中国科学: 信息科学 2013 年 第 43 卷 第 5 期: 599–610
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS

论 文

一种基于同步动力学模型的层次聚类方法

黄健斌[①]*, 康剑梅[①], 齐俊杰[①], 孙鹤立[②]

① 西安电子科技大学软件学院, 西安 710071
② 西安交通大学计算机科学与技术系, 西安 710049
* 通信作者. E-mail: jbhuang@xidian.edu.cn

收稿日期: 2012–02–20; 接受日期: 2012–11–05
国家自然科学基金 (批准号: 61173093, 61202182)、中国博士后科学基金 (批准号: 2012M521776) 和中央高校基本科研业务费 (批准号: K5051323001, 2012jdhz07) 资助项目

摘要 本文基于建模同步动力学行为的 Kuramoto 模型提出了一种新的有效层次聚类方法. 本文提出的方法基于局部邻域的概念, 能够实现稳定的局部同步聚类. 通过不断扩大对象同步的邻域半径, 所提出的方法能够实现层次化的同步聚类. 此外, 提出对象邻域闭包的概念, 在对象间到达完全同步之前就能预测出聚类的形成, 从而减少对象动态交互的时间. 本文的方法不依赖于任何数据分布假设, 无需任何手工参数设置, 可以检测出任意数量、形状和大小的聚类. 由于同步过程能够有效地规避离群点, 该方法有较强的噪声数据抑制能力. 在大量真实数据集和人工合成数据集上的实验结果表明本文的方法聚类准确率高, 且运行时间较同类基准算法显著缩短.

关键词 层次聚类 同步动力学模型 邻域闭包 离群点检测 无参数 轮廓宽度标准

1 引言

同步现象在自然界中广泛存在 [1], 例如: 萤火虫同步发光现象, 起搏细胞有节律的同步收缩等. 目前, 同步现象在物理学界已得到深入研究, 并提出了一些能有效捕捉同步动力过程的模型 [2], 例如: 广义 Kuramoto 模型 [3~5]. 受同步现象启发, Böhm 等 [6] 提出了一种基于同步原理的聚类算法 Sync, 利用同步动力模型来探测数据集中的聚类. 给定一个邻域半径, 一个对象在以自身为圆心的一个超球形邻域内的所有邻居对象的同步作用下产生位移. 这样, 在非线性作用力的影响下, 相近的对象将会同步达到相同的相位并形成聚类. Sync 聚类算法能很好地发现数据集中任意形状和大小的聚类, 并且能够有效识别噪声数据对象. 但是, 该算法的运行时间较长, 难以处理大规模的数据. 这主要是因为对于特定的同步邻域半径该方法只有在一组数据到达基本完全同步时才能识别出聚类结果. 此外, 由于需要尝试不同的邻域半径并选择一个最优的邻域半径参数值, 聚类过程中需要从数据的最初状态开始重复运行很多次.

据此, 本文提出了一种基于改进同步过程的层次聚类算法 SHC(synchronization-based hierarchical clustering). 本文的主要创新点如下: 1) 不同于 Sync 算法, 本文提出的同步聚类算法基于邻域半径的局部同步原理; 2) 本文提出邻域闭包的概念, 并利用邻域闭包预先预测出对象的同步趋势, 在对象到达完全同步之前检测出聚类, 从而显著减少对象的动态交互时间; 3) 本文提出的 SHC 算法能够在前

引用格式: 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, doi: 10.1360/112011-611

一次聚类的基础上, 通过逐步扩大同步邻域半径来进行层次化同步, 从而实现凝聚层次聚类; 4) 结合聚类质量评价准则函数, 本文的算法无需用户干预可以自动发现最佳聚类结果.

本文以下部分内容组织如下: 第 2 节综述了相关工作; 第 3 节介绍同步聚类模型, 描述了重定义的 Kuramoto 模型; 第 4 节详细叙述本文提出的层次同步聚类算法 SHC 的工作原理和过程; 第 5 节给出了实验结果和分析; 第 6 节总结全文并给出下一步的工作方向.

2 相关工作

聚类是挖掘数据中内在群组结构的一种重要方法. 传统的聚类方法可以分为基于划分的方法、基于密度的方法、层次聚类方法、基于图论的方法等不同种类.

K-Means 是一种经典的基于划分的聚类算法, 最终目标是得到紧凑且独立的聚类. 但 K-Means 算法需手工设置聚类个数 k、结果依赖于初始聚类中心的选择、容易陷入局部最优解、对噪声数据敏感、难以发现非凸形状的簇等. 对 K-Means 算法的研究已经非常深入, 并提出了很多改进的算法, 例如: X-Means[7] 等. DBSCAN[8] 是一种代表性的基于密度的聚类算法, 可在含噪声的空间数据集中快速发现密度超过给定阈值的任意形状聚类. 但是, 它把参数 Eps 和 MinPts 的设置任务留给用户, 且算法对参数 Eps 较为敏感. Meanshift[9] 是一种概率密度梯度估计方法, 其优点是用概率密度梯度代替具体的概率密度, 但是需要用户手工输入带宽阈值且算法的运行结果带有一定的随机性. 最近, Sun 等 [10] 提出了一种无需 Eps 参数设置的基于密度聚类算法 gSkeletonClu, 它通过自动检测核连通分量来自动发现基于密度的聚类.

聚类问题目前仍然是数据挖掘领域的研究热点之一, 不断有新的方法涌现. 基于图论的 Chameleon 聚类算法 [11] 将矢量数据建模为图, 通过引入互连性和近似性两个指标来控制簇的分裂和合并, 可以发现高质量的任意形状聚类. 2007 年, Frey 等 [12] 提出了基于近邻传播的聚类方法 AP(affinity propagation), 利用数据点相似度矩阵进行聚类. 2010 年, Böhm 等将建模同步过程的物理模型引入聚类, 提出了一种同步聚类算法 Sync. 该方法无需用户干预, 能够自动发现数据中的任意大小、形状和密度的聚类, 且有较强的噪声数据抑制能力, 但是聚类时间较长. 与 Sync 算法不同, 本文的方法是一种层次聚类方法且基于局部同步原理, 此外同步收敛条件得到优化.

