算法和 FKNN-DPC 算法的聚类效果较好, 在 iris、wine 和 WDBC三个数据集上获得了最好, 在 segmentation 数据集上 DF-DPC 算法聚类效果最好, 而在 waveform 数据集上, E-DPC 算法聚类效果最好. 再回到 DPC 算法和本文的 RP-DPC 算法, 在相同的参数设置下, 两种算法取得了完全一致的聚类效果. 再次证明, 本文的 RP-DPC 算法具有与 DPC 一样的聚类质量, 而仅仅是针对聚类效率提出改进.

3.1.2 聚类效率

本小节通过实验详细比较了 6 种算法的聚类效率. 表 4 给出了 6 种算法在 5 个人工数据集和 5 个 UCI 真实数据集上的运行时间.

DPC 算法的时间复杂度为 $\mathrm{O}(mn^2)$, DF-DPC 算法、E-DPC 算法、KNN-DPC 算法和 FKNN-DPC 算法的复杂度都是 $\mathrm{O}(mn^2)$. 但是与 DPC 算法相比, DF-DPC 算法多了阈值计算的步骤, E-DPC 算法多了子类合并的步骤, KNN-DPC 算法和 FKNN-DPC 算法多了 K 近邻的查找与两步样本分配步骤, 所以上述各算法的执行时间均远高于 DPC 算法. 而 RP-DPC 算法的复杂度为 $\max(\max(\mathrm{O}(mn^2/p), \mathrm{O}(mn^2/q)), \mathrm{O}(n(n-k)+\bar{\rho}k))$ 是低于 DPC 算法的. 表 4 列出的 6 种算法在人工数据集和 UCI 数据集上的运行时间也证实了上述结论.

