计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(4):126−136 [doi: 10.15888/j.cnki.csa.007343]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

自调节步长果蝇优化的自适应密度峰值聚类①

邓然然, 李　伟, 杨荣新

(长安大学 信息工程学院, 西安 710064)
通讯作者: 李　伟, E-mail: 815527752@qq.com

摘　要: 为了解决密度峰值聚类算法 (Density Peaks Clustering algorithm, DPC) 设置截止距离和选择聚类中心过程中的问题, 一种新的自调节步长果蝇优化算法被用于密度峰值聚类的重要参数截止距离的计算, 设计了一种自适应选择聚类中心的方法. 在截止距离计算过程中, 根据迭代过程中每一步之间的最优浓度与最差浓度的差值变化率动态的调节寻优步长, 其寻优效率与精度均优于现存的改进果蝇算法. 在聚类中心的选择过程中, 由局部密度与距离乘积的分布情况, 自适应的选择聚类中心. 本文提出的自调节步长果蝇优化的自适应密度峰值聚类算法的计算精度和效率均优于现存的密度峰值聚类改进算法, 并能完全自适应的实现数据的聚类.

关键词: 聚类; 密度峰值聚类; 果蝇优化; 自调节步长; 自适应

引用格式: 邓然然,李伟,杨荣新.自调节步长果蝇优化的自适应密度峰值聚类.计算机系统应用,2020,29(4):126–136. http://www.c-s-a.org.cn/1003-3254/7343.html

Adaptive Density Peak Clustering Based on Fruit Fly Optimization of Self-Adjusting Step-Size

DENG Ran-Ran, LI Wei, YANG Rong-Xin

(School of Information Engineering, Chang'an University, Xi'an 710064, China)

Abstract: In order to solve the problem of setting cut-off distance and selecting clustering center in Density Peak Clustering algorithm (DPC), a new self-adjusting step-size fruit fly optimization algorithm is used to calculate the cut-off distance and the important parameters in density peak clustering, an adaptive method for selecting clustering centers is designed. In the cut-off distance calculation process, the search step-size is dynamically adjusted according to the rate of change of the difference between the optimal concentration and the worst concentration in each step of the iterative process, and its optimization efficiency and accuracy are better than the existing improved fruit fly algorithm. In the selection process of clustering center, the clustering center is selected adaptively according to the distribution of the product of local density and distance. The computational accuracy and efficiency of the proposed algorithm are both better than the existing improved DPC algorithm, and it can realize data clustering completely adaptively.

Key words: clustering; Density Peak Clustering (DPC); fruit fly optimization; self-adjusting step-size; adaptive

聚类是一种常用的属于无监督学习的数据处理方法, 由聚类所挖掘出的信息可以为进一步深入地分析数据提供理论基础[1]. 聚类采用定量数学思想, 按照数据间的相似程度将其归到几个群, 其结果满足各个群内相似性较强, 而各群体之间相似性较小的特点. 同一组样本有时会因为不同的目的、数据输入方式、所选的分群特征或数据属性, 形成不同的分群结果. 聚类是数据挖掘的重要手段[2], 聚类算法分为: 划分聚类、层次聚类、密度聚类、网格聚类、图论聚类等[3].

快速峰值算法是 2014 年 6 月 Rodriguez Alex 在

① 基金项目: 国家自然科学基金面上项目 (51978071); 中央高校基本科研业务费专项资金 (300102249301, 300102249306)
Foundation item: General Program of National Natural Science Foundation of China (51978071); the Fundamental Research Funds for the Central Universities of China (300102249301, 300102249306)
收稿时间: 2019-08-19; 修改时间: 2019-09-06, 2019-09-19; 采用时间: 2019-09-29; csa 在线出版时间: 2020-04-05

Science 上提出的一种新型聚类算法, 该算法能自动选出样本的类中心, 而且能克服一般方法对数据要求的缺陷, 可以实现对非球形数据进行高效聚类[4]. 该算法选定聚类中心: 1) 较大的局部密度, 即中心点相邻点的密度值均小于该点; 2) 与其他密度更高的点之间的“距离”较大. DPC 算法原理简单、聚类高效[5], 在各个领域都有广泛的应用[6,7], 例如应用于高速公路收费数据分析[8]及异常事件挖掘[9]等. 然而, 通过聚类算法的评价[10], DPC 算法的缺点也十分明显, 需要根据经验值设定参数截断距离; 需要根据计算出的局部密度以及距离两个值生成用于选择聚类中心的决策图, 并人为在其中选取这两个值都较大的点为聚类中心. 这种选择存在较高的主观性与不稳定性, 严重影响后续非中心点的分配、优化以及噪声点的发现[11].

现今为止, 已有多种改进的 DPC 算法, 例如: K 近邻密度峰值聚类[12–14], 结合 K 近邻的概念重新定义截断距离和局部密度的度量方法, 避免截止距离的人为设置; 基于果蝇算法的密度峰值聚类[15–17], 利用果蝇优化算法得到最优截止距离, 进而完成对局部密度和与密度更高的点的距离的计算; 布谷鸟优化的密度峰值快速搜索聚类算法[18], 利用布谷鸟优化截止距离, 引入余弦相似度原理, 将方向与实际距离相结合, 能够更好的划分边界区域内的数据点; 除此之外, 还有热扩散密度峰值聚类[19]、自适应聚合策略优化的密度峰值聚类算法[20]等等. 但是现有的改进算法仍有改进的余地.

果蝇优化算法 (Fruit fly Optimization Algorithm, FOA) 是一种新的群体智能方法[21,22], 通过迭代搜索寻找最优解. 与此同时, 已有诸多学者致力于改进 FOA 算法, 例如新型改进果蝇优化算法[23]、改进的变步长果蝇优化算法[24]、改进步长与策略的果蝇优化算法[25]等. 但是果蝇优化算法还存在许多不足.

本文引入改进的自调节步长果蝇优化算法[26], 该算法根据当前浓度差值变化率, 对步长进行相应调整, 用过迭代计算, 得到最佳截止距离. 本文以更高的效率与准确率的得到这一关键参数, 并计算密度与距离的乘积, 根据乘积值自适应的选出备选聚类中心, 再从备选聚类中心中, 以同一类仅有一个中心为原则选出真正的聚类中心, 然后对其他非中心数据点进行分配, 最后通过类的边缘区域对聚类结果进行优化.

1 算法原理

1.1 密度峰值聚类算法

DPC 算法主要分 3 个步骤进行: 计算距离矩阵、选取聚类中心、剩余点的分配及优化.

1.1.1 计算距离矩阵

密度峰值算法的输入为待处理数据点的距离矩阵, 即每个点之间相似性. 在进行距离计算之前, 首先将原始数据的不同字段进行标准化处理, 使各维度的数据具有相同的量级; 然后计算数据间的距离, 最后输出距离矩阵. 本文选取欧几里得方式来计算距离$d(x_i, x_j)$, 并输出为若干行、三列的矩阵, 其中前两列表示两个数据点的序号, 第三列为前两列表示的两个数据点之间的距离, 例如矩阵的第 1 行为: 1 号数据、2 号数据、1 号 2 号数据间的距离, 第 2 行为: 1 号数据、3 号数据、1 号 3 号数据间的距离, 以此类推, 与此距离矩阵即为快速峰值算法的输入.

