论文

K 近邻优化的密度峰值快速搜索聚类算法

谢娟英^{①*}, 高红超^①, 谢维信^②^① 陕西师范大学计算机科学学院, 西安 710061^② 深圳大学信息工程学院 ATR 国家重点实验室, 深圳 518006

* 通信作者. E-mail: xiejuany@snnu.edu.cn

收稿日期: 2015-08-26; 接受日期: 2015-09-21; 网络出版日期: 2016-01-22

国家自然科学基金项目 (批准号: 31372250)、陕西省科技攻关项目 (批准号: 2013K12-03-24) 和中央高校基本科研业务费专项基金项目 (批准号: GK201503067) 资助

摘要 针对 2014 年 6 月发表在 *Science* 的密度峰值点快速搜索聚类算法的样本局部密度定义和样本分配策略的缺陷, 提出一种基于 K 近邻的快速密度峰值搜索并高效分配样本的聚类算法. 算法利用样本点的 K 近邻信息定义样本局部密度, 搜索和发现样本的密度峰值, 以峰值点样本作为初始类簇中心; 提出两种基于 K 近邻的样本分配策略, 依次分配样本到相应类簇中心, 得到数据集样本的分布模式. 理论分析和在经典人工数据集、UCI 数据集及 Olivetti 人脸数据集的对比实验表明: 提出的基于 K 近邻的密度峰值搜索聚类算法能快速发现任意形状、任意维度和任意规模数据集的类簇中心, 并合理分配样本到相应类簇, 揭示数据集样本的分布模式, 对噪声数据具有非常好的鲁棒性, 聚类结果优于 2014 年 6 月发表在 *Science* 的密度峰值点快速搜索聚类算法, 以及经典聚类算法 AP, DBSCAN 和 K-means. 本文算法是一种非常有效的聚类算法, 可用于发现任意数据集的隐藏模式与规律.

关键词 K 近邻 局部密度 密度峰值 类簇中心 聚类

1 引言

聚类是根据数据对象 (样本) 之间的相似性将数据集样本划分成合理类簇的过程, 聚类结果使得同一类簇内的对象具有较高相似性, 不同类簇间的对象相似性较低^[1~6]. 聚类是最重要的认识和了解世界的方式, 借助聚类可以从数据中发现知识, 揭示隐藏的模式和规律, 广泛应用于科学数据分析和工程系统等领域^[1~6]. 大数据背景下的海量和多样数据的存在, 使得具有自动理解、处理和概括数据的聚类算法研究迫在眉睫^[2,6].

聚类算法包括划分式聚类方法、层次聚类方法、基于密度的聚类方法和基于网格的聚类方法, 以及集成式聚类算法^[1,6]. K-means^[7] 是应用范围最广的划分式聚类算法^[2]. 然而, K-means 算法的聚类结果严重依赖于初始类簇中心, 很难发现非凸形状的簇, 对噪声点和离群点敏感, 且类簇数 K 需要事先设定^[1,2,6,8]. 针对 K-means 的缺陷, 出现了 GKM (Global K-means) 算法^[9] 等诸多改进算法^[2,6,10].

引用格式: 谢娟英, 高红超, 谢维信. K 近邻密度峰值快速搜索聚类. 中国科学: 信息科学, 2016, 46: 258–280, doi: 10.1360/N112015-00135

基于密度的聚类算法 DBSCAN (density based spatial clustering of applications with noise)^[11] 能发现任意形状的簇, 在邻域半径参数 ϵ 和核心对象邻域包含的最少样本数参数 MinPts 设置适当时, 能快速发现含噪声空间中任意形状的类簇^[1,6,8], 但如何设置这两个参数缺乏理论依据. 近邻传播聚类算法 AP (affinity propagation)^[4] 将所有样本看作网络中的一个顶点, 通过反复迭代交换近邻样本间的信息, 寻找最优的类代表点样本集合, 使所有样本与最近类代表点样本的相似度之和最大, 发现数据集样本的类簇分布. AP 算法具有简单、高效的优点, 特别是在类别数目较多情况下, 该算法具有非常好的聚类效果^[12], 但是该算法不能发现任意形状的簇. 基于同步动力学模型 Kuramoto 的层次聚类算法^[8] 无需人工设置任何参数, 可以检测出任意数量、形状和大小的类簇, 但层次聚类算法的错误累积缺陷无法避免.

2014 年 6 月 *Science* 发表了自动确定类簇数和类簇中心的新聚类算法 DPC (clustering by fast search and find of density peaks)^[5], 该算法能快速发现任意形状数据集的密度峰值点 (即类簇中心), 并高效进行样本点分配和离群点剔除, 适用于大规模数据的聚类分析. 然而, DPC 算法存在如下缺陷: 1) 样本密度度量方面, 没有采用统一的密度度量准则, 而是针对数据集规模采用不同的样本密度度量准则, 且在样本规模较小时, 度量样本密度主观选择的截断距离 d_c (cutoff distance) 对聚类结果影响较大; 2) 样本分配策略方面, 采用分配样本到距其最近且密度比它大的样本所在类簇, 该样本分配方法会形成类似“多米诺骨牌效应”, 一旦某一个样本分配错误, 会带来一连串样本分配错误, 造成错误的聚类结果, 降低聚类结果可信度, 特别是类簇间样本有重叠时, 比如测试聚类算法的经典数据集 Iris 采用 DPC 聚类的结果就很不好.

鉴于此, 本文提出一种采用 K 近邻优化样本局部密度的快速搜索样本密度峰值点, 并高效分配样本的聚类算法 KNN-DPC, 以避免 DPC 聚类算法的缺陷. KNN-DPC 的主要创新点包括: 1) 提出借助样本 K 近邻的适用于任意规模数据集的样本局部密度度量新准则, 快速搜索和发现样本密度峰值点; 2) 提出两种基于 K 近邻的样本分配策略, 依次完成对除类簇中心外, 其余样本的分配, 得到数据集样本的合理类簇分布, 避免 DPC 算法在样本分配时的缺陷.

2 DPC 算法及其缺陷分析

快速搜索和发现样本密度峰值的聚类算法 DPC^[5] 能自动发现数据集样本的类簇中心, 实现任意形状数据集样本的高效聚类. 其基本原理是: 理想的类簇中心 (density peaks) 具备两个基本特征: 1) 其局部密度大于围绕它的邻居的局部密度; 2) 不同类簇中心之间的距离相对较远. 为了找到同时满足上述条件的类簇中心, DPC 算法引入了样本 i 的局部密度 ρ_i 和样本 i 到局部密度比它大且距离它最近的样本 j 的距离 δ_i , 其定义如式 (1) 和 (2) 所示:

$$\rho_i = \sum_{j \neq i} \chi(d_{ij} - d_c), \quad (1)$$

其中, d_{ij} 为样本 i, j 间的欧式距离, d_c 为截断距离, 当 $x < 0$ 时, $\chi(x) = 1$, 否则 $\chi(x) = 0$;

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

对于局部密度 ρ_i 最大的样本 i , 其 $\delta_i = \max_j d_{ij}$.

由式 (1) 可知, 文献 [5] 引入的样本局部密度受截断距离 d_c 影响. 文献 [5] 指出在数据集规模 (即包含样本数) 较大时, DPC 算法的聚类结果受截断距离影响较小, 反之则较大. 为了避免截断距离对

样本局部密度乃至聚类结果的影响, DPC 算法对规模较小数据集采用指数核^[13] 计算样本密度, 计算公式如式 (3) 所示:

$$\rho_i = \sum_{j \neq i} \exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right). \quad (3)$$

由式 (2) 样本 i 的距离 δ_i 的定义可见, 若样本 i 的密度是最大局部密度或最大全局密度, 则样本 i 的 δ_i 远大于其最近邻样本的 δ 距离. 因此, 类簇中心往往是 δ 异常大的样本点, 这些样本点的密度 ρ 也相对较高. DPC 算法通过构造样本距离 δ 相对于样本密度 ρ 的决策图 (decision graph), 选择 ρ, δ 都较大的样本为类簇中心. 对于剩余样本 j , DPC 算法将其归入密度比 j 大且距离 j 最近的样本所在类簇, 一步完成对剩余样本 j 的分配. 一步分配策略使得 DPC 算法非常高效. 构造样本距离相对于样本密度的决策图, DPC 算法使得任意维数据集的类簇中心均可展示在 2 维平面, 从而实现对任意维数据的聚类分析.

然而, 当每个类簇的规模较小, 导致“理想”类簇数与数据集规模相当时, 样本的稀疏使得密度峰值点很不清晰, 上述密度估计法寻找密度峰值点将变得非常困难^[5]. 此时, 经常采用 $\gamma(\gamma_i = \rho_i * \delta_i, i = 1, \dots, n)$ 曲线构成的决策图来选择类簇中心 (密度峰值点)^[5].

为了避免噪声点对聚类结果的影响, DPC 算法定义了类簇的边界区域, 一个类簇的边界区域由那些属于该类簇但与其他类簇样本的距离小于截断距离 d_c 的样本构成. 以每一个类簇边界区域中密度最大样本的密度为阈值 ρ_b , 定义该类簇中密度大于 ρ_b 的样本为本类簇的核, 该类簇的其他样本为噪声点.

文献 [5] 使用大量实验证实了 DPC 算法的优良性能, 但是该算法存在如下不足: 首先, 度量样本密度使用了不同准则, 对规模较大数据集采用式 (1) 计算样本局部密度, 对于规模较小数据集采用式 (3) 计算样本局部密度. 而样本规模大小的度量缺少一个客观判断依据, 导致如果样本密度度量方法不同, 聚类结果将差异很大. 其次, 在数据集规模相对较小时, 截断距离 d_c 会影响聚类结果^[5]. 最后, 也是最重要的一点, DPC 算法聚类过程的样本一步分配策略会造成连带错误效应, 一旦一个样本分配错误, 便可能带来一系列样本的类簇分配错误. 图 1(a) 和 (b) 展示了在相同截断距离下 ($d_c = 2$ 时), DPC 算法对来自文献 [14] 的经典数据集 Path-based2 分别采用式 (1) 和式 (3) 的样本局部密度度量方法计算样本局部密度的聚类结果. 图 1(c) 展示了 DPC 算法采用式 (3) 估计样本局部密度, 对 Path-based2 数据集选择不同于图 1(a) 和 (b) 截断距离 (即 $d_c = 1$) 的聚类结果.

