基于密度分布的鲁棒谱聚类算法

李 超[1)] 廖红梅[1),2)] 徐 晓[1),2)] 郭丽丽[1),2)] 丁世飞[1),2)]

[1)](中国矿业大学计算机科学与技术学院 江苏 徐州 221116)

[2)](矿山数字化教育部工程研究中心(中国矿业大学) 江苏 徐州 221116)

摘 要 谱聚类作为一种基于图论的聚类方法,通过相似性矩阵对数据进行特征分解或将数据投影到低维空间以实现更好的数据划分.谱聚类因其适用于复杂数据和非凸子簇而受到广泛的关注,并已成功应用在很多领域.然而,计算复杂度高、噪声敏感等问题会限制其聚类效果的进一步提升.针对这些问题,本文提出了一种基于密度分布的鲁棒谱聚类算法.首先,设置噪声系数以过滤少量的低密度噪声点.其次,根据密度峰值聚类具有的特性,即尽可能多地划分数据能够保证子簇内数据标签的一致性,新提出的算法能够在较少的子簇数和更高的簇内标签一致性上达到平衡,实现了对数据更加优质的划分.最后,基于簇间密度分布的相似性度量改善了谱聚类在密度不均匀数据集上的聚类效果.合成数据以及真实数据上的实验充分证明了新算法在 9 个最新改进算法中的有效性.在保证聚类效率的前提下,新算法在真实数据上的准确率、调整兰德系数和调整互信息的平均值上至少分别提升了 10.02%、22.11%和 15.76%.

关键词 谱聚类;密度分布;子簇相似性;局部峰值;噪声检测

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2024.02645

Robust Spectral Clustering Based on Density Distribution

LI Chao[1)] LIAO Hong-Mei[1),2)] XU Xiao[1),2)] GUO Li-Li[1),2)] DING Shi-Fei[1),2)]

[1)](*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu* 221116)

[2)](*Mine Digitization Engineering Research Center, Ministry of Education (China University of Mining and Technology), Xuzhou, Jiangsu* 221116)

Abstract Spectral clustering, as a classic clustering method based on graph theory, uses the similarity matrix to decompose the data or project the data into a low-dimensional space to achieve better data partition. In spectral clustering, the similarity matrix of data needs to be constructed first, and the similarity between data points is usually calculated by the Gaussian kernel function or k-nearest neighbors method. Then, the similarity matrix is transformed into a Laplacian matrix, and the eigendecomposition of the Laplacian matrix is carried out, and the eigenvectors are obtained and clustered by the k-means algorithm method. Finally, according to the clustering results, the data points belong to the cluster. Spectral clustering is of great significance in the field of data mining and pattern recognition. It is not only suitable for clustering problems, but also can be applied to graph segmentation, dimensionality reduction, feature selection and other fields, so it has a wide range of application values. However, the computational complexity of spectral clustering is high and may be limited when dealing with large-scale data sets. In addition, spectral clustering is sensitive to noise, because noisy data points may affect the construction of the similarity matrix and the calculation of the eigenvectors, resulting in instability and a decrease

收稿日期:2024-05-21;在线发布日期:2024-09-05.本课题得到国家自然科学基金项目(62276265,61976216)资助.**李 超**,博士研究生,主要研究方向为聚类分析、数据挖掘.E-mail:chaoli@cumt.edu.cn.**廖红梅**(通信作者),博士,讲师,中国计算机学会(CCF)会员,主要研究方向为机器学习、模式识别.E-mail:lhm@cumt.edu.cn.**徐 晓**,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为机器学习、聚类分析.**郭丽丽**,博士,讲师,中国计算机学会(CCF)会员,主要研究方向为深度学习、多模态情感计算.**丁世飞**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)杰出会员,主要研究领域为人工智能、模式识别、机器学习、数据挖掘.E-mail:dingsf@cumt.edu.cn.

in the accuracy of the clustering results. Especially in the case of no noise preprocessing or denoising, spectral clustering may incorrectly divide noisy data points into a certain cluster, affecting the final clustering results. Therefore, when dealing with data containing noise, it is necessary to properly clean or denoise the data before using spectral clustering to improve the effect. To address these problems, this paper proposes a robust spectral clustering algorithm based on density distribution. Firstly, the noise points between subclusters have lower local density; therefore, this paper sets the noise coefficient to filter a small number of low-density noise points from the perspective of different density levels. Secondly, according to the characteristics of density peaks clustering, that is, dividing the data as much as possible can ensure the consistency of the data label within the subcluster, the newly proposed algorithm can achieve a balance between a smaller number of subclusters and a higher consistency of the label within the cluster, and achieve a better division of the data. Finally, based on the density distribution information between all clusters, including the density mean and density standard deviation, a new similarity measure is proposed to improve the clustering effect of spectral clustering on data sets with uneven density. In the proposed algorithm, the parameter in spectral clustering, the non-negative Gaussian kernel bandwidth, is replaced by the more easily adjusted k-nearest neighbors, which can select the optimal parameters of the algorithm in a more limited range. Experiments on synthetic data and real data fully demonstrate the effectiveness of the new algorithm in nine state-of-the-art improved algorithms. Under the premise of ensuring the clustering efficiency, the average values of accuracy, the adjusted rand index and the adjusted mutual information are increased by 10.02%, 22.11% and 15.76% at least, respectively.

Keywords spectral clustering; density distribution; sub-cluster similarity; local peak; noise detection

1 引 言

聚类作为一种数据预处理的有效方法,旨在将数据集中的对象分组成具有相似特征的簇.簇中的对象具有较高的相似性和更加一致的局部结构,相较于逐个处理数据的方式,能够从更加宏观的角度分析不同簇之间的全局不一致性[1].聚类分析能够帮助人们更好地理解和探索数据,揭示数据内在的组织和特征.此外,通过将数据分组成簇,可以将大量的数据压缩成更简洁的形式,从而减少数据的复杂性,使数据更易于理解和处理[2].聚类具有广泛的研究价值,包括预处理[3]、特征工程[4]、目标客户分析[5]、图像和文本分析[6]以及生物信息学[7].

根据其工作原理、算法复杂度和应用领域的不同,聚类算法可以分为基于原型的聚类、层次聚类、密度聚类、基于网格的聚类以及基于模型的聚类[8].在众多聚类算法中,图聚类是一种基于图划分理论的聚类算法[9].它将数据点表示为图的顶点,并根据数据间的相似性构建图的边.然后通过图划分算法将图划分成若干个连通子图,每个子图内的相似性尽可能大,而子图间的相似性尽可能小.图聚类能够识别任意形状的簇,并且在处理复杂数据时具有更高的准确性和稳定性.作为图聚类中的一种代表性算法,谱聚类受到极大的关注.不同的图划分方法产生了不同的谱聚类,包括最小切割准则、比率切割准则、平均切割准则、规范切割准则、最小最大切割准则、递归 2-way 划分和 k-way 划分[10].尽管不同的切割准则都对谱聚类有所改进,但从最细粒度划分图节点的特性导致它们在衡量节点相似性上出现偏差,并且具有较高的时间复杂度 $O(n^3)$.

为改进上述不足,许多学者做出了自己的优化.为提升谱聚类的聚类精度,Wang 等人[11]针对 p 谱聚类不适用于非线性结构的数据的问题,提出一种流形 p 谱聚类算法.该算法还使用一种混沌序列改进的麻雀搜索算法(Sparrow Search Algorithm,SSA)调整参数 p.Nie 等人[12]将谱聚类中相似性测量和数据聚类的两步方法整合到一个目标函数中,迭代地学习相似性矩阵和谱嵌入.Zhang 等人[13]提出一种局部密度自适应相似度量,利用两个数据点之间的局部密度来缩放高斯核函数.Bian 等人[14]使用双索引模糊 C-means 聚类算法来确定任何一对数

据点之间的适当模糊相似性.上述聚类算法仍然在单个数据点上计算相似性度量,无法改进谱聚类的运行效率.对于此,Wang 等人[15]将网格划分引入谱聚类,以在更粗粒度的层面计算对象间的相似性,加快速率.网格划分能够大大缩减谱聚类中待处理的数据量,但会忽略数据点的局部一致性,将不同子簇的边界点划分到同一网格中,影响聚类精度.二分谱图划分(Bipartite Spectral Graph Partition,BSGP)采用小规模相似矩阵的特征分解或奇异值分解(Singular Value Decomposition,SVD)来处理任意矩阵,具有比谱聚类中特征分解更快的效率[16-17].但是 SVD 的时间复杂度与特征维度的平方成正比,与数据量也成正比.当数据集的规模很大或特征维数很高时,该方法的使用会受到限制.Song 等人[17]提出加权双边 k-means 算法以提高 BSGP 在大规模或高维数据上的速率.基于锚点选择的快速谱聚类与 BSGP 十分相似,只是样本点和特征之间的相似性转变为样本点和锚点之间的相似性.鉴于锚点的数量可以控制在较小的范围内,从而基于锚点的谱聚类算法比 BSGP 更有效.Huang 等人[18]使用混合采样的策略选择高质量的锚点,提出 USPEC(Ultra-Scalable Spectral Clustering)模型,以提升谱聚类效果.另外,多个 USPEC 被进一步集成到集成聚类框架 USENC(Ultra-Scalable Ensemble Clustering)中,以增强基聚类的鲁棒性,同时保持高效.Li 等人[19]使用分治的锚点选择策略进一步提升 USPEC 的效果.该算法使用 k-means 选取锚点的复杂度由 $O(npdt)$ 缩减为 $O(nqd)(q<p)$,q 为每次 k-means 划分的类数目上限,需手动设置;p 为锚点个数;n、d、t 分别为样本量、维度以及迭代次数.除了使用二分图划分以加速谱聚类外,构建粗粒度子簇以缩小相似性矩阵的规模是另一种加速谱聚类的方法.Xie 等人[20]提出 GBSC(Efficient Spectral Clustering based on Granular-Ball),用粒球来缩减数据量,使输入到谱聚类的相似性矩阵的规模远小于原始的样本数.

密度峰值聚类(Density Peaks Clustering,DPC)[21]是密度聚类中的一个代表性算法,因其能够处理非线性的和复杂的数据集而被熟知.DPC 算法具有的缺点之一是非中心分配过程中的“误差累计”.当某一样本被错误分配后,后续样本的分配过程可能被该样本误导.因此,许多学者采用划分-合并的方法对 DPC 进行改进[22-24].通过对这些算法进行对比可知,DPC 在划分尽可能多的子簇时,不同子簇内的样本的一致性相对较高,即划分后的子簇为原始标签下某一类的子集.以 DPC 划分后的子簇作为谱聚类中的最小处理单元可以同时实现效率和精度上的提升.Cheng 等人[25]提出的近似谱聚类算法 DCDP-ASC(Novel Approximate Spectral Clustering with Dense Cores and Density Peaks),同时结合 DPC 和谱聚类的优点,以弥补彼此的不足.DCDP-ASC 和 GBSC 都是在更粗粒度的层次对谱聚类进行改进,但这些算法在计算子簇相似性时考虑并不充分,仍需进一步改善.

通过分析上述改进的聚类算法,本文提出一种基于密度分布的鲁棒谱聚类算法(Robust Spectral Clustering Algorithm based on Density Distribution,RSCDD),将 DPC 与谱聚类相结合,在保证算法效率的同时进一步提升聚类精度.相较于以往的基于 DPC 的谱聚类改进算法,如 DCDP-ASC、RSCDD 更注重子簇划分过程中的局部一致性,避免对不同密度子簇边界区域的错误分配.RSCDD 还从相邻子簇密度分布特性的角度出发,设计更加合理的子簇相似性准则,以进一步提升聚类精度.此外,针对数据集中原有的噪声,RSCDD 设计了有效的噪声识别策略,增强了算法的鲁棒性.本文的主要贡献如下:

(1)提出一种基于密度分布的鲁棒谱聚类算法,可以在保证算法效率的前提下,提升聚类结果的精度.

(2)根据互 k 近邻(Mutual K-Nearest-Neighbors,MKNN),使 DPC 构建尽可能多的子簇,增强子簇内的局部一致性,使得标签不一致的数据被充分划分到不同子簇.

(3)根据正态分布中的“3σ 原则”,删除低密度的噪声,从而增强算法的鲁棒性.

(4)通过合成数据上的可视化结果和真实数据集上的对照实验,证明了 RSCDD 对噪声的鲁棒性、对谱聚类精度的提升以及算法运行的有效性.

本文在第 2 节中,介绍谱聚类、部分算法对谱聚类的改进,用来与本文提出的算法进行对照;在第 3 节中,对 RSCDD 算法进行详述,包括噪声的识别、子簇内样本局部一致性的保持以及基于密度分布的子簇相似性的计算;第 4 节通过实验验证新算法的效果,证明其相对于其他的表现较好的方法的优越性;最后总结全文,对未来工作进行展望.

2　相关工作

2.1　谱聚类

给定数据集 $X=\{x_1,x_2,\cdots,x_n\}$,n 为数据容量.谱聚类的实现过程如下:

(1)定义任意样本对 (x_i,x_j) 之间的相似性 w_{ij},

获得相似性矩阵 $\boldsymbol{W}=(w)_{ij}$，$i,j\in\{1,2,\cdots,n\}$.

