基于快速密度峰值聚类离群因子的离群点检测算法

张忠平[1,2,3]，李森[1]，刘伟雄[1]，刘书霞[4]

（1. 燕山大学信息科学与工程学院，河北 秦皇岛 066004；
2. 河北省计算机虚拟技术与系统集成重点实验室，河北 秦皇岛 066004；
3. 河北省软件工程重点实验室，河北 秦皇岛 066004；
4. 河北科技师范学院，河北 秦皇岛 066004）

摘 要：针对密度峰值聚类算法需要人工设置参数、时间复杂度高的问题，提出了基于快速密度峰值聚类离群因子的离群点检测算法。首先，使用 k 近邻算法代替密度峰值聚类中的密度估计，采用 KD-Tree 索引数据结构计算数据对象的 k 近邻；然后，采用密度和距离乘积的方式自动选取聚类中心。此外，定义了向心相对距离、快速密度峰值聚类离群因子来刻画数据对象的离群程度。在人工数据集和真实数据集上对所提算法进行实验验证，并与一些经典和新颖的算法进行对比实验，从正确性和时间效率上验证了所提算法的有效性。

关键词：数据挖掘；密度峰值聚类；离群点；k 近邻；向心相对距离

中图分类号：TP311

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2022193

Outlier detection algorithm based on fast density peak clustering outlier factor

ZHANG Zhongping[1,2,3], LI Sen[1], LIU Weixiong[1], LIU Shuxia[4]

1. College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
2. The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province, Qinhuangdao 066004, China
3. The Key Laboratory of Software Engineering of Hebei Province, Qinhuangdao 066004, China
4. Hebei Normal University of Science and Technology, Qinhuangdao 066004, China

Abstract: For the problem that peak density clustering algorithm requires human set parameters and high time complexity, an outlier detection algorithm based on fast density peak clustering outlier factor was proposed. Firstly, k nearest neighbors algorithm was used to replace the density peak of density estimate, which adopted the KD-Tree index data structure calculation of k close neighbors of data objects, and then the way of the product of density and distance was adopted to automatic selection of clustering centers. In addition, the centripetal relative distance and fast density peak clustering outliers were defined to describe the degree of outliers of data objects. Experiments on artificial data sets and real data sets were carried out to verify the algorithm, and compared with some classical and novel algorithms. The validity and time efficiency of the proposed algorithm are verified.

Keywords: data mining, density peak clustering, outlier, k nearest neighbor, centripetal relative distance

收稿日期：2022−04−12；**修回日期**：2022−07−08

通信作者：李森，1033772004@qq.com

基金项目：国家自然科学基金资助项目（No.61972334）；国家社会科学基金资助项目（No.20BJ122）；河北省创新能力提升计划基金资助项目（No.20557640D）；四达铁路智能图像工件识别基金资助项目（No.x2021134）

Foundation Items: The National Natural Science Foundation of China (No.61972334),The National Social Science Foundation of China (No.20BJ122), Hebei Province Innovation Capability Improvement Plan Project (No.20557640D), The Intelligent Image Workpiece Recognition of Sida Railway (No.x2021134))

0　引言

离群点指的是数据集中偏离正常数据分布的数据，它们的观测值与其他观测值有着显著的不同。离群点检测就是在数据集中发现离群点的过程，是数据挖掘中重要的研究方向。目前，离群点检测已应用在很多领域，如工业无线传感器网络[1]、欺诈检测[2-3]、入侵检测[4]、心电图异常检测[5]等。研究者对离群点检测也提出了很多算法，主要有基于统计的算法[6-7]、基于距离的算法[8-9]、基于聚类的算法[10-12]和基于密度的算法[13-18]等。

自 Breunig 等[13]提出局部离群因子（LOF, local outlier factor）算法后，基于密度的离群点检测算法已成为离群点检测研究领域最热门的方向之一。该类算法的核心思想是通过密度估计算法来计算各数据对象的密度，根据数据对象所处的区域来判断离群点，离群点通常处于稀疏区域，离群点的密度相较于正常数据对象而言有明显差异。近年来，随着聚类方法的发展，基于聚类的离群点检测算法已成为离群点检测研究领域不可或缺的一部分。其核心思想是对数据集进行聚类，离群点通常处于那些仅包含极少数据对象的小规模集群中。从核心思想上可以看出，上述 2 类算法具有一定的相似性，小规模的集群往往也是稀疏区域。这 2 类算法有各自的优缺点，近几年研究者开始把研究重心放在如何结合两者的优点上，从而提出一种具有更强稳健性的离群点检测算法。

Rodriguez 等[19]提出了一种具有重要价值的新型聚类算法——密度峰值聚类（DPC, density peak clustering）算法。DPC 算法是一种简单高效的聚类算法，可以通过计算局部密度以及相对距离，将任意维度的数据集映射成一个二维的决策图，决策图可以清晰地反映数据集的层次关系；然后选取数据集中密度高且距离其他高密度数据对象较远的数据对象作为数据集的聚类中心，这些数据对象称为密度峰值点；最后将剩余的数据对象分配到距离其最近的聚类中心。相较于其他传统聚类算法，DPC 算法受离群点和噪声的影响较小，另外，DPC 算法在聚类的过程中不需要多次迭代，因此，不管是从检测精度还是从时间效率层面考虑，DPC 算法更适合与离群点检测算法相结合。

