软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(11):2760−2772 [doi: 10.3724/SP.J.1001.2011.03903]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

复杂分布数据的二阶段聚类算法*

公茂果[1,2+], 王 爽[1,2], 马 萌[1,2], 曹 宇[1,2], 焦李成[1,2], 马文萍[1,2]

[1](西安电子科技大学 智能感知与图像理解教育部重点实验室,陕西 西安 710071)

[2](西安电子科技大学 智能信息处理研究所,陕西 西安 710071)

Two-Phase Clustering Algorithm for Complex Distributed Data

GONG Mao-Guo[1,2+], WANG Shuang[1,2], MA Meng[1,2], CAO Yu[1,2], JIAO Li-Cheng[1,2], MA Wen-Ping[1,2]

[1](Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education, Xidian University, Xi'an 710071, China)

[2](Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China)

+ Corresponding author: E-mail: gong@ieee.org

Gong MG, Wang S, Ma M, Cao Y, Jiao LC, Ma WP. Two-Phase clustering algorithm for complex distributed data. *Journal of Software*, 2011,22(11):2760−2772. http://www.jos.org.cn/1000-9825/3903.htm

Abstract: In this paper, a Two-Phase Clustering (TPC) for the data sets with complex distribution is proposed. TPC contains two phases. First, the data set is partitioned into some sub-clusters with spherical distribution, and each clustering center represents all the members of its corresponding cluster. Then, by utilizing the outstanding clustering performance of the Manifold Evolutionary Clustering (MEC) for acomplex distributed data, the clustering centers obtained in the first phase are clustered. Finally, based on these two clustering results, the final results are obtained. This algorithm is based on an improved *K*-means, and the MEC. Manifold distance is introduced in evolutionary clustering to make the algorithm competent for the clustering of complex data sets. At the same time, the novel method reduces the computational cost brought by manifold distance. Experimental results on seven artificial data sets and seven UCI data sets with different structure show that the novel algorithm has the ability to identify clusters with simple or complex, convex, or non-convex distribution efficiently, compared with the genetic algorithm-based clustering, the *K*-means algorithm, and the manifold evolutionary clustering. Furthermore, TPC outperforms MEC obviously in terms of computational time.

Key words: data mining; clustering; *K*-means algorithm; evolutionary algorithm; manifold

摘 要: 提出了一种用于复杂分布数据的二阶段聚类算法(two-phase clustering,简称 TPC),TPC 包含两个阶段:首先将数据划分为若干个球形分布的子类,每一个子类用其聚类中心代表该类内的所有样本;然后利用可以处理复杂分布数据的流形进化聚类(manifold evolutionary clustering,简称 MEC)对第 1 阶段得到的聚类中心进行类别划分;最后综合两次聚类结果整理得到最终聚类结果.该算法基于改进的 *K*-均值算法和 MEC 算法.在进化聚类算法的基础上引入流形距离,使得算法能够胜任复杂分布的数据聚类问题.同时,算法降低了引入流形距离所带来的计算量.在

* 基金项目: 国家高技术研究发展计划(863)(2009AA12Z210); 新世纪优秀人才支持计划(NCET-08-0811); 陕西省科技新星支持计划(2010KJXX-03); 中央高校基本科研业务费重点项目(K50510020001)

收稿时间: 2009-07-01; 修改时间: 2010-03-04; 定稿时间: 2010-06-09

分布各异的 7 个人工数据集和 7 个 UCI 数据集测试了二阶段聚类算法,并将其效果与遗传聚类算法、*K* 均值算法和流形进化聚类算法做了比较.实验结果表明,无论对于简单或复杂、凸或非凸的数据,TPC 都表现出良好的聚类性能,并且计算时间与 MEC 相比明显减少.

关键词: 数据挖掘;聚类;*K*-均值算法;进化算法;流形

中图法分类号: TP181 **文献标识码:** A

聚类作为一种重要的数据分析方法已在许多领域得到广泛关注,例如,模式识别、机器学习、数据挖掘和图像处理[1].聚类的任务是在没有训练样本的情况下,仅利用样本间的相似性寻找样本集针对某个评判准则的最佳类别划分.*K*-均值算法[2]是最常用的聚类算法之一.它使用样本与其相应聚类中心的欧式距离之和作为评判准则,通过最小化该评判准则实现对样本集的划分.由于 *K*-均值算法的聚类目标函数是高度非线性和多峰的函数,因此标准的 *K*-均值算法用梯度下降法优化目标函数时,搜索方向总是沿着能量减小的方向,使算法很容易陷入局部极值点,只有当初始化较好时算法才能收敛到全局最优解.作为一类有效的全局优化技术,进化计算已经被很多学者用于聚类问题[3−9].研究人员不仅仅关注于进化操作的创新,对距离测度的研究也是进化聚类研究的热点问题[5,8,9].我们在文献[9]中提出了一种基于流形距离的相似性度量,在处理数据分布复杂的聚类问题时获得了不错的效果.然而,这种基于流形距离的相似性度量是在聚类性能和计算复杂度之间的一个折中.由于要用图论中的最短路径来计算流形距离,因此其计算复杂度要明显高于欧式距离的计算复杂度.

在设计基于进化计算的聚类算法时,最核心的两个问题就是进化个体的编码以及相似性度量.针对聚类问题的个体编码方式有很多,其中使用较多的是借用于 *K*-均值算法的编码方式.即每个个体只对 *K* 个聚类中心进行编码,然后对数据样本按照其与聚类中心的相似性进行类别划分.因此,相似性度量对这类算法的性能有重要影响.最简单的相似性度量应该是欧氏距离,但是以欧氏距离作为相似性度量的进化聚类算法虽然在全局最优化性能上比传统的基于梯度下降的 *K*-均值算法有较大提高,但同样存在一个重要的缺点,它们只对空间分布为球形或超球体的数据具有较好的性能,而对空间分布复杂的数据聚类效果很差,这是基于欧氏距离的相似性度量的缺陷导致的必然结果[10].

本文提出一种能够有效处理复杂分布的数据聚类问题且计算复杂度显著低于 MEC(manifold evolutionary clustering)的聚类算法——二阶段聚类算法(two-phase clustering,简称 TPC).该算法包含两个阶段:首先将数据划分为若干个球形分布的子类,每一子类用其聚类中心代表该类内的所有样本;然后利用可以处理复杂分布数据的 MEC 对第 1 阶段得到的聚类中心进行类别划分;最后,综合两次聚类结果整理得到最终聚类结果.对分布各异的 7 个人工数据集和 7 个 UCI 数据集的实验结果可以说明,无论对于简单或复杂、凸或非凸的数据,TPC 都表现出良好的聚类性能.