从图 1(a) 和 (b) 可以看出: DPC 算法采用式 (3) 的指数核度量样本局部密度的聚类结果要明显优于采用式 (1) 的密度估计度量样本局部密度的聚类结果, 两种样本密度计算策略得到的类簇中心大致相同, 但聚类结果相去甚远. 造成这样聚类结果的原因首先是不同的样本局部密度计算策略导致样本密度不同, 其次是 DPC 算法样本分配策略所隐藏的致命连带错误分配效应, 使得一旦一个样本分配错误, 将可能导致一连串的本样本分配错误.

DPC 算法中, 若样本点 i 的密度比其近邻样本的密度大, 则样本点 i 一定被其近邻样本包围, 那么, 样本点 i 到真实分布位于同一类簇的密度比它大的样本的最近距离 δ_{i,C_i} 将较大, 当 δ_{i,C_i} 大于样本 i 到其他类簇中密度较它大的样本点的距离时就会造成对样本 i 的错误分配. 按照 DPC 算法的样本分配策略, 样本点 i 的近邻将因为样本点 i 的错误分配而被错误分配到相应类簇, 因为 DPC 算法对样本点 i 的近邻的分配是依据样本点 i 的类簇信息. 更进一步, 因为错误分配的样本点 i 的近邻, 将可能影响到其他样本点的类簇分配, 形成类似“多米诺骨牌”效应. 如图 1(a) 所示, 第 55 个样本是样本点集合 $S_i = \{51, \dots, 69\}$ 的局部密度极大值点, 由于被近邻包围, 该点到其真实类簇中密度较大值点的距离大于它到其他类簇密度较大值点 (如图中第 193 个样本点) 的距离, 因此, 第 55 号样本被分

图 1 DPC 算法对 Path-based2 数据集采用不同样本密度度量方法和不同截断距离的聚类结果. (a) 式 (1) ($d_c = 2$); (b) 指数核函数 ($d_c = 2$); (c) 指数核函数 ($d_c = 1$)

Figure 1 The clustering of Path-based2 dataset by DPC in different approaches to estimate the local density of a point and different cut off distance. (a) In equation (1) ($d_c = 2$); (b) in exponential kernel function ($d_c = 2$); (c) in exponential kernel function ($d_c = 1$)

配到与第 193 号样本相同的类簇. 第 55 号样本点的错误分配, 导致样本点集合 S_i 中的样本均被错误分配. 同理, 样本点集合 $S_j = \{229, 230, \dots, 254\}$ 的局部密度最大样本点是第 230 个样本点, DPC 算法将其错误分配到第 55 个样本点所在类簇, 从而导致 S_j 集合的其他样本被错误分派. DPC 算法的这种样本分配的连带错误效应构成其致命弱点. 这种错误不仅源于 DPC 算法的样本分配策略, 也与其样本局部密度估计方法密切相关.

由式 (3) 可知, DPC 算法采用核函数估计的样本局部密度会受截断距离 d_c 影响. 图 1(b) 和 (c) 分别展示了 DPC 算法对 Path-based2 数据集在不同截断距离下采用式 (3) 指数核估计样本局部密度的聚类结果. 其中, 图 1(b) 的截断距离 $d_c = 2$, 图 1(c) 的截断距离 $d_c = 1$. 从图 1(b) 和 (c) 的聚类结果可见, 同样采用式 (3) 的指数核估计样本局部密度, DPC 算法在截断距离 $d_c = 2$ 时, 可以正确识别 Path-based2 数据集的各类簇, 但是当截断距离 $d_c = 1$ 时, DPC 算法不能完全正确识别 Path-based2 数据集的各类簇. 由此可见, DPC 算法即便采用相同的指数核度量样本局部密度, 截断距离 d_c 的取值通过影响样本局部密度影响聚类结果. 因此, 截断距离 d_c 对聚类结果的影响不可忽视. 这进一步说明了 DPC 算法的局限.

为了解决 DPC 算法的上述样本局部密度估计和样本分配策略中的问题, 本文提出了一种基于 K 近邻的密度峰值快速搜索聚类算法, 简称为 KNN-DPC. KNN-DPC 算法引入样本的 K 近邻信息, 将 DPC 算法度量样本局部密度的两种准则进行统一, 提出新的适应于不同规模数据集的统一的没有截

断距离影响的样本局部密度度量准则; 并将样本 K 近邻信息引入, 提出两种新的样本分配策略, 依次完成样本分配, 避免了 DPC 算法样本分配策略的潜在缺陷. KNN-DPC 算法的没有截断距离影响的统一样本局部密度估计准则, 能准确快速发现数据集的类簇中心, 基于 K 近邻的样本分配策略克服了 PDC 算法样本分配策略的连带错误效应, 能发现数据集样本的真实类簇分布.

3 本文 KNN-DPC 算法

本文 KNN-DPC 算法从样本局部密度定义和样本分配策略两方面对原 DPC 算法进行改进, 主要思想是: 基于样本 K 近邻信息定义样本局部密度和对样本进行类簇分配. 本节将从样本局部密度定义、样本分配策略两方面对 KNN-DPC 算法进行描述, 并从时间复杂度和聚类性能两个角度对本文 KNN-DPC 算法进行理论分析.

3.1 算法思想

本文算法采用宽度 $\delta = 1$ 的指数核函数, 根据样本 i 的 K 近邻信息定义样本 i 的局部密度 ρ_i , 寻找样本的密度峰值, 并根据样本 i 的 K 近邻信息分配除类簇中心外的其余样本. 本文采用样本 i 的 K 近邻信息定义样本 i 的局部密度 ρ_i 更能反映数据集样本的真实分布信息. 样本 i 的局部密度 ρ_i 定义如定义 1 所述.

定义1 样本局部密度: 样本 i 的局部密度 ρ_i 定义为式 (4), 其中 $\text{KNN}(i)$ 为样本 i 的 K 个近邻样本构成的集合, d_{ij} 为样本 i, j 间的欧式距离.

$$\rho_i = \sum_{j \in \text{KNN}(i)} \exp(-d_{ij}). \quad (4)$$

根据式 (4) 可知, 样本点 i 到其 K 近邻的距离越小, 其密度值 ρ_i 越大. 式 (4) 与式 (1) 和 (3) 比较可见, 本文 KNN-DPC 算法对样本点 i 的局部密度估计, 将密度计算范围从整个数据集样本缩减为样本 i 的 K 个近邻, 这样得到的样本密度仅与其 K 个近邻样本有关, 更能反映样本点 i 的局部信息, 在不考虑搜索样本 K 个近邻的条件下, 本文 KNN-DPC 算法的样本局部密度计算比 DPC 算法的样本局部密度计算更省时. 但是, 为计算样本点 i 的局部密度, 需要搜索其 K 个近邻, 其时间复杂度是 $O(n)$, 因此, 搜索数据集所有 n 个样本点的 K 近邻的总时间复杂度量级将是 $O(n^2)$.

本文依据式 (4) 计算每一个样本的局部密度, 用式 (2) 计算每个样本的 δ 值, 构造决策图, 以 ρ, δ 值较大样本点为类簇中心; 依次采用两种基于 K 近邻的分配策略分配样本到相应类簇. 为了避免离群点影响, 导致将两个类簇合并为一个类簇的现象出现, 在分配样本到相应类簇前, 先进行离群点剔除. 离群点如定义 2 描述.

定义2 离群点: 为了定义离群点, 先给出每一个样本的 KNN 距离 $k_{\text{dist}}(i)$ 定义^[15] 如式 (5) 所示, 然后定义阈值 threshold 如式 (6) 所示, 样本点 o 是离群点, 当且仅当其 $k_{\text{dist}}(o)$ 大于阈值 threshold. 离群点集合定义为式 (7),

$$k_{\text{dist}}(i) = \max_{j \in \text{KNN}(i)} \{d_{ij}\}, \quad (5)$$

$$\text{threshold} = \frac{1}{N} \sum_{i=1}^N k_{\text{dist}}(i), \quad (6)$$

$$\text{Outlier} = \{o | k_{\text{dist}}(o) > \text{threshold}\}, \quad (7)$$

其中, d_{ij} 为样本 i, j 间的欧式距离, $KNN(i)$ 表示样本点 i 的 K 个近邻构成的集合, N 为数据集规模, 即数据集的全部样本数.

由式 (4) 与 (2) 分别计算样本 i ($i = 1, \dots, N$) 的密度 ρ_i 与其距离 δ_i 值, 构造 δ 相对于 ρ 的决策图, 选择 ρ, δ 值较大的样本作为类簇中心. 然后, 将非离群点样本, 按照样本分配策略 1 分配给相应类簇中心, 这样分配到某一类簇的样本和类簇中心有很紧密的联系, 这些样本构成该类簇的核心样本. 对离群点样本和策略 1 未分配的非离群点样本采用样本分配策略 2 进行类簇分配. 经过样本分配策略 1 和策略 2 分配后, 若还有样本未被分配, 则将其视为噪声点, 根据距离噪声点最近且已被分配的样本的类簇信息对噪声点进行类簇分配. 样本分配策略 1 基于图的连通性, 采用广度优先搜索从寻找各类簇中心的 K 近邻样本开始对样本进行类簇分配, 样本分配策略 2 采用 KNN^[16] 思想进行样本分配. 下面是本文 KNN-DPC 算法的详细步骤描述, 以及分配策略 1 和分配策略 2 的详细描述.

1. 本文 KNN-DPC 算法的详细步骤描述

输入: 数据集 Data, 样本近邻数 K , 每个样本 i 的所属类簇 $C_i = -1$ (代表所有样本都未被分配).

输出: 聚类结果 C .

Step1. 数据预处理: 对缺失值处理和数据归一化;

Step2. 计算样本间 (欧式) 距离, 根据式 (4) 和 (2) 计算每个样本的 ρ, δ 值;

Step3. 从由 ρ 和 δ 构成的决策图中选择出类簇中心集合 CI ;

Step4. 根据式 (5)~(7) 发现离群点样本;

Step5. 对除类簇中心外的非离群点样本执行分配策略 1 进行分配;

Step6. 对于离群点和策略 1 未分配的非离群点样本按照分配策略 2 进行分配;

Step7. 对于第 5, 6 两步没有分配的极少量样本, 即噪声点, 归并其到距离最近的已分配样本所在类簇.

