文章编号: 1001-0920 (2016) 03-0423-06 **DOI:** 10.13195/j.kzyjc.2014.1945

基于自适应学习的演化聚类算法

王 玲, 孙 华

(北京科技大学 a. 自动化学院，b. 钢铁流程先进控制教育部重点实验室，北京 100083)

摘 要: 演化聚类算法 (ECM) 是一种有效的在线聚类算法, 能够根据输入数据实时调整聚类. 但是, 该聚类算法依赖于预先设置的最大距离阈值, 而且对数据输入次序敏感. 针对这些问题, 提出一种基于自适应学习的演化算法 (SALECM), 在无法获取数据先验知识的情况下, 无需人为预先定义参数, 可自适应地调整聚类. 实验结果表明, 与 ECM 相比, SALECM 可提高在线聚类的自适应性能, 也能在一定程度上缓解数据输入次序对算法的影响.

关键词: 演化聚类；在线聚类；数据次序；自适应

中图分类号: TP273 **文献标志码:** A

Evolving clustering method based on self-adaptive learning

WANG Ling, *SUN Hua*

(a. School of Automation and Electrical Engineering，b. Key Laboratory of Advanced Control of Iron and Steel Process of Ministry of Education，University of Science and Technology，Beijing 100083，China. Correspondent：WANG Ling，E-mail: ustbwangling@163.com)

Abstract: The evolving clustering method(ECM) is a kind of efficient online clustering method, adapting the clustering results with new incoming data samples automatically. However, the algorithm not only depends on the predefined maximum distance threshold but also be sensitive to the data order, which are easily misleading if the parameter is not appropriate or the data order is different. For the above problem，an evolving clustering method based on the self- adaptive learning ECM(SALECM) is proposed. Under the situation that no priori knowledge is accessible, it can adjust clusters adaptively without any predefined parameter. Experiment results show that the algorithm can improve online clustering adaptively compared with the ECM and relieve the influence of the input sequence to the ECM to some degree.

Keywords: evolving clustering；online clustering；data order；self-adaptive

0 引 言

随着信息技术的发展, 实际应用要求聚类算法能够对在线数据进行实时分析. 演化聚类算法 (ECM) 是 Ravi 等[1]提出的一种演化的在线聚类算法, 随着输入数据的不断到来, 该算法可实时地增加聚类个数、调整聚类中心和聚类半径. 文献 [2] 将 ECM 首次应用于构建动态模糊神经网络系统, 事实证明 ECM 是一种有效的在线聚类算法. 此外, ECM 在构建动态演化系统方面也有突出的效果, 已广泛用于生物信息学、大脑研究以及智能机器建模等领域[3-7]. 文献 [8] 将 ECM 和模糊 C-均值算法 (FCM) 相结合, 提出一种两阶段的聚类算法. 第 1 阶段利用 ECM 确定初始聚类中心, 第 2 阶段利用 FCM 改进聚类结果. 文献 [9] 提出一种演化模糊聚类算法 (EFCM), 无需预先确定聚类数, 不仅提高了在线聚类的速度, 而且聚类效果优于 ECM 和 FCM.

然而, 不论 ECM 还是 EFCM, 它们均受到最大距离阈值 Dthr 的约束. 由于 Dthr 决定着聚类半径的大小, 若设置不当, 则会直接影响最终聚类结果. ECM 直接将距离阈值设定为一常数. EFCM 采用了等步长的方式改变阈值, 在聚类算法寻优过程中获得了最佳的阈值, 但是人为设置阈值 Dthr 的初始值和步长仍然会影响在线聚类算法的效率和正确率. 此外, 这些算法对数据输入次序敏感, 即使在相同阈值 Dthr 的情况下, 不同的数据输入次序仍然会导致不同的聚类结果.

为了克服参数设置困难和数据输入次序敏感性

收稿日期: 2014-12-23；**修回日期:** 2015-03-10.

基金项目: 国家自然科学基金项目 (61572073)；中央高校基本科研业务费专项资金项目 (FRF-UM-15-052)；高等学校教育教学改革项目 (JG2014Z06).

作者简介: 王玲(1974−), 女, 副教授, 博士, 从事数据挖掘、机器学习的研究；孙华(1990−), 男, 硕士, 从事数据挖掘方法的研究.