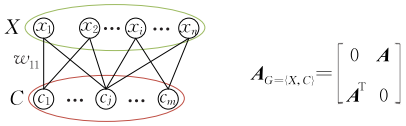
(2) 根据获得的相似性矩阵 $\boldsymbol{W}$，计算度矩阵 $\boldsymbol{D}=\mathrm{diag}(d_i)$，其中 d_i 是矩阵 $\boldsymbol{W}$ 的第 i 行(或第 i 列)中元素的和，计算公式为 $d_i=\sum_{t=1}^{n}w_{it}=\sum_{t=1}^{n}w_{ti}$，$i=1,2,\cdots,n$. 根据矩阵 $\boldsymbol{W}$ 和 $\boldsymbol{D}$，可以计算拉普拉斯矩阵 $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{W}$.

(3) 当根据相似性矩阵 $\boldsymbol{W}$ 划分数据集时，通常规定子图对应的子簇内的相似性尽可能大，而簇之间的相似性尽可能小. 根据 $\boldsymbol{W}$ 直接选择不相似的样本对 (x_i,x_j) 是 NP 难问题. 但是经过数学推导，如果把它们写成 Rayleigh 熵的形式，就可以利用 Rayleigh 熵的性质得到图划分目标函数的松弛解[26]. 因为 Rayleigh 熵的第 i 个最小值对应于拉普拉斯矩阵的第 i 个最小特征值，且极值在相应的特征向量处取得. 可见，图的拉普拉斯矩阵的特征向量中隐含了顶点的类别信息，于是可以通过计算拉普拉斯矩阵的特征值和特征向量，将图划分的组合优化问题转为数值优化问题，使其可以在较短的时间内解决[27].

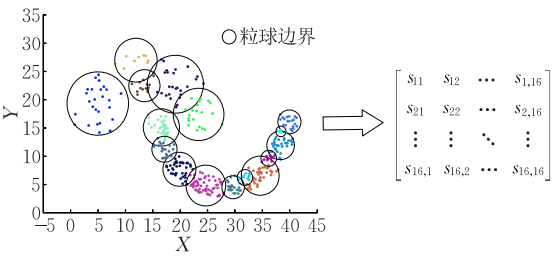
谱聚类利用拉普拉斯矩阵 $\boldsymbol{L}$ 的性质选择与最小的 c 个特征值对应的特征向量，形成特征向量矩阵 $\boldsymbol{F}\in\mathbb{R}^{n\times c}$，$c$ 为需要被划分的类数.

(4) 特征矩阵 $\boldsymbol{F}$ 的每一行都可以看作是空间 $\mathbb{R}^c$ 中样本的表示，最终结果可以通过使用 k-means 对 $\boldsymbol{F}$ 进行聚类获得.

从上述谱聚类的执行过程可以看出，它的时间复杂度主要由两部分组成，包括构建相似性矩阵($O(n^2)$)和拉普拉斯矩阵的特征分解($O(n^3)$)[28]. 为进一步改进谱聚类，依据不同改进策略的算法被相继提出. 其中，基于锚点选择的方法是一类具有代表性的算法[18-19,28]. 该类方法如图 1(a)所示，原始数据集 X 被连接到锚点集合 C 中的 k 个最近邻，以此生成亲和矩阵 $\boldsymbol{A}\in\mathbb{R}^{n\times p}$，$p(=|C|)$为锚点的个数，需手动设置. $\boldsymbol{A}$ 为稀疏矩阵，共有 $n\times k$ 个非零元素. 相较于直接计算 X 内样本的相似性，亲和矩阵 $\boldsymbol{A}$ 可用来描述 X 中样本的间接相似性，如图 1(a)中 x_1 和 x_2 都被连接到 c_1 和 c_j，因此 x_1 和 x_2 在 X 内会有更高的相似性. 假设 G 为 X 和 C 的并集，以 C 为过渡，X 的内部相似性进一步由 G 的亲和矩阵 $\boldsymbol{A}_G$ 间接表示，如图 1(a)的 $\boldsymbol{A}_G$ 所示. 当 X 容量过大时，以 X 和 C 为顶点的二分图(Bipartite Graph)[18-19]被用来加速计算 X 的谱嵌入，即拉普拉斯矩阵的特征向量. Song 等人[17]指出基于锚点的谱聚类通过矩阵 $\boldsymbol{A}$ 计算数据集 X 的邻接矩阵，表示为 $\boldsymbol{A}\boldsymbol{D}_{\boldsymbol{A}}^{-1}\boldsymbol{A}^{\mathrm{T}}\in\mathbb{R}^{n\times n}$，$\boldsymbol{D}_{\boldsymbol{A}}$ 为对角矩阵且第 (i,i) 个元素为 $\boldsymbol{A}$ 的第 i 列的和. 求解 $\boldsymbol{A}\boldsymbol{D}_{\boldsymbol{A}}^{-1}\boldsymbol{A}^{\mathrm{T}}$ 上的最小 Ncuts 问题时，应解决 $\boldsymbol{A}\boldsymbol{D}_{\boldsymbol{A}}^{-1}\boldsymbol{A}^{\mathrm{T}}$ 的特征分解，并找到与 c 个最大特征值对应的特征向量. 该过程相当于对矩阵 $\boldsymbol{D}_{\boldsymbol{A}}^{-1/2}\boldsymbol{A}^{\mathrm{T}}$ 执行 SVD，并找到与 c 个最大奇异值相关的右奇异向量. 上述 SVD 的时间复杂度为 $O(p^2n)$，$p\ll n$，因此对矩阵 $\boldsymbol{D}_{\boldsymbol{A}}^{-1/2}\boldsymbol{A}^{\mathrm{T}}$ 进行奇异值分解能够大大缩短算法的运行时间，并且在矩阵存储上也占用更少的资源. Huang 等人[18]使用 $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{D}_{\boldsymbol{A}}^{-1}\boldsymbol{A}\in\mathbb{R}^{p\times p}$ 的拉普拉斯矩阵的非负特征值和特征向量间接计算 $\boldsymbol{A}_G$ 的拉普拉斯矩阵 $\boldsymbol{L}_G=\boldsymbol{D}_G-\boldsymbol{A}_G$ 的非负特征值和特征向量 $\boldsymbol{u}_i\in\mathbb{R}^{(n+p)\times 1}$，$\boldsymbol{u}_i$ 的前 n 个值等于 X 的谱嵌入，此时时间复杂度为 $O(p^3)$. $\boldsymbol{A}^{\mathrm{T}}$ 和 $\boldsymbol{A}'$ 表示 $\boldsymbol{A}$ 的转置，$\boldsymbol{D}_G$ 表示 $\boldsymbol{A}_G$ 的度矩阵. 图 1(b)显示了在粗粒度上改进谱聚类的实现. 这类方法首先使用快速的划分方法生成尽可能少的子簇，然后构建子簇的相似性矩阵 $\boldsymbol{S}$ 进行特征分解. 由于生成的子簇数远小于初始的数据量，因而能够极大地缩减运行中的时间和存储消耗. 这一类的改进算法包括 GBSC[20]、DCDP-ASC[25]. 除上述方法外，一些算法还采用稀疏矩阵的方式改进谱聚类. Wu 等人[29]通过随机装箱特征(Random Binning Features，RB)生成的大型稀疏特征矩阵的内积来隐式地近似相似性矩阵，并由改进的 SVD 求解器来计算特征向量. Lucińska 等人[30]基于改进的互近邻图构建稀疏矩阵，使用高斯核评估相似性度量. Yang 等人[31]在使用近邻图构建稀疏矩阵的同时，定义了可调线段长度. 它可以调整不同密度区域中的距离，对高密度区域中的距离压缩，而放大低密度区域的距离.



(a) 基于锚点选择的方法



(b) 粗粒度划分的方法

图 1 谱聚类改进的两种示例

2.2 粗粒度下的谱聚类改进算法

本文采用图 1(b)中的方法改进谱聚类，因此需简要回顾相关算法(如 GBSC 和 DCDP-ASC)的实现过程，从而提出新的改进算法.

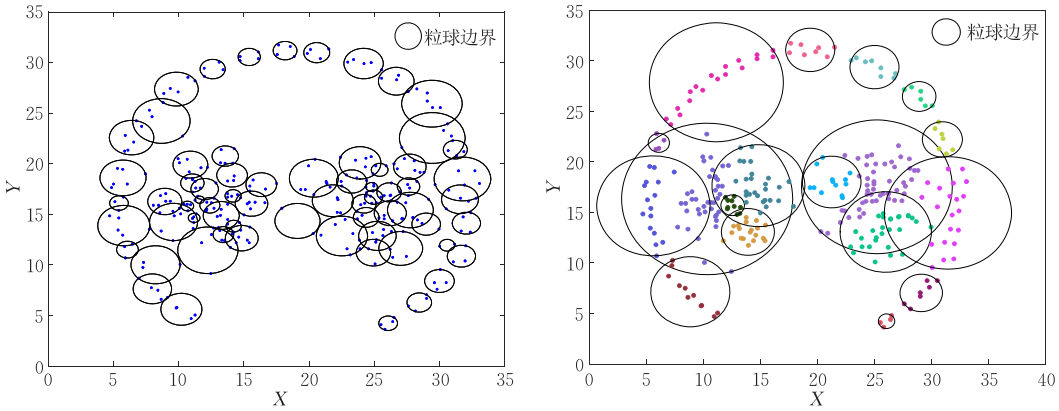
GBSC 采用粒球划分的方式生成图 1(b)中的子簇,能够在较短的时间内实现这一过程[20]. 当数据中存在噪声时,包含噪声的大半径粒球会影响聚类效果. 为解决这一问题,GBSC 将半径大于所有粒球半径的均值和中值的最大值二倍的粒球删除,即 $r_i>2\times\max(mean(r),median(r))$. 根据粒球间的距离计算高斯核相似性时,新定义的粒球距离计算公式如式(1)所示:

$$d(GB_i,GB_j)=d(cen_i,cen_j)-(r_i+r_j) \tag{1}$$

式中 GB_i 表示生成的第 i 个粒球,cen_i 为 GB_i 的中心,r_i 为其半径. $d(\cdot)$为点间的欧式距离. 当式(1)中的距离为负值时,将其设为 0. 由式(1)可知,粒球之间的相似性计算简单,能够保证计算效率. 但可能造成无法有效衡量复杂数据的相似性,比如在可变密度的数据中,粒球中心间的距离并不足以代表其不相似性.

DPC 通过局部密度 ρ_i 和相对距离 δ_i 获取峰值点和分配非中心点,其中 ρ_i 表示截断距离 d_c 邻域内的样本数,δ_i 表示距离密度更高点的最短距离. 由于 DPC 中存在“误差累计”,低密度子簇往往被误分配到相邻的高密度子簇. 通过设置更多的峰值点,每个子簇内的样本量将进一步减少,导致 DPC 在分配非中心点时由全局一致性向局部一致性转移. 也就是说,在尽可能多的簇数目下,DPC 和 k-means 算法具有相似的划分效果. 此时,DPC 划分后的任一子簇都是真实标签下某一类数据的子集,此时的簇内一致性达到最大.

尽管粒球和 DPC 都能对数据集进行划分,但 DPC 更容易在保持子簇内一致性的同时产生更少的子簇,如图 2 所示. 图 2(b)显示了 DPC 在 Pathbased 数据集上生成 18 个子簇后的结果,而图 2(a)是 Pathbased 数据集下粒球划分的结果,其粒球数目明显超过 18. 因此,DPC 的划分策略在确定子簇数目和保持簇内样本一致性方面能够达到很好的平衡.



(a)粒球划分　　(b)DPC划分

图 2　不同划分方法在 Pathbased 数据上的示例

DCDP-ASC 算法在 DPC 中引入 k 近邻来加速子簇的生成,并将子簇定义为密集核(Dense Cores). 通过密集核定义粗粒度子簇后,子簇相交区域中样本点的密度被用来计算子簇间改进的距离,具体公式如下:

$$CD(i,j)=\begin{cases}\dfrac{d(i,j)}{|CN(i,j)|\times\sum\limits_{o\in CN(i,j)}\rho_o}, & CN(i,j)\neq\varnothing\\ maxd\times d(i,j), & \text{其他}\end{cases} \tag{2}$$

式中,$CN(i,j)$表示两个子簇的交集,即子簇内所有样本的 k 近邻的集合与另一子簇重复存在的点. ρ_o 为局部密度,$maxd$ 为所有密集核之间欧式距离 $d(i,j)$的最大值. 然后,DCDP-ASC 根据式(2)中的距离计算所有簇之间的测地距离. 由测地距离计算的高斯核相似性被作为谱聚类的输入. DCDP-ASC 使用子簇交集的局部密度的绝对值而忽视了数据集内密度的变化,因而聚类结果进一步受限. 具体分析见 3.3 节.