尽管 DPC 算法可以简单高效地完成聚类，但仍存在一些问题。首先，由于 DPC 算法需要计算各数据对象密度，计算过程中会产生距离矩阵，在增加空间复杂度的同时也增大了时间复杂度，DPC 算法的时间复杂度为 $O(n^2)$，对于一些大规模数据集，其时间效率会严重下降。文献[20]提出了基于网格的密度峰值聚类（DPCG, density peak clustering based on grid)算法，在计算数据对象的局部密度时采用网格思想计算距离的方法来代替 DPC 算法中计算数据对象之间距离的方法，从而提高了算法计算速度。其次，DPC 算法在初始时需要设置截断距离 d_c 且需要人工选择聚类中心，由于算法对于截断距离 d_c 非常敏感，人工设置容易导致算法精度下降。Huang 等[21]提出自适应密度峰值检测的快速聚类算法，该算法采用非参数的高斯核密度估计代替 DPC 算法中的密度估计，不需要设定截断距离 d_c，还提出了一种基于轮廓优化算法的聚类中心选择方法来自动选择聚类中心，但该方法在面对一些稀疏且小规模的簇类数据集时，难以保持正常的精度。

本文针对 DPC 算法时间复杂度高、截断距离的设定和聚类中心选取的问题进行了改进，并将改进的 DPC 算法与离群点检测相结合，提出了基于快速密度峰值聚类离群因子（FDPC-OF, fast density peak clustering outlier factor）的离群点检测算法，简称 FDPC-OF 算法。本文算法主要分为 2 个阶段进行工作。第一阶段，使用改进的 DPC 算法计算聚类中心；第二阶段，基于聚类中心定义了向心相对距离，从而提出基于快速密度峰值聚类离群因子（简称 FOF）来刻画数据对象的离群程度。本文的主要贡献总结如下。

1) 本文算法采用 k 近邻（KNN, k-nearest neighbour）算法代替 DPC 算法中密度估计的部分，避免了设定截断距离 d_c，并使用 k 维树（KD-Tree, k-dimensional tree）索引数据结构计算各数据对象的 k 近邻，降低算法的时间复杂度。

2) 使用局部密度与相对距离的乘积来自动选择聚类中心，避免人为选取聚类中心。

3) 对 DPC 算法中的相对距离进行改进，定义了平均向心距离，并结合数据对象的相对距离以及平均向心距离定义了向心相对距离，结合向心相对距离和局部密度这 2 个指标，提出了基于快速密度峰值聚类离群因子来刻画数据对象的离群程度。

1 相关工作

1.1 密度峰值聚类 DPC 算法

经典的聚类算法 K-means[22]通过指定初步聚类中心再进行迭代更新找到最终聚类中心，由于每个点都被分配到距离最近的聚类中心，导致其不能检测非球面类别的数据分布。Ester 等[23]提出了 DBSCAN（density-based spatial clustering of application with noise）算法，可以对任意形状分布的数据进行聚类，但是必须指定一个密度阈值，从而去除低于此密度阈值的噪声点。Comaniciu 等[24]提出了 MeanShift 算法，采用均值偏移的方式使数据点向密度较大的方向延伸，直至收敛到簇中心。但是均值偏移是一个不断迭代的过程，时间复杂度较高。DPC 算法解决了以上不足，其原理简单，要求参数少，不需要迭代，并且能够识别不同形状大小的簇。

DPC 算法的核心思想是基于一种假设，即在任意数据分布的数据集中，聚类中心通常会被具有较低局部密度的数据对象所包围，并且这些聚类中心与其他具有较高局部密度的数据对象之间的距离都相对较大。在这个假设的基础上，DPC 算法需要计算 2 个指标，一个是数据对象的局部密度，另一个是数据对象的相对距离，这 2 个指标都需要通过数据对象之间的距离 d_{ij} 计算得出；然后，根据局部密度和相对距离来构建聚类中心决策图，以此选择聚类中心；最后，将数据集中剩余的数据对象分配到距离其最近的聚类中心，完成聚类。

定义 1 密度峰值聚类局部密度 ρ_i[19]。ρ_i 指数据对象 x_i 在截断距离内包含的其他数据对象的个数。其计算式如下

$$\rho_i = \sum_{j=1}^{n} \chi\left(d_{ij} - d_c\right) \tag{1}$$

其中，n 为数据集中数据对象的个数；d_{ij} 为数据对象 x_i 和 x_j 之间的欧氏距离；截断距离 d_c 需要在算法开始时进行初始化设置；$\chi(\cdot)$ 为指示函数，若 x<0，$\chi(x)=1$，若 $x \geqslant 0$，则 $\chi(x)=0$。简单来说，截断距离 d_c 类似于半径，ρ_i 则是通过指示函数计算在半径范围内其他数据对象的个数。

定义 2 密度峰值聚类相对距离 δ_i[19]。δ_i 指数据对象 x_i 与其他局部密度比它大的数据对象的最短距离。其计算式如下

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

对于 ρ_i 最大的数据对象，通常将它的 δ_i 设置为其与最远的对象之间的距离，即 $\delta_i = \max_j (d_{ij})$。值得注意的是，当 x_i 为 DPC 局部或者全局密度最大值时，DPC 相对距离会大于传统的 k 近邻距离，因此聚类中心的 DPC 相对距离会是非常大的值。

计算局部密度和相对距离后，DPC 算法根据这 2 个指标构建聚类中心决策图，以此选择聚类中心，该决策图的观测过程是 DPC 算法的核心。由 28 个数据对象生成的 DPC 二维数据分布如图 1 所示。从图 1 可以发现，1 号和 10 号数据对象分别位于各自簇内的密度峰值区域，因此将其作为聚类中心。