的问题, 本文提出一种基于自适应学习的演化聚类算法(SALECM). 首先, 采集若干在线数据集进行预聚类, 通过计算这些数据样本的相似度将最相似的样本分割为一个聚类簇; 然后, 对这些聚类簇进行融合作为初始的聚类结果, 并计算初始聚类簇的聚类半径和中心. 当有新的数据样本输入时, 根据该样本与最近的聚类中心的距离, 以及该聚类半径的关系来判断是否更新现有的聚类. 当不再有新数据输入时, 为了避免聚类冗余或生成过大的聚类簇, 采用聚类簇分割-融合方法对聚类结果进行调整以消除算法对数据输入次序的敏感性. 实验仿真表明, 与ECM相比, SALECM可提高在线聚类的自适应性能, 也可在一定程度上缓解数据输入次序对算法的影响.

1 演化聚类算法

演化聚类算法是一种演化的、在线的、受到某一最大距离约束的聚类算法[1]. 随着新数据的到来, 可动态地增加聚类个数或调整已有聚类中心以及聚类半径, 算法的具体描述如下.

Step 1: 创建第1个聚类$\mathrm{cluster}_1$. 将第1个数据点$\boldsymbol{x}_1$作为第1个聚类中心$\boldsymbol{C}_1$, 并令聚类半径$r_1=0$.

Step 2: 计算新输入数据$\boldsymbol{x}_i$与聚类中心$\boldsymbol{C}_j$之间的欧氏距离$d_{ij}=\|\boldsymbol{x}_i-\boldsymbol{C}_j\|$, $j=1,2,\cdots,m$, 其中m为聚类个数.

Step 3: 如果$d_{ij}\leqslant r_j$, $j=1,2,\cdots,m$, 且$\boldsymbol{C}_j=\arg\min\limits_{j=1,2,\cdots,m}(\|\boldsymbol{x}_i-\boldsymbol{C}_j\|)$, 则$\boldsymbol{x}_i\in\mathrm{cluster}_j$, 返回Step 2, 否则跳转到Step 4.

Step 4: 如果$d_{ij}>r_j$, $j=1,2,\cdots,m$, 则选出一个聚类中心$\boldsymbol{C}_j$, $j=1,2,\cdots,m$, 进而计算$\boldsymbol{C}_j$和$\boldsymbol{x}_i$之间的距离d_{ij}与相应聚类半径r_j这两项之和, 以获得最小的S_{ij}, 即

$$S_{ij}=d_{ij}+r_j=\min_{j=1,2\cdots,m}(S_{ij}). \tag{1}$$

Step 5: 如果$S_{ij}>2\times\mathrm{Dthr}$(Dthr为预先人为给定的阈值), 则$\boldsymbol{x}_i$不属于任何已有的聚类. 采用Step 1的方法创建一个新的聚类, 如果仍有新数据输入, 则返回Step 2.

Step 6: 如果$S_{ij}\leqslant 2\times\mathrm{Dthr}$, 则$\boldsymbol{x}_i\in\mathrm{cluster}_j$, 并按照以下公式更新$\mathrm{cluster}_j$的聚类半径:

$$r_j^{\mathrm{new}}=S_{ij}/2. \tag{2}$$

将新的聚类中心$\boldsymbol{C}_j^{\mathrm{new}}$移动至位于$\boldsymbol{x}_i$和$\boldsymbol{C}_j$的连线上, 使其满足$\left\|\boldsymbol{C}_j^{\mathrm{new}}-\boldsymbol{x}_i\right\|=r_j^{\mathrm{new}}$. 如果仍有新数据输入, 则返回Step 2.

2 基于自适应学习的演化聚类算法

通过对ECM的描述可知, 在无法获取数据集先验知识的情况下很难预先定义阈值Dthr, 而阈值Dthr设置不当将影响最终聚类效果, 而且ECM对数据输入次序敏感. 为此, 提出一种自适应聚类算法, 无需人为设定参数, 可实现自适应在线聚类, 并且能克服数据输入次序对算法的影响.

