文章编号: 1001-0920 (2016) 03-0410-07 **DOI:** 10.13195/j.kzyjc.2014.1978

基于改进流形距离和人工蜂群的二阶段聚类算法

夏卓群, 欧 慧, 李 平, 武志伟, 戴 傲

(长沙理工大学 计算机与通信工程学院，长沙 410114)

摘 要: 以改进的流形距离为相似度测度, 结合人工蜂群算法, 提出一种二阶段聚类算法. 首先根据局部密度、最大最小距离和近邻选择对数据集初步归类并得到簇代表点; 然后将聚类归属为优化问题, 通过改进的蜂群算法对簇代表点及没归类的样本点较快地搜索到最优聚类中心, 同时根据流形距离的全局一致性特征, 对样本进行精确的类别划分; 最后将两阶段算法综合归类. 实验结果表明, 所提出的算法可以获得良好的聚类效果.

关键词: 流形距离； 人工蜂群算法； 局部密度； 最大最小距离； 近邻选择

中图分类号: TP18 **文献标志码:** A

Two-phase clustering algorithm based on the improved manifold distance and the artificial bee colony algorithm

XIA Zhuo-qun, *OU Hui*, *LI Ping*, *WU Zhi-wei*, *DAI Ao*

(College of Computer and Communication Engineering，Changsha University of Science and Technology，Changsha 410114，China. Correspondent：OU Hui，E-mail：1016231422@qq.com)

Abstract: A two-phase clustering algorithm based on the improved manifold distance as the similarity measure combined with the bee colony algorithm is proposed. Firstly, based on local density, max-min distance and neighbors selecting, data set is initialized, and the representative points are obtained. Then, the clustering algorithm is viewed as an optimization problem, in which the correctly category is obtained by getting the optimal clustering center through the improved bee colony algorithm dealing with the representative and unclassified points, and obtaining the overall consistency information of the manifold distance. Finally, the two phase algorithms are classified. Experiment results show that the proposed algorithm has better clustering results.

Keywords: manifold distance； artificial bee colony algorithm； local density； max-min distance； neighbors selecting

0 引 言

聚类算法在数据挖掘中占据重要地位, 它按一定的要求和规律对事物进行区分和归类, 而距离是经常采用的相似度度量方式. 一般的聚类算法, 如 k-means, 以欧氏距离作为相似度测度, 但其目标函数是高度非线性的、多峰的, 所以 k-means 使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, 易导致局部最优, 且只对空间分布为球形或超球体数据集具有较好的效果. 为解决这一问题, Su 等[1]提出了非公制测度作为相似度测度, 对复杂流形结构数据集具有较好的性能; 在此基础上, Wang 等[2-3]提出了密度敏感的距离测度; Gong 等[4-6]提出流形距离测度等. 这些距离对数据全局一致性具有较好的体现, 但是文献 [5-6] 提出的流形距离中伸缩因子在较大范围内取值不敏感, 导致反应数据全局一致性问题上存在缺陷. 鉴于此, 本文提出一种改进的流形距离以解决这一问题.

在基于流形距离的聚类算法中有不少采用了进化算法. 如: 文献 [6] 采用经典的赌轮选择、交叉算子和变异算子搜索最优聚类划分; 文献 [7] 采用量子进化算法对数据集进行归类; 文献 [8] 采用多智能体进化对数据集进行聚类等. 这些进化算法在编码时采用随机选取的聚类中心进行编码, 从而导致算法精度较低. 对此, 本文提出将改进的人工蜂群算法与改进的流形距离相结合的方法, 以有效提高算法的精度.

基于以上分析并考虑到全局流形距离是计算图论中最短路径之和, 计算复杂度较高, 本文基于改进的流形距离, 采用二阶段聚类的方法对数据集进行归

收稿日期: 2014-12-30； **修回日期:** 2015-05-18.

基金项目: 湖南省自然科学基金项目(14JJ7043)； 湖南省教育厅重点项目(14A004).

作者简介: 夏卓群(1977−), 男, 副教授, 博士后, 从事数据挖掘、计算机网络等研究； 欧慧(1989−), 女, 硕士生, 从事数据挖掘的研究.

类. 首先采用局部密度、最大最小距离以及文献 [9] 中提出的近邻选择进行初步归类并得到簇代表点; 然后对簇代表点以及未归类的样本点采用改进的流形距离并结合人工蜂群算法进行精确归类, 使归类达到所需类数; 最后对两个阶段的数据进行整合, 得到总的聚类结果, 并通过实验验证了算法的有效性.