1.1.2 聚类中心的选择

聚类中心的特征是局部密度大而且与其它密度更大的点相距很远.

某点的局部密度ρ_i是以截止距离为半径的圆内的点的个数:

$$\rho_i = \sum\nolimits_j \chi\left(d_{ij} - d_c\right) \tag{1}$$

式中, i与j为两个互异的数据点, 当 a 小于 0 时, $\chi(a) = 1$, 反之$\chi(a) = 0$. d_{ij}为i点与j点之间的距离, d_c为截止距离, 由用户根据经验值设定. 一般而言d_c 的选取原则为: 将距离δ_i按递增方式进行排序并编号, 找出所有距离个数的 1%~2% 所对应的序号, 将d_c设置为该序号对应的距离值.

某一数据的δ_i值被定义为:

$$\delta_i = \min_{j:\rho_j > \rho_i} \left(d_{ij}\right) \tag{2}$$

对于密度最高的点, 由于不存在更高点, 故将其δ_i值定义为该点与其余所有的数据之间距离的最大值.

$$\delta_i = \max_j \left(d_{ij}\right) \tag{3}$$

计算出各点的这两个量之后, 将所有的数据点以ρ和δ作为两个维度进行可视化输出, 所输出的图形称为决策图.

1.1.3 非聚类中心点的分配及优化

在找到聚类中心之后, 确定类的数量, 然后合理分

配剩余点. 分配原则是: 每个剩余点逐个被分到与其距离最近的有较高密度的点所在的类簇, 且此操作以单步执行, 直到把所有的点全部分配到对应的类为止.

一般的聚类方法优化都是以使目标函数在每一次的迭代中达到最优为原则而实现的. 本文提出的算法中优化的方法为: 先为每个类划分一个边界区域, 边界区域的定义是分配给某一类簇的点距离另一类簇的点的距离小于截止距离d_c; 接着在每个边界区域内找出密度最高的点并标记其密度为ρ_b, 遍历类内的各个点, 密度大于ρ_b的点被分到类内, 反之被标记为噪声.

1.2 自调节步长果蝇优化算法

1.2.1 果蝇优化算法

FOA 模仿果蝇搜寻食物源, 果蝇通过嗅觉可以轻易的捕捉到范围内目标源散发的气味, 然后利用其视觉进行追踪, 不断的更新果蝇群位置, 逐渐靠近目标源. 算法的主要步骤如下:

1) 初始化种群:

果蝇数量为$S_{\text{ize_num}}$和算法所需循环执行总次数$M_{\text{ax_times}}$. 在确定活动区域中随机给定果蝇群体一个起点$S_{\text{tart_}X}$, $S_{\text{tart\_}Y}$.

2) 给每个果蝇个体随机的方向与距离, 用嗅觉寻找食物:

$$X_i = S_{\text{tart\_}X} + R_{\text{andom\_Value}} \tag{4}$$

$$Y_i = S_{\text{tart\_}Y} + R_{\text{andom\_Value}} \tag{5}$$

3) 计算果蝇到坐标原点的距离D_i以及味道浓度判别值S_i:

$$D_i = \left(X_i^2 + Y_i^2\right)^{0.5} \tag{6}$$

$$S_i = 1/D_i \tag{7}$$

4) 将浓度判别值S_i带入判定函数$F_{\text{unction}}(S_i)$求出其$S_{\text{mell_}i}$, 此处判定函数为使用果蝇优化算法对某一函数求解最优解时的判定函数, 根据实际被求解函数而定:

$$S_{\text{mell\_}i} = F_{\text{unction}}(S_i) \tag{8}$$

5) 找出群体中浓度最优的果蝇 (本文以最小值为最优解):

$$\left[b_{\text{est\_}}S_{\text{mell\_best\_}}i_{\text{ndex}}\right] = \min(S_{\text{mell}}) \tag{9}$$

6) 保留最佳浓度值$S_{\text{mell_best}}$与其坐标, 果蝇群体根据视觉飞往最优位置:

$$S_{\text{mell\_best\_}i} = b_{\text{est\_smell}} \tag{10}$$

$$S_{\text{tart\_}X} = X\left(b_{\text{est\_smell}}\right) \tag{11}$$

$$S_{\text{tart\_}Y} = Y\left(b_{\text{est\_smell}}\right) \tag{12}$$

7) 迭代寻优: 重复步骤 2)、3)、4)、5), 判断: $S_{\text{mell\_best\_}i} < S_{\text{mell\_best\_}i-1}$, 如果上式取值为真, 则转到步骤 6), 否则继续迭代进行优化.

1.2.2 自调节步长果蝇优化算法 (SFO)

传统的 FOA 算法的搜索半径是固定不变的, 即每一代果蝇以固定步长随机在寻找食物. 在算法迭代开始时, 较大的步长有利于全局搜索寻优, 而且可以有效的跳出局部最优. 但是, 寻优后期, 较大的步长会导致较弱的局部搜索, 可能会错过最佳值, 同时收敛精度也会减小. 较小的步长虽然可以提升收敛度, 但是在迭代前期降低了寻优速率. 传统的 FOA 算法难以权衡全局搜索能力和局部探索能力, 自调节步长果蝇优化算法根据浓度差值变化率的大小对步长进行动态更改; 并且在改变步长过程中引入指数与三角函数机制, 使步长变化具有非均匀性和随机性. 在迭代初期为了加快全局搜索速率, 适当增加步长值, 迭代后期为了能让算法能对局部进行精细搜索, 适当减小步长值, 该算法针对原果蝇算法的全局寻优速率慢和局部收敛精度不高而做出了改良. 算法的主要改进步骤如下:

1) 求出当前迭代果蝇群中的最优浓度$\min\left(S_{\text{mell_}i}\right)$与最差浓度$\max\left(S_{\text{mell_}i}\right)$之差, 前一代果蝇群中的最优浓度$\min\left(S_{\text{mell_}i-1}\right)$与最差浓度$\max\left(S_{\text{mell_}i-1}\right)$之差, 将差值作对比, 得出浓度差值变化率$S_{\text{Rate}}$:

$$S_{\text{Rate}} = \frac{\min\left(S_{\text{mell\_}i}\right) - \max\left(S_{\text{mell\_}i}\right)}{\min\left(S_{\text{mell\_}i-1}\right) - \max\left(S_{\text{mell\_}i-1}\right)} \tag{13}$$

本文将最小值设为最优值, 将最大值设为差值. 由图 1 可以看出果蝇寻优迭代前期的浓度差值变化率变化较大, 需要适当增加步长加快全局寻优, 而随着进入迭代后期, 浓度差值变化率变小, 逐渐趋近于 1, 并稳定在 0.7~1.3 的范围内, 说明果蝇群体距离食物源已经非常靠近, 浓度波动较小, 这时需要减小步长, 以提升精确度.