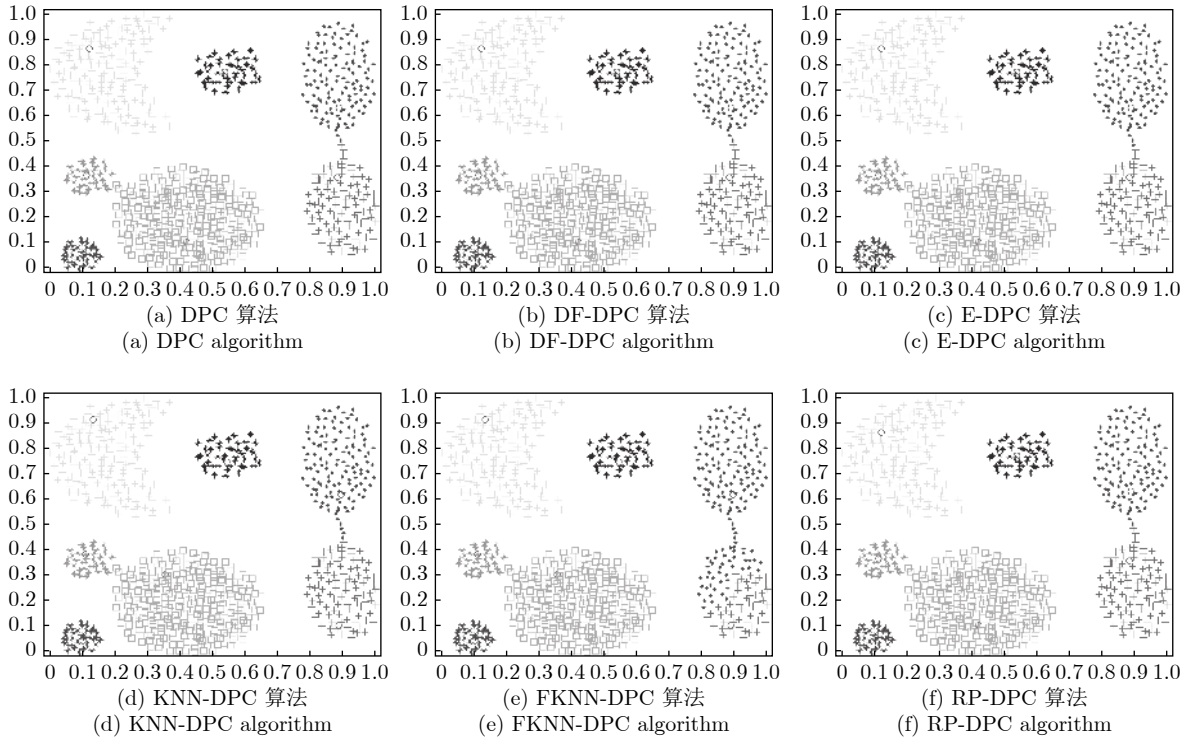
因为 DF-DPC 算法、E-DPC 算法、KNN-DPC



(a) DPC 算法 (a) DPC algorithm
(b) DF-DPC 算法 (b) DF-DPC algorithm
(c) E-DPC 算法 (c) E-DPC algorithm
(d) KNN-DPC 算法 (d) KNN-DPC algorithm
(e) FKNN-DPC 算法 (e) FKNN-DPC algorithm
(f) RP-DPC 算法 (f) RP-DPC algorithm

图 5 6 种算法在 spiral 数据集的聚类结果

Fig. 5 Clustering results of six algorithms on spiral dataset



(a) DPC 算法 (a) DPC algorithm
(b) DF-DPC 算法 (b) DF-DPC algorithm
(c) E-DPC 算法 (c) E-DPC algorithm
(d) KNN-DPC 算法 (d) KNN-DPC algorithm
(e) FKNN-DPC 算法 (e) FKNN-DPC algorithm
(f) RP-DPC 算法 (f) RP-DPC algorithm

图 6 6 种算法在 aggregation 数据集的聚类结果

Fig. 6 Clustering results of six algorithms on aggregation dataset

表 3 6 种算法在真实数据集上的聚类结果

Table 3 Clustering results of six algorithms on real datasets

数据集	算法	F/P	Par	Acc	AMI	ARI	算法	F/P	Par	Acc	AMI	ARI
iris	DPC	3/3	2	0.887	0.767	0.720	KNN-DPC	3/3	7	0.960	0.902	0.860
	DF-DPC	3/3	0.126	0.727	0.615	0.561	FKNN-DPC	3/3	7	0.973	0.912	0.922
	E-DPC	3/3	2	0.933	0.887	0.862	RP-DPC	3/3	2	0.887	0.767	0.720
wine	DPC	3/3	2	0.882	0.706	0.672	KNN-DPC	3/3	7	0.921	0.769	0.737
	DF-DPC	3/3	0.473	0.916	0.757	0.756	FKNN-DPC	3/3	7	0.949	0.831	0.852
	E-DPC	3/3	2	0.882	0.706	0.672	RP-DPC	3/3	2	0.882	0.706	0.672
WDBC	DPC	2/2	0.85	0.845	0.415	0.471	KNN-DPC	2/2	7	0.944	0.679	0.786
	DF-DPC	2/2	0.407	0.838	0.374	0.451	FKNN-DPC	2/2	7	0.944	0.679	0.786
	E-DPC	2/2	0.85	0.845	0.415	0.471	RP-DPC	2/2	0.85	0.845	0.415	0.471
segmentation	DPC	6/5	3	0.684	0.651	0.550	KNN-DPC	6/5	7	0.718	0.539	0.632
	DF-DPC	6/5	0.337	0.746	0.703	0.607	FKNN-DPC	6/5	7	0.716	0.655	0.555
	E-DPC	6/5	3	0.684	0.651	0.550	RP-DPC	6/5	3	0.684	0.651	0.550
waveform	DPC	3/3	0.5	0.586	0.318	0.268	KNN-DPC	3/3	5	0.696	0.338	0.313
	DF-DPC	3/3	0.604	0.492	0.325	0.276	FKNN-DPC	3/3	5	0.703	0.324	0.350
	E-DPC	3/3	2	0.716	0.368	0.338	RP-DPC	3/3	0.5	0.586	0.318	0.268