3 同步聚类模型

本节首先介绍建模同步过程的广义 Kuramoto 模型, 然后介绍利用同步模型发现数据聚类的过程.

物理学界经常发现这样的现象, 两个或多个动力学系统, 除了自身的演化外, 其间还有相互耦合作用, 这种作用既可以是单向的, 也可以是双向的. 当满足一定条件时, 在耦合的影响下, 这些系统的状态输出就会逐渐趋同, 进而完全相等, 这个过程称为同步. Kuramoto 模型作为建模同步行为最简单的可解模型得到广泛地应用 [4,13,14], 它不但将统计手段引入了动力学领域, 而且将同步动力学行为与统计物理中的非平衡相变联系在一起 [15,16]. 表 1 给出了本文以下使用的符号及其说明.

首先, 假定每个数据对象只与以它为中心的一个超球形区域内的有限个数据对象之间存在着同步耦合作用, 从而将对象受到的同步作用局部在一个特定邻域半径内. 为了能采用 Kuramoto 模型进行数据聚类, 将每个矢量数据对象 x 看成一个振子, 而每一维坐标值可以看作是对应振子的一个相位值. 那么, 在局部同步耦合作用下数据对象的每一维坐标分别进行同步, 受同步耦合相作用, 对象的坐标

表 1 符号表

Table 1 Table of adopted symbols

Symbol	Definition	Symbol	Definition
KM	Kuramoto model	d	The dimensionality of the data set
X	The data set	x	A data object in X
x_i	The ith dimension of the data object x	$x^{(i)}$	The ith data object in X
ε	Radius of neighborhood, $\varepsilon \in \mathbb{R}^+$	$N_\varepsilon(x)$	ε-neighborhood of the object x
$x(t)$	The renewal value of object x at time step t	k	The number of clusters in the data set
n	The number of objects in the data set	S	The coupling strength of the KM
C_i	The ith cluster of the data set	$\lvert C_i\rvert$	The number of object in the ith cluster

位置将不断变化, 最后将坐标位置趋同的一组数据对象看作是一个聚类.

定义 1 (ε– 邻域) 对象 x 的 ε– 邻域是到对象 x 的距离小于等于 ε 的所有对象组成的集合 $N_\varepsilon(x)$:

$$N_\varepsilon(x) = \{y \in X \,|\mathrm{dist}(x, y) \leqslant \varepsilon\}, \tag{1}$$

其中, $\mathrm{dist}(x, y)$ 是距离度量函数, 本文以下均采用 Euclidian 距离. 若对象 $y \in N_\varepsilon(x)$, 则称 y 是 x 的 ε– 邻居或从对象 x 到 y 直接 ε 可达, 记为 $x \mapsto_\varepsilon y$. 对象间的 ε– 邻居关系是对称的, 即若 $x \mapsto_\varepsilon y$, 则有 $y \mapsto_\varepsilon x$.

定义 2 (广义 Kuramoto 模型) 设 $x \in \mathbb{R}^d$ 是 d 维数据集 X 中的一个数据对象, x_i 为数据对象的第 i 维坐标. 将每个对象的每一维坐标都作为一个独立的相位振子, 并在其 ε– 邻域内同步. 因此, 对象 x 的每一维坐标值 x_i 的动态变化过程如下:

$$\frac{dx_i}{dt} = \omega_i + \frac{S}{|N_\varepsilon(x)|} \sum_{y \in N_\varepsilon(x)} \sin(y_i - x_i). \tag{2}$$

令 $dt = \Delta t$, 那么有

$$x_i(t+1) = x_i(t) + \Delta t \omega_i + \frac{\Delta t \cdot S}{|N_\varepsilon(x(t))|} \sum_{y \in N_\varepsilon(x(t))} \sin(y_i(t) - x_i(t)). \tag{3}$$

所有的对象都有一个独立的频率 ω, 由于对象之间存在的频率差异会干扰甚至阻止聚类的形成, 但 ω 对矢量数据的聚类的结果没有影响, 所以 $\Delta t \cdot \omega_i$ 这一项可以忽略. $\Delta t \cdot S$ 是一个常数, 为了简便, 将它设置为 1. 最后, 对象 x 的每一维坐标 x_i 随着时间的动态变化公式为

$$x_i(t+1) = x_i(t) + \frac{1}{|N_\varepsilon(x(t))|} \sum_{y \in N_\varepsilon(x(\mathrm{t}))} \sin(y_i(t) - x_i(t)). \tag{4}$$

对象 x 在时间段 $t = 0$ 时有 $x(0) = (x_1(0), \ldots, x_d(0))$, 表示了对象的初始状态. $x_i(t+1)$ 描述了随着时间 $t = (0, \ldots, T)$ 的演变, 对象 x 更新后的第 i 维坐标值.