1　本文相关定义

1.1　局部密度

定义 1 (局部密度[10])　记样本 v_i 与其 k 近邻点的距离之和为 D_i, 局部密度为 LD_i. D_i 和 LD_i 定义为

$$D_i = d_{i1} + d_{i2} + \cdots + d_{ij} + \cdots + d_{ik}, \tag{1}$$

$$\mathrm{LD}_i = 1/D_i. \tag{2}$$

其中: $d_{i1} \leqslant d_{i2} \leqslant \cdots \leqslant d_{ij} \leqslant \cdots \leqslant d_{ik}$, d_{ij} 表示 v_i 与 v_j 之间的距离. 可知: LD_i 越大, 表明 v_i 附近数据越集中; LD_i 越小, 表明 v_i 附近数据越稀疏.

1.2　流形距离及其改进

1.2.1　流形距离

文献 [2-6] 中提出的距离定义为

$$L(x_i, x_j) = \rho^{\mathrm{dist}(x_i, x_j)} - 1, \tag{3}$$

其中 $\mathrm{dist}(x_i, x_j)$ 为 x_i 与 x_j 之间的欧氏距离. 易知这个距离测度主要由欧氏距离决定参数 ρ 的变化, 而任意两点之间的欧氏距离较小. 所以当 ρ 在较大范围内变化时, 所有数据间的变化相差较小, 导致聚类结果在反应全局一致性上有缺陷. 鉴于此, 本文提出如下改进的流形距离.

1.2.2　改进的流形距离

定义 2 (局部流形距离)　数据集任意两点 x_i、x_j 之间局部流形距离 $L(x_i, x_j)$ 定义为

$$L(x_i, x_j) = \omega \cdot \mathrm{dist}(x_i, x_j)^{\ln\rho^2+1}, \tag{4}$$

其中 $\mathrm{dist}(x_i, x_j)$ 为 x_i 与 x_j 之间的欧氏距离. 在式 (4) 中, 对 ρ 平方取对数加 1 既能抑制 ρ 在较大范围内取值时结果突变, 也能使 ρ 在一个适当的梯度内取值. 在式 (4) 中增加平衡因子 ω 以对距离作适当的调整. 一般情况下 $\rho > 1$, 但 ρ 取越大距离变化越快, 这里 $\omega = 1/\rho^2$ 以稳定 ρ 取值对整个距离的影响. 例如在 Square1 中随机取 4 个数

$$a\,(10.918\,5, 4.889\,61),\ b\,(12.445\,4, 9.743\,54),$$

$$c\,(13.894\,3, 10.202\,3),\ d\,(6.531\,31, 9.248\,25),$$

当 ρ 取 2 和 5 时, 距离的变化分别如表 1 所示. 从表 1 中可以看出, 乘以 ω 后每条线段的距离适当地缩短了, 且后面要计算全局流形距离, 累加的结果会使距离的值更大, 所以局部流形距离乘以 $\omega = 1/\rho^2$ 可以平衡对整个距离的影响.

表 1　ω 对距离的影响

线段	$\rho = 2$		$\rho = 5$	
	乘以 ω 前	乘以 ω 后	乘以 ω 前	乘以 ω 后
ab	48.541 4	12.135 3	957.164 6	38.286 6
bc	2.715 1	0.678 8	5.846 9	0.233 9
cd	119.583 3	29.895 8	4 712.448 9	188.498 0

定义 3 (全局流形距离)　假设 $R = (R_1, R_2, \cdots, R_t)$ 是 x_i 与 x_j 之间的一条路径 ($R_t \in U$, U 为全集), 令 $R_s = \sum_{m=1}^{t-1} L(R_m, R_{m+1}), 0 < m < t$. 空间任意两点间的全局流形距离测度为连接两点间的所有局部流形距离之和的最小值, 即

$$G_d(x_i, x_j) = \min_{R \in R_{i,j}} R_s, \tag{5}$$

其中 $R_{i,j}$ 是 x_i 与 x_j 之间的所有路径.

2　算法描述及分析

2.1　基于近邻选择的粗聚类

这里称初步归类为“粗聚类”. 粗聚类主要是将数据集中的大部分样本点归类到具有代表性的样本中. 算法通过局部密度、最大最小距离以及文献 [9] 中提出的近邻选择初步归类. 假设基于欧氏距离的邻域大小参数为 K_d, 基于局部流形距离的邻域大小参数为 K_D, 且 $K_D \leqslant K_d$. 具体步骤如下.

Step 1: 计算 $X = \{x_1, x_2, \cdots, x_n\}$ 中任意样本 x_i 的局部密度 LD_i 并按升序记录于矩阵 M 中, 这里记为 $N_d(x_i)$.

Step 2: 由最大最小距离选择 $k'(k < k' < n)$ 个密度集: 局部密度最大的样本为第 1 个密度集 $N_d(x_1)$, 距离 $N_d(x_1)$ 最远的样本作为第 2 个密度集 $N_d(x_2)$; 对于剩余密度集, 根据 M 分别求出其到 $x_1, x_2, \cdots, x_m$ 之间的距离 $d_{i1}, d_{i2}, \cdots, d_{im}$, 取 $d_i = \min(d_{i1}, d_{i2}, \cdots, d_{im})$, $d = \max(d_i)$ 对应的密度集 $N_d(x_i)$, 以此类推, 直到 $N_d(x_{k'})$.