算法和 FKNN-DPC 算法的时间复杂度都是高于 DPC 算法的, 而本文 RP-DPC 算法复杂度是低于 DPC 算法的,所以我们接下来的时间比较都是以 DPC 算法为基准. 从表 4 的运行时间结果来看, 相比于 DPC 算法, DF-DPC 算法、E-DPC 算法、KNN-DPC 算法和 FKNN-DPC 算法在全部 10 个数据集上执行时间都要更久, 并且随着数据集规模的增大, 它们与 DPC 算法执行时间之间的差距也在加剧增大, 尤其是 DF-DPC 算法、KNN-DPC 算法和 FKNN-DPC 算法. 而 E-DPC 算法的相对增加幅度较为稳定, 是因为 E-DPC 算法相对 DPC 算法增加的子类合并步骤计算量与子类个数有关, 受数据规模的影响不大.

再来比较 RP-DPC 算法与 DPC 算法. 在 10 个数据集上,与 DPC 算法相比, RP-DPC 算法都花费了更少的时间. 在人工个数据集上, RP-DPC 算法相对 DPC 算法时间性能提升幅度最少的是在 flame 数据集, 提升约 2 倍, 性能提升最多的是在 D31 数据集上, 提高近 9 倍, 而在 UCI 数据集上, RP-DPC 算法相对 DPC 算法时间性能提高最少的是在 wine 数据集, 提升 3 倍多, 时间性能提高最多的是在 waveform数据集上, 提高近 19 倍. 同时, 我们还发现, 在 10 个数据集上, 随着样本规模的增大, DPC 算法时间消耗增加较快, 而 RP-DPC 算法的时间性能则相对比较稳定, 两个算法时间性能上的差距随着样本规模的增大总体呈正向增大.

为了进一步分析 RP-DPC 算法在聚类效率上的优势, 下面进一步对聚类过程中 6 种算法发生的样本间距离计算次数, 单个样本 ρ 值平均计算量上分别进行了统计对比. 需要说明的是 D31, segmentation 和 waveform 这 3 个数据集规模较大, 结果的数量级要高于其他数据集, 为了便于绘图比较, 图 7～10 中对这 3 个数据集的结果做了降低数量级的处理, 但不影响两个算法的对比效果.

由图 7 和图 8 可以看出, RP-DPC 算法与其余 5 种算法在各个数据集上距离计算次数的对比是很明显的. 这 5 种算法都需要计算全部样本两两之间的距离, 而 RP-DPC 算法只计算了双基准点映射模型与剪枝策略下必要的样本间距离, 因而距离计算量大大小于其余 5 种算法. 在人工数据集上, RP-DPC 算法与其余 5 种算法距离计算次数相差最少是在 flame 数据集, 相差最多的是在 D31 数据集, 而在真实数据集上, 距离计算次数相差最少是在 wine 数据集, 相差最多的是在 waveform 数据集, 这一结果与表 4 中的算法运行时间也是吻合的.

图 9 和图 10 给出了 RP-DPC 算法和其余 5 种算法在 10 个数据集上计算单个样本密度 ρ 的平均计算量. KNN-DPC 算法和 FKNN-DPC 算法虽然直接需要比较的样本是 K 个, 但是这两种算法首先需要从其他 $n-1$ 个样本中找出 K 近邻, 因而这两种算法计算单个样本密度 ρ 的平均计算量是两者之

表 4　6 种算法在数据集上的运行时间 (s)

Table 4　Running time of six algorithms on several datasets (s)