为了描述振子之间的同步水平, 即衡量局部对象群体的一致性, 定义一个聚类序参量 r_c:

$$r_c = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{\left|N_\varepsilon(x^{(i)})\right|} \sum_{y \in N_\varepsilon(x^{(i)})} e^{-\|y - x^{(i)}\|} \right). \tag{5}$$

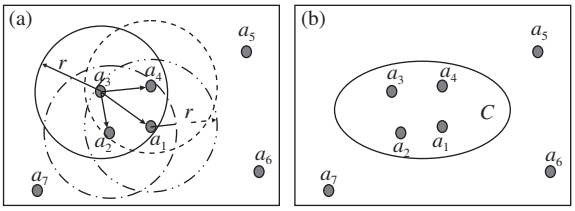

图 1　ε–邻域闭包 (a) 与 ε–同步聚类 (b)

Figure 1　ε-neighborhood closure (a) and synchronized cluster (b)

4　基于改进同步过程的层次聚类

在同步聚类过程中, 最初每个矢量数据对象具有各自的起始坐标值; 随着时间的推移, 在局部同步耦合作用下, 越来越多的对象趋于同步, 此时 r_c 的值也越来越大. 在 Sync 算法中, 当 r_c 达到某个接近 1 的预设阈值时, 动态聚类将结束, 此时认为到达完全同步, 同一个聚类中的对象都运动到几乎同一坐标点上. 但是, 到达完全同步需要一个长时间的动态同步过程, 这会使得聚类的时间效率低下. 通过观察发现, 对象同步到一定阶段就可以确定对象的局部同步结果, 从而更高效地探测出聚类.

4.1　对象的局部同步收敛准则

定义 3 (ε– 邻域闭包)　设数据对象集 $X' \subseteq X$, 在同步聚类的动态过程中, 如果 $\forall x, y \in X'$ 均满足 $x \mapsto_\varepsilon y$, 且 $\forall z \in X$ 若 $x \mapsto_\varepsilon z$, 则 $z \in X'$, 那么称 X' 为一个 ε– 邻域闭包, 即对于任一对象 $x \in X'$ 均有 $N_\varepsilon(x) = X'$.

图 1 采用二维数据作聚类对象描述了 ε– 邻域闭包的含义. 如图 1(a), 以点 a_1 为圆心的 ε– 邻域内有点 a_1, a_2, a_3, a_4, $N_\varepsilon(a_1) = \{a_1, a_2, a_3, a_4\}$, 同样 $N_\varepsilon(a_2) = N_\varepsilon(a_3) = N_\varepsilon(a_4) = \{a_1, a_2, a_3, a_4\}$, 满足定义 3, 所以 $\{a_1, a_2, a_3, a_4\}$ 是 ε– 邻域闭包. 此时, 在局部同步过程中 a_1, a_2, a_3, a_4 之间相互影响, 而无其他外力作用. 图 1(b) 表明 $\{a_1, a_2, a_3, a_4\}$ 为 ε– 邻域闭包, 所以 $\{a_1, a_2, a_3, a_4\}$ 将形成一个聚类 C, a_5, a_6, a_7 为离群点.

在 SHC 算法中用 ε– 邻域闭包替代了 Sync 算法中的参数 r_c, 无需对象到达完全同步, 减少了对象之间的动态交互时间.

定理 1　设数据对象集 $X' \subseteq X$, 如果 X' 是一个 ε– 邻域闭包, 那么 X' 中的所有对象在局部动态同步过程中将最终达到完全同步.

证明　根据式 (4), 每个对象都只受到其 ε– 邻域内邻居对象的影响. 当 $N_\varepsilon(x)$ 是 ε– 邻域闭包时, 集合内的对象互为 ε– 邻居, 且对象间的 ε– 邻居关系满足传递闭包性质. 所以任意一个 X' 中的对象只受到集合 X' 内其他对象的作用力, 而无任何来自集合 X' 外的作用力, 因此任意对象始终在 X' 位置区域内部运动; 而距离 X' 中心较远的边缘对象在集合 X' 内其他对象在同步影响下, 朝着 ε– 邻域的中心运动, 使得 X' 集合的半径不断缩小, 最终将到达完全同步.

由定理 1 可知, 如果 $N_\varepsilon(x)$ 是一个 ε– 邻域闭包, 则 $N_\varepsilon(x)$ 中的所有对象已经可以看成一个聚类, 使得在闭包中所有对象到达同步之前确定一个聚类.

定理 2　给定参数 ε, 在局部同步作用下数据对象集 X 中对象必将形成若干个 ε– 邻域闭包.

证明　对于数据集合 X, 给定一个 ε– 邻域, 任意对象 x_i 的 ε– 邻域 $N_\varepsilon(x_i)$ 均为集合 X 的子集. 如果存在数据 x_j, $x_j \notin N_\varepsilon(x_i)$, 且 $N_\varepsilon(x_i) \cap N_\varepsilon(x_j) \neq \emptyset$, 那么称数据 $y \in N_\varepsilon(x_i) \cap N_\varepsilon(x_j)$ 为不稳定数

据, 而称集合 $\{y|y \in N_\varepsilon(x_i) \cap N_\varepsilon(x_j), x_j \notin x_i\}$ 为不稳定数据集. 不稳定数据 y 与集合 X 中的稳定数据不同, 它不仅受到 $N_\varepsilon(y)$ 集合中数据的影响, 还受到一个或多个 $N_\varepsilon(x_i)$ 中的对象对它的作用力, $y \in N_\varepsilon(x_i), y \neq x_i$. y 有两种运动情况: 1) 分离: 朝着某一个 $N_\varepsilon(x_i)$ 运动, 从而脱离其他 $N_\varepsilon(x)$ 对其的作用力. 2) 合并: 多个 $N_\varepsilon(x)$ 中的数据对象受到 y 的影响, 逐渐向 y 所在的邻域靠拢, 最后合并成一个 $N_\varepsilon(x)$. 数据集 X 的不稳定数据会一直进行以上两种运动, 直至稳定. 所以任意数据集 X 在同步过程中, 一定会形成 ε– 邻域闭包.