本文第 1 节简要分析复杂分布数据聚类存在的问题,并针对这一问题定义基于流形距离的相似性度量.第 2 节将详细介绍本文提出的二阶段聚类算法,分两节详细阐述算法的两个阶段,最后提出二阶段聚类算法. 第 3 节为实验结果比较,我们将分别测试 TPC 在人工数据集和 UCI 数据集上的聚类结果.通过实验分析第 1 阶段采用不同聚类算法对算法性能的影响,并与遗传聚类算法[3]、*K*-均值算法[2]和流形进化聚类算法[9]对这些数据集的聚类结果做比较,同时对各算法鲁棒性进行分析.第 4 节是对工作的总结.

1 复杂分布数据聚类与流形距离测度

聚类算法中使用最为普遍的相似性测度应该是欧式距离.然而现实世界中的聚类问题,数据的分布往往具有欧式距离无法反映的复杂结构.图 1 给出了一个简单的例子,图中的数据呈两个圆环状分布,且两个圆环各为一类.采用以欧式距离为度量的聚类算法处理后得到的效果如图 1 所示.可以看出,以欧式距离作为相似性度量未能达到理想的聚类效果.

由上例可见,基于欧氏距离的相似性度量仅能反映聚类结构的局部一致性特征(即在空间位置上相邻的数据点具有较高的相似性[11]),而无法反映聚类的全局一致性(即位于同一流形上的样本具有较高的相似性[11]).对

于现实世界中的复杂的聚类问题,简单地采用欧氏距离作为相似性度量会严重影响聚类算法的性能.

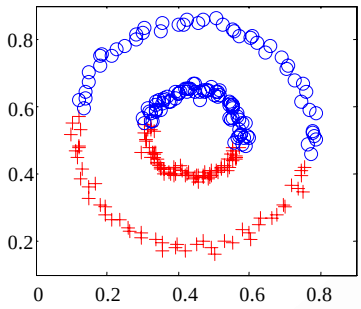


Fig.1 Euclidian distance metric cannot reflect the global consistency of the data set

图 1 欧氏距离无法反映样本的全局一致性

基于以上考虑,我们尝试设计一种能够反映聚类全局一致性的相似性度量,期望新的相似性度量能够打破在欧氏空间"两点之间直线最短"的定理,使得两点间直接相连的路径长度不一定最短.也就是说,新的相似性度量并不一定满足欧氏距离下的三角不等式定理.为了满足聚类的全局一致性,必须使位于同一流形上用较短边相连的路径长度比穿过低密度区域直接相连的两点间距离要短.为了达到这一目的,我们首先定义一个流形上的线段长度.

定义 1(流形上的线段长度). 空间两点 x_i 与 x_j 之间流形上的线段长度 $L(x_i,x_j)$按下式计算:

$$L(x_i,x_j)=\rho^{dist(x_i,x_j)}-1 \tag{1}$$

其中,$dist(x_i,x_j)$为 x_i 与 x_j 之间的欧氏距离,$\rho>1$ 为伸缩因子.

根据流形上的线段长度,我们可以进一步定义一个新的距离测度,称为流形距离.将样本看作是一个加权无向图 $G=(V,E)$的顶点 V,边集合 $E=\{W_{ij}\}$表示的是在每一对样本间定义的流形上的线段长度,则流形距离测度可定义如下.

定义 2(流形距离测度). 将样本看作是图 $G=(V,E)$的顶点,令 $\boldsymbol{p}=\{p_1,p_2,\ldots,p_l\}\in V^l$ 表示图上一条连接点 p_1 与 p_l 的路径.其中,边$(p_k,p_{k+1})\in E$,$1\leqslant k<l-1$.令 $\boldsymbol{P}_{i,j}$ 表示连接样本 x_i 与 x_j 的所有路径的集合,则 x_i 与 x_j 之间的流形距离测度定义为

$$D(x_i,x_j)=\min_{\boldsymbol{p}\in\boldsymbol{P}_{i,j}}\sum_{k=1}^{l-1}L(p_k,p_{k+1}) \tag{2}$$

其中,$L(a,b)$表示求两点间流形上的线段长度.

显然,新的距离测度满足距离测度的 4 个条件,即对称性:$D(x_i,x_j)=D(x_j,x_i)$;非负性:$D(x_i,x_j)\geqslant 0$;三角不等式:对于任意的 x_i,x_j,x_k,$D(x_i,x_j)\leqslant D(x_i,x_k)+D(x_k,x_j)$;自反性:$D(x_i,x_j)=0$,当且仅当 $x_i=x_j$.

流形距离测度可以度量沿着流形上的最短路径,这使得位于同一流形上的两点可以用许多较短的边相连接,而位于不同流形上的两点要用较长的边相连接,从而实现了放大位于不同流形上的样本间的距离,而缩短位于同一流形上的样本间的距离的目的.

2 二阶段聚类算法(TPC)

流形距离测度的引入,使复杂分布数据的结构特点得到较好的体现,对于现实世界中的聚类问题可以达到较好的效果.然而正如前文提到的,要用图论中的最短路径来计算流形距离,其计算复杂度要明显高于欧式距离的计算复杂度.随着数据集规模的增加,这个弊端尤为明显,就更无法应用于大规模聚类问题.

对于一个数据集,如果我们只利用其中一部分样本点的信息就可以得出对整个数据集的正确聚类,势必会大大缩减计算量.我们发现,对于一个数据集,无论是超球体分布还是复杂结构,我们总能将其划分为若干类,使

得每一类都是符合球形分布的简单子数据集.这一步可以通过选取合适的类数 K,对数据集进行 K-均值聚类轻易实现.而这些子数据集的聚类中心即可很好地代表整个数据集的分布特点,此时采用具有流形距离作为相似性测度的聚类算法来处理代表点集(即 K-均值聚类中心集),即可获得符合数据分布结构的合理聚类结果.这样就可以达到降低计算量的目的.

基于上面的思想,我们设计了如下的两阶段聚类算法.

2.1 基于欧氏距离测度的粗聚类

第 1 阶段旨在根据全体样本点提供的信息,寻找若干符合球形分布的子数据集,并构造出一个较小的代表点集,这些代表点能够较为完整、准确地表现整个数据集的分布形状及结构特点.我们选择采用欧氏距离测度的改进 K-均值算法作为第 1 阶段的选择手段,将得到的聚类中心作为数据集的代表点.对于数据集 $X=\{x_1,\ldots,x_N\}$,期望聚为 K 类,我们先通过 TPC 的第 1 阶段将其聚为 K'类,表示为 $C=\{C_1,C_2,\ldots,C_{K'}\}$,聚类中心为 $\mu=\{\mu_1,\mu_2,\ldots,\mu_{K'}\}$,其中,$K'>K$.