Step 3: 近邻选择.

for $i = 1$ to k'

令 $k = 1$

while $k < K_d$

取 $N_d(x_i)$ 中的第 k 点, 记为 x_{ik}

for $j = 1$ to K_D

令 $N_d(x_{ik})$ 中的第 j 点, 记为 x_{ikj}

if $L(x_i, x_{ik}) + L(x_{ik}, x_{ikj}) < L(x_i, x_{iK_d})$ 且 x_{ikj} 不在 $N_d(x_i)$ 中

将 x_{ikj} 按升序插入 $N_d(x_i)$, 并令 $L(x_i, x_{ikj}) = L(x_i, x_{ik}) + L(x_{ik}, x_{ikj})$, continue.

else if $L(x_i, x_{ik}) + L(x_{ik}, x_{ikj}) < L(x_i, x_{iK_d})$

更新 $L(x_i, x_{ikj}) = L(x_i, x_{ik}) + L(x_{ik}, x_{ikj})$,

将按照距离升序排列 $N_d(x_i)$; continue.

else if $L(x_i,x_{ik})+L(x_{ik},x_{ikj})>L(x_i,x_{ikd})$ or $(L(x_i,x_{ikj})=L(x_i,x_{ik})+L(x_{ik},x_{ikj})$ 且 $j==K_D)$, $k=k+1$; break;

end if

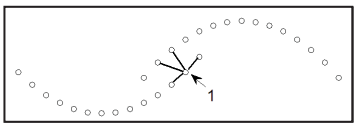
end for

end while

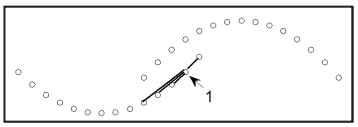
end for

Step 4: 对于 k' 个局部密度集中的 x_i, 取 $N_d(x_i)$ 中前 K_D 个点构成 x_i 的流形距离邻域 $N_D(x_i)$.

在 Step 2 中采用最大最小距离选择前 k' 个局部密度集, 具有代表性且保持了数据集的多样性. 通过 Step 3 计算后, 从中选择一部分以局部流形距离计算的近邻点. 通过以上步骤, 可以使大部分数据能服从流形结构归类, 从而保证细聚类归类时所有样本点都能正确归类, 如图 1 所示.



(a) 数据1以欧氏距离选择的近邻点



(b) 数据1邻域选择后选择的近邻点

图 1 数据 1 基于不同方式选择的近邻点

2.2 基于改进 ABC 算法的细聚类

这个阶段称精确归类为“细聚类”. 该步骤中通过引入改进的 ABC 算法不仅提高了算法的精度, 而且引入局部搜索算子加快了算法的收敛速度, 提高了整个算法的运行速度.

2.2.1 ABC 算法简介

蜂群算法[11]首先随机生成含 SN 个解的初始蜂群 $Z=\{Z_1,Z_2,\cdots,Z_{\rm SN}\}$, 每个解 Z_i 是一个 D 维向量; 然后引领蜂根据式 (6) 进行近邻搜索, 当搜索解优于记忆最优解时, 替换记忆解, 反之保持不变, 有

$$v_{ij}=Z_{ij}+\varphi_{ij}(Z_{ij}-Z_{kj}). \tag{6}$$

其中: $j\in\{1,2,\cdots,D\}$; $k\in\{1,2,\cdots,{\rm SN}\}$, k 随机选取且 $k\neq 1$; $\varphi_{ij}\in[-1,1]$ 中的随机数; v_{ij} 为新位置. 随后, 跟随蜂根据轮盘赌方式按下式中的概率选择引领蜂并由式 (6) 进行近邻搜索:

$$P_i=\frac{{\rm fit}_i}{\sum\limits_{n=1}^{{\rm SN}/2}{\rm fit}_n}, \tag{7}$$

其中 ${\rm fit}_i$ 表示第 i 只引领蜂适应度. 最后, 判断引领蜂 Z_i 连续 Limit 次迭代的情况, 若没有变化且没有达到最优解, 则引领蜂变为侦查蜂, 通过下式随机搜索一个新解代替 Z_i:

$$z_i^j=z_{\min}^j+{\rm rand}(0,1)(z_{\max}^j-z_{\min}^j). \tag{8}$$

其中: $j\in 1,2,\cdots,D$; $z_{\min}^j$ 和 $z_{\max}^j$ 表示所有蜜蜂中第 j 维的最小值和最大值.

