文章编号:1671-9352(2024)01-0046-10 DOI:10.6040/j.issn.1671-9352.0.2022.623

基于非洲秃鹫优化算法改进的密度峰值聚类

罗兴隆[1],贺兴时[1*],周洁[1],杨新社[2]
(1.西安工程大学理学院, 陕西 西安 710600; 2.密德萨斯大学科学与技术学院, 英国 剑桥 CB2 1TN)

摘要:密度峰值聚类算法是一种自动寻找簇中心的新型快速搜索算法。针对其截断距离的不确定和一步式分配策略不稳健等缺陷,本文提出一种基于非洲秃鹫优化算法改进的密度峰值聚类算法。通过准确率(accuracy, Acc)这一评价指标建立优化问题的目标函数,利用非洲秃鹫优化算法强大的寻优能力对不确定的截断距离 d_c 进行优化,降低了人为取值的不准确性;其次,根据数据集密度均值将其划分为高低不同的密度区域,对不同区域采用不同的分配策略,针对高密度区域内的数据点采用与原密度峰值聚类相同的分配方法,对低密度区域内数据点则根据其 k 近邻数量进行聚类;最后,将该算法在合成和真实数据集上进行实验验证,算法的聚类性能有了很大的提升,且对密度差异性较大的数据集划分也更加精确。

关键词:非洲秃鹫优化算法;密度峰值;截断距离;分配策略
中图分类号:TP301 **文献标志码**:A
引用格式:罗兴隆,贺兴时,周洁,等.基于非洲秃鹫优化算法改进的密度峰值聚类[J]. 山东大学学报(理学版),2024,59(1):46-55,71.

Improved density peak clustering approach based on African vultures optimization algorithm

LUO Xinglong[1], HE Xingshi[1*], ZHOU Jie[1], YANG Xinshe[2]
(1. School of Science, Xi'an Polytechnic University, Xi'an 710600, Shannxi, China; 2. School of Science and Technology, Middlesex University, Cambridge CB2 1TN, UK)

Abstract: Density peak clustering algorithm is a new fast search algorithm for automatically finding cluster centers. Aiming at the uncertainty of its cut-off distance and the instability of the one-step allocation strategy, an improved density peak clustering approach based on African vultures optimization algorithm is proposed. The objective function of the optimization problem is established through evaluating accuracy(Acc), and the uncertain cut-off distance d_c is optimized by the powerful optimization ability of the African vultures optimization algorithm, which reduces the inaccuracy of artificial values. Secondly, according to the average density of the data set, it is divided into different density areas, and different allocation strategies are used for different areas. For data points in the high-density area, the same allocation method as the original density peak clustering is used, and for data points in the low-density area, the *k*-nearest neighbor method is used for clustering. Finally, the algorithm is experimentally verified on synthetic and real data sets, the clustering performance of the algorithm has been greatly improved, and the division of data sets with large density differences is also more accurate.

Key words: African vultures optimization algorithm; density peak; cut-off distance; allocation strategy

0 引言

聚类是将有限的未标记对象分离为具有簇类相似和簇外相离的一种方法[1],已被广泛应用于机器学

收稿日期:2022-11-17; **网络出版时间**:2023-11-15 16:37:44
网络出版地址:https://link.cnki.net/urlid/37.1389.N.20231114.1635.002
基金项目:国家自然科学基金资助项目(12101477); 陕西省教育厅自然科学基金资助项目(21JK0654)
第一作者简介:罗兴隆(1997—),男,硕士研究生,研究方向为聚类分析、智能计算算法. E-mail:lxl_bert@ 163.com
*通信作者简介:贺兴时(1960—),男,教授,硕士,研究方向为优化算法及应用、数理统计等. E-mail:xsh1002@ 126.com

习、人工智能、模式识别、数据分析、图像分割[2]等领域。2014 年,Rodriguez 和 Laio[3]在《科学》杂志上提出了一种基于密度峰值的新型聚类算法(density peak clustering algorithm, DPC)。与其他经典的基于密度方法相比,DPC 在识别簇中心方面具有极大优势,选择截断距离 d_c 作为唯一参数,能聚类不同大小密度和任意形状的簇。

近些年,DPC 被广泛应用于离群值检测、图像处理、文档处理等[4],但仍然存在截断距离的选取不当和非簇中心数据点的分配不准确等问题,大量学者针对问题进行了相应的改进。2016 年,Xie 等[5]提出了基于模糊加权 k 近邻的密度峰值搜索和点分配算法,该算法通过 k 近邻思想来定义点的局部密度,在分配非簇中心阶段采取 2 种不同的策略,有效地解决了数据集中密度测量不均匀的问题。Mehmood等[6]提出了一种对截断距离参数灵敏度较低的非参数密度估计方法,通过热核方程来估计密度,使参数的敏感性大大降低。Tang 等[7]提出了一种增强的快速密度峰的聚类算法(E-FDPC),通过引入一个参数来控制归一化局部密度和簇内距离之间的权重,以此来提高聚类的效率。Wang 等[8]提出了一种利用原始数据集数据场的熵自动选取最优截断距离的方法,对于需要聚类的数据集,截断距离不再依赖经验估计,而是客观地从数据集中计算。

为了准确选取簇中心点和截断距离,Jiang 等[9]提出了基于基尼系数和 k 近邻计算截断距离的算法(G-KNN-DPC),该算法使用基尼系数找到最优的截断距离,但对于高维数据集却表现不理想。文献[10]提出了一种利用测地线距离进行密度峰聚类(DPC-GD),将测地线距离的思想引入到原始的 DPC 中。Wang 等[11]提出了一种具有自适应密度峰值检测的聚类方法,通过非参数多元核估计来计算局部密度。Du 等[12]提出了一种基于 k 近邻的密度峰聚类方法(DPC-KNN),该方法引入 k 近邻的思想,使用一种新的局部密度度量方式,并在高维数据中将主成分分析(PCA)引入算法模型中。文献[13]提出了一种自动选取聚类中心的算法,该算法引入 KL 散度的差异性度量准则实现非簇中心数据点的分配,通过 KL 排序图中的拐点自动完成簇中心的选取。