2.1.1 消除 KM 初始化敏感的初始中心选择策略

K-均值算法使用样本与其相应聚类中心的欧式距离之和作为评判准则,通过最小化该评判准则实现对样本集的划分,它是一种用于很多聚类应用领域的快速迭代算法.然而,K-均值算法的主要缺点之一是对于初始化聚类中心的位置很敏感.针对 K-均值算法对初始值敏感这一缺点,本文采用一种全局的方案选择初始点以解决这一问题.具体来说,为了解决一个 M 类聚类问题,首先寻找整个数据集的质心,作为将数据集聚为一类时的聚类中心.在已经将数据集聚为 $k-1$ 类后,k 类聚类问题的初始聚类中心为$(x_{i1},x_{i2},\ldots,x_{i(K-1)},x_{iK})$.其中,

$$b_N=\sum_{j=1}^{N}\max(d_{k-1}^{j}-\|x_n-x_j\|^2,0),i=\arg\max_N b_N \tag{3}$$

d_{k-1}^{j} 为样本点 x_j 与目前$(k-1)$个聚类中心中最近的一个之间的距离[12].通过上述过程,最终获得解决 M 类聚类问题的初始聚类中心.算法流程如图 2 所示,图 2(b)为图 2(a)中的局部搜索部分,其中,N 为样本个数,K 为聚类类数,$i\in[1,N]$,$k\in[1,K]$均为正整数.上述方案解决了初始聚类中心选择的问题.

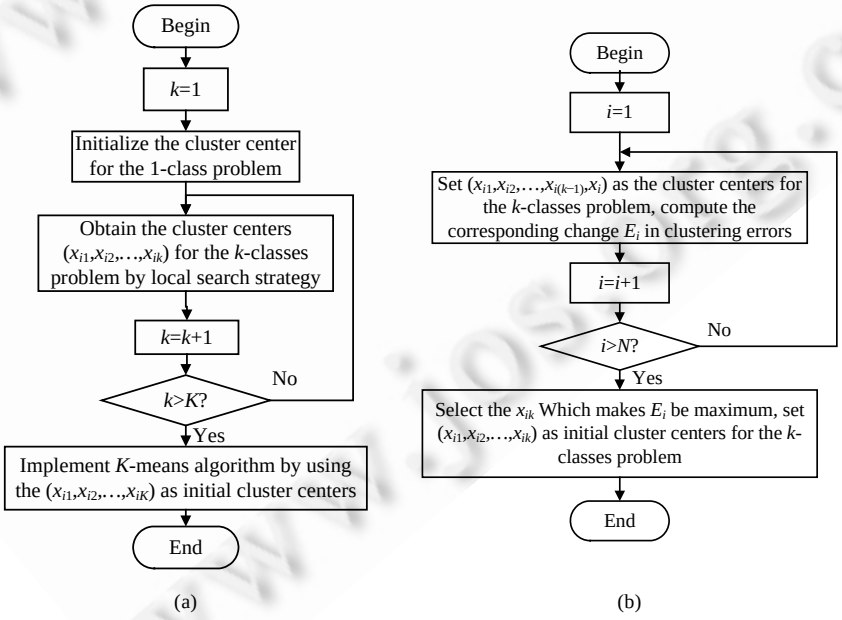


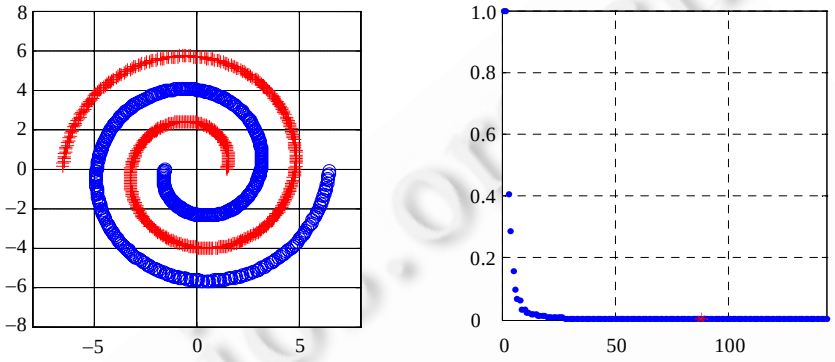
Fig.2 Flow chart of global selection strategy

图 2 全局选点策略的流程图

2.1.2 自适应类数选择

针对一个未知数据集,第 1 阶段需要聚成多少类才能较为完整地体现数据集的分布特点,是本节要解决的问题.

TPC 第 1 阶段中的改进 *K*-均值算法采用一个具有全局特性的初始聚类中心选择方法.此方案还可以发现合适的聚类类数,只要在每次添加新的聚类中心后引入适当的标准来决定当前的类别数是否为最恰当的类别数[13].我们通过寻找误差曲线拐点的方法来确定第 1 阶段的聚类类数.采用聚类时类内距离之和的变化量作为误差曲线,选取曲线中第 1 个低于某个阈值的点作为第 1 阶段的聚类类数,并结束第 1 阶段的处理,进入第 2 阶段.如图 3 所示,图 3(a)为二维人工数据 Spiral,图 3(b)是随着类别数的增加,聚类误差的变化曲线,浅色的点表示第 1 个低于选定阈值的点,该点对应的类别数 85 为第 1 阶段针对 Spiral 数据集合适的聚类类数.



(a) Spiral data set
(a) Spiral 数据集

(b) Clustering error curve
(b) 聚类误差曲线

Fig.3 Adaptive selection of the number of clusters for Sporal data set

图 3 Spiral 数据的自适应类别数选择

2.2 基于流形距离的二次聚类

对第 1 阶段得到的聚类中心$\mu=\{\mu_1,\mu_2,\ldots,\mu_{k'}\}$,我们在 TPC 的第 2 阶段使用 MEC 将其聚为 *K* 类.在设计基于进化计算的聚类算法时,最核心的两个问题就是进化个体的编码以及相似性度量.我们从组合优化的角度来考虑聚类问题,将指定类别数 *K* 的聚类问题建模为一个从数据集中选择 *K* 个典型样本来代表 *K* 个类别的优化问题,然后按照样本与 *K* 个典型样本的相似性对数据集进行类别划分.因此,每个个体代表一种典型样本序号的组合.显然,对于一个 *K* 类的聚类问题,一个个体的长度为 *K*,第 1 个基因位为代表第 1 个类别的样本序号,第 2 个基因位为代表第 2 个类别的样本序号,依此类推.为了更加具体地说明这种新的个体表示方式,我们考虑以下简单的例子:

例 1:假设数据集大小为 100,类别数目为 5,则个体(6,19,38,64,91)表示第 6 个样本、第 19 个样本、第 38 个样本、第 64 个样本、第 91 个样本分别代表第 1 类~第 5 类.需要注意的是,为了减少搜索空间,我们将个体中每个基因位按照从小到大的顺序排列.也就是说,个体(6,19,38,64,91)与个体(6,19,64,38,91)将被视为一个个体.