2.1 算法的相关知识

定义 1 聚类中心$\boldsymbol{C}_k$即第k个聚类中心. 每个聚类中心等于属于同一聚类簇中所有样本的平均值, 即

$$\boldsymbol{C}_k=\sum_{i=1}^{n_k}\boldsymbol{x}_i\Big/n_k. \tag{3}$$

其中: n_k为第k个聚类中样本个数, $\boldsymbol{x}_i(i=1,2,\cdots,n_k)$为属于该聚类的样本.

定义 2 聚类半径r_k即第k个聚类半径, 表示第k个聚类的聚类中心与该聚类簇中样本间距离最大值的一半. 假设第k个聚类簇为

$$\mathrm{cluster}_k=\{\boldsymbol{x}_i;i=1,2,\cdots,n_k,\boldsymbol{x}_i\in\boldsymbol{R}^p\},$$

聚类中心为$\boldsymbol{C}_k$, 则有

$$r_k=\frac{1}{2}\max_{i=1,2,\cdots,n_k}\{d(\boldsymbol{x}_i,\boldsymbol{C}_k)\}. \tag{4}$$

定义 3 假设有两个数据样本$\boldsymbol{x}_i$和$\boldsymbol{x}_j$, 令$\mathrm{Sim}(\boldsymbol{x}_i,\boldsymbol{x}_j)$表示样本$\boldsymbol{x}_i$和$\boldsymbol{x}_j$间相似度[10], 即

$$\mathrm{Sim}(\boldsymbol{x}_i,\boldsymbol{x}_j)=\frac{\sum\limits_{d=1}^{p}x_{id}\bigcap x_{jd}}{\sum\limits_{d=1}^{p}x_{id}\bigcup x_{jd}}. \tag{5}$$

其中: $\bigcap$表示取较小值, $\bigcup$表示取较大值.

定义 4 戴维森保丁指数(DBI)是一种评估准则[11], 以类间离散度和类内离散度为基本依据, 同一聚类中数据相似程度越高以及聚类之间相异性越高, 则DBI越小, 意味着聚类质量越好. 其计算公式如下:

$$\mathrm{DBI}=\frac{1}{m}\sum_{i=1}^{m}\max_{\substack{j\neq i,\\ j=1,2,\cdots,m}}\left\{\frac{S_i+S_j}{d(\boldsymbol{C}_i,\boldsymbol{C}_j)}\right\}. \tag{6}$$

其中: m为聚类总数; $d(\boldsymbol{C}_i,\boldsymbol{C}_j)$等于聚类中心$\boldsymbol{C}_i$和$\boldsymbol{C}_j$之间的欧氏距离; S_i表示第i个聚类中样本与$\boldsymbol{C}_i$的标准误差, 即

$$S_i=\sqrt{\frac{1}{n_i}\sum_{j=1}^{n_i}|\boldsymbol{x}_j-\boldsymbol{C}_i|^2}, \tag{7}$$

这里n_i为第i个聚类中样本数.

定义 5 为了避免聚类结果中存在冗余或者过大的聚类簇并消除算法对数据输入次序的敏感性, 提出一种新的聚类簇分割-融合方法. 当在线数据输入完毕后, 根据聚类数判断是对聚类簇进行分割或是融合, 分别计算聚类结果进行调整前后的聚类质量指标DBI, 根据聚类质量指标判断是否更新聚类结果, 直到聚类数收敛时算法停止.

1) 分割方法的具体步骤如下.

Step 1: 将有最多样本的聚类簇作为$\text{cluster}_{\text{win}}$.

Step 2: 按照以下公式计算邻域半径:

$$\text{Eps} = \min_{i=1}^{n}\left(\max_{j=1;j\neq i}^{n}(d(\boldsymbol{x}_i,\boldsymbol{x}_j))\right). \tag{8}$$

其中: n为$\text{cluster}_{\text{win}}$内样本的总数; $\boldsymbol{x}_i,\boldsymbol{x}_j \in \text{cluster}_{\text{win}}$.

Step 3: 按照以下公式计算该聚类簇中其余样本的密度:

$$\begin{aligned}&\text{density}_i = \text{num}(\boldsymbol{x}_i);\\ &\text{s.t.}\quad d(\boldsymbol{x}_i,\boldsymbol{x}_j) < \text{Eps}.\end{aligned} \tag{9}$$

其中: $\text{num}(\boldsymbol{x}_i)$表示样本$\boldsymbol{x}_i$在其邻域半径Eps范围内相邻的样本数, 记为样本$\boldsymbol{x}_i$的密度.