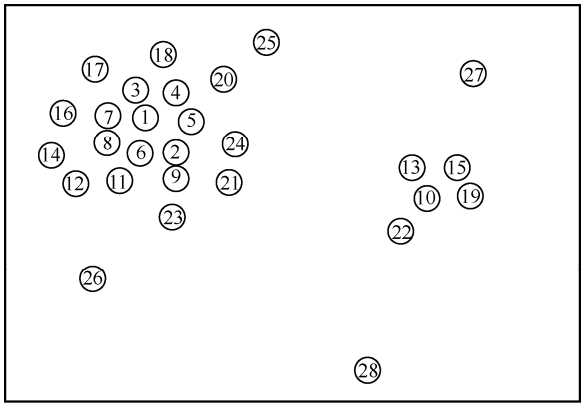

图 1 DPC 二维数据分布

由 DPC 局部密度和 DPC 相对距离所构建的 DPC 决策图如图 2 所示，从图 2 可以发现，首先，虽然 9 号和 10 号数据对象的 DPC 局部密度非常接近，但它们在决策图中所处的位置有非常大的区别，这是由于 10 号数据对象在其所在簇中的 DPC 局部密度为最大值，DPC 局部密度比其大的数据对象位于另一个簇中，而 9 号数据对象并非其所在簇中 DPC 局部密度的最大值，因此 10 号数据对象的 DPC 相对距离比 9 号要大得多，更加符合 DPC 算法对聚类中心的假设；其次，由于 1 号数据对象为 DPC 局部密度的最大值，因此通过该决策图可以轻松选择出 1 号和 10 号这 2 个聚类中心；最后，26～28 号数据对象具有较高的相对距离和较小的 DPC 局部密度，此类型的数据对象通常作为离群点单独成簇。

通过决策图完成聚类中心的选定后，DPC 算法将数据集中剩余的数据对象分配到距离其最近的聚类中心所在的簇中完成聚类。DPC 算法如算法 1

所示。需要特别注意的是，聚类数目 c 和聚类中心需要通过决策图来人工选定。

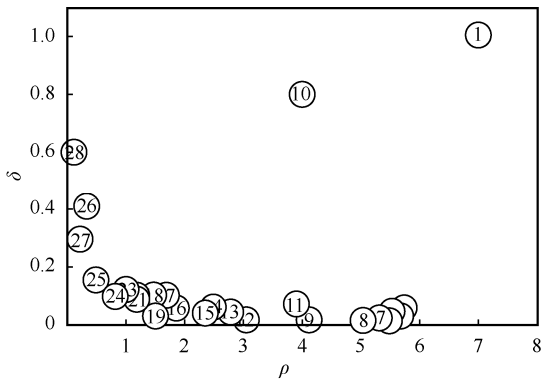

图 2　DPC 决策图

算法 1　密度峰值聚类 DPC 算法

输入　初始数据集 D 和截断距离 d_c

输出　c 个聚类

1) for each　$x \in D$　do

2) 计算 x 与其他数据对象之间的欧氏距离

3) end for

4) for each　$x \in D$　do

5) 采用式(1)和截断距离 d_c 计算 x 的 DPC 局部密度

6) end for

7) for each　$x \in D$　do

8) 遍历数据集 D

9) 采用式(2)计算 x 的 DPC 相对距离

10) end for

11) 根据计算得出的 DPC 局部密度和 DPC 相对距离构建二维决策图

12) 根据决策图人工选取聚类中心以及聚类数目 c

13) 将数据集中剩余的数据对象分配至距离其最近的聚类中心

14) 完成聚类并输出 c 个聚类

1.2　k 近邻

定义 3　k 近邻距离（KNN-dist, k-nearest neighbours distance）d_i。d_i 是指 x_i 到其 KNN 内所有数据对象的平均距离。其计算式如下

$$d_i = \frac{1}{k}\sum_{x_j \in \mathrm{KNN}_i} \mathrm{dist}(x_i, x_j) \tag{3}$$

其中，KNN_i 为数据对象 x_i 的 k 近邻的集合；$\mathrm{dist}(x_i, x_j)$ 为数据对象 x_i 和 x_j 之间的距离，通常采用欧氏距离。

定义 4　KNN 局部密度 den_i[25]。den_i 是根据 x_i 的 KNN 距离构造的局部密度估计度量指标。其计算式如下

$$\mathrm{den}_i = \frac{1}{d_i} = \frac{k}{\sum_{x_j \in \mathrm{KNN}_i} \mathrm{dist}(x_i, x_j)} \tag{4}$$

简单而言，数据对象 x_i 的 KNN 局部密度 den_i 等于其 KNN 距离 d_i 的倒数。

计算数据对象的 KNN 距离时，需要先计算各数据对象之间的欧氏距离，时间复杂度高达 $O(n^2)$。当面对大规模数据集时，算法的时间效率会随着数据集的增大而快速下降。为了提高算法在大规模数据集上的适用性，本文采用 KD-Tree 索引数据结构来优化 KNN 距离的计算过程，首先对数据集进行建模，构建 KD-Tree；然后，根据该 KD-Tree 获取近邻样本数据。通过采用 KD-Tree 索引结构来计算数据对象的 KNN 距离，能将算法的时间复杂度优化至 $O(n\log n)$。

2　FDPC-OF 算法

2.1　快速密度峰值聚类算法

针对 DPC 算法中需要初始化设置截断距离参数以及需要人工选定聚类中心的问题，本文提出采用 KNN 局部密度代替 DPC 算法中局部密度估计，从而避免了人工设置截断距离参数，解决了 DPC 算法对于截断距离参数敏感的问题；并且使用局部密度与相对距离的乘积这一指标来选定聚类中心，避免了人为选取聚类中心可能出现的问题。另外，为了解决 DPC 算法时间复杂度高的问题，本文在计算 KNN 局部密度的过程中，采用 KD-Tree 索引数据结构来帮助计算各数据对象的 k 近邻，降低算法的时间复杂度，并且在计算相对距离的时候，只考虑数据对象的 k 近邻，避免在计算相对距离时进行全局搜索。

定义 5　KNN 相对距离 ω_i。ω_i 是指数据对象 x_i 与 k 近邻内其他 KNN 局部密度比它大的数据对象中的最短距离。其计算式如下

$$\omega_i = \left\{ \min(\mathrm{dist}(x_i, x_j)) \middle| x_j \in \mathrm{KNN}_i, \mathrm{den}_j > \mathrm{den}_i \right\} \tag{5}$$