显然,这种编码方式没有涉及数据的维数,因此,搜索空间的大小与数据维数无关.而 Maulik 等人提出的遗传聚类算法[3]将 *K* 个聚类中心编码为个体,这样,对于 *d* 维的数据聚类问题,其编码长度为 $d\times K$,且该编码方式决定了该算法为一个连续空间的优化问题.而我们提出的编码方式,编码长度为 *K*,与 *d* 无关,且为离散空间的优化问题,降低了搜索空间的大小.

针对聚类问题,个体适应度按如下方式计算:首先,根据个体表示的各类别的典型样本,以流形距离作为相似性度量,将所有无类属的样本数据划分到不同的类别中.将点 $x_i,i=1,2,\ldots,n$ 划分到类 $C_j,j\in\{1,2,\ldots,K\}$,遵循下列原则:

$$j=\arg\min_{j=1,2,\ldots,K}(D(x_i,u_j)) \tag{4}$$

其中,u_j为代表第 j 类的典型样本.类别划分完成之后,个体 $\boldsymbol{b}$ 的个体适应度值按公式(5)计算得到:

$$Aff(\boldsymbol{b})=\frac{1}{1+\sum_{C_k\in C}\sum_{i\in C_k}D(i,\mu_k)} \tag{5}$$

其中,C 为个体 $\boldsymbol{b}$ 对应的类别划分,μ_k为代表类别 C_k 的典型样本,$D(i,\mu_k)$为类别 C_k 中的第 i 个样本与μ_k之间的流形距离.

基于以上个体编码方式和适应度计算方法,流形进化聚类算法(MEC)使用经典的赌轮选择方法、交叉算子和变异算子,搜索最优的数据划分.MEC 的步骤如下所示:

算法 1. 流形进化聚类算法(MEC).

Begin

1. t=0
2. 随机初始化种群 $P(t)$
3. 根据流形距离将所有的点聚类,并计算目标函数值 $P(t)$
4. if $t<G_{\max}$
5. $t=t+1$
6. 用经典赌轮选择法从 $P(t-1)$中选择 $P(t)$
7. 交叉 $P(t)$
8. 变异 $P(t)$
9. 转至第 3 步
10. end if
11. 输出最优解并停止

End

其中,$G_{\max}$为最大迭代次数.第 2 步中的初始化是从[1,N]中随机生成实整数 K 个,其中,N 为数据集的大小.重复这个过程 P 次以生成种群中的 P 个染色体,其中,P 为种群的大小.

根据第 1 阶段聚类结果整理最终聚类结果,即μ_i最终被聚为哪一类,则它所代表的点集 C_i 就属于哪一类.我们通过下面的例子进一步说明.

例 2:假设数据集 X 的大小为 1 000,期望聚类类数为 4.通过第 1 阶段将其聚为 100 类,由 $C=\{C_1,C_2,\ldots,C_{100}\}$表示,这 100 类的聚类中心表示为$\mu=\{\mu_1,\mu_2,\ldots,\mu_{100}\}$,则第 2 阶段将聚类中心$\mu$聚为 4 类.假设第 1 类为$\{\mu_1,\mu_2,\ldots,\mu_{n1}\}$,第 2 类为$\{\mu_{n1+1},\mu_{n1+2},\ldots,\mu_{n2}\}$,第 3 类为$\{\mu_{n2+1},\mu_{n2+2},\ldots,\mu_{n3}\}$,第 4 类为$\{\mu_{n3+1},\mu_{n3+2},\ldots,\mu_{100}\}$,那么最终的聚类结果为$\{C_1,C_2,\ldots,C_{n1}\}$为第 1 类,$\{C_{n1+1},C_{n1+2},\ldots,C_{n2}\}$为第 2 类,$\{C_{n2+1},C_{n2+2},\ldots,C_{n3}\}$为第 3 类,$\{C_{n3+1},C_{n3+2},\ldots,C_{100}\}$为第 4 类.

基于前面的内容, TPC 的步骤如下所示:

算法 2. 二阶段聚类算法(TPC).

Begin

1. 对数据集 $X=\{x_1,\ldots,x_N\}$使用改进的 K-均值算法,得出聚类中心$\mu=\{\mu_1,\mu_2,\ldots,\mu_{k'}\}$
2. 对$\mu=\{\mu_1,\mu_2,\ldots,\mu_{k'}\}$执行 MEC,将其分为 K 类
3. 结合 Step 1 与 Step 2 两次聚类的结果,得出最终聚类结果
4. 输出解并停止

End

TPC 在保证对复杂数据有较好聚类性能的同时,大大缩减了计算量.另一方面,由文献[14]的分析方法可知,如果原始数据存在噪声,则通过两阶段的聚类可以在一定程度上减少噪声的影响,从而使聚类结果更加合理.

3 对比实验与结果分析

我们将 TPC 与 *K*-均值算法(*k*-means,简称 KM)[2]、遗传聚类算法(genetic algorithm-based clustering technique,简称 GAC)[3]和流形进化聚类算法(manifold evolutionary clustering,简称 MEC)[9]进行性能比较.采用的测试问题包括人工数据集聚类、UCI 数据集[15]聚类.表 1 给出了这些数据集的性质.

Table 1 Data sets used in experiments

表 1 实验中使用的数据集

Data sets	Number of samples	Number of features	Number of classes
Square1	1 000	2	4
Square4	1 000	2	4
Long1	1 000	2	2
Spiral	1 000	2	2
Sizes5	1 000	2	4
Line-Blobs	266	2	3
Sticks	512	2	4
Iris	150	4	4
Breast	277	9	2
Zoo	101	16	7
German	1 000	24	2
Pimaindians	768	8	2
Musk	6 598	166	2
Page	5 473	10	5

我们采用聚类正确率指标来衡量各算法的性能,聚类正确率即为正确聚类的样本个数与数据集样本总数的百分比.聚类正确率指标是区间[0,1]中的正数,值越大,说明聚类效果越好.