4.2 层次同步聚类算法 SHC

给定含有 n 个对象的 d 维空间数据集 X, 对于邻域半径参数 ε 的某一取值, 一趟同步聚类过程 SYN 的详细步骤如下:

1) 初始时 ($t = 0$), 数据集 X 中的所有对象都设为活动对象, 它们具有各自独立的坐标位置, 形成均由单个孤立点构成的 n 个互不关联的聚类;

2) t 的值加 1 , 根据式 (4) 计算每个活动对象与其 ε– 邻域内的对象相互作用后的新坐标位置;

3) 若一个对象 x 的 ε– 邻域 $N_\varepsilon(x)$ 形成 ε– 邻域闭包, 则识别出包含对象 x 的聚类, 此时将 $N_\varepsilon(x)$ 中的所有对象设为静止对象, 停止同步;

4) 重复步骤 2) 直到所有对象停止同步.

以下, 在一趟同步聚类过程 SYN 的基础上, 提出一种不断增大 ε 值的自动同步层次聚类算法 SHC:

1) 设定参数 ε 的初始值;

2) 调用 SYN 过程, 完成一趟同步聚类;

3) 若所有点都归为一个聚类, 则终止; 否则, 增大参数 ε 的值, 重复步骤 2).

Algorithm 1: SHC( )

Input: 矢量数据集 $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$, 聚类有效性评价函数 Q

Output: 最佳聚类结果 C

```
t ← 0; ε ← ε0;
C(0) = {{x(1)}, {x(2)}, ..., {x(n)}};
while |C(n)| > 1 do
    while true do
        for each x ∈ X^t do
            根据式 (4) 更新数据 x;
        if each x ∈ X^t Closure(Nε(x)) then
            C(t) = DetectClusters(X^t, ε);
            break;
    ε ← ε + Δε;
    t ← t + 1;
C = agr max_{C(i)} Q(X,C(i));
return C;
```

Algorithm 1 给出了同步层次聚类算法 SHC 的详细描述. 如果需要从数据中识别出一个最优聚类结果, 可以采用一个聚类有效性评价函数 Q, 对同步过程中所产生的所有 ε 聚类结果的有效性进行度

表 2 实验所采用的人工和真实数据集

Table 2 Synthetic and real-world data sets adopted in the experiments

Synthetic data sets				Real-world data sets			
Data set	No. of objects	No. of dim.	No. of clusters/ noise	Data set	No. of objects	No. of dim.	No. of clusters/noise
DS1	1033	2	5/0	BreastTissue	106	9	6/0
DS2	1252	2	2/0	Iris	150	4	3/0
DS3	615	2	4/15	Wine	178	13	3/0
DS4	1209	2	7/25	Ecoli	336	7	8/0
DS5	1000	15	5/0	Wisconsin	683	9	2/0
DS6	1000	30	5/0	Shuttle	58000	9	7/0

量, 并选择值最优者作为最终结果.

本文中, 初始 ε 值为样本集合中所有对象的 3– 最近邻距离的平均值, ε 的增量值 $\Delta\varepsilon$ 则采用 4 – 最近邻距离的平均值与 3– 最近邻距离的平均值的差值.

5 实验评价

本节在多个人工合成和真实数据集上对所提出的算法 SHC 的性能进行实验评价. 将 SHC 与 Sync[6], X-Means[7], DBSCAN[8], gSkeletonClu[10], Chameleon[11], Meanshift[9], AP[12] 等经典聚类算法进行了对比分析. SHC 算法则采用 ANSI C++ 编写, Sync, X-Means, gSkeletonClu, Chameleon, Meanshift, AP 算法采用作者提供的源代码, 其他聚类算法则采用 Weka[1)] 软件包中的 Java 实现. 所有实验均在配置 Intel Centrino 2.0 GHz CPU 和 2 GB RAM, 安装 Windows XP 操作系统的 PC 机上完成.

5.1 实验数据集

表 2 给出了实验所采用的人工和真实数据集的详细描述. 使用工具软件生成了 6 个合成数据集 DS1∼ DS6, 用于分析算法发现任意形状、大小聚类, 以及抑制噪声数据的能力. 此外, 实验中还使用了 6 个 UCI 真实数据集: BreastTissue, Iris, Wine, Ecoli, Wisconsin 和 Shuttle [2)].

5.2 聚类有效性评价函数

目前, 已提出很多有效的聚类有效性评价准则 [17∼21], 这里将同步聚类过程与 Silhouette 准则函数 (silhouette width criterion, SWC) 相结合 [18], 寻找最佳聚类结果.

给定数据集 X, 假设第 j 个对象 $x^{(j)}$ 属于聚类 $p \in \{1, \ldots, k\}$. 对象 $x^{(j)}$ 与集群 p 中所有对象的距离平均值定义为 $a_{p,j}$. $x^{(j)}$ 与另一个聚类 $q(q \neq p)$ 中所有对象的距离的平均值定义为 $d_{q,j}$. $b_{p,j} = \min\{d_{q,j}\}$, $q = 1, \ldots, k$, $q \neq p$. $b_{p,j}$ 表示 $x^{(j)}$ 与其最近的周边聚类的平均相异度. 因此, 单个对象 $x^{(j)}$ 的 Silhouette 值定义为

$$s_x(j) = \frac{b_{p,j} - a_{p,j}}{\max\{a_{p,j}, b_{p,j}\}}. \tag{6}$$

1) http://www.cs.waikato.ac.nz/ml/weka/.

2) http://archive.ics.uci.edu/ml/index.html.