Step 4: 将$\text{cluster}_{\text{win}}$中密度最大的样本作为分割后的第一个聚类中心$\boldsymbol{C}_{\text{new}}^{(1)}$, 并按照以下公式得到第 2 个聚类中心

$$\begin{aligned}&\boldsymbol{C}_{\text{new}}^{(2)} = \boldsymbol{x}_i = \arg\max\{d(\boldsymbol{x}_i,\boldsymbol{C}_{\text{new}}^{(1)})*\text{density}_i\},\\ &\boldsymbol{x}_i \in \text{cluster}_{\text{win}}.\end{aligned} \tag{10}$$

Step 5: 计算$\text{cluster}_{\text{win}}$中其余样本距离这两个新聚类中心的距离, 如果某一样本距离$\boldsymbol{C}_{\text{win}}^{(1)}$更近, 则将其归类到$\text{cluster}_{\text{new}}^{(1)}$中, 否则该样本属于$\text{cluster}_{\text{new}}^{(2)}$.

Step 6: 计算在执行以上分割操作步骤前后的聚类质量指标DBI和DBI′, 如果DBI >DBI′, 则执行分割.

2) 融合方法的具体步骤如下.

Step 1: 首先找到当前聚类数最少的聚类簇记为$\text{cluster}_{\text{win}}^{(1)}$, 以及距离其最近的聚类簇$\text{cluster}_{\text{win}}^{(2)}$.

Step 2: 将这两个聚类簇合并为一个聚类簇并计算新的聚类簇的中心和半径.

Step 3: 计算执行以上合并操作前后的聚类质量指标DBI和DBI′, 如果DBI >DBI′, 则执行聚类簇的融合.

2.2 SALECM 步骤

SALECM的具体步骤如下.

输入: 随时间变化不断到来的数据样本;

输出: 目标聚类簇集合, 每个聚类簇的中心和半径.

Step 1　首先在线地采集t个数据样本, 并对这t个样本进行预聚类. 如果数据集满足$4p^2 < n$, 其中p为数据的维度, 则$t = \max(4p^2, 50)$[12], 否则令$t = 50$.

Step 2　预聚类.

Step 2.1: 按照小数定标规范化公式对数据进行归一化预处理, 即

$$x_{id} = x_{id}/10^l,\ d = 1, 2, \cdots, p. \tag{11}$$

其中: x_{id}为样本$\boldsymbol{x}_i$的d维数据, l为能够使$x_{id}/10^l < 1$的最大正整数.

Step 2.2: 根据式(5)计算样本间的相似度, 将互为最相似的样本两两结合作为聚类簇, 并根据式(3)计算每个聚类簇的中心, 同时根据式(4)自适应地计算聚类半径.

Step 2.3: 对于已存在的任何两个聚类i, j, 如果它们的聚类中心的距离$d(\boldsymbol{C}_i,\boldsymbol{C}_j) \leqslant 2\times\max(r_i, r_j)$, 则将这两个聚类簇合并为一个新的聚类簇, 并重新计算其聚类中心和半径. 一直重复这一步骤, 直到聚类数不再变化时为止, 并继续执行以下步骤.

Step 3　在线聚类.

Step 3.1: 假设新输入数据样本为$\boldsymbol{x}_{t+1}$, 首先根据式(11)对其进行归一化处理.

Step 3.2: 将所有已有聚类中心中距离$\boldsymbol{x}_{t+1}$最近的聚类中心记为$\boldsymbol{C}_{\text{win}}$, 如果满足$d(\boldsymbol{C}_{\text{win}},\boldsymbol{x}_{t+1}) \leqslant 2\times r_m$, $1 \leqslant m \leqslant k$($k$为已经存在的聚类个数)或者$\boldsymbol{C}_{\text{win}}$对应的聚类中只有一个数据样本, 则$\boldsymbol{x}_{t+1}$属于该聚类, 并更新其聚类中心和半径, 不断重复这个过程, 直到不满足条件; 否则, 创建一个新的聚类, 令聚类数$k = k+1$, 新的聚类中心$\boldsymbol{C}_k = \boldsymbol{x}_{t+1}$, 聚类半径$r_k = 0$.