2.2.2 蜂群初始化的改进

种群初始化在进化算法中严重影响算法的全局收敛速度和解的质量, 而传统的 ABC 算法都是随机初始化蜂群, 可能没有提取到有效信息而分配不均, 导致优良蜂可能没被选到. 对此, 本文提出局部密度和最大最小距离初始化蜂群方法, 具体步骤如下.

Step 1: 计算粗聚类阶段 k' 个代表点以及没归类样本中任意两个点间的全局流形距离 $G_d(x_i,x_j)$ 并记录在矩阵 D 中.

Step 2: 根据式 (2) 和 D 计算任一样本 x_i 的局部密度并按从大到小的顺序排列.

Step 3: 将局部密度最大的样本作为第 1 个初始中心, 再采用最大最小距离得到其他 $k-1$ 个初始中心, 将得到的 k 个初始中心作为一个蜜蜂的位置编码, 并计算其适应度值得到第一个蜜蜂. 反复执行 SN 次, 生成含 SN 个蜜蜂的初始蜂群 $Z=\{Z_1,Z_2,\cdots,Z_{\rm SN}\}$.

2.2.3 适应度函数

适应度函数能对群体的进化方向、进化行为、迭代次数和解的质量起重要的作用, 本文根据局部密度提出一种新的适用度函数, 即

$$\begin{cases} J_i=\sum\limits_{j=1}^{k}\sum\limits_{p\in c_j}|p-z_{ij}|,\\ {\rm fit}_i=\dfrac{{\rm LD}_i}{\sum\limits_{n=1}^{{\rm SN}/2}{\rm LD}_n\cdot J_i}. \end{cases} \tag{9}$$

其中: J_i 表示 i 只蜜蜂的类内距离, z_{ij} 表示第 i 只蜜蜂所表示的第 j 个聚类 c_j 的中心点, p 表示类 c_j 中非聚类中心点, ${\rm LD}_i$ 表示 i 只蜜蜂所表示的 k 个聚类中心的平均局部密度. 聚类中心密度越大, 类内聚集程度越大, 表示越接近最优解, 则解的质量就越好.

2.2.4 局部搜索算子

局部搜索算子是由 Hamzacebi 等[12]提出的, 因其开发能力较强, 收敛速度快, 且搜索步长随着迭代的进行不断减小, 能在后期达到更加精密的搜索,目前已在遗传算法[13]、粒子群算法[14]中都有应用. 本文将其嵌入到 ABC 算法中, 使当前蜜源快速收敛到最优解. 具体算法如下.

Step 1: 设置初始条件. 局部搜索代数 E, 区间最大迭代数 N, 搜索步长上限 $\alpha_k=x_{\rm best}$, $x_{\rm best}=$

x_{current}, $f(x_{\text{best}}) = f(x_{\text{current}})$, epoch $= 0$, $k = 0$.

Step 2: 重置迭代计数器, $n = 0$.

Step 3: 产生随机步长向量 d_x 且满足 $-\alpha_k \leqslant d_x \leqslant \alpha_k$.

Step 4: 更新 epoch, 即 epoch $=$ epoch $+ 1$.

Step 5: $f_{\text{new}} = f(x_{\text{current}} + d_x)$

if $f_{\text{new}} < f_{\text{best}}$, then $f_{\text{best}} = f_{\text{new}}$, $x_{\text{best}} = x_{\text{current}} + d_x$. $n = n + 1$ 并转 Step 7.

if $f_{\text{new}} < f_{\text{current}}$, then $f_{\text{current}} = f_{\text{new}}$, $x_{\text{current}} = x_{\text{current}} + d_x$. $n = n + 1$ 并转 Step 7.

Step 6: $f_{\text{new}} = f(x_{\text{current}} - d_x)$

if $f_{\text{new}} < f_{\text{best}}$, then $f_{\text{best}} = f_{\text{new}}$, $x_{\text{best}} = x_{\text{current}} - d_x$. $n = n + 1$ 并转 Step 7.

if $f_{\text{new}} < f_{\text{current}}$, then $f_{\text{current}} = f_{\text{new}}$, $x_{\text{current}} = x_{\text{current}} - d_x$. $n = n + 1$ 并转 Step 7.

Step 7: if $n < N$, 则转 Step 3.

Step 8: $k = k + 1$.

Step 9: $\alpha_k = \alpha_{k-1} \times 0.5$.

Step 10: 如果 epoch $= E$, 则算法终止, 否则转 Step 2.

其中: epoch 为局部搜索迭代计数器, x_{best} 为目前找到的最优解, f_{best}、f_{new}、f_{current} 为对应解处的适应度值, 一般 $N = E/5$.

2.2.5　改进 ABC 算法的细聚类

1) 蜜蜂编码.

为减少蜜蜂编码长度, 本文采用实数编码方式, 由蜜蜂的位置和适应度组成, 且位置由 k 个聚类中心组成. 编码如下:

$$Z_i = (z_{i1}, z_{i2}, \cdots, z_{ik}, \text{fit}_i).$$

2) 算法步骤.