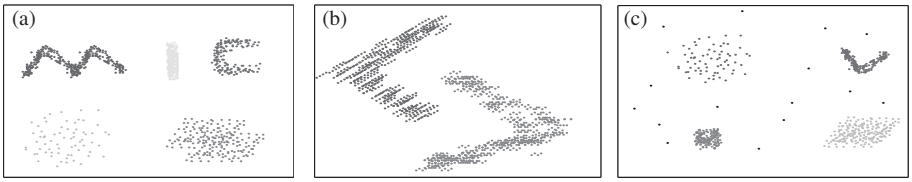

图 2 包含不同大小、形状、密度聚类以及噪声的 3 个人工数据集

Figure 2 Three synthetic data sets consisting of clusters with various size, shape and density as well as noises. (a) DS1; (b) DS2; (c) DS3

显然, 如果 $s_x(j)$ 越大, 则表示将 $x^{(j)}$ 分配给 p 的正确性越高. 如果 p 仅是由一个对象 $x^{(j)}$ 构成的聚类, 则 $x^{(j)}$ 表示了一个孤立点, 我们定义 $s_x(j) = 0$. SWC 为所有对象的 $s_x(j)$ 的平均值, $j = 1, 2, \ldots, n$, 即

$$\mathrm{SWC} = \frac{1}{n}\sum_{j=1}^{n} s_x(j). \tag{7}$$

将 SWC 最大者作为最终的聚类结果, 这意味着最小化聚类内部的距离 $a_{p,j}$, 最大化聚类之间的距离 $b_{p,j}$.

5.3 聚类结果准确性的度量标准

本文采用 NMI(normalized mutual information), AMI(adjusted mutual information) 和 ARI(adjusted rand index) 这 3 个指标对聚类结果准确性进行评价 [22]. NMI, AMI 和 ARI 的取值范围为 $[0, 1]$, 值越高表示聚类效果越好.

5.4 算法聚类性能分析

5.4.1 人工数据集上的实验分析

首先在二维人工矢量数据 DS1∼DS3 上进行测试. DS1 包含 5 个不同大小和密度的聚类, 如图 2(a); DS2 包含 2 个不同形状的聚类, 如图 2(b); DS3 包含 4 个不同大小、形状和密度的聚类以及 15 个离群点. SHC 算法完全准确地检测出这些聚类和离群点. 即, 基于动态同步过程的 SHC 算法不依赖于数据的任何分布假设, 可以准确地检测出任意数量、形状和大小的自然聚类, 有较强的噪声数据抑制能力.

接下来, 在包含 7 个任意大小、形状、密度的聚类和 25 个离群点的人工合成数据集 DS4 上与多个经典聚类算法的进行性能比较. Sync 算法 [6] 在该数据集上的聚类结果如图 3(a) 所示, 有 3 个聚类未被正确识别, 其中一个聚类中的一小部分数据被错分到另两个聚类中. 对于 DBCSAN 算法 [8], 对参数 MinPts 和 Eps 的取值进行多次尝试, 选择一个较好的聚类结果, 如图 3(b). 因为数据集包括不同密度的聚类, DBCSAN 算法仅发现 5 个聚类. 图 3(c) 为 gSkeletonClu 算法 [10] 的聚类结果, 该算法基于密度聚类且能自动发现优化的 Eps 参数, 识别出 3 个聚类. 由此可以看出, 传统基于密度的聚类算法能识别出不同形状和大小的数据, 对于噪声数据也有一定的抑制能力. 但是, 它们识别变密度聚类的能力较差, 且参数的设置也是一项相当困难的任务. 图 3(d) 给出了 X-Means 算法 [7] 的聚类结果, X-Means 算法也无需任何输入参数, 但该算法难以处理非球形的聚类, 且无法识别离群点, 所有离群点均被划分到邻近的聚类中, 因而聚类准确性较差. 如图 3(e) 为 AP 算法 [12] 的聚类结果, 该算法

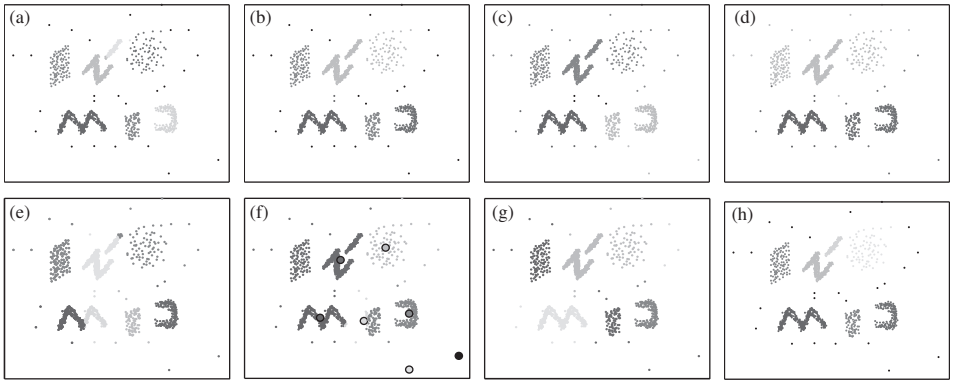

图 3　不同算法在数据集 DS4 上的聚类结果

Figure 3 Clustering results of various clustering algorithms on data set DS4. (a) Sync; (b) DBSCAN; (c) gSkeletonClu; (d) X-Means; (e) AP; (f) MeanShift; (g) Chameleon; (h) SHC

表 3　人工数据集上多个基准聚类算法的准确率比较

Table 3 Accuracy comparison on synthetic data sets with various baseline clustering algorithms