Step 4　聚类簇的分割与融合.

首先, 根据当前聚类数判断是对当前聚类簇进行分割还是融合, 如果$k < 3$, 则执行Step 4.1, 否则执行Step 4.2. 但不管是对聚类簇做进一步的分割还是融合, 都需要判断聚类簇执行分割或融合前后的聚类质量指标(令执行前后的聚类质量指标分别为DBI和DBI′), 如果DBI >DBI′, 则表明聚类簇分割或融合后是可行的.

Step 4.1: 分割. 找出具有最多样本的聚类簇. 根据定义 5 中的分割方法对该聚类簇进行分割, 并计算分割后两个新聚类簇的中心和半径, 令聚类数$k = k+1$.

Step 4.2: 融合. 找出当前聚类簇中样本数最少的聚类簇以及距离其最近的聚类簇, 将这两个聚类簇融合为一个聚类簇, 并重新计算其聚类中心和半径, 令聚类数$k = k-1$.

3　实验结果与分析

为了验证SALECM的有效性, 以表 1 中的数据为实验数据, 以聚类正确率和目标函数值两个评价标准对比SALECM和其他算法[13]的性能. 其中算法的正确率[14]计算公式如下:

$$\text{Accuracy} = \frac{\sum_{m=1}^{k} a_m}{n}, \tag{12}$$

其中 a_m 表示第 m 个聚类簇中算法聚类结果和实际聚类相一致的样本个数. 由式(12)可知, 分子部分的 $a_m(m = 1, 2, \cdots, k)$ 值越大说明聚类正确的样本数越多, 则聚类正确率越高. 目标函数为

$$J = \sum_{i=1}^{k}\sum_{j=1}^{n}(\mu_{ij} \times \|\boldsymbol{x}_j - \boldsymbol{C}_i\|)^2, \tag{13}$$

其中 μ_{ij} 表示样本 $\boldsymbol{x}_j$ 对聚类中心 $\boldsymbol{C}_i$ 的隶属度. 由于聚类的目标是保证最终每个聚类簇内样本与聚类中心距离尽量小, 式(13)的目标值越小表示聚类效果越好.

表 1 实验数据

数据集	样本数	聚类数	属性
Iris	150	3	4
Wine	178	3	13
Seed	210	3	7
Breast Cancer	683	2	9
Dataset3_2	300	3	2

3.1 阈值对 ECM 的影响

ECM 需要获取数据先验知识或者经过反复实验才能选出最佳阈值, 如果阈值设置不当, 则将无法得到正确聚类数. 表 2 给出了 ECM 对于不同阈值得到的聚类数. 对于 Iris 数据集, 当阈值 Dthr 分别设置为 0.5 和 0.75 时, ECM 得到的聚类数分别为 7 和 4, 当阈值设置为 1.1 时, 最终得到正确的聚类数. 对于 Seed、Breast Cancer 和 Dataset 3_2 数据集也同样如此, 阈值的改变都会影响最终的聚类结果. 对比这 5 个数据集可知, 不同数据集阈值的取值范围也不同. 其中: Iris、Seed 和 Dataset3_2 阈值的取值范围是 (0, 2]; 而 Wine 数据集阈值取值范围是 (0, 90]; Breast Cancer 数据集阈值取值范围是 (0, 4]. 由此可知, 在无法获得先验知识的情况下, ECM 难以确定阈值, 需要大量反复的实验试凑.

3.2 聚类正确率

根据数据原始输入次序, 各个算法的聚类正确率如表 3 所示 (其中: #ac 表示聚类正确率, #c 表示聚类数). 由表 3 可见, 除 Seed 数据集以外, SALECM 对其他数据集的聚类正确率均要高于其余 5 个算法. SALECM 对 Seed 数据集的聚类正确率为 89.0 %, 虽然低于 FCM、KM 和 SC 算法, 但仍高于 ECM 和 EFCM 的正确率. 除此之外, 还可以看出 SALECM 对各个数据集的聚类数均正确.