2) 步长改进: 根据S_{Rate}的取值范围, 对步长进行相应的修改:

如果$S_{\text{Rate}} \le 0.7$或者$S_{\text{Rate}} \ge 1.3$:

$$L_i = L_{i-1} - \left(1 - \frac{t_{\text{imes\_}i}}{M_{\text{axtimes}}}\right) \times D_{\text{ynamic\_factor}} \times 0.7 \quad (14)$$

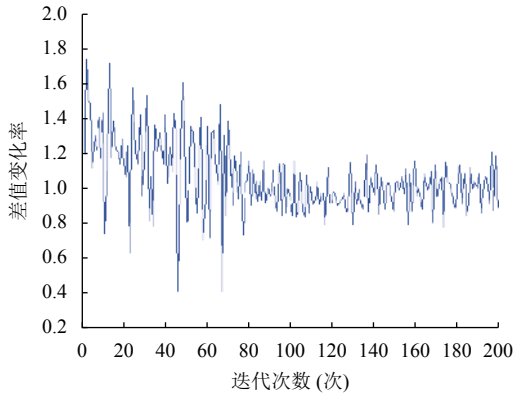

图 1 浓度差值变化率

步长调节因子:

$$D_{\text{ynamic\_factor}} = \exp\left(m_{\text{up}} \times S_{\text{Rate}}\right) \times \sin\left(\text{rem}\left(t_{\text{imes\_}i}, \pi\right)\right) \quad (15)$$

如果$0.7 < S_{\text{Rate}} < 1.3$:

$$L_i = L_{i-1} - \left(1 - \frac{t_{\text{imes\_}i}}{M_{\text{axtimes}}}\right) \times D_{\text{ynamic\_factor}} \times 1.3 \quad (16)$$

步长调节因子:

$$D_{\text{ynamic\_factor}} = \exp\left(m_{\text{down}} \times S_{\text{Rate}}\right) \times \sin\left(\text{rem}\left(t_{\text{imes\_}i}, \pi\right)\right) \quad (17)$$

式中, L_i与L_{i-1}分别为当前果蝇寻优迭代步长和上一次果蝇寻优迭代步长; m_{up}和m_{down}分别在不同所属浓度差值变化率时所采用的步长动态变化参数, 需要增大步长时采用m_{up}(取值大于 1, 本文取 1.3), 需要减小步长时采用m_{down}(取值为 (0,1), 本文取 0.7); $t_{\text{imes_}i}$为当前执行算法所处的运行次数; M_{axtimes}为算法所需循环执行次数的最大值. 根据每次求的浓度差值变化率S_{Rate}, 确定其所属范围, 对步长进行相应的修改.

图 2 为步长调节因子在取值连续的情况下所显示的结果. 指数机制可以使步长变化具有非均匀性, 在果蝇寻优迭代过程中, 如果浓度变化较大, 那么要适当增加步长, 非均匀变化的步长相对于原果蝇算法中的固定步长更容易捕捉到最优值, 有利于全局搜索.

3) 果蝇个体位置计算:

$$X_i = S_{\text{tart\_}X} + L_i \times X_{i-1} \times R_{\text{andom\_Value}} \quad (18)$$

$$Y_i = S_{\text{tart\_}Y} + L_i \times Y_{i-1} \times R_{\text{andom\_Value}} \quad (19)$$

当浓度差值变化率较小时, 浓度变化率比较稳定, 这时候果蝇迭代寻优可能进入两种状态, 第一种状态是进入迭代后期, 要适当的减小步长进行局部探索, 第二种状态是果蝇寻优迭代陷入局部最优化, 那么引入三角函数机制是利用三角函数变化的特性, 使得步长变化随机, 提高了算法跳出局部极值的能力.

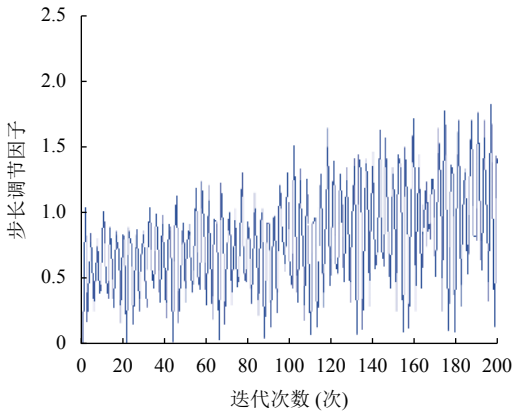

图 2 步长调节因子变化曲线

在同样设置种群规模为 20, 最大迭代次数为 1000 时, 对比 FOA 和 SFO 的测试性能, 采用平均值代表平均精度值, 方差代表测试稳定性. 对于不同的测试函数, FOA 在多数猜测是函数下未能达到理想精度, 容易陷入局部最佳, SFO 算法均达到了目标精度, 且方差相对较小, 证明本文算法具有更好的稳定性和优越性.

1.3 基于自调节步长果蝇优化的自适应密度峰值聚类算法

1.3.1 信息熵

信息熵为系统不稳定性的度量. 已知空间中的数据集$D = \{x_1, x_2, \cdots, x_n\}$包含 n 个数据, 每个数据的密度函数值为:

$$\varphi_i = \sum_{j=1}^{n} e^{\left(\frac{\|x_i - x_j\|}{\sigma}\right)^2} \quad (20)$$

则密度估计可以定义为:

$$H = -\sum_{i=1}^{n} \frac{\varphi_i}{Z} \log\left(\frac{\varphi_i}{Z}\right) \quad (21)$$

其中, $Z = \sum_{i=1}^{n} \varphi_i$为标准化因子.

1.3.2 基于自调节步长果蝇优化的自适应密度峰值聚类

对小规模数据集进行聚类时, 密度峰值聚类算法

采取指数方式计算数据密度, 计算公式如下:

$$\rho_i = \sum_{j\neq i} \exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right) \tag{22}$$

对比式 (20)、(22) 发现, 如果选择高斯核函数进行每个数据点密度的计算, 截断距离参数d_c与σ具有的意义是统一的, 优化σ本质上就是对截断距离参数d_c的优化. 而且, 若将整个数据集看成一个系统, 一个好的聚类结果是系统最稳定, 数据间关系确定性最好的状态, 因此, 以信息熵最小值为判定依据, 利用 FOA 算法的全局寻优特性, 对σ进行优化, 优化后得出的σ值即为最优的截断距离参数d_c.

将式 (21) 所示的信息熵函数作为自调节步长果蝇优化算法的浓度判定函数$F_{\text{unction}}(S_i)$进行优化, 求出最优σ值, 即截断距离d_c. 按式 (1)~(3) 计算得到ρ_i和δ_i后, 自适应的选择聚类中心.