为了克服一步分配策略的缺陷,文献[14]基于 k 近邻的思想改进了数据点分配策略,文中使用多步分配策略法来对算法进行优化。文献[15]使用加权局部密度序列和两阶段分配策略的方法,并通过加入最近邻动态表来提高聚类效率。文献[16]提出了一种基于图标签传播的方法,算法采用 k 近邻的思想来计算各点的全局截断距离和局部密度。文献[17]提出了一种基于共享最近邻的快速搜索密度峰值算法(SNN-DPC),该算法重新定义了局部密度和相对距离,并引入了样本点之间的共享邻居和局部密度信息,但计算复杂度高,对多密度峰的簇分类也不准确。为了自动选取簇中心,文献[18]提出了一种基于共享最近邻和自适应聚类中心的快速搜索密度峰值聚类算法(DPC-SNNACC),它可以根据不同数据集的特征,自动确定决策图中膝关节点的数量,根据膝关节数量确定聚类中心。文献[19]提出了一种新的自适应聚合策略算法,引入了 KNN 的思想计算局部密度和一种密度可达的合并策略,但该算法在复杂结构数据集上效果不佳。

为了降低受密度核和密度差异的影响,文献[20]采用基于相对密度作为识别聚类中心的标准。文献[21]提出了一种基于层次方法的新聚类算法,使用切割的方法将具有代表性的数据点自动分类为不同的密度水平。

以上改进算法相较于 DPC 虽然有较好的表现,但对 DPC 截断距离难选取和一步分配策略不稳定等缺点依旧没有彻底解决,为了克服以上问题的不足,本文提出了基于非洲秃鹫优化算法改进的密度峰值聚类(improved density peak clustering approach based on African vultures optimization algorithm, DPC-AVOA),该算法首先根据非洲秃鹫优化算法(AVOA)强大的寻优能力,通过建立以准确率(accuracy, Acc)为目标函数,利用 AVOA 不断优化目标函数,找到最好的截断距离;其次,在分配非簇中心数据点时,根据局部密度的均值将数据集划分为 2 种密度区域,针对高密度区域数据点按照 DPC 相似的分配原则,而对低密度区域数据点则根据其 k 近邻的数量属于相应簇的数量进行分配,从而实现对不同密度范围内的数据点准确地划分到合适的簇。

1　背景知识

1.1　DPC

DPC[3]的思想为假设簇中心周围邻居点的局部密度较低,并且它们与任何局部密度较高的点都有较大

的距离,对于任意的样本点 x_i,该算法计算它的局部密度 ρ_i 和与它较高密度点的距离 δ_i,数据点的局部密度 ρ_i 和 δ_i 只依赖于数据点间的距离。数据点 x_i 的局部密度 ρ_i 计算方式有 2 种,即高斯核(Gaussian kernel)和截断核(cut-off kernel)。分别如下:

cut-off kernel:
$$\rho_i=\sum_{j\neq i}\chi(d_{ij}-d_c); \tag{1}$$

Gaussian kernel:
$$\rho_i=\sum_{j\neq i}\exp(-(d_{ij}/d_c)^2)。 \tag{2}$$

对于每个数据点 i 和与之相邻数据点之间的距离 δ_i,计算方式如下:
$$\delta_i=\min_{j:\rho_j>\rho_i}(d_{ij})。 \tag{3}$$

对于密度最高的数据点,计算为
$$\delta_i=\max_{i\neq j}(\delta_j)。 \tag{4}$$

计算出每个数据点的 ρ_i 和 δ_i 之后,在 DPC 中,将按照同时具有更大 ρ_i 和 δ_i 乘积的数据点确定为簇中心,即使用式(5)计算出每个数据点 x_i 的决策值 γ_i,然后排序选择簇中心。
$$\gamma_i=\rho_i\cdot\delta_i。 \tag{5}$$

1.2 AVOA

AVOA 的基本思想[22]是模拟秃鹫在自然界中通过群体狩猎食物来摆脱饥饿状态,是一种新型的元启发式算法,具有较低的计算复杂度且寻优过程更加灵活,主要分为 4 个阶段。

(1) 确定最优秃鹫阶段。形成初始种群并计算所有解的适应度,选择最佳解作为第一组的最优秃鹫,选择次优解作为第二组的最优秃鹫,其他解利用式(6)向各自组的最优解移动。
$$R(i)=\begin{cases}\text{best}\,V_1, & q_i=L_1,\\ \text{best}\,V_2, & q_i=L_2。\end{cases} \tag{6}$$

$$q_i=F_i/\sum_{i=1}^{n}F_i。 \tag{7}$$

式(6)中 L_1 和 L_2 是 0~1 的参数,2 个参数的值之和为 1。式(7)为其他秃鹫属于最优组或次优组概率的计算方式。

(2) 秃鹫饥饿率阶段。秃鹫通过自身的饥饿状态来寻找食物,如果它们处于饱足状态,就有更高的能量去更远的地方寻找食物,但当处于饥饿状态时,则没有充足能量长时间飞行,只能跟随强壮的秃鹫寻找食物,同时变得很有进攻性。该行为方式使用式(8)、(9)进行计算:
$$\lambda=a\cdot\left(\sin^w\left(\frac{\pi\cdot\text{iter}_i}{2\cdot\text{Miter}}\right)+\cos\left(\frac{\pi\cdot\text{iter}_i}{2\cdot\text{Miter}}\right)-1\right), \tag{8}$$

$$F=(2\cdot\text{rand}_1+1)\cdot b\cdot\left(1-\frac{\text{iter}_i}{\text{Miter}}\right)+\lambda, \tag{9}$$

其中,w 是初始固定的参数;a 是[-2,2]之间的随机数;iter_i 是当前迭代次数;Miter 是总迭代次数;rand_1 是 0~1间的随机数;b 是[-1,1]之间的随机数,如果 b 值小于 0 时,代表秃鹫饥饿了,当 b 值增加到 0 时,意味着秃鹫是饱足状态。

(3) 探索阶段。如果$|F|\geqslant 1$,算法进入探索阶段,该阶段有 2 个不同的策略方式,通过设置 0~1 的参数 f_1 来选择,在该阶段初始,生成一个 0~1 的随机数 $\text{rand}\,f_1$,$\text{rand}\,f_1$ 通过与 f_1 比较确定探索方式,过程如式(10)所示。
$$P(i+1)=\begin{cases}R(i)-|X\cdot R(i)-P(i)|\cdot F, & f_1\geqslant\text{rand}\,f_1;\\ (i)-F+\text{rand}_2\cdot((ub-lb)\cdot\text{rand}_3+lb), & f_1<\text{rand}\,f_1,\end{cases} \tag{10}$$