Step 1: 根据局部密度和距离生成初始蜂群.

Step 2: 根据蜜蜂的适应度反序排序, 前 50% 为引领蜂, 后 50% 为跟随蜂. 每只引领蜂根据式 (6) 作近邻搜索产生新解, 搜索完毕后由式 (7) 计算概率 p_i.

Step 3: 每只跟随蜂按轮盘赌方式根据 p_i 选择引领蜂并由式 (6) 作近邻搜索产生新解.

Step 4: 根据适应度记忆最优蜜源即最优解, 用局部搜索算子对当前蜜源最优位置进行局部搜索, 更新最优位置.

Step 5: Limit 次后判断是否存在没变化的蜜源即解, 若存在, 则该引领蜂变为侦查蜂并按式 (8) 产生一个新解代替.

Step 6: 对每只蜜蜂进行一次聚类, 用得到的新的聚类中心更新蜂群.

Step 7: 若当前迭代次数大于最大次数 MCN, 则停止迭代; 否则, 转 Step 2, 且 $t = t + 1$.

2.3　优化阶段

Step 1: 对数据集粗聚类, 并得出 k' 个簇代表点;

Step 2: 对粗聚类的簇代表点及没归类的样本点细聚类, 得到 k 类;

Step 3: 结合 Step 1 和 Step 2 的聚类结果得到最终聚类结果;

Step 4: 算法停止并输出结果.

2.4　算法时间复杂度分析

在粗聚类阶段: Step 1 中, 计算局部密度并排序的时间复杂度为 $O(n \times K + (n \times K) \times \log(n \times K))$, 其中 K 为近邻点数; Step 2 中, 最大最小距离时间复杂度为 $O(k')$; Step 3 中, 近邻选择时间复杂度为 $O(k_D \times k_d \times k')$, 因为 $k_D \leqslant k_d < K < k' \ll n$, 所以粗聚类阶段总的时间复杂度为 $O(n \times \log n)$.

在细聚类阶段: Step 1 中, 蜂群初始化的时间复杂度为 $O(|k' + M|^3 + (k' + M) \times \log(k' + M) + k \times \text{SN})$, 其中 M 为粗聚类阶段未归类的样本; Step 2 中, 蜜蜂排序及引领蜂邻域搜索的时间复杂度为 $O(\text{SN} \times \log\text{SN} + k \times \text{SN}/2)$; Step 3 中, 跟随蜂邻域搜索时间复杂度为 $O(k \times \text{SN}/2)$; Step 4 中, 局部搜索算子时间复杂度为 $O(E \times N)$; Step 6 中, 每只蜜蜂进行一次聚类的时间复杂度为 $O(k \times n)$, 总共有 SN 只蜜蜂, 则时间复杂度为 $O(k \times n \times \text{SN})$. 因为 $k < k' < \text{SN} \leqslant n$, $E < N \leqslant n$, $M \leqslant n$, 所以细聚类阶段总的时间复杂度可以写为 $O(k \times n \times \text{SN})$.

因为总共循环了 MCN 次, 且 $n < (n \times \log n)$, 所以两阶段总的时间复杂度为 $O(\text{MCN} \times n \times \log n)$.

3　实验结果与分析

实验环境: 操作系统为 Windows7, 集成开发环境为 Eclipse, Matlab R2011b, 硬件为 Intel(R) Core(TM) i5-3210MCPU @2.50 GHz, 内存 4 GB.

算法的第 1 阶段主要将数据集初步归类, 可能会有重复归类的情况, 但每个样本都能正确归类. 在第 2 阶段中采用了全局流形距离, 结果会影响整个算法的性能, 所以算法的聚类效果主要由第 2 阶段体现. 本

表 2　实验中涉及到的数据集

数据集	样本数目	属性维数	类别数
Line-blobs	266	2	3
Sticks	512	2	4
Square1	1 000	2	4
Long1	1 000	2	2
Iris	150	4	3
Wine	178	13	3

表 3 各数据集参数设置

数据集	参数设置							
	ρ	k'	K_d	K_D	SN	k	Limit	MCN
Line-blobs	4.5	40	15	6	90	3	15	200
Sticks	5	60	15	8	100	4	15	200
Square1	5	100	12	7	100	4	15	200
Long1	5	100	8	6	100	2	15	200
Iris	3	20	6	4	100	3	15	200
Wine	2.6	25	6	5	100	3	15	200

文着重对第2阶段进行分析, 并结合两阶段得出最后聚类结果. 文中用到的各数据集特征如表2所示, 经过50次实验, 本文算法聚类效果最好时参数设置如表3所示.

3.1 第2阶段分析

这里以数据集Line-blobs、Sticks和Square1为例进行第2阶段分析. 本文算法对这3个数据集的适应度随迭代次数增加的变化情况如图2所示.