Data sets	SHC			Sync			DBSCAN			gSkeletonClu		
	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI
DS1	**1**	**1**	**1**	**1**	**1**	**1**	0.996 (Eps=1.3)	0.990	0.981	0.716	0.812	0.691
DS2	**1**	**1**	**1**	**1**	**1**	**1**	**1 (Eps=0.6)**	**1**	**1**	**1**	**1**	**1**
DS3	**1**	**1**	**1**	**1**	**1**	**1**	0.997 (Eps=3.1)	0.994	0.993	0.811	0.825	0.739
DS4	**0.999**	**0.998**	**0.999**	0.993	0.988	0.988	0.768(Eps=2.5)	0.884	0.794	0.573	0.716	0.560
DS5	0.752	0.847	0.801	0.699	0.781	0.723	0.001(Eps=10.7)	0.030	0.017	0.743	0.482	0.567
DS6	0.753	0.849	0.802	0.712	0.783	0.730	0.001(Eps=10.7)	0.033	0.017	0.728	0.457	0.551
Data sets	X-Means			AP			MeanShift			Chameleon		
	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI
DS1	0.814	0.902	0.822	0.919	0.940	0.938	0.928	0.935	0.926	0.809	0.889	0.821
DS2	0.700	0.668	0.588	0.625	0531	0.474	0.643	0.549	0.496	0.401	0.458	0.380
DS3	0.968	0.946	0.919	0.968	0.947	0.919	0.974	0.950	0.929	0.968	0.947	0.919
DS4	0.572	0.707	0.554	0.725	0.842	0.835	0.664	0.790	0.677	0.754	0.853	0.751
DS5	**0.906**	**0.782**	**0.816**	0.448	0.700	0.539	0.637	0.700	0.706	0.571	0.612	0.596
DS6	**0.904**	**0.782**	**0.810**	0.437	0.601	0.479	0.604	0.566	0.578	0.550	0.598	0.501

需要多次参数尝试方能得到相应聚类个数的聚类结果, 且存在聚类被分裂和错误合并的情况. 图 3(f) 为 MeanShift 算法 [9] 在设定一精度阈值后, 多次执行算法后得到较为满意的一个, 算法虽然识别出 7 个聚类, 但与正确结果相差较大, 且该算法难以处理噪声数据. 图 3(g) 为 Chameleon 算法 [11] 在人工输入聚类个数为 7 时的聚类结果, 该算法能够识别任意形状的聚类, 但是它对噪声数据较敏感, 因而将右下角的一个离群点单独作为一个聚类. 本文提出的算法 SHC 同样基于同步原理, 它完全正确的检测出所有的聚类, 如图 3(h), 正确识别出 24 个离群点, 仅有一个离群点被错分到邻近的聚类中.

表 3 给出了聚类结果的 AMI, NMI 和 ARI 得分. 通过综合比较结果可以看出, SHC 算法在不同大小、形状、密度或大量离群点的数据集上都显示了较高的聚类准确率, 性能均优于以上基准方法.

表 4 真实数据集上多个基准聚类算法的聚类准确率比较

Table 4 Accuracy comparison on real-world data sets with various baseline clustering algorithms

Data sets	SHC			Sync			DBSCAN			gSkeletonClu		
	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI
BreastTissue	**0.378**	**0.442**	**0.295**	0.208	0.430	0.276	0.151($\varepsilon = 0.34$)	0.361	0.242	0.160	0.401	0.253
Iris	**0.663**	**0.748**	**0.733**	0.568	0.734	0.577	0.568 ($\varepsilon = 0.67$)	0.734	0.577	0.568	0.734	0.577
Wine	**0.755**	**0.776**	**0.695**	0.426	0.432	0.375	0.315 ($\varepsilon = 0.46$)	0.405	0.383	0.271	0.368	0.279
Ecoli	**0.677**	**0.642**	**0.539**	0.456	0.547	0.408	0.038($\varepsilon = 0.56$)	0.118	0.006	0.067	0.221	0.135
Wisconsin	**0.869**	**0.782**	**0.777**	**0.869**	**0.782**	**0.777**	0.012 ($\varepsilon = 7.0$)	0.011	0.005	0.012	0.011	0.005
Shuttle	**0.570**	**0.600**	**0.614**	0.211	0.419	0.403	–	–	–	0.432	0.486	0.454
Data sets	X-Means			AP			MeanShift			Chameleon		
	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI	ARI	NMI	AMI
BreastTissue	0.217	0.435	0.282	0.098	0.294	0.184	0.052	0.244	0.126	0.022	0.216	0.093
Iris	0.568	0.734	0.577	0.558	0.699	0.551	0.568	0.734	0.577	0.642	0.722	0.696
Wine	0.257	0.248	0.203	0.377	0.443	0.391	0.293	0.405	0.322	0.276	0.389	0.301
Ecoli	0.422	0.512	0.371	0.437	0.611	0.526	0.440	0.546	0.427	0.351	0.514	0.446
Wisconsin	0.497	0.561	0.446	0.841	0.741	0.735	0.809	0.700	0.697	0.013	0.022	0.011
Shuttle	0.300	0.334	0.315	–	–	–	0.005	0.016	0.040	–	–	–

此外, 还采用两个分别含有 15 和 30 个属性的人工合成数据集 DS5∼DS6 测试了不同聚类算法处理高维数据的能力, 聚类结果如表 3. 本文提出的 SHC 算法在这两个数据集上的得分略低于 X-Means, 但是依然保持了较高的聚类准确率, 这与算法采用维数独立的聚类原理相关.

5.4.2 真实数据集上的实验分析

我们在 6 个 UCI 真实数据集上对 SHC 算法的聚类性能与其他几种基准聚类算法进行比较, 聚类结果的 NMI 等值如表 4. 选取聚类结构较为明显的 Iris, Wine 和 Wisconsin 进行聚类性能分析.

Iris 数据集包含 150 个对象, 被平均分为 3 类: Setosa, Versicolor 和 Virginica, 每个对象有 4 个数值属性. SHC 算法在该数据集上检测出 3 个聚类, 第 1 个聚类包含 Setosa 类的全部 50 个对象, 第 2 个聚类中有 71 个对象 {Versicolor:50, Virginica:21}, 另一个聚类中包含 Virginica 类的 29 个对象. Chameleon 算法检测出 3 个聚类 {{Setosa:50}, {Versicolor:23, Virginica:49}, {Versicolor:27, Virginica:1}}, 但与 SHC 的聚类结果相比稍差. Sync, X-Means, DBSCAN, gSkeletonClu, AP 和 MeanShift 算法在该数据集上均只检测出 2 个聚类, 是因为 Iris 数据集中有两个聚类明显连接在一起呈线性不可分状态.