2. 样本分配策略 1: 针对非离群点进行分配

(1) 从类簇中心集合 CI 选出一个未访问样本点 ci , 作为一个新类簇的类簇中心, 标记 ci 为已访问;

(2) 将 ci 点的 K 近邻集合 $KNN(ci)$ 中的样本并入 ci 所在类簇, 初始化队列 V_q , 并将 $KNN(ci)$ 中样本依次入队列 V_q ;

(3) 取队列 V_q 的队头样本 q (即从队列 V_q 删除队头样本 q), 对于集合 $KNN(q)$ 中每个样本 r , 若满足条件: a) 没有被分配, b) 非离群点, c) $d_{q,r} \leq \text{mean}(\{d_{r,j} | j \in KNN(r)\})$, 则将 r 归入 q 所属类簇, 并将样本 r 加入队列 V_q 尾部;

(4) 若队列 V_q 不为空, 则转 (3);

(5) 若 CI 中还有未被访问的样本点, 则转 (1), 否则结束策略 1.

3. 样本分配策略 2: 对离群点和经过策略 1 后未被分配的非离群点进行分配

(1) 对每一个未被分配样本 i , 统计其 K 近邻 $KNN(i)$ 中属于类簇 c ($c = 1, \dots, |CI|$) 的样本数 $N_c(i)$, 得到一个 $1 \times |CI|$ 的向量 $N(i)$, 对所有未分配样本 (设规模为 nr) 构成一个 $nr \times |CI|$ 识别矩阵 S , 其中, $s(i, j) = N_j(i)$, $j = 1, \dots, |CI|$, $i = 1, \dots, nr$;

(2) 从识别矩阵 S 中选择出一 (或多) 个最有可能被正确分配的样本 p , 归入相应类簇. 样本 p 的选择方法是: 由矩阵 S 得到样本 i ($i = 1, \dots, nr$) 所属类簇 k ($k = 1, \dots, |CI|$), 使 $N_k(i) = \max\{N_c(i), c = 1, \dots, |CI|\}$; 每一个样本的 $N_k(i)$ 组成向量 $V_{\max}$; 设 $V_{\max}$ 分量最大值对应的样本为 p , 即 $N_k(p) = \max\{N_k(i), i = 1, \dots, nr\}$. 样本 p 按照以下方式分配:

a) 若 $N_k(p) = K$, 则将识别矩阵 S 中最大值为 K 的所有样本分配到最大值对应的类簇;

b) 若 $0 < N_k(p) < K$, 则从识别矩阵 S 中随机选择一个最大值等于 $N_k(p)$ 的样本, 进行类簇分配, 并记被分配样本为 p (注: 这里只选择一个样本进行分配以保障错分样本的概率最低);

c) 否则结束策略 2;

(3) 更新识别矩阵 S : 对于 $KNN(p)$ 中没有分配的样本 q , 置 $N_k(q) = N_k(q) + 1$, 并将样本 p 在识别矩阵 S 的对应向量 $N(p)$ 置 0 (注: 这里 k 是上步 (2) 给样本 p 所分配的类簇号);

(4) 若无未分配样本, 则结束策略 2, 否则转 (2).

3.2 算法分析

对样本规模为 n 的数据集, DPC 算法存储距离矩阵的空间复杂度为 $O(n^2)$, 也是该算法空间复杂度的主要来源. KNN-DPC 算法比 DPC 算法增加了存储每个样本到其 K 个近邻的距离和分配策略中的识别矩阵, 但增加的空间复杂度不超过 $O(|CI|n)$, 而且表示类簇数的 $|CI|$ 通常较小, 因此, 本文 KNN-DPC 算法的空间复杂度与 DPC 算法的空间复杂度量级相同.

DPC 算法中时间复杂度主要来自 3 部分: a) 计算样本间的距离; b) 计算每个样本的密度 ρ ; c) 计算每个样本的 δ 距离. 各部分的时间复杂度都为 $O(n^2)$, 所以总的时间复杂度为 $O(n^2)$.

本文 KNN-DPC 算法的时间复杂度由以下 4 部分决定: a) 计算样本间的距离, 时间复杂度为 $O(n^2)$; b) 计算每个样本的密度 ρ : 在 KNN-DPC 中为计算样本 i 的密度 ρ_i , 需要寻找样本 i 的 K 个近邻, 时间复杂度为 $O(n)$, 则搜索 n 个样本的 K 近邻的时间复杂度为 $O(n^2)$, 因此, 单考虑本文 KNN-DPC 计算样本 i 的密度 ρ_i 只需要其 K 个近邻, 则本文 KNN-DPC 计算样本 i 的密度 ρ_i 比 DPC 省时, 但因需要搜索样本 i 的 K 个近邻, 则其总时间复杂度量级与 DPC 算法相同, 为 $O(n^2)$; c) 计算每个样本的 δ 距离, 时间复杂度为 $O(n^2)$; d) 分配样本: 时间复杂度为 $O(n^2)$, 其中, 策略 1 类似于图的广度优先搜索遍历, 时间复杂度为 $O(n_1(|CI| + 1))$ (n_1 为非离群点规模), 由式 (5)~(7) 可知, 离群点 o 的 $k_{\text{dist}}(o)$ 值要远大于非离群点的相应值, 本文选用的阈值为均值, 所以非离群点规模 n_1 比较大, 各类簇的核心样本规模 $n'_1 (< n_1)$ 也相对较大. 策略 2 的时间复杂度主要取决于其第 2 步, 时间复杂度为 $O(|CI|n_2 + n_2)(n_2 = n - n'_1)$, 因此策略 2 的总复杂度为 $O(n_2^2(|CI| + 1))$. 整个分配策略的时间复杂度量级为 $O(n^2)$. 因此, 本文 KNN-DPC 算法的时间复杂度量级为 $O(n^2)$, 与 DPC 算法的时间复杂度量级相同.

DPC 算法的核心思想是寻找密度峰值点采用的启发式准则 (决策图), 本文 KNN-DPC 算法保留了 DPC 算法的启发式准则来寻找密度峰值点 (类簇中心). 但本文 KNN-DPC 与 DPC 的不同在于: 本文 KNN-DPC 算法定义了基于样本 K 近邻的新的样本局部密度度量准则, 该密度度量准则适用于任意规模的数据集, 而且能更精确地度量样本局部密度, 克服了 DPC 算法需要人为根据数据集规模选择不同样本密度度量准则的缺陷, 并避免了 DPC 算法的样本局部密度依赖于截断距离的缺陷. DPC 算法指出确定截断距离 d_c 的参考是: 使样本的近邻数大约是整个数据集规模的 1%~2% [5]. 本文 KNN-DPC 算法的 K 值比 DPC 算法的截断距离 d_c 更容易确定, 以便保障计算样本密度时的样本近邻数达到整个数据集规模的 1%~2%. 同时, 因为只涉及样本的 K 近邻, 在不考虑搜索样本 K 近邻时间消耗的条件下, 本文 KNN-DPC 算法的样本密度计算会比原 DPC 算法的样本密度计算更高效.

另外, 获得数据集类簇中心后, 本文 KNN-DPC 算法采用策略 1 分配的样本和类簇中心的联系程度非常紧密, 使得错分样本的概率很低, 这些样本构成了各类簇的核心样本. 对经过策略 1 后未被分配的样本和离群点, 策略 2 根据每个样本的 K 近邻的类簇信息, 将各样本分配到最合适的类簇, 该分配策略保证了正确分配的概率是当前条件下最大的, 并尽可能地利用了由策略 1 所获得的样本类簇信息, 具有很好的鲁棒性.

本文 KNN-DPC 算法引入了样本的 K 近邻信息计算样本局部密度和对样本进行类簇分配, 参数 K 对样本局部密度大小有一定影响, 但类簇中心分布在各类簇的最密集区域, K 值变化对各样本局部密度的相对大小影响较小, 因此对选择类簇中心没有影响; 但是 K 值大小会影响每个样本的近邻数, 影响识别矩阵, 从而会影响本文 KNN-DPC 算法的聚类结果, 实验中 K 的取值一般在 5~7 之间. 尽管如此, 本文 KNN-DPC 算法在实际应用问题中对噪声的鲁棒性还是优于 DPC 算法对噪声的鲁棒性, 这一点可从第 4.3.1 小节的实验结果可见. 分析原因是, K 值大小会影响 KNN-DPC 算法的样本分配策略 2 对极端孤立的离群点的分配结果, 但实际情况中极端孤立的离群点微乎其微, 因此, 本文 KNN-DPC 算法在相同的 K 值下对包含噪声的 Waveform (noise) 数据集的聚类结果优于 DPC 算法通过调整截断距离取得的最优聚类结果.

4 实验结果与分析

实验采用测试聚类算法性能非常经典的人工数据集和真实数据集两大类数据集对本文 KNN-DPC 算法进行测试和评价, 对比算法为 DPC [5], AP [4], DBSCAN [11] 和 K-means [7]. 其中 AP, DBSCAN 算法采用作者提供的源码, K-means 算法则调用 MATLAB 中的库函数, DPC 算法是文献 [5] 作者提供的源代码, 需要说明的是我们对 DPC 算法源码中的矩阵元素运算采用 Matlab 提供的矩阵运算实现, 这样使得 DPC 算法的运行时间大大缩短, 本文 KNN-DPC 算法采用 Matlab R2012a 实现. 所有实验的实验环境均为 Win7 32bit 操作系统, Matlab R2012a 软件, 4 GB 内存, Intel(R) Core(TM)2 Quad CPU Q9500@2.83 GHz.

实验结果评价采用聚类准确率 (Acc), AMI (adjusted mutual information) 和 ARI (adjusted rand index) 3 种度量指标进行评价 [17], 3 种指标的取值上界为 1, 取值越大表示聚类结果越好. 其中, AMI 基于信息论, ARI 基于样本对计数, 分别定义如下.