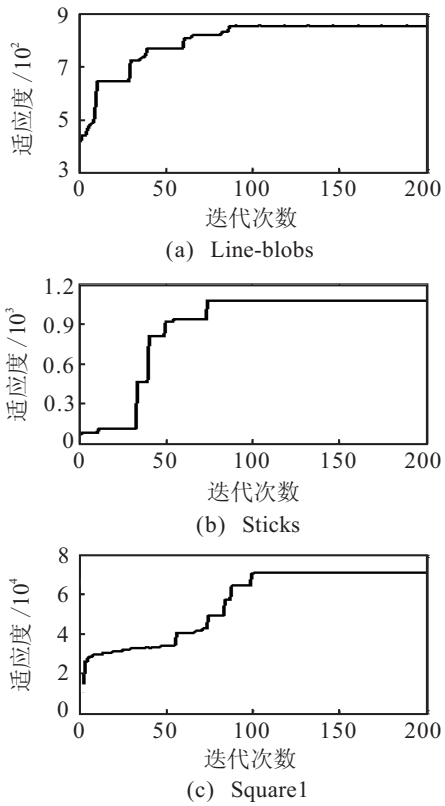


图 2 数据集上适应度变化情况

由图2可知: 在Line-blobs中, 本文算法适应度初值为$4.130\,404\,24\times10^2$, 经过后期的一系列搜索, 算法最终达到全局最优值$8.526\,594\,841\times10^2$, 比初值提高了$4.396\,190\,601\times10^2$; 在Sticks中, 本文算法适应度为$0.061\,763\,503\times10^3$, 经过后期的一系列搜索, 算法最终达到全局最优值$1.068\,244\,958\times10^3$, 比初值提高了$1.006\,481\,455\times10^3$; 在Square1中, 本文算法适应度初值为$1.153\,111\,836\times10^{-4}$, 经过后期的一系列搜索, 算法最终达到全局最优值$7.098\,000\,014\times10^{-4}$, 比初值提高了$5.944\,888\,178\times10^{-4}$. 由此可见, 采用距离和局部密度初始化的方法, 提高了算法搜索起点, 且采用局部搜索算子加快了算法的收敛速度.

同时, 为了表现第2阶段聚类效果和参数对聚类的影响, 本文对这3类数据集的细聚类分布如图3～图5所示.