当数据对象 x_i 是 KNN_i 内的局部密度最大值时，将其放入邻居密度最大值集合 maxden 中，然后将 x_i 与 maxden 中其他数据对象之间距离的最小值作为 x_i 的 KNN 相对距离 ω_i，即

$$\omega_i = \left\{ \min(\mathrm{dist}(x_i, x_j)) \middle| x_j \in \mathrm{maxden} \right\} \tag{6}$$

定义 6 聚类中心决策指标 cen_i。cen_i 指数据对象 x_i 的 KNN 局部密度和 KNN 相对距离的乘积。其计算式如下

$$\text{cen}_i = \text{den}_i \omega_i \tag{7}$$

得到 KNN 局部密度 den_i、KNN 相对距离 ω_i 以及聚类中心决策指标 cen_i 后，根据用户初始化输入的聚类参数 c 选择 cen_i 最大的前 c 个数据对象作为聚类中心。最后将数据集中剩余的数据对象依次分配到距离其距离最近的聚类中心完成聚类。快速密度峰值聚类（FDPC, fast density peak clustering）算法如算法 2 所示。

算法 2 FDPC 算法

输入 初始数据集 D、参数 k 以及聚类数目 c

输出 c 个聚类

1) 根据数据集 D 构建 KD-Tree

2) for each $x \in D$ do

3) 根据 KD-Tree 和式(3)计算数据对象 x 的 KNN 距离

4) end for

5) for each $x \in D$ do

6) 采用式(4)计算 x 的 KNN 局部密度

7) end for

8) for each $x \in D$ do

9) 采用式(5)和式(6)计算 x 的 KNN 相对距离

10) end for

11) for each $x \in D$ do

12) 采用式(7)计算 x 的聚类中心决策指标

13) end for

14) 将聚类中心决策指标 cen 前 c 大的数据对象作为聚类中心

15) 将数据集中剩余的数据对象分配至距离其最近的聚类中心

16) 完成聚类并输出 c 个聚类

2.2 FDPC-OF 算法描述

本文结合快速密度峰值聚类的度量指标提出 FDPC-OF 算法。在完成聚类算法后，为了更好地刻画离群点在一些具有不同密度的数据集上的离群程度，将进一步对快速密度峰值聚类中的 KNN 相对距离进行改进。

定义 7 平均向心距离 davg_i。davg_i 指聚类 C_i 中所有数据对象与聚类中心的平均距离。其计算式如下

$$\text{davg}_i = \frac{1}{m} \sum_{x_j \in C_i} \text{dist}(x_c, x_j) \tag{8}$$

其中，m 表示聚类 C_i 的数据对象个数；$\text{dist}(x_c, x_j)$ 表示数据对象 x_j 和其所属聚类的聚类中心 x_c 之间的距离。

定义 8 向心相对距离 discen_i。discen_i 指数据对象 x_i 的 KNN 相对距离 ω_i 和其所属聚类的平均向心距离 davg_i 的比值。其计算式如下

$$\text{discen}_i = \frac{\omega_i}{\text{davg}_i} \tag{9}$$

向心相对距离很好地刻画了离群点在一些不同密度的数据集上的离群程度。FDPC 二维数据分布如图 3 所示。从图 3 可以看出，24 号数据对象为全局离群点，其相对距离为最大值；25 号数据对象为局部离群点，其相对距离比稀疏聚类中的正常数据对象小，若把 KNN 相对距离作为离群指标，局部离群点容易被稀疏区域的正常数据对象淹没。因此，本文在 KNN 相对距离的基础上提出了平均向心距离。由于 25 号数据对象属于左侧聚类，KNN 相对距离与左侧聚类的平均向心距离的比值放大了该局部离群点的离群特征，更好地刻画了离群点的离群程度。

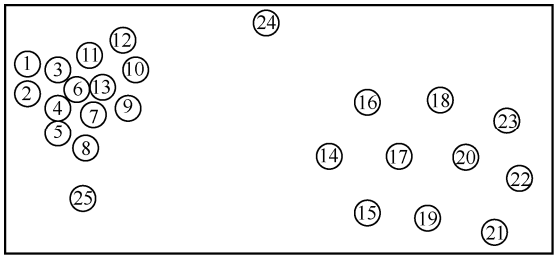

图 3 FDPC 二维数据分布

定义 9 快速密度峰值聚类离群因子 FOF_i。FOF_i 指数据对象 x_i 的向心相对距离 discen_i 和 KNN 局部密度 den_i 的比值。其计算式如下

$$\text{FOF}_i = \frac{\text{discen}_i}{\text{den}_i} \tag{10}$$

从密度上看，本文首先根据数据集构建 KD-Tree，然后根据该索引结构计算数据对象的 KNN 局部密度。该局部密度能很好地表示数据对象所处区域的密集程度。一般来说，离群点通常处于局部密度较低的区域。从距离上看，向心相对距离利用了聚类中心与数据对象之间的关系，能在一些具有不同密度的数据集上有较好的适应性，通常离群点的向心相对距离会比正常点大得

多。因此，快速密度峰值聚类离群因子能很好地刻画数据对象的离群程度，通常情况下离群因子较大的点更有可能为离群点，因此算法选取排序后的离群因子中前 o 个点作为离群点输出，在算法实际应用中，o 的取值是根据实际数据集的情况人为设定的。在本文实验中，为了验证所提 FDPC-OF 算法在不同数据集上的有效性，o 的取值根据数据集已有的离群点标签个数设定。FDPC-OF 算法如算法 3 所示。