当数据输入次序打乱后, 各个算法的聚类正确率如表 4 所示. 从表 4 可以看出, ECM 和 EFCM 都对数据输入次序敏感, SALECM 的正确率仍优于其他算法. SALECM 对这 5 个打乱次序数据集的聚类正

表 2 不同阈值对 ECM 的影响

数据集	Iris			Wine			Seed			Breast Cancer			Dataset3_2		
阈值 Dthr	0.5	0.75	1.1	50	55	90	0.9	1.4	1.8	3	3.5	4	0.5	0.75	1
聚类数	7	4	3	6	5	4	9	4	3	10	5	2	16	9	3

表 3 各个算法对原始输入次序数据的聚类正确率

Dataset	ECM		EFCM		FCM		KM		SC		SALECM	
	#ac/%	#c	#ac/%	#c	#ac/%	#c	#ac/%	#c	#ac/%	#c	#ac/%	#c
Iris	86.0	3	81.3%	3	87.1%	3	89.3	3	89.3	3	91.3	3
Wine	<60	4	<60	3	<60	3	70.2	3	60.6	4	91.0	3
Seed	85.7	3	88.9	3	90.0	3	92.4	3	95.2	3	89.0	3
Breast	87.2	2	90.1	3	94.7	2	96.0	2	72.3	3	96.5	2
Dataset3_2	100	3	100	3	100	3	100	3	100	3	100	3

表 4 各个算法对打乱输入次序数据的聚类正确率

Dataset	ECM		EFCM		FCM		KM		SC		SALECM	
	#ac/%	#c	#ac/%	#c	#ac/%	#c	#ac/%	#c	#ac/%	#c	#ac/%	#c
Iris	62.0	3	81.3	3	89.3	3	89.3	3	89.3	3	90.7	3
Wine	<60	3	<60	3	68.3	3	86.5	3	61.8	4	96.1	3
Seed	66.7	3	80.0	4	89.5	3	65.7	3	67.6	3	90	3
Breast	72.7	2	79.7	4	95.6	2	96.0	2	72.3	3	96.5	2
Dataset3_2	100	3	100	3	100	3	100	3	100	3	100	3

确率均大于90%, 远高于ECM和EFCM. 特别是, 对于打乱次序的Wine和Breast Cancer数据集, ECM和EFCM的正确率损失较多. 除此之外, 当数据输入次序打乱了之后, SALECM仍可得到每个数据集正确的聚类数. 由此可知, SALECM对数据输入次序不敏感, 而且与ECM和EFCM相比, 具有较高的自适应性.

数据按照原始次序输入时, 由式(13)计算可得各个算法的目标函数如表5所示. 从表5可以看出, 除Wine和Seed数据集以外, SALECM对其余数据集的聚类目标函数均低于其余算法. SALECM对按照原始输入次序的Wine数据集的聚类目标函数值高于SC算法, 但仍然低于其余4个算法. 对于Seed数据集的聚类目标函数值也有类似的结论.

表 5　各个算法对原始输入次序数据集的聚类目标函数值

Dataset	ECM	EFCM	FCM	KM	SC	SALECM
Iris	5.45	5.53	5.45	5.28	5.28	5.13
Wine	36.44	38.07	57.37	31.09	23.85	23.91
Seed	15.84	33.47	16.44	15.64	19.49	15.65
Breast	185.97	219.29	184.39	185.02	243.80	184.21
Dataset3_2	4.43	4.43	4.43	4.43	4.43	4.43

当数据输入次序打乱后, 各个算法对每个数据集的聚类目标函数值如表6所示. 由表6可以看出, SALECM对各个数据集聚类目标函数值的变化情况比其余5个算法稳定. 特别是, 针对打乱输入次序的Breast Cancer数据集, SALECM得到的目标函数值为184.21, 要远远低于ECM和EFCM的目标函数值. 由此可知, SALECM对数据输入次序的敏感度要低于ECM和EFCM.

表 6　各个算法对打乱输入次序数据集的聚类目标函数值

Dataset	ECM	EFCM	FCM	KM	SC	SALECM
Iris	6.42	5.53	5.45	5.28	5.28	5.25
Wine	36.44	39.41	57.37	29.10	24.31	23.10
Seed	16.06	42.01	16.44	31.85	17.50	15.60
Breast	262.12	275.16	184.39	185.0	243.8	184.21
Dataset3_2	4.43	4.43	4.43	4.43	4.43	4.43

3.3　算法复杂度分析

SALECM主要由预聚类、在线聚类、聚类分割与融合3部分算法组成. 下面分别讨论这3部分算法的时间复杂度.