考虑以上算法存在的不足,本文进一步提出密度分布下的鲁棒谱聚类算法,在保证算法效率的同时进一步提升精度. 综合分析上述 DCDP-ASC 和 GBSC 的实现过程,新提出的 RSCDD 与 DCDP-ASC 和 GBSC 具有如下的异同:

(1)相同点. 三个算法都是粗粒度下的谱聚类改进算法,首先将数据集划分成数量更少的粗粒度子簇,然后设计新的子簇间的距离以计算子簇相似性矩阵,最后将相似性矩阵输入谱聚类获得聚类结果.

(2)不同点. 在具体方法实现上,三个算法有所不同,主要体现在粗粒度子簇生成以及子簇间距离计算上. GBSC 使用粒球划分数据,粒球划分会生成更多的子簇,进而影响谱聚类的效率. DCDP-ASC 和 RSCDD 采用图 2(b)所示的 DPC 划分,将 DPC 中

最近的高密度点压缩至 k 近邻或互 k 近邻，最近的高密度点又称为代表点(见第 3.2 节定义 1). DCDP-ASC 通过自然近邻方法确定近邻数 $nb(i)$，其代表点为 $Rep(i)=\arg\max_{\tau}(\rho_{\tau})$，$\tau$ 属于 x_i 和 x_i 的 $nb(i)$ 个近邻的并集. RSCDD 的代表点为 $Rep(i)=\arg\min_{\tau}(d(i,\tau))$，$\tau$ 属于 x_i 的互 k 近邻且 $\rho_{\tau}>\rho_i$. 因此 RSCDD 的样本与其代表点的距离更近，相似性也更高. 在计算子簇间的距离时，GBSC 仅采用簇中心间的欧式距离，如式(1). DCDP-ASC 在距离中加入簇间交点的密度绝对值，相似性有所改进，但不适用于密度变化的数据集，如 3.3 节图 7 所示. RSCDD 使用更加复杂的簇间距离，综合考量簇间分布的相对密度信息与距离信息，如 3.3 节的式(9)，聚类效果有所提升.

RSCDD 算法的具体实现过程将在第 3 节中详细介绍.

3 方 法

在本节中，RSCDD 算法被详细地描述. 其实现过程主要包括三个部分：噪声预处理、子簇划分和子簇相似性计算. 噪声预处理是在所有步骤前删除一些对结果产生偏差的样本点. 子簇划分将 MKNN 引入子簇生成过程以避免相邻密度差对划分结果的影响. 子簇相似性计算是 RSCDD 算法的核心，其直接决定了最终聚类结果的质量. 本小节最后给出 RSCDD 算法的伪代码以及算法的复杂度分析.

3.1 噪声预处理

在 RSCDD 中，首先使用 k 近邻来计算新的样本局部密度[32]，公式如下：

$$\rho_i=\frac{k}{\sum_{j\in knn(i)}d(i,j)} \tag{3}$$

式中，$knn(i)$ 表示第 i 个样本的 k 近邻.

考虑到数据集中噪声对聚类结果的影响，本文在聚类开始前使用式(3)删除一些低密度点，这些点的密度阈值如下：

$$\rho_{\theta}=\bar{\rho}_X-\mu\times\sigma_X \tag{4}$$

式中，$\bar{\rho}_X$ 和 σ_X 分别表示数据集 X 中所有点密度的均值与标准差. 由高斯分布中的“3σ 原则”，可以将噪声系数 μ 的取值范围设为{1,2,3}以识别不同密度水平下的低密度点. 值得注意的是，因为噪声点的密度通常较低，所以式(4)中的密度阈值不考虑密度过高的异常值. 低于密度阈值的异常点不再参与数据的子簇划分.

图 3 显示了 RSCDD 在四个合成数据上的噪声识别情况，蓝色点表示正常点，红色点为识别出的噪声. 可以看出，RSCDD 在不破坏子簇结构的情况下基本上识别出数据中的噪声点. 因为 E6、T4 和 T7 含有更多的噪声点，所以它们的噪声系数 μ 设为 1.

相较于 T4 和 T7，E6 中识别的噪声包含部分正常点. 图 4 中的密度降序曲线用于进一步分析其中的原因，式(3)中 $k=5$. 由图 4 可知，T4 和 T7 的

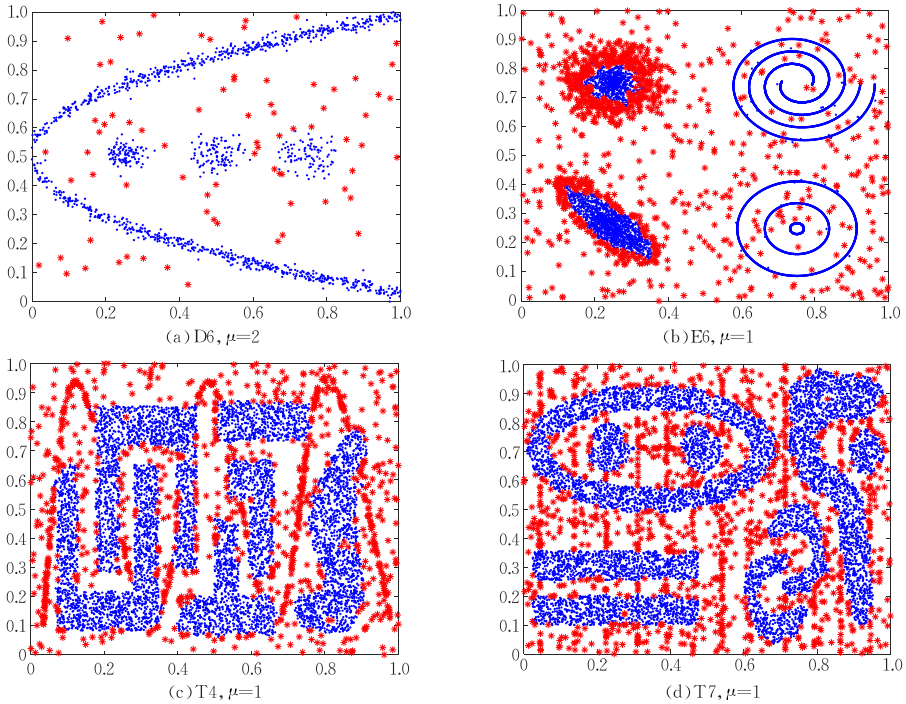


图 3 RSCDD 在四个合成数据上的噪声识别

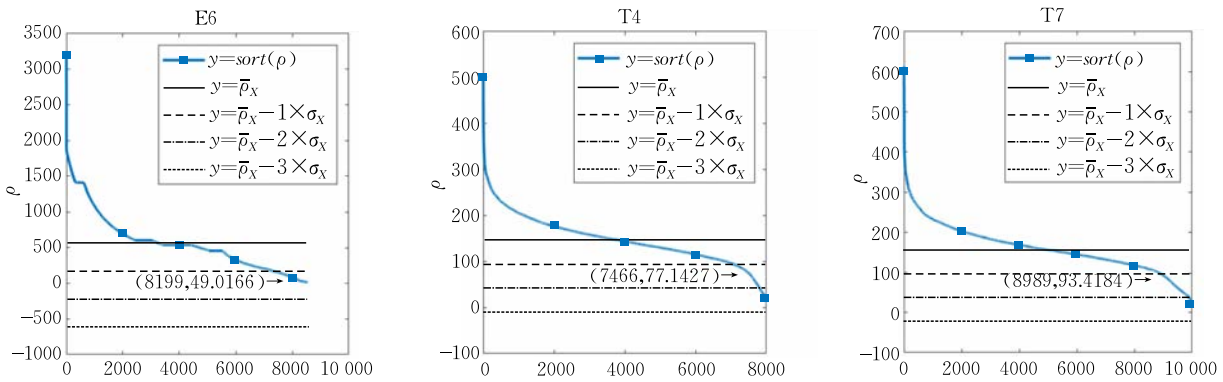
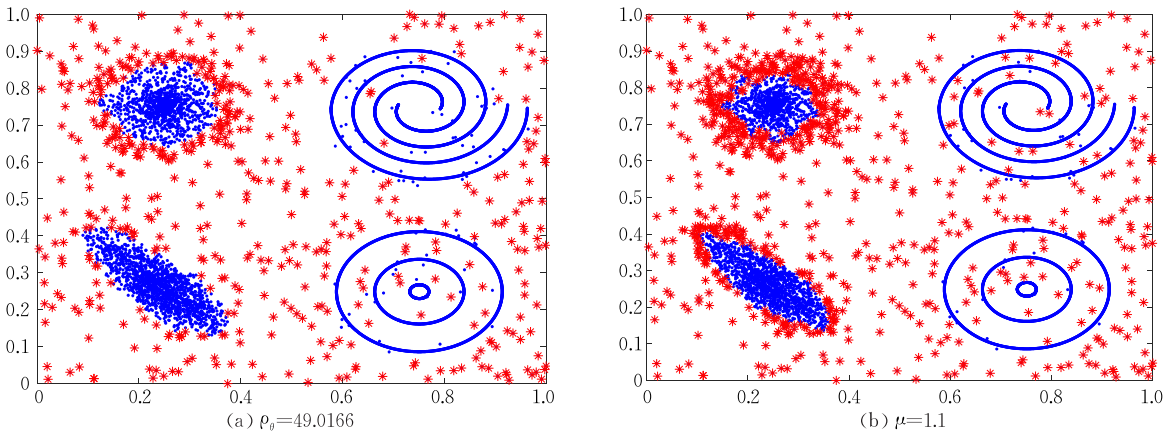


图 4　密度降序曲线

低密度区域要比 E6 的更容易识别,且 T4 和 T7 的低密度区域位于 $\mu=1$ 的阈值附近,8199、7466 和 8989 分别为 E6、T4 和 T7 在低密度区域的斜率变化最大的点. E6 的低密度区域的范围较小,根据 μ 设定的取值范围,只有 $\mu=1$ 时,才能识别全部噪声点和部分正常点.

根据图 4 中显示的 E6 在低密度区域中斜率变化最大点的纵坐标,图 5(a)显示了 $\rho_\theta=49.0166$ 时 E6 数据集上的噪声识别情况. 可以看到,E6 中左侧凸形子簇中的更多点被正确识别为正常点. 但是根据密度降序曲线确定密度阈值在更多数据上是十分困难的,基于此,实验 4.4 节中的参数分析给出了复杂数据上参数 μ 的默认值为 1.1. 图 5(b)显示了 $\mu=1.1$ 时,E6 中噪声的识别情况. 相对于图 3(b),$\mu=1.1$ 确保 RSCDD 能够正确识别更多的正常点. 此外,图 3(b)中被识别为噪声的正常点多为凸形子簇的边界点,它们的分配依赖于中心的高密度点,因此对 RSCDD 的聚类结果影响较小.



(a) $\rho_\theta=49.0166$　　(b) $\mu=1.1$

图 5　RSCDD 在 E6 上的噪声识别

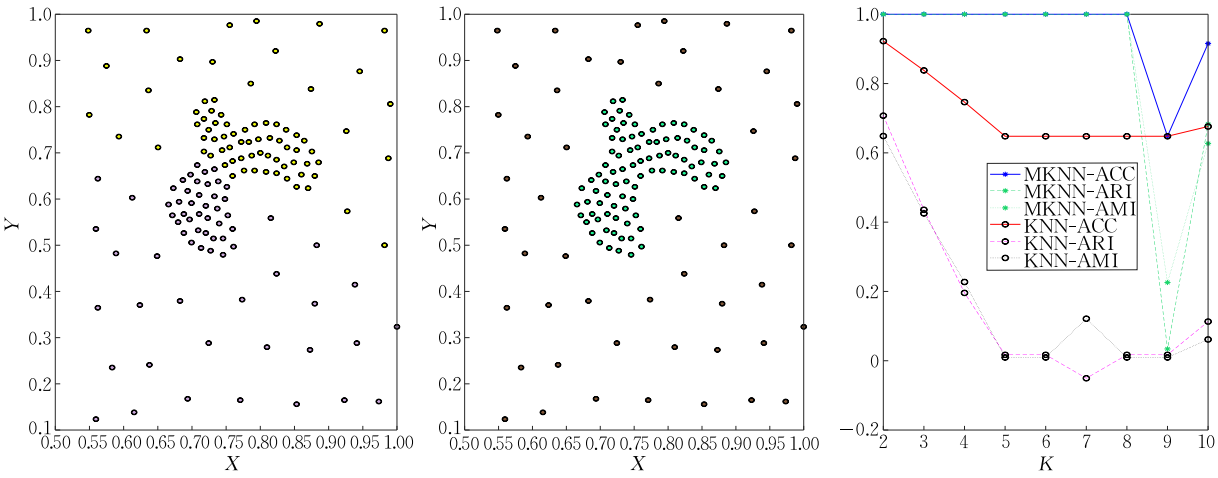
3.2　子簇划分

删除低密度点后,子簇由代表点相同的所有点组成. 代表点定义如下:

定义 1.　代表点. 代表点初始化为空,表示为 $Rep(i)=0$. 如果 $j\in knn(i)\wedge i\in knn(j)$ 且 $\rho_j>\rho_i$,则 $Rep(i)=\arg\min_j d(i,j)$.