	数据集	DPC	DF-DPC	E-DPC	KNN-DPC	FKNN-DPC	RP-DPC
人工数据集	flame	0.103	0.185	0.142	0.179	0.188	0.054
	pathbased	0.137	0.265	0.201	0.193	0.260	0.036
	spiral	0.132	0.313	0.211	0.221	0.311	0.027
	aggregation	0.354	0.567	0.524	0.717	0.764	0.035
	D31	1.825	3.603	3.592	4.382	4.671	0.221
真实数据集	iris	0.088	0.142	0.131	0.137	0.156	0.020
	wine	0.108	0.172	0.131	0.141	0.165	0.032
	WDBC	0.372	0.836	0.482	0.705	0.904	0.058
	segmentation	1.836	2.763	2.013	4.220	4.918	0.191
	waveform	9.551	16.381	13.106	22.528	24.741	0.537

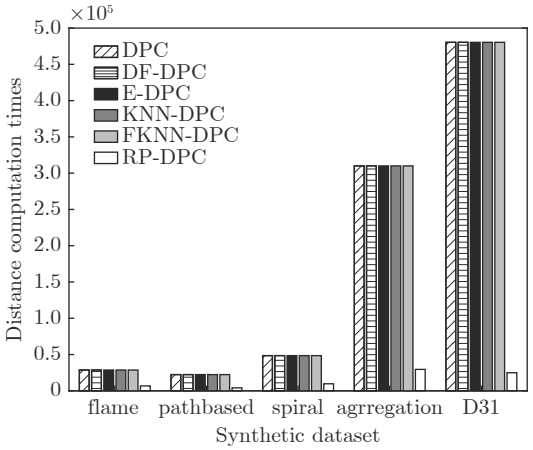


图 7　6 种算法在各人工数据集上距离计算次数

Fig. 7　Distance computation times on synthetic data sets of six algorithms

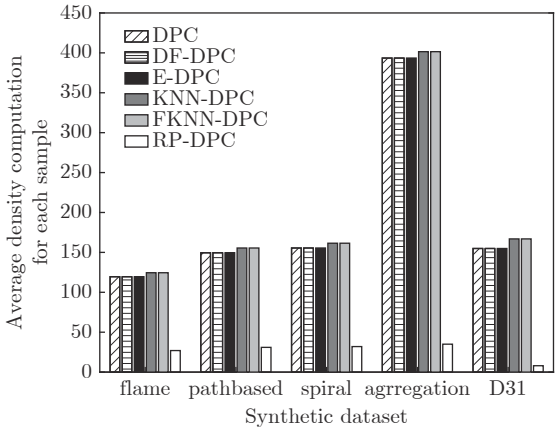


图 9　6 种算法在各人工数据集上单个样本密度平均计算量

Fig. 9　Average computation for single sample density on synthetic data sets of six algorithms

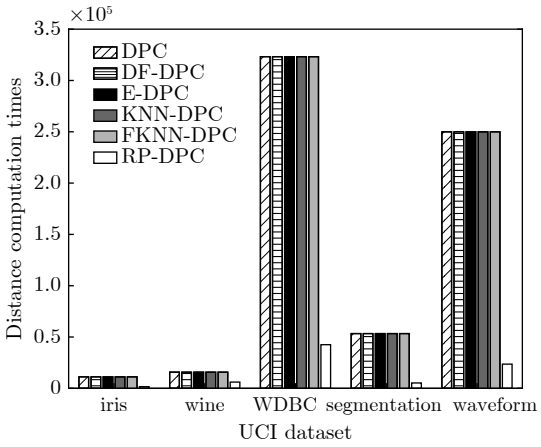


图 8　6 种算法在各真实数据集上距离计算次数

Fig. 8　Distance computation times on real data sets of six algorithms

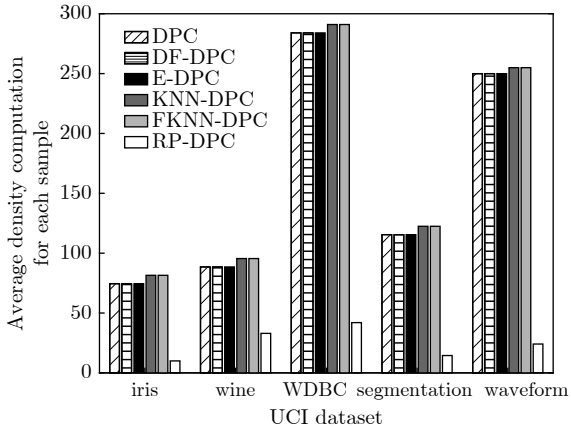


图 10　6 种算法在各真实数据集上单个样本密度平均计算量

Fig. 10　Average computation for single sample density on real data sets of six algorithms