预聚类算法中, 每次只将最相似的样本合并作为聚类簇, 因此如果要将预收集的t个样本两两结合作为聚类簇, 则计算量为t. 对聚类中心和聚类半径计算量为$2t$. 计算聚类之间的距离, 需要的计算量为$0.5t$, 同时重新计算聚类中心和聚类半径, 计算量为t, 因此总的计算量为$t+2t+0.5t+t=4.5t$.

在线聚类算法中, 只需要重新计算新样本与已存在k个聚类中心的距离, 其计算量为k, 同时重新计算聚类中心和聚类半径, 计算量为$2k$, 因此总的计算量为$k+2k=3k$.

聚类分割-融合算法包括以下两个过程.

1) 分割过程. 计算$\mathrm{cluster}_{\mathrm{win}}$邻域半径的计算量为$n_{\mathrm{win}}$($\mathrm{cluster}_{\mathrm{win}}$类内的样本个数), 样本密度计算量为$n_{\mathrm{win}}$, 求出第2个聚类中心计算量为$n_{\mathrm{win}}-1$, 计算$\mathrm{cluster}_{\mathrm{win}}$中其余样本距离这两个新聚类中心的距离的计算量为$2(n_{\mathrm{win}}-2)$. DBI分割前计算量为

$$k'-1+\sum_{i=1}^{k'-1}n_i.$$

其中: $k'-1$为分割前的聚类个数, n_i为第i个聚类内的样本数. DBI分割后计算量为

$$k'+\sum_{i=1}^{k'}n_i.$$

因此分割过程总的计算量为

$$n_{\mathrm{win}}+n_{\mathrm{win}}+n_{\mathrm{win}}-1+2(n_{\mathrm{win}}-2)+$$
$$k'-1+\sum_{i=1}^{k'-1}n_i+k'+\sum_{i=1}^{k'}n_i=$$
$$5n_{\mathrm{win}}+2k'+\sum_{i=1}^{k'}n_i+\sum_{i=1}^{k'-1}n_i-6.$$

2) 合并过程. 融合后重新计算聚类中心和半径, 计算量为$2n_{k''}$, $n_{k''}$为融合后的第k''聚类内的样本数, DBI融合前计算量为

$$k''+\sum_{i=1}^{k''}n_i,$$

n_i为第i个聚类内的样本数. DBI融合后的计算量为

$$k''-1+\sum_{i=1}^{k''-1}n_i.$$

因此融合过程总的计算量为

$$2n_{k''}+k''+\sum_{i=1}^{k''}n_i+k''-1+\sum_{i=1}^{k''-1}n_i=$$
$$2k''+2n_{k''}+\sum_{i=1}^{k''}n_i+\sum_{i=1}^{k''-1}n_i-1.$$

综上所述, SALECM的时间复杂度为

$$\Omega=4.5t+3k+5n_{\mathrm{win}}+2k'+\sum_{i=1}^{k'}n_i+\sum_{i=1}^{k'-1}n_i+$$
$$2k''+2n_{k''}+\sum_{i=1}^{k''}n_i+\sum_{i=1}^{k''-1}n_i-7.$$

由此看出, 该算法具有近似线性时间复杂度, 执行效率很高.

图1比较了3种算法对不同数据集聚类所需要的平均时间. 从图1可以看出, SALECM的速度要快于其余两种算法. 这是由于ECM和EFCM在优化阈值参数Dthr时需要花费大量的时间. 例如: 对于

Iris数据集, SALECM的平均聚类时间仅为0.20 s, 而ECM和EFCM分别需要0.29 s和0.56 s. 虽然对于其余数据集, SALECM所需时间多于Iris数据集所需时间, 但SALECM的速度均快于ECM和EFCM.