根据式(3)中的局部密度和定义 1 中的代表点,每个样本可被划分到其代表点所在的子簇. 如果 $Rep(i)=0$,则该点被选为中心点,以形成新的子簇. DCDP-ASC[25] 同样给出代表点的定义,但代表点 j 仅在 i 的 k 近邻内,而没有规定 i 在 j 的 k 近邻内,即只满足 $j\in knn(i)$. 定义 1 的不同之处在于代表点一定在该点的 MKNN 范围内,即 $j\in knn(i)\wedge i\in knn(j)$. 通过这一条件,RSCDD 能够更好地处理存在密度差的相邻子簇. 图 6 给出了条件 $i\in knn(j)$ 的有效性测试. 如果定义 1 存在 $i\in knn(j)$,则样本和其代表点之间满足互 k 近邻关系(MKNN),反之,样本和其代表点之间仅存在 k 近邻关系(KNN). 因此,分别以 MKNN 和 KNN 来表示定义 1 中 $i\in knn(j)$ 的有无. ACC、ARI 和 AMI 为聚类评价指标,可参考第 4 节.

从图 6 中的比较结果可知,在相邻子簇间存在密度差的数据集上,采用 MKNN 条件的 RSCDD 算法能更加有效.



(a) KNN, k=5 (b) MKNN, k=5 (c) KNN vs MKNN

图 6 MKNN 和 KNN 条件下,RSCDD 在不均匀密度数据上结果的比较

3.3 子簇相似性

在子簇划分完毕后,RSCDD 需要计算任意两个子簇间的相似性以便于后续的子簇合并. 新提出的相似性度量更注重于子簇间的密度相对值而非绝对值. 图 7 显示了 Pathbased 数据上不同密度子簇的划分,其中 O1 位于低密度子簇间的交集,而 O2 位于不同密度子簇的边界. 在式(2)中,当其他变量一样的情况下,子簇间交集的密度越大,则两个子簇间的距离越近,其相似性越高. 因此,O2 区域相邻的两个子簇更容易被合并到一起. 为此,本文将子簇交集的密度与相邻子簇的密度进行比较,以确定密度相似性的相对值而非绝对值. 以下定义给出了子簇间相似性的具体计算过程.

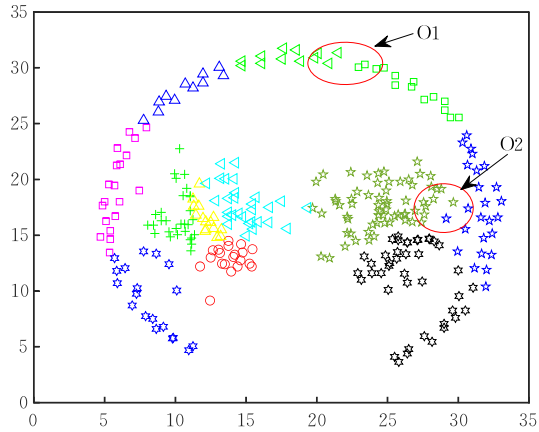


图 7 不同密度子簇间交集的相似性

定义 2. 相交系数. 设 CE_i 为子簇 C_i 的扩展集,即子簇 C_i 中的点与它们的 k 近邻的并集. 子簇 C_i 和 C_j 的相交系数为

$$perc_{ij}=sign_{ij}\times sign_{ji}\times\frac{|CE_{ij}|}{|C_i|+|C_j|} \tag{5}$$

式中 CE_{ij} 为 CE_i 和 CE_j 的交集. $sign_{ij}=sign(|CE_i\cap C_j|)$,$sign(\cdot)$ 为符号函数,当 $x>0$ 时,$sign(x)=1$;当 $x=0$ 时,$sign(x)=0$.

定义 3. 连通性. 根据子簇 C_i 和 C_j 之间样本点的距离,子簇间的连通性如下:

$$con_{ij}=mean(\{d(p,q)\}) \tag{6}$$

其中,$p\in CE_i\cap C_j$,$q\in CE_j\cap C_i$.

定义 4. 密度均值相似性.

$$pavg_{ij}=\frac{\min(\bar{\rho}_{C_i},\bar{\rho}_{C_j},\bar{\rho}_{CE_{ij}})}{\max(\bar{\rho}_{C_i},\bar{\rho}_{C_j},\bar{\rho}_{CE_{ij}})} \tag{7}$$

式中,$\bar{\rho}_{C_i}$,$\bar{\rho}_{C_j}$,$\bar{\rho}_{CE_{ij}}$ 表示集合 S 中所有点密度的均值.

定义 5. 密度标准差相似性.

$$std_{ij}=\frac{\min(\sigma_{C_i},\sigma_{C_j},\sigma_{CE_{ij}})}{\max(\sigma_{C_i},\sigma_{C_j},\sigma_{CE_{ij}})+\sigma_{C_{ij}}} \tag{8}$$

式中,σ_{C_i},σ_{C_j},$\sigma_{CE_{ij}}$,$\sigma_{C_{ij}}$ 表示集合 S 中所有点密度的标准差,C_{ij} 为 C_i 和 C_j 的并集.

最后,子簇间的距离公式为

$$newd_{ij}=perc_{ij}\times con_{ij}\times(1-pavg_{ij}^2)\times(1-std_{ij}) \tag{9}$$

式(9)的子簇距离的计算过程只在相邻子簇存在交集的情况下进行,即 $perc_{ij}>0$;否则子簇间不存在直接连通的距离,可通过 Floyd 算法由相邻簇间的距离间接得到. 假设由式(9)得到的子簇间的测地距离为 $newgd_{ij}$,通过高斯核函数可计算出簇间的相似性 SIM_{ij},具体公式如式(10)所示.

$$SIM_{ij}=\exp\left(-\left(\frac{newgd_{ij}}{\sigma}\right)^2\right) \tag{10}$$

$$\sigma=mean(\{newd_{ij}\mid perc_{ij}>0\}) \tag{11}$$

3.4 算法流程与分析

本小节给出 RSCDD 算法的流程,如算法 1 所示.

算法 1. RSCDD 算法.

输入:数据集 $X\in\mathbb{R}^{n\times m}$,簇数目 $nclust$,近邻数 k,噪声系数 μ

输出:聚类结果$\{S_i\}$,$i=1,2,\cdots,nclust$

1. 将数据 X 进行 Min-Max 归一化,对于第 v 维特征,$x_{\cdot v}=(x_{\cdot v}-\min(x_{\cdot v}))/(\max(x_{\cdot v})-\min(x_{\cdot v}))$;

2. 通过 kd-tree 和式(3),计算样本新的局部密度;
3. 由系数 μ 和式(4)计算密度阈值 ρ_θ,删除低密度点集 $\{i|\rho_i<\rho_\theta, i=1,2,\cdots,n\}$;
4. 对于剩余的高密度点,通过 kd-tree 和式(3),重新计算样本的局部密度;
5. 由定义 1,计算样本的代表点 $Rep(i)$以生成子簇;
6. 根据式(10),计算所有子簇间的相似性 SIM;
7. 将相似性矩阵作为谱聚类的输入获得新的簇标签 $label(C_i)$;
8. 具有相同标签的簇包含的所有点被聚类到同一子簇,即 $x_p\in C_i$, $x_q\in C_j$,如果 $label(C_i)=label(C_j)$,则 $label(x_p)=label(x_q)$;
9. RETURN 聚类结果$\{S_i\}$.

在算法 1 中,由于谱聚类算法以 k-means 划分降维后的点获得聚类结果,因此多次的聚类结果可能不一致.对此,在实验时设置固定的种子,以保证其他条件不变时 k-means 每次的运行结果都相同.

在算法 1 中,步骤 1～3 是算法的预处理,用于数据归一化并删除低密度的噪声点.其时间复杂度主要由 kd-tree 的近邻搜索所决定,为 $O(n\log(n))$.步骤 4 与步骤 2 的执行过程相同,但却是在去除噪声后的样本上执行的,因此时间复杂度略低于 $O(n\log(n))$.由于数据中的噪声数量所占比重较小,因此步骤 4 的时间复杂度近似为 $O(n\log(n))$.步骤 5～6 用于计算子簇间的相似性.假设生成的子簇数为 $nsc(\ll n)$,则步骤 6 的时间复杂度为 $O(nsc^3)+O(nsc^2)$.步骤 5 查找代表点,时间复杂度为 $O(nk)$.步骤 7 和 8 的主要时间消耗在于谱聚类的运行,时间复杂度为 $O(nsc^3)$.

因此,算法 1 的时间复杂度为 $O(2n\log(n))+O(nsc^3)+O(nsc^2)+O(nk)+O(nsc^3)\sim O(2n\log(n))$, $nsc,k\ll n$.因此 RSCDD 的时间复杂度较低,保证了算法的运行效率.

RSCDD 在运算过程中,主要的空间消耗在于存储 k 近邻和子簇的相似性矩阵.因此 RSCDD 的空间复杂度为 $O(nk)+O(nsc^2)$.

4　实验与分析

在本节中,采用合成数据和真实数据对 RSCDD 算法的有效性进行验证.合成数据皆为二维数据,以便于对聚类结果进行可视化.真实数据集上的结果通过三个常用的聚类评价指标进行定量分析,它们分别是准确率 ACC(Accuracy)[22]、调整兰德系数 ARI(Adjusted Rand Index)[33]和调整互信息 AMI(Adjusted Mutual Information)[33].对于 ACC 来说,当处理类不平衡数据时,由于某些类别的样本数量较少,即使这些少数类被错误地划分,对 ACC 的影响可能并不明显.ACC 的取值范围为[0,1],值越大表示聚类结果越好.ARI 和 AMI 被用来弥补 ACC 在不平衡数据上的不足.相比之下,ARI 和 AMI 考虑了真实类别和聚类结果之间的匹配关系,可以更好地衡量聚类的效果,受不平衡数据的影响较小.ARI 和 AMI 的取值范围为[－1,1],值越大表示聚类结果与真实情况越吻合.

所有实验的环境为:Windows11 操作系统,MATLAB R2021b 编译环境,i9-12900HX 处理器和 16 GB 的 RAM.

4.1　合成数据上的比较

在合成数据集上,RSCDD 与相似的改进算法 DCDP-ASC[25]和 GBSC[20]进行比较.采用的合成数据如图 8、图 9 和图 10 所示.DCDP-ASC算法通过