其中:$P(i+1)$是秃鹫在下一次的迭代位置;$P(i)$是秃鹫当前的矢量位置;$R(i)$是最好的秃鹫之一;X 是秃鹫随机移动的地方,在每次迭代中发生变化,使用等式 $X=2\cdot\text{rand}$ 得到,rand 是 0~1 的随机数;$\text{rand}\,f_1$、rand_2 和 rand_3 为 0~1 的随机值,lb 和 ub 为变量的上界和下界。

(4) 开发阶段。如果$|F|<1$,算法进入开发阶段。该阶段又进而分为 2 个阶段,各阶段使用不同的策略。

当|F|的值满足 0.5≤|F|<1 时,算法进入开发的第一阶段。在该阶段,执行旋转飞行和围攻作战策略,f_2 用于第一阶段的选择参数,rand f_2 是初始生成 0~1 的随机参数,通过比较 f_2 与 rand f_2 确定开发方式,该过程如式(11)所示。

$$P(i+1)=\begin{cases}|X\cdot R(i)-P(i)|\cdot(F+\text{rand}_4)-d(t), & f_2\geqslant \text{rand}\, f_2;\\ R(i)-Z_1-Z_2, & f_2<\text{rand}\, f_2。\end{cases} \tag{11}$$

$$d(t)=R(i)-P(i), \tag{12}$$

$$Z_1=R(i)\cdot\left(\frac{\text{rand}_5\cdot P(i)}{2\pi}\right)\cdot\cos P(i), \tag{13}$$

$$Z_2=R(i)\cdot\left(\frac{\text{rand}_6\cdot P(i)}{2\pi}\right)\cdot\sin P(i), \tag{14}$$

在式(11)—(14)中:秃鹫与 2 组中最优秃鹫之间的距离可以通过该矢量位置得到;rand_4、rand_5 和 rand_6 是 0 到 1 之间的随机数。

当|F|<0.5 时,则进入第二阶段。f_3 是第二阶段选择的参数,在该阶段初始,生成 1 个 0~1 的随机数 rand f_3,通过比较 f_3 与 rand f_3 来确定开发策略,该过程如式(15)所示。

$$P(i+1)=\begin{cases}(W_1+W_2)/2, & f_3\geqslant \text{rand}\, f_3;\\ R(i)-|d(t)|\cdot F\cdot\text{levy}(d), & f_3<\text{rand}\, f_3。\end{cases} \tag{15}$$

$$W_1=\text{best}\, V_1(i)-\frac{\text{best}\, V_1(i)\cdot P(i)}{\text{best}\, V_1(i)-P(i)^2}\cdot F, \tag{16}$$

$$W_2=\text{best}\, V_2(i)-\frac{\text{best}\, V_2(i)\cdot P(i)}{\text{best}\, V_2(i)-P(i)^2}\cdot F, \tag{17}$$

其中:best $V_1(i)$ 为当前迭代第一组中最优秃鹫;best $V_2(i)$ 为当前迭代第二组中最优秃鹫;$d(t)$ 通过式(12)得出,表示秃鹫到 2 组最好秃鹫之一的距离;Levy(d)表示莱维飞行[23]。

2　基于非洲秃鹫优化算法改进的密度峰值聚类

为了克服原 DPC 在人为选择 d_c 时的不确定和数据点的分配不当问题,本文主要针对截断距离和分配策略进行改进。首先,利用非洲秃鹫优化算法的寻优快、效率高等特性,优化 DPC 中的截断距离 d_c;其次,在对 DPC 分配非簇中心点时进行了改进,将一步分配策略改为 2 种不同的分配方式实现。

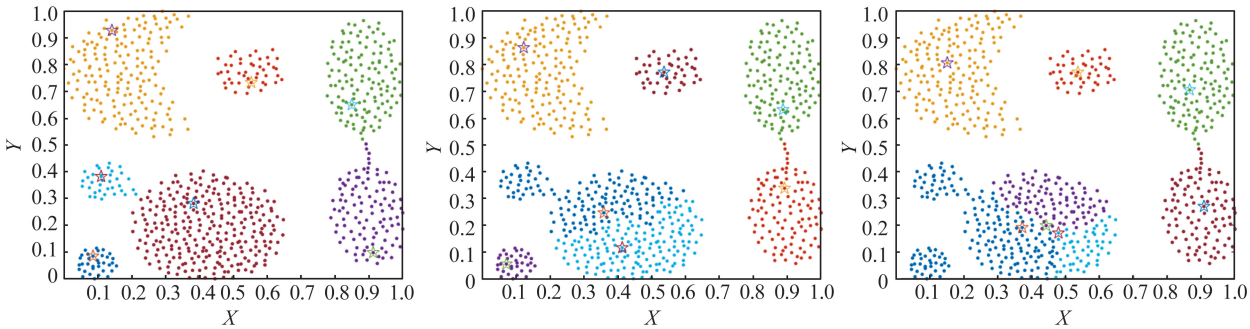
2.1　截断距离 d_c 的选取原则

在 DPC 中,作者[3]给出了 d_c 的取值为:$d_c=d_{[\text{Num}\times p\%]}$,其中 Num=$N(N-1)/2$,$N$ 为数据集的样本点数,p 的取值范围为 1-2,[Num×p%]为 $d_{[\text{Num}\times p\%]}$ 的下标,$R=[d_1,d_2,\cdots,d_{\text{Num}}]$ 为数据集中升序排列的每任意两点间全部距离的集合,$d_{[\text{Num}\times p\%]}$ 是属于 R 的元素。这种取值方式需要人为确定,当取不同参数时聚类结果的波动比较大,受参数影响较为严重。图 1 为 DPC 在 Aggregation 数据集上取不同 d_c 时的聚类图,可以看出,在该数据集中,对于 d_c 不同取值得到的聚类结果差异也较大。当 d_c=0.04 时,聚类能得到正确簇中心,然而当 d_c 取 0.09 和 0.13 时,对左下角 3 个簇却无法得到正确聚类。主要原因一方面是由于 d_c 值的大小使得计算得数据点的局部密度不同,影响了簇中心的寻找;另一方面,原始 DPC 采取一步式的分配策略,从而使得如果一个数据点分配错误,导致其周围数据点也分配不佳,因此,对 DPC 的 d_c 优化变得极为重要。