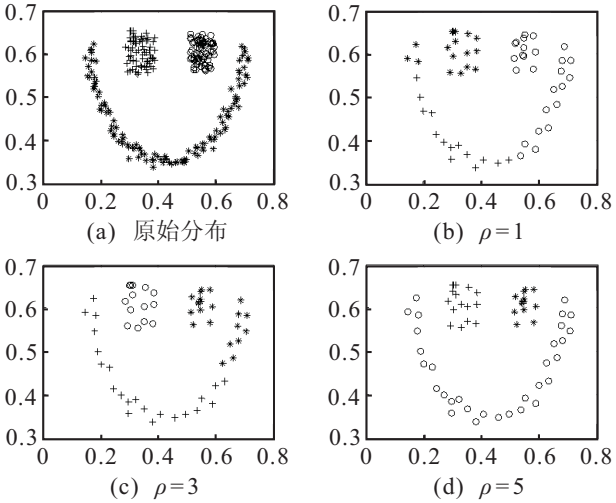


图 3 数据集Line-blobs细聚类

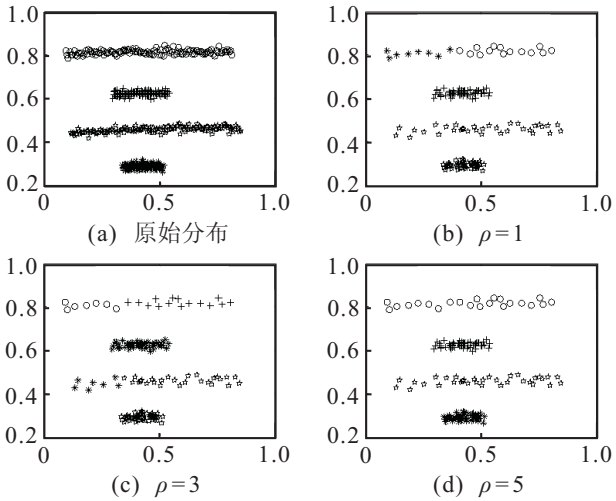


图 4 数据集Sticks细聚类

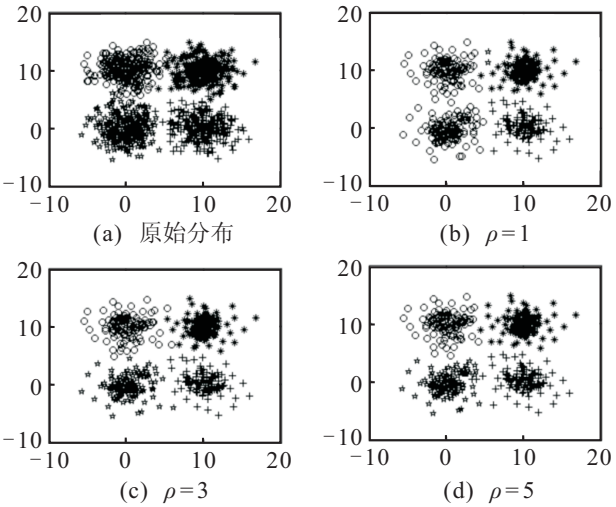


图 5 数据集Square1细聚类

由图 3～图 5 可知, 在细聚类阶段参数设置为 5 时数据集基本能达到最优聚类效果, 因在初步归类时能把大部分数据正确归类, 所以在细聚类时极大减少了样本个数, 使第 2 阶段能有更好的聚类效果.

3.2　两阶段算法对数据集聚类性能综合比较

为验证本文算法总体的有效性和可行性, 对数据集进行准确率和时间的测试. 用到的比较算法有 GAC、QEAM[7]、TPC[6]、TPSC[15]及文献 [7] 中 IEAM. 对实验进行 50 次, 具体结果如表 4、表 5 所示.

由表 4 可知: 在准确率上, 对于数据集 Line-blobs、Sticks 及 Long1, 本文算法和 QEAM 及 TPC 达到最高; 而对于数据集 Square1, 本文算法优于其他算法, 最大达到 99.22%, 平均值比 GAC 高出 4.74%, 比 TPC 高出 0.23%; 对于数据集 Iris 和 Wine, 虽然本文算法不是最高的, 但也优于其他聚类算法, 如 Iris 平均值比 TPSC 高出 1.86% 等, 且每个数据集的标准差均较小, 更进一步说明了本文算法的稳定性.

由表 5 可知: 在时间上, 对于数据集 Line-blobs、Iris 及 Wine, 本文算法所需时间最少, 且远远低于 QEAM 和 IEQM; 在数据集 Sticks、Square1 和 Long1 上, 虽然本文算法所需时间比 TPC 高, 但是远远低于其他算法.

表 4　各算法准确率比较

数据集	本文算法				GAC	QEAM	IEAM	TPC	TPSC
	最小	平均	最大	标准差					
Line-blobs	1	1	1	0	0.751 0	1	1	1	—
Sticks	1	1	1	0	0.739 0	1	1	1	—
Square1	0.979 5	0.989 5	0.992 2	0.004 3	0.942 1	—	—	0.987 2	—
Long1	1	1	1	0	0.525 0	1	0.967 0	1	—
Iris	0.900 5	0.911 6	0.930 0	0.008 7	0.838 0	0.967 0	0.960 0	0.907 7	0.893 0
Wine	0.841 0	0.863 9	0.882 0	0.021 0	0.615 0	0.938 0	0.826 0	—	0.747 0

表 5　各算法时间比较

数据集	本文算法			QEAM	IEAM	TPC	TPSC
	最小	平均	最大				
Line-blobs	0.875 0	0.976 5	1.106 0	52.840 0	99.920 0	1.471 8	—
Sticks	6.912 0	8.002 7	9.860 0	488.470 0	712.160 0	2.146 8	—
Square1	140.230 0	155.625 8	165.860 0	—	—	6.543 4	—
Long1	600.120 0	624.921 4	650.460 0	1 364.710 0	3 659.040 0	4.247 1	—
Iris	0.720 4	0.803 87	0.951 3	21.140 0	38.240 0	1.341 2	1.570 0
Wine	0.862 0	0.988 72	1.124 0	24.790 0	42.900 0	—	2.700 0

综合准确率和所需时间性能比较, 本文因采用近邻选择及改进的流形距离和蜂群算法, 准确率大部分有所提高; 因采用二次聚类算法以及在蜂群中引入局部搜索算子, 大大地减少了运行时间, 保证准确率的同时也缩减了运行时间, 从而有一个较好的折中.

4　参数对算法性能的影响

为验证改进流形距离中伸缩因子对聚类性能的影响, 将 ρ 分别取 0.5、1、1.5、1.8、2、2.5、3、5、15、20 时对数据集 Line-blobs、Sticks 和 Square1 的准确率进行实验对比, 结果如图 6 所示.

由图 6 可知, 本文改进的流形距离中伸缩因子对聚类性能逐渐增加, 且在较短区间内有明显提高, 后来趋向于平缓, 即算法达到稳定. 因此调节的范围较宽, 克服了文献 [5-6] 中提到的参数本身在较大范围内变化时对数据集聚类效果不明显的缺陷.