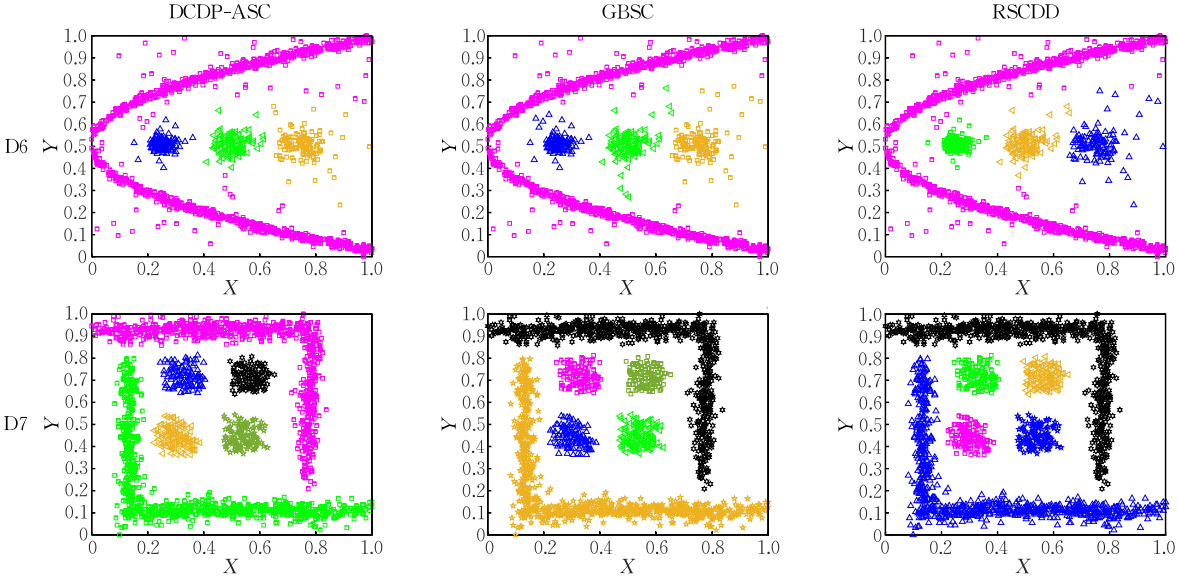


图 8　合成数据 D6、D7 上的聚类结果可视化

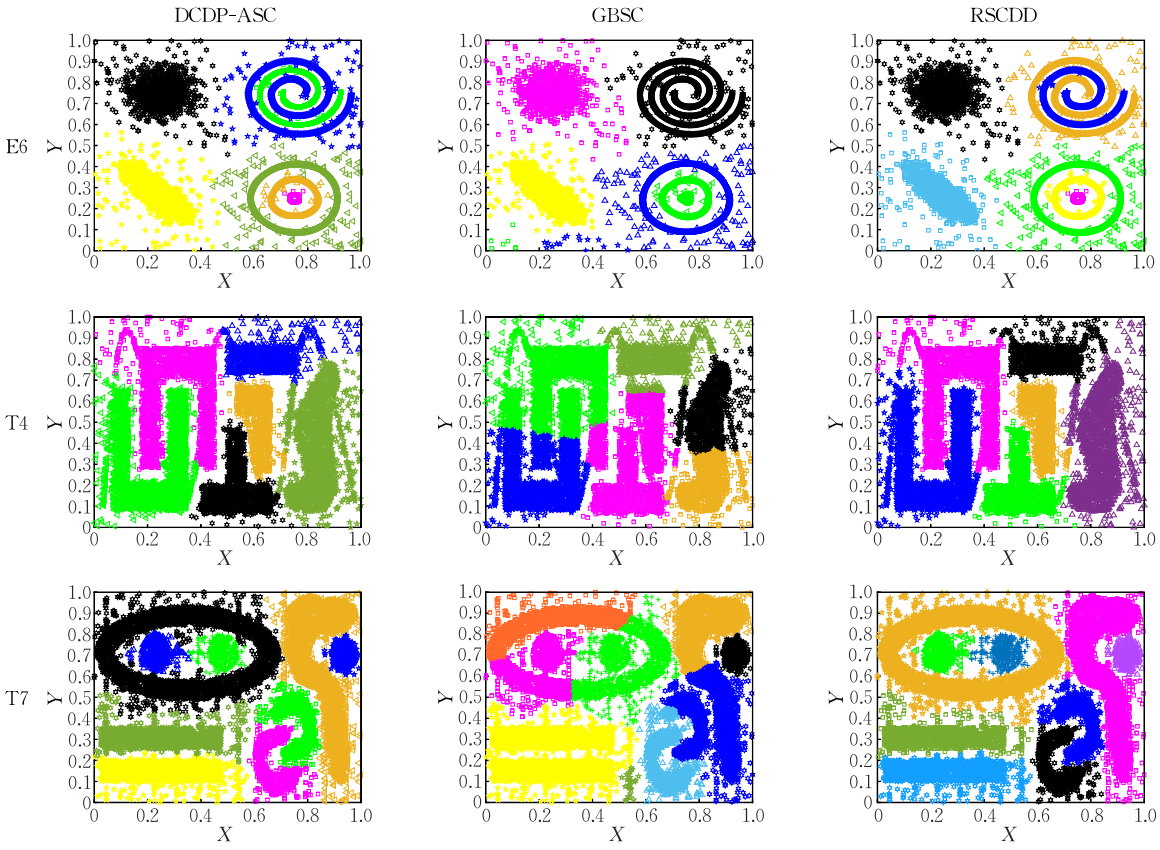


图 9 合成数据 E6、T4 和 T7 上的聚类结果可视化

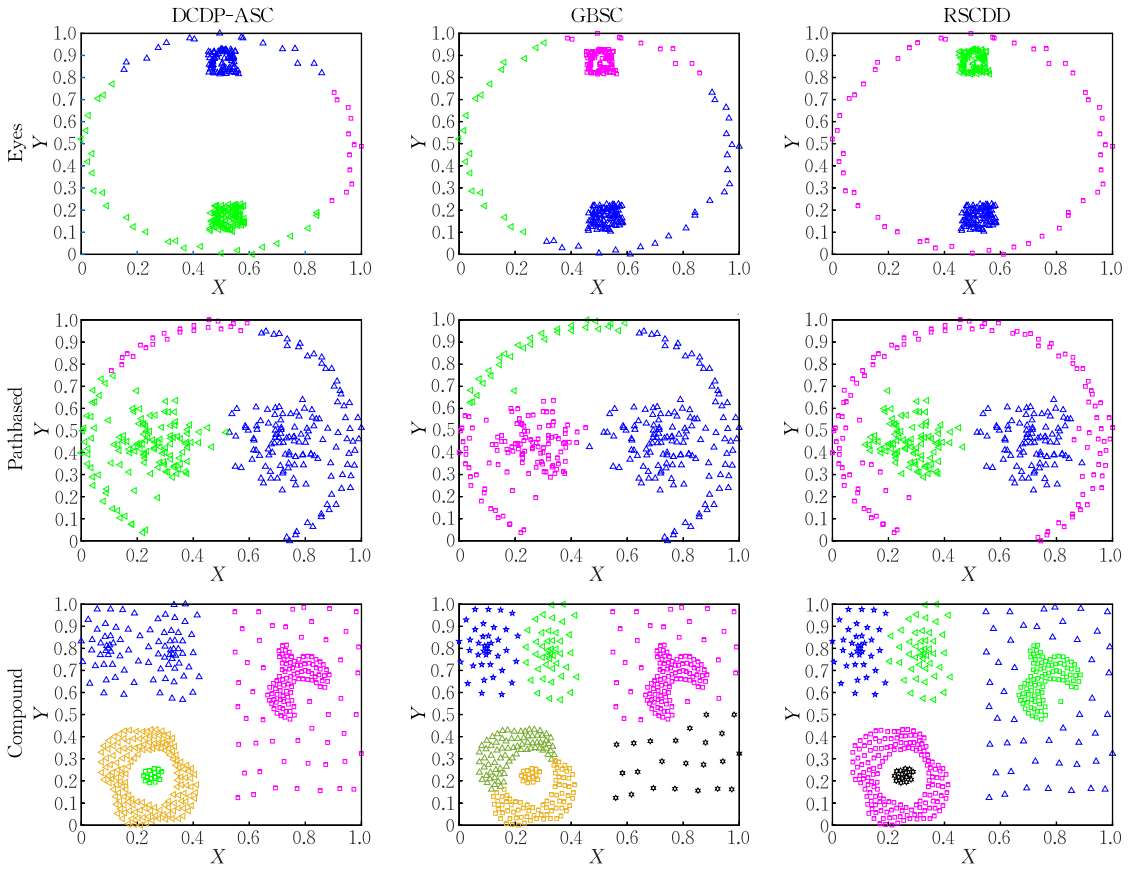


图 10 合成数据 Eyes、Pathbased 和 Compound 上的聚类结果可视化

自然近邻[33]的思想自适应确定 k 近邻以生成近邻子簇.除此之外,DCDP-ASC 设置比例系数 α 用来删除一些低密度点.对于参数 α,在一些低噪声的合成数据上,被设置为 0;在噪声数较多的合成数据集上,使用原文提供的参数.在选择聚类中心时,DCDP-ASC 使用初始子簇的峰值点作为候选点来构建决策图,手动选择局部密度和相对距离较大的峰值点作为 k-means 算法的初始中心.因此,DCDP-ASC 还需要确定真实的子簇数目 *nclust*,这是一个固定值.GBSC 的输入参数包含一个可调整的高斯核带宽 σ,以及真实的子簇数 *nclust*.根据原文提供的参数范围,在 0.05 和区间[0.1,1]内选择最优的 σ,区间步长取 0.1.表 1 给出合成数据集上,DCDP-ASC、GBSC 和 RSCDD 的具体参数设置.子簇数目 *nclust* 为共有参数且为固定值,不再显示.RSCDD 的两个参数表示为 k/μ.

表 1　合成数据集上的参数设置

数据集	DCDP-ASC	GBSC	RSCDD
D6	0.055	0.05	20/2
D7	0.03	0.05	20/3
E6	0.1	0.05	20/1
T4	0.18	0.05	20/1
T7	0.18	0.05	20/1
Eyes	0.1	0.05	20/3
Pathbased	0	0.1	7/3
Compound	0	0.1	7/3

在图 8 中,选取了两个较为简单的合成数据集,它们都是由流形子簇和凸形子簇组成.相较于数据 D7,D6 中包含更多的噪声点,但噪声数量有限,对聚类结果的影响较小.DCDP-ASC、GBSC 和 RSCDD 三个算法都能识别不同形状的子簇,对噪声的分配较为合理.DCDP-ASC 和 RSCDD 在参数设置上过滤掉更多 D6 中的噪声点.

在图 9 中,三个具有明显噪声的合成数据集被用来测试 RSCDD 的有效性.从图 9 中的比较结果来看,DCDP-ASC 和 RSCDD 在处理噪声数据集时具有更好的鲁棒性,GBSC 因为受到噪声的影响而无法正确识别三个数据集中的子簇.在 E6 上,GBSC 将右上方的两个螺旋形子簇识别为一个,并且将右下方嵌套的内部环形子簇划分在一起.在 T4 和 T7 上,GBSC 对数据集的划分都存在偏差.DCDP-ASC 将 E6、T4 和 T7 上的比例系数依次设为 0.1、0.18 和 0.18.RSCDD 在识别噪声时,三个数据集上的噪声系数 μ 都为 1.因此,式(4)能够帮助 RSCDD 有效识别数据集中的噪声,增强了 RSCDD 的鲁棒性.

最后,图 10 中三个常用的密度分布不均匀的合成数据进一步验证了 RSCDD 相较于 DCDP-ASC 和 GBSC 所具有的优越性.在 Eyes、Pathbased 和 Compound 数据上,DCDP-ASC 和 GBSC 因为没有考虑到不同密度水平下簇间的相似性而导致有限的聚类结果.RSCDD 在计算子簇间的相似性时,更注重不同密度水平下簇间的密度分布特性,使用密度均值和密度标准差来衡量两个子簇.RSCDD 的子簇相似性能够使谱聚类关注更多的密度信息,进一步提升谱聚类在密度不平衡数据集上的聚类效果.在 Compound 上,DCDP-ASC 构建的决策图只有四个明显突出的峰值点,而其他的局部峰值点都位于坐标轴上.因此,DCDP-ASC 在 Compound 上生成的最终子簇数目要少于 GBSC 和 RSCDD.在面对复杂数据时,手动选择初始中心是 DCDP-ASC 的一个不足之处,需要被进一步的改进.

4.2　真实数据上的比较

除合成数据外,一些真实的数据也被用来验证算法的有效性,它们的详细信息如表 2 所示.

表 2　真实数据集

数据集	容量	维度	类别
iris	150	4	3
wpbc	198	33	2
sonar	208	60	2
balance	625	4	3
Segmentation	2310	19	7
abalone	4177	8	3
USPS	11000	256	10
drybean	13611	16	7
Gamma	19020	10	2
MNIST	70000	784	10

除真实数据集外,RSCDD 还与 DPC[21]、MDPC+[34]、谱聚类(Spectral Clustering, SC)[35]、USPEC[18]、USENC[18]、DenSC[19]进行比较,DCDP- ASC 和 GBSC 仍作为对照算法.DPC 需要人为设置截断距离 d_c,通过对距离矩阵中的元素以升序排列后,d_c 应选取前 1%～2%位置上对应的最优值.为了扩展 DPC 在复杂数据上的适应性,DPC 的取值范围增加至[0.1%,5%],步长为 0.1%.MDPC+用于改进 DPC 在多峰值簇上的不足,提高流形子簇内样本的紧密程度,使它们更容易被聚在一起.MDPC+提供了默认的近邻值 $k=round(\sqrt{n}\,)$.本文额外地选取了区间[5,30]内的最优结果,与 MDPC+的默认值下的结果进行比较,选取最优的 k.SC 需要带宽 σ 确定样本间的相似性.本文采用 MATLAB 中内置的 SC,参数为默认值.USPEC、USENC 和 DenSC 都是 2.1 节中的基于锚点改进的谱聚类算法.虽然它们

主要用于提升 SC 的运行效率,但在聚类效果上也有所提升. USENC 是多个 USPEC 的集成. USPEC 需要确定锚点数 p 和锚点的近邻数 k,并在 USENC 中增加了集成规模 m. 文献[18]对这些参数进行了详细地分析,其中 p 和 m 对时间的影响明显高于 k. 为了兼顾 USPEC 和 USENC 的效率和精度,本文仅对参数 k 进行调整以确定最优聚类结果. DenSC 主要针对 USPEC 中的锚点选择进行改进,并在聚类结果上有所提升. 同样的,仅对 DenSC 的参数 k 进行调整,以保持同类算法间的一致性. 由于 USPEC、USENC 和 DenSC 的随机性,本文将运行 20 次结果的平均值作为最终的结果. 根据原文的参数分析,DCDP- ASC 在无法获知噪声情况的真实数据上将比例系数 α 设为 0. 同时,为了便于在决策图中选取中心点,部分真实数据上的 α 设为 0.1. GBSC 的参数设置与 4.1 小节的内容一致. 表 3 总结了对照算法在真实数据集上具体的参数设置. 由于 SC 的参数无需手动设置,因此不再统计. 子簇数 *nclust* 是固定值,无需调整.

表 4～表 6 分别展示了 9 个对照算法在 10 个真实数据集上的 *ACC*、*ARI* 和 *AMI*,其中同一数据集上的最优结果以粗体表示. 在 MNIST 数据集上,DPC、SC 和 GBSC 需要计算距离矩阵,因此会造成内存不足,无聚类结果输出. MDPC＋、DCDP-ASC 和 RSCDD 中的 kd-tree 适用于低维数据集,而 USPEC、USENC 和 DenSC 在处理大规模高维数据集时效率更高. 本文采用 t-SNE 降维弥补 kd-tree 在高维数据上的不足.