表 5 RP-DPC 算法在 10 个数据集上的剪枝率
Table 5 Pruning ratio of RP-DPC algorithm on ten data sets

数据集	数据集规模	RP-DPC	剪枝率(%)	数据集	数据集规模	RP-DPC	剪枝率(%)
flame	240	5	97.9	iris	150	15	90
pathbased	300	8	97.3	wine	178	9	94.9
spiral	312	10	96.8	WDBC	569	21	96.3
aggregation	788	23	97.1	segmentation	2 310	55	97.6
D31	3 100	49	98.4	waveform	5 000	89	98.2

和. DPC 算法、DF-DPC 算法和 E-DPC 算法在计算每个样本的密度时需要将该样本与其他 $n-1$ 个样本作比较, 只有 RP-DPC 算法在计算每个样本的密度时仅仅需要将该样本与相邻的相对邻域中的样本作比较, 因而 6 种算法在单个样本平均密度计算量上的巨大差距是合理的. 在人工数据集上, RP-DPC 算法与其余 5 种算法在单个样本平均密度计算量上相差最少是 flame 数据集, 相差最多的是 D31 数据集, 近 30 倍差距. 而在真实数据集上, 相差最少是在 wine数据集, 相差最多的是在 waveform 数据集, 这一统计结果与表 4 中的算法运行时间同样是吻合的.

在 δ 值计算步骤中, RP-DPC 算法引入了剪枝策略, 将被剪枝样本的 值计算范围从原来的样本集缩小到邻域内, 以此来提升算法的时间性能. 实验中我们统计了被剪枝样本数量与原数据集样本规模之比,定义为剪枝率. RP-DPC 算法在 10 个数据集上的剪枝率的结果见表 5.

在 10 个数据集上, RP-DPC 算法的剪枝率均达到了 90 % 以上, 最高达到 98.4 % 是在 D31 数据集上. 这一实验结果表明绝大多数的非类簇中心点样本都是在其邻域范围内计算 δ 值, 因而对整个算法时间性能的提升是明显的.

3.2 结论

本文针对密度中心聚类算法 DPC 及其改进算法效率不高的问题, 基于相对邻域映射与剪枝策略提出了一种高效的密度中心聚类 RP-DPC 算法. 与 DPC 算法及其改进算法相比, RP-DPC 通过双基准点的距离映射给出了样本间的相对位置, 缩小了样本密度统计的范围, 从而避免了很多冗余的距离计算. 在样本 δ 值计算中引入剪枝策略, 缩小样本的 δ 值计算范围, 有效地提高了算法的效率. 如何将邻域信息引入到非类簇中心点样本的类簇分配, 实现高效且高质量的聚类是下一步将研究的问题.

References

1 Xu R, Wunsch D C. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 2005, **16**(3): 645−678

2 Frey B J, Dueck D. Clustering by passing messages between data points. *Science*, 2007, **315**: 972−976

3 Jain A K. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 2010, **31**: 651−666

4 Williamson B, Guyon I. Clustering: science or art? *Journal of Machine Learning Research*, 2012, 27: 65−80

5 Kaufman L, Peter R. Clustering by means of medoids. *Statistical Data Analysis Based on the L1 Norm and Related Methods, North-Holland: North-Holland Press*, 1987. 405−416

6 MacQueen J. Some methods for classification and analysis of multivariate observation. In: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. Los Angelas, University of California, USA: 1967. 281−297

7 Zhang W, Wang X G, Zhao D L, Tang X O. *Graph Degree Linkage: Agglomerative Clustering on a Directed Graph*. Berlin: Springer, 2012. 428−441