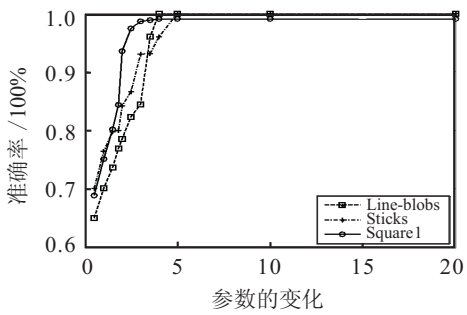


图 6　参数 ρ 对聚类性能的影响

5　结　论

本文提出了一种改进流形距离和人工蜂群的二阶段聚类算法, 在粗聚类中采用局部密度、最大最小距离及近邻选择使数据集能有一个大致的归类, 且没有采用全局流形距离, 减少了所需时间; 在细聚类阶

段采用改进的流形距离和人工蜂群算法细聚类, 因数据相对整个数据集大幅度减少, 故所需时间也相应减少, 在蜂群中采用局部密度和最大最小距离初始化蜂群能得到适应度较大的蜂群, 为后续的寻优奠定了更好的基础. 通过引进局部搜索算子加快了算法的收敛速度, 使算法更快地达到全局最优. 最后综合二阶段聚类结果得到最终的聚类结果, 加快了算法的执行, 且克服了流形距离中参数因子在较大范围内变化时对数据聚类效果不明显的缺陷.

参考文献(References)

[1] Su Mu C, Chou Chien H. A modifled version of the K-means algorithm with a distance based on cluster symmetry[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2001, 23(6): 674-680.

[2] Wang L, Bo Lie F, Jiao Li C. Density-Sensitive semi-supervised spectral clustering[J]. J of Software, 2007, 18(10): 2412-2422.

[3] Wang L, Bo Lie F, Jiao Li C. Density-sensitive spectral clustering[J]. Acta Electronica Sinica, 2007, 35(8): 1577-1581.

[4] Gong Mao G, Jiao Li C, Bo Lie F, et al. Image texture classification using a manifold distance based evolutionary clustering method[J]. Optical Engineering, 2008, 47(7): 077201-1-077201-10.

[5] 公茂果, 焦李成, 马文萍, 等. 基于流形距离的人工免疫无监督分类与识别算法[J]. 自动化学报, 2008, 34(3): 367-375.
(Gong M G, Jiao L C, Ma W P, et al. Unsupervised classification and recognition using an artificial immune system based on manifold distance[J]. Acta Automatica Sinica, 2008, 34(3): 367-375.)

[6] Gong M G, Wang S, Ma M, et al. Two-phase clustering algorithm for complex distributed data[J]. J of Software, 2011, 22(11): 2760-2772.

[7] 李阳阳, 石洪竺, 焦李成, 等. 基于流形距离的量子进化聚类算法[J]. 电子学报, 2011, 39(10): 2343-2347.
(Li Y Y, Shi H Z, Jiao L C, et al. Quantum-inspired evolutionary clustering algorithm based on manifold distance[J]. Acta Electronica Sinica, 2011, 39(10): 2343-2347.)

[8] Pan X Y, Liu F, Jiao L C. Density sensitive based multi-agent evolutionary clustering algorithm[J]. J of Software, 2010, 21(10): 2420-2431.

[9] Wei L, Wang Shou J. Semi-supervised discriminant analysis based on manifold distance[J]. J of Software, 2010, 21(10): 2445-2453.

[10] 吴健, 崔志明, 时玉杰, 等. 基于局部密度构造相似矩阵的谱聚类算法[J]. 通信学报, 2013, 34(3): 14-22.
(Wu J, Cui Z M, Shi Y J, et al. Local density-based similarity matrix construction for spectral clustering[J]. J on Communications, 2013, 34(3): 14-22.)

[11] Zhang C S, Ouyang D T, Ning J X. An artificial bee colony approach for clustering[J]. Expert Systems with Applications, 2010, 37(7): 4761-4767.

[12] Hamzacebi C, Kutay F. Continuous functions minimization by dynamic random search technique[J]. Applied Mathematical Modelling, 2007, 31(10): 2189-2198.

[13] Hamzacebi C. Improving genetic algorithms' performance by local search for continuous function optimization[J]. Applied Mathematics and Computation, 2008, 196(1): 309-317.

[14] 高卫峰, 刘三阳. 一种高效粒子群优化算法[J]. 控制与决策, 2011, 26(8): 1158-1162.
(Gao W F, Liu S Y. An efficient particle swarm optimization[J]. Control and Decision, 2011, 26(8): 1158-1162.)

[15] Qi J K, Ying Q, Li J S, et al. Two-phase spectral clustering based on discretization[J]. Intelligent Human-Machine Systems and Cybernetics(IHMSC), 2013, 15(1): 245-249.

(责任编辑：李君玲)