表 3 真实数据集上的参数设置

数据集	DPC(d_c)	MDPC＋(k)	USPEC(k)	USENC(k)	DenSC(k)	DCDP-ASC(α)	GBSC(σ)	RSCDD(k/μ)
iris	0.2	7	5	5	15	0	0.1	12/3
wpbc	0.1	7	5	12	7	0	0.6	6/1
sonar	0.5	14	5	5	7	0	0.5	5/3
balance	3	29	14	25	29	0.1	0.2	7/1
Segmentation	1	48	5	5	5	0	0.1	82/1
abalone	2	65	19	20	30	0	0.1	17/3
USPS	0.01	9	5	5	2	0.1	0.2	9/3
drybean	1	117	30	5	50	0.1	0.1	32/1
Gamma	1	10	5	5	2	0	0.1	8/3
MNIST	N/A	265	3	5	2	0.1	N/A	52/3

表 4 真实数据集上对照算法的 *ACC*

数据集	DPC	MDPC＋	SC	USPEC	USENC	DenSC	DCDP-ASC	GBSC	RSCDD
iris	0.9400	0.9660	0.8933	0.9467	0.7937	0.9163	**0.9733**	0.8467	0.9667
wpbc	0.7626	0.7677	0.7626	0.7626	0.7626	0.7626	0.7677	0.7626	**0.7727**
sonar	0.5769	**0.6154**	0.5289	0.5315	0.5298	0.5428	0.5433	0.5577	**0.6154**
balance	0.6848	0.4656	0.5872	0.6771	0.6572	0.6853	0.5264	0.6222	**0.7120**
Segmentation	0.6442	**0.7311**	0.5784	0.2846	0.7092	0.3155	0.6303	0.5987	0.7204
abalone	0.5150	0.5272	0.5308	0.5147	0.4068	0.5125	0.5150	0.5312	**0.5358**
USPS	0.2971	0.3970	0.1072	0.6443	**0.7940**	0.6555	0.3517	0.3832	0.6787
drybean	0.5913	0.7284	0.7905	0.6017	0.6131	0.5637	0.5144	0.8102	**0.8734**
Gamma	0.6484	0.6892	0.6484	0.6489	0.6637	0.6618	0.6484	0.7223	**0.7241**
MNIST	N/A	0.8128	N/A	0.7624	0.8081	0.7944	0.8001	N/A	**0.8135**
Average	0.5660	0.6700	0.5427	0.6374	0.6738	0.6410	0.6271	0.5835	**0.7413**

表 5 真实数据集上对照算法的 *ARI*

数据集	DPC	MDPC＋	SC	USPEC	USENC	DenSC	DCDP-ASC	GBSC	RSCDD
iris	0.8343	0.9009	0.7312	0.8512	0.6222	0.7800	**0.9222**	0.6498	0.9038
wpbc	0.0036	0.1285	0.0132	0.0335	0.0282	0.0314	0.0222	0.0666	**0.1464**
sonar	0.0190	0.0487	−0.0035	−0.0011	−0.0019	0.0034	0.0041	0.0086	**0.0496**
balance	0.1720	0.0266	0.0622	0.1679	0.1426	0.1892	0.0155	0.0986	**0.1937**
Segmentation	0.4081	0.5831	0.4040	0.0977	0.5685	0.1278	0.4544	0.4762	**0.5889**
abalone	0.1349	0.1512	0.1624	0.1315	0.0146	0.1302	0.1364	0.1638	**0.2019**
USPS	0.1112	0.2667	0.0001	0.5028	**0.6722**	0.5111	0.1757	0.1872	0.5551
drybean	0.4149	0.5564	0.6104	0.4037	0.3611	0.3660	0.2931	0.6692	**0.7075**
Gamma	−0.0009	0.0698	0.0383	0.0034	0.0772	0.0242	0.0000	**0.1919**	0.1479
MNIST	N/A	0.7323	N/A	0.6060	0.6860	0.6573	0.7109	N/A	**0.7352**
Average	0.2097	0.3464	0.2018	0.2797	0.3171	0.2821	0.2735	0.2512	**0.4230**

表 6　真实数据集上对照算法的 *AMI*

数据集	DPC	MDPC+	SC	USPEC	USENC	DenSC	DCDP-ASC	GBSC	RSCDD
iris	0.8222	0.8815	0.7672	0.8429	0.7113	0.8006	**0.8999**	0.7101	0.8836
wpbc	−0.0060	0.0427	0.0225	0.0303	0.0276	0.0327	0.0158	0.0231	**0.0534**
sonar	0.0161	0.0429	0.0393	0.0677	0.0322	0.0167	0.0207	0.0046	**0.1096**
balance	0.1288	0.0451	0.0439	0.1511	0.1244	0.1689	0.0430	0.0874	**0.2667**
Segmentation	0.5870	**0.7142**	0.6563	0.3280	0.6898	0.3651	0.5954	0.6673	0.6969
abalone	0.1482	0.1589	0.1689	0.1463	0.0338	0.1280	0.1320	0.1629	**0.1934**
USPS	0.2690	0.5224	0.0091	0.6477	**0.7533**	0.6548	0.3523	0.3373	0.7040
drybean	0.5147	0.7057	0.7305	0.5116	0.5080	0.5077	0.4826	0.7293	**0.7644**
Gamma	0.0000	0.0918	0.0131	0.0023	0.0317	0.0338	0.0000	0.1181	**0.1692**
MNIST	N/A	0.8354	N/A	0.6961	0.7609	0.7392	0.8114	N/A	**0.8367**
Average	0.2480	0.4041	0.2451	0.3424	0.3673	0.3448	0.3353	0.2840	**0.4678**

通过比较结果可知,RSCDD 要明显优于其他对照算法.在 7 个真实数据集上,RSCDD 的 *ACC*、*ARI* 和 *AMI* 都要高于其他算法.在 iris 上,RSCDD 的结果仅低于 DCDP-ASC;在 Segmentation 上,RSCDD 的结果仅低于 MDPC+;在 USPS 上,RSCDD 的结果仅低于 USENC.在所有数据集上的平均指标也证明了 RSCDD 的聚类结果要好于其他算法,如在 *ACC* 上,RSCDD 的平均 *ACC* 相对于 DPC、MDPC+、SC、USPEC、USENC、DenSC、DCDP-ASC 和 GBSC 分别提升了 30.97%、10.64%、36.59%、16.30%、10.02%、15.65%、18.21%和 27.04%.在平均 *ARI* 上,RSCDD 的值为 0.4230,除 MDPC+的值为 0.3464 以及 USENC 为 0.3171,其他算法的值都在 0.3 以下.在平均 *AMI* 上,只有 RSCDD 和 MDPC+的值超过了 0.4.

通过对表 4～表 6 中聚类结果的进一步分析可知,RSCDD 存在一些不足之处.RSCDD 的三个指标皆低于 DCDP-ASC(iris 数据上),低于 USENC(USPS 数据上).因此,本文在这两个数据集上具体分析 RSCDD 聚类效果较差的原因.

在 iris 上,每 50 个序号相邻样本属于同一簇,因此能够直观地识别出错误分配的样本点.DCDP-ASC 的错误点为$\{x_{71},x_{73},x_{84},x_{107}\}$,RSCDD 的错误点为$\{x_{71},x_{107},x_{120},x_{134},x_{135}\}$.除共同的误分配点 x_{71} 和 x_{107} 外,RSCDD 将更多的第三类中的点$\{x_{120},x_{134},x_{135}\}$聚类到第二类中.鉴于这些点都是重叠度较高的边界点,因此 RSCDD 在处理边界点时,需要更加充分地考虑邻域信息.

在 USPS 上,RSCDD 的聚类结果虽然高于其他算法,但却明显低于 USENC.这是由于 USENC 作为集成的谱聚类算法,能够更加有效地处理高维图像数据集.而 RSCDD 虽然采用子簇相似性作为谱聚类的输入,但在计算近邻点时仍使用欧式距离,不能很好地处理图像数据中的复杂非线性结构.通过 t-SNE 降维,RSCDD 在 USPS 上的 *ACC*、*ARI* 和 *AMI* 分别为 0.8085、0.6775 和 0.7695,而 USENC 的结果分别为 0.7746、0.6416 和 0.7521.可以看出,t-SNE 能够有效增加 RSCDD 在高维图像中的聚类效果.

综上所述,相比于对照算法,RSCDD 在真实数据上具有更优的聚类效果,但在分配边界点以及处理高维非线性数据集时仍存在改进的空间.

4.3　时间的比较

由 3.4 节的时间复杂度可知,RSCDD 的运行时间相对较少.为验证算法的运行效率,RSCDD 与 4.2 节中的对照算法在真实数据集上进行时间对照,记录 10 次运行结果的平均值.4.2 节的对照算法在原算法的运行速率上具有明显的改进效果,因此与这些算法进行对照能充分检验 RSCDD 算法的效率.表 7 给出了不同算法主要时间复杂度的比较,并且为了保持不同算法间的一致性,去除了有关谱聚类算法在数据维度上的分析.n 和 k 分别为样本数和近邻数.p 表示锚点数,m 为基聚类的数目,ω 用来限制锚点生成过程中每次 k-means 划分的最大类数.p、m 和 ω 由人工设置,为较小的常数.由表 7 可知,除 DPC 和 SC 外,其他算法都具有较高的计算效率.因此,通过与对照算法进行比较,能够证明 RSCDD 的高效性.

值得注意的是,所采用的对照算法中,只有 GBSC 使用 Python 进行实现.考虑不同环境对运行时间的影响,本文不再统计 GBSC 的时间.尽管如此,表 8 中的统计结果仍能显示 RSCDD 在缩短运行时间上的优越性.

表 7　对照算法的主要时间复杂度的比较

DPC[21]	USPEC[18]	DCDP-ASC[25]	MDPC+[34]	USENC[18]	GBSC[20]	SC[28]	DenSC[19]	RSCDD
$O(n^2)$	$O(n(p^{1/2}))$,$p\ll n$	$O(n\log(n))$	$O(n\log(n)+kn)$	$O(nm(p^{1/2}))$,m,$p\ll n$	$O(n\log(n))$	$O(n^2+n^3)$	$O(n\omega)$,$\omega\ll n$	$O(2n\log(n))$

表 8 DCDP-ASC、GBSC 和 RSCDD 生成子簇数的比较

算法	iris	wpbc	sonar	balance	Segmentation	abalone	USPS	drybean	Gamma	MNIST
DCDP-ASC	16	5	6	79	36	69	17	252	35	1121
GBSC	35	47	50	141	516	1001	2461	3125	4314	N/A
RSCDD	22	12	32	5	12	56	245	65	302	493

虽然没有 GBSC 与 RSCDD 运行时间的比较，但本文分析了两个算法在生成子簇数目上的比较，如表 8 所示. DCDP-ASC 作为同类型的改进算法，其生成的子簇数目也被统计到表 8 中进行对照. 子簇数目越少，则谱聚类中的时间复杂度越低，算法效率越高. 此外，在 2.2 节，本文提及由 DPC 划分生成的子簇数目明显少于粒球的方法，表 8 的结果能够在实验上对这一结论进行有效验证. 由表 8 可知，RSCDD 和 DCDP-ASC 在所有真实数据集上生成的子簇数目都要明显少于 GBSC 生成的数目. DCDP-ASC 与 RSCDD 各自在一半的数据集上生成更少的簇数，且 RSCDD 在规模更大的数据集上子簇数更少. 图 11 中 DCDP-ASC 和 RSCDD 在真实数据集上的时间比较也说明了 RSCDD 在大规模数据集上效率更高.

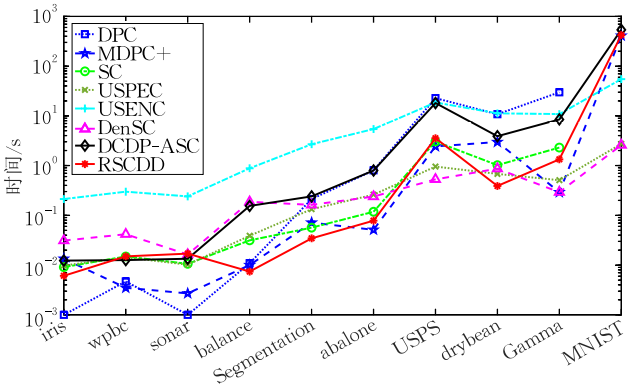


图 11 对照算法在真实数据集上的时间比较

图 11 显示了 8 个对照算法在 10 个真实数据集上运行时间的比较. 从时间的比较可知，当数据维度较低时，RSCDD 的运行效率能保持在一个较高的水平. 在 9 个数据集的运行时间上，除 MNIST，RSCDD 仅在 wpbc、sonar 和 USPS 上表现出不足，而在其他数据集上的效率处于中上水平. 如 balance、Segmentation 和 drybean 上，RSCDD 的运行时间最短，在 iris 和 abalone 上仅比一个算法消耗更多的时间. 因此，可以说 RSCDD 在保持运行效率的同时显著提升了聚类结果. 在 MNIST 上，由于 MDPC+、DCDP-ASC 和 RSCDD 中的 kd-tree 的效率受数据高维特征的影响，它们的运行时间大幅度增加. 尽管 t-SNE 被用来降低数据维度，但数据降维的时间远大于算法在降维后的数据运行的时间，因此 MDPC+、DCDP-ASC 和 RSCDD 在 MNIST 的时间消耗大致相等.