设 $U = \{U_1, \dots, U_R\}$ 和 $V = \{V_1, \dots, V_C\}$ 分别表示数据集 $S = \{s_1, s_2, \dots, s_N\}$ 的真实划分和聚类结果划分. U 和 V 之间的互信息可以表达为一个 $R \times C$ 的相依表 $M = [n_{ij}]_{j=1, \dots, C}^{i=1, \dots, R}$, 令 $n_{ij} = |U_i \cap V_j|$, 则任选一个样本属于 U_i 的概率 $P(i) = \frac{|U_i|}{N}$, 原始划分的熵 $H(U) = -\sum_{i=1}^R P(i) \log P(i)$. 划分 V 的熵 $H(V) = -\sum_{j=1}^C P'(j) \log P'(j)$, 其中 $P'(j) = \frac{|V_j|}{N}$. 令 $P(i, j) = \frac{|U_i \cap V_j|}{N}$, 则原始划分 U 与聚类结果 V 之间的互信息 $MI(U, V) = \sum_{i=1}^R \sum_{j=1}^C P(i, j) \log \frac{P(i, j)}{P(i)P'(j)}$. 原划分 U 与聚类结果 V 间的期望互信息为

$$E\{MI(U, V)\} = \sum_{i=1}^R \sum_{j=1}^C \sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i, b_j)} \frac{n_{ij}}{N} \log \left(\frac{N \cdot n_{ij}}{a_i b_j} \right) \times \frac{a_i! b_j! (N - a_i)! (N - b_j)!}{N! n_{ij}! (a_i - n_{ij})! (b_j - n_{ij})! (N - a_i - b_j + n_{ij})!},$$

其中, $(a_i + b_j - N)^+ = \max(1, a_i + b_j - N)$, $a_i = \sum_{j=1}^C n_{ij}$, $b_j = \sum_{i=1}^R n_{ij}$, 则 AMI 定义为式 (8).

$$AMI(U, V) = \frac{MI(U, V) - E\{MI(U, V)\}}{\max\{H(U), H(V)\} - E\{MI(U, V)\}}. \quad (8)$$

设 a 表示在 U 和 V 均在同一类簇的样本对数目, b 表示在原划分 U 中在同一类簇, 但聚类结果 V 中不在同一类簇的样本对数目, c 表示在 U 中不在同一类簇, 但在 V 中在同一类簇的样本对数目, d 表示在 U 和 V 均不在同一类簇的样本对数目, 则 ARI 的定义如式 (9) 所示:

$$ARI = \frac{2(ad - bc)}{(a + b)(b + d) + (a + c)(c + d)}. \quad (9)$$

表 1 本文实验用人工数据集和真实数据集

Table 1 The synthetic datasets and real world datasets used in this paper

Synthetic datasets			Real-world datasets		
Dataset	No. records/attributes	No. clusters/source	Dataset	No. records/attributes	No. clusters/source
Path-based1	300/2	3/ [14]	Iris	150/4	3/ [22]
Path-based2	312/2	3/ [14]	Wine	178/13	3/ [22]
Flame	240/2	3/ [18]	WDBC	569/30	2/ [22]
Aggregation	788/2	7/ [19]	Seeds	210/7	3/ [22]
SD (noise)	2000/2	5/ [5]	Segmentation	2310/19	7/ [22]
S2 (noise)	5000/2	15/ [20]	Libras movement	360/91	15/ [22]
DIM512	1024/512	16/ [21]	Ionosphere	351/34	2/ [22]
DIM1024	1024/1024	16/ [21]	Waveform	5000/21	3/ [22]
			Waveform (noise)	5000/40	3/ [22]
			Olivetti face	400/92×112	40/ [23]

对二维人工数据集的聚类结果, 采用不同颜色和形状的图形进行展示. 由于 DPC 和 KNN-DPC 算法的核心思想是发现数据集的类簇中心, 所以图形展示的实验结果中标出了各算法发现的类簇中心, 同时在各算法的评价指标比较中也比较了发现的类簇数及其所在的真实类簇数. 本文 KNN-DPC 算法的样本分配策略 2 中选择样本的随机性会对实验结果有一定影响, 但是影响较小. 尽管如此, 实验结果评价中本文 KNN-DPC 算法的 Acc, AMI 和 ARI 评价指标值是其 20 次重复运行的平均实验结果. K-means 算法因为其聚类结果严重依赖于初始聚类中心, 因此, K-means 的实验结果评价指标值也是其重复运行 20 次的平均实验结果.

实验比较的 5 种算法都有相应的参数, 本文 KNN-DPC 算法需要设定样本的近邻数 K ; DPC 算法首先要选择合适的样本密度度量准则, 实验中对于样本数大于 2000 的数据集采用式 (1) 计算样本密度, 否则使用式 (3) 度量样本密度, 两种准则都需要设定截断距离 d_c , 该距离定义为所有样本间距离升序排序后的第 $Par/100$ 位置处的距离, Par 参数会随数据集的不同而不同, 当数据集规模较大时, DPC 算法对此参数不敏感^[5]. AP 算法需要设定每个样本被选中作为类代表点的偏向参数 p , p 值大小与最终类簇数有一定关系 (p 值越大类簇个数越多, 反之亦然)^[4, 12]. DBSCAN 算法需要设定两个参数: 邻域半径 ε 及邻域内包含的最少样本数 $MinPts$. K-means 算法需要给定类簇数 K 及初始聚类中心. 下面实验中 DBSCAN 算法采用网格搜索方法选择最优参数, 其他算法采用多次实验尝试获得最佳参数. 实验展示和比较的 DPC, AP 和 DBSCAN 的实验结果是其最好的实验结果, 这样比较的目的是为了进一步说明本文 KNN-DPC 算法的良好性能.

4.1 数据集预处理

本文实验用的人工数据集来源于同类研究的参考文献^[5, 14, 18~21], 真实数据集来源于 UCI 机器学习数据库^[22] 和 Olivetti Face 数据库^[23]. 这些数据集是测试聚类算法性能非常流行的数据集, 他们在样本规模、特征数、类簇数方面差别很大. 选择这些数据集的目的是为了更好地比较和验证本文算法可以发现和聚类任意形状类簇, 且不受数据集样本量、特征数和噪声等影响的优良性能. 表 1 详细描述了本文实验所用的数据集, 其中来自文献^[5] 的人工模拟数据集命名为 SD (synthetic dataset).

数据预处理包括处理缺失数据和对数据进行标准化. 对于缺失数据采用均值代替, 数据标准化采

图 2 4 种算法对 Path-based1 数据集的聚类结果. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

Figure 2 The clustering of Path-based1 by four algorithms. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

用式 (10) 所示的最大最小化方法. 经过数据标准化, 不仅消除不同量纲对实验结果的影响, 而且降低算法运行时间开销.

$$x_{i,j} = \frac{x_{i,j} - \min(x_j)}{\max(x_j) - \min(x_j)}, \quad (10)$$

其中, $x_{i,j}$ 表示第 i 个样本的第 j 个特征 (属性) 的取值, $\max(x_j)$, $\min(x_j)$ 分别表示第 j 个特征的最大、最小值.

4.2 人工数据集实验结果分析

本文 KNN-DPC 算法和 DPC, AP, DBSCAN 算法在给定参数后, 都能自动寻找和发现类簇. 本部分采用 6 个二维人工数据集检验 4 种算法发现类簇的能力, 图 2~7 展示了各算法在 6 个二维人工数据集的聚类结果, 其中本文 KNN-DPC 算法的聚类结果是 20 次实验中某 1 次的结果. DBSCAN 算法随机选择一个 ε 邻域内至少有 MinPts 个对象的样本作为初始类簇中心, 所以在图 2~7 中只标示了除 DBSCAN 算法外的 KNN-DPC, DPC 和 AP 算法发现的类簇中心. 表 2 给出了包括 K-means 在内的全部 5 种聚类算法对表 1 所示 8 个人工数据集聚类结果的三种评价指标 Acc, AMI, ARI 的值, 以及 KNN-DPC, DPC 和 AP 发现的类簇中心数 (F) 与这些类簇中心真实分布的类簇数 (P), 以及各算法的参数值 (Par). 其中 SD 数据集因原始数据集无类标信息, 因此对该人工数据集只有聚类结果图示, 没有聚类结果的性能指标比较; 5 种聚类算法对 2 个高维人工数据集的聚类结果相同, 因此表 2 只列出了各算法对 DIM512 的聚类结果指标; 表 2 中的 “—” 表示没有对应值, 加粗加黑的值表示较好的实验结果.

表 2 人工数据集上各聚类算法的 Acc, AMI 和 ARI 聚类评价指标值比较

Table 2 The comparison of clustering algorithms in terms of Acc, AMI and ARI on synthetic datasets

Algorithm	Path-based1					Path-based2					Flame				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
KNN-DPC	0.983	0.924	0.951	3/3	6	1.000	1.000	1.000	3/3	6	0.996	0.967	0.983	2/2	5
DPC	0.733	0.500	0.453	3/3	2	1.000	1.000	1.000	3/3	2	1.000	1.000	1.000	2/2	5
AP	0.740	0.506	0.458	3/3	10	0.343	-0.006	-0.006	3/3	19	0.863	0.452	0.524	2/2	35
DBSCAN	0.647	0.543	0.500	—	0.05/6	1.000	1.000	1.000	—	0.1/5	0.983	0.872	0.934	—	0.09/8
K-means	0.743	0.510	0.461	—	3	0.343	0.006	0.006	—	3	0.848	0.421	0.483	—	2
Algorithm	Aggregation					S2 (noise)					DIM512				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
KNN-DPC	0.999	0.995	0.997	7/7	8	0.958	0.930	0.914	15/15	12	1.000	1.000	1.000	16/16	8
DPC	0.997	0.992	0.996	7/7	4	0.970	0.945	0.937	15/15	2	1.000	1.000	1.000	16/16	2
AP	0.770	0.805	0.713	7/5	7.7	0.908	0.913	0.878	14/14	20	1.000	1.000	1.000	16/16	20
DBSCAN	0.996	0.987	0.992	—	0.05/8	0.724	0.800	0.654	—	0.23/7	1.000	1.000	1.000	—	0.3/7
K-means	0.747	0.789	0.695	—	7	0.872	0.897	0.840	—	15	0.765	0.867	0.751	—	16

从图 2 可以看出, 本文 KNN-DPC 算法能同时发现 Path-based1 数据集的正确类簇中心及其类簇分布. DPC, AP 和 DBSCAN 算法不能对 Path-based1 数据集样本进行正确聚类, 但 DPC, AP 能准确找到该数据集的类簇中心. 图 2(b) 和 (c) 展示的聚类结果显示, AP 和 DPC 算法的聚类结果接近. 但该两算法引起聚类错误的原因不同: DPC 算法的样本分配策略使得一旦密度较大样本点的类簇分配错误, 便可引起该样本近邻一系列样本点的类簇分配错误, 形成类似“多米诺骨牌效应”, 如图 2(b) 显示 DPC 算法将类簇 1 的部分样本错误归并到类簇 2 和类簇 3 中; AP 算法的代表点 C_i 对其附近样本点吸引力较大, 导致将不属于 C_i 的附近样本点归入 C_i , 得到图 2(c) 的聚类结果. DBSCAN 基于图的连通性进行样本分配, 图 2(d) 展示的聚类结果揭示: DBSCAN 错误地将类簇 1 和类簇 3 合并为一个类簇, 将类簇 2 分成两个类簇.