聚类中心的ρ_i和δ_i两个值都较大, 本文引入γ_i

$$\gamma_i = \rho_i \times \delta_i \tag{23}$$

则γ_i较大的点, 就很有可能是聚类中心; 换言之, 聚类中心的γ_i一定较大, 可以先挑出γ_i值较大的点, 在从中去选择真正的聚类中心. 将γ_i值按降序排序. 定义临界点P, 表示$\gamma_{[1\sim P]}$与$\gamma_{[P\sim n]}$变化程度最大的点, 本文用γ_i值降序排列图的斜率来表示其变化程度, 则 P 服从以下条件:

$$P = \max\{i \,|\, |k_i| - |k_{i+1}| \| \geq \beta, i = 1,2,\cdots,n-2\} \tag{24}$$

$$\beta = \alpha(j)/(n-2) \tag{25}$$

$$\alpha(j) = \sum_{j=1}^{n-2} \| |k_j| - |k_{j+1}| \| \tag{26}$$

式中, k_i表示第i个点和第$i+1$个点之间的线段的斜率; $\alpha(j)$表示在γ_i值降序排列图中两个邻居点的斜率差的总和; β表示斜率差的平均值; i是斜率差大于等于均值β的点中序号最大的点及临界点.

那么聚类中心就可能存在于式 (27) 表示的范围内, 称这个范围内的点为伪中心.

$$s_{pq} = \{q \,|\, \gamma_q \geq \gamma_p, q = 1,2,\cdots,p\} \tag{27}$$

在同一范围内存在多个伪中心, 因此需要将同一范围内多余的伪聚类中心排除. 在密度峰值算法中, 聚类中心与密度点更高的点相距较远, 因此, 本文取同一区域中的第 1 个伪中心作为聚类中心, 判断其他的伪中心到该点的距离, 若小于$d(x_i, N_k(x_i))$则将其消除; 如果大于, 则将其作为新的聚类中心. 在聚类过程中自适应的根据实际数据选出聚类中心, 聚类中心个数即为最终聚类数目.

1.4 算法流程及复杂度分析

1.4.1 算法流程

算法 1. 基于自调节步长果蝇优化的自适应快速峰值聚类算法

输入: 样本数据集 X, 参数 m, n
输出: 数据的分类标签 y
1) 初始化果蝇群体规模以及最大迭代次数;
2) 将浓度测定值带入信息熵函数, 确定果蝇个体的味道浓度;
3) 按式 (14)~(17) 自调节步长并得到最优σ值, 即截断距离d_c;
4) 计算原始数据的距离矩阵作为输入并根据式 (1)(2)(3) 计算局部密度和到高密度点的距离;
5) 按式 (23)~(27) 选择聚类中心;
6) 分配其他非中心点并完成结果优化;
7) 聚类结果可视化并输出分类标签.

1.4.2 算法复杂度分析

假设果蝇群体大小为 S, 最大迭代次数为 M, 个体浓度跟新计算量为 O(1), 计算浓度差值变化率所需的时间复杂度为 O(M), 并且动态调整步长的时间复杂度为 O(M), 所以计算最优截止距离部分的时间复杂度为 O((N+2)*M). 假设由自调节步长果蝇优化的自适应密度峰值聚类算法 (Adaptive Density Peak Clustering based on Fruit fly Optimization of Self-adjusting step-size, SFO-ADPC) 处理的数据集有 N 个数据点, 则存储每个数据点与其他点的距离的时间复杂度为 O(N^2), 计算每个数据点的局部密度及距离所需的时间复杂度为 O(2N), 计算边界区域需要的时间复杂度为 O(N^2). 综上, 本文所提算法的时间复杂度为 O((N+2)*M+(N+1)*N).

2 实验与结果分析

2.1 实验数据集

为测试所提 SFO-ADPC 算法的准确性和有效性, 本文选择了 5 个人工数据集和 5 个常用于聚类测试的真实数据集. 表 1 中为本文所用数据集, 包括环形、月牙形、聚集、火焰、螺旋形、Iris、Wine 和 Soybean 数据集. 5 个人工数据集的数据分布情况如图 3 所示, 图中数据点横纵坐标均为数值, 无具体物理意义, 故无单位.

聚集和火焰数据是典型的聚类算法性能测试数据, 环形、月牙形和螺旋形数据为测试算法能否准确地对非球形数据聚类的典型数据集. Iris 是鸢尾花卉数据集, 其中有 150 个样本, 属于 3 个类别, 每个类别有 50 个

样本, 每个样本具有 4 个属性, 即花萼长宽以及花瓣长宽. 3 个类别分别为 Setosa, Versicolour 和 Virginica. Wine 是酒数据集, 有 178 个样本, 每个样本有 13 个属性, 属于 3 个类别, 每个类别数量不同. Soybean 数据集是大豆疾病数据集, 包含 47 个数据, 每个数据包含 35 个属性, 分为 4 类, 是线性可分的, 其所有属性都可作为分类属性. Ecoli 数据集是大肠杆菌数据, 由 336 个样本组成, 每个样本有 7 个属性, 分为 8 类. Seeds 数据集包含 210 个样本, 每个样本有 7 个属性, 每个样本分为 3 个类.

表 1 实验数据集

数据集	维度	类簇数	数据量
环形数据	2	2	1000
月牙形数据	2	2	1000
聚集数据	2	7	788
火焰数据	2	2	240
螺旋形数据	2	3	312
Iris 数据	4	3	150
Wine 数据	13	3	178
Soybean 数据	35	4	47
Ecloi 数据	7	8	336
Seeds 数据	7	3	210