DPC 在 iris、wpbc 和 sonar 上有着不错的表现. 这是因为 DPC 采用了 MATLAB 中快速的距离矩阵计算方法（$pdist2(\cdot)$函数），因此在一些小容量数据集上，DPC 有着更高的效率. SC 在 Segmentation 和 abalone 上的效率也明显高于改进的谱聚类算法. 同样是因为 MATLAB 自带的谱聚类是通过稀疏矩阵加速的改进算法，有着更高的效率.

4.4 参数分析

考虑到参数 k 和 μ 对 RSCDD 结果的影响，本小节对这两个参数进行分析. 参数分析实验采用 9 个真实数据集，分析不同参数下 *ACC* 的变化情况，结果如图 12 所示. 其中 k 值选择为 RSCDD 在每个数据集上最优值附近的区间. 在 balance 上，当 k 值大于 7 时，生成的子簇数目要少于真实簇数，因此 k 的更大取值无意义.

由图 12 可知，RSCDD 只在部分数据集上受参数影响变化不明显，如 wpbc、sonar、USPS 和 Gamma. 因此，可以得出 RSCDD 的聚类精度较为依赖于参数 k 和 μ，而如何在每个数据集上确定调参区间甚至自适应确定参数的取值仍是 RSCDD 的一个待解决的问题. 尽管如此，可以采用以下方法避免参数的过度选取. 由于参数 k 相较于 μ 的取值范围更大，因此进一步分析了参数 k 的选取方法.

为了更加快速地确定 RSCDD 在新的数据集上的最优参数 k，本文建议先以较大的步长（如 $step=5$）确定少量的全局最优值，然后在这些最优值附近进行更加细致的调参，此时步长可缩减至 1. 图 12 中受 k 影响较大的数据有 iris、balance、Segmentation、abalone 和 drybean，但 balance 只在 $k=5,6,7$ 上有结果. 图 13 进一步展示了参数 k 对 iris、Segmentation、abalone 和 drybean 的影响，调整间隔为 5. 在 iris、Segmentation 和 drybean 上，前三个最大的 *ACC* 对应的区间依次为[10，20]、[75，90]和[10，45]，因此最优参数 k 在这些区间附近并通过步长为 1 的间隔进一步调整取得. 但 abalone 的最优参数 k 并不

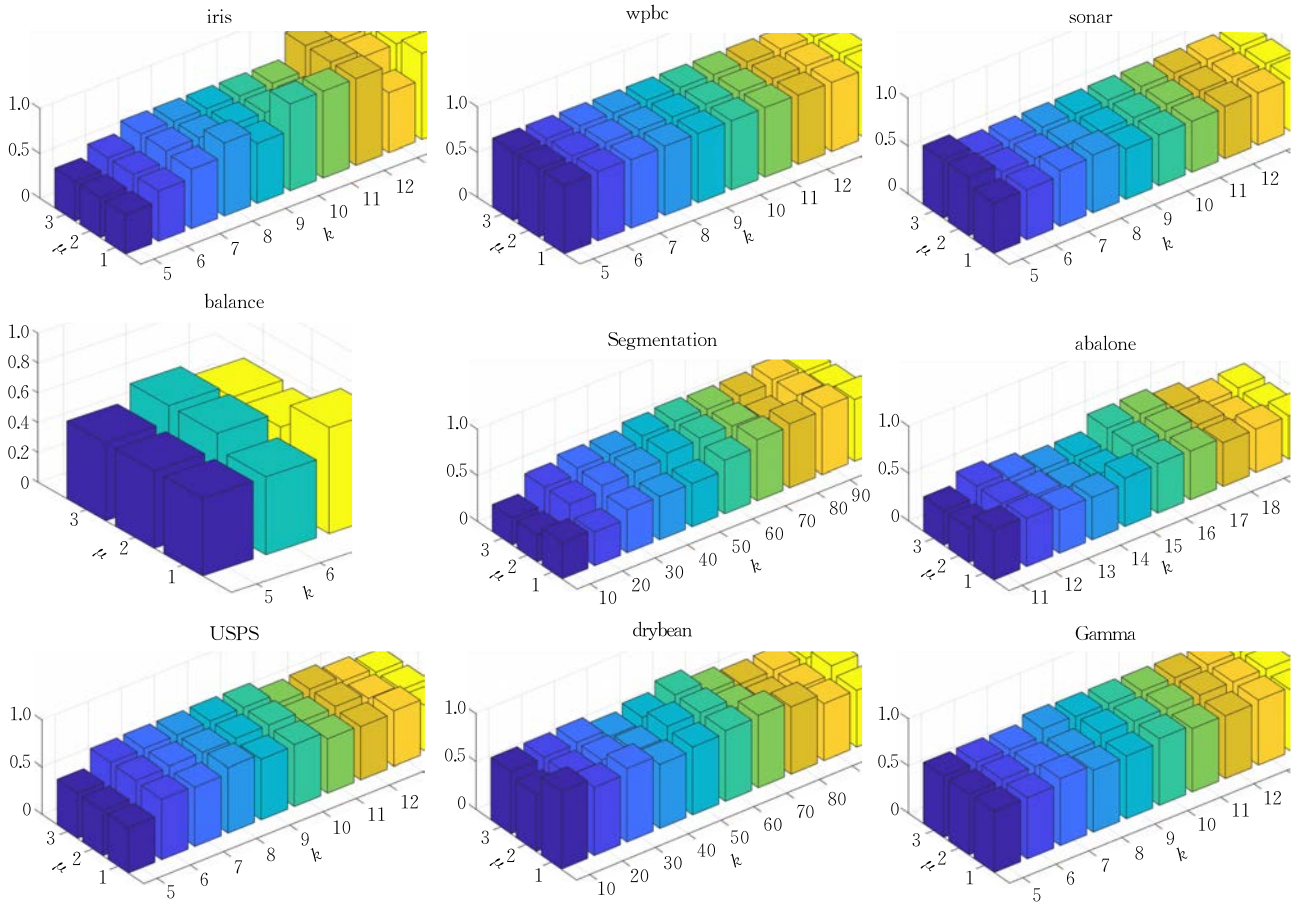


图 12　RSCDD 算法的参数 k 和 μ 分析

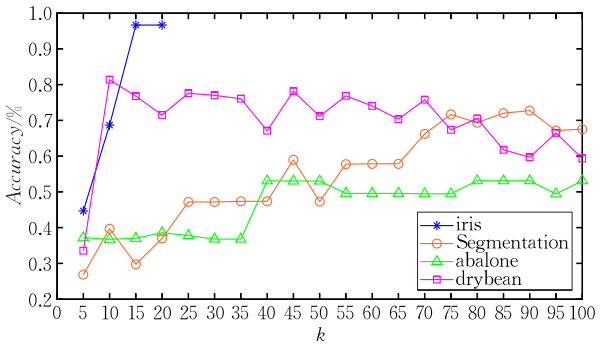


图 13　间隔为 5 时,RSCDD 中参数 k 对敏感数据的影响

符合这一调参方法,因为 abalone 的最优参数 17(表 3)较远地偏离前三个 ACC 所对应的 k 值区间[80,90].因此,本节的参数 k 调试方法在面对复杂数据时并非十分有效,但是相较于间隔为 1 的调参方法能够更加高效.

噪声系数 μ 的确定依赖于数据中的噪声强度,而真实数据中的噪声强度是未知的.基于此,本文分析了在真实数据集下,不同 μ 对 RSCDD 的聚类结果的影响.参数 μ 的区间为[0.1,3],间隔为 0.1,这样能够避免参数的过多选择.图 14 显示了不同粒度间隔(0.2/0.1)下的 μ 的调试结果,纵轴表示 RSCDD 使用相同的 μ 在表 2 的真实数据集上的 ACC、ARI 和 AMI 的总和.

图 14 中左图 μ 的间隔为 0.2.从图 14 的左图可以看到 iris 和 MNIST 的参数敏感性较小,且当 μ 大于 1 时,MNIST 的结果稍微增加.drybean 的前两个最优值对应的 $\mu=1$ 和 1.2,Segmentation 的最优值对应的 $\mu=1.4$.balance 有一个最优区间[0.4,1.4].虽然 Gamma 和 wpbc 的最优值对应的 μ 不在区间[1.2,1.4]内,但它们在该区间内的聚类指标仍然是次优的.由图 14 左图可知,μ 的最优取值应在 1.2 和 1.4 的邻域内查找,因此 $\mu=\{1.1,1.2,1.3,1.4,1.5\}$,如图 14 右图所示.由图 14 的右图可知,$\mu=1.1$ 能进一步确保 RSCDD 在真实数据集上取得最优值.因此,在噪声强度未知的情况下,RSCDD 的参数 μ 默认为 1.1.在查找 μ 的最优取值的过程中,参数间隔先粗后细的调参方式与图 13 中介绍的调参方式一致.μ 默认为 1.1 是根据表 2 中的真实数据集得到的,在面对新的未知标签的真实数据集时,可以将此参数设为默认值.

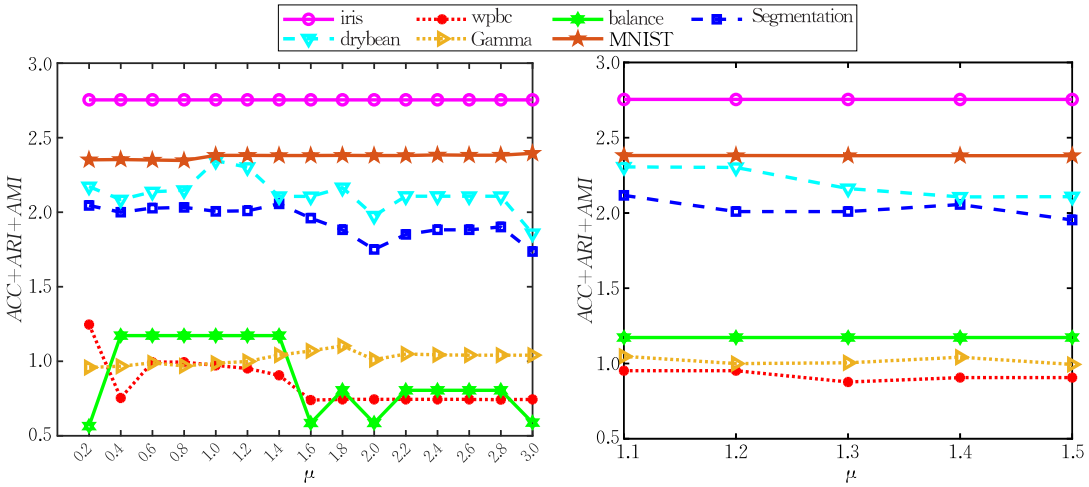


图 14 不同粒度间隔下参数 μ 对 RSCDD 聚类结果的影响

表 9 显示了在默认值 $\mu=1.1$ 下，RSCDD 的聚类结果. 由表 4～表 6 和表 9 可知，μ 固定为 1.1 后，RSCDD 在 iris、sonar、balance 和 USPS 上的结果未发生变换；在 wpbc、abalone、drybean 和 MNIST 上，RSCDD 的聚类指标略微下降，但仍要好于其他对照算法；在 Segmentation 和 Gamma 上(Gamma 的 *AMI* 除外)，RSCDD 的结果有所提升. 总体来说，将 μ 默认为 1.1 既能减少噪声强度未知时参数 μ 调整的困难，又能保证 RSCDD 的聚类效果.

表 9 $\mu=1.1$ 时 RSCDD 的聚类结果

数据集	*ACC*	*ARI*	*AMI*	*k*
iris	0.9667	0.9038	0.8836	12
wpbc	0.7677	0.1377	0.0465	6
sonar	0.6154	0.0496	0.1096	5
balance	0.7120	0.1937	0.2667	7
Segmentation	0.7688	0.6239	0.7243	75
abalone	0.5358	0.1988	0.1909	31
USPS	0.6787	0.5551	0.7040	9
drybean	0.8655	0.6905	0.7502	31
Gamma	0.7275	0.1994	0.1206	7
MNIST	0.8130	0.7325	0.8359	139

4.5 消融实验

在本小节中，通过消融实验对定义 1 中提到的 MKNN 和 KNN 条件做进一步验证，数据集为表 2 中的真实数据. 如图 15 所示，RSCDD-KNN 表示定义 1 的 $j\in knn(i)\wedge i\in knn(j)$ 被替换为 $j\in knn(i)$. 图 15 结果表明，RSCDD 在不同数据集上的聚类结果不弱于 RSCDD-KNN，并且在 balance、Segmentation 和 USPS 上的 *ACC*、*ARI* 和 *AMI* 的和明显高于 RSCDD-KNN. 因此，RSCDD 比 RSCDD-KNN 的聚类效果更好，证明了 $i\in knn(j)$ 和 $j\in knn(i)\wedge i\in knn(j)$ 对 RSCDD 的效果提升是有益的.