从图 3 所示 4 种算法对 Path-based2 数据集的聚类结果可见, KNN-DPC, DPC 和 DBSCAN 三个算法能实现对 Path-based2 的正确聚类, AP 算法不能实现正确聚类该数据集样本, 因为对于这样的非凸数据集, AP 算法选择的类代表点样本会吸收属于其他类簇的样本. 另外, 图 3 所示的实验结果还显示, KNN-DPC, DPC 和 AP 三个算法均可以发现 Path-based2 数据集的正确类簇中心, 差别仅在于类簇中心的位置不同, 本文 KNN-DPC 算法发现的类簇中心更靠近各类簇的起始中心, DPC 和 AP 算法发现的类簇中心依次逐渐远离各类簇的起始中心. DBSCAN 算法随机选择一个 ε 邻域内至少包含 MinPts 个对象的样本作为类簇中心, 故图 3(d) 没有标出相应类簇中心.

图 4 所示各算法在 Flame 数据集的聚类结果显示: KNN-DPC, DPC 和 DBSCAN 都能对该数据集实现正确聚类, AP 算法不能正确聚类该数据集. 另外, KNN-DPC, DPC 和 AP 均发现了正确的类簇中心, 区别仅在于各算法发现的类簇中心不尽相同.

图 5 展示的聚类结果显示: 除 AP 算法外, 本文 KNN-DPC 算法, 以及 DPC 和 DBSCAN 三种算法都能实现对 Aggregation 数据集样本的正确聚类. 从发现的类簇中心来看, 本文 KNN-DPC 算法和 DPC 算法可以发现该数据集的正确类簇中心, DBSCAN 算法没有标出类簇中心, AP 算法不能完全发现正确类簇中心, 存在从一个类簇选择两个代表点和从多个类簇选择一个代表点的问题, 但所发现

图 3 4 种算法对 Path-based2 数据集的聚类结果. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

Figure 3 The clustering of Path-based2 by four algorithms. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

的正确代表点样本均位于相应类簇的中心.

图 6 的聚类结果揭示, 本文 KNN-DPC 算法和 DPC 算法均能准确发现 SD 数据集的正确类簇中心, 并对样本进行正确分配, 得到相应类簇. AP 算法从密度较大的第 3 个类簇中找到了两个代表点, 据此将该类簇划分成两个类簇. DBSCAN 算法将重叠区域较大的第 1, 2 类簇合并成一个类簇, 并将属于离群点的样本点聚成一个新类簇, 该结果也佐证了 DBSCAN 算法对重叠区域较大的类簇和噪声点敏感.

S2 是含有噪声且部分类簇高度重叠的数据集, 从图 7 展示的 4 种算法对该数据集的聚类结果可见, 本文提出的 KNN-DPC 算法和原始 DPC 算法均能准确发现 S2 的正确类簇中心, 然而 KNN-DPC 算法对第 14 个类簇周围的部分离散点进行了错误分配, DPC 算法的分配结果较为理想, AP 算法能发现 14 个代表点, 其中前 13 个代表点落在了相应类簇的中心附近, 第 14 个代表点位于重叠区域较大的第 2 和第 15 类簇的重叠区域, 因此 AP 算法的聚类结果只有 14 个类簇, 其中第 2 和第 15 类簇被聚成一个类簇; DBSCAN 仅能正确发现 9 个类簇, 第 2, 5, 15 三个类簇被合并为一个类簇, 第 10, 4, 7 三个类簇被聚成一个类簇, 并将离群点样本错误识别为两个新类簇.

从表 2 展示的本文 KNN-DPC 算法, 原始 DPC 算法, AP 算法, DBSCAN 算法和 K-means 算法共 5 种聚类算法对各人工数据集聚类结果的 Acc, AMI 和 ARI 等评价指标值的比较可以看出: 本文 KNN-DPC 和 DPC 算法整体上优于其他算法, 其中本文 KNN-DPC 算法除了在 Flame 和 S2 两个数据集的聚类指标值略低于 DPC 算法, 位居第二外, 在其他数据集的聚类指标值均不低于 DPC 算法, AP, DBSCAN 以及 K-means 算法的相应指标值. 具体情况是: 在 Path-based1 数据集上, 本文 KNN-DPC 算法的各评价指标值明显优于其他算法, 这反应了本文所采用的样本密度计算方法和样本分配策略的

图 4 4 种算法对 Flame 数据集的聚类结果. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

Figure 4 The clustering of Flame dataset by four algorithms. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

优越性; 在 Path-based2 数据集上, 本文 KNN-DPC 算法与 DPC 和 DBSCAN 算法的性能相当, AP 算法的性能最差; 对 Flame 数据集, 本文 KNN-DPC 算法的性能略次于 DPC 算法, 但绝对优于 AP, DBSCAN 和 K-means 算法; 在 Aggregation 数据集上, 本文 KNN-DPC 算法的聚类性能最优, DPC 算法位居第二; 在 S2 数据集上, 本文 KNN-DPC 算法的指标值略低于 DPC 算法, 但是要明显优于其他算法; 对高维人工模拟数据集 DIM512 和 DIM1024, 本文 KNN-DPC 算法的聚类指标值和 DPC, AP, DBSCAN 算法相当, 均能 100% 的正确聚类该数据集样本, 因为各算法对这两个数据集的聚类结果指标值相同, 故在表 2 只展示了本文 KNN-DPC, 以及 DPC, AP 和 DBSCAN 算法对 DIM512 数据集的聚类结果指标值. 另外, 表 2 展示的结果还显示: 在维数较低数据集上, AP 算法和 K-means 算法的 Acc, AMI 和 ARI 聚类结果指标值相近, 且都不适合聚类非凸数据集, 但在高维数据集 DIM512 和 DIM1024 上, AP 算法的性能优于 K-means 算法. DBSCAN 算法在各数据集上也能取得较好的聚类效果, 但对包含噪声点和类簇重叠区域较大数据集的聚类性能较差, 如在 S2 和 Path-based1 数据集的聚类结果.

从表 2 各算法发现类簇中心能力的比较可见, KNN-DPC 和 DPC 算法均能准确发现数据集的类簇中心, AP 算法不能完全搜索到全部正确的类簇中心. AP 算法存在从一个类簇中找到两个代表点 (Aggregation 和 SD 数据集) 或从两个或多个重叠区域较大类簇发现一个代表点 (Aggregation 和 S2 数据集) 的缺陷, 从而导致将类簇强行分开或错误合并的现象. 这从图 5~7 展示的 AP 算法对 Aggregation, SD 和 S2 的聚类结果中可以明显看到.

各算法的参数选择方面, DPC 算法对参数的鲁棒性较强^[5], 参数取值在 2 附近, 第 2 节实验结果证实 DPC 算法的聚类结果受参数 (截断距离) 影响. 本文 KNN-DPC 算法需要设定近邻样本数 K , 从表 2 实验结果可见, 参数 K 的取值在 7 左右, 且对原始样本分布较密集的数据集的参数 K 取值大于

图 5 4 种算法对 Aggregation 数据集的聚类结果. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN
Figure 5 The clustering of Aggregation dataset by four algorithms. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

7. AP 算法的初始偏向参数 p 值对聚类结果影响很大, 最佳 p 值在不同的数据集变化很大. DBSCAN 算法中有两个参数, 且算法对每一个参数都敏感, 所以在各数据集上的最优参数选择难度较大.

4.3 真实数据集实验结果分析

本小节选用 9 个 UCI 数据集^[22] 和 Olivetti 人脸数据库^[23] 来验证本文 KNN-DPC 算法的聚类性能. 选用的数据集从样本规模、特征个数和类簇数目都有较大差别.

4.3.1 UCI 数据集实验结果分析

真实数据集的聚类结果采用 Acc, AMI 和 ARI 三个聚类指标, 以及各算法搜索到的类簇中心数与这些类簇中心实际分布的类簇数之比来评价. 表 3 给出了本文 KNN-DPC 算法, 以及对比算法 DPC, AP, DBSCAN 和 K-means 在 9 个 UCI 数据集上的聚类结果. 除了上述的 4 个评价指标外, 表 3 还列出了实验中各算法的参数取值. 表 3 展示的对比较算法 DPC, AP, DBSCAN 的参数是聚类性能最好时的参数. 表中的“—”表示没有对应值. 加粗加重的值表示较优的实验结果.

表 3 的 Iris 是聚类分析最常用的数据集, 包含 150 个样本, 3 个类簇 (Versicolor, Virginica 和 Setosa), 每个类簇含有 50 个样本, 其中第 2, 3 两个类簇非线性可分, 因此有些算法将该数据集聚成了两个类簇^[12]. 从表 3 实验结果可见: KNN-DPC, DPC 和 AP 算法均能准确发现 Iris 数据集的类簇中心, 但是这三个聚类算法的聚类结果性能指标值差别较大. 本文 KNN-DPC 算法能达到 0.973 的聚类

图 6 4 种算法对 SD 数据集的聚类结果. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

Figure 6 The clustering of SD dataset by four algorithms. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

准确率 (第 3 类簇中的 4 个样本被错误分配到第 2 个类簇), 优于其他对比算法. 目前, 关于 Iris 数据集的最好聚类准确率是采用 Bayesian Ying-Yang (BYY) 模型的模拟退火算法得到的 0.98 (错分 3 个样本)^[24], 但该算法寻找最佳模型参数的时间复杂度较高. 本文 KNN-DPC 算法对 Iris 数据集聚类结果的 AMI 和 ARI 评价指标值也远优于 DPC, AP, DBSCAN 和 K-means 算法.