3.1 人工数据集测试结果

为了能够直观地考察 TPC 算法的性能,我们首先将该算法用于 7 个人工数据集的聚类问题.这 7 个人工数据集分别命名为 Square1 数据集、Square4 数据集、Long1 数据集、Spiral 数据集、Sizes5 数据集、Line-blobs 数据集和 Sticks 数据集.其中,TPC 的参数设置如下:第 1 阶段的最大迭代次数设置为 100,停止阈值设置为 10^{-4};第 2 阶段终止条件为迭代次数 100,种群规模为 20,交叉概率为 0.8,变异概率为 0.1.GAC 终止条件为迭代次数 100,种群规模为 20,交叉概率为 0.8,变异概率为 0.1.KM 的最大迭代次数设置为 100,停止阈值设置为 10^{-4}.基于上述 7 个人工数据集的参数敏感性实验表明,收缩因子ρ在(1,e^{18}]范围内取值时,对 TPC 性能的影响很不明显,在以下实验中令$\rho=e^2$.

a) 人工数据集 A

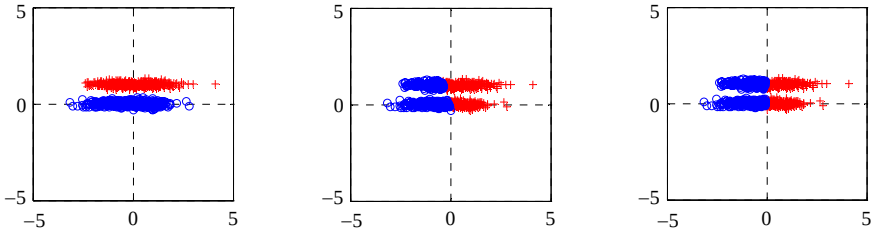
为了更好地说明算法对于分布特点不同的数据集的聚类性能,我们将人工数据分为两组.人工数据集 A 为 4 个具有复杂流形分布的数据集,分别是数据集 Long1,Spiral,Lineblobs 和 Sticks.以下分别展示上述 3 种算法对于人工数据集 A 的典型聚类结果,以使数据分布情况及聚类效果更直观地显现.

对于每一个数据集,我们独立运行 50 次.表 2 列出了各算法在求解上面 4 个聚类问题时得到的聚类正确率平均值和标准差.从表 2 中的统计数据和如图 4 所示的典型聚类结果可以清楚地看到,对于 Long1,Spiral,Lineblobs,Sticks 这 4 个具有明显流形分布特点的数据集,第 2 阶段采取流形距离的 TPC 取得了完全正确的聚类结果,而另外两种采用欧氏距离作为相似性度量的聚类算法对这 4 个数据集的聚类效果则非常差.

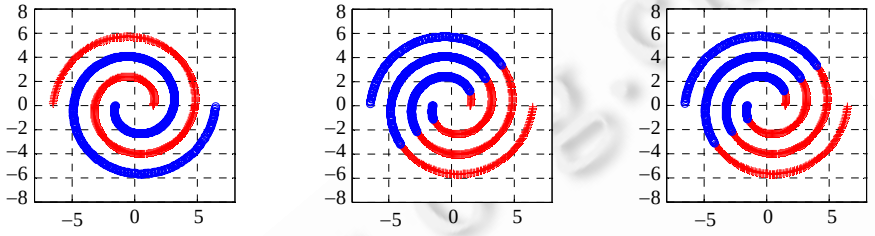
Table 2 Performance comparison of three algorithms in solving the artificial data sets A

表 2 3 种算法在求解人工数据集 A 时的性能比较

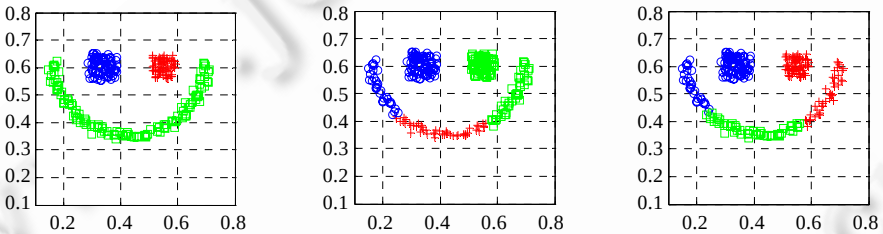
Data sets	Clustering correct ratio (standard deviation)		
	TPC	GAC	KM
Long1	1 (0)	0.5140 (1.59E−02)	0.5464 (1.33E−01)
Spiral	1 (0)	0.5920 (2.30E−03)	0.5927 (4.80E−03)
Lineblobs	1 (0)	0.7406 (1.40E−03)	0.7425 (4.58E−02)
Sticks	1 (0)	0.7188 (4.90E−03)	0.6895 (3.87E−02)



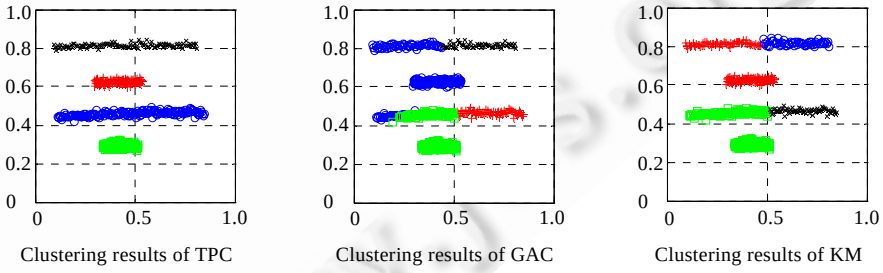
(a) Typical clustering results on Long1 obtained from three algorithms
(a) 3 种算法对数据集 Long1 的典型聚类结果



(b) Typical clustering results on Spiral obtained from three algorithms
(b) 3 种算法对数据集 Spiral 的典型聚类结果



(c) Typical clustering results on Lineblobs obtained from three algorithms
(c) 3 种算法对数据集 Lineblobs 的典型聚类结果



(d) Typical clustering results on Sticks obtained from three algorithms
(d) 3 种算法对数据集 Sticks 的典型聚类结果

Fig.4 Typical clustering results of three algorithms on the artificial data sets A
图 4 3 种算法对人工数据集 A 的典型聚类结果

b) 人工数据集 B

人工数据集 B 为 3 个具有球形分布的数据集,分别是数据集 Square1,Square4 和 Sizes5.下面分别给出上述

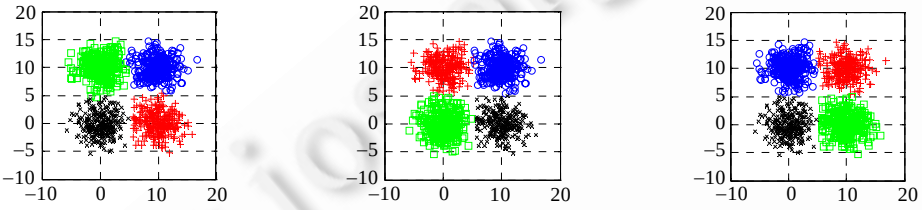
3 种算法对于人工数据集 B 的聚类结果.