由于 AVOA[22]有较强的寻优性能,本文利用该特性实现对 d_c 的选取,实现过程中使用 AVOA 对聚类数据集进行多次迭代寻优。主要使用聚类性能指标 Acc[10]来作为优化算法的目标函数,找出能使 Acc 最佳时对应的截断距离 d_c,依次进行,直到满足条件为止。Acc 指标是一种数据信息检索的评价指标,很好地实现对聚类结果的评价,取值范围为 0~1,值越大聚类效果越好,其计算公式如下:

$$\text{Acc}=\frac{1}{N}\sum_{i=1}^{N}\delta(\text{map}(y_i),l_i), \tag{18}$$

其中：l_i 为数据集的真实标签；y_j 为数据集预测标签，N 为样本个数；map(·)是通过匈牙利算法将每个聚类标签映射到 1 个类别标签；$\delta(x,y)$ 为指示函数，如果 $x=y$，则 $\delta(x,y)=1$，否则，$\delta(x,y)=0$。



(a) d_c=0.04 (b) d_c=0.09 (c) d_c=0.13

图 1 不同 d_c 在 DPC 上的聚类效果

Fig.1 Clustering effects different d_c on DPC

2.2 不同区域数据点的分配策略

在原始 DPC 的分配非簇中心数据点方式中，采取一步式分配方法，在面对到多个峰值聚类簇或交叉缠绕数据集时，往往会出现将位于不同簇的数据点划分为同一簇，这样将导致聚类效果不理想，因此，本文将采用 2 种不同分配策略来分配非簇中心数据点。

首先，计算出全部数据点的局部密度，然后通过式(19)计算出局部密度的均值 ρ_{mean}，以此为分界线将数据集划分为高密度区和低密度区，再采用 2 种分配策略分别对数据点进行分配，使每个数据点都能划分到正确的簇。

$$\rho_{\text{mean}}=\frac{1}{N}\sum_{i=1}^{N}\rho_i \tag{19}$$

对于高密度区域数据点，将采用与 DPC 类似的方法，分配原则为优先分配密度较大的数据点，然后再分配密度较小的数据点。在簇中心的选取方面，通过计算该区域所有数据点的 ρ 和 δ 值，构造出 γ 决策图，然后按照排序选取出簇中心。

与高密度区域数据点的分配策略不同，对于低密度区域内数据点，我们引入了数据点的 k 近邻数[14]，分配策略则需要按照其 k 近邻的数量属于相应簇的数量进行分配，分配策略如下：

算法 1 低密度区域数据点分配

输入 低密度区域数据集 $\boldsymbol{X}=\{x_1,x_2,\cdots,x_{N-m}\}$ 和聚类数 c，近邻数 k。

输出 低密度区域数据点的分类。

步骤 1 计算区域内每一个未分配数据点 $x_i(i=1,2,\cdots,N-m)$ 的 k 近邻集合 knn(x_i)，接下来统计 knn(x_i)中属于每个类簇 $C(C=1,2,\cdots,c)$ 的样本数 M_i^C(M_i^C 表示样本 x_i 的 k 近邻数属于类簇 C 中的数量)，依次得到数据点 x_i 属于每个类簇的 k 近邻集合 $\boldsymbol{M}_i\in R^{1\times c}$($\boldsymbol{M}_i$ 为 1 行 c 列的向量)，最后对区域内所有数据点统计 $\boldsymbol{M}_i$，构成判断矩阵 $\boldsymbol{M}(\boldsymbol{M}\in R^{(N-m)\times c})$，$\boldsymbol{M}$ 的行对应每个需要分配数据点，列对应得到聚类中心。

步骤 2 从判断矩阵 $\boldsymbol{M}$ 中选择出每一行中最大值对应的簇，将该行的最大值对应的样本分配给对应的簇，依次对每个数据点进行类簇分配。

步骤 3 如果还有未分配数据点，更新矩阵 $\boldsymbol{M}$，再转步骤 2，否则结束该分配策略。

2.3 本文算法的步骤

输入 数据集 $\boldsymbol{X}=\{x_1,x_2,\cdots,x_N\}$，聚类数 c。

输出 聚类簇标签。

步骤 1 设置 AVOA 的最大迭代次数 T、种群数 p、d_c 的取值范围。

步骤 2 初始化 AVOA 的种群位置(截断距离 d_c)。

步骤 3 求出数据集中所有样本的局部密度 ρ_i 和相对距离 δ_i，并代入式(5)，选取前 c 个 γ 值最大的数据点，得到聚类簇中心位置。

步骤 4　根据先分配高密度区域数据点,然后根据算法 1 对低密度区域数据点进行聚类划分,得到当前 d_c 对应的聚类结果,使用 Acc 作为评价指标,并将其设置成 AVOA 的目标函数,记录当前 d_c 对应目标函数的值,通过比较与上一代的值,找出 Acc 最优时对应的 d_c。

步骤 5　通过 AVOA 的寻优过程来迭代更新 d_c。

步骤 6　判断 AVOA 是否达到终止条件,如果是,则结束迭代,转步骤 7;否则,转步骤 3 继续寻找。

步骤 7　将选取的最优 d_c 对应的结果输出,实现最终聚类结果。

2.4　算法的复杂度分析

已知样本点的个数为 N,最近邻数为 k,高密度区域的样本数为 m,迭代次数为 T,种群大小为 p,算法的复杂度主要由以下几部分构成:

(1) 计算数据集中两两数据点间的距离,时间复杂度为 $O(N^2)$。

(2) 计算所有数据点彼此间的最近邻数,时间复杂度为 $O(N^2)$。

(3) 计算高密度区域中每个数据点的局部密度和相对距离的复杂度为 $O(km+m^2)$。

(4) 计算距离最近的较大密度点距离的时间复杂度为 $O(N^2)$。

(5) 低密度区域数据点分配的时间复杂度为 $O((N-m+k)\cdot N^2)$。

(6) DPC-AVOA 算法在迭代寻找 d_c 过程中,时间复杂度为 $O(p\cdot T)$,优化过程中,会每次重新计算局部密度和相对距离,算法的时间复杂度主要来源这一部分,为 $O(N^2\cdot T)$。