Wine 数据集包含 178 个样本 3 个类簇, 每一个样本用 13 个特征表示. KNN-DPC, DPC 和 AP 算法均能发现其正确类簇中心. 从聚类结果指标 Acc, AMI 和 ARI 来看, 本文 KNN-DPC 的聚类结果指标值最优, K-means 在该数据集上的平均聚类结果仅次于本文 KNN-DPC 算法, 其次是 DPC 算法, AP 和 DBSCAN 算法对该数据集聚类结果相当.

乳腺癌数据集 WDBC 包含两个类簇 (良性和恶性乳腺癌), 每一个样本用 30 个实值特征描述. 表 3 实验结果显示, 本文 KNN-DPC 算法和 AP 算法能准确发现其类簇中心, 而 DPC 算法发现的两个类簇中心位于同一真实类簇. 从聚类结果指标 Acc, AMI 和 ARI 来看, 本文 KNN-DPC 算法的聚类性能最优, DPC 算法的聚类性能最差. 导致 DPC 聚类结果指标值最差的原因之一是其发现的两个类簇中心位于同一真实类簇. DBSCAN 算法对该数据集的聚类性能优于 DPC 算法. AP 算法和 K-means 算法在确定初始类簇中心后的样本分配过程较类似, 因此该两种算法对 WDBC 的聚类结果相近, 且优于 DBSCAN 算法的聚类结果.

Seeds 数据集包含 3 类小麦种子, 每个样本由种子的 7 个几何参数描述. 表 3 的实验结果显示, 本文 KNN-DPC 算法和 DPC, AP 算法均能准确发现该数据集的相应类簇中心. 聚类结果性能指标 Acc,

图 7 4 种算法对 S2 数据集的聚类结果. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

Figure 7 The clustering of S2 dataset by four algorithms. (a) KNN-DPC; (b) DPC; (c) AP; (d) DBSCAN

AMI 和 ARI 的比较揭示, 本文 KNN-DPC 算法优于 DPC 和 AP 算法, 该三种算法的聚类性能优于 DBSCAN 算法和 K-means 算法. 因此, 对 Seeds 数据集本文 KNN-DPC 算法的聚类性能最佳. 从表 3 实验结果可见, DBSCAN 算法对该数据集的聚类效果最差.

Segmentation 包含 7 类 2310 个样本, 是一个图像分割数据集. 从表 3 实验结果可见, KNN-DPC 和 DPC 算法均能找到 6 个分布在 5 个类簇的类簇中心, AP 算法发现了属于 5 个类簇的 7 个代表点 (类簇中心). 对 Segmentation 数据集, 本文 KNN-DPC 的聚类准确率高于 DPC 算法, 聚类结果的其他两个指标 AMI 和 ARI 低于 DPC 算法, 但该两算法的聚类效果明显优于 AP, DBSCAN 和 K-means 算法. DBSCAN 在该数据集的聚类效果最差.

Libras movement 和 Ionosphere 是两个特征数较多的真实数据集. 其中, Libras movement 包含 15 类手势移动数据, 每类含有 24 个样本. 较少的簇内样本数给 KNN-DPC 和 DPC 算法的密度估计带来一定困难. 本文 KNN-DPC 算法发现 13 个分布在 9 个类簇的类簇中心, AP 算法找到分布在 10 个类簇中的 16 个代表点 (类簇中心), DPC 算法找到 9 个分布在 7 个类簇的类簇中心. 聚类结果性能指标 Acc, AMI 和 ARI 的值显示: 本文 KNN-DPC 算法最优, 其次是 AP 算法, 接着是 K-means 算法和 DPC 算法, DBSCAN 算法对该数据集的聚类效果最差. Ionosphere 是包含两个类簇的雷达数据, 本文 KNN-DPC 算法和 DPC 算法均找到了分布在 2 个类簇的 3 个类簇中心, AP 算法发现了 2 个分布在同一个类簇的类簇中心. 从表 3 的聚类结果评价指标来看, 本文 KNN-DPC 算法的性能最优, 然后依次是 K-means, AP, DPC 和 DBSCAN. 初始类簇中心在同一个类簇的 AP 算法的聚类准确率高

表 3 UCI 数据集上各聚类算法的 Acc, AMI 和 ARI 聚类评价指标值比较

Table 3 The comparison of clustering algorithms in terms of Acc, AMI and ARI on UCI datasets

Algorithm	Iris					Wine					WDBC				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
KNN-DPC	0.973	0.912	0.922	3/3	7	0.948	0.829	0.844	3/3	7	0.943	0.675	0.783	2/2	7
DPC	0.887	0.767	0.720	3/3	2	0.882	0.706	0.672	3/3	2	0.613	0.009	-0.011	2/1	9
AP	0.907	0.756	0.757	3/3	6	0.854	0.686	0.616	3/3	6	0.924	0.602	0.718	2/2	40
DBSCAN	0.893	0.775	0.732	—	0.14/9	0.876	0.678	0.660	—	0.42/10	0.870	0.419	0.542	—	0.27/7
K-means	0.825	0.692	0.660	—	3	0.932	0.815	0.830	—	3	0.928	0.611	0.730	—	2

Algorithm	Seeds					Segmentation					Libras movement				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
KNN-DPC	0.923	0.758	0.788	3/3	8	0.717	0.632	0.539	6/5	7	0.453	0.523	0.331	13/9	9
DPC	0.900	0.717	0.734	3/3	2	0.684	0.651	0.550	6/5	3	0.361	0.390	0.214	9/7	0.5
AP	0.895	0.685	0.715	3/3	10	0.670	0.605	0.502	7/5	25	0.450	0.497	0.277	16/10	2.5
DBSCAN	0.881	0.644	0.686	—	0.17/8	0.441	0.435	0.227	—	0.25/10	0.350	0.408	0.154	—	0.96/5
K-means	0.890	0.671	0.705	—	3	0.602	0.578	0.483	—	6	0.443	0.519	0.304	—	15

Algorithm	Ionosphere					Waveform					Waveform (noise)				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
KNN-DPC	0.729	0.241	0.297	3/2	8	0.696	0.313	0.338	3/3	5	0.654	0.245	0.265	3/3	5
DPC	0.681	0.238	0.276	3/2	0.65	0.586	0.318	0.268	3/3	0.5	0.535	0.184	0.164	3/3	0.3
AP	0.709	0.127	0.173	2/1	15	—	—	—	—	—	—	—	—	—	—
DBSCAN	0.607	0.086	0.036	—	0.2/7	—	—	—	—	—	—	—	—	—	—
K-means	0.712	0.129	0.178	—	2	0.501	0.364	0.254	—	3	0.512	0.364	0.252	—	3

于 DPC 算法的原因是: AP 算法找到的两个代表点 {345,285}虽属于第 1 个类簇, 但该两个代表样本的 K 近邻 ($K=10$) 的真实分布情况是, 第 345 代表点的 K 近邻全部分布在第 1 个类簇中, 而代表点 285 的 K 近邻中有 80% 分布在第 2 个类簇, 因此, AP 算法将代表点 285 作为第 2 个类簇的类簇中心得到的聚类结果较优.

Waveform 和 Waveform (noise) 是两个规模较大的真实数据集, 分别包含 3 类波形. 其中, 后者含有 19 个均值为 0, 方差为 1 的噪声特征. 由于 DBSCAN 和 AP 算法时间复杂度较高, 难以进行参数寻优操作, 因此表 3 中没有该两算法对数据集 Waveform 和 Waveform (noise) 的聚类结果指标值. 从表 3 的实验结果可见, 本文 KNN-DPC 算法和 DPC 算法均能发现该两数据集的正确类簇中心. KNN-DPC 算法对 Waveform 的聚类准确率 Acc 和聚类结果指标值 ARI 优于 DPC 算法, DPC 算法对 Waveform 的聚类结果指标值 AMI 略高于 KNN-DPC 算法的相应值. 对于带有噪声的 Waveform (noise) 数据集, 本文 KNN-DPC 算法的三项聚类指标 Acc, AMI 和 ARI 均优于 DPC 算法的相应指标值. 因此本文 KNN-DPC 算法对噪声的鲁棒性更好. 从表 3 所示 KNN-DPC 和 DPC 算法在 Waveform 和 Waveform (noise) 数据集的参数可见, KNN-DPC 在两数据集的参数 K 取值相同, 均为 5, 而 DPC 算法在该两数据集的截断距离参数取值不同, 分别为 0.5 和 0.3, 由此可见, 本文 KNN-DPC 算法对参数 K 更鲁棒. K-means 算法对该两个数据集都取得了相当不错的聚类效果, 其聚类结果的 AMI 指标甚至达到了 KNN-DPC, DPC 和 K-means 三种算法的最优值. 表 3 的实验结果还揭示, 本文 KNN-DPC 算法和原 DPC 算法对含噪声的 Waveform (noise) 数据集的聚类结果指标值略分别低于其对不含噪声的

图 8 Olivetti 人脸数据集决策图. (a) KNN-DPC($K=4$); (b) DPC($d_c = 0.5$)

Figure 8 The decision graphs of KNN-DPC and DPC on Olivetti face database. (a) KNN-DPC($K=4$); (b) DPC($d_c = 0.5$)

Waveform 数据集的聚类结果指标值, 这说明 KNN-DPC 和 DPC 算法的聚类性能均不同程度地受到了噪声的影响. 尽管如此, 由各算法对 Waveform (noise) 和 Waveform 数据集的聚类结果可见, 本文 KNN-DPC 算法对噪声的鲁棒性, 以及对参数的鲁棒性强于 DPC 算法, 这也说明了本文提出的基于 K 近邻的样本密度定义, 以及基于 K 近邻的样本分配策略具有非常优越的性能. 随机选择初始聚类中心的 K-means 算法, 对该两数据集的聚类结果没有受到噪声影响.

以上对 UCI 真实数据集聚类结果的分析揭示: 本文提出的 KNN-DPC 算法具有非常好的性能, 能发现真实数据集的类簇中心和类簇分布, 具有较强鲁棒性, 聚类性能优于 DPC, AP, DBSCAN 和 K-means 算法.