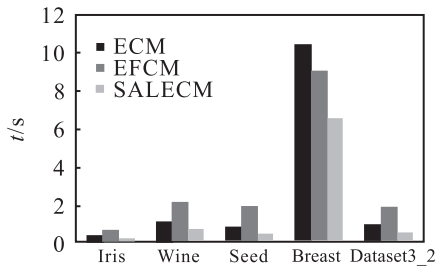

图 1 各个算法的平均运行时间

4 结 论

SALECM在无需预先确定聚类数和阈值参数的情况下可自适应地完成聚类. 该算法对若干数据样本进行预聚类得到初始的聚类结果. 当有新的数据样本输入时, 根据其与最近聚类中心的距离和该聚类半径的关系来判断是否新增聚类. 为了避免冗余或生成过大的聚类簇, 采用一种聚类簇分割-融合方法对聚类结果进行调整, 以消除算法对数据输入次序的敏感性. 实验结果表明, SALECM的自适应性优于ECM, 而且对数据输入次序不敏感.

参考文献(References)

[1] Ravi V, Srinivas E R, Kasabov N K. On-line evolving fuzzy clustering[C]. Int Conf on Computational Intelligence and Multimedia Applications. Sivakasi: IEEE, 2007, 1: 347-351.

[2] Song Q, Kasabov N. Dynamic evolving neuro-fuzzy inference system: On-line learning and application for time-series prediction[C]. Proc of 6th Int Conf on Soft Computing. Kobe, 2000: 696-701.

[3] Kasabov N K. Evolving connectionist systems: Methods and applications in bioinformatics, brain study and intelligent machines[M]. Berlin Heidelberg: Springer, 2003: 39-49.

[4] 张烃, 刘建成, 李树旺. 一种基于演化聚类的动态TSK模型建模方法[J]. 计算机测量与控制, 2006, 14(4): 528-529.
(Zhang J, Liu J C, Li S W. Modeling approach of dynamic tsk model based on evolving clustering method[J]. Computer Measurement & Control, 2006, 14(4): 528-529.)

[5] Kasabov N K, Song Q. Dynamic evolving neural-fuzzy inference system and its application for time-series prediction[J]. IEEE Trans on Fuzzy Systems, 2002, 10(2): 144-154.

[6] Widiputra H, Kho H, Pears R, et al. A novel evolving clustering algorithm with polynomial regression for chaotic time-series prediction[C]. Neural Information Processing. Heidelberg: Springer, 2009: 114-121.

[7] Widiputra H, Pears R, Kasabov N. Multiple time-series prediction through multiple time-series relationships profiling and clustered recurring trends[C]. Advances in Knowledge Discovery and Data Mining. Heidelberg: Springer, 2011: 161-172.

[8] 杜根远, 田胜利, 苗放. 结合ECM和FCM聚类的遥感图像分割新方法[J]. 计算机应用研究, 2009, 26(10): 3995-3997.
(Du G Y, Tian S L, Miao F. Remote sensing image segmentation based on evolving clustering and fuzzy C-means[J]. Application Research of Computers, 2009, 26(10): 3995-3997.)

[9] Lekova A K, Dimitrova M I. Hand gestures recognition based on lightweight evolving fuzzy clustering method[C]. 2013 IEEE 2nd Int Conf on Image Information Processing. Shimla: IEEE, 2013: 505-510.

[10] 金阳, 左万利. 一种基于动态近邻选择模型的聚类算法[J]. 计算机学报, 2007, 30(5): 756-762.
(Jing Y, Zuo W L. A clustering algorithm using dynamic nearest neighbors selection model[J]. Chinese J of Computers, 2007, 30(5): 756-762.)

[11] Arbelaitz O, Gurrutxaga I, Muguerza J, et al. An extensive comparative study of cluster validity indices[J]. Pattern Recognition, 2013, 46(1): 243-256.

[12] Lughofer E. Flexible fuzzy Inference system: A robust incremental learning approach for evolving TS fuzzy models[J]. IEEE Trans on Fuzzy Systems, 2008, 16(6): 1393-1410.

[13] Rashmi Dutta Baruah, Plamen Angelov. Clustering as a tool for self-generation of intelligent systems: A survey[C]. Proc of Int Conf on Evolving Intelligent Systems. Leicester, 2010: 34-41.

[14] Sun Y, Zhu Q M, Chen Z X. An iterative initial-points refinement algorithm for categorical data clustering[J]. Pattern Recognition Letters, 2002, 23(7): 875-884.

(责任编辑：滕 蓉)