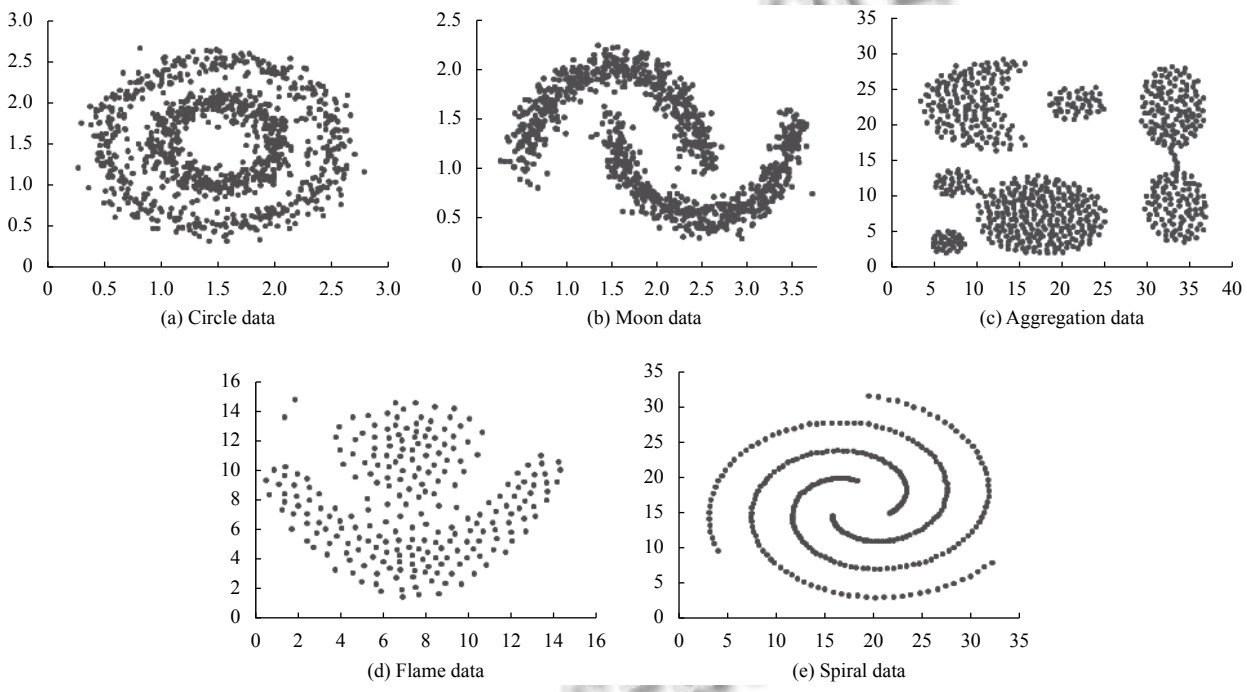

图 3 原始数据集

2.2 实验数据集

本文主要使用两个评价指标来测试所提的算法: 内部轮廓系数 (Silhouette) 和外部 F 值 (F-measure).

1) Silhouette 指标

对于其中的一个样本点i来说, 需要计算两个量: $a(i)$与$b(i)$,

那么i向量轮廓系数就如下式:

$$S(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}} \tag{28}$$

$a(i)$: i向量与同一类中每个点的不相似性的平均值.

$b(i)$: i向量与其他类的点的不相似性的最小平均值.

使用所有数据 S 平均值评估簇的紧密性和可索引性. S 介于–1 和 1 范围内, 对于同一数据集, S 越大聚类越好.

2) F-measure 指标

F-measure 是一种结合准确率和召回率的外部评估指标. 对于真实类P_j与聚类所产生的类C_i, 分别定义准确率$P(i,j)$和召回率$R(i,j)$. 其表示为:

$$P(i,j)=\frac{|P_j\cap C_i|}{|C_i|} \tag{29}$$

$$R(i,j)=\frac{|P_j\cap C_i|}{|P_j|} \tag{30}$$

$P(i,j)$和$R(i,j)$的都介于 0 和 1 之间, 且两者的数值大小与相对应的正确率和召回率成正比.

相应的 F-measure 指标计算:

$$F\left(P_j,C_i\right)=\frac{2\times P\left(P_j,C_i\right)\times R\left(P_j,C_i\right)}{P\left(P_j,C_i\right)+R\left(P_j,C_i\right)} \tag{31}$$

将各类 F-measure 指标作平均则可以求出最终 F-measure 值, 其表示为:

$$F=\sum_j\frac{|P_j|}{N}\max_i F\left(P_j,C_i\right) \tag{32}$$

上述公式中 N 为数据总个数, 计算所得 F-measure 值越大, 则算法准确程度越高.

2.3 实验结果与分析

本文将所提算法 SFO-ADPC 与 FODPC、DPC、DBSCAN、K-Means 在数据集上比较. 图 4 中分别为 K-Means 在环形、月牙形、聚集、火焰、以及螺旋数据集上的聚类结果, 图 5 中分别为 DBSCAN 在环形、月牙形、聚集、火焰、以及螺旋数据集上的聚类结果, 图 6 中分别为 DPC 在环形、月牙形、聚集、火焰、以及螺旋数据集上的聚类结果, 图 7 中分别为 FODPC 在环形、月牙形、聚集、火焰、以及螺旋数据集上的聚类结果, 图 8 中分别为 SFO-ADPC 在环形、月牙形、聚集、火焰、以及螺旋数据集上的聚类结果.

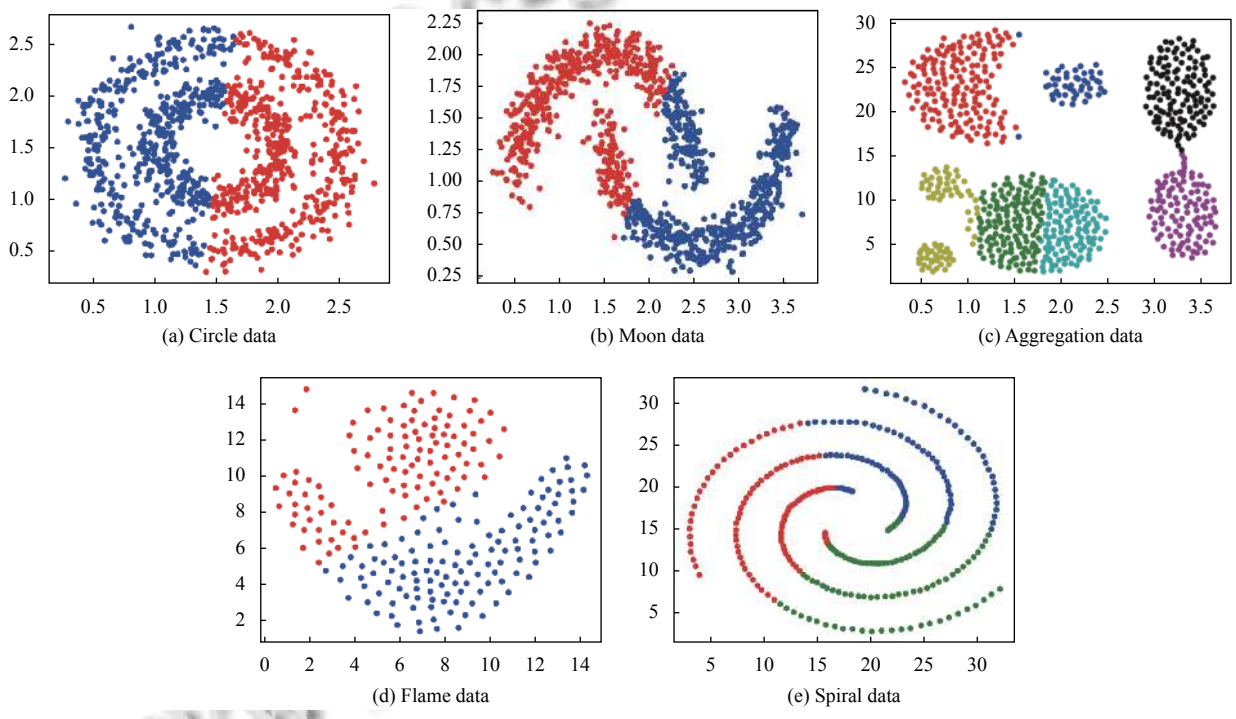

图 4 K-Means 算法聚类结果

测试算法中使用的数据集分布大多为非球形的, K-Means 无法准确对其进行聚类, 同时, 聚集数据集虽然是球形分布的, 但由于样本量分布不均匀, K-Means 仍然无法达到最好的聚类效果. DBSCAN 可以有效地对测试数据集进行聚类, 但其参数确定过程较为繁琐, 不同的数据集的聚类参数相差较大, 用户需要花费大量的时间用于调参.