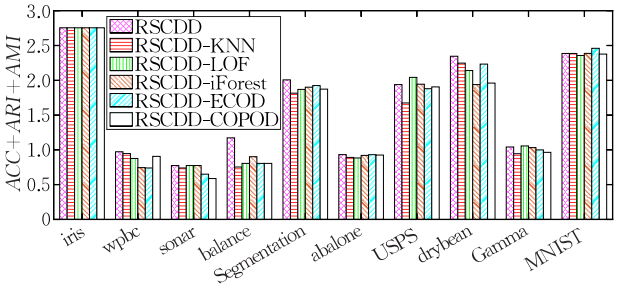


图 15 不同噪声识别方法和 MKNN 条件对 RSCDD 的影响

此外，RSCDD-LOF、RSCDD-iForest、RSCDD-ECOD 和 RSCDD-COPOD 分别表示采用不同的噪声检测算法替换 RSCDD 中的噪声识别方法. 局部离群因子(Local Outlier Factor，LOF)[36] 和孤立森林(isolation Forest，iForest)[37] 是具有代表性的离群检测算法，但它们需要设置不同的参数且不适用于容量大和维度高的数据集. LOF 和 iForest 的参数为给定的默认值. 针对参数设置的问题，ECOD[38] 和 CODOP[39] 是无参的噪声检测算法，不需要有关噪声强度的先验知识. 由图 15 可知，在噪声检测的对照实验中，RSCDD 在 USPS 上弱于 RSCDD-LOF 和 RSCDD-iForest；在 Gamma 上弱于 RSCDD-LOF；在 MNIST 上仅高于 RSCDD-LOF 和 RSCDD-COPOD. 其余数据集上，RSCDD 总能获取最优的比较结果. 在 USPS 和 Gamma 上，RSCDD-LOF 的值分别为 2.0417 和 1.0563. 在这两个数据集上，当 RSCDD 的参数 μ 分别为 0.7 和 1.8 时，*ACC*+*AMI*+*ARI* 的值分别提高到 2.0049 和 1.1035，因此 RCDDD 的聚类结果能够进一步提升.

在噪声检测方法的选择上，RSCDD 需要比较 $\mu=1,2,3$ 下的不同结果，相较于 RSCDD-ECOD 和

RSCDD-COPOD,参数设置更加困难.对此,本文根据前一小节对 μ 的分析,在噪声强度未知的数据上,将 μ 设为 1.1 的固定值,以减少参数调整同时保证算法效果.

综上所述,RSCDD 的聚类效果被充分证明,算法效率相较于改进算法也并未降低,而且分析了噪声强度未知情况下的参数设置问题.尽管如此,RSCDD 仍存在以下问题.首先,受 kd-tree 的影响,RSCDD 在高维数据上的运行时间大幅增加,需要采用一些更加高效的近邻搜索方法,如近似近邻搜索;其次,虽然设置 μ 的默认值为 1.1 能保证 RSCDD 在多数数据上的效果,但自适应的参数设置可能更适用于真实数据集.此外,RSCDD 在高度重叠的边界点上应考虑更多的近邻信息;最后,RSCDD 需要改进在高维非线性数据集上的聚类效果,如采用非线性降维算法.上述不足表示 RSCDD 仍存在改进的空间,需要在接下来的工作中进一步优化.

5　结　论

本文提出一种基于密度分布的鲁棒谱聚类算法,在保证谱聚类运行效率的前提下,提升聚类结果的准确性.该算法首先根据噪声系数过滤低密度的噪声数据,有效避免了噪声数据的影响,提升了算法的鲁棒性.新算法根据互 k 近邻构建待合并的子簇,能够保证划分子簇内样本的一致性.相较于为了实现簇内样本具有相同的标签而过度划分数据集的方法,新算法中的子簇划分在实现更少的子簇数目和更高的簇内一致性上达到了平衡.最后,新算法采用子簇间的密度分布信息计算相似性矩阵,以便于更好地处理密度不均匀的数据集.在实验中,合成数据和真实数据被用来验证改进谱聚类的有效性,证明了新算法在保证效率的前提下,聚类精度有所提升.但新算法的参数对结果影响较大,严重依赖于先验知识和重复的实验.因此,算法的参数优化将是下一步的工作.

参　考　文　献

[1] Chen J, Li Y, Yang X, et al. VGHC: A variable granularity hierarchical clustering for community detection. Granular Computing, 2021, 6(1): 37-46

[2] Pedrycz W. Granular data compression and representation. IEEE Transactions on Fuzzy Systems, 2023, 31(5): 1497-1505

[3] Tsvetkov D, Iliev G. Channel activity analysis of cognitive radio with PCA preprocessing and different clustering methods //Proceedings of the 29th National Conference with International Participation. Sofia, Bulgaria, 2021: 20-23

[4] Liu Y, Li F, Shang J, et al. scFED: Clustering identifying cell types of scRNA-Seq data based on feature engineering denoising. Interdisciplinary Sciences: Computational Life Sciences, 2023, 15(4): 590-601

[5] Cui H, Niu S, Li K, et al. A k-mean++ based user classification method for social e-commerce. Intelligent Automation & Soft Computing, 2021, 28(1): 277-291

[6] Zhang W, Dong C, Yin J, et al. Attentive representation learning with adversarial training for short text clustering. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(11): 5196-5210

[7] Li Y, Lin Y, Hu P, et al. Single-cell RNA-Seq debiased clustering via batch effect disentanglement. IEEE Transactions on Neural Networks and Learning Systems, 2023, DOI = 10.1109/TNNLS.2023.3260003

[8] Ding S, Du W, Li C, et al. Density peaks clustering algorithm based on improved similarity and allocation strategy. International Journal of Machine Learning and Cybernetics, 2023, 14: 1527-1542

[9] Luxburg U. A tutorial on spectral clustering. Statistics and Computing, 2007, 17(4): 395-416

[10] Ding L, Li C, Jin D, et al. Survey of spectral clustering based on graph theory. Pattern Recognition, 2024, 151: 110366

[11] Wang Y, Ding S, Wang L, et al. A manifold p-spectral clustering with sparrow search algorithm. Soft Computing, 2022, 26: 1765-1777

[12] Nie F, Wang X, Huang H. Clustering and projected clustering with adaptive neighbors//Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA, 2014: 977-986

[13] Zhang X, Li J, Yu H. Local density adaptive similarity measurement for spectral clustering. Pattern Recognition Letters, 2011, 32(2): 352-358

[14] Bian Z, Ishibuchi H, Wang S. Joint learning of spectral clustering structure and fuzzy similarity matrix of data. IEEE Transactions on Fuzzy Systems, 2019, 27(1): 31-44

[15] Wang L, Ding S, Wang Y, et al. A robust spectral clustering algorithm based on grid-partition and decision-graph. International Journal of Machine Learning and Cybernetics, 2021, 12: 1243-1254

[16] Yang H, Gao Q, Xia W, et al. Multiview spectral clustering with bipartite graph. IEEE Transactions on Image Processing, 2022, 31: 3591-3605

[17] Song K, Yao X, Nie F, et al. Weighted bilateral k-means algorithm for fast co-clustering and fast spectral clustering. Pattern Recognition, 2021, 109: 107560

[18] Huang D, Wang C, Wu J, et al. Ultra-scalable spectral clustering and ensemble clustering. IEEE Transactions on Knowledge and Data Engineering, 2020, 32(6): 1212-1226

[19] Li H, Ye X, Imakura A, et al. Divide-and-conquer based large-scale spectral clustering. Neurocomputing, 2022, 501: 664-678

[20] Xie J, Kong W, Xia S, et al. An efficient spectral clustering algorithm based on granular-ball. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(9): 9743-9753

[21] Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492-1496

[22] Cheng D, Zhu Q, Huang J, et al. Clustering with local density peaks-based minimum spanning tree. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(2): 374-387

[23] Qiu T, Li Y. Fast LDP-MST: An efficient density-peak-based clustering method for large-size datasets. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(5): 4767-4780

[24] Li C, Ding S, Xu X, et al. Fast density peaks clustering algorithm based on improved mutual *K*-nearest-neighbor and sub-cluster merging. Information Sciences, 2023, 647: 119470

[25] Cheng D, Huang J, Zhang S, et al. A novel approximate spectral clustering algorithm with dense cores and density peaks. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(4): 2348-2360

[26] Mohar B. Some applications of Laplace eigenvalues of graphs. Graph Symmetry, 1997, 497(22): 225-275

[27] Newman M. Finding community structure in networks using the eigenvectors of matrices. Physical Review E, 2006, 74(3): 036104

[28] Cheng D, Liu S, Xia S, et al. Granular-ball computing-based manifold clustering algorithms for ultra-scalable data. Expert Systems with Applications, 2024, 247: 123313

[29] Wu L, Chen P, Yen I, et al. Scalable spectral clustering using random binning features//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. London, UK, 2018: 2506-2515

[30] Lucińska M, Wierzchoń S T. Spectral clustering based on *k*-nearest neighbor graph. Computer Information Systems and Industrial Management, 2012, 7564: 254-265

[31] Yang P, Zhu Q, Huang B. Spectral clustering with density sensitive similarity function. Knowledge-Based Systems, 2011, 24(5): 621-628

[32] Wang G, Song Q. Automatic clustering via outward statistical testing on density metrics. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(8): 1971-1985

[33] Ding S, Du W, Xu X, et al. An improved density peaks clustering algorithm based on natural neighbor with a merging strategy. Information Sciences, 2023, 624: 252-276

[34] Guan J, Li S, He X, et al. Clustering by fast detection of main density peaks within a peak digraph. Information Sciences, 2023, 628: 504-521

[35] Shi J, Malik J. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(8): 888-905

[36] Breunig M, Kriegel H, Ng R, et al. LOF: Identifying density-based local outliers. ACM SIGMOD Record, 2000, 29(2): 93-104

[37] Liu F, Ting K, Zhou Z. Isolation forest//Proceedings of the 2000 Eighth IEEE International Conference on Data Mining. Pisa, Italy, 2008: 413-422

[38] Li Z, Zhao Y, Hu X, et al. ECOD: Unsupervised outlier detection using empirical cumulative distribution functions. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(12): 12181-12193

[39] Li Z, Zhao Y, Botta N, et al. COPOD: Copula-based outlier detection//Proceedings of the IEEE International Conference on Data Mining. 2020: 1118-1123

LI Chao, Ph. D. candidate. His research interests include clustering analysis, data mining.

LIAO Hong-Mei, Ph. D., lecturer. Her research interests include machine learning, pattern recognition.

XU Xiao, Ph. D., lecturer. Her research interests include machine learning, clustering analysis.

GUO Li-Li, Ph. D., lecturer. Her research interests include deep learning, multimodal emotional computing.

DING Shi-Fei, Ph. D., professor, Ph. D. supervisor. His research interests include artificial intelligence, pattern recognition, machine learning, data mining.

Background

Clustering analysis is a data mining and statistical analysis method that aims to group objects in a data set into subsets with similar characteristics, which are called "clusters". Spectral clustering is a clustering algorithm based on graph theory and matrix computation. It implements clustering by eigendecomposition of the similarity matrix of the data or

projecting the data into a low-dimensional space, and then applying traditional clustering algorithms (such as k-means) in the new space. Spectral clustering can deal with non-convex shape clusters and high-dimensional data, and can guarantee optimal clustering results under certain conditions. However, the computational complexity of spectral clustering is high, which may be limited when dealing with large-scale data sets. Spectral clustering is sensitive to noise, because noisy data points may affect the construction of the similarity matrix and the calculation of the feature vector, which leads to the instability and accuracy of the clustering results. In addition, the parameter selection of spectral clustering is also critical, which needs to be adjusted according to the specific problem and data set to obtain the best clustering results.

In order to solve the above problems, many scholars have used the divide and conquer method to greatly compress the scale of the Laplacian matrix in spectral clustering, and the efficiency of the algorithm has been greatly improved. However, these algorithms often have limited improvement in clustering accuracy because they do not fully consider the similarity between coarse-grained objects.

In order to further improve the clustering accuracy of spectral clustering without sacrificing the efficiency of the algorithm, we propose a robust spectral clustering algorithm based on density distribution. Firstly, the algorithm used a density-based clustering algorithm to decompose the whole data set into more compact sub-clusters, which ensured the label consistency of the samples in the cluster and avoided the generation of too many sub-clusters. Then, the dimension of the coarse-grained objects is reduced by measuring the similarity of the sample density distribution between the sub-clusters, so as to divide the sub-clusters. Similarity based on density distribution is more effective in datasets with varying densities. Finally, an effective noise identification method was designed through the probability of the variable in the Gaussian distribution deviating from the mean, which effectively avoided the interference of noise.