8 Wang W, Yang J, Muntz R. Sting: a statistical information grid approach to spatial data mining. In: Proceedings of International Comferenceon Very Large Date Bases (VLDB'97). San Francisco, CA, USA: Morgan Kaufman, 1997. 186−195

9 Ester M, Kriegel H P, Sander J, Xu X W. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of ACM KDD'96. New York, USA: ACM, 1996. 226−231

10 Rodriguez A, Laio A. Clustering by fast search and find of density peaks. *Science*, 2014, **344**(6191): 1492−1496

11 Wang S L, Wang D K, Li C Y, Li Y, Ding G Y. Clustering by fast search and flnd of density peaks with data fleld. *Chinese Journal of Electronics*, 2016, **3**(25): 397−402

12 Zhang W K, Li J. Extended fast search clustering algorithm: widely density clusters, no density peaks. *Computer Science and Technology*, 2015.

13 Xie Juan-Ying, Gao Hong-Chao, Xie Wei-Xin. *K*-nearest neigh-

bors optimized clustering algorithm by fast search and flnding the density peaks of a data set. *SCIENTIA SINICA Informationis*, 2016, **46**(2): 258−280
(谢娟英, 高红超, 谢维信. K 近邻优化的密度峰值快速搜索聚类算法. 中国科学: 信息科学, 2016, **46**(2): 258−280)

14 Xie J Y, Gao H C, Xie W X. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors. *Information Science*, 2016, **354**: 19−40

15 Gong Shu-Feng, Zhang Yan-Feng. EDDPC: an efficient distributed density peaks clustering algorithm. *Journal of Computer Research and Developmnet*, 2016, **53**(6): 1400−1409
(巩树凤, 张岩峰. EDDPC: 一种高效的分布式密度中心聚类算法. 计算机研究与发展, 2016, **53**(6): 1400−1409)

16 Liu Y, Huang W L, Jiang Y L, Zeng Z Y. Quick attribute reduction algorithm for neighborhood rough set model. *Information Sciences*, 2014, **271**: 65−81

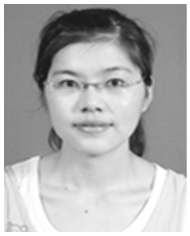
17 Gionis A, Mannila H, Tsaparas P. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data*, 2007, **1**(1): 1−30

18 Fu L M, Medico E. Flame, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioininformatics*, 2007, **8**(1): 3−17

19 Chang H, Yeung D Y. Robust path-based spectral clustering. *Pattern Recognition*, 2008, **41**(1): 191−203

20 Veenman C J, Reinders M J T, Baker E. A maximum variance cluster algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, **24**(9): 1273−1280

21 Lichman M. UCI machine learning repository. http://archive.ics.uci.edu/ml. Irvine: University of California, March 21, 2017



纪 霞 博士, 安徽大学计算机科学与技术学院讲师. 主要研究方向为数据挖掘与机器学习, 粒计算与粗糙集理论. 本文通信作者.
E-mail: jixia1983@163.com
(**JI Xia** Ph.D., lecturer at the School of Computer Science and Technology, Anhui University. Her research interest covers data mining and machine learning, granular computing, and rough set theory. Corresponding author of this paper.)

姚 晟 博士, 安徽大学计算机科学与技术学院讲师. 主要研究方向为粗糙集理论与粒计算.
E-mail: fishenyao@126.com
(**YAO Sheng** Ph.D., lecturer at the School of Computer Science and Technology, Anhui University. Her research interest covers rough set theory and granular computing.)

赵 鹏 博士, 安徽大学计算机科学与技术学院副教授. 主要研究方向为机器学习, 智能信息处理.
E-mail: zhaopeng_ad@163.com
(**ZHAO Peng** Ph.D., associate professor at the School of Computer Science and Technology, Anhui University. Her research interest covers machine learning and intelligent information processing.)