综上,本文算法的所有时间复杂度约为 $O((N-m+k)\cdot N^2\cdot T+m^2\cdot T+p\cdot T)$。

3　实例分析

为了检验本文算法的性能,选取了 9 个合成数据集和 9 个真实数据集[24-30],分别与 *K*-means、DBSCAN (density-based spatial clustering of applications with noise)、DPC[3] 和 DPCSA[15] 算法作对比分析,其中 *K*-means的聚类结果为计算 10 次后取的平均值。采取 Ami、Ari、Fmi 作为聚类性能的评价指标,这些指标的最大值为 1,值越高聚类效果越好,越接近 0,效果越差。表 1 为相关数据集的详细信息,表 2 和表 3 为 4 种算法在所有数据集下的相关性能对比值, 其中每个数据集在某项指标下的最优值进行了加粗。

表 1　所有数据集的信息

Table 1　Information for all datasets

类型	数据集名称	样本量	维度	簇
合成数据集	Flame	240	2	2
	Aggregation	788	2	7
	Spiral	312	2	3
	D31	3 100	2	31
	Jain	373	2	2
	R15	600	2	15
	Three-circle	299	2	3
	Compound	399	2	6
	A3	266	2	3
真实数据集	Ionosphere	351	34	2
	Libras	360	90	15
	Seeds	210	7	3
	Iris	150	4	3
	Ecoli	336	8	8
	Wine	178	13	3
	Pima	768	8	2
	WDBC	569	30	2
	Waveform (noise)	5 000	40	3

图 2 是在 Jain 数据集上对比算法的聚类可视图。Jain 数据集包含 2 个有折叠和弯曲簇的类,当采用 DPC 时,由于人为选择截断距离的缺陷,导致不能很好地找到 d_c。另外,DPC 的单一分配策略,导致在分配非中心数据点时误将下簇的左端分配给上簇。从 DBSCAN 算法效果图看出,它虽然可以准确地识别 2 个簇,却误将上簇的左端划分为一个新的簇。传统 *K*-means 算法得到的聚类效果最差,它将下簇的左右两端部分数据点分给了上簇,这是由使用欧氏距离和传统的分配方式导致的。反观本文算法的聚类效果图,由于截断距离的优化和对一步分配策略的改进,使算法能很好地对该数据集中数据点实现正确聚类。

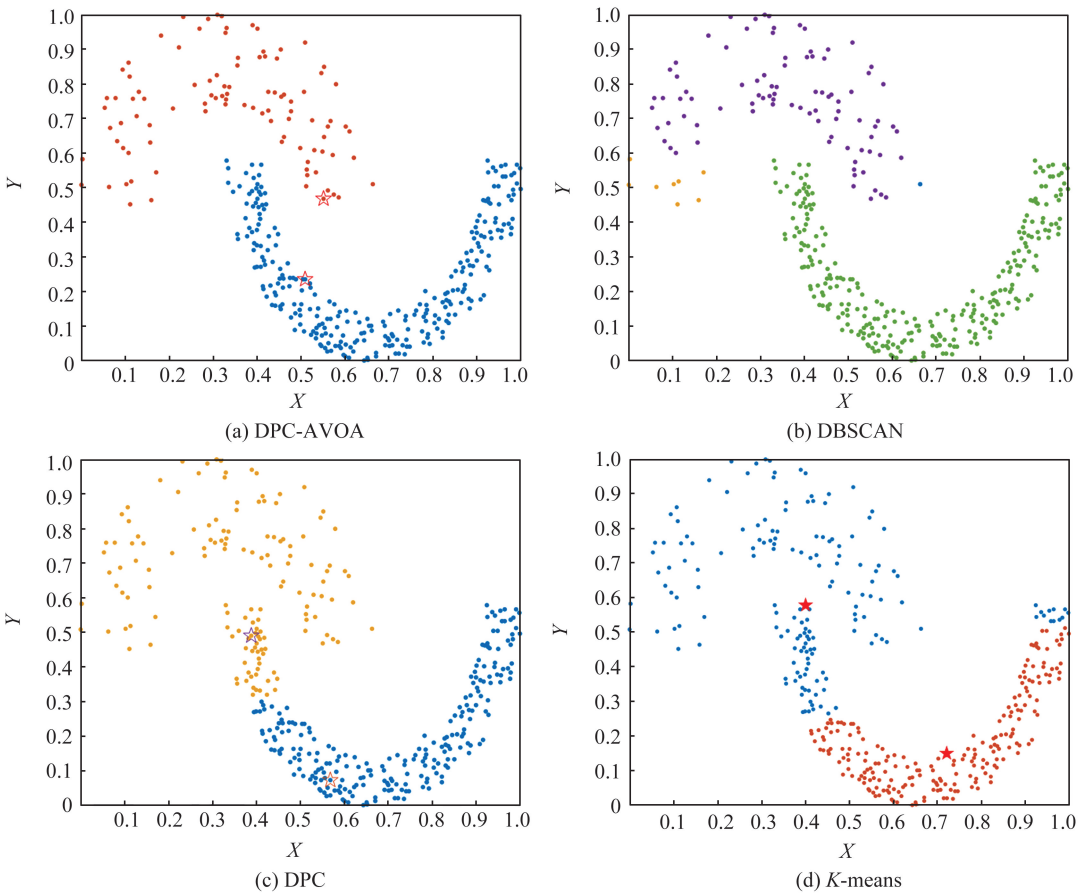


图 2 DPC-AVOA、DBSCAN、DPC、*K*-means 在 Jain 数据集上的聚类图

Fig.2 Cluster plot of DPC-AVOA, DBSCAN, DPC, and *K*-means on Jain database

从图 3 在 three-circle 数据集上的效果图可以看出,由于传统的 *K*-means 算法其分配非簇中心点方式和计算距离的缺陷,导致 3 个圆环簇被错误划分,*K*-means 算法错误地将数据集从左到右按照距离划分为 3 份,由于随机性,导致簇中心点也寻找错误;而在本文算法、DPC 和 DBSCAN 算法上,得到的效果明显最佳,也验证了本文算法在处理环形密度不均匀数据集时的有效性。

由图 4 的聚类图可得,*K*-means、DPC 和 DBSCAN 都不能正确地聚类。DBSCAN 将左上角的部分簇和下半环形簇聚类在一起,DPC 和 *K*-means 将下半部分半环形簇平均分为 3 类。本文算法能够正确地找到簇中心并得到精确的聚类结果。