DPC 及其改进算法可以有效对非球形数据进行聚类, 同时聚类的准确率大大提高, FODPC 及本文提出的 SCFO-ADPC 给出了更好的截断距离, 克服 DPC 需要人工设置截断距离的缺陷; 同时, 本文算法提供的截断距离更好, 提升了聚类准确率. 并且, 本文所提算法能够自适应的选择聚类中心, 改善了原始 DPC 算法需要手动选取聚类中心的缺陷.

表 2 中列出了上述 5 种算法在 5 个人工数据集及 5 个真实数据集的聚类效果评价得分.

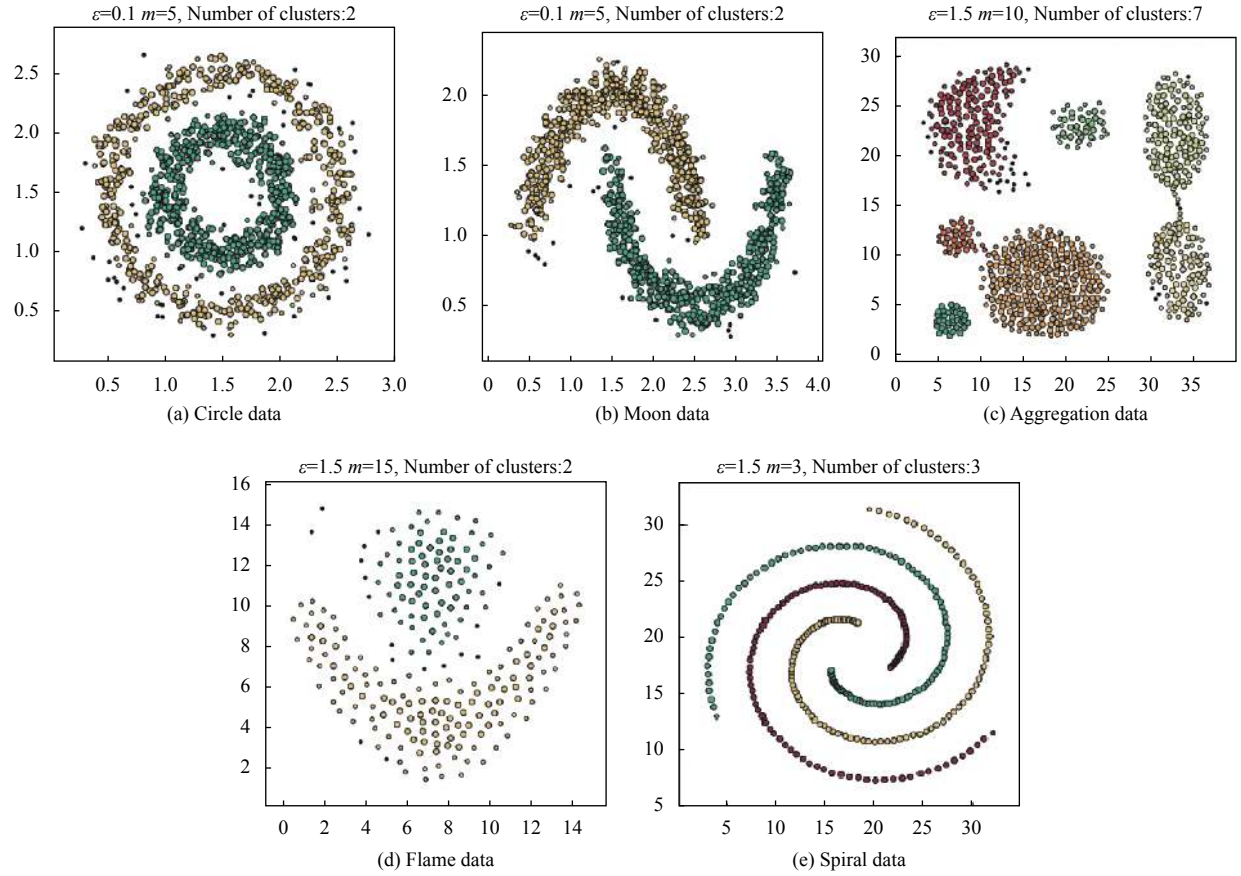

图 5 DBSCAN 算法聚类结果