算法 3　FDPC-OF 算法

输入　初始数据集 D、参数 k 以及聚类数目 c

输出　o 个离群点

1) 调用算法 2 对数据集 D 进行聚类，输出 c 个聚类

2) for each $c \in C$ do

3) 采用式(8)计算 c 的平均向心距离

4) end for

5) for each $x \in D$ do

6) 采用式(9)计算 x 的向心相对距离

7) end for

8) for each $x \in D$ do

9) 采用式(10)计算 x 的快速密度峰值聚类的离群因子

10) end for

11) 降序排列快速密度峰值聚类的离群因子

12) 输出前 o 个点作为离群点

2.3　FDPC-OF 算法正确性

FDPC-OF 算法中，KD-Tree 索引数据结构能有效降低算法时间复杂度，从而提高算法的计算速率，基于 KNN 对每个数据对象进行密度估计能很好地表示数据对象所处区域的疏密程度。根据算法 2 进行聚类，对每个聚类计算平均向心距离，将 KNN 相对距离与平均向心距离的比值作为向心相对距离能更好地在一些具有不同密度分布的数据集上放大离群点的离群特征。通过计算快速密度峰值聚类离群因子来刻画数据对象的离群程度，并对其进行降序排列，输出前 o 个离群点。

2.4　FDPC-OF 算法复杂性

FDPC-OF 算法的时间复杂度主要由以下 2 个部分组成。1) 为计算数据对象的 KNN 局部密度构建 KD-Tree，时间复杂度为 $O(n\log n)$，其中 n 为数据集的数据对象的个数。2) 计算数据对象的快速密度峰值聚类离群因子 FOF(x)，时间复杂度为 $O(n)$。因此，FDPC-OF 算法总的时间复杂度为 $O(n\log n)+O(n)\approx O(n\log n)$。

3　实验与分析

为了评估本文所提 FDPC-OF 算法的性能，本节将在真实数据集和人工数据集上进行实验，并选取了几种不同的离群点检测算法进行对比实验，包括 COF（connectivity-based outlier factor）算法[26]、NOF（natural outlier factor）算法[21]、RDOS（relative density-based outlier score）算法[27]、LDF（local density factor）算法[28]、IForest（isolation forest）算法[29]、NANOD（natural neighbour-based outlier detection）算法[30]和 MOD（mean-shift outlier detector）算法[31]。实验环境如表 1 所示。

表 1　实验环境

软硬件环境	参数
CPU	2.60 Hz Inter i7-4720HQ
硬盘/GB	512.0
内存/GB	16.0
开发环境	PyCharm
编译环境	Python 3.8
可视化工具	PyCharm

3.1　实验评价指标

本文针对算法有效性的实验将选取 3 种性能评估指标，分别为：精确率 Pr，计算方法如式(11)所示；加权评价指标 F1 值，计算方法如式(13)所示；运行时间。其中，精确率也被称为查准率，F1 值结合了精确率和召回率 Re 的表现。以上 Pr 和 F1 值的取值为 0～1，其值越大表示离群点检测算法的检测效果越好。

$$\mathrm{Pr}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{11}$$

$$\mathrm{Re}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{12}$$

$$\mathrm{F1}=\frac{2\,\mathrm{Pr}\times\mathrm{Re}}{\mathrm{Pr}+\mathrm{Re}} \tag{13}$$

其中，TP 为真阳性，表示算法正确检测为离群点的个数；FP 为假阳性，表示算法错误检测为离群点的个数；FN 为假阴性，表示算法将离群点检测为正常点的个数。

FDPC-OF 算法的相关参数设置如下：k 近邻中设置 k=5，自动选取聚类中心时，设置聚类数量参数 c=2。

对比算法的相关参数设置如下：RDOS 算法和 COF 算法中，k=5；LDF 算法中，$\theta=0.5$，k=5；NOF 算法和 NANOD 算法为无参算法；IForest 算法和 MOD 算法均采用原文献中的默认设置。

3.2 人工数据集

为了验证本文算法在各种不同的数据分布下离群点的检测能力，本文使用图 4 所示的二维人工数据集 D_1～D_4 进行对比实验，其中，〇表示离群点。D_1～D_4 的数据特征如表 2 所示。

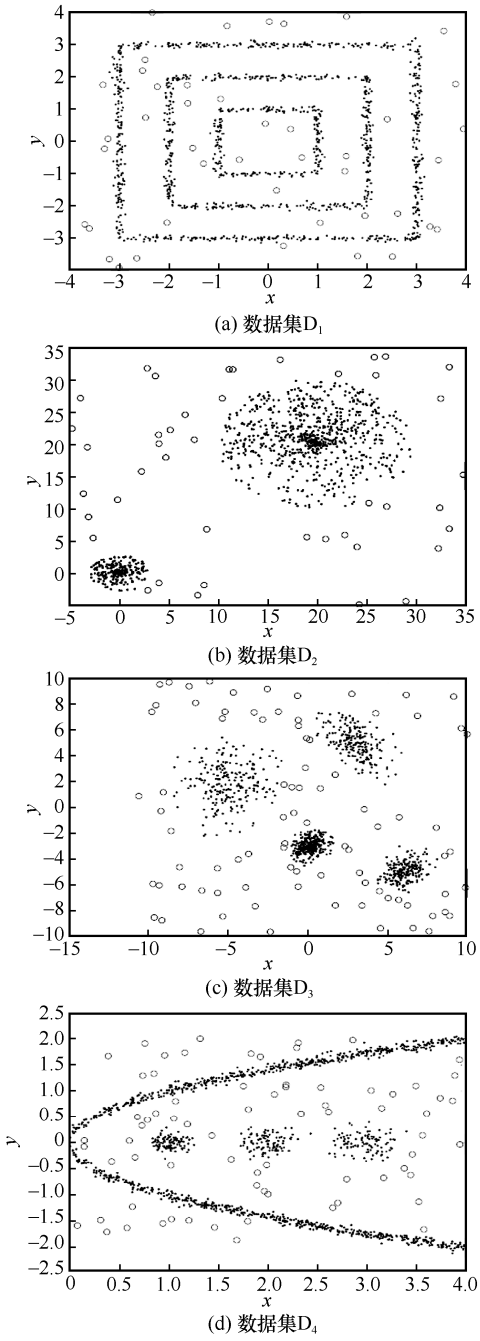

图 4 二维人工数据集

表 2 D_1～D_4 的数据特征

数据集	样本个数/个	离群点个数/个	离群点比例
D_1	1 256	43	3.4%
D_2	1 043	43	4.1%
D_3	1 000	85	8.5%
D_4	1 372	72	5.2%