由表 2 中合成数据集的性能指标值可得:本文算法、DPCSA 和 DPC 算法整体上优于 *K*-means 和 DBSCAN算法,本文算法除了在 Aggregation 和 R15 数据集的聚类指标值略低于 DPC,在其他数据集的聚类指标值均高于所对比算法。另外,在 D31、Jain、Compound 和 A3 数据集上,本文算法的各项性能值明显优于传统算法和 DPC 算法,在 Compound 和 A3 数据集上跟 DPCSA 算法旗鼓相当,反映了本文算法在选取截断距离 d_c 和所采用样本分配策略时的优越性。在 Three-circle 和 Sprial 数据集上,本文算法、DPC 和 DBSCAN 算法的性能保持一致,*K*-means 算法的聚类指标值最低。在 Flame 数据集上,本文算法、DPC 和 DPCSA 算法都能达到精确度 1,DBSCAN 算法略低于这 3 种算法,*K*-means 算法的表现最为不佳。

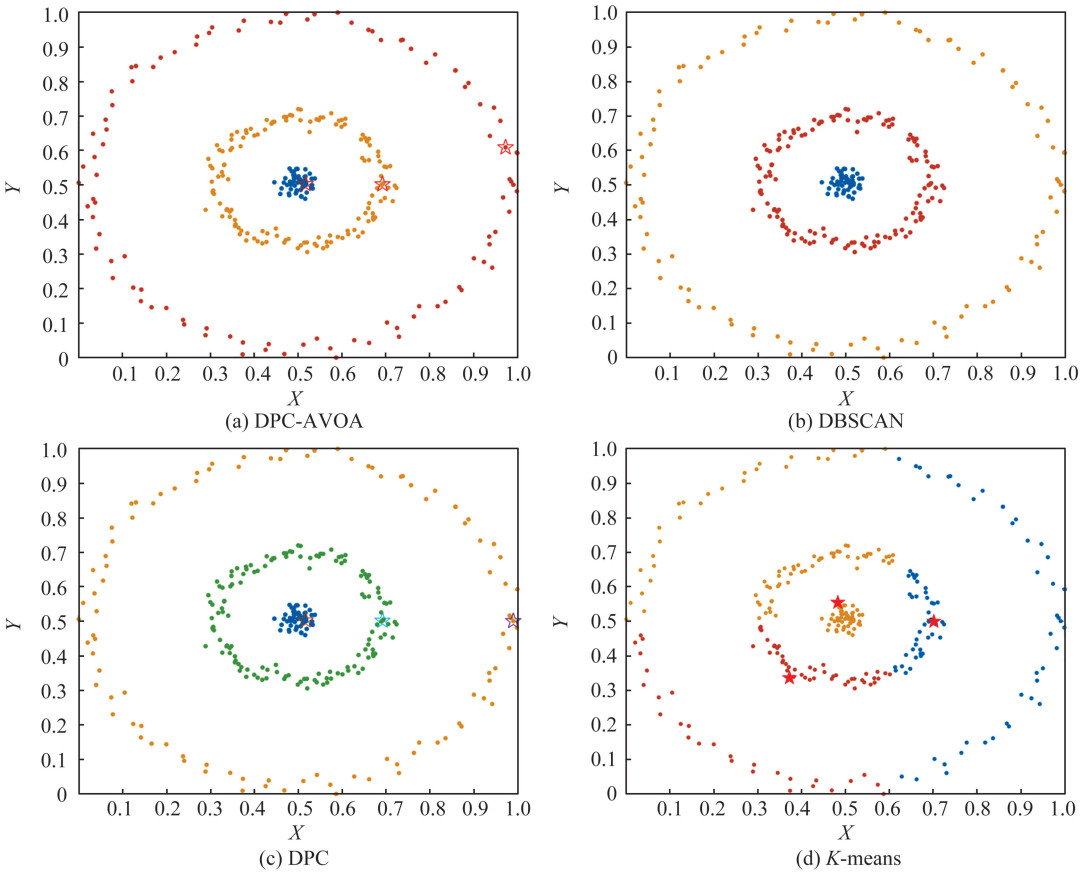


图 3　DPC-AVOA、DBSCAN、DPC、*K*-means 在 three-circle 数据集上的聚类图

Fig.3　Cluster plot of DPC-AVOA, DBSCAN, DPC, and *K*-means on three-circle database