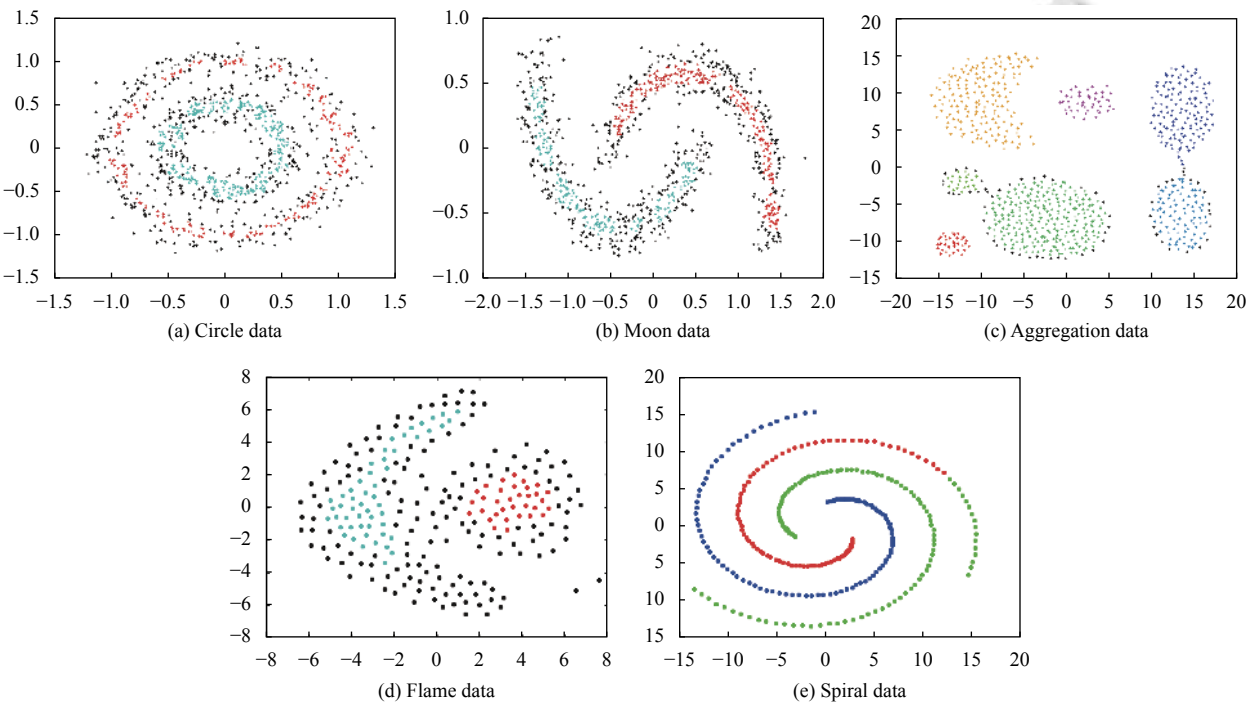

图 6 DPC 算法聚类结果

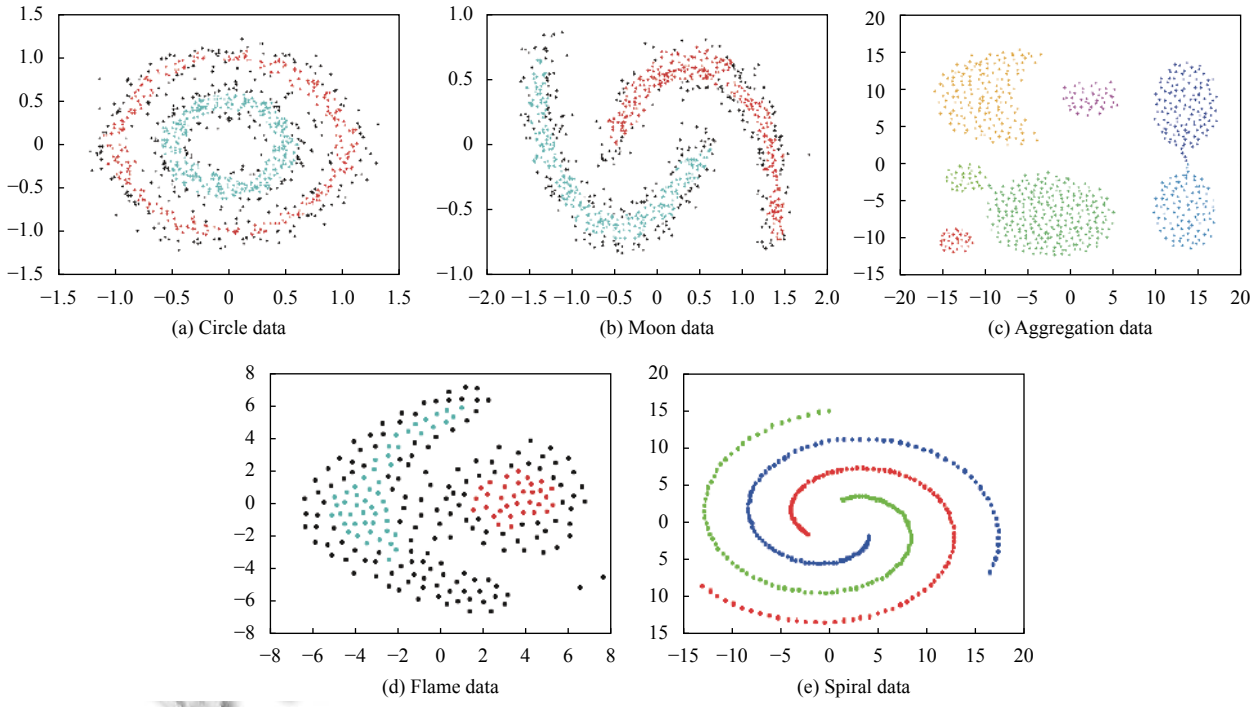

图 7 FODPC 算法聚类结果

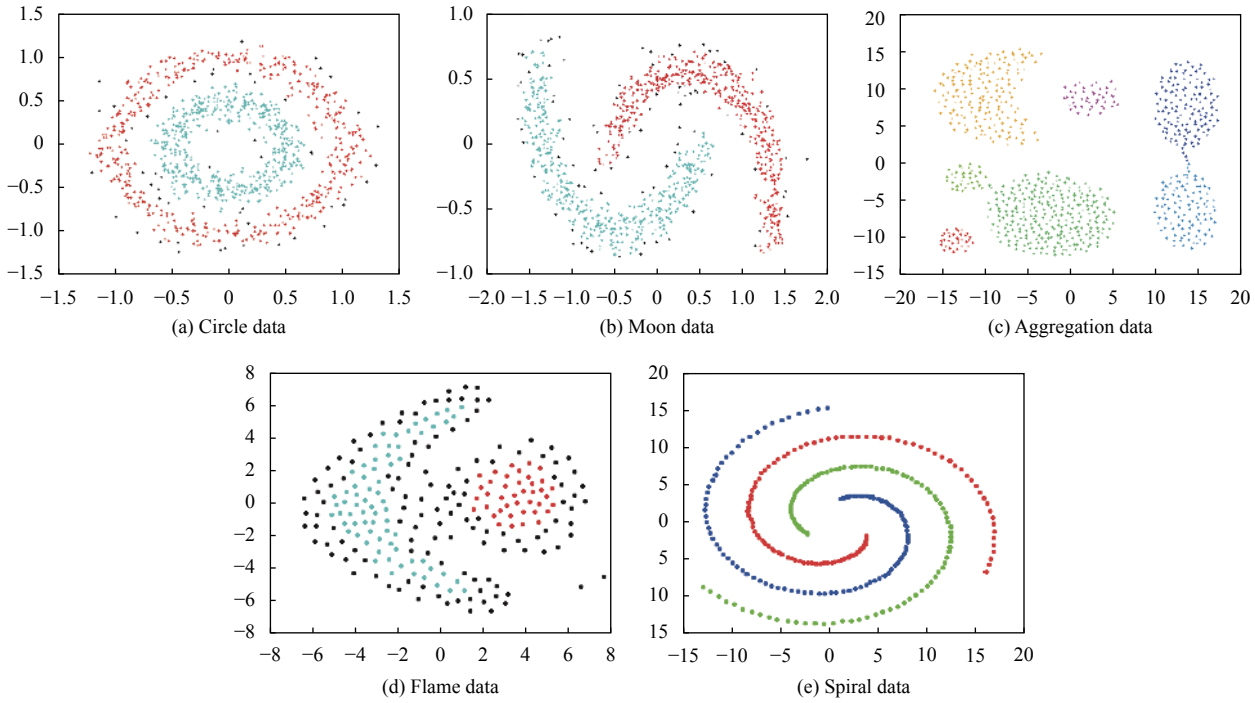

图 8 SCFO-ADPC 算法聚类结果

上述 5 种聚类算法应用于实际数据时, SCFO-ADPC 的准确率及召回率更高, 本文所提算法能有效的对任何形状的数据进行聚类, 对于具有较高维度的数据集也可有效进行聚类, 并且聚类效率及效果有一定提高.