FDPC-OF 算法与对比算法在人工数据集上的精确率如表 3 所示。从表 3 可以看出，FDPC-OF 算法在 4 种人工数据集上的 Pr 均值在 95%以上，显著优于对比算法，在数据集 D_1 上的 Pr 高达 100%；在数据集 D_2 上的 Pr 与 LDF 算法相等，均在 97%以上。FDPC-OF 算法在 4 种人工数据集上的离群点检测稳定性优于对比算法，这是由于 FDPC-OF 算法结合聚类中心的概念提出了向心相对距离，提高了算法在不同密度的数据集上的适用性。

表 3 不同算法在人工数据集上的精确率

算法	D_1	D_2	D_3	D_4
FDPC-OF	**100.00%**	**97.67%**	**87.05%**	**95.83%**
COF	86.04%	95.34%	76.47%	93.05%
NOF	95.34%	41.86%	57.64%	75.00%
LDF	93.02%	**97.67%**	84.70%	81.94%
RDOS	97.67%	32.55%	58.82%	84.72%
IForest	53.48%	93.02%	76.47%	26.38%
NANOD	48.83%	32.55%	71.76%	18.05%
MOD	83.72%	93.02%	76.47%	88.89%

FDPC-OF 算法与对比算法在人工数据集上的 F1 值如表 4 所示。从表 4 可以看出，FDPC-OF 算法在 4 个人工数据集上的 F1 值的均值在 93.5%以上，高于对比算法。LDF 算法在数据集 D_2 上的 F1 值与 FDPC-OF 算法相等，均在 95%以上；RDOS 算法的在数据集 D_1 和 D_4 上的 F1 值表现较好，在 D_1 上 F1 值高达 96.6%，但在数据集 D_2 和 D_3 上表现不佳；NANOD 算法在 4 个人工数据集上整体表现不理想；MOD 算法在 4 个人工数据集上表现稳定。从总体上看，FDPC-OF 算法的 F1 值优于对比算法，在不同人工数据集上有着稳定的离群点检测效率，因此可以证明本文算法在人工数据集上的有效性。

表 4　不同算法在人工数据集上的 F1 值

算法	D_1	D_2	D_3	D_4
FDPC-OF	**97.72%**	**95.45%**	**85.05%**	**93.87%**
COF	85.23%	93.18%	75.29%	91.84%
NOF	93.18%	40.90%	56.32%	74.15%
LDF	92.05%	**95.45%**	83.91%	80.27%
RDOS	96.60%	31.81%	58.05%	82.99%
IForest	52.27%	92.05%	74.71%	25.85%
NANOD	47.72%	31.81%	70.11%	18.35%
MOD	81.82%	90.91%	76.44%	88.44%

FDPC-OF 算法与对比算法在人工数据集上的运行时间如图 5 所示。从图 5 可以看出，FDPC-OF 算法在 4 种人工数据集上的运行时间的均值仅为 0.29 s，快于对比算法。IForest 算法由于不需要计算各数据对象之间的距离，因此时间复杂度为线性，在人工数据集上运行时间略长于 FDPC-OF 算法；COF 算法、NOF 算法、RDOS 算法以及 LDF 算法需要计算各数据对象之间的距离，因此时间复杂度至少为 $O(n^2)$，与 FPDC-OF 算法和 IForest 算法有较大差距；MOD 算法在寻找 KNN 邻域质心的过程中需要迭代计算，当数据量和维度增加时，该算法的时间复杂度会极大增加。FDPC-OF 算法在人工数据集上的运行时间在整体上优于对比算法，这是由于 FDPC-OF 算法在计算 k 近邻时使用了索引结构，从而降低了算法的时间复杂度，同时 FDPC-OF 算法计算过程简单。

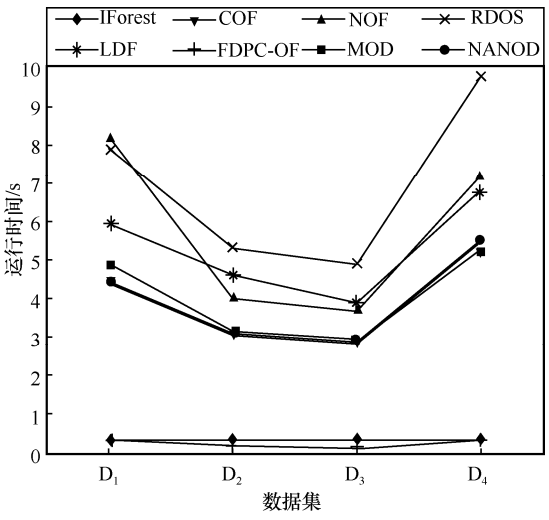

图 5　不同算法在人工数据集上的运行时间

3.3　真实数据集

本节采用 4 种来自 UCI 的真实数据集[32]进行实验，分别是 Ionosphere、Iris、Wdbc 和 Vowels。真实数据集选取的离群点比例为 3.4%~35.9%，属性个数为 4~34 个，保证了离群点比例以及数据集的属性个数的多样性和复杂性，能更好地检验本文算法以及对比算法在真实数据集上的离群点检测能力。真实数据集的数据特征如表 5 所示。

表 5　真实数据集的数据特征

数据集	样本个数/个	属性个数/个	离群点个数/个	离群点比例
Ionosphere	351	34	126	35.9%
Iris	110	4	10	9.1%
Wdbc	390	30	33	8.4%
Vowels	1 456	12	50	3.4%