4.3.2 Olivetti 数据集实验结果分析

Olivetti 人脸数据库^[23]是机器学习领域广泛使用的数据库^[5], 包含 40 个人的脸图像, 每个人的脸图像是 10 幅不同角度的图像. 由于人数 (类簇数) 比每个人的脸图像数 (簇内样本数) 多, 给 DPC 算法的样本密度估计带来很大挑战, 而 KNN-DPC 算法中的密度估计仅取决于每个样本的 K 近邻分布, 所以受其影响较小. 为了降低存储和计算成本, 本文实验首先将每幅图像 (92×112) 缩放到 15×15 的矩阵中, 然后对其采用 PCA (principal component analysis)^[25]主成分分析进行降维. 降维后的图像数据采用欧氏距离度量其间的距离. 图 8 给出了本文 KNN-DPC 和原始 DPC 算法得到的决策图. 图 9 展示了本文 KNN-DPC 和原 DPC 算法对 Olivetti 人脸数据集聚类得到的各类簇的核, 同一类簇用相同颜色表示, 灰色代表相应图像不属于任何类簇, 白色圆点表示了类簇中心. 图 9(a) 展示了 KNN-DPC 算法发现 34 个类簇中心的聚类结果, 图 9(b) 是 DPC 算法发现 20 个类簇中心的聚类结果.

从图 8 实验结果可见, 本文 KNN-DPC 算法得到的决策图可以选择到 34 个或 45 个密度峰值点, 而 DPC 算法只能发现 20 个密度峰值点, 因此 KNN-DPC 算法更容易寻找到类簇中心. 找到密度峰值点后, 对于该数据集, DPC 算法采用更严格的分配策略: 当且仅当样本 i 到距离最近的密度较大点 o 的距离小于截断距离 d_c 时, 才将 i 点归并到 o 点所属类簇^[5]. DPC 算法称被分配的样本点构成类簇的核. 本文 KNN-DPC 算法中经过分配策略 1 分配的样本构成类簇的核.

图9 KNN-DPC 和 DPC 算法对 Olivetti 人脸数据集分别选择 34 个类簇中心和 20 个类簇中心进行聚类得到的各类簇核, 同一类簇用相同颜色表示, 灰色代表不属于任何类簇, 白色圆点代表类簇中心. (a) KNN-DPC; (b) DPC

Figure 9 Pictorial representation of cluster cores for the Olivetti face database. Faces with the same color belong to the same cluster, whereas gray images are not assigned to any cluster. Cluster centers are labeled with white dots. (a) the cores of 34 centers by KNN-DPC; (b) the cores of 20 centers by DPC

从图 9(a) 可见, KNN-DPC 算法发现的 34 个类簇中心分布在 30 个人的人脸图像中, 第 1, 28 和第 40 个人的图像都只有一幅被分配, 且分别被错误分配至第 17 个, 第 37 个和第 5 个人的人脸图像簇中, 即第 1, 第 28 和第 40 个人的人脸图像只有一幅被识别, 且分别被误判为第 17, 第 38 和第 5 个人的人脸图像. 导致该现象的原因是: 我们对于 Olivetti 人脸数据集的预处理方式带来了信息损失; 另外 KNN-DPC 算法的分配策略 1 导致了该现象的发生, 比如 s_1^{10} (第 1 个人的第 10 幅图像) 属于 s_{17}^5 的 $K(=4)$ 近邻, 由于 s_{17}^5 是密度峰值点, 根据分配策略 1, s_1^{10} 被分配到 s_{17}^5 所属类簇中. 从图 9(b) 的实验结果可以看出, DPC 算法得到的 20 个类簇中心分布在 19 个人的人脸图像中, 与 KNN-DPC 算法一样, 受所采用的数据预处理方式影响, 也存在着将不同人误识为同一个人的问题, 如第 5 个人和第 40 个人被误识为同一个人的人脸图像. 另外, 从图 9(a) 和 (b) 可见, 本文的 KNN-DPC 算法和原始 DPC 算法都存在将一个人的 10 幅人脸图像分成两个类簇的情况, 且本文提出的 KNN-DPC 算法由于发现的类簇中心更多, 从而将一个人误判为两个人的现象更加普遍.

为了能更充分说明本文 KNN-DPC 算法的性能, 我们比较 KNN-DPC, DPC, AP, DBSCAN 和 K-means 各算法将 Olivetti 人脸数据集分别聚类得到 40 个类簇时的聚类指标值. 其中, KNN-DPC 和 DPC 算法采用选择前 40 个 γ 值最大的样本点为类簇中心 (密度峰值点); AP, DBSCAN 算法通过调整参数来实现将 Olivetti 人脸数据集聚成 40 个类簇. 表 4 给出了各算法的聚类性能比较, 以及各算法发现的类簇中心与其真实分布的类簇中心之比和各算法的相应参数. 其中, DPC 算法的聚类性能是指经过两步分配得到的聚类结果指标值: 即首先对人脸图像经过较为严格地分配得到相应类簇的核, 然后对经过较为严格分配后未分配的样本, 硬性分配到距离该样本最近且已被分配的样本所属类簇, 得到聚类结果, 计算聚类结果指标值. DPC+ 策略 2 的聚类结果是指: 第 1 步采用 DPC 算法较为严格的样本分配策略, 第 2 步不采用 DPC 的硬性分配策略, 而是采用本文 KNN-DPC 算法的样本分配策略 2 对 DPC 算法第 1 步未分配的样本进行类簇分配所得聚类结果的聚类指标值. 因为 KNN-DPC

表 4 Olivetti 人脸数据库上各算法的 Acc, AMI 和 ARI 聚类评价指标值比较
Table 4 The comparison of clustering algorithms in terms of Acc, AMI and ARI for Olivetti database

Algorithm	KNN-DPC					DPC					DPC + 策略 2				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
	0.803	0.814	0.692	40/36	4	0.665	0.728	0.560	40/32	0.5	0.725	0.782	0.646	40/32	0.5
	0.677	0.732	0.584	34/30	4	0.460	0.587	0.401	20/19	0.5	—	—	—	—	—
	0.768	0.815	0.669	45/37	4	—	—	—	—	—	—	—	—	—	—

Algorithm	AP					DBSCAN					K-means				
	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par	Acc	AMI	ARI	F/P	Par
	0.763	0.737	0.619	40/36	2.1	0.648	0.691	0.526	—	0.6/4	0.681	0.742	0.585	—	40

可以发现的类簇中心为 34 个或 45 个, DPC 算法发现的类簇中心为 20 个, 因此表 4 还展示了本文 KNN-DPC 算法发现 34 个和 45 个类簇中心时的聚类结果指标值, 以及 DPC 算法发现 20 个类簇中心时的聚类结果指标值.

从表 4 的实验结果得知, 在选出的 40 个 γ 值最大的代表点中, KNN-DPC 算法能够发现 36 个人的人脸图像, 而 DPC 算法却只能发现 32 个人的人脸图像, 且 KNN-DPC 算法聚类结果的 Acc, AMI, ARI 指标值远优于 DPC 算法聚类结果的相应指标值. 采用本文提出的样本分配策略 2 分配非核样本的 DPC 算法 (DPC + 策略 2) 能发现 32 个人的人脸图像 (同 DPC), 其聚类结果评价指标值高于 DPC 算法的聚类结果评价指标值. 原因是 DPC 算法采用较为严格的样本分配策略导致了只有一小部分样本成为类簇的核, 比如前 40 个值最大的点作为类簇中心得到类簇核中只包含 182 个样本, 据此对剩余 $400-182=218$ 个样本进行硬性分配, 导致聚类结果与真实情况误差较大. 此时, 若对剩余的 218 个非核样本采用本文提出的基于 KNN 的样本分配策略 2 进行类簇分配, 将能更充分地利用已经分配类簇的 182 个样本的类簇信息, 得到更合理的类簇, 因此 DPC + 策略 2 的聚类结果远优于采用硬性分配非核样本的 DPC 算法的聚类结果. 这也说明了本文提出的样本分配策略 2 的优越性. AP 算法在该数据集上能发现 36 个人的人脸图像, 该算法的聚类结果评价指标值低于 KNN-DPC 算法, 但优于其他所有算法. DBSCAN 算法对该人脸数据库的聚类效果较差, 甚至不如 K-means 算法的聚类结果.

从以上各算法聚类结果的评价指标分析来看, 本文 KNN-DPC 算法最好, 其次是 AP 算法和对非核样本采用本文提出的样本分配策略 2 进行类簇分配的 “DPC + 策略 2” 算法, 接着是 K-means 算法和 DPC 算法, DBSCAN 算法性能最差.

由表 4 展示的本文 KNN-DPC 算法和原始 DPC 算法通过决策图得到的类簇中心 (密度峰值点) 分布来看, 本文 KNN-DPC 算法发现的 34 个类簇中心覆盖了 30 个人的人脸图像, 发现的 45 个类簇中心覆盖了 37 个人脸的人脸图像; DPC 算法发现的 20 个类簇中心覆盖了 19 个人的人脸图像. Olivetti 人脸数据库包含 40 个人的每人 10 幅共 400 幅人脸图像, 本文 KNN-DPC 算法可以识别出 37 个人, 而 DPC 算法只识别了 19 个人, 由此可见: 本文提出的 KNN-DPC 算法具有非常好的发现数据类簇中心的能力. 另外, 表 4 展示的本文 KNN-DPC 算法发现 34 个类簇中心和 45 个类簇中心对应的聚类结果评价指标值和 DPC 算法发现 20 个类簇中心的聚类结果评价指标值比较揭示: 本文 KNN-DPC 算法的聚类性能远优于 DPC 算法, 其中 KNN-DPC 发现 45 个类簇中心的聚类结果指标值 AMI 是 6 种聚类算法中最优的.