同样,对于每一个数据集,我们独立运行 50 次,表 3 列出了各算法在求解 Square1 数据集、Square4 数据集和 Sizes5 数据集这 3 个聚类问题时得到的聚类正确率的平均值和标准差.从表 3 中的统计数据和如图 5 所示的典型聚类结果可以看到,对于 Square1,Square4 和 Sizes5 这 3 个具有球形分布的数据集,3 种算法均没有得到完全正确的聚类结果,且表现比较接近.在处理 Sizes5 数据集时,TPC 得到了更好的聚类结果.这是由于 Sizes5 这个数据集虽然是球形分布的数据集,但是类与类之间规模差距悬殊,结构比较复杂,而 TPC 能够更好地处理复杂数据聚类.对于数据集 Square1 和数据集 Square4,KM 和 GAC 算法取得了较好的聚类结果.但可以看出,TPC 与 KM 的聚类结果相差甚微.

Table 3 Performance comparison of three algorithms in solving the artificial data sets B

表 3 3 种算法在求解人工数据 B 时的性能比较

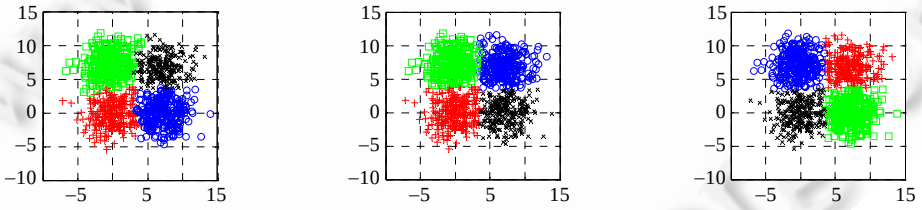
Data sets	Clustering correct ratio (standard deviation)		
	TPC	GAC	KM
Square1	0.9880 (3.36E−16)	0.9900 (7.16E−04)	0.9900 (5.00E−02)
Square4	0.9285 (1.40E−03)	0.9350 (2.10E−03)	0.9350 (9.13E−04)
Sizes5	0.9838 (1.12E−15)	0.9750 (4.30E−03)	0.7744 (1.71E−01)



Clustering results of TPC Clustering results of GAC Clustering results of KM

(a) Typical clustering results on Square1 obtained from three algorithms

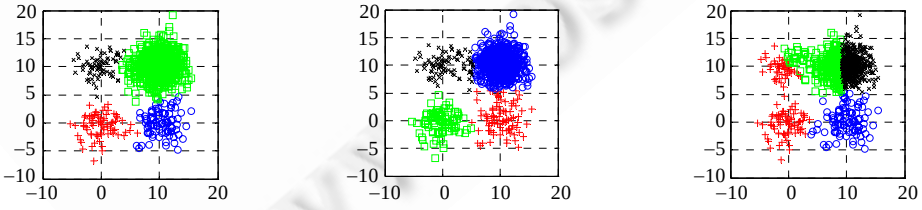
(a) 3 种算法对数据集 Square1 的典型聚类结果



Clustering results of TPC Clustering results of GAC Clustering results of KM

(b) Typical clustering results on Square4 obtained from three algorithms

(b) 3 种算法对数据集 Square4 的典型聚类结果



Clustering results of TPC Clustering results of GAC Clustering results of KM

(c) Typical clustering results on Sizes5 obtained from three algorithms

(c) 3 种算法对数据集 Sizes5 的典型聚类结果

Fig.5 Typical clustering results of three algorithms on the artificial data sets B

图 5 3 种算法对人工数据集 B 的典型聚类结果

以上两组实验充分说明,TPC 在处理数据分布复杂的数据聚类问题时十分有效.

3.2 UCI数据集测试结果

此外,我们还选择了 7 个 UCI 数据集分别对 3 种算法进行测试,以便更全面地考察算法的性能.这 7 个 UCI 数据集分别为 Iris 数据集、Breast 数据集、Zoo 数据集、German 数据集、Pimaindians 数据集、Musk 数据集和 Page 数据集.各算法参数与第 3.1 节相同.

对每一个数据集独立运行 50 次后,表 4 分别列出了各算法在求解这 7 个聚类问题时得到的聚类正确率的平均值和标准差.从表 4 中的统计数据可以看到,对于 Iris,Balance,Breast,Zoo,German,Pimaindians,Musk 和 Page 这 7 个 UCI 数据集,3 种算法均没有得到完全正确的聚类结果.总体来说,TPC 优于另外两种算法.对 Breast, German,Pimaindians 和 Musk 这 4 个数据的处理,TPC 明显表现得比另外两种算法优秀.对于另外 3 个数据,TPC 在 3 种聚类算法中的聚类正确率指标也有较大优势,说明 TPC 算法对于现实世界的复杂分布数据聚类问题有很好的性能.

Table 4 Performance comparison of three algorithms in solving the UCI data sets

表 4 3 种算法在求解 UCI 数据集时的性能比较

Data sets	Clustering correct ratio (standard deviation)		
	TPC	GAC	KM
Iris	0.9267 (1.12E−15)	0.8933 (2.60E−03)	0.8116 (1.50E−01)
Breast	0.7076 (0)	0.5471 (1.12E−16)	0.4897 (9.85E−02)
Zoo	0.8614 (0)	0.7619 (3.36E−16)	0.7007 (1.04E−01)
German	0.7000 (1.12E−16)	0.5970 (4.10E−03)	0.5970 (4.49E−16)
Pimaindians	0.6510 (5.61E−16)	0.5287 (7.00E−03)	0.5482 (1.20E−03)
Musk	0.8386 (3.73E−14)	0.5040 (3.67E−03)	0.5150 (6.38E−03)
Page	0.8771 (6.21E−15)	0.8030 (2.44E−03)	0.7312 (5.76E−03)

3.3 第1阶段采用不同算法的性能比较

上面的实验展现了 TPC 算法与其他算法的性能比较,但是 TPC 算法是分为两个阶段的,而且第 1 阶段算法的选择对于第 2 阶段以及整个算法的性能都有一定的影响,所以下面我们对于第 1 阶段选择了另外两种传统经典的算法 KM 和 FCM 与本文的第 1 阶段算法(标记为 IKM)做了比较.表 5 是 TPC 第 1 阶段使用本文算法与 KM 和 FCM 对于上述 14 个数据独立运行 50 次的运行时间、平均聚类正确率和标准差.