FDPC-OF 算法与对比算法在真实数据集上的精确率如表 6 所示。从表 6 可以看出，FDPC-OF 算法在 4 种真实数据集上的 Pr 均值约为 80%，高于对比算法。在 Iris 数据集上，NANOD 算法表现最好，Pr 达到 80%；在 Wdbc 数据集上，FDPC-OF 算法的 Pr 与 NANOD 算法和 MOD 算法相等，为最高值。FDPC-OF 算法在不同规模、不同属性个数以及不同离群点比例的真实数据集上有着稳定的离群点检测效率。

表 6　不同算法在真实数据集上的精确率

算法	Ionosphere	Iris	Wdbc	Vowels
FDPC-OF	**91.26%**	70%	**78.78%**	**78%**
COF	77.77%	60%	36.36%	50%
RDOS	53.17%	60%	15.15%	4%
LDF	87.30%	70%	33.33%	48%
NOF	69.04%	50%	12.12%	26%
IForest	63.49%	50%	57.57%	14%
NANOD	73.81%	**80%**	**78.78%**	52%
MOD	84.92%	60%	**78.78%**	42%

FDPC-OF 算法与对比算法在真实数据集上的 F1 值如表 7 所示。从表 7 可以看出，FDPC-OF 算法在 4 种人工数据集上的 F1 值的均值在 78.6%以上，高于对比算法。LDF 算法在 Ionosphere 以及 Iris 数据集上的 F1 值表现优异，均值在 75%以上，但在其余数据集上表现一般；NANOD 算法和 MOD 算法的 F1 值接近，在 4 个真实数据集上都有着不错的表现，NANOD 算法在 Iris 数据集上的 F1 值表现最优，达到 80%，MOD 算法在 Wdbc 数据集上的表现与 FDPC-OF 算法相等，为最高值。从总体上看，FDPC-OF 算法在 F1 值的指标评估中有着不错的表现，结合精确度指标，本文提出的 FDPC-OF

算法在不同的真实数据集上有着稳定的离群点检测效率，因此可以证明本文 FDPC-OF 算法在真实数据集上能稳定高效地检测出离群点。

表 7　不同算法在真实数据集上的 F1 值

算法	Ionosphere	Iris	Wdbc	Vowels
FDPC-OF	**89.14%**	70%	**79.09**%	**76.47**%
COF	76.36%	60%	37.27%	49.01%
RDOS	53.17%	60%	14.92%	3.92%
LDF	85.27%	70%	32.83%	48.03%
NOF	68.61%	50%	11.94%	26.45%
IForest	62.01%	50%	56.71%	14.65%
NANOD	72.48%	**80**%	77.61%	52.90%
MOD	84.12%	60%	**79.09**%	42.15%

FDPC-OF 算法与对比算法在真实数据集上的运行时间如图 6 所示。从图 6 可以看出，FDPC-OF 算法在 4 种人工数据集上的运行时间的均值仅为 0.16 s，快于其他对比算法。在 Ionosphere 以及 Wdbc 数据集上，FDPC-OF 算法、COF 算法以及 IForest 算法有着优异的运行速率，运行时间均少于 1 s；在 Iris 数据集上，各数据集的运行速率均有着不错的运行速率，运行时间均少于 1 s；而在 Vowels 数据集上除了本文 FDPC-OF 算法以及 IForest 算法的运行时间少于 1 s 以外，其他算法的运行速率表现不佳，其中 NOF 算法、RDOS 算法以及 LDF 算法的运行时间超过 40 s。本文算法在处理大规模数据集时，仍然有着十分优异的运行速率，在真实数据集上的运行速率在整体上快于其他对比算法，因此本文算法的计算效率优异。

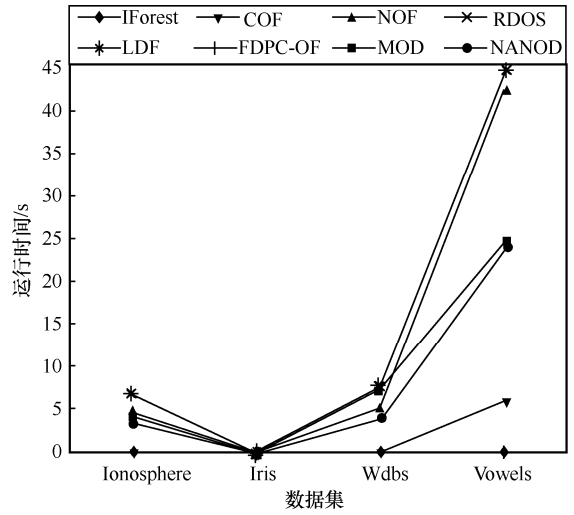

图 6　不同算法在真实数据集上的运行时间

4　结束语

本文首先分析了密度峰值聚类 DPC 算法和其算法思想，针对密度峰值聚类算法时间复杂度高以及需要人工设置参数的问题进行了深入的研究，给出了一种快速密度峰值聚类算法，使用 k 近邻算法代替密度峰值聚类中的密度估计，使用索引数据结构对距离计算进行优化，极大地减少了聚类的时间复杂度，并避免了设置截断距离 d_c。本文定义了向心相对距离，给出了基于快速密度峰值聚类离群因子来刻画数据对象的离群程度，从而提出了基于快速密度峰值聚类离群因子的离群点检测算法。对所提算法的正确性和复杂性进行分析，在人工数据集和真实数据集上的实验证明，FDPC-OF 算法能够高效全面地检测出离群点。

参考文献：

[1] RAMOTSOELA D, ABU-MAHFOUZ A, HANCKE G. A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study[J]. Sensors (Basel, Switzerland), 2018, 18(8): 2491.

[2] AVDIIENKO V, KUZNETSOV K, ROMMELFANGER I, et al. Detecting behavior anomalies in graphical user interfaces[C]//Proceedings of IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). Piscataway: IEEE Press, 2017: 201-203.

[3] NGAI E W T, HU Y, WONG Y H, et al. The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature[J]. Decision Support Systems, 2011, 50(3): 559-569.