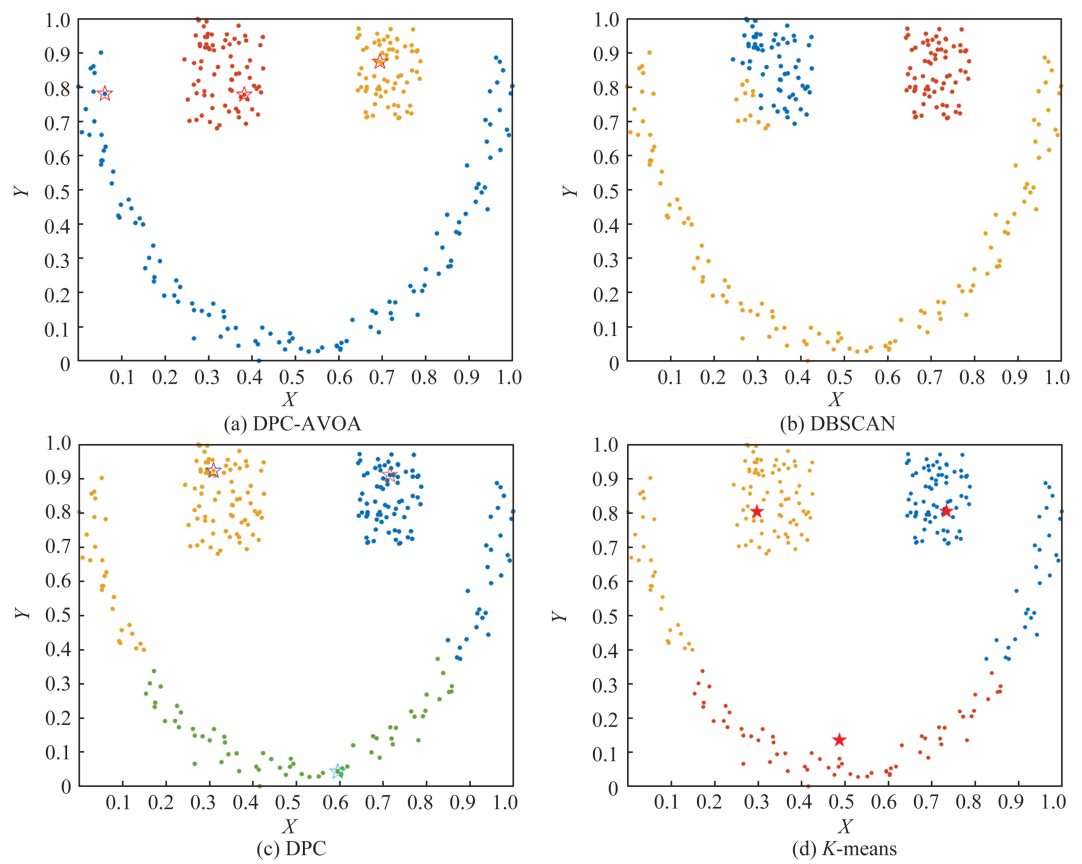


图 4　DPC-AVOA、DBSCAN、DPC、*K*-means 在 A3 数据集上的聚类图

Fig.4　Cluster plot of DPC-AVOA, DBSCAN, DPC, and *K*-means on A3 database

表 2 在合成数据集上的 Ami、Ari、Fmi 指标

Table 2 Ami, Ari and Fmi indices on the artificial dataset

算法	Flame			Aggregation			Spiral		
	Ami	Ari	Fmi	Ami	Ari	Fmi	Ami	Ari	Fmi
K-means	0.405 9	0.453 5	0.736 1	0.774 8	0.673 5	0.742 9	−0.005 7	−0.006 3	0.328 1
DPC	**1.000 0**	**1.000 0**	**1.000 0**	**0.992 1**	**0.995 6**	**0.996 5**	**1.000 0**	**1.000 0**	**1.000 0**
DBSCAN	0.897 5	0.938 8	0.971 2	0.911 4	0.910 2	0.932 7	**1.000 0**	**1.000 0**	**1.000 0**
DPCSA	**1.000 0**	**1.000 0**	**1.000 0**	0.953 7	0.958 1	0.967 3	**1.000 0**	**1.000 0**	**1.000 0**
DPC-AVOA	**1.000 0**	**1.000 0**	**1.000 0**	0.991 1	0.9942	0.9954	**1.000 0**	**1.000 0**	**1.000 0**
算法	D31			Jain			R15		
	Ami	Ari	Fmi	Ami	Ari	Fmi	Ami	Ari	Fmi
K-means	0.941 6	0.898 9	0.902 3	0.404 4	0.413 4	0.742 5	0.981 0	0.973 2	0.975 0
DPC	0.954 6	0.934 5	0.936 6	0.503 7	0.569 1	0.815 9	**0.993 7**	**0.992 7**	**0.993 2**
DBSCAN	0.889 5	0.807 8	0.818 6	0.885 1	0.975 8	0.990 6	0.982 2	0.981 9	0.983 1
DPCSA	0.955 2	0.935 3	0.937 4	0.216 7	0.044 2	0.592 4	0.988 5	0.985 7	0.986 6
DPC-AVOA	**0.957 9**	**0.9407**	**0.942 6**	**1.000 0**	**1.000 0**	**1.000 0**	0.990 7	0.989 1	0.989 8
算法	Three-circle			Compound			A3		
	Ami	Ari	Fmi	Ami	Ari	Fmi	Ami	Ari	Fmi
K-means	0.223 2	0.131 9	0.437 7	0.657 3	0.528 9	0.634 9	0.579 2	0.481 0	0.661 1
DPC	**1.000 0**	**1.000 0**	**1.000 0**	0.775 4	0.591 0	0.687 6	0.582 8	0.487 5	0.665 0
DBSCAN	**1.000 0**	**1.000 0**	**1.000 0**	0.7340	0.802 9	0.862 7	0.858 7	0.866 1	0.914 4
DPCSA	0.604 3	0.515 1	0.729 6	**0.839 2**	0.828 4	0.870 7	**1.000 0**	**1.000 0**	**1.000 0**
DPC-AVOA	**1.000 0**	**1.000 0**	**1.000 0**	0.803 1	**0.830 5**	**0.879 6**	**1.000 0**	**1.000 0**	**1.000 0**

由表 3 中高维数据集的指标结果可以看出,本文算法除了在 Wine、WDBC 和 Ionosphere 数据集上的聚类性能略低于 *K*-means 和 DBSCAN 算法,在其他 6 个数据集(Seeds、Iris、Ecoli、pima、Waveform(noise)、Libras)中均明显优于对比算法。另外在加入噪声的 Waveform(noise)数据集上,除了评价指标 Ami 低于 *K*-means和 Fmi 低于 DBSCAN 算法的指标值,其余指标均高于所对比的算法,算法的指标更是比原始 DPC 和同类型的 DPCSA 优越许多,也充分展示了本文算法对有异常值和噪声数据集的抗干扰能力和稳定性。