Table 5 Performance comparison of three algorithms in the first stage

表 5 第 1 阶段 3 种算法的性能比较

Data sets	Clustering correct ratio (standard deviation)			Running time (s)		
	KM	FCM	IKM	KM	FCM	IKM
Square1	0.9889 (2.0E−03)	1 (0)	0.9880 (0)	3.428 8	18.765	6.543 4
Square4	0.9281 (3.7E−03)	0.9289 (3.0E−03)	0.9285 (1.4E−03)	3.419 6	16.147	7.390 6
Long1	1 (2.8E−04)	0.9995 (7.6E−04)	1 (0)	3.164 6	8.489 6	4.247 1
Spiral	1 (0)	1 (0)	1 (0)	3.207 1	18.765	6.702 1
Sizes5	0.9888 (2.3E−03)	0.9891 (2.4E−03)	0.9838 (1.1E−15)	3.369 6	15.951	7.190 6
Line blobs	0.9979 (1.1E−02)	1 (0)	1 (0)	1.342 5	1.478 4	1.471 8
Sticks	0.9977 (1.6E−02)	1 (0)	1 (0)	1.884 4	2.907 1	2.146 8
Iris	0.9077 (2.6E−02)	0.9038 (1.9E−02)	0.9267 (1.1E−15)	0.803 6	1.279 3	1.341 2
Breast	0.6763 (4.1E−02)	0.6744 (1.4E−02)	0.7076 (0)	1.267 5	1.436 8	2.004 6
Zoo	0.7713 (4.8E−02)	0.7976 (3.5E−02)	0.8614 (0)	1.473 4	1.616 8	1.543 4
German	0.6997 (7.41E−04)	0.7000 (1.2E−16)	0.7000 (1.1E−16)	2.954 1	42.64 8	6.978 1
Pimaindians	0.64901 (9.2E−04)	0.651 (5.6E−16)	0.6510 (5.6E−16)	2.786 8	14.017	5.964 0
Musk	0.8017 (2.94E−4)	0.8321 (5.37E−10)	0.8386 (3.73E−14)	203.253 6	1537.665 7	807.190 6
Page	0.8238 (3.29E−6)	0.8569 (6.37 E−16)	0.8771 (6.21 E−15)	168.354 9	1375.682 3	501.368 7

从表 5 的统计结果可以看出,第 1 阶段采用不同的算法对整体算法的性能有着较大的差异.在聚类正确率方面,对于流形分布的复杂数据,本文设计的IKM在聚类正确率和稳定性上具有明显的优势;对于特征维数较高的 UCI 数据,IKM 也获得了较高的聚类正确率.在运行时间方面,虽然没有 *K*-均值算法速度快,但要快于 FCM.

考虑到聚类正确率和稳定性,本文算法第 1 阶段采用了改进的 *K*-均值算法.

3.4 运行时间比较

从上面的实验可以看出,TPC 的聚类性能比较令人满意.如果我们直接用 MEC 来处理这些数据,所有样本点对于数据集分布提供的信息都将被利用,聚类正确率将得到保障.然而,这样做势必会带来很大的计算量.我们对 TPC 和 MEC 处理上述 14 个聚类问题所用的 CPU 时间做了测试.表 6 是 TPC 与 MEC 对于上述 14 个数据独立运行 50 次的平均运行时间(其中,“—”表示在 24 小时内没有运行出结果).

Table 6 Comparison of computational time of the two algorithms

表 6 两种算法的运行时间比较

Data sets	Clustering correct ratio		Running time (s)	
	TPC	MEC	TPC	MEC
Square1	0.987 2	**0.989 5**	**6.543 4**	14.740 6
Square4	0.928 5	**0.933 6**	**7.390 6**	14.953 1
Long1	**1**	**1**	**4.247 1**	14.131 3
Spiral	**1**	**1**	**6.702 1**	13.584 4
Sizes5	0.983 8	**0.988 5**	**7.190 6**	15.015 6
Line blobs	**1**	**1**	**1.471 8**	2.271 9
Sticks	**1**	**1**	**2.146 8**	4.906 3
Iris	**0.907 7**	0.901 3	**1.341 2**	1.825 6
Breast	**0.707 6**	**0.707 6**	**2.004 6**	2.680 9
Zoo	**0.861 4**	0.792 1	**1.543 4**	1.982 8
German	**0.700 0**	**0.700 0**	**6.978 1**	14.115 6
Pimaindians	**0.651 0**	**0.651 0**	**5.964 0**	21.250 9
Musk	**0.838 6**	—	**807.190 6**	—
Page	**0.877 1**	—	**501.368 7**	—

从表 6 的数据信息可以看出,TPC 在保持较高聚类正确率的同时,大大提高了运算效率,在运行时间上远远优于 MEC,对数据规模大于 500 的数据上尤为明显,如 Square1,Long1,Spiral,Sizes5,Sticks,German 和 Pimaindians,TPC 可以节约一半以上的时间.对于数据规模比较小的数据集,如 Lineblobs,Iris 和 Zoo,TPC 相比 MEC 在时间上也有所降低.而对于数据集 Breast,Musk 和 Page,由于 MEC 计算复杂度过高而在 24 小时内没有运行出结果,TPC 在很短的时间里得到了较好的聚类结果.

3.5 鲁棒性分析

为了考察 3 种算法的鲁棒性,我们采用文献[16]中的鲁棒性分析方法对 3 种算法在求解上述 14 个问题时的鲁棒性进行比较.具体来说,算法 m 在某一特定数据集上的相对性能用该算法获得的聚类正确率指标的值 R_m 与所有算法在求解该问题时得到的最大聚类正确率指标的值相除来衡量,即

$$b_m = \frac{R_m}{\max\limits_k R_k} \tag{6}$$

因此,在某个数据集上表现最好的算法 m^*的相对性能 $b_{m^*}=1$,而其他算法的相对性能 $b_m \leqslant 1$.b_m 的值越大,则算法 m 在所有算法中的相对性能越好.所以,使用算法 m 在所有数据集上的 b_m 值的总和来客观评价算法的鲁棒性,总和越大鲁棒性越好.

图 6 为 3 种算法的鲁棒性比较结果.从中可以看出,TPC 获得了最高的 b_m 总和,达到了 13.990 2.而采用欧式距离作为相似性度量的 GAC 和 KM,b_m 总和值明显小于采用流形距离作为相似性度量的 TPC.这充分说明了基于流形距离的相似性度量对聚类问题具有很好的鲁棒性.其中,TPC 的 b_m 值对于具有流形分布的测试数据均为 1,对除了 Square4 以外的其他球形分布数据集也都表现出了很好的性能.可以说,TPC 在所有比较的 3 种算法中具有很好的鲁棒性.