表 2 聚类结果评价

数据集	性能	K-Means	DBSCAN	DPC	FODPC	SCFO-ADPC
环形数据	S	0.349 556	0.241 050	0.250 468	0.248 161	0.256 485
	F	0.501 000	0.946 000	0.912 000	0.958 000	0.984 000
月牙形数据	S	0.484 504	0.228 626	0.285 164	0.305 468	0.254 654
	F	0.751 000	0.981 000	0.729 000	0.985 000	0.988 000
聚集数据	S	0.480 982	0.479 736	0.456 456	0.465 774	0.416 489
	F	0.803 299	0.965 736	0.997 462	1	1
火焰数据	S	0.378 494	0.199 965	0.154 596	0.187 913	0.201 564
	F	0.837 500	0.925 000	0.754 100	0.781 569	0.804 561
螺旋形数据	S	0.360 163	0.134 429	0.126 546	0.156 464	0.156 448
	F	0.346 154	1	0.803 679	1	1
Iris 数据	S	0.680 813	0.686 393	0.751 657	0.751 964	0.781 640
	F	0.893 333	0.666 667	0.906 667	0.931 567	0.944 564
Wine 数据	S	0.571 138	0.564 847	0.516 489	0.648 941	0.601 654
	F	0.696 629	0.231 068	0.703 567	0.729 546	0.731 649
Soybean 数据	S	0.320 398	0.523 664	0.284 684	0.305 645	0.315 468
	F	0.723 404	0.255 3119	0.893 617	0.924 682	0.957 892
Ecoli 数据	S	0.273 145	0.074 971	0.562 589	0.645 305	0.683 154
	F	0.616 071	0.399 2857	0.809 524	0.824 682	0.857 892
Seeds 数据	S	0.471 933	0.250 138	0.501 382	0.530 645	0.544 689
	F	0.895 238	0.419 048	0.895 476	0.914 862	0.919 278

3 结语

本文提出了一种基于自调节步长果蝇优化的自适应密度峰值聚类算法 (SCFO-DPC). SCFO-DPC 算法通过可自调节步长的果蝇优化算法得到最佳截止距离, 解决了需要根据人工设置截止距离参数的问题, 并且得到的参数优于其他优化算法; 能够自适应的选择聚类中心, 有效的解决了人工选择聚类中心为聚类算法带来的不稳定性. 通过测试, SCFO-DPC 算法在对人工数据集、UCI 真实数据集的处理上具有相当大的优越性, 比 FODPC、DPC、DBSCAN 和 K-Means 算法更准确有效, 且受人为影响更少.

参考文献

1 孙吉贵, 刘杰, 赵连宇. 聚类算法研究. 软件学报, 2008, 19(1): 48–61.

2 Kumar N, Verma V, Saxena V. Cluster analysis in data mining using k-means method. International Journal of Computer Applications, 2013, 76(12): 11–14. [doi: 10.5120/13298-0748]

3 贺玲, 吴玲达, 蔡益朝. 数据挖掘中的聚类算法综述. 计算机应用研究, 2007, 24(1): 10–13. [doi: 10.3969/j.issn.1001-3695.2007.01.003]

4 Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492–1496. [doi: 10.1126/science.1242072]

5 蒋礼青, 张明新, 郑金龙, 等. 快速搜索与发现密度峰值聚类算法的优化研究. 计算机应用研究, 2016, 33(11): 3251–3254.

6 王华秋, 聂珍. 快速搜索密度峰值聚类在图像检索中的应用. 计算机工程与设计, 2016, 37(11): 3045–3050, 3057.

7 张萌, 吕艳, 倪益华, 等. 基于密度峰值聚类的随机森林室内定位. 计算机工程与设计, 2018, 39(5): 1490–1496.

8 赵怀鑫, 邓然然, 张英杰, 等. 一种用于高速公路通行情况分析的收费数据挖掘方法. 中国公路学报, 2018, 31(8): 155–164. [doi: 10.3969/j.issn.1001-7372.2018.08.017]

9 赵怀鑫, 张英杰, 邓然然, 等. 基于快速峰值聚类的高速公路异常事件识别方法. 长安大学学报 (自然科学版), 2018, 38(5): 205–212.

10 王海燕, 李晓玲. 聚类评价在聚类算法选择中的应用研究. 福建电脑, 2015, 31(3): 26–28. [doi: 10.3969/j.issn.1673-2782.2015.03.014]

11 杨震, 王红军, 周宇. 一种截断距离和聚类中心自适应的聚类算法. 数据分析与知识发现, 2018, 2(3): 39–48.

12 Xie JY, Gao HC, Xie WX, *et al*. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors. Information Sciences, 2016, 354: 19–40. [doi: 10.1016/j.ins.2016.03.011]

13 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 曾嘉豪. 基于 K 近邻的密度峰值聚类算法. 中国国际财经 (中英文), 2018, (5): 190–191.

15 Xiao WC, Yang Y, Xing HL, *et al*. Clustering algorithm

based on fruit fly optimization. Proceedings of 10th International Conference on Rough Sets and Knowledge Technology. Tianjin, China. 2015. 408–419.
16 周瑞红. 基于群智能优化理论的聚类改进方法及应用研究[博士学位论文]. 长春: 吉林大学, 2017.
17 于婷. 密度峰值聚类算法的若干改进及其应用[硕士学位论文]. 长春: 吉林财经大学, 2017.
18 郑虹, 周丽媛, 韩旭明. 布谷鸟优化的密度峰值快速搜索聚类算法. 长春工业大学学报, 2018, 39(3): 253–260.
19 Mehmood R, Zhang GZ, Bie RF, *et al*. Clustering by fast search and find of density peaks via heat diffusion. Neurocomputing, 2016, 208: 210–217. [doi: 10.1016/j.neucom.2016.01.102]
20 钱雪忠, 金辉. 自适应聚合策略优化的密度峰值聚类算法. 计算机科学与探索. http://kns.cnki.net/kcms/detail/11.5602.TP.20190418.1746.022.html. (2019-05-15).
21 潘文超. 应用果蝇优化算法优化广义回归神经网络进行企业经营绩效评估. 太原理工大学学报 (社会科学版), 2011, 29(4): 1–5.
22 吴小文, 李擎. 果蝇算法和 5 种群智能算法的寻优性能研究. 火力与指挥控制, 2013, 38(4): 17–20, 25. [doi: 10.3969/j.issn.1002-0640.2013.04.005]
23 丁国绅, 邹海. 新型改进果蝇优化算法. 计算机工程与应用, 2019, 52(21): 168–174.
24 桂龙, 王爱平, 丁国绅. 改进步长与策略的果蝇优化算法. 计算机工程与应用, 2018, 54(4): 148–153, 184. [doi: 10.3778/j.issn.1002-8331.1609-0141]
25 朱富占, 邹海, 丁国绅. 改进的变步长果蝇优化算法. 微电子学与计算机, 2018, 35(6): 36–40.
26 盛超, 邹海, 朱富占. 一种新型自调节步长果蝇优化算法. 微电子学与计算机, 2019, 36(2): 62–67.