Wine 数据集包含 178 个对象, 每个对象有 13 个属性, 所有对象被分为 3 类: A:59, B:71 和 C:48. SHC 算法正确识别出 3 个聚类, 结果为: {{A:59, B:55}, {B:5, C:46}, {B:11, C:2}}, 其中有些对象被错分. Sync 算法识别出 4 个聚类 {{A:57}, {B:49}, {C:44}, {A:2, B:21, C:5}}, 聚类结果较 SHC 算法差. 其他聚类算法在该数据集上表现不佳, 检测出多个小聚类, 因此 NMI, AMI 和 ARI 的得分要比 SHC 和 Sync 明显低.

Wisconsin 数据集包含 683 个对象, 每个对象有 9 个数值属性, 数据被分为恶性肿瘤 (M: 239 例) 和良性肿瘤 (B: 444 例) 两类 (16 个遗漏值的实例已经从原始数据集中删除). 基于同步原理的 SHC

表 5　SHC 算法与 Sync 算法同步次数和聚类时间比较

Table 5　Comparison of synchronization steps and time consuming between algorithms SHC and Sync

Algorithm	Sync		SHC	
The No. of objects	Steps	Time(s)	Steps	Time(s)
100	194	0.906	22	0.437
1,000	256	110.862	84	65.441
10,000	196	10173.433	61	6019.372

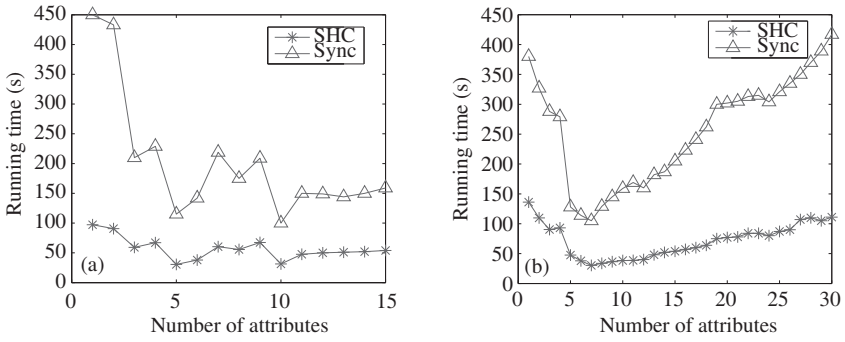

图 4　不同维数数据的同步时间比较

Figure 4　Comparison of running time on two synthetic data set with various attribute numbers. (a) DSS; (b) DS6

算法和 Sync 算法都准确检测出两个聚类 {{M:6, B:427}, {M:233, B:17}}, 第一个聚类包含 433 个对象, 第二个聚类包含另外 250 个对象, 其中有 23 个对象被错分. AP, Meanshift 和 Chameleon 算法也检测出二个聚类, 聚类结果结果分别为 {{M:19, B:435}, {M:220, B:9}}, {{M:14, B:424}, {M:225, B:20}}和 {{M:5}, {M:234, B:444}}, 聚类质量均较前两个基于同步原理的算法差. 而 X-Means 算法在该数据集上聚类效果不佳, 从该数据集中检测出 3 个聚类: {M:23, B:174},{B:261},{M:216, B:9}}. DBSCAN 和 gSkeletonClu 均识别出 1 个包含 682 对象的大聚类以及 1 个噪声数据点, 可以看出基于密度的聚类这个高维数据上的整体聚类效果不佳.

5.5　算法的运行时间分析

对于一维数据, SHC 算法一次动态同步聚类的时间复杂度为 $O(T \cdot n^2)$, 其中 n 是数据对象的个数, T 是形成 ε– 邻域闭包时的同步次数. 如果采用有效的数据索引, 算法的时间复杂度可以降为 $O(T \cdot n \log n)$. 本文提出的 SHC 算法将同步聚类模型与 SWC 评价准则相结合用于选择最优聚类结果, 所以总体时间复杂度为 $O(L \cdot T \cdot n \log n)$, 其中 L 指参数 ε 的取值个数.

SHC 算法和 Sync 算法都基于同步聚类思想, 保证了同步聚类的理想特性. 为比较 SHC 和 Sync 聚类算法的时间效率, 生成数据规模从 100 到 10000 不等的 3 个数据集. 表 5 给出了这两个算法在不同数据集上所需的同步次数和总体聚类时间, 其中 Sync 算法中同步停止参数 r_c 取 0.99. 从表 5 中可以看出, 对于相同规模的数据集 SHC 聚类算法的同步次数明显减少, 其总体运行时间大约仅为 Sync 算法的一半.

为了分析数据维数对算法运行时间的影响, 在数据集 DS5 和 DS6 上分别随机选取 1,2,3, ... 等不同维数, 采用 SHC 和 Sync 算法对数据进行聚类, 图 4 是两种算法相对不同维数数据的运行时间曲

线. SHC 算法和 Sync 算法同步一次都需对每一维数据进行更新, 由于同步的次数还受到每维数据的分布情况影响, 所以从图 4 中可以看到: 运行时间与维数之间的关系不是简单的线性关系. 虽然 SHC 与 Sync 相对于数据维数的运行时间曲线形状接近, 但依然可以清晰的看出, SHC 的总体运行时间均不到 Sync 的 1/2.

6 总结

本文提出了一种基于同步原理的层次聚类算法 SHC. 在 Kuramoto 模型基础上, 提出基于 ε– 邻域的局部同步聚类方法. 此外, 提出 ε– 邻域闭包的概念, 优化了聚类停止条件并可实现层次化的聚类. 实验结果显示, 本文提出的 SHC 算法对噪声不敏感, 对于存在任意大小、形状和密度聚类的中、小规模多维空间矢量数据集能够产生高质量和稳定的聚类结果. 由于能够在数据对象到达局部同步前预测出聚类, 显著减少了数据同步所需要的次数和时间. 算法运行时间与数据维数的关系, 如何进一步提高同步聚类的时间效率, 以及如何将同步聚类方法应用到不同的数据挖掘应用中均是有意义的研究课题.