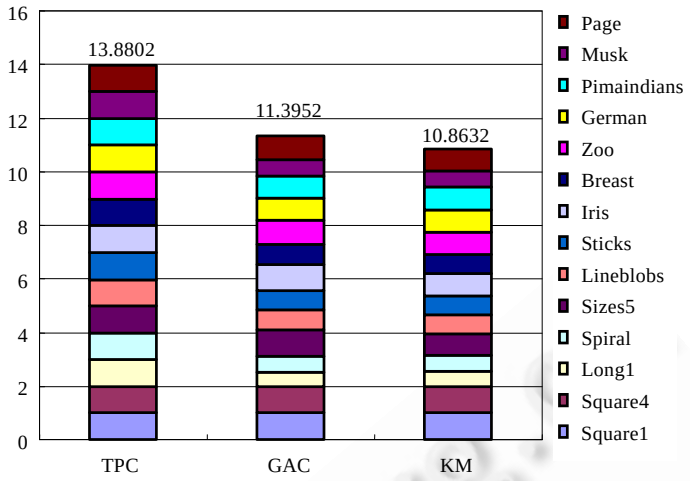


Fig.6 Comparison of robustness of three algorithms

图 6 3 种算法的鲁棒性比较

4 结 论

本文提出了一种能够处理复杂分布数据聚类问题的二阶段聚类算法.该方法在解决了 *K*-均值算法初始化敏感问题的基础上,利用基于欧式距离的聚类算法寻找数据集的代表点,并自适应地选择代表点个数,再通过基于流形距离的进化算法对代表点聚类来缩减对于复杂聚类问题的计算量.该算法的主要优点是可以用来处理复杂聚类问题,算法较为稳定,并且在保持聚类正确率的同时,降低了流形距离作为相似性度量的计算代价.

我们在不同的数据集上对上述算法与另外 3 种聚类算法进行了对比测试,以证明它对于复杂分布数据聚类问题的有效性和高效性.在人工数据集的测试中,TPC 对流形分布数据的聚类效果是十分突出的;对于球形分布的人工数据,TPC 也与最优聚类结果相当.对于 UCI 数据集,TPC 的聚类效果明显好于其他对比算法.最后,我们用上述人工数据集和 UCI 数据集比较了 TPC 和 MEC,在聚类性能相当的前提下,TPC 在时间上明显优于 MEC,有效而且高效.

References:

[1] Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Computing Surveys, 1999,31(3):264−323. [doi: 10.1145/331499.331504]

[2] Hartigan JA, Wong MA. A *k*-means clustering algorithm. Applied Statistics, 1979,28(1):100−108. [doi: 10.2307/2346830]

[3] Maulik U, Bandyopadhyay S. Genetic algorithm-based clustering technique. Pattern Recognition, 2000,33(9):1455−1465. [doi: 10.1016/S0031-3203(99)00137-5]

[4] Sheng WG, Swift S, Zhang LS, Liu XH. A weighted sum validity function for clustering with a hybrid niching genetic algorithm. IEEE Trans. on System, Man Cybernetics-part B: Cybernetics, 2005,35(6):1156−1167. [doi: 10.1109/TSMCB.2005.850173]

[5] Gong MG, Jiao LC, Wang L, Bo LF. Density-Sensitive evolutionary clustering. In: Proc. of the 11th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD 2007). Springer-Verlag, 2007. 507−514. [doi: 10.1007/978-3-540-71701-0_52]

[6] Sarafis IA, Trinder PW, Zalzala AMS. NOCEA: A rule-based evolutionary algorithm for efficient and effective clustering of massive high-dimensional databases. Applied Soft Computation, 2007,7(3):668−710. [doi: 10.1016/j.asoc.2006.01.011]

[7] Das S, Abraham A, Konar A. Automatic clustering using an improved differential evolution algorithm. IEEE Trans. on System, Man Cybernetics-part A: System, Humans, 2008,38(1):218−237. [doi: 10.1109/TSMCA.2007.909595]

[8] Bandyopadhyay S, Saha S. A point symmetry-based clustering technique for automatic evolution of clusters. IEEE Trans. on Knowledge and Data Engineering, 2008,20(11):1441−1457. [doi: 10.1109/TKDE.2008.79]

[9] Gong MG, Jiao LC, Bo LF, Wang L, Zhang XR. Image texture classification using a manifold distance based evolutionary clustering method. Optical Engineering, 2008,47(7):077201-1−077201-10. [doi: 10.1117/1.2955785]

[10] Su MC, Chou CH. A modified version of the k-means algorithm with a distance based on cluster symmetry. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2001,23(6):674−680. [doi: 10.1109/34.927466]

[11] Zhou DY, Bousquet O, Lal TN, Weston J, Schölkopf B. Learning with local and global consistency. In: Advances in Neural Information Processing Systems 16. Cambridge: MIT Press, 2004. 321−328.

[12] Likas A, Vlassis N, Verbeek JJ. The global *k*-means clustering algorithm. Pattern Recognition, 2003,36(2):451−461. [doi: 10.1016/S0031-3203(02)00060-2]

[13] Salvador S, Chan P. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In: Proc. of the 16th IEEE Int'l Conf. on Tools with Artificial Intelligence (ICTAI 2004). Florida: Florida Institute of Technology, 2004. 576−584. [doi: 10.1109/ICTAI.2004.50]

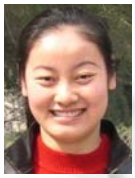
[14] Zhou ZH, Jiang J. NeC4.5: Neural ensemble based C4.5. IEEE Trans. on Knowledge and Data Engineering, 2004,16(6):770−773. [doi: 10.1109/TKDE.2004.11]

[15] Blake CL, Merz CJ. UCI machine learning repository. 2010. http://archive.ics.uci.edu/ml/

[16] Geng X, Zhan DC, Zhou ZH. Supervised nonlinear dimensionality reduction for visualization and classification. IEEE Trans. on Systems, Man Cybernetics-Part B: Cybernetics, 2005,35(6):1098−1107. [doi: 10.1109/TSMCB.2005.850151]

公茂果(1979－),男,山东蒙阴人,博士,教授,CCF 高级会员,主要研究领域为计算智能,数据挖掘.

曹宇(1986－),男,硕士生,CCF 会员,主要研究领域为计算智能,数据挖掘.



王爽(1978－),女,博士,副教授,CCF 高级会员,主要研究领域为多尺度几何分析,图像处理,SAR 图像处理.

焦李成(1959－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为自然计算,智能信息处理.

马萌(1986－),女,硕士生,主要研究领域为进化计算,数据挖掘.

马文萍(1981－),女,博士,讲师,CCF 会员,主要研究领域为人工免疫系统,进化计算,图像处理.