4.4 各算法运行时间比较

本小节以 UCI 数据集为例, 比较本文提出的 KNN-DPC 算法与原始 DPC 算法, 以及经典的 AP,

表 5 UCI 数据集上 5 种聚类算法的运行时间

Table 5 The run time of five clustering algorithms on UCI datasets

Dataset	Iris	Wine	WDBC	Seeds	Libras movement	Ionosphere	Segmentation	Waveform	Waveform (noise)
KNN-DPC	0.131	0.142	0.419	0.152	0.295	0.231	2.555	7.747	7.731
DPC	0.049	0.048	0.092	0.05	0.068	0.064	0.806	3.51	3.784
AP	0.565	0.832	6.115	0.973	3.016	2.018	66.79	—	—
DBSCAN	0.059	0.098	0.884	0.122	0.309	0.349	8.727	—	—
K-means	0.014	0.013	0.018	0.014	0.075	0.021	0.062	0.067	0.109

DBSCAN 和 K-means 算法的时间性能. 由 3.2 小节分析可知, DPC 和 KNN-DPC 的时间复杂度量级都是 $O(n^2)$, n 是数据集规模, 即样本数. 但是 DPC 算法的时间消耗主要用于计算样本间距离, 每个样本点的局部密度和其 δ 值. 本文 KNN-DPC 算法与 DPC 算法的主要区别在于, 本文 KNN-DPC 算法需要搜索每个样本的 K 近邻, 以便计算其局部密度, 对除类簇中心外的样本依次采用样本分配策略 1 和样本分配策略 2 进行类簇分配. 因此, 虽然计算样本局部密度只涉及样本的 K 个近邻, 计算时间少于 DPC 算法的样本局部密度计算时间, 但 KNN-DPC 算法增加了寻找每个样本 K 近邻的时间开销, 因此, KNN-DPC 算法计算样本局部密度所需要的总时间复杂度量级与 DPC 算法计算样本局部密度的时间复杂度量级相同, 都是 $O(n^2)$; 另外, KNN-DPC 算法的两步样本分配策略的时间消耗大于 DPC 算法的一步样本分配策略的时间消耗. 因此, 尽管 KNN-DPC 和 DPC 算法的时间复杂度都是 $O(n^2)$ 量级, 但是处理实际数据集的时间消耗是有差别的, 本文 KNN-DPC 算法的实际时间消耗应该大于原始 DPC 算法的时间消耗. 表 5 展示的各算法对 UCI 真实数据集的运行时间比较 (时间单位是秒) 也证实了这点.

从表 5 实验结果来看, 在样本规模较小时, 本文 KNN-DPC 算法的运行时间大约是 DPC 运行时间的 3 倍, 其中相差的部分主要来自样本分配策略 2. 当样本规模较大时, DPC 算法计算样本局部密度的时间消耗增大很快, 而 KNN-DPC 算法的样本分配策略 2 的时间消耗则增大较慢, 导致两种算法的时间消耗差距减少, 体现在表 5 的 KNN-DPC 和 DPC 算法在 2 个 Waveform 数据集的运行时间差距明显减少. AP 算法计算相似度矩阵的时间复杂度为 $O(n^2)$, 然而其迭代次数决定了该算法的时间消耗, 所以 AP 算法的时间消耗很大, 表 5 数据也证实了这点; 另外, 样本规模较大时, AP 算法很难进行参数寻优, 因此表 5 关于 AP 算法对两个 Waveform 数据集的运行时间为 ‘—’. DBSCAN 算法的时间复杂度也是 $O(n^2)$, 表 5 的实验结果显示: 在样本规模较小时, 该算法与 KNN-DPC 和 DPC 算法的时间消耗相差不大, 但随着样本规模的增大, DBSCAN 算法的时间消耗增大较快, 且由于该算法需要预先设定两个参数, 在样本规模较大时很难进行参数寻优操作, 因此表 5 关于该算法对两个 Waveform 数据集的运行时间为 ‘—’. K-means 算法的复杂度为 $O(n)$, 表 5 各数据集的 K-means 算法运行时间也显示出来这点, 这也是 K-means 算法被广泛应用的重要原因.

5 结论

本文提出了一种快速发现类簇中心并合理分配样本的聚类算法 KNN-DPC, 该算法引入样本的 K 近邻信息度量样本密度, 提出两种基于 K 近邻的样本分配策略, 避免了 DPC 算法度量样本密度的缺陷, 及其分配样本时产生的类似“多米诺骨牌效应”的连带样本分配错误. 经典人工数据集, UCI 真实数据集以及 Olivetti 人脸数据集的实验结果表明, 提出的 KNN-DPC 算法能更准确地发现类簇中心,

并分配样本到合适的类簇, 是一种非常有效的自适应聚类算法, 能发现任意形状类簇, 可用于聚类任意维度、任意规模的数据, 对噪声、类簇大小有较强鲁棒性. 然而, 如何自动确定 KNN-DPC 算法的近邻样本数 K 是需要进一步研究的问题.

参考文献

- 1 Han J W, Kamber M. *Data Mining Concepts and Techniques*. 2nd ed. New York: Elsevier Inc, 2006. 383–424
- 2 Jain A K. Data clustering: 50 years beyond K-means. *Pattern Recogn Lett*, 2010, 31: 651–666
- 3 Williamson B, Guyon I. Clustering: science or art?. *J Mach Learn Res*, 2012, 27: 65–80
- 4 Frey B J, Dueck D. Clustering by passing messages between data points. *Science*, 2007, 315: 972–976
- 5 Rodríguez A, Laio A. Clustering by fast search and find of density peaks. *Science*, 2014, 344: 1492–1496
- 6 Xu R, Wunsch D. Survey of clustering algorithms. *IEEE Trans Neural Netw Learn Syst*, 2005, 16: 645–678
- 7 McQueen J. Some methods for classification and analysis of multivariate observations. In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. Los Angeles: University of California, 1967. 281–297
- 8 Huang J B, Kang J M, Qi J J, et al. A hierarchical clustering method based on a dynamic synchronization model. *Sci Sin Inform*, 2013, 43: 599–610 [黄健斌, 康剑梅, 齐俊杰, 等. 一种基于同步动力学模型的层次聚类方法. *中国科学: 信息科学*, 2013, 43: 599–610]
- 9 Likas A, Vlassis N, Verbeek J J. The global K-means clustering algorithm. *Pattern Recogn*, 2003, 36: 451–464
- 10 Xie J Y, Jiang S, Xie W, et al. An efficient global K-means clustering algorithm. *J Comput*, 2011, 6: 271–279
- 11 Ester M, Kriegel H P, Sander J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of ACM SIGKDD'96*, Portland, 1996. 226–231
- 12 Xiao Y, Yu J. Semi-supervised clustering based on affinity propagation algorithm. *J Softw*, 2008, 19: 2803–2813 [肖宇, 于剑. 基于近邻传播算法的半监督聚类. *软件学报*, 2008, 19: 2803–2813]
- 13 Cheng Y. Mean shift, mode seeking, and clustering. *IEEE Trans Pattern Anal Mach Intel*, 1995, 17: 790–799
- 14 Chang H, Yeung D Y. Robust path-based spectral clustering. *Pattern Recog*, 2008, 4: 191–203
- 15 Jin W, Tung A K H, Han J, et al. Ranking outliers using symmetric neighborhood relationship. In: Wee-Keong N, ed. *Advances in Knowledge Discovery and Data Mining*. Berlin: Springer, 2006. 577–593
- 16 Cover T, Hart P. Nearest neighbor pattern classification. *IEEE Trans Inf Theory*, 1967, 13: 21–27
- 17 Vinh N X, Epps J, Bailey J. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: *Proceedings of ICML'09*, Montreal, 2009. 1073–1080
- 18 Fu L, Medico E. FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data. *BMC Bioinformatics*, 2007, 8: 3
- 19 Gionis A, Mannila H, Tsaparas P. Clustering aggregation. *ACM Trans Knowl Discov Data (TKDD)*, 2007, 1: 1–30
- 20 Fränti P, Virtajoki O. Iterative shrinking method for clustering problems. *Pattern Recog*, 2006, 39: 761–775
- 21 Fränti P, Virtajoki O, Hautamaki V. Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans Pattern Anal Mach Intel*, 2006, 28: 1875–1881
- 22 Lichman M. UCI machine learning repository. <http://archive.ics.uci.edu/ml>. Irvine: University of California, 2013
- 23 Samaria F S, Harter A C. Parameterisation of a stochastic model for human face identification. In: *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, 1994. 138–142
- 24 Ma J, Liu J. The BYY annealing learning algorithm for Gaussian mixture with automated model selection. *Pattern Recog*, 2007, 40: 2029–2037
- 25 Person K. On lines and planes of closest fit to systems of points in space. *Edinburgh: Philosophical Magazine*, 1901, 2: 559–572

K-nearest neighbors optimized clustering algorithm by fast search and finding the density peaks of a dataset

Juanying XIE^{1*}, Hongchao GAO¹ & Weixin XIE²

¹ School of Computer Science, Shaanxi Normal University, Xi'an 710062, China;

² School of Information Engineering, National Laboratory of ATR, Shenzhen University, Shenzhen 518006, China

*E-mail: xiejuany@snnu.edu.cn

Abstract In order to overcome the deficiencies of the clustering algorithm published in Science in June 2014 for performing fast searches and finding density peaks, a new K-nearest neighbors-based clustering algorithm is proposed in this paper. The proposed algorithm defines the local density of a point based on its K-nearest neighbors, and thus the density peaks can be found accurately. Then, two new assignment strategies for points are developed based on the K-nearest neighbors theory. The density peaks are regarded as the initial centers for clusters, and the two new assignment strategies are, in turn, used to assign points to the density peaks in order to construct the clusters of a dataset. The theory analyzes the principles of the new algorithm, and thorough experiments on several popular test cases that include both synthetic and real-world datasets from the UCI machine learning repository and Olivetti face database, we demonstrate that the proposed K-nearest neighbors-based clustering algorithm can detect the right cluster centers by searching for density peaks. Furthermore, we show that the proposed algorithm can recognize clusters regardless of their shape and the dimensionality of the space where the clusters are embedded, independently from the size of the dataset. The algorithm is also robust to outliers, and it outperforms the original algorithm published in Science and other popular clustering algorithms, including AP, DBSCAN, and K-means. The new algorithm is a powerful clustering algorithm that can be used to discover the patterns and rules hidden in real-world datasets.

Keywords K-nearest neighbors, local density, density peaks, cluster centers, clustering

Juanying XIE was born in 1971. She received her Ph.D. degree in Signal and Information Processing from Xidian University, Xi'an, in 2012. Currently, she is an associate professor and Master's supervisor at Shaanxi Normal University. Her research interests include machine learning and data mining.

Hongchao GAO was born in 1988. He received his B.S. degree from Henan Polytechnic University, Jiaozuo, in 2011. Currently, he is pursuing his M.S. degree at Shaanxi Normal University. His research area is in data mining.

Weixin XIE was born in 1941. He received his B.S. degree from Xidian University, Xi'an, in 1965. Currently, he is a professor and doctoral supervisor at Shenzhen University. His research interests include intelligent information processing and fuzzy information processing.