表 3 在真实数据集上的 Ami、Ari、Fmi 指标

Table 3 Ami, Ari and Fmi indices for real-world datasets

算法	Ionosphere			Libras			Seeds		
	Ami	Ari	Fmi	Ami	Ari	Fmi	Ami	Ari	Fmi
K-means	0.068 2	0.100 7	0.568 8	0.509 4	0.291 5	0.342 4	0.667 3	0.695 4	0.796 4
DPC	0.089 7	0.143 3	0.595 3	0.38 7	0.207 9	0.301 4	0.714 4	0.744 7	0.829 6
DBSCAN	**0.5520**	**0.683 5**	**0.857 5**	0.128 3	0.028 4	0.242 7	0.401 6	0.402 5	0.635 4
DPCSA	0.135 5	0.218 3	0.643 2	0.493 9	0.268 3	0.357 2	0.660 9	0.687 3	0.791 8
DPC-AVOA	0.372 1	0.503 4	0.782 9	**0.584 2**	**0.381 3**	**0.439 5**	**0.807 3**	**0.850 6**	**0.900 0**
算法	Iris			Ecoli			Wine		
	Ami	Ari	Fmi	Ami	Ari	Fmi	Ami	Ari	Fmi
K-means	0.757 3	0.743 6	0.829 4	0.486 8	0.374 9	0.516 7	**0.856 4**	**0.880 3**	**0.920 5**
DPC	0.766 8	0.719 5	0.815 8	**0.517 8**	0.436 5	0.569 2	0.706 4	0.672 3	0.783 4
DBSCAN	0.676 8	0.6120	0.729 1	0.451 6	0.525 5	0.662 3	0.434 7	0.418 7	0.623 4
DPCSA	0.883 1	0.903 8	0.935 5	0.4170	0.478 3	0.674 2	0.74 8	0.741 4	0.828 3
DPC-AVOA	**0.897 1**	**0.903 9**	**0.935 6**	0.4730	**0.547 4**	**0.713 3**	0.843 1	0.866 6	0.911 3
算法	WDBC			Pima			Waveform (noise)		
	Ami	Ari	Fmi	Ami	Ari	Fmi	Ami	Ari	Fmi
K-means	**0.637 5**	**0.754 8**	**0.887 6**	0.046 3	0.087 1	0.576 6	**0.361 1**	0.251 5	0.502 3
DPC	0.008 6	−0.010 7	0.715 5	0.034 1	0.0780	0.588 8	0.087 8	0.067 1	0.458 5
DBSCAN	0.333 4	0.453 9	0.747 4	0.011 3	0.041 5	0.693 6	0.000 0	0.000 0	**0.577 3**
DPCSA	0.336 1	0.377 1	0.759 5	0.001 7	0.014 3	**0.711 9**	0.152 4	0.134 9	0.462 3
DPC-AVOA	0.448 7	0.523 1	0.801 9	**0.047 9**	**0.093 9**	0.709 4	0.274 7	**0.255 5**	0.507 2

总体上来说,本文算法在面对低维人工数据集时,具有很高的聚类准确率,且避免了参数敏感的问题,在高维真实数据集上,性能相比于另外 4 种算法也有显著提高。

4　结束语

本文针对 DPC 算法存在的缺陷,提出了基于非洲秃鹫优化算法改进的密度峰值聚类。首先,通过非洲秃鹫算法较好的寻优性能来寻找截断距离 d_c 值,并利用评价指标 Acc 作为适应度函数,避免了人为选参的敏感性和不确定性;其次,在分配非簇中心点时利用 2 种不同的方法来聚类,能很好地将位于不同密度位置的数据点分类,对弯曲、环状结构的数据集有更准确的分配;最后,在人工和真实数据集上进行实例验证,并在几个合成数据集上进行可视化实现。通过与对比算法的可视化结果分析及聚类性能的比较得出,本文所改进的算法有较高的准确率,能够有效地克服数据集密度差异太大的影响,有效地处理不同密度大小的数据集。下一步的研究重点是如何提高算法在大数据集中的效率。

参考文献:

[1] STREHL A, GHOSH J. Relationship-based clustering and visualization for high-dimensional data mining[J]. INFORMS Journal on Computing, 2003, 15(2):208-230.

[2] DONG Gaogao, TIAN Lixin, DU Ruijin, et al. Analysis of percolation behaviors of clustered networks with partial support-dependence relations[J]. Physica A: Statistical Mechanics and its Applications, 2014, 394:370-378.

[3] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191):1492-1496.

[4] CHEN Yewang, LAI Dehe, QI Han, et al. A new method to estimate ages of facial image for large database[J]. Multimedia Tools and Applications, 2016, 75(5):2877-2895.

[5] XIE Juanying, GAO Hongchao, XIE Weixin, et al. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted *K*-nearest neighbors[J]. Information Sciences, 2016, 354:19-40.

[6] MEHMOOD R, ZHANG Guangzhi, BIE Rongfang, et al. Clustering by fast search and find of density peaks via heat diffusion [J]. Neurocomputing, 2016, 208:210-217.

[7] TANG Guihua, JIA Sen, LI Jun. An enhanced density peak-based clustering approach for hyperspectral band selection[C]// 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). New York: IEEE, 2015:1116-1119.

[8] WANG Shuliang, WANG Dakui, LI Caoyuan, et al. Clustering by fast search and find of density peaks with data field[J]. Chinese Journal of Electronics, 2016, 25(3):397-402.

[9] JIANG Dong, ZANG Wenke, SUN Rui, et al. Adaptive density peaks clustering based on *K*-nearest neighbor and Gini coefficient[J]. IEEE Access, 2020, 8:113900-113917.

[10] DU Mingjing, DING Shifei, XU Xiao, et al. Density peaks clustering using geodesic distances[J]. International Journal of Machine Learning and Cybernetics, 2018, 9(8):1335-1349.

[11] WANG Xiaofeng, XU Yifan. Fast clustering using adaptive density peak detection[J]. Statistical Methods in Medical Research, 2017, 26(6):2800-2811.

[12] DU Mingjing, DING Shifei, JIA Hongjie. Study on density peaks clustering based on *k*-nearest neighbors and principal component analysis[J]. Knowledge-Based Systems, 2016, 99:135-145.

[13] 丁志成,葛洪伟,周竞.基于 KL 散度的密度峰值聚类算法[J].重庆邮电大学学报(自然科学版),2019,31(3):367-374.
DING Zhicheng, GE Hongwei, ZHOU Jing. Density peaks clustering based on Kullback Leibler divergence[J]. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition), 2019, 31(3):367-374.

[14] 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[J].中国科学(信息科学),2016,46(2):258-280.
XIE Juanying, GAO Hongchao, XIE Weixin. *K*-nearest neighbors optimized clustering algorithm by fast search and finding the density peaks of a dataset[J]. Science China(Information Science), 2016, 46(2):258-280.

[15] YU Donghua, LIU Guojun, GUO Maozu, et al. Density peaks clustering based on weighted local density sequence and nearest neighbor assignment[J]. IEEE Access, 2019, 7:34301-34317.

[16] SEYEDI S A, LOTFI A, MORADI P, et al. Dynamic graph-based label propagation for density peaks clustering[J]. Expert Systems with Applications, 2019, 115:314-328.

[17] LIU Rui, WANG Hong, YU Xiaomei. Shared-nearest-neighbor-based clustering by fast search and find of density peaks[J]. Information Sciences, 2018, 450:200-226.

(下转第 71 页)