[4] 季一木, 杨卫东, 李奎, 等. 基于主机系统调用频率的容器入侵检测方法[J]. 网络与信息安全学报, 2021, 7(4):18-29.
JI Y M, YANG W D, LI K, et al. Container intrusion detection method based on host system call frequency[J]. Chinese Journal of Network and Information Security, 2021, 7(4):18-29.

[5] ANDRYSIAK T. Sparse representation and overcomplete dictionary learning for anomaly detection in electrocardiograms[J]. Neural Computing and Applications, 2020, 32(5): 1269-1285.

[6] ROUSSEEUW P J, LEROY A M. Robust regression and outlier detection[M]. New Jersey: John Wiley & Sons, 1987.

[7] BARNETT V, LEWIS T, ABELES F. Outliers in statistical data[M]. New Jersey: John Wiley & Sons, 1994.

[8] KNORR E M, NG R T, TUCAKOV V. Distance-based outliers: algorithms and applications[J]. The VLDB Journal, 2000, 8(3/4): 237-253.

[9] KNORR E M, NG R T. A unified approach for mining outliers: properties and computation[C]//Proceedings of Conference of the Centre for Advanced Studies on Collaborative Research. [S.n.:s.l.], 1997: 219-222.

[10] JAIN A K, MURTY M N, FLYNN P J. Data clustering[J]. ACM Computing Surveys, 1999, 31(3): 264-323.

[11] ESTER M, KRIEGEL H, SANDER J, et al. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise[C]//International Conference on Knowledge Discovery & Data Mining. New York: ACM Press, 1996: 226-231.
[12] KARYPIS G, HAN E H, KUMAR V. Chameleon: hierarchical clustering using dynamic modeling[J]. Computer, 1999, 32(8): 68-75.
[13] BREUNIG M M, KRIEGEL H P, NG R T, et al. LOF: identifying density-based local outliers[C]//Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2000: 93-104.
[14] 杨晓晖, 刘晓明. 基于双向邻居修正的局部异常因子算法[J]. 通信学报, 2020, 41(8): 130-140.
YANG X H, LIU X M. Local outlier factor algorithm based on correction of bidirectional neighbor[J]. Journal on Communications, 2020, 41(8): 130-140.
[15] ZHANG K, HUTTER M, JIN H D. A new local distance-based outlier detection approach for scattered real-world data[C]//Advances in Knowledge Discovery and Data Mining. Berlin: Springer, 2009: 813-822.
[16] WANG L N, FENG C, REN Y J, et al. Local outlier detection based on information entropy weighting[J]. International Journal of Sensor-Networks, 2019, 30(4): 207.
[17] SCHUBERT E, ZIMEK A, KRIEGEL H P. Generalized outlier detection with flexible kernel density estimates[C]//Proceedings of the 2014 SIAM International Conference on Data Mining. [S.n.:s.l.], 2014:542-550.
[18] WAHID A, ANNAVARAPU C S R. NaNOD: a natural neigh-bour-based outlier detection algorithm[J]. Neural Computing and Applications, 2021, 33(6): 2107-2123.
[19] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191): 1492-1496.
[20] XU X, DING S F, DU M J, et al. DPCG: an efficient density peaks clustering algorithm based on grid[J]. International Journal of Machine Learning and Cybernetics, 2018, 9(5): 743-754.
[21] HUANG J L, ZHU Q S, YANG L J, et al. A non-parameter outlier detection algorithm based on natural neighbor[J]. Knowledge-Based Systems, 2016, 92: 71-77.
[22] MACQUEEN J. Some methods for classification and analysis of multivariate observations[C]//Proceedings of Berkeley Symposium on Mathematical Statistics & Probability. Berkeley: University of California Press, 1967: 281-297.
[23] ESTER M, KRIEGEL, H, SANDER J, et al. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise[C]//International Conference on Knowledge Discovery & Data Mining. New York:ACM Press, 1996: 226-231.
[24] COMANICIU D, MEER P. MeanShift: a robust approach toward feature space analysis[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002, 24(5): 603-619.
[25] DANG T T, NGAN H Y T, LIU W. Distance-based k-nearest neighbors outlier detection method in large-scale traffic data[C]//Proceedings of IEEE International Conference on Digital Signal Processing. Piscataway: IEEE Press, 2015: 507-510.
[26] TANG J, CHEN Z X, FU A W C, et al. Enhancing effectiveness of outlier detections for low density patterns[C]//Advances in Knowledge Discovery and Data Mining. Berlin: Springer, 2002: 535-548.
[27] TANG B, HE H B. A local density-based approach for outlier detection[J]. Neurocomputing, 2017, 241: 171-180.
[28] LATECKI L J, LAZAREVIC A, POKRAJAC D. Outlier detection with kernel density functions[C]//Machine Learning and Data Mining in Pattern Recognition. Berlin: Springer, 2007: 61-75.
[29] LIU F T, TING K M, ZHOU Z H. Isolation-based anomaly detection[J]. ACM Transactions on Knowledge Discovery from Data, 2012, 6(1): 1-39.
[30] WAHID A, ANNAVARAPU C S R. NaNOD: a natural neigh-bour-based outlier detection algorithm[J]. Neural Computing and Applications, 2021, 33(6): 2107-2123.
[31] YANG J W, RAHARDJA S, FRÄNTI P. Mean-shift outlier detection and filtering[J]. Pattern Recognition, 2021, 115: 107874.
[32] FRANK A, ASUNCION A. UCI machine learning repository[R]. 2010.

[作者简介]

张忠平（1972− ），男，吉林松原人，博士，燕山大学教授，主要研究方向为大数据、数据挖掘、半结构化数据等。

李森（1997− ），男，河南周口人，燕山大学硕士生，主要研究方向为数据挖掘。

刘伟雄（1997− ），男，广东广州人，燕山大学硕士生，主要研究方向为数据挖掘。

刘书霞（1974− ），女，河北邢台人，博士，河北科技师范学院讲师，主要研究方向为大数据技术、深度学习、区块链等。