参考文献

1 Pikovsky A, Rosenblum M, Kurths J. Synchronization, a universal concept in nonlinear sciences. Cambridge: Cambridge University Press, 2001. 1–23
2 Boccalettia S, Kurths J, Osipov G, et al. The Synchronization of chaotic systems. Phys Rep, 2002, 366: 1–101
3 Kuramoto Y. Self-entrainment of a population of coupled non-linear oscillators. In: International Symposium on Mathematical Problems in Theoretical Physics, Tyoto, 1975. 420–422
4 Acebron J A, Bonilla L L, Vicente C J P, et a1. The Kuramoto Model: A simple paradigm for synchronization phenomena. Rev Mod Phys, 2005, 77: 137–185
5 Kuramoto Y. Chemical Oscillations, Waves, and Turbulence. Berlin: Springer-Verlag, 1984. 5–21
6 Böhm C, Plant C, Shao J M, et al. Clustering by synchronization. In: Proceedings of ACM SIGKDD'10, Washington, 2010. 583–592
7 Pelleg D, Moore A. X-means: Extending K-means with efficient estimation of the number of clusters. In: Proceedings of ICML'00, Stanford, 2000. 727–734
8 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
9 Comaniciu D, Meer P. Mean Shift: A robust approach toward feature space analysis. IEEE Trans Patt Analy Mach Intell, 2002, 24: 603–619
10 Sun H L, Huang J B, Han J, et al. GSkeletonClu: Density-based network clustering via structure-connected tree division or agglomeration. In: Proceedings of IEEE ICDM'10, Sydney, 2010. 481–490
11 George K, Eui-Hong H, Vipin K. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Comput, 1999, 32: 68–75
12 Frey B J, Dueck D. Clustering by Passing messages between data points. Science, 2007, 315: 972–976
13 Arenas A, Diaz-Guilera A, Perez-Vicente C J. Synchronization reveals topological scales in complex networks. Phys Rev Lett, 2006, 96: 114102
14 Kim C S, Bae C S, Tcha H J. A phase synchronization clustering algorithm for identifying interesting groups of genes from cell cycle expression data. BMC Bioinf, 2008, 9: 1471–2105
15 Pecora L M, Carroll T L. Synchronization in chaotic systems. Phys Rev Lett, 1990, 64: 821
16 Zheng Z G, Hu G, Hu B. Phase slips and phase synchronization of coupled oscillators. Phys Rev Lett, 1998, 81: 5318
17 Liu Y C, Li Z M, Xiong H, et al. Understanding of internal clustering validation measures. In: Proceedings of IEEE ICDM'10, Sydney, 2010. 911–916

18 Vendramin L, Campello R J G B, Hruschka E R. On the comparison of relative clustering validity criteria. In: Proceedings of SIAM SDM'09, Sparks, 2009. 733–744
19 Vendramin L, Campello R J G B, Hruschka E R. A robust methodology for comparing performances of clustering validity criteria. In: Proceedings of SBIA'08, Salvador, 2008. 237–247
20 Vinh N X, Epps J, Bailey J. Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res, 2010, 11: 2837–2854
21 Gräunwald P. A tutorial introduction to the minimum description length principle. In: Grunwald P, Myung I J, Pitt M, eds. Advances in Minimum Description Length: Theory and Applications. Cambridge: MIT Press, 2005. 1–80
22 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

A hierarchical clustering method based on a dynamic synchronization model

HUANG JianBin[1*], KANG JianMei[1], QI JunJie[1] & SUN HeLi[2]

1 *School of Software, Xidian University, Xi'an* 710071*, China;*
2 *Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an* 710049*, China*
*E-mail: jbhuang@xidian.edu.cn

Abstract Clustering is an essential method for analyzing and mining the intrinsic group in data. This paper presents a novel synchronization-based hierarchical clustering method based on an extended Kuramoto dynamic synchronization model. Each data object is regarded as a phase oscillator and interacts dynamically with its neighboring objects. As time evolves, objects synchronize naturally. With regard to the local diameter of the neighborhood, the proposed method finds local synchronization-based natural clusters. Hierarchical clustering results are achieved by enlarging the local neighborhood distance of objects synchronizing continuously. Using a neighborhood closure, our method predicts clusters before the objects reach local synchronization, thereby significantly reducing the dynamic interaction time. To select the optimal clusters automatically, this hierarchical clustering method based on a dynamic synchronization model is combined with a clustering validation measure known as the silhouette width criterion. Combined with the silhouette width criterion, the proposed method is parameter-free. Moreover, the proposed method can detect clusters in data of arbitrary shapes, sizes and numbers without any data distribution assumptions. This synchronization-based clustering also allows natural outlier identification, since outliers do not synchronize with data objects in clusters. Extensive experiments on several synthetic and real-world data sets demonstrate that the proposed method achieves high clustering accuracy with lower execution time and fewer synchronization steps compared to the state-of-the-art method.

Keywords hierarchical clustering, dynamic synchronization model, neighborhood closure, outlier detection, parameter-free method, silhouette width criterion

HUANG JianBin was born in 1975. He received his Ph.D. in Pattern Recognition and Intelligent Systems from Xidian University, Xi'an, in 2007. Currently, he is an Associate Professor at the School of Software at Xidian University. His research interests include knowledge discovery and data mining, information network analysis and big data management. Dr. Huang is a senior member of CCF and member of IEEE and ACM.

KANG JianMei was born in 1989. She received her bachelor's degree in Computer Science and Technology from Xidian University, Xi'an, in 2011. Currently, she is a master's student at Xidian University. Her research interests include data mining and machine learning.